第16卷 第7期
2006年7月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol.16 No.7
Jul. 2006

简单多边形快速Delaunay三角剖分算法

刘建新,卢新明,岳 昊
(山东科技大学 信息科学与工程学院,山东 青岛 266510)

摘 要:简单多边形的Delaunay三角剖分,在计算机图形学及地学问题三维建模领域有着广泛的应用。文中在借鉴他人的基础上,提出了一种时间复杂度为$O(mn)$的基于三角形权值最大的简单多边形Delaunay三角剖分算法。三角剖分结果中的三角形形态达到了最优或次优,并进行了理论上的严格证明,对算法的时间复杂度进行了分析,并给出了一个实例。实验结果表明,该方法对于随机生成的简单多边形域三角化速度快,平均计算时间呈近似线性。

关键词:多边形;三角形权值;Delaunay三角剖分;时间复杂度;计算几何
中图分类号:O18;TP301.6 **文献标识码**:A **文章编号**:1673-629X(2006)07-0126-03

Fast Algorithm for Delaunay Triangulation of Simple Polygon Based on Maximum Triangle Weights

LIU Jian-xin, LU Xin-ming, YUE Hao
(Sch. of Info. Sci. and Eng., Shandong Univ. of Sci. and Techn., Qingdao 266510, China)

Abstract: The Delaunay triangulation of simple polygon, being basic methods of calculating geometry, has been widely applied to computer graphics, 3D geographic modeling. This paper presents a fast algorithm for Delaunay triangulation of simple polygon based on maximum triangle weights referring to others. The state of triangle is best or better in the result of triangulation. The correctness and efficiency of the algorithm are confirmed strictly and the algorithm's time complexity is illustrated. Finally an example is given. The tested analysis shows that for simple polygonal domains randomly generated, the algorithm is efficient in computation and has an almost linear in running time.

Key words: polygon; triangle weights; Delaunay triangulation; time complexity; computational geometry

0 引 言

三角形作为最简单的平面图形,在计算几何、计算机图形学、科学计算可视化等许多领域中应用广泛。平面任意简单多边形的三角剖分问题是计算几何研究的一个基本问题。简单多边形根据形态特征可以分为不同的类,不同类的特殊多边形可以有不同的适于该类的三角剖分算法[1~4]。对于任意简单多边形三角剖分的通用算法而言,Chazelle[1]于1990年提出了一个线性时间的算法,在理论上是一个重大突破。但是任意简单多边形的三角剖分问题依然是一个热点研究问题,这是因为:第一,Chazelle的算法难于编程,难于实现;第二,Chazelle的最优算法没有考虑三角剖分后三角形的形态质量。Kong[3]给出了一个时间复杂度为$O(mn)$的算法(m为凹点数,n为多边形顶点数),但是Kong算法其剖分结果与扫描起始点的选择有关,且很难保证剖分获得的三角形形态质量;杨杰[4]在Kong算法的基础上对三角剖分结果进行局部三角形优化,提高了三角形的形态质量,但是依然得不到最好的结果;马小虎[5]提出了基于三角形权值最大的Delaunay三角剖分算法的思想,很好地解决了Kong算法的问题,但它的时间复杂度为$O(n^3)$,大大高于最优算法。文中提出了一种时间复杂度为$O(mn)$的基于三角形权值最大的简单多边形Delaunay三角剖分算法,三角剖分结果中的三角形形态达到了最优或次优,文中对算法进行了理论上的严格证明,对算法进行了分析并给出了一个实例。

收稿日期:2005-11-02
作者简介:刘建新(1980-),男,山东聊城人,硕士研究生,研究方向为计算机图形学;卢新明,教授,博士生导师,主要从事CAD图形图像的研究以及软件研发与产品设计工作。

1 基本概念

简单多边形P有n个顶点($n \geqslant 3$),顶点依次按逆时针方向排列,记为$p_1, p_2, \cdots, p_n$。

定义1 设p_i是简单多边形P的一个凸点,三角形$p_{i-1}p_ip_{i+1}$不包含P的其他顶点且线段$p_{i-1}p_{i+1}$全部在多边形P内部,则称顶点p_i为P的第一类凸点(或耳)。

定义2 设p_i是简单多边形P的一个凸点,三角形$p_{i-1}p_ip_{i+1}$包含P的除p_i, p_{i-1}, p_{i+1}之外的其他顶点,则称顶点p_i为P的第二类凸点,判定简单多边形P的顶点是第一类凸点、第二类凸点还是凹点的过程称为对顶点归类。

