DOI:10.13203/j.whugis2003.05.029
第28卷第5期 武汉大学学报·信息科学版 Vol. 28 No. 5
2003年10月 Geomatics and Information Science of Wuhan University Oct. 2003

文章编号:1671-8860(2003)05-0615-04 文献标识码:A

一种多边形交、并、差运算的有效算法

于雷易[1] 边馥苓[2] 万 丰[3]

(1 武汉大学遥感信息工程学院,武汉市珞喻路129号,430079)
(2 武汉大学空间信息与数字工程研究中心,武汉市珞喻路129号,430079)
(3 武汉大学计算机学院,武汉市珞喻路129号,430079)

摘 要:以周培德的Z_{54}算法为参考,提出了一种简单多边形交、并、差运算算法——IBO算法。该算法能够处理二维现实世界中的各种情况,对于地理信息系统的空间分析有较好的应用价值。

关键词:简单多边形;扫描线算法;空间关系;IBO算法;时间复杂度
中图法分类号:P208

多边形交、并、差运算是计算几何、计算机图形学和地理信息系统的空间分析领域的重要问题。然而,现有的一些算法虽然有较高的效率,却往往对参与计算的多边形作出种种限制。IBO(inner, border and outer)算法在保证运算效率的同时,获取足够充分的点和边的信息,以多边形的边与另一多边形的空间关系(内边、外边、同向重叠边和异向重叠边)为依据,进行结果多边形边界搜索,确保算法对于各种实际情况的有效性(如顶点重合、边重合)。经过实际检验,本算法具有很好的稳定性、可靠性和较高的效率。

1 算法的基本思想

多边形的边分为内边、外边、同向重叠边和异向重叠边,分别表示边在另一多边形内、边在另一多边形外、方向相同的重叠边、方向不同的重叠边。

针对多边形交、并和差,IBO算法可以分成IBO-Intersection算法、IBO-Union算法和IBO\|Difference算法。以IBO-Intersection算法为例,首先定义一个存储搜索信息的结构CStructIntersection类,此类主要存储两个多边形由顶点和交点按逆时针形成的混合点序列(本文中所有多边形的点序列均按逆时针排列),以及多边形的边被交点分割后的小块线段与另一多边形的空间关系(以LineLocation表示),这些都是必要的搜索信息。CStructIntersection类的主要成员变量如下。

1)arrayPtsMixed1, arrayPtsMixed2:两个点序列,分别存储A、B两个多边形的混合点序列(包括顶点、交点)。

2)arrayType1, arrayType2:两个整型序列,表示arrayPtsMixed1和arrayPtsMixed2中点的类型,0是顶点,1是交点,2既是顶点又是交点。

3)arrayLineLocation1, arrayLineLocation2:两个整型序列,表示arrayPtsMixed1和arrayPtsMixed2中点后边的线段类型。多边形的边上的小块线段分为内边、外边、同向重叠边和异向重叠边。这4种关系分别用LineLocation = 1、-1、2和-2表示。

IBO-Intersection算法的基本思想是,已知两个多边形的两个顶点序列,通过修改一般的扫描线算法获取对应于两个多边形的CStructIntersection对象,从第一个多边形的混合点序列中找到第一个未遍历的交点,记录该交点,作为起点和当前点。逆时针循环判断当前点后边的线段是否有LineLocation = 1或2,并且没有遍历过。如果是,则记录下一点,下一点作为起点,并继续判断;如果不是,则在另一多边形的点序列中找到该交点,并作相同的判断。循环过程中,如果回到了起始交点,则完成一个交集。然后在第一个多边形的交点序列中寻找下一个未遍历的交点,重复上述循环过程,直到遍历所有交点。

