第33卷 第17期 计 算 机 工 程 2007年9月
Vol.33 No.17 Computer Engineering September 2007
·软件技术与数据库· 文章编号：1000—3428(2007)17—0093—03 文献标识码：A 中图分类号：TP391

基于最优凸壳技术的Delaunay三角剖分算法

陈学工，黄晶晶
(中南大学信息科学与工程学院，长沙 410083)

摘 要：提出了一种基于最优凸壳技术的Delaunay三角剖分算法。该算法对离散点进行扫描线方式排序，利用最优凸壳技术进行凸壳的生成和三角网联结，最后利用有向边的拓扑结构进行三角网优化。该算法不但避免了所有的交点测试，而且使得新加入点与凸壳边的平均比较次数不大于4，从而实现了高效的三角剖分。

关键词：Delaunay三角剖分；凸壳；三角网优化

Algorithm of Delaunay Triangulation Based on Optimal Convex Hull Technology

CHEN Xue-gong, HUANG Jing-jing
(School of Information Science and Engineering, Central South University, Changsha 410083)

【Abstract】 A Delaunay triangulation algorithm based on optimal convex hull technology is presented. The algorithm makes the discrete points sort in scan manner, and secondly it constructs convex hull and triangulates the sorted points by the optimal convex hull technology which is proved by the author, and optimizes triangles utilizing topological structures of directed edges. The algorithm avoids the test of point of intersection. Moreover, the average test times of a newly added point is under 4, so that the high efficiency of triangulation can be sure.

【Key words】 Delaunay triangulation; convex hull; triangles optimization

DEM中的TIN模型的优点是它能以不同层次的分辨率来描述地形表面，因此被广泛使用[1]。在目前的三角化算法中，以DT的应用最为广泛。DT经典算法根据其实现过程可分为分治算法、逐点插入算法和三角网生长法[2]3种。现在国内外多数的研究集中在逐点插入算法的搜索策略的改进[3~5]和分治算法与逐点插入算法的合成[6,7]两个方面。而文献[8]提出了一个基于凸壳技术的DT快速算法，该算法以有序点着眼点并利用了有序点子集的凸壳特性，彻底避免了三角化时的交点测试，大大提高了TIN的生成效率。但文献[8]中算法在判断新加入点与凸壳边能否联结成合理三角形时，未充分利用凸壳特性，使得参与判断的凸壳边条数较多，影响了构网效率。为此笔者提出了一种基于最优凸壳技术的DT算法，该算法能大大地减少三角网联结过程中参与判断的凸壳边的条数并保持文献[8]中算法的优点，因此能进一步提高三角网的生成速度。

1 最优凸壳技术及算法基本思想

定义1 组成三角网的所有边都是有向的，其方向为组成边的两个点的次序。有向边左侧的三角形称为边的左三角形，右侧的三角形称为边的右三角形。

定义2 对一个给定的平面点集SP，包含SP的最小凸多边形称为SP的凸壳。本文规定这个凸多边形的点按逆时针方向存储。

定义3 如果新生成的三角形与原来的三角网的三角形不相交且不重叠，称这些三角形为合理三角形。这里如果新加入的点在凸壳边的右侧，则可以构成合理三角形。

设SP为离散点集，P是SP中的点按扫描线方式排序后的有序点集，假设P中无重复的点。$S=\{p_1,p_2,...,p_{i-1}\}$，S是已参与构造三角网的点集，S的凸壳为K，$p_i$是即将进行三角网构造的点。如图1所示。

