A Fast Corner Detection Algorithm Based on Area Deviation

Zhou Hao, Shao Lejun

School of Electrical and Electronics Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 2263

Abstract

A new algorithm, based on area-deviation, is proposed for the detection of corner points of digitized curves. The algorithm consists of two steps. In the first step, a fixed-length chord is moving along the digitized curve step by step, and the area between the chord and the associated curve segment is measured. The area values are used to represent the average curvature values for the corresponding curve segments. The curve segments with their area values having reached a local maximum and exceeded a threshold value will be identified. Each such curve segment will contain one corner point. In the second step, the exact position of the corner point in each identified curve segment is found by comparing the changes in the curve's directions at each point of the curve segment. A lot of graphical objects including Chinese and English fonts have been tested. The algorithm is proved to be efficient and gives results close to human expectations.

1. Introduction

Corners are very useful features in image matching, shape analysis, and curve fitting. They serve to simplify the analysis of images by dramatically reducing the amount of data to be processed, while at the same time preserving important information about the object. However, corner detection from an image is never a trivial problem since it must distinguish between angles that are due to the discreteness of the digital curve and the angles that represent significant changes in the direction of curves.

Corner detection algorithms can be classified as either using the information contained in the local gray-level values, or involving the analysis of boundary features of the object. Many gray-level based corner detectors have been reported in the literature [4,5,7,9]. In boundary-analysis based approach, only binary images are used. The image is first segmented and the boundary of the object is traced. If, at a point, boundary makes discontinuous changes in direction, or the curvature of the boundary is above some threshold, that point is declared as a corner point.

In [6]'s paper, six boundary-analysis based corner detection algorithms were discussed. In that paper, the author used a set of test images to judge the goodness of those algorithms and found that the Beus-Tiu's algorithm gave better results in all the examples. We have tried to use both Freeman-Davis and Beus-Tiu's algorithms to test their effectiveness with scanned Chinese character images. It was found that both algorithms are not suitable for detecting corners of rough, uneven boundaries. A detailed discussion is given in [2] and [8]'s papers.

In this paper, a new fast and robust corner detection algorithm is presented, In our algorithm, only binary images are used. The algorithm uses areas between a fixed-length chord and a series of curve segments along the ditigized curve to measure the curvatures for the corresponding curve segments and identifies corner points based on their area-deviation patents. The algorithm consists three steps: (1). boundary extraction; (2). corner detection; and (3). corner localization. In the following, the detailed algorithm is described and their performance is presented.

2. Corner Detection Algorithm

Our algorithm is based on the computation of curve's curvature values. Suppose a curve is expressed as $y = f(x)$. The curvature k_i at a point (x_i, y_i) on the curve is defined as

$$k_i = \frac{y_i''}{[1 + (y_i')^2]^{\frac{2}{3}}} \qquad (1)$$

The corner points are usually related to the points having maximum or minimum curvature values or points having a sharp change in their curvature values.

For a digitized curve, however, this formula cannot be used. In our algorithm, the area values between a fixed-length chord and a series of curve segment along the digitized curve are calculated and are used to represent the average curvature values for the corresponding curve segments. The corner points are then identified based on their area values. Our algorithm consists of the following three steps:

2.1 Boundary extraction

An *erosion* operator is applied to the image to extract the image boundaries. The image boundaries are then traced to produce its chain-code digitized curve representation.

2.2 Corner Detection

In this step, rough locations of the possible corner points are identified. A fixed-length chord is moving along the digitized curve step by step, and the area between the chord and the associated curve segment is measured (see Fig. 1 - Fig. 2).

