一种有效的多边形缓冲区生成算法

郭　会[①②④]　宋关福[①③]　李绍俊[①③④]　周　芹[①②④]　裘　立[③]　罗飞雄[①②③]

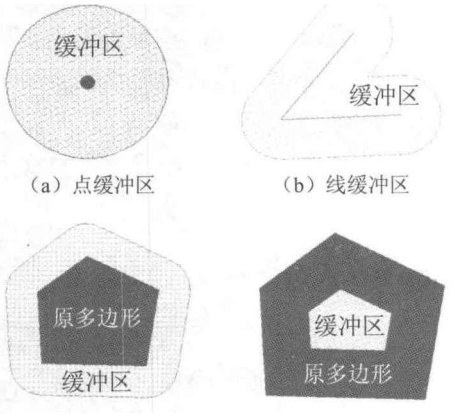
①(中国科学院地理科学与资源研究所，北京　100101)　②(中国科学院研究生院，北京　100039)
③(北京超图地理信息技术有限公司，北京　100085)

摘　要　缓冲区分析是常用的GIS临近度分析方法，而对多边形的缓冲区生成算法研究较少．本文提出了基于弧段和线段的扫描线算法(Sweep-line Algorithm-Based Arcs and Lines，SLA-BAL)的多边形膨胀和紧缩缓冲区生成算法，主要包括生成缓冲线、缓冲线求交、去除非边界线、构面等步骤，真实数据下时间复杂度为 $o[n\times\log(n)]$.

关键词　多边形　缓冲区　地理信息系统　基于弧段和线段的扫描线算法

1　引言

缓冲区是离开多边形边界、线中心线以及点和点群不超过一定距离的范围所形成的区域[1]，也被称为 Minkowski Sum[2]．由于缓冲区在地理分析和工程项目规划中简单而明确地描述了数量化的临近空间，是一个成熟的应用模型，在交通、林业、资源管理、城市规划、环境与生态保护等领域有着广泛的应用[3]．图1所示为点缓冲区、线缓冲区以及多边形的膨胀式缓冲区和多边形的紧缩式缓冲区．



(a) 点缓冲区　(b) 线缓冲区　(c) 膨胀式缓冲区　(d) 紧缩式缓冲区

图1　点、线、多边形的缓冲区

缓冲区生成算法主要分为栅格和矢量两种方法．栅格方法的算法思想较为简单，但其精度受到了限制，矢量方法可以在一定精度范围内不降低原始精度，但原理复杂、不易实现[1]．当前对缓冲区算法的研究主要集中在线缓冲区，对多边形缓冲区生成算法的研究还较少．多边形边界通常采用点串来描述．图2多边形由两个子多边形 A 和 B 构成．A 和 B 的边界由 $P_1—P_2—P_3—P_4—P_5—P_1$ 和 $P_6—P_7—P_8—P_6$ 围成，A 称为岛，B 称为洞．通常采用逆时针方向的外边界表示岛(如子多边形 A)；采用顺时针方向的外边界表示洞(如子多边形 B)．

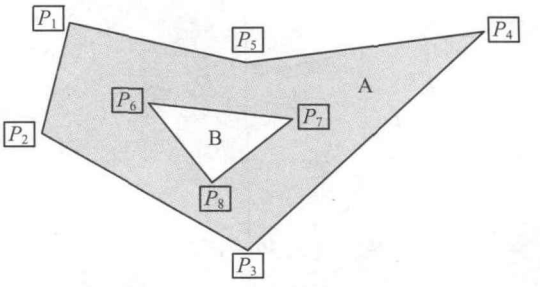


图2　多边形及其边界

2 多边形的缓冲区生成算法

多边形的缓冲区生成算法包括四个主要步骤. (1)构线：根据多边形外边界走向，构造缓冲边界线. (2)线段求交：求出(1)的边界缓冲线交点. (3)去除非边界线：在求交后，会出现一些在缓冲区内部的线段，将其去除. (4)构面：对剩余的线段首尾相接构造结果缓冲面，并判断岛洞关系. 由于面对象的膨胀式缓冲区和紧缩式缓冲区生成算法差异较小，本文除特殊说明外，均介绍膨胀式缓冲区生成算法.

2.1 构线

构线就是根据多边形的外边界点串生成多边形的缓冲区的外边界线. 构线规则为：根据多边形外边界折线走向构建线段右边缓冲线，非转角处生成距离为半径的平行线，大于180°的转角采用圆弧连接，等于180°的转角采用线段直接连接，小于180°的转角处做线段的平行线[2,3].