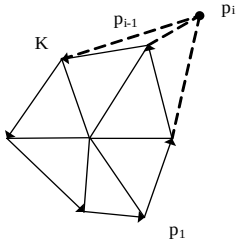

图1 凸壳K与P的关系

定理1 P中的第1个点p_1必在凸壳上，而p_i必在凸壳K外部[8]。

定理2 p_i至少与凸壳K的一条凸壳边构成三角形，并且这些边是连续的。同时必有两条新边与凸壳K中的一些边构成{S，p_i}的新凸壳K'[8]。

定理3 p_i不可能与点集P的第1个点p_1的两条凸壳边构成合理三角形[8]。

定理4 p_i能与凸壳K中的两条以上(包括两条)的边构成合理三角形时，凸壳K上的一些点会从凸壳上删除。

证明 由定理1和定理2，经过p_i的新生成的凸壳边替代能与其生成合理三角形的凸壳边，当K中这样的边存在两条以上(包括两条)时，其中被这些边中的两条边共用的点就会被从凸壳K中删除。

定理5 p_{i-1}两条凸壳边中至少有一条能与p_i构成合理三角

基金项目：国家“863”计划基金资助项目(2002AA135160)
作者简介：陈学工(1965－)，男，副教授、博士，主研方向：地理信息系统等；黄晶晶，硕士研究生
收稿日期：2006-09-07 **E-mail：**cn1233@263.net

形。

证明 因为P是SP中的点按扫描线方式排序后的有序点集。只有p_i是凸壳内部的点的时候，才会使得p_i和p_{i-1}的两条凸壳边构成不了两个合理三角形，与定理 1 相矛盾。

定理 6 如果从p_{i-1}两条凸壳边开始分别向凸壳K的起点和终点做p_i与凸壳边能否构成合理三角形的判断，则新加入点与凸壳边的平均比较次数不多于 4。计算P的凸壳的时间复杂度为O(n)。

证明 新点p_i的插入可能使得原凸壳发生只有p_i的插入和p_i的插入引起了原凸壳上的某些点的删除两种情况。由定理 4，第 1 种情况时p_i只能与凸壳K中的一条边构成合理三角形，第 2 种情况时p_i只能与凸壳K中的两条以上的边构成合理三角形。因此将点的插入都转换为点的删除进行讨论，可以整体得出每个点的平均判断次数的一个大概值。如果P中的点都在凸壳上，则在每插入一个点时无点从原凸壳上删除，根据定理 2 和定理 5，可得到这个点最多只要与凸壳中的 3 条边比较。如果P只有 3 个点在凸壳上，参与判断的凸壳边最多的情况是每一点的加入只删除一点，根据定理 2 和定理 5，则这个点最多只要与凸壳中的 4 条边比较。所以，从整体上来看，新加入点与凸壳边的平均比较次数不多于 4，计算P的凸壳的时间复杂度为O(n)。

只要采用时间复杂度为O(nlogn)的排序算法，利用定理 1～定理 6 就能使得计算凸壳的时间复杂度为O(nlogn)。由于计算平面上n个点的凸壳的时间复杂度的下界为$\Omega(n\log n)$[9]，这种凸壳技术可以称为最优凸壳技术。本文的算法基本思想是：首先对所有SP中的点按照扫描线方式排序为P；然后$\{p_1,p_2,....,p_{i-1}\}$生成第 1 个凸壳(凸壳从p_1为起点的凸壳边开始到p_1为终点的凸壳边结束逆时针方向存储)；再进行有序点集的下一点p_i与已生成的凸壳的凸壳边能否构成合理三角形的判断(判断的过程基于最优凸壳技术，从p_{i-1}所在的凸壳边向凸壳的两端搜索使得p_i在凸壳边右侧的边)，找到能构成合理三角形的凸壳边，构成三角形，并且形成新的凸壳，重复以上过程直到所有的点都加入三角网；最后进行三角网优化。文献[8]中算法只利用了定理 1～定理 3 证明的凸壳特性，因此计算凸壳的时间复杂度为$O(n^2)$[9]。而本文算法利用了定理 1～定理 6 证明的凸壳特性，所以本文的算法保持了文献[8]中算法的优点，且时间效率将高于文献[8]中的算法，浮点运算的减少也会提高算法的健壮性。