收稿日期:2003-05-26。

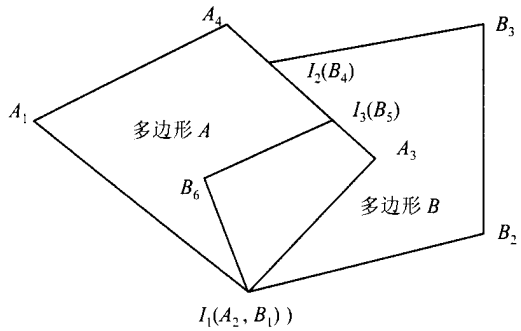

图 1 多边形交示例 1

Fig. 1 Example of Polygon Intersection 1

如图 1 所示，由扫描线算法可以得到一个 CStructIntersection 对象，各成员变量分别是 arrayPtsMixed1 { A_1, I_1, A_3, I_3, I_2, A_4 }、arrayPtsMixed2{ I_1, B_2, B_3, I_2, I_3, B_6}、arrayType1 {0, 2, 0, 1, 1, 0}、arrayType2{ 2, 0, 0, 2, 2, 0}、arrayLineLocation1{ −1, 1, 1, −2, −1, −1}、arrayLineLocation2{−1, −1, −1, −2, 1, 1}。搜索时，首先从 arrayPtsMixed1 中取出第一个交点 I_1，arrayLineLocation1 中对应的值为 1，表示 I_1 后面的边在多边形 B 内，所以取 I_1 后面的点 A_3 加入结果点序列，并作为当前点。根据基本思想的判断条件 LineLocation = 1 或 2 继续判断，可以得到多边形交集的点序列{ I_1, A_3, I_3, B_6}。

2 扫描线算法获取 CStructIntersection 对象

扫描线算法求解多条线段的交点，可以减少线段的求交次数，提高计算效率。IBO-Intersection 算法中，CStructIntersection 对象的计算即是使用扫描线算法。但由于一般的扫描线算法没有考虑一些奇异情况，如竖直线段的处理等，而且 IBO-Intersection 算法中需要获取多边形的混合点序列以及边上小段线段的 LineLocation，以便搜索交集多边形，所以 IBO-Intersection 算法中对扫描线算法作了较大的修改。

已知两个多边形的顶点序列 arrayPts1 和 arrayPts2，要求得 CStructIntersection 对象，需要使用 CStructE 和 CStructT 两个对象。其中 CStructE 主要存储事件点，即两个多边形的顶点和交点；CStructT 主要描述扫描线状态，按照从下向上的顺序存储某一事件点上与扫描线有交的线段的序列。CStructE 和 CStructT 同时要存储事件点和扫描线上线段的相关信息，并实现事件点和扫描线上线段的插入、删除和排序操作。

使用扫描线算法获取 CStructIntersection 对象的过程描述如下。

1）创建 CStructE、CStructT 和 CStructIntersection 三个对象，其中 CStructIntersection 中需要先按照顺序写入两个多边形的顶点。

2）按 x、y 坐标将两个多边形的所有顶点进行词典式分类，并将其存入 CStructE，每次取出 CStructE 中最小的点 p_i。如果 p_i 不是交点，判断 p_i 属于哪个多边形，并取出 p_{i-1}（前一点）和 p_{i+1}（后一点），得到两条线段 L_1 和 L_2；如果 p_i 是交点，获取 p_i 所对应的两条边 L_1 和 L_2。对于 L_1 和 L_2，分别作如下处理。

如果 p_i 是右端点，将 L_1 和 L_2 从 CStructT 中删除，寻求 CStructT 中线段的新的相邻关系并求交点，如果有交点，进行操作 1。如果 p_i 是左端点，将 L_1 和 L_2 插入 CStructT（对于竖直线段，下端点为左，上端点为右），寻求 CStructT 中线段的新的相邻关系并求交点，如果有交点，进行操作 1。使用 CStructT::Insert()插入线段时根据定理 1 确定插入点右边的小段线段的 LineLocation。如果 p_i 是交点，交换 L_1 和 L_2 在 CStructT 中的顺序，寻求 CStructT 中线段的新的相邻关系并求交点，如果有交点，进行操作 1。

操作 1：多边形的交点分为三种情况，即一般交点、交于端点处的交点（即端点交点，不包括重合线段的端点）和重合线段的端点。根据交点类型，确定交点是否需要插入到 CStructE 中，插入到多边形的两个混合点序列中，并且刷新被交点、顶点所分割的各线段的 LineLocation。求解 LineLocation 时，可以使用定理 1。