多边形外边界点串缓冲线具有封闭性，需要将外边界的点串向前扩展. 图2中子多边形A的外边界点串 $P_1—P_2—P_3—P_4—P_5—P_1$ 扩展为 $P_1—P_2—P_3—P_4—P_5—P_1—P_2$，位于缓冲线点串左侧的区域为缓冲区有效区域. 图2中子多边形A的缓冲线如图3所示.

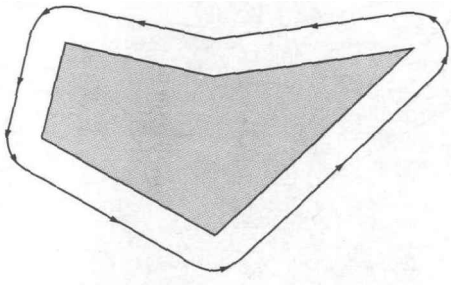


图3 构线

缓冲线包括线段和圆弧两种类型，其存储结构为：

```
struct BufferOutLine{
    //开始点
    Point  start;
    //结束点
    Point  end;
    //中心点，圆弧有效，线段为 null
    Point* pCentral;}
```

pCentral为null时表示为一条从start开始到end的一条线段；pCentral不为null时，表示以*pCentral为圆心，从start开始按照逆时针方向到end的圆弧，如图4所示.

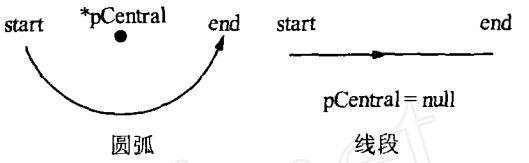


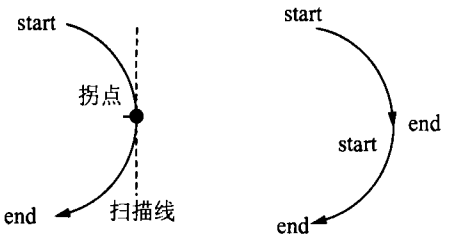
图4 缓冲线的线段和圆弧

2.2 线段求交

由于2.1的线段和弧段组成的缓冲线相交而出现多边形叠加和岛洞多边形，需要计算这些线段和弧段之间的交点. 扫描线算法(Sweep-line algorithm, SLA)是常用的求交算法[2]，基本思想是将所有线段按照 x 坐标值进行排序并生成扫描线，从左至右检测扫描线上的交点. 为了处理弧段与线段混合求交，本文采用基于弧段和线段的扫描线算法(Sweep-line Algorithm-Based Arcs and Lines, SLA-BAL). SLA-BAL在使用SLA之前先对弧段进行分割，确保弧段在扫描线方向单调递增(减)，即将弧段在拐点处进行分割并保存为多条弧段(图5). 线段与线段、弧段与弧段、弧段与线段之间求交点算法见文献[4].

2.3 去除非边界线

非边界线是指位于其他边界点串缓冲区内的某些缓冲线. 设某条缓冲线到多边形第 $k(0\leqslant k<n)$ 条边界的距离为 Δl_k，缓冲半径为 r，集合 A 为构成结果缓冲面边界线段的



（a）分割前　　（b）分割后

图 5　圆弧分割

集合，非边界线集合为 $\overline{A}$，则：

$$A=\{\forall k,\ 0\leqslant k<n,\ \Delta l_k\geqslant r\},$$

$$\overline{A}=\{x\mid x\notin A\}=\{\exists k,\ 0\leqslant k<n,\ \Delta l_k<r\}.$$

非边界线将不作为结果缓冲面的边界点串，应该去除. 为了加快去除非边界线的速度，对面对象的所有外边界线段按照 x 坐标排序，分别求出缓冲线的每条线段或弧段距离最近的外边界线段，再与缓冲半径进行比较[3].