Suppose the digitized curve contains n points, represented as $P_0, P_1, ...P_{n-1}$, and the fixed chord length is L. Starting from the first point P_0, compute the length of chord P_0P_1. If the length is smaller then L, compute the length of chord P_0P_2. If it is still smaller than L, move to the next point P_3. This process stops at point P_i where the length of chord P_0P_i is equal to or longer than L. Calculate the area between the chord P_0P_i and the associated curve segment. If the coordinates for point P_k is $P(x_ky_k)$, the formula to compute the area is,

$$S_{0i} = \begin{vmatrix} x_0 & x_1 & x_2 & \cdots & x_i & x_0 \\ y_0 & y_1 & y_2 & \cdots & y_i & y_0 \end{vmatrix} \qquad (2)$$

Expressing the formula in another way, we have

$$S_{0i} = \sum_{p=0}^{i} (x_p y_{p+1} - x_{p+1} y_p) + (x_i y_0 - x_0 y_i) \qquad (3)$$

Next, move starting point from P_0 to P_1 and look for the end point P_j so that the length of chord P_1P_j satisfies the same condition: it is equal to or greater than L. This may be in point P_{i+1}, in point P_{i+2}, etc. Starting from second point, an incremental method will be used to compute the area.

In general, suppose the chord-curve area from point P_i to point P_j S_{ij} has been calculated, we want to compute the chord-curve area S_{i+1k} which is from point P_{i+1} to point P_k (see Fig. 3).

First, the starting point is moved one step forward from P_i to P_{i+1} and the algorithm looks at the area change ΔS_{ij}^{i+1} between area S_{ij} and area S_{i+1j} which is the triangular formed by three vertices of P_i, P_{i+1}, and P_j. The area is equal to

$$\Delta S_{ij}^{i+1} = -\begin{vmatrix} x_i & x_{i+1} & x_j & x_i \\ y_i & y_{i+1} & y_j & y_i \end{vmatrix} \qquad (4)$$

Next, the ending point is moved l steps forward from P_j to P_{j+l}, and we define the area increment from S_{i+1j} to S_{i+1j+l} as ΔS_{i+1j}^{j+l}. ΔS_{i+1j}^{j+l} can be expressed as

$$\Delta S_{i+1j}^{j+l} = \begin{vmatrix} x_j & x_{j+1} & \cdots & x_{j+l} & x_{i+1} & x_j \\ y_j & y_{j+1} & \cdots & y_{j+l} & y_{i+1} & y_j \end{vmatrix} \qquad (5)$$

Let $l = k - j$ and combine the result from above equations, we have

$$S_{i+1k} = S_{ij} - \begin{vmatrix} x_i & x_{i+1} & x_j & x_i \\ y_i & y_{i+1} & y_j & y_i \end{vmatrix} + \begin{vmatrix} x_j & x_{j+1} & \cdots & x_k & x_{i+1} & x_j \\ y_j & y_{j+1} & \cdots & y_k & y_{i+1} & y_j \end{vmatrix} \qquad (6)$$

The formula 6 uses the result from earlier calculation and only area change needs to be computed. This makes the algorithm very efficient.

Having computed the areas between the fixed-length chord and each associated curve segment, we will have n chord-curve area values. They correspond to chord-curve $P_0P_{k_0}$, $P_1P_{k_1}, ..., P_{n-2}P_{k_{n-2}}, P_{n-1}P_{k_{n-1}}$. These area values will be used to identify potential corner points.

We will scan the area values and look for local maxima. If the area values reached a local maximum at $P_mP_{k_m}$, and that area value is larger than a predefined threshold value v, then that value is compared with the area values of two other chord-curves whose starting points are L distance away from the starting point P_m and located at both sides of the point P_m. If area value is 30% bigger than those two area values, this curve segment is considered to have a corner point and will be marked for further processing. After this step, a list of curve segments will be marked, each of which contains a candidate corner point. That is, we have roughly located positions of all the corner points.

2.3 Corner Localization

The exact location of corner point on each marked curve segment will be decided in the final step. This is done by comparing the changes of directions of the marked digitized curve segment at each point. Usually the chord length value L is quite small. We can simply select the middle point of the curve segment as the corner point without much error. However, to get an accurate location of the corner point within the curve segment, we can choose the point in the segment whose direction change is the biggest as the corner point. The calculation of direction change at a point can be done as follows:

For that point, two chords are used, one is going forward, another is going backward. The length of the two chords can be chosen the same as length L. Let S be the area of the triangular formed by the two chords and an additional line segment (see Fig. 4). If the length of the additional line segment is b, the direction change e at that point can be expressed as

$$e = S/b \tag{7}$$

The point having the maximum direction change value will be assigned as the corner point.

