

# 基于交点有序化的简单多边形布尔运算

魏胜利,李 源

(安阳工学院 计算机科学与信息工程学院,河南 安阳 455000)

**摘 要:**在分析现有算法的基础上,提出了一种基于交点有序化的简单多边形布尔运算算法。该算法以循环单链表数据结构存储多边形顶点和交点,在交点按顺序插入到多边形链表环节提出基点的概念。对于采用时间复杂度为  $O(n+k)\log^m$  的算法所求出无序多边形交点,以邻接表暂存这些交点,把具有相同基点的交点按交点到基点的距离从小到大排序以实现无序交点的有序化,然后通过一次遍历邻接表把交点依次插入到多边形链表中。在循环单链表中,主多边形和裁剪多边形共享同一个交点,以哈希表存储交点的地址,以提高查找效率。根据多边形顶点的进出性追踪多边形的交、并、差。最后对算法进行了编程实现并与其他同类算法进行了比较,结果表明该算法具有更高的执行效率。

**关键词:**布尔运算;多边形;基点;交点;计算几何

**中图分类号:**TP391

**文献标识码:**A

**文章编号:**1673-629X(2019)08-0081-05

**doi:**10.3969/j.issn.1673-629X.2019.08.016

## A Simple Polygon Boolean Operation Based on Sorted Intersection Points

WEI Sheng-li, LI Yuan

(School of Computer Science & Information Engineering, Anyang Institute of Technology, Anyang 455000, China)

**Abstract:**Based on the analysis of existing algorithms, a simple polygon Boolean algorithm based on sorted intersection points is presented. The algorithm stores the vertices and intersection points of polygons in a cyclic single-linked list data structure. The concept of base points is presented at the intersection points inserted sequentially into the polygonal linked list links. For the disorder intersection points of polygons with  $O(n+k)\log^m$  time complexity, the adjacent tables is used to store these intersection points temporarily. The intersection points with the same basis point from small to large by the distance of intersection point to basis point are sorted, so that the disorder intersections will be ordered. The intersection points are inserted into the lists of the polygons sequentially by traversing through the adjacent tables once. In the circular single linked list, the main polygon and the clipping polygon share the same intersection points, the address of the intersection points is stored in a hash table to improve the search efficiency. In terms of the entry and exit of polygon vertices the intersection, union and difference of the polygons are traced. Finally, the algorithm is programmed and compared with other similar algorithms, which shows that it has higher efficiency.

**Key words:** Boolean operations; polygon; basis point; intersection point; computational geometry

## 0 引 言

多边形布尔运算,即多边形间的交、并和差,是计算机图形学、计算几何中一个最基本的算法,广泛应用于几何实体造型、地理信息系统(GIS)等领域。国内外许多专家对多边形布尔运算算法进行了大量研究,并提出了相应的算法。Andereev<sup>[1]</sup>、Foley<sup>[2]</sup>、Maillot<sup>[3]</sup>等提出的算法仅适用于凸多边形。Weiler等<sup>[4]</sup>提出的算法使用树形数据结构,Vatti<sup>[5]</sup>、Greiner\_Hormann<sup>[6]</sup>提出的算法使用双性链表数据结构,刘勇奎<sup>[7]</sup>提出的

算法使用单链表结构。朱雅音等<sup>[8]</sup>提出了基于边的奇偶性质的任意简单多边形的布尔运算,很好地处理了各种奇异情况。闫浩文等<sup>[9]</sup>提出一种复合多边形求差的矢量算法。崔璨等<sup>[10]</sup>提出一种基于梯形剖分的多边形布尔运算方法。Rivero等<sup>[11]</sup>提出一种全新的适用于任意多边形的布尔运算,该算法用简单链表示两个多边形,得到一个同样用简单链表示的结果多边形;董未名等<sup>[12]</sup>将Rivero算法扩展到可以应用于带圆锥曲线边的平面简单多边形。朱二喜等<sup>[13]</sup>从图形内角

收稿日期:2018-10-16

修回日期:2019-02-18

网络出版时间:2019-03-28

基金项目:河南省科技攻关项目(162102210130)