定义 3　左转,右转[2]　若 $p_1p_2p_3$ 是一个优角($\geq \pi$),则称 $p_1p_2p_3$ 是一个右转,否则是一个左转。

定义 4　三角形权值　三角形权值定义为三角形 3 个内角的最小值。

2　算法描述

算法 1

输入:任意简单多边形 P 的顶点 $p_1, p_2, \cdots, p_i$ 按逆时针方向排列。

输出:Delaunay 三角剖分 P 所得的三角形序列,存放在三角网格表 TML 中。

Step1　顺序读入简单多边形 P 的顶点,建立双向循环链表 L_1。

Step2　判别 P 中每个顶点 p_i 的凹凸性,作出标记,并将凹点存放在单链表 L_3 中。

Step3　对 P 中的顶点进行归类,确定每一个顶点是第一类凸点、第二类凸点或是凹点,并标记(本步由算法 2 实现)。

Step4　对多边形 P 顶点的双向循环链表 L_1 中的每一个第一类凸点 p_i,计算其对应的 $\triangle p_ip_{i-1}p_{i+1}$ 的权值,并将这些凸点序号按相应三角的权值从大到小排序,放入另外一个单链表 L_2 中。

Step5　若 L_1 中结点数 $\leqslant 3$,转 Step9,否则转 Step6。

Step6　从存放第一类凸点的链表 L_2 中取出第一个点 $q_1(p_i)$,不妨仍以 p_{i-1}, p_{i+1} 记 p_i,此时在双向链表 L_1 中的前驱和后继,将 q_1 对应的 $\triangle p_ip_{i-1}p_{i+1}$ 保存到 TML 表中。

Step7　修改单链表 L_2,使得它依然是全部第一类凸点的集合,且依然按所对应三角形权值非递增有序,修改单链表 L_3,使得它依然是全部凹点的集合(本步由算法 3 实现)。

Step8　从 L_2 中删除顶点 q_1,从 L_1 中删除顶点 p_i,转 Step5。

Step9　由链表 L_1 中最后 3 个结点所对应的多边形顶点构成一个三角形,保存到 TML 表中,删除链表中最后 3 个结点。

Step10　由三角形网格表 TML 得到三角形网格的边表。

Step11　对边表中的每一条边,若该边非边界边,则考察以该边为公共边的两个三角形,它们构成一个四边形,如果这个四边形是凹的,则不处理,否则计算两个三角形的最小内角 α_1,交换对角线计算两个新的三角形的最小内角 α_2,若 $\alpha_2 > \alpha_1$,则以交换对角线所得到的两个新三角形替换原来的两个三角形,并按新的三角形修改 TML。

在该算法 Step2 中判定顶点 p_i 的凹凸性的方法[2]:先确定 $p_{i-1}p_{i+1}$ 是左转还是右转,设 $p_i = (x, y)$, $p_{i-1} = (x', y')$, $p_{i+1} = (x'', y'')$ 由 $\triangle = \begin{vmatrix} x' & y' & 1 \\ x & y & 1 \\ x'' & y'' & 1 \end{vmatrix}$ 的正负号,当 $\triangle > 0$ 时是左转,当 $\triangle \leqslant 0$ 时是右转;若 $p_{i-1}p_ip_{i+1}$ 是一个右转,则 p_i 是凹点,否则 p_i 是凸点。

算法 2

确定每一个凸顶点是第一类凸点还是第二类凸点。

(1) $i \leftarrow 1$;

(2) 若 $i < (n-1)$ 则转(3),否则转(6);

(3) 若 p_i 是凸点,转(4),否则转(5);

(4) 若 $\triangle p_{i-1}p_ip_{i+1}$ 中不包含存放多边形凹点的链表 L_3 中的其它点,则 p_i 为第一类凸点;否则标记 p_i 为第二类凸点。转(5);

(5) $i \leftarrow i+1$,转(2);

(6) 用同样的方法判定 p_{n-1} 是第一类凸点、第二类凸点还是凹点,结束。

在该算法中,第(4)步测试 $\triangle p_{i-1}p_ip_{i+1}$ 是否包含凹点 s,方法为分别测试 $p_{i-1}p_is$, $p_ip_{i+1}s$, $p_{i+1}p_{i-1}s$ 的转向,若全是左转则 s 在 $\triangle p_{i-1}p_ip_{i+1}$ 内(如图 1 所示)。

