快速搜索任意形状二维目标质心策略

栾 新 朱铁一
(青岛海洋大学计算机系,青岛 266003)

摘 要 快速搜索任意形状二维目标的质心,一直是模式识别、目标跟踪等领域中的关键问题。通过对矩方法的进一步分析,提出基于目标质心与目标上各点间距离之和取得最小值这一特性的质心快速搜索策略。该策略从目标的边缘信息直接搜索得到目标质心的精确位置,目标形状为任意,具有极强的鲁棒性,适用于自主机器人目标定位、导航、跟踪和对接等系统中。

关键词 二维目标 质心 内积 矩方法

0 引 言

在实时图象处理和图象识别中,人们一直渴望获得目标与平移、比例和旋转无关的特征。这个问题可以通过采用 Hu 等定义的七个不变矩来解决[1],但其计算量大,不能满足实时性要求。基于对人类视觉机理的研究,Messner 提出了一种模拟人眼与大脑皮层间映射结构的极-对数坐标变换,该变换与傅立叶变换相结合,可快速提取出二维目标与平移、比例和旋转无关的特征[2],但这种方案与目标质心定位的准确性直接相关[3]。所以,在以多角度目标特征视图实现三维目标识别的策略中,快速搜索目标特征视图质心很重要[4]。质心的快速搜索也是运动目标跟踪、空间机器人定位、目标的捕捉及飞行器对接系统中的关键技术[5]。二维目标质心搜索方法很多,可分为两大类。其中一类仅限于规则二维目标,主要以平面几何理论和方法为基础[6]。另一类则对目标形状无任何限制。本文将对第二种方法进行研究。

通过计算图象的中心矩而得到图象中二维目标质心坐标是一种较为常用的方法,但计算时要做三个二重循环,计算量将随图象维数增大而变得难以接受[1]。利用极-对数坐标系与笛卡尔坐标系之间的迭代运算,也可以较好地定位二维图象中的目标质心[5],该方法的计算量同样很大,而且其坐标变换只是准备工作,主要工作是图象的相关匹配,若没有专门的硬件支持,时间和空间上的开销都将是无法承受的。Fong 和 Brown 提出了一个采用加权模板结构查找目标质心的方案[7],思想和方法都很好,但其计算量仍很大。为解决上述问题,本文提出质心快速搜索策略。

1 原理解析

1.1 矩方法计算质心坐标分析

对于一个 $N\times N$ 维的数字图象 $f(i,j)$(为简化说明,其定义如式(1)),利用中心矩计算质心坐标 (x_c,y_c) 如式(2):

$$f(i,j)=\begin{cases}1, & (i,j)\in \text{目标}\\ 0, & (i,j)\notin \text{目标}\end{cases} \tag{1}$$

$$\begin{aligned}x_c &= m(1,0)/m(0,0)\\ y_c &= m(0,1)/m(0,0)\end{aligned} \tag{2}$$

其中 $m(u,v)=\sum_{i=1}^{N}\sum_{j=1}^{N}f(i,j)\ i^u j^v$。

可见,x_c、y_c 分别为目标上点的横、纵坐标的均值。由概率统计知识可知:

$D_x=\sum_{i=1}^{N}(x_i-x)^2$ 当 $x=x_c$ 时 D_x 取得最小值;

$D_y=\sum_{i=1}^{N}(y_i-y)^2$ 当 $y=y_c$ 时 D_y 取得最小值。

其中,x_i, y_i 满足 $f(x_i,y_i)=1$。

将 D_x 和 D_y 合并:

收稿日期:1998-09-03;收到修改稿日期:1998-10-11

$$D_x + D_y = \sum_{i=1}^{N}(x_i - x_c)^2 + \sum_{i=1}^{N}(y_i - y_c)^2$$

$$= \sum_{i=1}^{N}[(x_i - x_c)^2 + (y_i - y_c)^2] \quad (3)$$

令 $d_i = (x_i - x_c)^2 + (y_i - y_c)^2$，则 $D_x + D_y = \sum_{i=1}^{N} d_i$。$d_i$实质为目标上的$(x_i, y_i)$点与质心点$(x_c, y_c)$的距离平方。显然，在$(x_c, y_c)$处 $D_x + D_y$ 取得最小值，$\sum_{i=1}^{N}\sqrt{d}$ 同样也取得最小值。