作者简介:魏胜利(1974-),男,硕士,副教授,研究方向为计算机图形、数控插补算法等。

网络出版地址:<http://kns.cnki.net/kcms/detail/61.1450.TP.20190327.1633.078.html>

概念开始,对多边形布尔运算重新全局化建模,实现其算法。赵军等<sup>[14-15]</sup>利用最小回路提出分别求简单和带洞多边形的布尔运算的方法。

文中在对 Greiner\_Hormann 和刘永奎提出的算法深入分析的基础上,提出了一种基于交点有序化的简单多边形布尔运算算法。以单链表数据结构为多边形的存储结构,采用基于扫描线的线段求交算法求多边形的交点,借助邻接表实现对无序交点的有序化处理,通过一次性遍历邻接表把交点依次插入到多边形链表中,然后再分别追踪出多边形交、并、差的结果。最后将该算法与 Greiner\_Hormann 和刘永奎算法进行了比较,结果表明该算法在执行效率方面优于上述两种算法。

## 1 基本概念与定义

为了便于算法的描述,先介绍一些有关多边形布尔运算的基本概念和术语。

### (1) 主多边形和裁剪多边形。

多边形的布尔运算包括交、并、差,多边形的布尔运算与多边形的裁剪运算没有本质的区别,都需要先对数据进行处理,才能追踪结果。裁剪运算只是得到多边形的交,布尔运算还需要后续输出并和差的结果。因此,为了便于问题的描述,借鉴裁剪运算的概念,把其中一个多边形称为主多边形,用  $S$  表示;另一个多边形称为裁剪多边形,用  $C$  表示。

### (2) 多边形布尔运算的表示。

对于多边形  $S$  和  $C$ ,定义  $S$  与  $C$  的交表示为  $S \cap C$ ,  $S$  与  $C$  的并表示为  $S \cup C$ ,  $S$  与  $C$  的差表示为  $S - C$ ,  $C$  与  $S$  的差表示为  $C - S$ 。

### (3) 多边形边的方向与内外区域的关系。

如果多边形边的方向是逆时针,则沿着多边形边界方向前进,左侧区域为多边形的内部;相反如果多边形边的方向是顺时针,则沿着多边形边界方向前进,右侧区域为多边形的内部。如无特别说明,这里所述的多边形都是按逆时针方向存储的。

### (4) 进点和出点的定义。

设  $I$  是多边形  $S$  和  $C$  的一个交点,如果沿着  $S$  的边界方向在  $I$  点从  $C$  的外部进入  $C$  的内部,则称  $I$  为  $C$  的一个进点。反之,如果  $S$  在  $I$  点从  $C$  的内部到  $C$  的外部,则称  $I$  为  $C$  的一个出点。

例如,对于图 1 所示的多边形  $S$  和  $C$  及其交点  $I$ ,若  $S$  的边为逆时针方向,则  $I_3$ 、 $I_1$  为  $C$  的进点,  $I_2$ 、 $I_4$  为  $C$  的出点;若  $S$  的边为顺时针方向,则  $I_3$ 、 $I_1$  为  $C$  的出点,  $I_2$ 、 $I_4$  为  $C$  的进点。