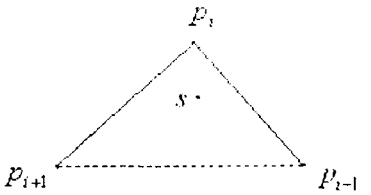

图 1　判断点 s 是否在 $\triangle p_{i-1}p_ip_{i+1}$ 内

算法 3

修改单链表 L_2,使得它依然是全部第一类凸点的集合,且依然按所对应三角形权值非递增有序,修改单链表 L_3,使得它依然是全部凹点的集合。

算法 1 的 Step6 处理的第一类凸点为 $q_1(p_i)$,如图 2 所示。

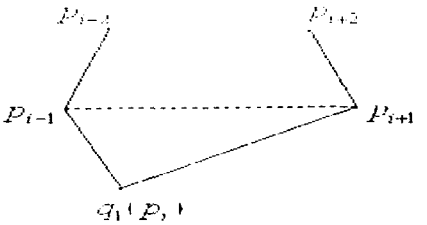

图 2　第一类凸点为 $q_1(p_i)$

不妨仍以 p_{i-1}, p_{i+1} 分别记 p_i 在此时的多边形双向链表 L_1 上的前驱和后继,仍以 p_{i-2}, p_{i+2} 分别记此时在 L_1 上 p_{i-1} 的前驱和 p_{i+1} 的后继。则算法描述如下:

(1) 若 p_{i+1} 是第一类凸点,则在存放第一类凸点信息的单链表 L_2 中修改 p_{i+1} 对应的三角形的权值,调整 L_2 使得 L_2 依然按其存放的第一类凸点所对应三角形的权值非递增有序,转(5);否则转(2)。

(2) 若 p_{i+1} 是第二类凸点且 $\triangle p_{i-1}p_{i+1}p_{i+2}$ 不包含存放凹点的单链表 L_3 中的点,则在 L_1 中修改 p_{i+1} 的标记为第一类凸点,计算 $\triangle p_ip_{i+1}p_{i+2}$ 的权值,将 p_{i+1} 插入 L_2,并使得 L_2 依然按其所对应三角形权值非递增有序,转(5);否则转(3)。

(3) 若 p_{i+1} 是凹点，且 $\angle p_{i-1}p_{i+1}p_{i+2} < \pi$，且 $\triangle p_{i-1}p_{i+1}p_{i+2}$ 不包含存放凹点的单链表 L_3 中的点，则在 L_1 中修改 p_{i+1} 的标记为第一类凸点，计算 $\triangle p_ip_{i+1}p_{i+2}$ 的权值，将 p_{i+1} 插入 L_2，并使得 L_2 依然按其所对应三角形权值非递增有序，在 L_3 删除顶点 p_{i+1}；否则转(4)。

(4) 若 p_{i+1} 是凹点，且 $\angle p_{i-1}p_{i+1}p_{i+2} < \pi$，且 $\triangle p_{i-1}p_{i+1}p_{i+2}$ 包含存放凹点的单链表 L_3 中的点，则在 L_1 中修改 p_{i+1} 的标记为第二类凸点，在 L_3 删除顶点 p_{i+1}；否则转(5)。

(5) 同样处理点 p_{i-1}。

(6)结束。

3 算法正确性

引理 1 两耳定理。非三角形的任意简单多边形至少有两个第一类凸点。

引理 2 在简单多边形 P 中，顶点 p_i 是第二类凸点，它的前驱顶点和后继顶点分别为 p_{i-1} 和 p_{i+1}，则 $\triangle p_{i-1}p_ip_{i+1}$ 中包含多边形 P 的凹点。

证明：p_i 是第二类凸点，故 p_i 包含多边形 P 的顶点，设 p_k 是这些顶点中距离直线 $p_{i-1}p_{i+1}$ 最远的。下面证明 p_k 是一个凹点：以 r 和 s 分别记过 p_k 且平行于 $p_{i-1}p_{i+1}$ 的直线与 p_ip_{i-1} 和 p_ip_{i+1} 的交点(如图 3 所示)，$\triangle sp_ir$ 的内部区域全部是多边形 P 的内部区域，所以 $p_{k-1}p_kp_{k+1}$ 是一个右转，故 p_k 是一个凹点。

