任意多边形裁剪算法研究

范华燃, 刘强, 罗宗铭, 李辰辉, 杨杰

(成都理工大学, 空间信息技术系)

摘要:多边形裁剪算法被广泛应用于空间叠置分析、图像处理等各个领域，具有非常大的价值和研究意义。在多边形裁剪的诸多算法中，主要有Sutherland-Hodman算法,梁友栋-Braian算法等，但是这些算法在裁剪多边形后均不能消除多余边和边的自身相交。梁－Braian算法针对这一问题作了解决，但该算法需作很多的乘除法，不利于计算机内部实现。因此本文主要学习研究了一种基于出入点判断的任意多边形裁剪算法，即Weiler-Atherton裁剪算法，该算法利于快速简洁、计算机实现，且具有一般性。

关键词: 多边形裁剪、任意多边形、矢量数组、时间复杂度

Arbitrary Polygon Clipping Algorithms

Fan HuaRan, Liu Qiang, Luo Zongming, Li ChengHui, Yang Jie

(Department of Spatial Information Technology, Chengdu University of Technology)

Abstract: Polygon clipping algorithms are widely used in space superimposed analysis, image processing and other fields, has great value and research significance. In a lot of algorithm, polygon clipping Sutherland - Hodman algorithm, main Liang Youdong - Braian algorithm, etc., but these algorithms are not eliminate redundant after cutting polygon edge and the edge of the intersection. Beam - Braian algorithm to solve the problem in order to solve this problem, but the algorithm requires a lot of, method, is not conducive to internal computer implementation. So this paper mainly studies a kind of arbitrary polygon clipping algorithm based on the access point judgement, namely Weiler Atherton - cutting algorithm, the algorithm for fast computer .

Keywords: Polygon clipping、vector、Arbitrary polygon、Time complexity

1 引言

原有的多边形裁剪算法，例如Sutherland-Hodman算法，梁友栋－-Braian算法，在裁剪多边形后均不能消除多余边和边的自身相交。Weiler-Atherton裁剪算法基于上述问题提出了一种新的分类：入点、不出点，并由此提出一种快速的多边形裁剪算法。该算法裁剪思想新颖，方法简洁，裁剪一次完成，从而提高了多边形的生成速度。并消除了多余边和自身边相交的问题，从而可以适用于凸的、凹的和带孔的多边形裁剪过程，有较高的研究价值和应用价值。

2 方法

首先假设被裁剪多边形和裁剪窗口的顶点序列都按顺时针方向排列。当两个多边形相交时，交点必然成对出现，其中一个是从被裁剪多边形进入裁剪窗口的交点，称为“入点”，另一个是从被裁剪多边形离开裁剪窗口的交点，称为“出点”。

算法从被裁剪多边形的一个入点开始，碰到入点，沿着被裁剪多边形按顺时针方向搜集顶点序列；而当遇到出点时，则沿着裁剪窗口按顺时针方向搜集顶点序列。

按上述规则，如此交替地沿着两个多边形的边线行进，直到回到起始点。这时，收集到的全部顶点序列就是裁剪所得的一个多边形。

由于可能存在分裂的多边形，因此算法要考虑：将搜集过的入点的入点记号删去，以免重复跟踪。将所有的入点搜集完毕后算法结束。

算法流程：

1、顺时针输入被裁剪多边形顶点序列Ⅰ放入数组1中。

2、顺时针输入裁剪窗口顶点序列Ⅱ放入数组2中。

3、求出被裁剪多边形和裁剪窗口相交的所有交点，并给每个交点打上“入”、“出”标记。然后将交点按顺序插入序列Ⅰ得到新的顶点序列Ⅲ，并放入数组3中，同样也将交点按顺序插入序列Ⅱ得到新的顶点序列Ⅳ，放入数组4中；

4、初始化输出数组Q，令数组Q为空。接着从数组3中寻找“入”点。 如果“入”点没找到，程序结束。  
 　5、如果找到“入”点，则将“入”点放入S中暂存。

6、将“入”点录入到输出数组Q中。并从数组3中将该“入”点的“入”点标记删去。

7、在数组3中从该入点之后顺序取顶点（循环的过程，并不是搜寻到末尾就结束，那时候要回退到第一个顶点搜寻）： 如果顶点不是“出点”，则将顶点录入到输出数组Q中。 否则，流程转第8步。

8、在数组4中从该出点之后顺序取顶点（循环的过程，并不是搜寻到末尾就结束，那时候要回退到第一个顶点搜寻）： 如果顶点不是“入点”，则将顶点录入到输出数组Q中，否则，流程转第9步。