因此可得结论：目标质心与目标上所有各点间距离之和与其它各点相比，值最小。

1.2　基于距离之和质心搜索策略

根据上述结论，便可以开始二值图象中目标质心的搜索过程。

取图象 $f(i,j)$上任一点作为质心搜索过程的起始点，称为初始动态质心。搜索过程将由此点开始直至搜索到目标的质心位置。令(k, l)为当前动态质心点坐标，做欧氏距离模板 $d(i-k, j-l) = [(i-k)^2 + (j-l)^2]^{\frac{1}{2}}$ 与图象 $f(i-j)$的内积：

$$h(k, l) = \sum_{i=1}^{N}\sum_{j=1}^{N} d(i-k, j-l)f(i, j) \quad (4)$$

其值表示图象与模板在动态质心点(k, l)的相关值，实质上是图象上所有点到动态质心(k, l)的距离和。

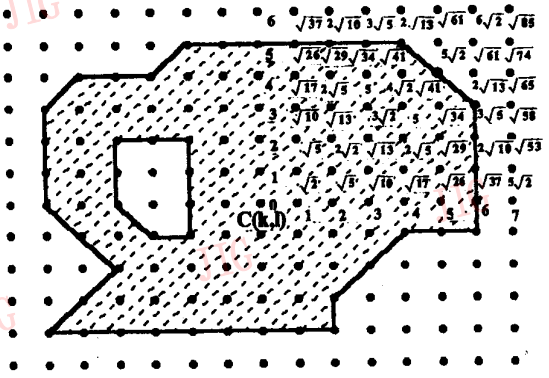


图 1　内积的求取过程
（由于模板的对称性，故只标出第一象限的权值）

2　基于“街区距离”的质心快速搜索策略

图 3 中，设 C'点为当前动态质心，C 为移动前动态质心。图中粗线表示目标移动前的轮廓线，细线为目标下移一个象素后的轮廓线。事实上，目标下移相当于模板上移。通过对比移动前后目标轮廓线的变化可以看出，一部分目标上的点变为背景（称为移出点），同时一部分背景点被目标所覆盖（称为移入点）。显然，向某一方向 p 移动前后按式(4)所

内积求取过程的直观表示如图 1。搜索质心就是计算内积值 $h(k, l)$ 沿某个轨迹下降的极小化过程。按式(4)搜索质心，确定初始动态质心走向时，要做 9 个内积运算，而在其后的每步搜索过程中都要做 8 个内积运算。若图象的维数很大，所选择的初始动态质心距真实质心较远，则搜索的时间和计算量将会变得相当大。

现分析质心搜索过程中动态质心的行走方向。走向编码如图 2 所示，在图 1 中，C 点为当前的动态质心，模板中心正好与其重合，假设真实质心在第Ⅰ象限中。质心是目标质量中心对称点，C 点向 4、5、6 三个方向移动，会使质心所在象限中目标的点数增加或不变（当目标全在此象限内时不变），这时将会变得更加不平衡（内积值 $h(k, l)$增大）；C 点向 0、1、2 三个方向移动时，第Ⅲ象限中的目标点数增加或不变，第Ⅰ象限中的目标点数减少或不变，动态质心向真实质心逼近；向方向 7 的移动是方向 0、6 的合成，此时第Ⅰ象限中目标点数减少或增加不定，既使减少，其程度也不如方向 0；向方向 3 移动与方向 7 类似。又由于模板的权值本身也是中心对称的，所以内积值 $h(k, l)$下降最快的方向一定是使质心所在象限内目标点数减少最多的方向，且当动态质心与真实质心重合时达到平衡状态。若再向其它方向移动，都将导致内积值增加。如图 1，在质心的搜索过程中，动态质心只能在 0、1、2 三个方向移动，不必在 8 个方向上搜索和判断。