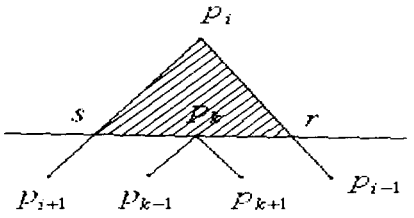

图 3 确定 p_k 是一个凹点

引理 3 算法 2 是正确的。

证明：由引理 2 易知。

引理 4 在简单多边形 P 中(如图 2 所示)割去第一类凸点 p_i 对应的三角形 $\triangle p_{i-1}p_ip_{i+1}$ 不会影响与 p_i 不相邻的顶点 p_k 的归类。

证明：若 p_k 是第一类凸点或凹点则保持归类不变；若 p_k 是第二类凸点，由引理 2 知在未割去 $\triangle p_{i-1}p_ip_{i+1}$ 之前 $\triangle p_{k-1}p_kp_{k+1}$ 包含 P 的一个凹点，在割去 $\triangle p_{i-1}p_ip_{i+1}$ 之后依然包含这个顶点，因此 p_k 依然是第二类凸点。

引理 5 算法 3 是正确的。

证明：由引理 4 知当在多边形 P 中割去第一类凸点 p_i 对应的三角形 $\triangle p_{i-1}p_ip_{i+1}$ 后，仅需考虑 p_{i-1} 与 p_{i+1} 的归类修改，以 p_{i+1} 为例分析各种情况：

情况 1，p_{i+1} 是第一类凸点，则它依然是第一类凸点(算法 3 第(1) 步)；

情况 2，p_{i+1} 是第二类凸点，则它可能依然是第二类凸点，也可能变为第一类凸点(算法 3 第(2) 步)；

情况 3，p_{i+1} 是凹点，则它可能依然是凹点，也可能变为第一类或第二类凸点(算法 3 第(3)、(4)步)。

算法 3 考虑到了所有可能的情况，因此是正确的。

4 算法的时间复杂度分析

定理 2 算法的时间复杂度分析为 $O(mn)$(n 为简单多边形 P 的顶点数，m 为简单多边形 P 的凹顶点数)。

证明：算法 1 中 Step1 有 $O(n)$ 次对链表的操作(包括初始化链表、新建结点、往结点中写信息、移动指针)；Step2 耗费 $O(n)$ 次确定三个点是右转或左转的操作(每次需要 6 次乘法 5 次加法)，$O(n)$ 次对链表的操作；Step3 耗费 $O(mn)$ 次判定一个点是否在一个三角形内的操作(每次需要 18 次乘法、15 次加减法和 3 次比较)，$O(n)$ 次对链表 L_1 的操作(包括移动指针、往结点中写信息)；Step4 耗费 $O(n)$ 次求夹角的运算，$O(n)$ 次比较和 $O(n\log n)$ 次对链表的操作；Step5 至 Step8 是一个双重循环，外循环中的每一次 Step5 是一次比较，Step6 花费 $O(n)$ 次比较和在链表 L_1 上移动指针寻找 p_i，内循环 Step7 是算法 3，算法 3 花费 $O(m)$ 次判定一个点是否在一个三角形内的操作以及 $O(n)$ 次对链表的操作(移动指针、比较)；Step8、Step9 耗费常数时间；Step10 和 Step11，由于 TML 表中三角形共有 $(n-2)$ 个，其边也是 $O(n)$ 条，因此花费为 $O(n)$ 次乘法、反三角函数和比较。由于对链表的操作(移动指针、比较)耗时较少，可以不做为主要运算，综合以上分析可以知道整个算法的时间复杂度为 $O(mn)$。算法过程如图 4～图 6 所示。

5 结 论

提出了一种时间复杂度为 $O(mn)$ 的基于三角形权值最大的简单多边形 Delaunay 三角剖分算法。三角剖分结果中的三角形形态达到了最优或次优，算法在执行过程

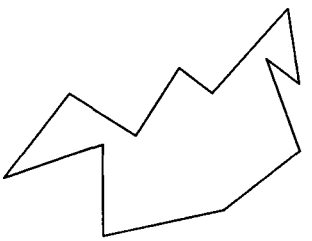

图 4 简单多边形 P

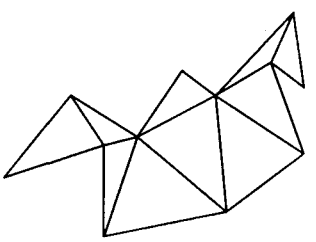

图 5 执行 step1～9 的结果

(下转第 185 页)

```
……
//分配内存
    sw_ priv = kmalloc(sizeof(struct switch_ enet_ private),
GFP_KERNEL);
    //网卡设备数据结构初始化
    dev=init_etherdev(0,0);
    ……
    sw_priv→dev=dev;
    dev→priv=sw_priv;
    dev→irq=SW_IRQ;
    dev→open=switch_enet_open;
    dev→stop=switch_enet_close;
    dev→hard_start_xmit=switch_enet_start_xmit;
    ……
    }
```