9、如果顶点不等于起始点S，流程转第6步，继续跟踪数组3。否则，将数组Q输出，流程转第4步，寻找可能存在的分裂多边形。

算法在第4步：满足“入”点没找到的条件时，算法结束。

算法流程图如下：

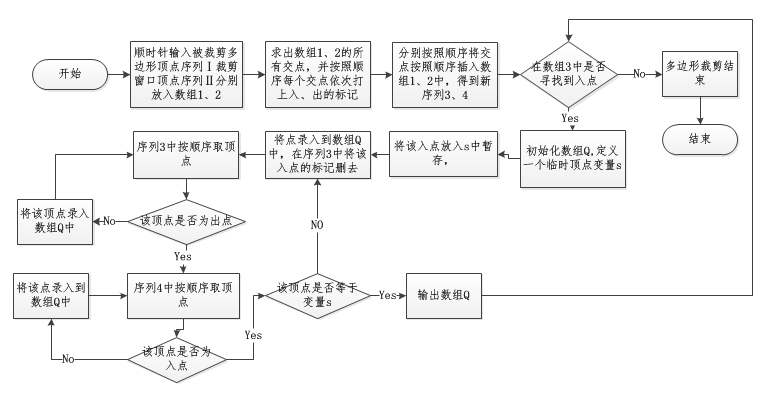


图 1 Weiler-Atherton流程图

针对该算法设计如下数据结构:

//点结构体设计

struct WeilerPoint

{

float x;

float y;

//0为普通顶点 1为入点 2为出点 3为交点待确定出入点标志

int pointFlag;

//当为交点时所在原裁剪多边形的点索引

int clippedIndex;

//当为交点时所在原裁剪窗口的点索引

int clippingIndex;

WeilerPoint(float x1,float y1)

{

x=x1;

y=y1;

pointFlag=0;

}

WeilerPoint(float x1,float y1,int \_flag)

{

x=x1;

y=y1;

pointFlag=\_flag;

}

WeilerPoint CopyPoint()

{

WeilerPoint point(x,y,pointFlag);

return point;

}

WeilerPoint()

{

}

};

//多边形结构体设计

struct WeilerPolygon

{

vector<WeilerPoint> points;

// WeilerPolygon()

};

//裁剪结果多边形结构体设计

struct WeilerGenPolygon

{

vector<WeilerPolygon> polygons;

};

算法核心代码:

WeilerGenPolygon CWeilerAthertonDemoView::WeilerAtherton(WeilerPolygon polygonCliped,WeilerPolygon polygonCliping)

{

//输出结果变量

WeilerGenPolygon resultPolygons;

//求得交点集

vector<WeilerPoint> interSectionPoints=CWeilerAthertonDemoView::GetIntersectionPoint(polygonCliped,polygonCliping);

//将交点插入被裁剪多边形顶点序列中

vector<WeilerPoint> newClippedPoints=CWeilerAthertonDemoView::GetNewListPointClipped(polygonCliped,interSectionPoints);

//将交点插入裁剪多边形顶点序列中

vector<WeilerPoint> newClippingPoints=CWeilerAthertonDemoView::GetNewListPointClipping(polygonCliping,interSectionPoints);

//搜寻点

WeilerPoint point;

//暂存点

WeilerPoint storagePoint;

//被裁减多边形搜寻索引

int clippedSearchIndex=0;

//裁剪窗口搜寻索引

int clippingSearchIndex=0;

//当在newClippedPoints寻找到入点

while(CWeilerAthertonDemoView::GetIntoPoint(newClippedPoints,point))

{

//暂存入点

storagePoint=point.CopyPoint();

//定义当前查找的输出结果

WeilerPolygon polygon;

do

{

//获取入点所在索引

clippedSearchIndex=CWeilerAthertonDemoView::GetIntoPoint(newClippedPoints,point);

//将入点存入输出结果

CWeilerAthertonDemoView::InsertTail(polygon.points,point);

//删除newClippedPoints中该点的入点标记

CWeilerAthertonDemoView::DeletePointIntoFlag(newClippedPoints,point);

//在newClippedPoints中取得入点后面顺序取顶点 碰上末尾则返回第一个元素

for(int i=0;i<newClippedPoints.size();i++)

{

int index=clippedSearchIndex+i+1;

if(index>newClippedPoints.size()-1)

{

index=index-newClippedPoints.size();

}

point=newClippedPoints[index];

//如果是出点则跳出

if(point.pointFlag==2)

{

break;

}

//如果不是出点则存入输出结果

CWeilerAthertonDemoView::InsertTail(polygon.points,point);

}

//获取出点所在索引

clippingSearchIndex=CWeilerAthertonDemoView::GetOutPoint(newClippingPoints,point);

//沿newClippingPoints顺序取顶点

for(int j=0;j<newClippingPoints.size();j++)

{

int index=clippingSearchIndex+j;

if(index>newClippingPoints.size()-1)

{

index=index-newClippingPoints.size();

}

point=newClippingPoints[index];

//如果是入点则跳出

if(point.pointFlag==1)

{

break;

}

//如果不是入点则存入输出结果

CWeilerAthertonDemoView::InsertTail(polygon.points,point);