2.4　构面

文献[2]和[3]详细介绍了构面过程. 为了提高构面的效率，先对缓冲线按照横(纵)坐标值排序，然后构建节点关系邻接矩阵. 邻接矩阵的结构为：

```
//邻接矩阵结点
struct LineSegNode{
    //开始点
    Point2D pntFromNode;
    //结束点索引
    Array<int> nArrToIndex;
    //圆心坐标(非圆弧，则为 NULL)
    Array<Point2D*> pArrPntCentral;
};
//邻接矩
struct LineSegAdjMatrix{
    //结点，按照 pntFromNode 排序
    LineSegNode * pLineSegNode;
    //结点个数
    int nNodeCount;
};
```

如果某个节点上有多个出度，则按照最大转角法选择下一条边[2]，通过射线法判断结果缓冲面的岛洞关系[2,3].

2.5　紧缩缓冲区生成算法

紧缩缓冲区是在缓冲区的边界上截取距离为半径的面环. 与膨胀缓冲区的生成算法相比，只需生成多边形边界的左侧缓冲线，算法其他步骤一致.

3　分析与实验

设面对象的线段数为 n，构面的复杂度为 $o(n)$；采用扫描线算法求交的最高复杂度为 $o(n^2)$，真实数据的复杂度为 $o(n\times\log(n))$[2]；去除非法线的最高复杂度为 $o(n^2)$，而排序后进行比较其复杂度降为 $o(n\times\log(n))$；构面时线段数量不多于 n，快排序复杂度为 $o(n\times\log(n))$，邻接矩阵扫描构面复杂度为 $o(n)$[2]，则构面整体复杂度为 $o(n\times\log(n))$. 因此，算法的最高复杂度为 $o(n^2)$，采用真实数据做缓冲区的复杂度为 $o(n\times\log(n))$.

实验的硬件平台为 P4 3.0 G、内存 1 GB，软件平台为 Windows XP、VC++8.0. 对 13 439 个坐标点串的北京市区多边形做半径为 20 km 的膨胀和紧缩缓冲区，结果如图 6 所示.

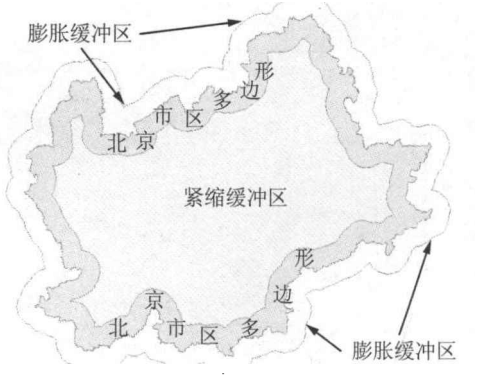


图 6　北京市缓冲区

对北京市区多边形边界点串重采样，提取边界点串数为 1 000 到 13 000 之间的多边形，进行半径为 20 km 的膨胀缓冲区分析. 图 7 为分析时间.

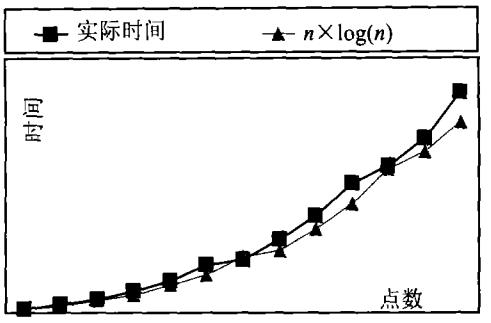


图 7　分析时间

4　总结

本文结合文献[2，3]中提出的线缓冲区生成算法，提出了多边形的膨胀缓冲区和紧缩缓冲区的生成算法，包括构线、线段求交、去除非边界线和构面等步骤．根据分析和实验表明，真实数据的算法复杂度为 $o[n\times\log(n)]$，能够有效生成多边形的缓冲区．但是，本文未考虑大量多边形生成单个缓冲区以及多边形边界点串标准化的方法，算法通用性还有待加强．

参 考 文 献

[1] 吴华意．拟三角网数据结构及其算法研究．武汉：武汉大学，1999．

[2] Borut Zalik，Mirko Zadravec，Gordon J. Clapworthy. Construction of a non-symmetric geometric buffer from a set of line segments. Computers & Geosciences，2003，29(23)：53-63.

[3] 李金山，方金云．基于平面扫描的双线圆弧缓冲区．计算机工程与应用，2007，43(23)：28-31.

[4] Middleditch，A. E.，Stacey，T. W.，Tor S. B. Intersections algorithms of lines and circles. ACM Transactions on Graphics，1989，8(1)：25-40.

[5] DONG Peng，YANG Chongjun，RUI Xiaoping，et al. An effective buffer generation method in GIS. IEEE IGRASS，2003：3706-3708.