利用凸壳建 TIN的算法研究

姚圣华, 方源敏
(昆明理工大学 国土资源工程学院, 云南 昆明 650093)

摘要: 首先阐述了如何利用凸壳建 TIN的原理和方法, 并对相关算法进行了综合和改进; 然后基于“分而治之”的思想提出了一种格网数据筛选法, 用来提高凸壳建 TIN的效率; 最后通过实例测试进行比较和验证. 在三角网优化过程中, 采用 LOP优化原则, 使得建 TIN结果满足最小角最大的性质, 当平面点集满足 D-三角网构网条件时, 所得三角网为 D-三角网.

关键词: 凸壳; 格雷厄姆方法; Delaunay三角网; 不规则三角网
中图分类号: P209　文献标识码: A　文章编号: 1007-855X(2006)02-0008-06

The New Study of Establishing TIN Based on Convex Shell

YAO Sheng-hua, FANG Yuan-min
(Faculty of Land Resource Engineering, Kunming University of Science and Technology, Kunming 650093, China)

Abstract: Firstly, the principle how to establish TIN based on convex shell is described, and a series of algorithms are compounded and improved. Secondly, according to the thought of the "divide-conquer", a way to filter data in order to increase the efficiency in establishing TIN is put forward. Finally, through a series of test with real DEM data, a comparison is made under different circumstances. And in the process of optimizing the TIN, the author adopts the principle of LOP, so the TIN maximizes the minimum internal angle. If the real DEM data accords with the conditions of establishing delaunay triangulation, delaunay triangulation can be established, it can be found that the algorithms works better.

Key words: convex shell; graham method; Delaunay triangulation; triangulation irregular network

0 引 言

利用离散点数据建 TIN, 是建立 DEM的一种重要手段. 在各种建 TIN的算法中, 分治算法、逐点插入法和三角网生长法被普遍接受和采用, 而利用凸壳建 TIN的文献并不多见[1,2]. 其实, 在分治算法中, 凸壳已得到应用. 本文围绕如何利用凸壳建 TIN进行了研究, 其中如何快速建立凸壳集则是本文研究的重点. 本文利用“分而治之”的思想, 提出了一种格网数据筛选法, 大大缩短了超大点集建凸壳集所用时间, 保证了凸壳快速构网的优越性. 最后, 对比成熟的商用软件 CASS5.0给出了实例的测试结果.

本文建网时采用了 LOP优化方法, 所以建网后所得三角网满足最小角最大的性质, 并且当平面点集满足 D-三角网构网条件时, 所得三角网为 D-三角网.

1 利用凸壳建 TIN的原理及过程

利用凸壳建立三角网的基本原理为[1,2]: 将用于建网的点集划分成互不相交的凸壳, 生成凸壳集, 然后从里到外进行凸壳或凸壳间环域的剖分, 并在剖分的过程中逐渐优化, 最后得到三角网.

下面分别对建立凸壳、剖分凸壳及剖分凸壳间环域进行阐述.

1.1 凸壳的建立

考虑到平面点集建立凸壳, 本文采用格雷厄姆方法. 算法原理是: 设凸集中 y坐标最小的点为 p_1, 把 p_1

收稿日期: 2005-04-18.

第一作者简介: 姚圣华(1978~), 男, 硕士研究生. 主要研究方向: 地理信息系统. E-mail: flycat_ysh@163.com

同凸集中其它各点用线段连接, 并计算这些线段与水平线的夹角. 然后按夹角大小及到 p_1 的距离进行词典式分类, 并删除不是凸壳顶点的点, 得到一序列 $p_1, p_2, \cdots, p_m$, 依次连接这些点, 便得到凸壳多边形. 基于这一原理, 顾及多点共线的情况, 详细的算法如下:

1) 搜索所有点, 找出 y 坐标最小的点 P. 在搜索比较的过程中, 删除重复点, 并对 x, y 坐标相等的点提示用户进行取舍, 当 y 坐标相等时, 取 x 较小的点为 P.

2) 计算 P 与点集中其它点的连线和水平线的夹角, 然后按夹角从小到大排序, 并按下列规则处理角度值相等的点: 当角度不是所有角中的最大角时, 把点按其与 P 点距离从小到大排列; 反之, 按其与 P 点距离从大到小排列. 在排序的过程中, 删除重合的点. 排序时, 采用快速排序.