}

}

//等于暂存点则退出

while(point.x!=storagePoint.x||point.y!=storagePoint.y);

//保存当前寻找到的多边形

resultPolygons.polygons.push\_back(polygon);

}

return resultPolygons;

}

3 讨论

Weiler-Atherton算法将多边形交点插入到裁剪多边形和被裁减多边形顶点矢量数组中,通过记录交点及其前驱、后继信息,可快速生成结果多边形。其中,时间复杂度为O((m+k)×k),m是两多边形中顶点数较大者,k是两多边形的交点数。该算法简化了交点的数据结构,节省了存储空间,降低了算法的时间复杂度,该算法简单、易于编程实现、运行效率高。Sutherland－Hodgeman[算法](http://lib.csdn.net/base/datastructure)解决了裁剪窗口只适用于凸多边形窗口的问题，无法解决任意多边形窗口（含凹多边形窗口）的裁剪。梁友栋-Braian算法虽然具有一般性，不过算法中应用大量的乘除法，不利于计算机的内部实现。所以Weiler-Atherton算法比其他算法拥有更多的优势。

4 结论

Weiler-Atherton算法可以被广泛应用于各种裁剪和研究领域中，具有很高的价值，该算法主要有以下优点：

　1、裁剪窗口可以是矩形、任意凸多边形、任意凹多边形。

　2、可实现被裁剪多边形相对裁剪窗口的内裁或外裁，即保留窗口内的图形或保留窗口外的图形，因此在三维消隐中可以用来处理物体表面间的相互遮挡关系。

　3、裁剪思想新颖，方法简洁，裁剪一次完成，与裁剪窗口的边数无关。

不过Weiler－Atherton算法的编程实现比Sutherland-Hodgman算法稍难，主要难在入、出点的查寻以及跨数组搜索上。

参考文献:

[1] [宋树华](http://xueshu.baidu.com/usercenter/data/author?cmd=authoruri&wd=authoruri%3A%283a37a28ea14d50a0%29%20author%3A%28%E5%AE%8B%E6%A0%91%E5%8D%8E%29%20%E5%8C%97%E4%BA%AC%E5%A4%A7%E5%AD%A6%E9%81%A5%E6%84%9F%E4%B8%8E%E5%9C%B0%E7%90%86%E4%BF%A1%E6%81%AF%E7%B3%BB%E7%BB%9F%E7%A0%94%E7%A9%B6%E6%89%80)，[濮国梁](http://xueshu.baidu.com/s?wd=authoruri%3A%28c4cf974acb945997%29%20author%3A%28%E6%BF%AE%E5%9B%BD%E6%A2%81%29%20%E4%B8%AD%E5%9B%BD%E8%B5%84%E6%BA%90%E5%8D%AB%E6%98%9F%E5%BA%94%E7%94%A8%E4%B8%AD%E5%BF%83&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight%3Dperson&sort=sc_cited). [简单多边形裁剪算法](http://xueshu.baidu.com/s?wd=paperuri%3A%28ae9fa24aa7b7dea06ab349c86386a747%29&filter=sc_long_sign&tn=SE_xueshusource_2kduw22v&sc_vurl=http%3A%2F%2Fwww.cnki.com.cn%2FArticle%2FCJFDTotal-SJSJ201401036.htm&ie=utf-8&sc_us=12973151367241887824)[J]. 计算机工程与设计,2014,01期:192-197.

[2] 朱长青，史文中. 空间分析与建模[M]. 科学出版社,2006.

[3] [温俊文](http://www.cnki.net/KCMS/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CJFQ&sfield=au&skey=%e6%b8%a9%e4%bf%8a%e6%96%87&code=09361337;). 裁剪任意多边形的一种新算法[J]. 湛江师范学院学报(自然科学版),2000,02期:40-42.

[4] [王艳丽](http://www.cnki.net/KCMS/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CMFD&sfield=au&skey=%e7%8e%8b%e8%89%b3%e4%b8%bd&code=07175749;). 空间矢量数据的叠置算法研究[J]. 河南大学, 2010,01期: 3-4.

[5] [吴兵](http://www.cnki.net/KCMS/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CJFQ&sfield=au&skey=%e5%90%b4%e5%85%b5&code=06264560;06267753;06276873;),[尹伟强](http://www.cnki.net/KCMS/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CJFQ&sfield=au&skey=%e5%b0%b9%e4%bc%9f%e5%bc%ba&code=06264560;06267753;06276873;),[凌海滨](http://www.cnki.net/KCMS/detail/%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20/kcms/detail/search.aspx?dbcode=CJFQ&sfield=au&skey=%e5%87%8c%e6%b5%b7%e6%bb%a8&code=06264560;06267753;06276873;). 具有拓扑关系的任意多边形裁剪算法[J]. 小型微型计算机系统,2000,11期: 1166-1168.