第17卷 第4期　　自 动 化 学 报　　Vol. 17, No. 4
1991 年 7 月　　ACTA AUTOMATICA SINICA　　July, 1991

具有线性加法复杂度的二值图象不变矩计算方法

李炳成　沈 俊
（东南大学生医系，南京）

摘　　要

不变矩方法是模式识别的一个重要方法，然而由于不变矩计算复杂，从而限制了它的应用．本文运用格林定理，将不变矩计算由平面域转化为曲线域．在此基础上，提出了边界跟踪迭代的不变矩计算方法．新方法不仅不需要乘法，而且加法次数亦从 $O(N^2)$ 降低到 $O(N)$．

关键词： 不变矩，格林定理，边缘跟踪，计算复杂度．

一、引　　言

不变矩模式识别方法由于具有对物体的平移、旋转及比例变化不变的优点，已经在模式识别领域中得到较成功的应用[1-5]．目前，计算不变矩通常是在平面域内完成，计算复杂，实时实现困难，从而限制了该方法的应用[5]．本文运用曲线积分的格林定理，将不变矩计算由二维降低到一维，并采用边缘跟踪迭代的方法，不仅将乘法转化为加法，而且保证加法次数为线性复杂度．

二、不变矩计算的格林定理

二值图象 $f(x, y)$ 可以表示为

$$f(x, y) = \begin{cases} 1, & (x, y) \in D, \\ 0, & (x, y) \notin D. \end{cases} \tag{1}$$

这里 D 为物体的图象区域．由(1)式可以给出矩的计算公式

$$m_{pq} = \iint_D x^p y^q dx dy. \tag{2}$$

式(2)计算可以通过格林定理简化．

定理 1 （格林定理）．设区域 D 为单连域，l 是绕 D 的正向轮廓线，则有

$$m_{pq} = \frac{1}{p+1} \int_l x^{p+1} y^q dy, \tag{3}$$

本文于 1990 年 3 月 1 日收到．

$$m_{pq}=-\frac{1}{q+1}\int_l x^p y^{q+1}dx. \tag{4}$$

如果D为多连通域,可以将其化为单连域,因此,本文仅考虑D为单连域的情况.

定理1表明,m_{pq} 计算有两种方法,一是对横坐标积分,一是对纵坐标积分,两种积分都是在曲线上进行,与直接方法(或文献[5])比较,计算复杂度由 $O(N^2)$ 降低到 $O(N)$.

将式(3)与(4)离散化得

$$m_{pq}=\frac{1}{p+1}\sum_{(i,j)\in l} i^{p+1}\cdot j^q\Delta j, \tag{5}$$

$$m_{pq}=-\frac{1}{q+1}\sum_{(i,j)\in l} i^p\cdot j^{q+1}\Delta i. \tag{6}$$

这里 Δi 及 Δj 为

$$\Delta i=\begin{cases}1, & \text{在 }(i,j)\text{ 处},\ l\text{ 与 }x\text{ 轴夹锐角},\\ 0, & \text{在 }(i,j)\text{ 处},\ l\text{ 与 }x\text{ 轴夹直角},\\ -1, & \text{在 }(i,j)\text{ 处},\ l\text{ 与 }x\text{ 轴夹钝角},\end{cases} \tag{7a}$$

$$\Delta j=\begin{cases}1, & \text{在 }(i,j)\text{ 处},\ l\text{ 与 }y\text{ 轴夹锐角},\\ 0, & \text{在 }(i,j)\text{ 处},\ l\text{ 与 }y\text{ 轴夹直角},\\ -1, & \text{在 }(i,j)\text{ 处},\ l\text{ 与 }y\text{ 轴夹钝角}.\end{cases} \tag{7b}$$

由(5)式及(6)式知,运用曲线积分的办法虽能将矩计算降低到 $O(N)$,但仍需要计算乘法. 为进一步降低计算复杂度,下面给出不需乘法的矩计算方法.

三、不变矩计算的边缘跟踪算法

不变矩的曲线积分计算方法分两步: 边界跟踪算法及曲线积分.

