林业科学研究 1996,9(1):75～79
Forest Research

GIS中目标选取算法的研究

乔彦友

摘要 本文分别对点、线及多边形的选取算法进行了研究。在点状图元选取中,以绝对距离代替通用的平方根距离,提高了选取效率。对于线状图元的选取,定义了一种运算量较小的点到曲线的距离,同样提高了选取效率。为了提高多边形的选取效率,对常用的判定一个点是否为多边形内点的"半直线"方法进行了改进,得到了一种运算量较小的算法。

关键词 地理信息系统、目标选取、多边形

在基于矢量存储的地理信息系统中,现实世界中的实体被抽象为点、线和多边形三种最基本的图形要素,通过为三种图形要素建立数据模型并进行存储就实现了空间数据库的建立。在建立空间数据库的过程中,除了要对底图进行数字化外,还必须对通过数字化所得到的数字化原图进行大量的编辑工作,以便空间数据库中各实体之间的空间关系满足相合性(Consistency)的要求。而在对空间数据库的交互编辑过程中,首先要用鼠标在屏幕上选择一个目标实体(或图元、图素),然后才能对被选定的图元进行编辑。用光标指定一个图元后,如何快速地从空间数据库中找出其对应的数据描述是一个值得重视的算法问题。本文将结合作者从事GIS开发工作的经验分别论述点、线和多边形三种基本图形要素选取的算法问题。

1 点状图素的选取算法

在矢量存储的GIS中,点状图素的存储形式如图1所示。

ID	X	Y
1	x_1	y_1
⋮	⋮	⋮

图1 点状图素的存储方式

用鼠标在屏幕上选取一个点状图素的过程是将鼠标对准屏幕上的某个点状图素按特定的鼠标键,然后根据这点的坐标从空间数据库中找出该图元。设鼠标指定的点 P_0 的坐标为(x_0,y_0),且已经转换为实际存储的坐标系下的值。

从点状图素的存储可以看出,它的选取算法比较简单,只须找出离 P_0 最近的点状图素即可。设空间数据库中共有 n 个点状图素,其对应的点分别为 $P_1\cdots P_n$,任意两点选取流程可由图2表示。

上述流程结束后,变量 m 的值就是所选取的点状图素的序号值,在程序实现过程中,可采用绝对值距离函数:

$$d(A,B) = |x_A - x_B| + |y_A - y_B|$$

1995—01—21收稿。

乔彦友助理研究员(中国林业科学研究院资源信息研究所 北京 100091)。

而不用平方根距离函数：

$$d^*(A,B) = [(x_A - x_B)^2 + (y_A - y_B)^2]^{1/2}$$

$d(A,B)$相对于 $d^*(A,B)$的优越之处在于它不用进行乘法和开平方运算，能缩短运算时间。

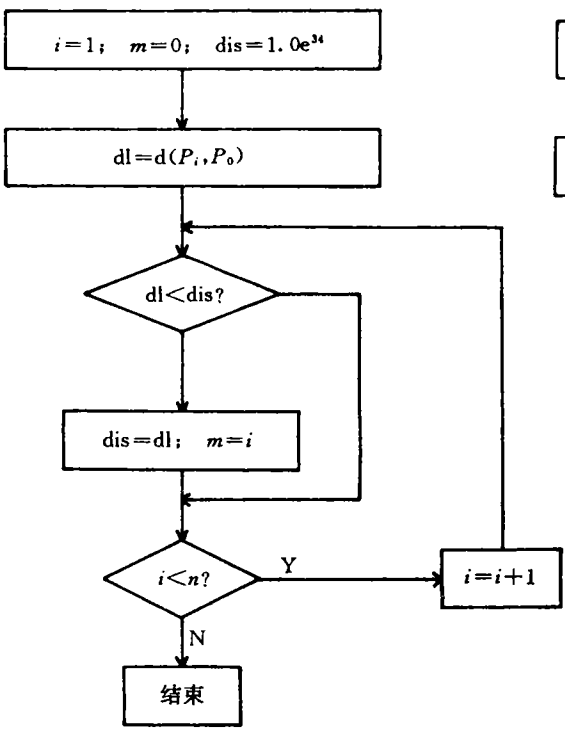

图 2 点状图素选取流程图

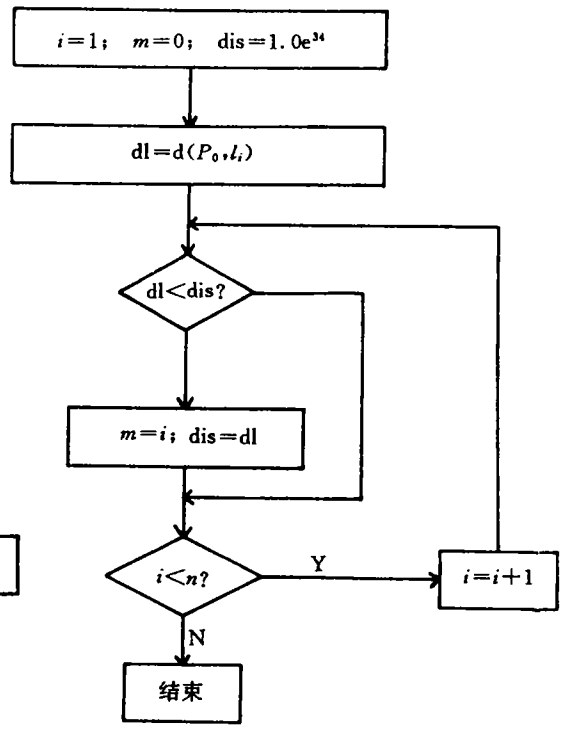

图 3 线状图素选取流程图

2 线状图素的选取算法

线状图素(弧段)的选取过程是用鼠标对准弧段上的任一位置按特定的鼠标键，然后根据一定的算法在空间数据库中找出该弧段所对应的数据。设鼠标指定的点 P_0 的坐标为(x_0,y_0)，线状图素的选择原则应当是找出离点 P_0 最近的弧段。

设空间数据库中共有 n 条弧段，记为$\{l_i\}$，$i=1,\cdots,n$。点 P_0 到任一弧段 l 的距离函数为 $\tilde{d}(P_0,l)$，用变量 m 记录最终所选到的弧段的序号，那么线状图元的选择流程可由上述图 3 所示。

在程序的实现过程中，点到弧段的距离函数 $\tilde{d}(P,l)$算法的好坏决定了线状图素选择算法的总体效率。下面简单论述它的定义及计算。

设任意两点之间的距离采用平方根距离：

$$d(A,B) = [(x_A - x_B)^2 + (y_A - y_B)^2]^{1/2}$$

设任一条直线段$\overline{P_1P_2}$所在的直线 l 的方程为

$$Ax + By + C = 0$$

过点 P_0 做 l 的垂线,并求得其与 l 的交点$(\tilde{x}_0,\tilde{y}_0)$,那么点 P_0 到线段$\overline{P_1P_2}$的距离为

$$d_1(P_0,\overline{P_1P_2})=\begin{cases}\dfrac{|Ax_0+By_0+C|}{\sqrt{A^2+B^2}} & \text{如}(\tilde{x}_0,\tilde{y}_0)\text{ 在}\overline{P_1P_2}\text{ 上}\\ \min(d(P_0,P_1),d(P_0,P_2)) & (\tilde{x}_0,\tilde{y}_0)\text{ 不在}\overline{P_1P_2}\text{ 上}\end{cases}$$

由于任一弧段由一系列线段组成,记为$\overline{P_iP_{i+1}}$,$i=1,\cdots,m-1$,因而点 P_0 到弧段的距离可定为:

$$\tilde{d}(P_0,l)=\min_{1\leqslant i<m-1}d_1(P_0,\overline{P_iP_{i+1}})$$

上述 $\tilde{d}(P,l)$的定义比较严格,在编制程序时如采用它,每次选取弧段都要计算点 P_0 到所有弧段上的所有线段的距离,运算效率非常低。在实际编制程序时可用下面的$\tilde{d}_1$来代替 d_1:

$$d_1(P_0,\overline{P_1P_2})=\begin{cases}d_1(P_0,\overline{P_1P_2}) & \text{当 }\min(x_1,x_2)-\delta\leqslant x_0\leqslant\max(x_1,x_2)+\delta,\\ & \min(y_1,y_2)-\delta\leqslant y_0\leqslant\max(y_1,y_2)+\delta\\ \infty & \text{其它}\end{cases}$$

其中,(x_1,y_1)为 P_1 点的坐标,(x_2,y_2)为 P_2 点的坐标,δ 为一个比较小的正数值,可以取为系统的容差值(Tolerance)或者它的倍数值。采用上式后,只有当点 P_0 离线段$\overline{P_1P_2}$比较近时,才计算它们之间的距离,这就大大地减少了求点到线段的距离的次数,提高了运算效率。

3　多边形的选取算法

多边形的选取过程是用鼠标对准多边形内任一点按某一特定鼠标键,然后根据一定算法从空间数据库中找出这一多边形。

设(x_0,y_0)为鼠标指定的多边形内的任一点,函数 isinpolygon(P_0,Poly)用于判定 P_0 是否为多边形 Poly 的内点,空间数据库中共有 n 个多边形,记为$\{\text{Poly}_i\}i=1,\cdots,n$。那么多边形的选取流程就是依次对各个多边形调用函数 isinpolygon,判断 P_0 是否为其内点,直至找到某个多边形,使 P_0 是其内点时为止。

