TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol.12, No.5, May 2014, pp. 3818 ~ 3824
DOI: http://dx.doi.org/10.11591/telkomnika.v12i5.5099

One-pass Moment Algorithm for Graphical Primitives

Ju Zhiyong*, Su Chunmei
Shanghai Key Lab of Modern Optical System, University of Shanghai for Science and Technology, Shanghai 200093, China
Corresponding author, e-mail: juzy@usst.edu.cn, suchunmei7685051@163.com

Abstract

Moment is a useful tool in character and drawing recognition. As the moment computation is time-consuming, the main aim in this area is to develop a fast algorithm. The commonly-used method is the discrete Green's theorem, which transforms the double summation into a summation on region contour. An improved discrete Green's theorem proposed in this paper, which removes the constraint on parity pairing, thus extends its validity in applying the algorithm to complex regions. The contour tracing is fulfilled by a boundary-tracing-automation, and the central moments for graphical primitives are computed by one-pass scanning algorithm, the calculation of the moment is greatly simplified.

Keywords: *graphical recognition, moment method, multiply connected, boundary tracking*

1. Introduction

It is well known that the geometric properties of a shape are systematically described by their geometric moments [11]. The first 10 lowest order moments represent fundamental geometric properties of an object, including area, centre of mass, and radius of gyration, orientation and skewness. Therefore the geometric moments and their invariants are useful in the recognition of graphical primitives. Generally, an engineering drawing, a map, or a table, contains many graphical primitives, and their recognition is based on computation of high order moments, and the moment computation is too expensive, thus speedup algorithm is an important issue on the problem.

Many fast algorithms of moment computation had been proposed [1-6]. The commonly-used technique is to apply the so-called discrete Green's theorem, which transforms the double-summation of moment computation into addition operations along the region contours. One disadvantage in these researches is to pair leftmost pixel with its rightmost partner for all segments of a region, which is time consuming, and restrain its applications on complex shapes.

In the existing methods, only one region is treated, or boundary pixels and their coordinates are given at beginning. The problem of obtaining coordinates of the boundary pixels was discussed only in paper [6]. It is well known that scarcely an image contains only one object. It is evident also that the graphical primitives cannot be recognized till their locations in the image are discerned, and their moments are computed. In this paper, we formulate an improved discrete Green's theorem, and propose an algorithm, by which moment computation for graphical primitives is completed by one pass of image scanning.

2. Geometric Moments and Discrete Green's Theorem

The geometric moments of graphical primitives are defined by:

$$m_{pq} = \sum_{i=0}^{M} \sum_{j=0}^{N} i^p j^q f(i, j) \quad (1)$$

Where $f(i, j)$ is the grey level of pixel (i, j). Shapes in a binary digital image are completely determined by their contours. For this reason binary images are often used in drawing

Received November 10, 2013; Revised December 30, 2013; Accepted January 13, 2014

representation and drawing recognition. For two-level image, the definition of the geometric moments is:

$$m_{pq} = \sum_{(x,y)\in\Omega} x^p y^q \tag{2}$$

Where Ω is the region occupied by the graphical primitives. It is a double summation defined on the object regions. Though summation is a simple operation in computer, in order to recognize the characters and symbols in a drawing, large quantity of high order moments must be calculated for numerous objects, thus the speed of the algorithm is crucial to its applications. The main technique used in speedup the moment algorithm is to transform original double summation into a summation along the contour of the object by applying various versions of the discrete Green's theorem.

By some manipulation on formula (2), we have:

$$m_{pq} = \sum_{y=y_{\min}}^{y_{\max}} \sum_{x=x_{\min}(y)}^{x_{\max}(y)} x^p y^q = \sum_{y=y_{\min}}^{y_{\max}} y^q \left(\sum_{x=x_{\min}(y)}^{x_{\max}(y)} x^p \right) \tag{3}$$

Where $y_{\min}$ and $y_{\min}$ are, respectively, the minimum and maximum ordinates of the region, and $x_{\min}(y), x_{\max}(y)$ are, respectively, the leftmost and rightmost abscissa of y segment. By introducing notation:

$$S_p\left(x_{\min}(y), x_{\max}(y)\right) = \sum_{x=x_{\min}(y)}^{x_{\max}(y)} x^p \tag{4}$$

Formulae (4) is expressed as:

$$m_{pq} = \sum_{y=y_{\min}}^{y_{\max}} y^q S_p\left(x_{\min}(y), x_{\max}(y)\right) \tag{5}$$

It is the conventional discrete Green's theorem used in moment calculations. It is worth noting that in some versions of discrete Green's theorem, one need to perform calculations at both top and bottom boundary pixels. Formula (5) is a good version of discrete Green's theorem, as it needs only to perform calculations on right and left boundary. Now the only problem is to locate the leftmost and rightmost point and pair them, which will be discussed in next section.