Two parameters are used in this algorithm: chord length and the threshold value. They are depended on the image's resolution and the noise level and can be calculated automatically.

3. Experimental Results

This algorithm has been successfully used for the automatic generation of outlined font description from its bit-mapped images. We have used this algorithm on many Oriental font images, all with a very good result. See Fig. 5 for an example of a Chinese character.

In addition, many other images are also tested to examine the robustness of the algorithm. Fig. 6 shows the testing results on eight popular images cited from literature [6]. We can conclude that the algorithm is proved to be efficient and gives results close to human expectations.

4. Conclusion

We have presented a new corner detection algorithm. This algorithm uses the area values between a fixed-length chord and a series of curve segments along the digitized curve to represent the average curvature for the corresponding curve segment. An incremental formula is used for the computation. This makes the algorithm very efficient. The algorithm has been tested on many images and achieved very good result. It is also integrated into our outlined font generation system.

References

[1] H. Lynn Beus and Steven S.H. Tiu, *"An improved corner detection algorithms based pn chain-code plane curves"*, Pattern Recognition, Vol. 20, No. 3, 1987, pp.291-296.

[2] Chan Luen Kai and Tan Min Shyan, *Outline font description from bit-mapped data"*, FYP P286/93, School of EEE, NTU, Singapore, pp.37-55.

[3] H. Freeman and L.S. Davis, *"A corner finding algorithm for chain-coded curves"*, IEEE Trans. Computer, C-26, 1977, pp.297-303.

[4] L. Kitchen and A. Rosenfield, *"Gray-level corner detection"*, Pattern Recognition Letters, 1:95-102, 1982.

[5] C. Harris, *"Determination of ego-motion from mathced points"*, Proc. 3rd Alvey Vision Conference, Cambridge, Sept. 1987, pp.189-192.

[6] Hong-Chin Liu and M.D. Srimath, *"Corner detection from chain-code"*, Pattern Recognition, Vol. 3, No.12, 1990, pp.51-68.

[7] Rajiv Metrotra and Sanjay Nichani, *"Corner detection"*, Pattern Recognition, Vol. 23, No.11, 1990, pp.1223-1233.

[8] Shao Lejun, Chan Luen Kai, and Tan Min Shyan, *"On Corner Detection from Chain-Code"*, Proc. of 5th Int'l Symposium on IC Technology, Systems & Applications, September 15-17, 1993, pp.553-557, Singapore.

[9] Han Wang and Michael Brady, *"Corner detection with subpixel accuracy"*, Technical Report No. OUEL 1925/92, Dept. of Engineering Science, University of Oxford, 1992.