2 数据结构

本文的算法用到以下数据结构：

(1)有序点表。该表保存散乱点集和排序后的点序列。表中每个元素的结构为

```
typedef struct {
double x,y,z;//点的坐标
}Point;
```

(2)有向边表。该表保存三角形的边以及与该边相邻的三角形信息。表中每个元素的结构为

```
typedef struct{
long TriNo[2];//两相邻三角形编号
Point s,e;//有向边的始末点
}Edge;
```

(3)三角形表。该表保存生成的三角形的边信息和顶点信息。表中每个元素的结构为

```
typedef struct{
long eNo[3];//3 条边号按逆时针存储
}Triangle;
```

(4)凸壳边表。该表保存点集凸壳的边在边集中的位置和顶点信息。表中的每个元素的结构为

```
typedef struct{
long eNo;//有向边表中的边号
Point s,e;//有向边的始末点
}Convex;
```

3 算法关键步骤

本文算法的主要步骤如下：(1)将点集按扫描线方式排序；(2)建立第 1 个凸壳；(3)利用最优凸壳技术将有序点集的所有点联结成三角网；(4)对三角网进行优化生成D-三角网。具体的点与有向边关系的判断采用CCW方法[10]。

3.1 点集排序

定理 1～定理 5 的前提条件，即数据集内的数据是排好序的。所以首先选择快速排序算法将原始数据集 SP 按扫描线方式排序生成有序点集 P。

3.2 建立第 1 个凸壳

用 cEN 表示凸壳链表 CList 中凸壳边的数目，eN 表示边数组 EArray 中有向边的数目，tN 表示三角形数组 TArray 中三角形的数目。

(1)P中顺序查找第 1 个不在有向边p_1p_2上点p_i。

(2)如果 i=n，则ｎ个点共线，结束。

(3)如果p_i在有向边p_1p_2的右侧，则将有向边p_1p_i，p_ip_{i-1}，$p_{i-1}p_{i-2}$，…，p_2p_1添加到CList和EArray，令cEN=i，eN=i；将有向边p_ip_{i-2}，p_ip_{i-3}，…，p_ip_2添加到EArray，令eN+=i-3；将新产生的△$p_ip_{i-1}p_{i-2}$，△$p_ip_{i-2}p_{i-3}$，…，△$p_ip_2p_1$添加到TArray中，令tN=i-2；转(5)。

(4)如果p_i在p_1p_2的左侧，则将有向边p_1p_2，p_2p_3，…，$p_{i-1}p_i$，p_ip_1添加到CList和EArray，令cEN=i，eN=i；将有向边p_ip_{i-2}，p_ip_{i-3}，…，p_ip_2添加到EArray，令eN+=i-3；将新产生的△$p_ip_{i-2}p_{i-1}$，△$p_ip_{i-3}p_{i-2}$，…，△$p_ip_1p_2$添加到TArray中，令tN=i-2；

(5)输出 CList、EArray 和 TArray。

3.3 三角网联结

(1)假设当前处理的点为p_i且p_1，…，p_{i-1}点已经联结成三角网。p_{i-1}所在的两条凸壳边所在CList中的位置分别为c1 和 c2，c1 在c2 的前面。

(2)从CList中位置c1 往前搜索直至凸壳边p_jp_k使得p_i在其左侧或凸壳边p_jp_k为空值(即搜索不到凸壳边时)，记录凸壳边p_jp_k的前一个被搜索到的凸壳边$p_{j-1}p_{k-1}$在凸壳中的位置c3；从CList中位置c2 往后搜索直至凸壳边p_mp_n使得p_i在其左侧或凸壳边p_mp_n为空值(即搜索不到凸壳边时)，记录凸壳边p_mp_n的前一个被搜索到的凸壳边$p_{m-1}p_{n-1}$在凸壳中的位置c4。