定理 1：使用扫描线算法对两个简单多边形的边进行求交过程中，插入线段到 CStructT 中之后，如果此线段在插入点之后的部分不与其他线段重合，则累计 CStructT 中位于此线段下边的另一多边形边的条数（竖直线段不计），如果条数为偶数，则此线段在另一多边形外，LineLocation=1；如果条数为奇数，则此线段在另一多边形内，LineLocation=−1。此定理的原理与计算机图形学中区域填充的有序边算法相同。

3 边界搜索

定理 2：CStructIntersection 对象中 LineLocation =1 的线段的左域是两个多边形的内域，右域是本多边形（目标线段所在多边形为本多边形，另一多边形为次多边形）的外域，次多边形的内域；

LineLocation = −1 的线段的左域是本多边形的内域, 次多边形的外域, 右域是两个多边形的外域; LineLocation=2 的线段的左域是两个多边形的内域, 右域是两个多边形的外域; LineLocation = −2 的线段的左域是本多边形的内域, 次多边形的外域, 右域是本多边形的外域, 次多边形的内域。

证明: 因为两个多边形的顶点都按逆时针排列, 所以 CStructIntersection 对象中任一线段的左域必然是本多边形内域, 右域必然是本多边形外域。其中 LineLocation=1 的线段由于是次多边形的内边, 所以其左域和右域也必然是次多边形内域; LineLocation = −1 的线段由于是次多边形的外边, 所以其左域和右域也必然是次多边形外域; LineLocation = 2 的线段是同向重叠边, 线段的左域显然同是内域, 右域显然同是外域; LineLocation = −2 的线段是异向重叠边, 线段的左域显然是次多边形的外域, 右域显然是次多边形的内域。

由定理 2 可知, 按照前面所述 IBO-Intersection 算法的基本思想计算所得的多边形必然是两个多边形的交集。根据基本思想, IBO-Intersection 算法的边界搜索如下。

1)根据 CStructIntersection 对象中的交点和线段的 LineLocation, 可以判断出多边形的空间关系。对于 Disjoint、Touches、Equals、Contains、Within 等 5 种关系, 可以直接获得交集。

2)对于 Overlaps, 从多边形 A 的混合点序列中找到第一个交点, 记录该交点, 作为当前点和起点, 从起点开始在多边形两个混合点序列中逆时针搜索。循环判断当前点后边的线段 L 的 LineLocation。如果 (LineLocation = 1 || LineLocation = 2) && L 没有遍历过, 记录下一点并继续判断; 否则, 在另一多边形的混合点序列中找到该交点, 并作相同的判断。如果(LineLocation = 1 || LineLocation = 2) && L 没有遍历过, 记录下一点继续判断; 否则终止此次循环, 当前循环的起点是独立交点或独立重合线段的端点。

循环过程中, 如果回到了起始交点, 则对所得点序列组成交集多边形。然后在多边形 A 的混合点序列中寻找下一个未遍历的交点, 重复上述循环过程, 直到遍历所有交点。

另外, 为了提高算法效率, 可以使用边界盒方法, 在算法开始之前对多边形的关系进行粗判。

由定理 2 同样可以推导出 IBO-Union 算法和 IBO-Difference 算法。它们的基本原理与 IBO-Intersection 算法的基本原理相同, 首先要使用改进的扫描线算法获取 CStructIntersection 对象, 只是搜索结果多边形边界的条件略有不同。