(2)设备打开操作。

打开操作完成请求注册切换中断 SW_IRQ,启动定时器等操作。

PPC8250 中请求中断,采用下面的方式:

```
result=request_8xxirq(dev→irq,switch_enet_interrupt,0,dev→
name,dev)
```

(3)发送操作。

发送过程跟普通的网卡驱动一样,只需要把待发送的数据添加到发送队列中即可。

(4)切换中断服务程序。

切换中断服务程序完成 DPRAM 的读写操作。每一次中断来临后,会读取 DPRAM 的所有数据,然后调用 netif_rx 传送到网络协议的上层;同时,读取所有待发送队列中的数据,将数据写到 DPRAM 中。

3　网络环境配置以及安全设置

要完成内外网的数据传送,需要通过正确配置网桥工作方式;同时,加入各种网络安全措施,比如设置 netfilter 过滤规则可以显著提高物理隔离器的安全性。

3.1　网桥设置

在命令行状态下,输入下列命令,配置网桥[5]:

```
#brctl addbr br0
#brctl addif br0 eth0
#brctl addif br0 eth1
#ifconfig eth0 down
#ifconfig eth1 down
#ifconfig eth0 0.0.0.0 up
#ifconfig eth1 0.0.0.0 up
#ifconfig br0 192.168.0.2 up
```

3.2　过滤规则设置

如果内核中 netfilter 功能被正确启用后,用户通过命令行或者其他方式输入的过滤规则,会被系统匹配,以决定是否允许数据包通过。命令行方式下的规则设置使用 iptables。

4　结束语

文中从整体上描述了物理隔离器的实现方案和原理,并阐述了开发编译环境建立、网卡驱动实现等。随着网络安全的重要性进一步加强,物理隔离器将得到更加广泛的应用。

参考文献:

[1] Yaghmour K. Building Embedded Linux Systems[M]. USA: O'Reilly,2003.54-64.

[2] Rubini A,Corbet J. Linux Device Driver(2nd Edition)[M]. USA:O'Reilly,2001.425-469.

[3] 毛德操,胡希明. Linux 内核源代码情景分析[M]. 杭州:浙江大学出版社,2002.119-559.

[4] 陈莉君. 深入分析 Linux 内核源代码[M]. 北京:人民邮电出版社,2002.210-325.

[5] Dyson P,Kelly-Bootle S. UNIX 大全[M]. 北京:电子工业出版社,2000.51-62.

(上接第 128 页)

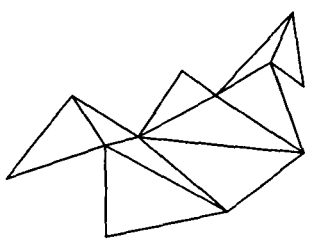

图 6　最终结果

中,由于在输出一个三角形后多边形顶点中归类受到影响的仅有有限的两个,相对于文献[5]中的算法,本算法避免了大量的不必要的运算,同时又设计了巧妙的数据结构来存放不同类别顶点的信息,因此算法效率大大提高,时间复杂度为 $O(mn)$。

参考文献:

[1] Chazelle B. Triangulation a simple polygon in linear time[R]. Technical Report CS-TR-264-90,Dept. of Computer Science ,Princeton University,1990.

[2] Preparata F P,Shamos M I. Computational Geometry[M]. New York:Springer-Verlag,1985.

[3] Kong X S, Everett H,Toussaint G T. The Graham Scan Triangulates Simple Polygons[J]. Pattern Recognition Letters, 1990,11:713-716.

[4] 杨　杰. 基于凹凸顶点判定单多边形的三角剖分[J]. 小型微型计算机系统,2000,21(9):974-975.

[5] 马小虎. 基于凹凸顶点判定的简单多边形 Delaunay 三角剖分[J]. 计算机辅助设计与图形学学报,1999,11(1):1-3.