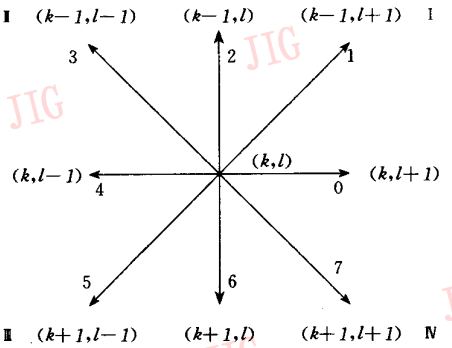
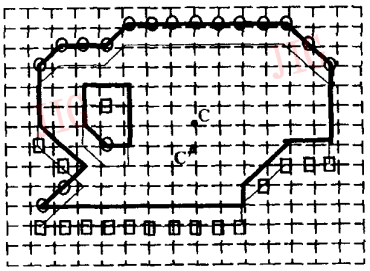


图 2　动态质心(k,l)的邻点，走向编码$p=0,1,\cdots,7$及区域划分

得到的两个内积值具有如下迭代关系：

$$h(k+a,l+b)$$
$$=h(k,l)+(\text{移入点权值和}-\text{移出点权值和})$$
$$=h(k,l)+\Delta_p h \quad p \text{ 在 } \{0,1,\cdots,7\} \text{ 中取值} \quad (5)$$

式中 a 和 b 根据移动方向 p 在 $\{-1,0,1\}$ 中取值，但 a、b 不能同时为零。表1给出了移动方向 p 和对应 a、b 的取值。对应图3，$a=1,b=0,p=6$。



□：表示移入点； ○：表示移出点

图3 模板移动示意图

表1 a、b的取值与移动方向p之间的关系

p	0	1	2	3	4	5	6	7
a	0	-1	-1	-1	0	1	1	1
b	1	1	0	-1	-1	-1	0	1

式(5)中，动态质心移动方向的确定与 $\Delta_p h$ 有关，动态质心应向内积减少最快的方向移动。在最佳路径上必有：

$$\Delta_p h=\min\{(\Delta_p h \mid p=0,1,\cdots,7\}<0 \quad (6)$$

满足式(6)的 p 即为动态质心的本次走向。当然，在后面的讨论中可知，确定初始动态质心走向也不必计算9个邻点处的内积值，而且在搜索过程中每次走向判断也不必计算3个邻点处的内积值。

因为每次模板只移动一个象素，故 $\Delta_p h$ 只与目标的边缘点有关。因此，在图象二值化后，保留图象中目标的边缘点链码，同时记录每个边缘点的邻点属性信息(即各邻点是否属于目标)，利用这个信息很容易计算该点在移动一个象素后的属性变化。对于有孔目标，在孔处也同样处理。这样计算量将大幅度减少。图3所示目标中，目标上共有114个象素点，移动一个象素后，属性发生变化的象素只有34个(由背景点变为目标点和由目标点变为背景点各17个象素)，约占图象的1/3。

上面计算中采用的模板 d，是中心对称的欧氏距离模板。欧氏距离与坐标位置呈平方和开方的关系，其计算速度受大量的浮点运算影响。为此，这里采用中心对称街区距离模板代替欧氏距离模板，如图4(a)所示。中心对称街区距离模板又可进一步分解为横轴对称距离模板和纵轴对称距离模板，分别示于图4(b)、图4(c)中。事实上，这种分解的街区距离模板与原来的中心对称欧氏距离模板本质上是一致的。欧氏距离模板平方后便可分解为图4(b)和图4(c)两个模板之和，分别对应横轴和纵轴对称距离模板的平方形式。为进一步简化计算，图4(b)和图4(c)中的模板去掉了平方形式。采用分离的模板简化了真实质心的方向和动态质心走向的判断，因而计算速度有极大的提高。

⋮ ⋮ ⋮ ⋮ ⋮	⋮ ⋮ ⋮ ⋮ ⋮	⋮ ⋮ ⋮ ⋮ ⋮
⋯ 4 3 2 3 4 ⋯	⋯ 2 2 2 2 2 ⋯	⋯ 2 1 0 1 2 ⋯
⋯ 3 2 1 2 3 ⋯	⋯ 1 1 1 1 1 ⋯	⋯ 2 1 0 1 2 ⋯
⋯ 2 1 0 1 2 ⋯	⋯ 0 0 0 0 0 ⋯	⋯ 2 1 0 1 2 ⋯
⋯ 3 2 1 2 3 ⋯	⋯ 1 1 1 1 1 ⋯	⋯ 2 1 0 1 2 ⋯
⋯ 4 3 2 3 4 ⋯	⋯ 2 2 2 2 2 ⋯	⋯ 2 1 0 1 2 ⋯
⋮ ⋮ ⋮ ⋮ ⋮	⋮ ⋮ ⋮ ⋮ ⋮	⋮ ⋮ ⋮ ⋮ ⋮