1) 边界跟踪算法. 该算法是: 已知 (i_k,j_k) 在边界上,在曲线 l 的正方向上找出下一点 (i_{k+1},j_{k+1}),该算法可采用 Ullmann 八邻域跟踪算法实现[6].

2) 曲线积分的计算.

由 Ullmann 跟踪算法知,(i_{k+1},j_{k+1}) 有8种可能: $(i_k,j_k\pm1)$, $(i_k\pm1,j_k)$, $(i_k\pm1,j_k+1)$ 及 $(i_k\pm1,j_k-1)$. 又由式(5)和(6)知, m_{pq} 是坐标单项式基的和,因此可以用迭代方法计算(5)和(6)式.

在计算不变矩时,不考虑(5)式及(6)式前的常数,只计算下式:

$$u_{pq}=\sum_{(i,j)\in l} i^{p+1}\cdot j^q\Delta j, \tag{8a}$$

$$v_{pq}=-\sum_{(i,j)\in l} i^p\cdot j^{q+1}\cdot\Delta i. \tag{8b}$$

在模式识别领域里,常使用前10个低阶矩, 即 m_{00}, m_{01}, m_{02}, m_{03}, m_{21}, m_{11}, m_{12}, m_{20}, m_{10}, m_{30},若前五个矩使用 u_{pq} 计算, 后五个矩使用 v_{pq} 计算, 由式(8a)及(8b)知,只需要计算下列单项式的基: i, $i\cdot j$, $i\cdot j^2$, $i^2\cdot j$, $i^3\cdot j$,然后对边界上每一点求和即求得 u_{pq} 和 v_{pq}.

为了计算 i, $i \cdot j$, $i \cdot j^2$, $i^2 \cdot j$, $i^3 \cdot j$,只需要计算下式:

$$A(i, j) = (i, i^2, i^3, i \cdot j, i^2 \cdot j, i^3 \cdot j, i \cdot j^2, i \cdot j^3, j, j^2, j^3).$$

由于 (i_{k+1}, j_{k+1}) 与 (i_k, j_k) 有八种可能的关系,分别分八种情况考虑.

1) 当 $i_{k+1} = i_k + 1$, $j_{k+1} = j_k$ 时,有

$$\left.\begin{array}{l} i_{k+1} = i_{k+1},\ \beta_1 = i_k^2 + i_k,\ i_{k+1} = i_{k+1} + \beta_1,\ i_{k+1}^3 = i_k^3 + i_{k+1}^2 + i_k + \beta_1, \\ j_{k+1} = j_k,\ j_{k+1}^2 = j_k^2,\ j_{k+1}^3 = j_k^3,\ i_{k+1}j_{k+1} = i_k j_k + j_k, \\ i_{k+1}j_{k+1}^2 = i_k j_k^2 + j_k^2,\ \beta_2 = i_k^2 j_k + i_k j_k,\ i_{k+1}^2 j_{k+1} = i_{k+1} j_k + \beta_2, \\ i_{k+1}^3 j_{k+1} = i_k^3 j_k + i_{k+1}^2 j_k + i_k^2 j_k + \beta_2. \end{array}\right\} \tag{9}$$

2) 当 $i_{k+1} = i_k - 1$, $j_{k+1} = j_k$ 时,有

$$\left.\begin{array}{l} i_{k+1} = i_k - 1,\ \beta_3 = i_k^2 - i_k,\ i_{k+1}^2 = \beta_3 - i_{k+1},\ i_{k+1}^3 = i_k^3 - i_k^2 - \beta_3 - i_{k+1}^2 \\ j_{k+1} = j_k,\ j_{k+1}^2 = j_k^2,\ j_{k+1}^3 = j_k^3,\ i_{k+1}j_{k+1} = i_k \cdot j_k - j_k, \\ i_{k+1}j_{k+1}^2 = i_k j_k^2 - j_k^2,\ \beta_4 = i_k^2 j_k - i_k j_k,\ i_{k+1}^2 j_{k+1} = \beta_4 - i_{k+1} j_{k+1}, \\ i_{k+1}^3 \cdot j_{k+1} = i_k^3 \cdot j_k - i_k^2 \cdot j_k - i_{k+1}^2 j_{k+1} - \beta_4. \end{array}\right\} \tag{10}$$

3) 当 $i_{k+1} = i_k$, $j_{k+1} = j_k + 1$ 时,只需将情况 1)中 i, j 交换一下,即可算出 $A(i_{k+1}, j_{k+1})$.