(3)将p_i与p_{j-1}连接，形成$p_{j-1}p_i$，添加到EArray，令eN++；对CList中位置c3 和c4 间的凸壳边p_sp_e做如下操作 将p_i与p_e连接，形成p_ip_e，添加到EArray，令eN++，新形成的三角形△$p_sp_ip_e$添加到TArray并设置相关边的相邻三角形信息，令tN++。

(4)删除CList中位置c3 和c4 间的凸壳边，数量为t，将$p_{j-1}p_i$和p_ip_{n-1}从位置c3 添加到CList，cEN-=t-2。

(5)继续处理下一点，直至所有 P 中的点都处理完毕。

3.4 三角形优化

由于以上所联结的三角形不符合D-三角网的条件，因此需要采用三角形优化处理。根据D-三角网所具有的空圆准则，结合本文定义的数据结构，本文采取了以有向边作为优化着

眼点的处理算法。如图 2 所示，边e的相邻三角形为T_1、T_2。如果T_1与T_2形成凸四边形，则根据是否满足空圆规进行优化。点D与T_1的外接圆的关系如图 2 所示。根据圆周角性质有以下判断：sin(∠C＋∠D)>0，点D在圆外，须进行优化；sin(∠C＋∠D)<0，点D在圆内，不进行优化；sin(∠C＋∠D)=0，点D在圆上，可以也可以不做优化。如果直接计算出sin(∠C＋∠D)则过于复杂，本文只关心它的符号，不关心它的数值，可以简化为只求式(1)的符号即可，但这个式子得到的前提是三角形的各个顶点满足逆时针存储。式(1)中的x_1、y_1，x_2、y_2，x_3、y_3和x_4、y_4分别是A、B、C和D 4 点坐标。维护凸壳和三角形的逆时针存储为三角形的优化减少了计算时间，提高计算精度。

$$\begin{aligned}&\sin(\angle C+\angle D)\approx\\&[(x_1-x_3)*(y_2-y_3)-(x_2-x_3)*(y_1-y_3)]\\&*[(x_2-x_4)*(x_1-x_4)+(y_2-y_4)*(y_1-y_4)]\\&+[(x_2-x4)*(y_1-y_4)-(x_1-x_4)*(y_2-y_4)]\\&*[(x_1-x_3)*(x_2-x_3)+(y_1-y_3)*(y_2-y_3)]\end{aligned}\quad(1)$$

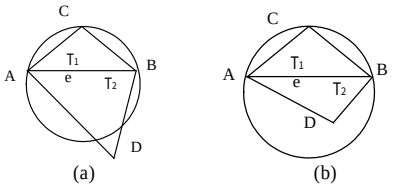

图 2　三角形优化

4　算法分析与结论

本文算法假定输入离散点的数目为*n*，按扫描线方式使用快速排序法排序，时间复杂度$T_1(n)=O(n\log n)$。三角网联结在凸壳的生成过程中完成，其时间复杂度即计算凸壳的时间复杂度$T_2(n)=O(n)$。文献[10]中证明了每个点涉及的三角形优化的数学期望为 6。因此三角网优化的时间复杂度$T_3(n)=O(n)$。本文算法时间复杂度$T(n)=T_1(n)+T_2(n)+T_3(n)=O(n\log n)$。

为了比较本文算法利用的最优凸壳技术和文献[8]中算法利用的凸壳技术，笔者将两种算法开发了相应的程序，在同一实验环境下，对 6 组地形数据的三角网联结过程中新加入点与凸壳边的比较次数进行了跟踪，并记录了 D-三角网的生成时间。结果如表 1 和表 2 所示。

在三角网联结过程中，表 2 表明，文献[8]中算法新加入点与凸壳边的平均比较次数随着数据的复杂程度和数据量的增加有了明显的增加。而表 1 表明，本文算法新加入点与凸壳边的平均比较次数保持在了 4 次之内。