可以看出,上述流程的核心就是判断点 P_0 是否为一个多边形的内点的函数 isinpolygon 的效率的问题。现在普遍采用的算法是铅锤线法,它的基本原理是如果要判断 P_0 是否为一个多边形的内点,可先在多边形外找一点 P,求线段$\overline{P_0P}$与多边形的交点的个数,如果交点数为奇数则 P_0 是多边形的内点,如果交点数为偶数,则 P_0 不是多边形的内点。特别地,可取 P 的纵坐标与 P_0 的相同,P 的横坐标为正无穷大,那么问题就化为求由 P_0 向右延伸的射线 l_0 与多边形的交点数目的问题。

由于多边形是由一系列线段组成,因而求 l_0 与多边形的交点数等价于求 l_0 与各线段的交点数的总和。又由于一个多边形一般由一个外围多边形和其内的若干岛多边形组成,求 l_0 与这个多边形的交点数等价于分别求 l_0 与外围多边形及求 l_0 与各个岛多边形的交点数,然后求和,因而在以后的论述中,我们可以假定多边形内不存在岛。

在实现上述算法时,有几种极端情况需要考虑。

第一种情况是点 P_0 在多边形的边界上,例如 P_0 与多边形的一个顶点重合,P_0 在多边形的一条水平线段上或者 P_0 在多边形的一条非水平线段上。当这些情况出现时,P_0 不是多边形

的内点，运算不必继续进行。

第二种情况是 l_0 与多边形的一个顶点相交，如图 4 所示。该两图中 P, P_1, P_2 都为多边形的顶点，而 l_0 与多边形交于顶点 P。它们的处理方法是：图 4(a)中的 P 点算为一个交点，因为过这点的两条线段 $\overline{P_1P}$ 及 $\overline{P_2P}$ 在 l_0 的两侧；图 4(b)中的 P 点算为两个交点(也可算为零个交点)，因为 $\overline{P_1P}$ 及 $\overline{P_2P}$ 在 l_0 的一侧。这种处理方法必须判断 $\overline{P_1P}$ 及 $\overline{P_2P}$ 是否在 l_0 的同一侧，增加了机时开销。

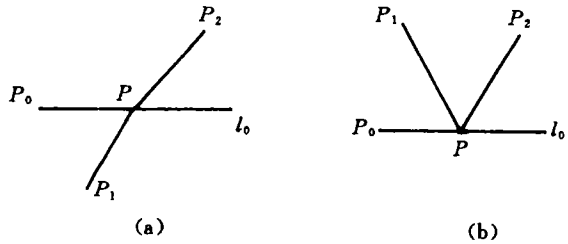

图 4 l_0 与多边形顶点相交的情况

第三种极端情况是 l_0 与多边形上的一段或数段水平线重合，如图 5 所示。它们的一种处理算法是将水平线段 $\overline{P_1P_2}$ 的两个端点合并为一个点，这时分别得到图 6(a)及 6(b)，这时就化为 l_0 与多边形的某个顶点相交的情况了。这种处理算法不但要判断水平线，而且要重新整理多边形的点的坐标，增加了额外的机时开销。

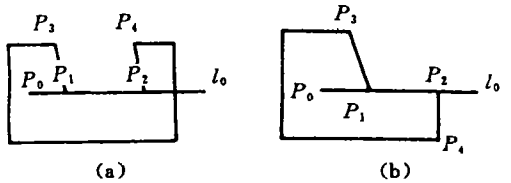

图 5 l_0 与多边形中一段水平线重合的情况

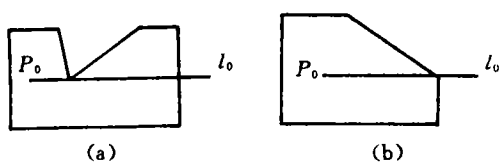

图 6 l_0 与多边形中一段水平线重合时的一种解决方案

通过对铅锤线法的描述和上述几种极端情况处理方法的分析，我们可以设想能否设计一种通用的算法，使它既能处理上述几种极端情况，又能克服前面处理方法所带来的缺陷，为此，本文提出了一种统一的算法流程。下面就详细叙述这种流程。

设多边形的顶点 $\{P_i\}i=1,\cdots,n$，l_0 与 $\overline{P_iP_{i+1}}$ 的交点数为 m_i，那么 l_0 与多边形的交点数为 Σm_i。为了描述求 m_i 的流程时方便，我们用“i 步结束”表示求 l_0 与 $\overline{P_iP_{i+1}}$ 交点的过程的结束，用“总结束”表示判断 P_0 是否为多边形内点的总过程的结束，那么求 m_i 的过程可以如下所述：

