第 33 卷 第 7 期 计 算 机 工 程 2007 年 4 月
Vol.33 No.7 Computer Engineering April 2007
·软件技术与数据库· 文章编号：1000—3428(2007)07—0070—03 文献标识码：A 中图分类号：TP391.41

对构建 Delaunay 三角网中凸壳算法的研究与改进

袁 翰，李伟波，陈婷婷
（武汉工程大学计算机科学与工程学院，武汉 430073）

摘 要：在介绍 Delaunay 不规则三角网基本概念和 TIN 数据结构的基础上，主要对平面离散点构建凸壳的格雷厄姆算法进行了研究和改进，提出了一种“斜率扫描线法”，并进行了编程实现。实验表明改进后的算法实现简单，容易理解，对于 D-TIN 模型的生成行之有效。
关键词：Delaunay 三角剖分；不规则三角网；凸壳

Research and Improvement of Convex Hull Algorithm in Construction of Delaunay Triangulation

YUAN Han, LI Weibo, CHEN Tingting
(School of Computer Science and Engineering, Wuhan Institute of Technology, Wuhan 430073)

【Abstract】While infroducing the essential meaning and the data structure of TIN, this paper studies and improves the Graham algorithm which constructs the convex hull based on the plane discrete points, proposes one “slope-scan-line” method, and a programming algorithm is achieved also. Experimental results show that the method is easy to obtain and to understand. It is effective to build Delaunay TIN.
【Key words】Delaunay triangulation; Triangulation irregular network (TIN); Convex hull

数字地面模型(Digital Terrain Modal, DTM)是地表勘测成果的数字化表现形式，它广泛地应用于地理信息系统等领域中，在重大危险源监测管理系统中也需要使用 DTM。

DTM 的主要实现形式是通过不规则三角网(Triangulation Irregular Network, TIN)，用一系列互不交叉的、互不重叠的连接在一起的三角形来表示地形表面。Delaunay 三角形具有空外接圆，以及最小角最大的性质，可最大限度地保证网中三角形满足近似等边(角)性，避免了过于狭长和尖锐的三角形的出现，是公认的最优三角网。不规则三角网具有许多优点，如利用原始数据作为网格节点；不改变原始数据及其精度；保存了原有的关键地形特征；利用 TIN 追踪等高线的算法相对简单；能够很好地适应不规则地形等。鉴于此，对二维域内点集进行 Delaunay 三角剖分是当今流行的 TIN 构网技术。

1 基本概念

1.1 不规则三角网

基于不规则三角网的数字高程模型，如图 1 所示。

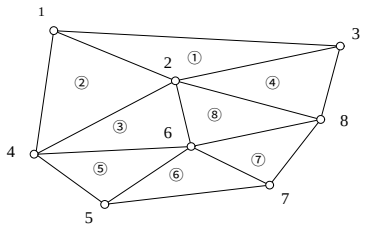

图 1 不规则三角网

1.2 凸壳定义

对于一个给定的平面散乱点集S，S被称为凸壳，当且仅当对于任意两点$p,q \in S$，线段 $\overline{pq}$ 都完全属于S(如图 2 示)[2]，集合S的凸壳就是包含S的最小凸多边形。

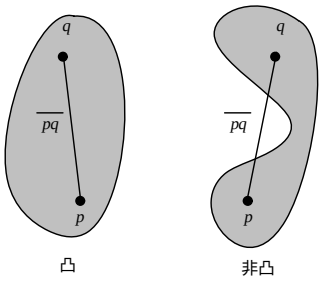

图 2 凸集与非凸集

2 TIN 的数据结构

在构建不规则三角网的过程中，算法的效率与数据结构之间有着密切的关系，它包含着点、线、面之间的组织关系和拓扑关系。一个好的数据结构有利于数据的执行和处理，是值得用心考虑的。基于此原则采用双向链表作为存储手段，用到的数据结构如下：

(1)点链表。该链表保存散乱点集预处理后的点序列，点的数据结构为

```
struct   Vertex {
    double x, y ; //点的坐标
    int VID; //点的序号
    Vertex *pleft ; //指向前一个点的指针
    Vertex *pright; //指向后一个点的指针
}
```

(2)边链表。该链表保存三角形的边以及与该边相邻的三角形的信息，边的数据结构为

基金项目：湖北省教育厅资助重点项目(2004D003)
作者简介：袁 翰(1980－)，男，硕士生，主研方向：图形处理技术，GIS；李伟波，教授；陈婷婷，硕士生
收稿日期：2006-08-02 **E-mail：**slamdunkyh@163.com

```
struct    Edge {
    int SPointID, EPointID; //起始点和终点的编号
    int EID; //边的编号
    Edge *pleft; //指向前一条边的指针
    Edge *pright; //指向后一条边的指针
    int TriNo[2]; //两相邻三角形的编号
}
```