3. Improved Discrete Green's Theorem

It is a cumbersome task to pair right boundary pixels with their left partners, which makes also discrete Green's theorem invalid for concave polygons and multi-connected regions. For a given y , we need to calculate:

$$S_p(l,r) = \sum_{x=l}^{r} x^p \tag{6}$$

That is to perform the summation between the leftmost and the rightmost pixel of a give segment. If the contribution to $S_p(l,r)$ from the left boundary pixel and the right boundary pixel can be discomposed, the difficulty in pairing is removed.

After selecting a coordinate origin, there are three configurations which form by the positions of the left and the right boundary pixel with respect to the origin. We must analysis in

detail the contributions to $S_p(l,r)$ from the left and the right boundary pixel in these configurations.

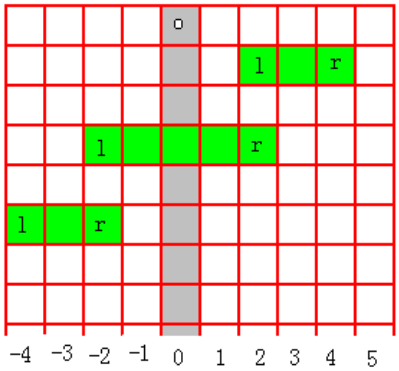

Figure 1. Three Configurations Formed by the Locations of the Left and Right Boundary Pixel with Respect to Coordinate Origin

Consider first the case of $p=0$. When $p=0$, in all these three configurations, we have formula,:

$$S_0(l,r) = r - l + 1 \tag{7}$$

As for $p \neq 0$, it is not difficult to find the contributions of the leftmost pixel or the rightmost pixel in all these three configurations, but the derivation is some tedious, thus it is convenient to write directly the results, which correspond, respectively, to top, middle and bottom configuration:

$$S_p(l,r) = f_p(r) - f_p(l-1) \tag{8}$$

$$S_p(l,r) = (-1)^p f_p(|l|) + f_p(r) \tag{9}$$

And,

$$S_p(l,r) = (-1)^p \left[f_p(|l|) - f_p(|r|-1) \right] \tag{10}$$

Where we have used the notation:

$$f_p(s) = \sum_{x=0}^{s} x^p \tag{11}$$

In following discussions, we will select a boundary pixel as the origin of the reference coordinates, and perform moment computations with respect to this point. Let coordinates of these three points in the world system are, respectively, x_o, x_l, and x_r. By using following notations:

$$\hat{r} = \begin{cases} x_r - x_o, & if \quad x_r > x_o \\ |x_r - x_0| - 1, & if \quad x_r < x_o \end{cases} \tag{12}$$

$$\hat{l} = \begin{cases} x_l - x_o - 1, & if \quad x_l > x_o \\ |x_l - x_0|, & if \quad x_l < x_o \end{cases} \tag{13}$$

Formulae (8)-(10) for all three configurations can be expressed as:

$$S_p(l,r) = sig_p(x_r - x_o)\, f_p(\hat{r}) - sig_p(x_l - x_o)\, f_p(\hat{l}) \tag{14}$$

Where:

$$sig_p(a-b) = \begin{cases} -1, & if \quad a-b<0, \quad and \quad p = odd \\ 1, & otherwise \end{cases} \tag{15}$$

Therefore, the contributions to $S_p(l,r)$ from the leftmost point and the rightmost point can be calculated separately, if their positions with respect to the origin of reference system are calculated. Thus, the remaining problem is to locate the coordinates for all the leftmost and the rightmost pixels.

4. Contour Following and Boundary Pixel Locating

4.1. Boundary-tracing Automaton

If an image contains only one region, its boundary can be located by a boundary-tracing algorithm [10]. The most efficient algorithm in boundary tracing was proposed by Rosenfeld [7]. Present problem is not simply to trace a given simple-connected region, but is to follow contour, to locate and mark the boundary pixels of graphical primitives, and to compute their moments. We design an automaton to meet all these requirements.

If a boundary pixel is located, and we like to follow region out of the region perimeter, an automaton can be designed to carry this work. The Inner states of the automaton are given in Figure 2.

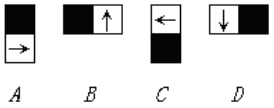

Figure 2. Inner States of the Boundary-tracing Automaton

The black pixel in Figure 2 denotes the region of the graphical primitives, and the arrow in white pixel denotes both the direction of automaton movement, and its present location. The transition mapping of the automaton in state A is given in Figure 3. The transition mappings for other three states are obtained by simply rotating Figure 3 by 90, 180 and 270 degrees.