### (5) 基点。

所谓基点就是按照多边形边的方向交点所在边的

起点。例如,在图 1 中,对于多边形  $C$  来说,  $C_1$  就是  $I_4$  和  $I_3$  的基点。

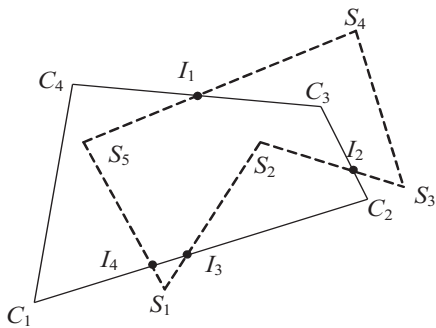


图 1 相交多边形

## 2 算法的数据结构

多边形布尔运算算法需要一个合适的数据结构来存储多边形及交点,并能够在其上进行正确追踪布尔运算的操作。Weiler 算法中把输入的多边形组成一个树形结构, Greiner\_Hormann 算法采用双向链表的结构,每个多边形由一个双向链表来表示,并依此把所求得的交点分别按多边形顶点的有序序列插入到主多边形和裁剪多边形的两个双向链表中。刘勇奎对 Greiner\_Hormann 算法进行了改进,采用循环单链表来存储多边形,同时仅为交点分配一个存储空间,与 Greiner\_Hormann 中的双向链表结构相比,因减少了一个指针域而节省了存储空间,且仅为每个交点建立 1 个节点的存储空间,并分别插入到两个多边形的单链表中。而 Greiner\_Hormann 算法为每个交点建立两个存储空间,然后再分别将每个交点插入到对应的多边形链表中。刘勇奎算法设计了两种不同的数据结构分别存储多边形顶点信息和交点信息,但又把以这两种不同结构表示的多边形顶点和交点插入到同一个链表中,这在采用带有指针类型的高级语言编程时是无法实现的,因为这类编程语言要求链表中的节点必须具有相同类型的结构。可以把刘勇奎算法中所设计的两种不同的数据结构归结为一类,即每个单链表节点中含有两个指针,其中一个指针只有当该节点为交点时才有意义。这样既保留了刘勇奎算法中单链表的优点,又便于编程实现。

在基于交点有序化的简单多边形布尔运算算法中,单链表节点数据结构为:

```
点坐标信息:
struct Point
{
    double x,y    //点坐标
};
```

多边形顶点及后续求的交点坐标都存储到一个点坐标数组中,点的编号即为点在数组中的下标。

单链表节点信息:

```
struct Vertex
{
    bool Intersect; //布尔型
    Vertex * next1, * next2; //节点指针
    bool used; //布尔型
    int index; //点编号
};
```

其中 Intersect 用于标识该节点是多边形顶点还是

交点,指针域用于指向下一个节点,如果该节点是多边形顶点,则 next1 指向单链表的后继,next2 无意义;如果该节点为交点,则 next1,next2 分别指向主多边形和裁剪多边形单链表的后继节点,实现同一个交点分别插入到两个多边形单链表中。used 用来标识在追踪多边形布尔运算过程中该节点是否使用过。index 为该顶点在点坐标数组中的索引值。图 2 为对图 1 采用文中算法得到的的数据存储结构。

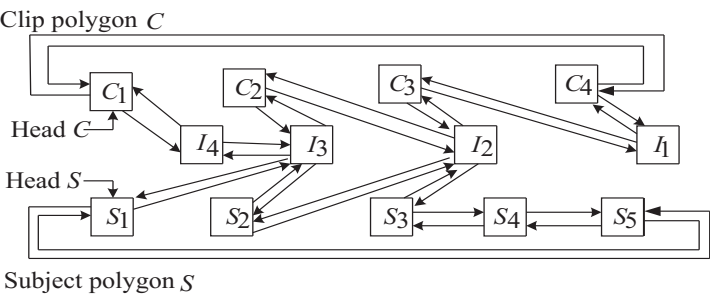


图 2 数据存储结构

3 无序交点的有序化

多边形求交是影响多边形布尔运算效率的关键因素,Greiner\_Hormann 算法和刘勇奎算法都是通过测试两个多边形的全部线段对求交点,效率很低。文中采用 Park<sup>[16]</sup>提出的多边形链求交算法求多边形的交点,其时间复杂度为  $O(n + k) \log^m$ ,其中  $m$  为判断是否存在交点的线段数, $n$  为两个多边形的顶点个数, $k$  为交点个数。但 Park 算法求得的多边形链的交点是无序的。文中采用与邻接表类似的存储结构分别暂时存储主多边形和裁剪多边形的交点信息,以实现无序交点的有序化。在邻接表中,为多边形的每个顶点建立一个单链表,每个单链表中的节点存储以该顶点为基点对应的交点信息。每个单链表上附设一个表头节点,在表头节点中,除了设有指针域指向链表中第一个节点,还设有存储基点编号的信息域。建立两个数组,分别存储主多边形和裁剪多边形的表头节点。