(3)三角形链表。该链表保存生成的各个三角形的边信息和顶点信息，三角形的数据结构为

```
struct Triangle {
    int eNo[3]; //三条边的编号
    int vNo[3]; //三个顶点的编号
    int TID; //三角形的编号
    Triangle *pleft; //指向前一个三角形的指针
    Triangle *pright; //指向后一个三角形的指针
}
```

(4)凸壳边链表。链表保存凸壳的边的信息，数据结构为

```
struct Convex {
    int eNo , vNo[2]; //点和边的编号
}
```

3 TIN 三角网中的凸壳算法

构建三角网的方法可以分为 3 类：分治法，逐点插入法和三角网生长法。三角网生长算法由于算法效率相对较低，目前较少采用；分治算法[5]效率虽高，但相对复杂，且对内存要求较高，采用得也不多；逐点插入法简单而且高效，在实际的应用中具有良好的表现。因而利用凸壳特性结合逐点插入算法[1]构建三角网的格雷厄姆凸壳算法[3]是采用得比较多的，本文在此算法的基础上，考虑到角度计算比较复杂，提出了一种改进的算法。下面先介绍格雷厄姆凸壳算法，再介绍改进算法及其实现步骤。

3.1 格雷厄姆凸壳算法介绍

该算法包含以下几个步骤：

(1)找出点集中纵坐标最小的点(假设为P_1)。

(2)将P_1和点集中其他各点用线段连接，并计算这些线段与水平线的夹角。

(3)按夹角大小对数据点进行排序，如果夹角相同，则按距离排序。设得到的点序列为$P_1,P_2,\ldots,P_n$。

(4)依次连接所有点，得到一个多边行。显然，$P_1,P_2,\ldots,P_n$是凸壳边界上的点。然后，根据“凸多边形的各顶点必须在该多边形的任意一条边的同一侧”这一定理，删去边界序列中的非凸壳顶点，最后得到凸壳点集。

3.2 算法的改进

在格雷厄姆凸壳算法基础上进行了研究，得到一个改进算法，算法主要分为 5 步：(1)预处理，将点集按 x 坐标从小到大排列放入链表中，找出 x 值最小和最大的点，y 值最小和最大的点；(2)构造凸壳；(3)初始化边和三角形链表；(4)利用逐点插入法结成三角网；(5)用 *LOP* 算法对三角网进行局部优化。

(1)预处理

将点集 S 排序存入链表 VList 中，寻找出点集 S 中 x 坐标最小的点 min x，如果这样的点不唯一，则选取其中 y 坐标最小的点。再找出 x 坐标最大的点 max x，不唯一则选取 y 坐标最小的点。

同理找出y坐标最小和最大的点min y、max y，存在临时数组A_1中。如图 3 中的点(P_1,P_3,P_6,P_8)。

(2)构造凸壳

设计了一种“斜率扫描线法”，根据斜率的最大值与最小值来确定外围点，该算法实现比较简单，理解也很容易。算法从 4 个方向扫描，见图 4。

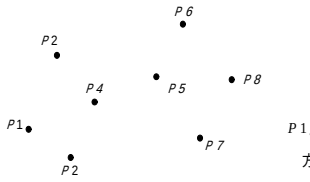

图 3 离散点集

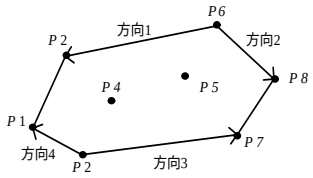

图 4 扫描方向

下面就以图 3 的 8 个离散点为例说明凸包产生过程。

第 1 步 以P_3(min y)为起始点计算线段P_3P_8的斜率$k0$ 作为初始值，找出点集链表中x值大于P_3，y值小于P_8(max x)的点(如图 4 中的点P_4，P_5，P_7)，分别计算它们与点P_3所连成的线段的斜率$k1$，若$k1$小于$k0$，则把$k1$ 的值给$k0$，并记录该点编号vID。

第 2 步 计算完成后vID中记录的就是凸壳上点(如图 4 中的点P_7)，把线段P_3P_7存入凸壳链表中CList。再以该点为起始点按第 1 步的方法查找，直到该点与点max x(P_8)重合，方向 3 的部分就扫描结束。