表 1　本文算法三角网联结时凸壳边比较次数和生成时间

散乱点数	最少比较次数	最多比较次数	平均比较次数	时间/s
11 297	2	68	3.995	0.5
22 685	2	135	3.998	0.9
37 794	3	221	3.999	1.6
56 770	3	338	3.999	2.7
113 613	2	678	3.999	5.7
227 121	2	1 359	3.999	11.4

表 2　文献[8]算法三角网联结时凸壳边比较次数和生成时间

散乱点数	最少比较次数	最多比较次数	平均比较次数	时间/s
11 297	3	79	17.7	0.6
22 685	3	146	23.6	1.1
37 794	3	234	31.9	2.1
56 770	3	353	40.4	3.3
113 613	3	697	69.3	7.6
227 121	3	1 388	127.1	18.1

实验结果证明了本文算法利用的凸壳技术远远优于文献[8]中算法利用的凸壳技术，也证实了定理 5 和定理 6 的正确性。本文算法的 D-三角网的生成时间也比文献[8]中算法要快，而且随着数据的复杂程度和数据量的增加提高得越多。

参考文献

1 Zeiler M. Modeling our world[M]. US: ESRI Press, 1999.

2 武晓波, 王世新, 肖春生. Delaunay 三角网的生成算法研究[J]. 测绘学报, 1999, 28(1): 28-35.

3 Kolingerova I, Zalik B. Improvements to Randomized Incremental Delaunay Insertion[J]. Computers & Graphics, 2002, 26(3): 477-490.

4 宋占峰, 蒲　浩, 詹振炎. 基于三角网数字地面模型快速定位算法的研究[J]. 中国铁道科学, 2002, 23(1): 63-66.

5 Zhou Sheng, Jones C B. HCPO: An Efficient Insertion Order for Incremental Delaunay Triangulation[J]. Information Processing Letters, 2005, 93(1): 37-42.

6 向传杰, 朱玉文. 一种高效的 Delaunay 三角网合并生成技术[J]. 计算机应用, 2002, 22(1l): 34-36.

7 吴宇晓, 张登荣. 生成 Delaunay 三角网的快速合成算法[J]. 浙江大学学报, 2004, 31(3): 343-348.

8 章孝灿, 黄智才, 戴企成, 等. GIS 中基于拓扑结构和凸壳技术的快速 TIN 生成算法[J]. 计算机学报, 2002, 25(11): 1212-1218.

9 Rourke J O. Computational Geometry in C[M]. 北京: 机械工业出版社, 2005.

10 Mostafavi M A, Gold C, Dakowicz M. Delete and Insert Operations in Voronoi/Delaunay Methods and Applications[J]. Computers & Geosciences, 2003, 29(4): 523-530.

（上接第 92 页）

使得移植过程更加简单。同时将整个嵌入式软件系统分成不同的层次和模块，易于大规模的软件开发，便于软件的管理。采用这种方法，软件的每个功能模块可同时并行开发，从而大大缩短了系统的开发时间。因此，这种软件设计方案可以广泛应用于实际的工程应用中。该设计方案已应用于实际项目，并取得了很好的移植效果。

参考文献

1 王　涛, 张伟良, 冯重熙. 嵌入式系统硬件抽象层的原理与实现[J]. 电子技术应用, 2001, 27(10): 26-28.

2 李善平, 刘文峰, 王焕龙, 等. Linux 与嵌入式系统[M]. 北京: 清华大学出版社, 2002.

3 Sutter E. 嵌入式固件揭密[M]. 张晓林, 译. 北京: 电子工业出版社, 2003.

4 Bovet D P, Cesati M. 深入理解 LINUX 内核[M]. 陈莉君, 冯　锐, 牛欣源, 译. 北京: 中国电力出版社, 2005.

5 Samsung Electronics Inc.. S3C2410X 32-Bit RISC Microprocessor User's Manual[Z]. 2003.