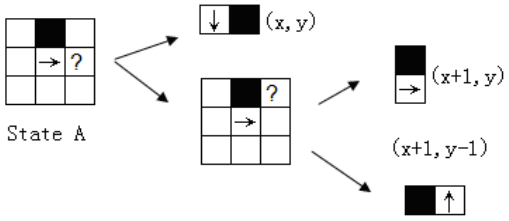

Figure 3. Transition Mapping of Automaton in State A

The pixel with "?" is to be checked: If it is black, take downward path, otherwise, upward path. The drawings at the right side of the large arrow give the state of the automaton at next time step, and the numbers in brackets gives the increments in coordinates between successive time steps. Color blue the pixel occupied by automaton when its state is D, and color red the left pixel of the automaton when its state is B. Beginning with the position of the starting pixel, and add successively the increments in coordinates, which gives the coordinates of every boundary pixel with respect to the origin of the reference system.

4.2. Boundary Marking to Avoid Re-tracing [9]

For the images containing many objects, it must take a measure to avoid re-tracing any region boundary [12]. The problem is not simple as its appearance, and the difficulty motivated Seong-Dae Kim et al. to propose a two-phases-algorithm in which a binary image is first written in run-length coding and then convert them into chain code [8]. Owing to the ability of automaton in discerning the left and right boundary pixels, it is easily to prevent from re-tracing by using boundary marking.

Figure 4. Character Perimeters Give Full Information about English and Chinese Sentence

The first and the second line in Figure 3 are, respectively, “This is a sample.” in English and in Chinese. Though only character perimeters are drawn, it gives full information about the words. This example illustrates obviously that in drawing recognitions only the size and the contour of an object is important. Thus it is adequate to consider the method that computes only the geometrical moments for each perimeter of the graphical primitives in an image.

A flag is used to denote whether the automaton enters or leaves the perimeter of a graphical primitive. It is named m_Flag and set $m\_Flag = 0$ at the beginning of the computation. The scanning procedure begins with the bottom line of the image, and it checks pixel colors from left to right, and from bottom to top. In this manner of scanning, only the left boundaries and the right boundaries can be detected, thus if the scanning undergoes a white-to-black transition and $m\_Flag = 0$, a new perimeter is detected, and the automaton commences the contour following. Encountering blue pixel sets $m\_Flag = 1$, and encountering red pixel sets $m\_Flag = 1$. It is ease to see that when $m\_Flag = 1$, the pixel under checked is certainly within a perimeter. Marking boundary pixels in this manner prevents automaton from re-tracing contours of graphical primitives.

5. Central Moments for Graphical Primitives

If the origin of the reference coordinates in the world system is (x_o, y_o), the geometric moments of a graphical primitive with respect to this reference point are:

$$m_{pq}(x_o, y_o) = \sum_{(x,y)\in\Omega} (x - x_o)^p (y - y_o)^q \tag{16}$$

The letters in bracket denote explicitly the location of the reference point. The centroid of a graphical primitive is calculated by:

$$(x_c, y_c) = \left(m_{10}(x_o, y_o), m_{01}(x_o, y_o)\right) / m_{00}(x_o, y_o) \tag{17}$$

Moments computed with respect to centroid are the central moments,

$$m_{pq}(x_c, y_c) = \sum_{(x,y)\in\Omega} (x - x_c)^p (y - y_c)^q \tag{18}$$

The central moments have some special meanings in the recognition of graphical primitives. For example, the central moments of second order are used for determining the principle axes of graphical primitives, and are important in investigating their symmetrical properties.

There is a simple relationship between the central moments and their general geometrical moments:

$$m_{pq}(x_c, y_c) = \sum_{(x,y)\in\Omega} (x - x_o - x_{co})^p (y - y_o - y_{co})^q \quad (19)$$

Formulae of central moments expressed by geometrical moments are obtained by expanding formula (19), and the formulae for central moments of lower orders are listed.

$$m_{00}(x_c, y_c) = m_{00}(x_o, y_o) \quad (20)$$

$$m_{10}(x_c, y_c) = m_{10}(x_o, y_o) - x_{co} m_{00}(x_o, y_o) \quad (21)$$

$$m_{01}(x_c, y_c) = m_{01}(x_o, y_o) - y_{co} m_{00}(x_o, y_o) \quad (22)$$

$$m_{11}(x_c, y_c) = m_{11}(x_o, y_o) - x_{co} m_{01}(x_o, y_o) - y_{co} m_{10}(x_o, y_o) + x_{co} y_{co} m_{00}(x_o, y_o) \quad (23)$$

$$m_{20}(x_c, y_c) = m_{20}(x_o, y_o) - 2x_{co} m_{10}(x_o, y_o) + x_{co}^2 m_{00}(x_o, y_o) \quad (24)$$

$$m_{02}(x_c, y_c) = m_{02}(x_o, y_o) - 2y_{co} m_{01}(x_o, y_o) + y_{co}^2 m_{00}(x_o, y_o) \quad (25)$$

Where $(x_{co}, y_{co}) = (x_c - x_o, y_c - y_o)$ is the projection of the line connecting point o with object centroid on the world system.