第 3 步 方向 4 部分同样以P_3为起点，计算线段P_3P_1的斜率$k0$ 作为初始值，找出点集链表中x值小于P_3，y值小于P_1(min x)的点，分别计算它们与点P_3所连成的线段的斜率$k1$，和第 2 步不同的是若$k1$大于$k0$，则把$k1$ 的值给$k0$，并记录该点编号vID(如图 4，若没有数据点，说明线段P_3P_1就是凸壳边，直接存入链表CList)，方向 4 部分完成。

第 4 步 方向 1、方向 2 部分再以P_6(min y)为起始点，同样按照第 1 步、第 2 步的方法类似地查找，直到达到点P_1，P_8为止，并把边存入链表中。

第 5 步 凸壳完成，输出 CList。

(3)初始化三角网

第 1 步 凸壳完成后，把边加入到边链表 EList 中，得到初始化的边链表。

第 2 步 从点集S中去掉凸壳上的点，得到一个新的点集链表$S1$，从$S1$ 中选取编号为 1 的点P_1，连接P_1与各个凸壳上的点，如图 5 中的P_4。

第 3 步 形成初始化的三角网，把增加的边加入边链表中，把新增加的三角形编号后加入到三角形链表 TList 中，并按前面说讲的数据结构把相应的拓扑关系填入链表元素中，完成三角网的初始化。

需要注意的是三角形边的存储顺序，本文选取顺时针的顺序存储。

(4)用逐点插入法构网

三角网初始化后，就可用逐点插入法，完成所有点集的构网。

第 1 步 从点集链表$S1$ 种取出第 2 个点P_2，在已存在的三角网中查找包含P_2的三角形t(如图 4 中的点P_5和三角形$\triangle P_6P_4P_8$)。