其中,firstpoint 域指向该基点的第一个交点,next 域指向该基点所对应的下一个交点,distancetobasepoint 为交点到基点的距离,每个基点对应的交点按 distancetobasepoint 从小到大排序,这样就实现了无序交点的有序化,且有序化后的交点顺序与多边形顶点的存储顺序一致。

设图 1 中主多边形的顶点从  $S_1$  至  $S_4$  的编号分别为 1 到 5,裁剪多边形的顶点从  $C_1$  至  $C_4$  的编号分别为 6 到 9,则主多边形  $S$  上的交点有序化后的存储信息如图 3(a)所示,裁剪多边形  $C$  上的交点有序化后的存储信息如图 3(b)所示。

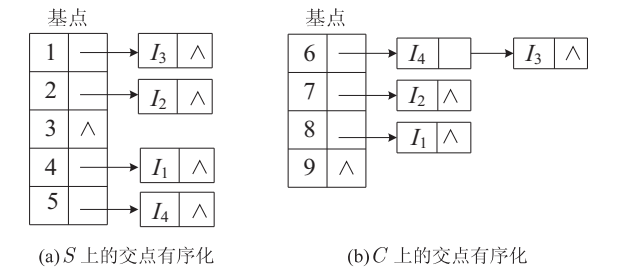


图 3 多边形交点有序化

4 交点插入多边形链表

完成无序交点的有序化后,下一步需要把交点按多边形顶点的顺序插入到多边形链表中。把交点插入到多边形链表时需要设置交点的进出性,分析发现沿着  $S$  的边界,对于  $C$  的进点和出点是交替出现的,所以只需判断第一个交点是进点还是出点,其他交点的进出性则可依次确定。确定第一个交点的进出性的方法是,在  $S$  的交点信息表中,找到第一个交点  $I$ ,并判断  $I$  的基点  $S_i$  与多边形  $C$  的位置关系,如果  $S_i$  在  $C$  的外

存储交点的单链表节点的结构为:

```
struct IntersectPoint
{
    double distancetobasepoint; //交点到基点的距离
    IntersectPoint * next; //指针
    int index; //点编号
};
```

表头节点的结构为:

```
struct BaseHead
{
    int index; //基点编号
    IntersectPoint * firstpoint; //指向第一个交点
};
```

部,则  $I$  为  $C$  的进点,反之  $I$  为  $C$  的出点。

把交点插入到  $S$  多边形链表中,先遍历  $S$  的每个交点,并在  $S$  的多边形链表中找到该交点的基点在多边形链表中的位置即可把交点按照多边形链表节点结构的形式插入到  $S$  的多边形链表中。由于与  $S$  对应的有序化后的交点顺序与  $S$  的顶点顺序是一致的,因此在查找下一个交点的基点时可在多边形链表中当前交点的后继节点中查找,而不需重新遍历链表。在插入交点的过程中,可依据第一个交点的进出性交替设置其他交点的进出性。

按照文中采用的多边形存储结构,主多边形链表和裁剪多边形链表共享同一个交点存储单元。由于在把交点插入到  $S$  多边形链表时已经为交点分配了存储空间,所以在插入  $C$  的交点到  $C$  的多边形链表时只需修改指针的指向关系即可。但是  $S$  多边形链表中有序交点对于  $C$  多边形又是无序的。为了在常数时间内查找到每个交点的存储地址,可用哈希表存储每个交点的存储地址。在该哈希表中以交点的编号为关键字,交点的个数为哈希表的长度,哈希函数采用除留余数法,即:

$$H(\text{key}) = (\text{xindex} - \text{index}_{\min}) \text{MOD } p$$

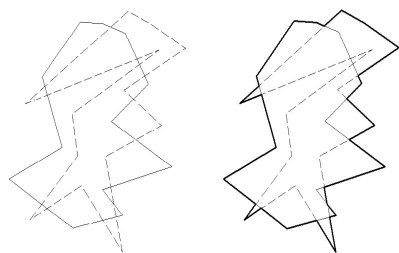
其中,  $\text{xindex}$  为交点编号;  $\text{index}_{\min}$  为交点编号的最小值;  $p$  为哈希表长度。每个交点经过哈希函数处理后都有一个唯一的值,所以不需要设计处理冲突的方法。