If we compute moments directly in world system, $f_p(s) = \sum_{x=0}^{s} x^p$ should be calculated 0 through image width. Alternatively, if moments are computed with respect to a boundary pixel of graphical primitive, the calculation of $f_p(s) = \sum_{x=0}^{s} x^p$ is restricted in the width of the graphical primitive. As for central moments, if the calculation is performed with reference to the centroid, the coordinates of a centroid are generally not integers, thus the calculations are performed with floating numbers. Besides, the coordinates of a centroid cannot be located till contour following is finished, thus the calculations of the central moments cannot be completed by one-pass scanning in this manner.

Now moment computation algorithm is formulated as follows. The image is scanned to detect the boundary pixel. A white-to-black transition with $m\_Flag = 0$ means a boundary pixel on a new graphical primitive, thus its coordinates in the world system is recorded. The automaton starts to trace the contour, and compute the geometric moments of the graphical primitives by applying the improved discrete Green's theorem. When automaton backs to its starting position, central moments for graphical primitives are computed by using formulae (19).

6. Conclusions

Figure 5 displays the results of this moment computation algorithm. Every left boundary pixel is blue-colored, and every right boundary pixel red-colored, which prevent region contours from re-tracing, and trace only the perimeters of the characters. Green pixels within character frames denote the centroids of the characters.

Figure 5. Trace the Outline of Characters

In this paper, we formulate first a new version of the discrete Green's theorem, which need only to treat the leftmost and the rightmost points of region segments. By discomposing the contributions from both the leftmost point and rightmost point, the difficulty in parity-pairing in conventional discrete Green's theorems is removed. Thus this improved discrete Green's theorem is not only capable of treating concave polygons, but also valid for multi-connected regions, and has no extra-calculation risen by the shape complexity. We design an automaton to fulfill the contour following, the leftmost pixel and rightmost pixel discerning, and the boundary marking. In this algorithm, geographical moments and central moments of all the graphical primitives in the image are computed by one-pass of image scanning. As a fast algorithm of moment computation, it provides a useful tool for drawing vectorization and drawing recognition. The method is hopeful in its applications in recognition of graphic symbols in charts and diagrams.

References

[1] YS Abu-Mostafa, D Psatlis. Recognitive aspects of moment invariants. *IEEE Trans. Pattern Analysis Mach. Intell.* 1984: 698-706.
[2] C The, RC Chin. On image analysis by the method of moments. *IEEE Trans. Pattern Analysis Mach. Intell.* 1988: 496-513.
[3] A Khotanzad, YH Hong. Invariant image recognition by Zernike moments. *IEEE Trans. Pattern Analysis Mack Intell.* 1990: 489-497.
[4] MF Zakaria, LJ Vroomen, PJA Zsombor-Murray, JMHM van Kessel, Fast algorithm for computation of moment invariants. *Pattern Recognition* .1987:639-643.
[5] BC Li, J Shen. Fast computation of moment invariants, *Pattern Recognition.* 1991; 24: 807-813.
[6] SH Lai, S Chang. Estimation of 3-D translational motion parameters via Hadamard transform. *Pattern Recognition Lett.* 1988; 8: 341-345.
[7] A Rosenfeld. Algorithms for Image/Vector Conversion. *Computer Graphics.* 1978; 20: 135-139.
[8] Seong- Dae Kim, Jeong-Hwan Lee, and Jae-Kyoon Kim, A new chain-coding algorithm for binary images using run-legth codes, CVGIP. 1988; 41: 114-128.
[9] Xiai Chen, Ling Wang, Wenquan Chen, Yanfeng Gao, Detection of Watermelon Seeds Exterior Quality based on Machine Vision. *TELKOMNIKA Indonesian Journal of Electrical Engineering.* 2013; 11(6): 2991-2996.
[10] Jun Yang, Tusheng Lin, Xiaoli Jin, An Image Sparse Representation for Saliency Detection. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2013; 11(10): 6143-6150
[11] TJ Watson. Research Center, Yorktown Heights NY. *An improved data stream algorithm for frequency moments.* SODA '04 Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms. 2004: 151-156
[12] T Herman, DF Robinson. Digital boundary tracking. Pattern Analysis & Applic.1998: 2-17