第 2 步 把P_2与t的三个顶点相连，形成t的 3 个初始三角剖分(如图 6)。

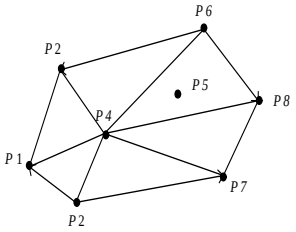

图 5 初始化三角网

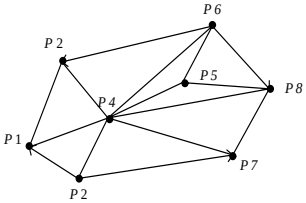

图 6 逐点插入

第 3 步 用 *LOP* 算法对初始三角剖份进行优化处理；处理外围三角形。

重复第 1 步~第 3 步，直到所有非凸壳点都插入完。

(5)*LOP* 优化算法

设新插入点P，包含P点的三角形为T，与T相邻的 3 个三角形分别为T_1、T_2、T_3。

第 1 步 把包含P的三角形的 3 个邻接三角形T_1、T_2、T_3放入堆栈中；

第 2 步 从堆栈中弹出三角形T_3；

第 3 步 检查P点是否在T_3的外接圆中，如在则交换对角线，反之弹出下一个三角形；

第 4 步 如交换对角线，则把原来与T_3相邻的两个三角形入栈；

第 5 步 重复第 2 步~第 4 步，直到堆栈为空，则优化结束。

3.3 凸壳算法流程

图 7 的流程说明了构造凸壳的过程。

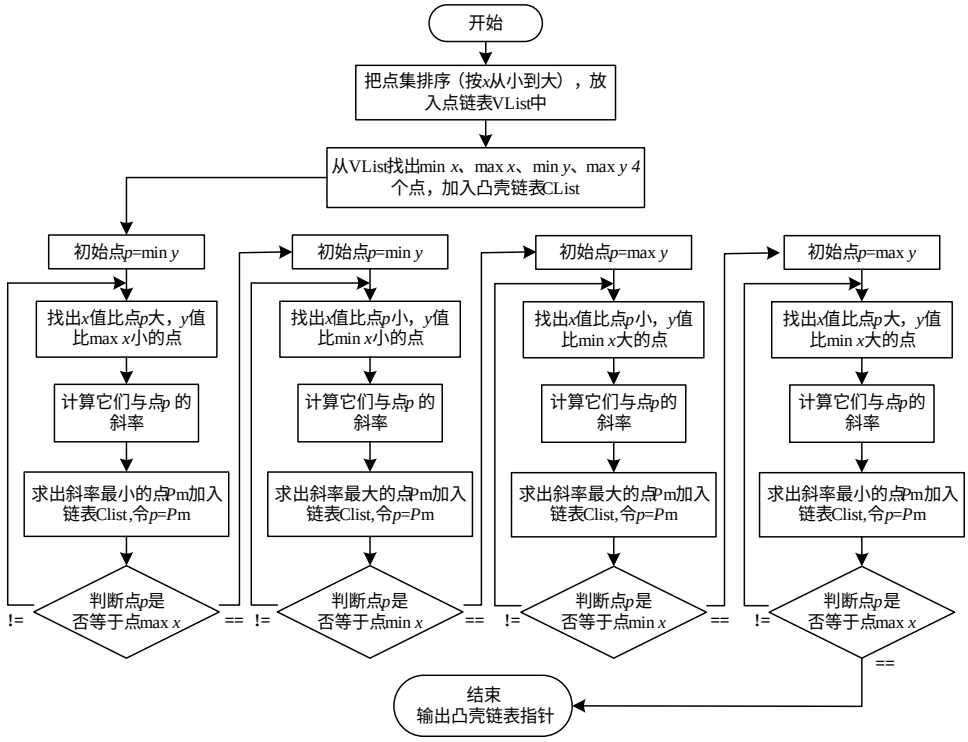

图 7 构造凸壳的过程

4 程序实现与结论

改进算法已经用 VC++6.0 在计算机上实现，通过离散点生成了凸壳(见图 8)。实验结果表明，本算法生成的 Delaunay 三角网的凸壳能得到很好的结果。

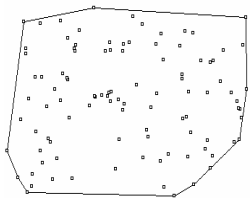

图 8 200 个离散点生成的初始凸壳

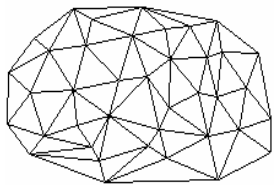

图 9 Delaunay 三角网

通过该算法再结合逐点插入法完成了 Delaunay 三角网，图 9 是 40 个离散点生成的 Delaunay 三角网。

参考文献

1 吴立新, 史中文. 地理信息系统原理与算法[M]. 北京: 科学出版社, 2005.

2 Berg M. 计算几何—算法与应用[M]. 邓俊辉, 译.北京: 清华大学出版社, 2005.

3 汤国安, 刘学军, 闾国年. 数字高程模型及地学分析的原理与方法[M]. 北京: 科学出版社, 2005.

4 陈学工, 陈树强, 王丽青. 基于凸壳技术的 Delaunay 三角网生成算法[J]. 计算机工程与应用, 2006, 42(6): 27-29.

5 戴晓明, 朱 萍. 平面散乱点三角剖分分治算法的实现[J]. 计算机技术与发展, 2006, 16(1): 11-12.

（上接第 69 页）

4 总结评价

使用 68000、8086、MCS51、TMS320C25 4 款处理器的程序目标码的反汇编结果文件进行框架分析测试，结果都能正常进行。子程序调用和被调用关系、子程序范围、子程序调用次数、是否递归调用等结果与实际人工分析的结果进行对比，基本是一致的，表明程序结构分析结果基本是准确、可靠的。为用户掌握反汇编结果程序的流程和框架结构提供了捷径，提高了用户程序解读的工作效率。测试结果如表 1。

表 1 测试结果

程序编号	处理器名字	处理器类型	程序大小(Bytes)	实际子程序数	正确划分个数	子程序划分速度(s)	划分准确率	调用关系正确率
1	68000	MCU	3 827	35	32	5.2	91.4%	94.3%
2	8086	MPU	53 723	372	353	75.8	94.8%	97.2%
3	MCS51	MCU	5 471	55	49	5.6	89.1%	93.5%
4	TMS320C25	DSP	13 256	96	87	18.6	90.6%	94.6%

本算法是基于反汇编结果文件的，算法设计中既考虑到对框架信息的抽取，也考虑了对反汇编结果文件的修改，由于需要多次扫描反汇编结果文件，因此算法所耗费的时间代价比较大。

参考文献

1 Appel A W. Modern Compiler Implementation in Java[M]. Cambridge University Press, 1998.

2 Altman E R, Kaeli D, Sheffer Y. Welcome to the Opportunities of Binary Translation[J]. Computer, 2000, 33(3): 40-45.

3 Hsieh W C, Engler D, Back G. Reverse-engineering Instruction Encodings[C]//Proc. of USENIX Annual Technical Conference, Boston, Mass. 2001-06: 133-146.

4 Larus J R, Schnarr E. EEL: Machine-Independent Executable Editing[C]//Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation. 1995: 291-300.

5 Theiling H. Extracting Safe and Precise Control Flow from Binaries[C]//Proceedings of the 7th International Conference on Real-time Computing Systems and Applications. 2000-07.