采用与  $S$  的交点插入的  $S$  的多边形链表中类似的方式,找到  $C$  多边形每个交点的基点在  $C$  多边形链表中的位置,并按所设计的哈希函数在常数时间内定位到交点的存储地址,修改指针的指向关系,即可将交点插入到  $C$  多边形链表中。这样把全部交点插入到多边形链表的时间复杂度为  $O(n + 2k)$ 。

## 5 追踪多边形布尔运算

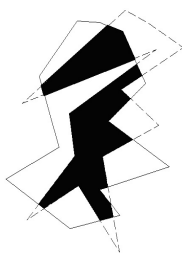
把交点按照多边形顶点的存储顺序分别插入到多边形链表后即可追踪多边形布尔运算的结果。

### (1) 追踪多边形并。

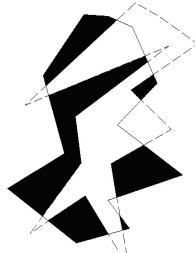


(a)  $S$  和  $C$

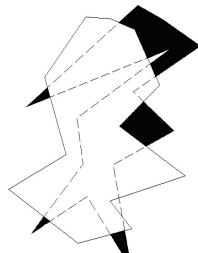
(b)  $S \cup C$



(c)  $S \cap C$



(d)  $S - C$



(e)  $C - S$

图 4 多边形布尔运算结果

追踪多边形并的步骤为:在主多边形  $S$  中查找第一个其后继为裁剪多边形进点的节点  $p_s$ ,沿  $S$  的多边形链表,从  $p_s$  开始逆时针遍历,当遇到交点时,则转到裁剪多边形  $C$  上继续逆时针遍历,遇到交点时,再次转到多边形  $S$  上继续逆时针遍历,重复以上步骤直至起点  $p_s$ ,追踪多边形并结束。从  $p_s$  开始到  $p_s$  结束按上述过程遍历到点集就构成多边形的并。

### (2) 追踪多边形交。

追踪多边形交的步骤为:在主多边形  $S$  中查找第一个为裁剪多边形进点的交点  $p_s$ ,将其标记为已访问,然后转到裁剪多边形  $C$  上逆时针遍历,遇到交点时,再次转到多边形  $S$  上继续逆时针遍历,重复以上步骤直至起点  $p_s$ 。追踪出多边形交的一部分。在上述过程中,每遍历到一个节点都标记该点为已访问。在主多边形  $S$  中查找下一个为裁剪多边形进点且尚未访问的交点,重复以上过程,直至所有交点都被访问,最后各个部分组合在一起即为多边形的并。

### (3) 追踪多边形差。

求多边形差与多边形的并和交略有不同,如果求  $S - C$  则要求裁剪多边形  $C$  顶点按顺时针存储。追踪多边形  $S - C$  的步骤为:在主多边形  $S$  中查找第一个其后继为裁剪多边形进点的节点  $p_s$ ,将其标记为已访问,沿  $S$  的多边形链表,从  $p_s$  开始逆时针遍历,当遇到交点时,则转到裁剪多边形  $C$  上继续顺时针遍历,遇到交点时,再次转到多边形  $S$  上继续逆时针遍历,重复以上步骤直至起点  $p_s$ ,追踪出多边形  $S - C$  差的一部分。在上述过程中,每遍历到一个节点都标记该点已访问。在主多边形  $S$  中查找下一个其后继为裁剪多边形进点且尚未被访问的节点,重复以上过程,直至所有交点都被访问,最后各个部分组合在一起即为多边形  $S - C$ 。追踪多边形  $C - S$  则只需把上述主多边形和裁剪多边形角色互换即可。