4) 当 $i_{k+1} = i_k$, $j_{k+1} = j_k - 1$ 时,只需将情况 2)中 i, j 交换一下即可求出 $A(j_{k+1}, j_{k+1})$.

5) 当 $i_{k+1} = i_{k+1}$, $j_{k+1} = j_k + 1$ 时,先按情况 1)做一次运算,再按情况 3)做一次运算即可.

6) 当 $i_{k+1} = i_k - 1$, $j_{k+1} = j_k + 1$ 时,按情况 1)与 3)各做一次运算即可.

7) 当 $i_{k+1} = i_k + 1$, $j_k = j_k - 1$ 时,按情况 1)和 4)各做一次运算即可.

8) 当 $i_{k+1} = i_k - 1$, $j_{k+1} = j_k - 1$ 时,按情况 2)和 4)各做一次运算即可.

由情况 1)—8)知,若 $A(i_k, j_k)$ 已知,只需加法即可求出 $A(i_{k+1}, j_{k+1})$, 由式(8a)和式(8b)知,求 u_{pq} 和 v_{pq} 不需要乘法,亦即求矩不需要乘法.

四、复杂度分析

在进行计算复杂度分析时,将一次比较折合成一次加法.

1) 比较次数. 设曲线 l 的象素点数为 W, 则边界跟踪及计算式(8a)和(8b)所需要的比较次数为 $12W$.

2) 加法次数. 设沿曲线行走一周, 在 x 轴及 y 轴方向上的变动的象素数分别为 N_x

表 1

方　　法	直接方法	文献[5]方法	本文方法
乘法次数	$O(N^2)$	$O(N)$	无
加法次数	$O(N^2)$	$O(N^2)$	$O(N)$

和 N_y,则计算多项式积及式(8a)和(8b)所需要的加法次数为 $19(N_x+N_y)$.

设一幅图象的尺寸为 $N\times N$,显然有

$$N_x\propto N,\quad N_y\propto N,\quad W\propto N.$$

因此,加法复杂度为 O(N),与其它方法计算次数的比较见表 1.

由表 1 可见,本文方法计算复杂度最低.

五、结束语

本文提出了二值图象不变矩的快速算法,该方法简单,计算复杂度低,可望在模式识别领域里得到应用. 本文讨论仅对二值图象而言,如何将本文方法应用到灰度图象矩的计算上有待于进一步研究.

参考文献

[1] Hu, M. K., Visual Pattern Recognition, IRE. Trans. Inform. Theory, IT-8, 179—187, 1962.

[2] Sajadi, F. A., et al., 3-D Moment Invariants, *IEEE. Trans. PAMI*-2, 2, 127—136, 1980.

[3] Abu-Mostafa, Y. S. et al., Recognitive Aspects of Moments, *IEEE Trans. PAMI*-6, 698—706, 1984.

[4] Lo, C. H. et al., 3-D Moment Forms, *IEEE Trans PAMI*-11, 10, 1053—1064, 1989.

[5] Zakaria, M. F. et al., Fast Algorithm For Computation of Moment Invariants, Pattern Recognition, **20**(1987), 6.

[6] Ullmann, J. R., Pattern Recognition Techniques, London, Butterwords, 1973.

AN INVARIANT MOMENT CALCULATION METHOD FOR BINARY IMAGE WITH LINEAR COMPLEXITY

LI BINGCHENG, SHEN JUN

(Dept of Biomedical Engineering, South-east Unversity, Nanjing.)

ABSTRACT

The invariant Moment method is an important technique means for pattern recognition. However, its application is limited by its computation complexity. .In this paper, Green theorem is used to change the plane integral into curve domain integral, and based on this, an iterative algorithm of edge tracing is proposed to calculate moment invariants. Compared with the known method, the new algorithm does not need multiplication operation and its addition complexiey decrease from $(O(N^2)$ to $O(N)$.

Key words: Moment Invariants, Green Theorem, Edge-tracing, Computation Complexity.