4　算法分析

IBO 算法使用扫描线算法减少多边形的边求交的次数, 在扫描过程中同时获取必要的边界搜索信息, 边界搜索时的判断条件十分简单, 所以算法的效率很好。假设两个多边形 A 和 B 的顶点数均为 n, 则它们的交点数最多是 n^2, 那么扫描线算法的每次循环时间需要 $O(\lg n)$, 最多执行 (n^2+2n)次, 共需要 $O(n^2\lg n)$, 而搜索结果多边形的时间耗费不会超过 $O(n^2)$。所以 IBO 算法的时间复杂度最差是 $O(n^2\lg n)$, 与周培德的 $Z_{5\text{-}4}$算法相同。以图 2 为例, IBO 算法中判断是否求交的循环次数是 9 次, 求交次数是 3 次, 搜索边界时判断次数是 5 次, 而 $Z_{5\text{-}4}$ 算法也是如此。不同之处在于 IBO 算法在求交时需要判断交点类型, 求交之后需要将交点之后的小段线段的 LineLocation 变反; 而 $Z_{5\text{-}4}$算法在边界搜索时需要计算线段端点与另一多边形的空间关系。

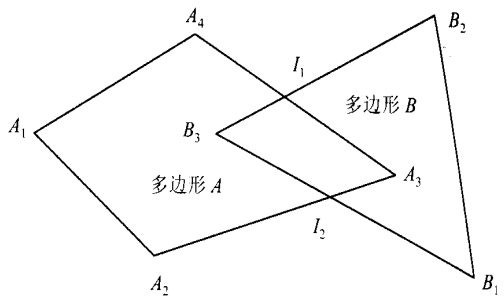

图 2　多边形交示例 2

Fig. 2　Example of Polygon Intersection 2

由于 IBO 算法在扫描线扫描过程中获取了各小段线段与另一多边形的空间关系, 这种空间关系所包含的信息比 $Z_{5\text{-}4}$ 算法单纯的交点信息丰富, 所以在遇到重合交点和重合线段时仍然有效。如图 1 所示, IBO 算法能够正确搜索到交集。$Z_{5\text{-}4}$ 算法的边界搜索方法是从某交点出发, 按逆时针方向沿多边形边(该边前进方向上另一端点在另一多边形内部或该边前进方向上有偶数个交点)交替前进, 直至回到起始交点。对于图 1 的示例, 如果从 I_3出发, 则搜索到 I_1时, 沿多边形 B 的前进方向上有 2 个交点(B_1B_2同时与 A_1A_2和 A_2A_3 相交), 满足前进条件, 继续沿多边形 B 搜索, 显然这是错误的。同样, 如果从 I_1出发, 搜索到 I_3 时, 也会发生错误。可见 $Z_{5\text{-}4}$算法的边界搜索信息在遇到重合交点和重合线段时是不足的。

参 考 文 献

1 周培德. 计算几何——算法分析与设计. 北京: 清华大学出版社, 2000

2 Zalik B. Two Efficient Algorithms for Determining Intersection Points Between Simple Polygons. Computers & Geoscience, 2000, 26(2): 137~151

3 Rivero M, Feito F R. Boolean Operations on General Plannar Polygons. Computers & Graphics, 2000, 24(6): 881~896

4 O' Rourke J. Computational Geometry in C. Cambridge: Cambridge University Press, 1993

第一作者简介: 于雷易, 博士生。从事地理信息系统方面的研究。E-mail: yuleiyi@163.net.

An Efficient Algorithm for Intersection, Union and Difference Between Polygons

YU Leiyi[1] *BIAN Fuling*[2] *WAN Feng*[3]

(1 School of Remote Sensing and Information Engineering, Wuhan University, 129 Luoyu Road, Wuhan, China, 430079)

(2 Research Center of Spatial Information and Digital Engineering, Wuhan University, 129 Luoyu Road, Wuhan, China, 430079)

(3 School of Computer, Wuhan University, 129 Luoyu Road, Wuhan, China, 430079)

Abstract: By the reference of the Z5-4 algorithm developed by Zhou Peide, the paper presents an algorithm for intersection, union and difference between simple polygons, which is named IBO algorithm. In the algorithm, the result polygon is tracked according to the spatial relation between the edge of one polygon and another polygon. The algorithm is valid for all the cases in the real two-dimensional world and has high efficiency. It is applicable in spatial analysis of geographic information system.

Key words: simple polygon; sweep-line approach; spatial relation; IBO algorithm; time complexity

About the first author: YU Leiyi, Ph. D candidate. His major research orientation is geographic information system. E-mail: yuleiyi@163.net.

(责任编辑: 涓涓)