## 6 实验结果

对文中提出的求多边形布尔运算的算法在 VC++ 环境下进行编程实现,结果如图 4 所示。

表 1 为相同的数据组在相同的硬软件环境下文中 对比的结果。  
算法分别与 Greiner\_Hormann 算法和刘勇奎算法进行

表 1 不同算法实验所需的时间比较				s
<i>N</i>	Greiner_Hormann 算法	刘勇奎算法	文中算法	
10	0.151	0.134	0.125	
30	2.684	2.014	1.528	
50	10.873	8.973	6.821	

从表 1 中可知,文中算法在求多边形布尔运算(交、并、差)时优于 Greiner\_Hormann 算法和刘勇奎算法。文中算法在求多边形布尔运算过程中,时间复杂度为  $O(n + k) \log^m$  的 Park 算法求多边形的交点,对于所求得的无序交点,在插入交点到多边形链表前先以邻接矩阵的形式存储多边形交点,并把相同基点的交点按交点到基点的距离从小到大排序,实现无序交点的有序化。然后再依次把交点插入到多边形链表中,把交点插入到多边形链表中的时间复杂度为  $O(n + 2k)$ 。实验结果表明,该算法显著提高了多边形布尔运算的执行效率。

### 7 结束语

对于简单多边形的布尔运算,在分析已有研究的基础上,提出了一种基于交点有序化的简单多边形布尔运算算法。借助邻接矩阵存储多边形交点,并把相同基点的交点按交点到基点的距离从小到大排序以实现无序交点的有序化,并根据顶点的进出性追踪多边形的布尔运算结果。最后该算法进行编程实现,并和 Greiner\_Hormann 算法和刘勇奎算法进行了对比,结果证明了该算法的高效性和可行性。

#### 参考文献:

[1] ANDEREEV R D. Algorithm for clipping arbitrary polygons [J]. Computer Graphics Forum,1989,8(2):183-191.  
[2] FOLEY J D,DAM A,FEINER S K,et al. Computer graphics, principles and practice, reading [M]. MA: Addison-Wesley, 1990.  
[3] MAILLOT P G. A new, fast method for 2D polygon clipping: analysis and software implementation [J]. ACM Transactions

on Graphics,1992,11(3):276-290.  
[4] WEILER K, ATHERTON P. Hidden surface removal using polygon area sorting [J]. ACM SIGGRAPH Computer Graphics,1977,11(2):214-222.  
[5] VATTI B R. A generic solution to polygon clipping [J]. Communications of the ACM,1992,35(7):56-63.  
[6] GREINER G, HORMANN K. Efficient clipping of arbitrary polygons [J]. ACM Transactions on Graphics,1998,17(2):71-83.  
[7] 刘勇奎,高云,黄有群. 一个有效的多边形裁剪算法 [J]. 软件学报,2003,14(4):845-856.  
[8] 朱雅音,王化文,万丰,等. 确定两个任意简单多边形交、并、差的算法 [J]. 计算机研究与发展,2003,40(4):576-583.  
[9] 闫浩文,张黎明,李茜茜,等. 复合多边形求差的高效矢量算法 [J]. 计算机应用研究,2013,30(10):3192-3194.  
[10] 崔璨,王结臣. 一个基于梯形剖分的多边形布尔运算方法 [J]. 测绘学报,2011,40(1):104-110.  
[11] RIVERO M,FEITO F R. Boolean operations on general planar polygons [J]. Computer & Graphics,2000,24(6):881-896.  
[12] 董未名,玛依拉·巴榜,周登文,等. 平面扩展简单多边形的布尔运算 [J]. 计算机辅助设计与图形学学报,2003,15(9):1134-1140.  
[13] 朱二喜,何援军. 一种利用图形内角的多边形布尔运算新算法 [J]. 工程图学学报,2011,32(2):10-19.  
[14] 赵军,刘荣珍. 用最小回路求两个简单多边形的交、并、差集 [J]. 计算机应用,2012,32(11):3164-3167.  
[15] 赵军,郝永兴. 一种求含孔洞多边形交、并、差集的新方法 [J]. 图学学报,2014,35(4):498-503.  
[16] PARK S C, SHIN H. Polygonal chain intersection [J]. Computer & Graphics,2002,26(2):341-350.