(a) 中心对称街区距离权值模板　(b) 横轴对称距离权值模板　(c) 纵轴对称距离权值模板

图4 中心对称街区距离权值模板及分解

经上述简化与分解，模板 d 变为街区距离：

$$d(i-k,j-l)=d'(i-k)+d'(j-l)$$
$$d'(i-k)=|i-k| \quad (7)$$
$$d'(j-l)=|j-l|$$

$d(i-k,j-l)$和 $f(i,j)$的内积为：

$$h(k,l)=\sum_{i=1}^{N}\sum_{j=1}^{N}d(i-k,j-l)f(i,j)$$
$$=\sum_{i=1}^{N}|i-k|f(i,j)+\sum_{j=1}^{N}|j-l|f(i,j)$$
$$=h(k)+h(l) \quad (8)$$

即在点(k,l)的内积值为横、纵轴对称街区距离模板的中心与(k,l)点重合后所得到的街区距离之

和。向 p 方向移动一步(一个象素)后,式(5)中的迭代关系变为如下形式:

$$\begin{aligned} & h(k+a, l+b) \\ &= h(k, l) + \Delta_p h \\ &= h(k) + h(l) + \Delta_p h(k) + \Delta_p h(l) \end{aligned} \tag{9}$$

式中 $\Delta_p h(k)$和 $\Delta_p h(l)$分别是横、纵轴对称模板移动前后的权值变化。

式(8)中的 $h(k, l)$实际上是计算目标上所有的点到点(k, l)的街区距离和,$h(k)$、$h(l)$分别是所有目标上点到水平直线 $x = k$ 和垂直直线 $y = l$ 的距离之和。图 4(a)模板的中心与目标的质心重合时,式(8)取得最小值。图 4(b)、图 4(c)的对称轴穿过目标质心时,$h(k)$和 $h(l)$分别取得最小值。两对称轴同时穿过目标重心时,$h(k)$、$h(l)$同时取得最小值,式(8)也取得最小值。纵轴对称模板对于目标在垂直方向上的移动不起作用(内积值不产生任何变化)。同样,横轴对称模板对于目标在水平方向的移动不起作用。因此,对于一点(k, l),其邻点的内积值 h 的计算变得简单了。目标在(k, l)点的 0、4 方向移动一个象素,只要计算 $\Delta_0 h(l)$或 $\Delta_4 h(l)$即可。目标在(k, l)点的 2、6 方向移动一个象素,只要计算 $\Delta_2 h(k)$或 $\Delta_6 h(k)$即可。目标只有在 1、3、5、7 方向上移动时才需要 $\Delta_p h(k)$和 $\Delta_p h(l)$都做计算。

容易看出,若真实的目标质心与当前动态目标质心在同一水平或垂直线上,则只需用一个横轴或纵轴对称模板计算,这时的动态质心的移动轨迹必定是一条水平或垂直直线,由初始点直接到达目标真实质心。若真实目标质心在某一象限区域内,不妨假设在第Ⅰ象限中,此时动态质心向 0、2 方向移动都将使内积值减小,而方向 1 为方向 0、2 的合方向,因此向方向 1 移动,横、纵轴对称模板都将移动,这将使总的内积值达到最小。对于其它 5 个方向的移动都将使内积值增加。因此,动态质心必定应先走 1 方向的斜线,直到其与目标的真实质心处在同一水平或垂直线位置上为止,然后再走水平线或垂直线,最后到达目标的真实质心位置。