(Ⅰ)若 $x_i<x_0, x_{i+1}<x_0$，则 $m_i=0$，i 步结束；

(Ⅱ)若 $x_i==x_0$，则 P_0 不是内点，总结束；

(Ⅲ)若 $y_i==y_{i+1}==y_0$，且 $\min(x_i,x_{i+1})\leqslant x_0\leqslant\max(x_i,x_{i+1})$，则 P_0 不是内点，总结束；

(Ⅳ)如果条件 $\min(y_i,y_{i+1})\leqslant y_0$ 和 $\max(y_i,y_{i+1})>y_0$ 不全满足，那么 $m_i=0$，i 步结束；

(Ⅴ)如果条件 $\min(y_i,y_{i+1})\leqslant y_0$ 和 $\max(y_i,y_{i+1})>y_0$ 都满足，则求 l_0 与 $\overline{P_iP_{i+1}}$ 的交点的横坐标 $\tilde{x}$

(a)如果 $\tilde{x}==x_0$，则 P_0 不为内点，总结束；

(b)如果 $\tilde{x}\neq x_0$，则 $m_i=1$，i 步结束。

上述流程是自上而下依次进行的，其中(Ⅰ)保证了多边形只与 l_0 求交；(Ⅱ)及(Ⅲ)很容易地处理了 P_0 是多边形的一个顶点和在多边形的一条水平线上的情况，(Ⅳ)使得参与与 l_0 求

交的线段尽可能地少;(Ⅴ)中的(a)处理了 P_0 在多边形的一条非水平边上的情况,(Ⅴ)中的(b)则是当 l_0 与线段有交点,且交点不与 P_0 重合的情况。

这里需特别指出的是上述流程解决了 l_0 与多边形交于某个顶点和 l_0 与多边形的水平线段部分重合的情况,消除了以前处理这两种情况的缺陷。例如按本算法,在图 4(a)中,l_0 与$\overline{PP_1}$交点数为 0,与$\overline{PP_2}$的交点数为 1,因而 P 算为一个交点。在图 4(b)中 l_0 与$\overline{P_1P}$的交点数为 1,l_0 与$\overline{P_2P}$的交点数为 1,因而 P 算为两个交点。还有一种 l_0 与多边形交于顶点的情况就是过这点的两条线段在 l_0 下方,如图 7 所示。

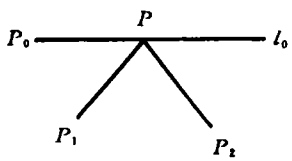

图 7　l_0 与多边形交于顶点的另一种情况

在图 7 中 l_0 与$\overline{P_1P}$及$\overline{P_2P}$的交点数都为 0(按上述流程),因而 P 算为 0 个交点,是偶数这不会影响 P_0 内点属性的判别。

在 l_0 与多边形的水平线部分重合的情况,按本流程,在图 5(a)中,P_3P_1与 l_0 的交点数为 1,$\overline{P_1P_2}$与 l_0 的交点数为 0,$\overline{P_4P_2}$与 l_0 的交点数为 1,因而 l_0 与$\overline{P_3P_1}$,$\overline{P_1P_2}$,$\overline{P_4P_2}$的交点总数为 2。同样可算出图 5(b)中 l_0 与$\overline{P_3P_1}$,$\overline{P_1P_2}$,$\overline{P_4P_2}$的交点总数为 1。由于第(Ⅲ)步已经排除了 P_0 在水平线$\overline{P_1P_2}$上的可能,因而这种算法不影响 P_0 是否在多边形内的判断。

本算法在笔者参与开发的微机地理信息系统 PCGIS 中得到应用,运算效率比较高。

参 考 文 献

1　乔彦友,唐守正,刘继宏,等. 微机地理信息系统 PCGIS 及其在林业资源经营管理方面的应用. 林业科学研究,1991,4(增):32～37.

2　田永林,高显连,李应国. WINGIS 的多边形内点判别算法. 林业资源管理,1994,(6):79～81.

Algorithms for Object Selection in GIS

Qiao Yanyou

Abstract　Algorithms for the selection of point, line and polygon are studied respectively in this paper. In the selection of a point, absolute distance between two points are used instead of the commonly used square root distance, and higher efficiency is achieved. For the selection of a line, a new distance between a point and a curve is defined, and higher efficiency is also achieved. In order to raise the efficiency of polygon selection, some improvements have been made to the "semiline" method which is used to judge whether a point is inside a polygon, and a new algorithm which needs less computation.

Key words　GIS, object selection, polygon

Qiao Yanyou, Assistant Professor (The Research Institute of Forest Resource Information Techniques, CAF　Beijing 100091).