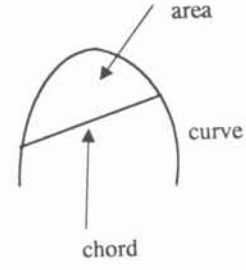

Fig. 1 Chord, Area, and Curve

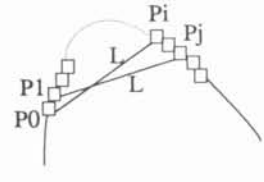

Fig. 2 Movement of the chord along the curve

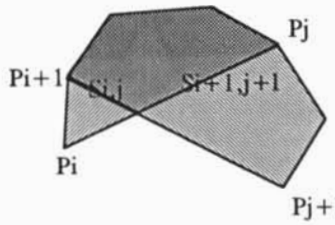

Fig. 3 Chord-curve area computation using incremental method

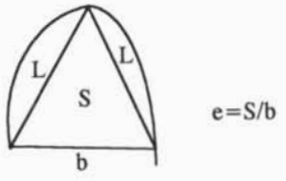

Fig. 4 Compute the direction change at a point

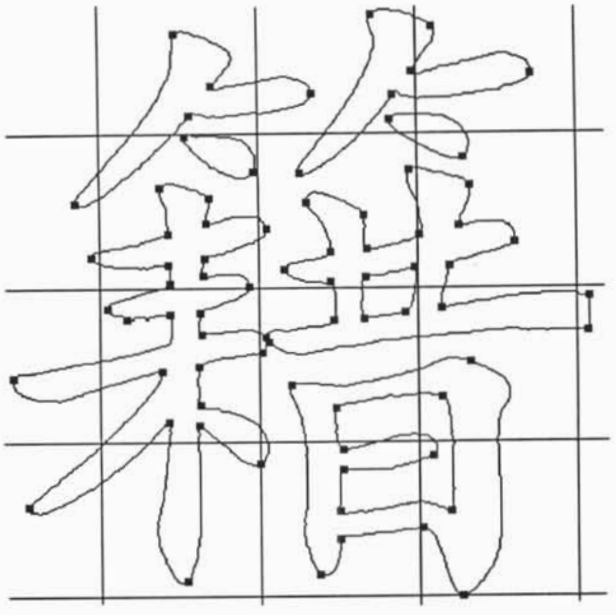

Fig. 5 Corner detection for a Chinese character

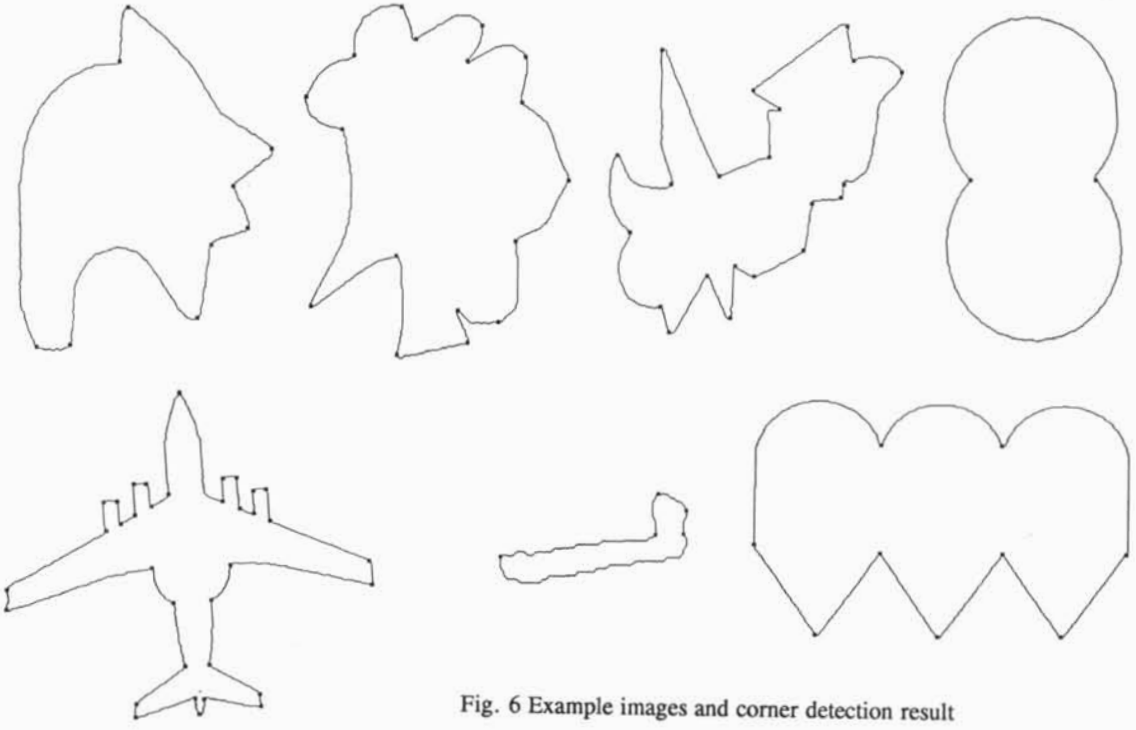

Fig. 6 Example images and corner detection result