3　实　验

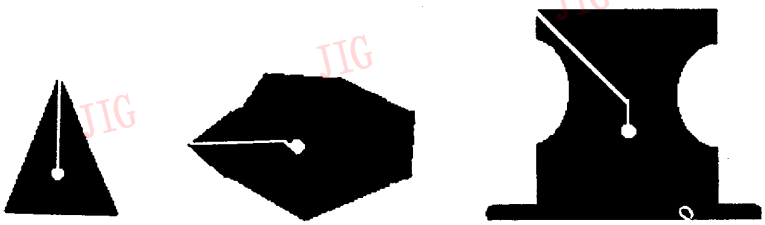


图 5　三个质心搜索实验结果

采用上述策略处理图 5 中三幅二值图象,其目标大小分别是 128 × 128、256 × 256 和 512 × 512。表 2是本算法与中心矩方法的结果对照。图 5 中目标上的白线是质心搜索的轨迹,圆的白点是目标的质心位置,白线的另一端是随机选取的质心搜索起始点(不失一般性,分别选取图象扫描时遇到的第一个边缘点)。从图中轨迹可以看出,搜索过程总是以最佳路线逼近目标真实质心。当图象很小(小于 100 × 100)时,由于新算法需要一定时间计算边缘属性信息,因而较计算中心矩方法慢,但随着图象的增大,新算法的优越性变得特别明显。图 6 是大量实验统计出的时间曲线。

表 2　中心矩方法和本文新算法的对比

	图象尺寸	计算中心矩方法		新算法		结果误差
		质心坐标	时间(s)	质心坐标	时间(s)	(pixel)
图 5(a)	128 × 128	(200.36,250.49)	0.2006	(200,250)	0.1982	0.6080
图 5(b)	256 × 256	(182.93,253.82)	0.7346	(183,254)	0.4024	0.1931
图 5(c)	512 × 512	(252.16,275.80)	2.2402	(252,276)	1.0026	0.1720

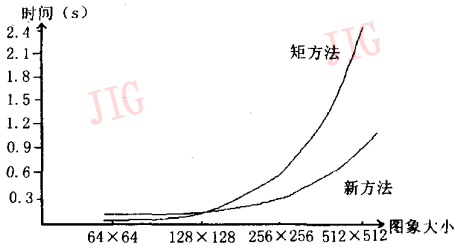


图6 计算中心矩方法和新算法的时间对比

4 结 论

本文分析了计算中心矩而获取目标质心坐标的方法所存在的问题，提出了一个基于目标质心与目标上各点间距离和取得最小值的质心搜索算法。欧氏距离模板又可进一步简化为两个街区距离模板，从而使原来的浮点运算(平方和开方)变为简单的整数加减运算，极大提高了计算速度。本文提出的新算法对高分辨率的高维图象极为适用，其精度和效率大大优于以往的质心搜索算法。

参 考 文 献

1 Hu M K. Visual pattern recognition by moment invariants. IRE Trans, 1962, IF8, 179～187.

2 Messner R A. An image processing architecture for real time generation of scale and rotation invariant patterns. CVGIP, 1985,(3): 50～60.

3 Bailey J G, Messner R A. Docking target design spacecraft tracking system stability. SPIE, 1989, 1192:820～831.

4 Seibert M, Waxman A M. Adaptive 3D object recognition from multiple views. IEEE Trans Pattern Analysis and Machine Intelligence, 1992, 14(7):107～124.

5 Verghese G, Gale K L, Dyer C R. Real-time motion tracking of 3D objects. In: Proc National Conf Artificial Intelligence, 1990.

6 Ballard D H, Brown C M. Computer vision. Department of Computer Science, University of Rochester, Rochester, New York, 1982.

7 Fong Yu-shan, Brown D H. A centriod tracking scheme in a weighted coordinate system. IEEE Trans Pattern Recognition and Computer Vision, 1985, 219～221.

栾 新 毕业于哈尔滨工业大学，获博士学位。现任青岛海洋大学计算系副教授。主要研究方向为计算视觉、模式识别技术。

朱铁一 毕业于哈尔滨工业大学，获博士学位。主要研究方向为计算机视觉、模式识别及机器人技术。

The Fast Algorithm for Searching Centroids of 2D Arbitrary Shape Object

Luan Xin, Zhu Tieyi

(*Department of Computer Science, Ocean University, Qingdao* 266003)

Abstract Fast searching centroids of 2D arbitrary shape is still a key problem in the fields of pattern recognition and object tracking. This paper presents a new fast algorithm by analyzing the moment method, which is based on the character that the sum of distances between centroid and other points is the minimum. The algorithm gets the centroid position just from edge information. The algorithm is robust and can be used in the systems of autonomous robot locating, navigation, tracking and docking.

Keywords 2D object, Centroid, Inner product, Moment method