3) 删除 $p_3, p_4, \cdots, p_{n-1}$ 中不是凸壳顶点的点. 该步详细算法可参考文献 [1].

1.2 凸壳的三角剖分

凸壳的三角剖分是利用凸壳建 TIN 的基础和核心, 利用凸壳建 TIN 的过程实质就是不断的进行凸壳三角剖分的过程. 进行凸壳三角剖分时, 无论采用什么方法, 最基本的一条原则就是避免在剖分的过程中产生共线的三角形及共线的三角形对. 基于这一原则, 本文提出了一种利用凸壳直径的性质对凸壳进行剖分的算法, 剖分时直接将凸壳直径顶点与其前后两点组成三角形. 具体算法如下:

设 $C = \{P_1, P_2, \cdots, P_n\}$ 为一凸壳点集, $n \geqslant 3$

第一步: 求 C 的直径, 设 $l(P_i, P_j)$ 是 C 的直径.

第二步: if (C 中 n 点共线) return;

else if (C 中有 $n-1$ 个点共线, P_m 为不共线的点) then 连接 P_m 与线上各点.

else

{

连接 $P_{i-1}P_{i+1}$, 删去点 P_i, 输出 (P_i, P_{i+1}, P_{i-1}), 加入到三角形集中;

将边 $P_{i-1}P_{i+1}$ 加入边集中;

if ($j > i+1$)

{

$j = j-1$;

连接 $P_{j-1}P_{j+1}$, 删去点 P_j, 输出 (P_j, P_{j+1}, P_{j-1}), 加入到三角形集中;

将边 $P_{j-1}P_{j+1}$ 加入边集中;

}

}

第三步: 重复以上步骤, 直到 C 分割完毕.

第四步: 优化 TIN. 对以边集中的边为公共边的三角形对, 用 LOP 方法进行优化, 在优化的过程中, 处理新边的加入.

如何保证所有的三角形对都被优化, 是三角网优化的一个难点. 本文采用先局部后整体的优化原则, 先进行凸壳内的三角形对的局部优化, 后进行以凸壳各边为公共边的三角形对的整体优化. 优化时, 把要优化的边从集合中删除, 直到集合中边为空时, 优化结束. 优化时如三角形对满足交换条件, 则以该三角形对其余各边为公共边搜索三角形对进行优化, 这是一个递归的过程.

凸壳间环域的三角剖分:

凸壳间环域的三角剖分, 其本质是凸壳的剖分, 即将环域分解成多个可剖分的凸壳进行剖分. 下面分述如下:

设 convexhullinner为内凸壳顶点集, convexhullouter为外凸壳顶点集.

① convexhullinner中只有一个点的情况

记 convexhullinner中的点为 pt_inner, 顺次连接 pt_inner与 convexhullbuter中的点, 组成三角网, 并把 pt_inner与 convexhullouter中各点的连线加入边集, 然后进行三角网的局部优化. 当 convexhullouter的顶点数为 3时, 无需局部优化.

② convexhullinner和 convexhullouter均是凸多边形的情况

以 convexhullinner中的边为基础,从第一条边开始,直到最后一条边,顺次在 convexhullouter中找出位于其右边的点集,形成凸壳集并剖分优化.在生成凸壳集的过程中,处理相邻边搜寻的右凸壳相交、相离以及内凸壳多点共线的三种特殊情况时,遵循下面的原则:相交时,如图 1,分别调整两凸壳,使之以边 i或边 ii为公共边;相离时,如图 2,加入三角形 ABC到三角形链表中;内壳多点共线时,要进行凸壳的合并.如图 3,合并凸壳 4521和凸壳 4532成新的凸壳 45321.

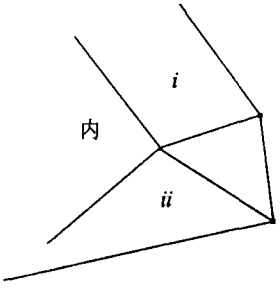

图1 两右凸壳相交
Fig.1 The two convex hull are intersectant

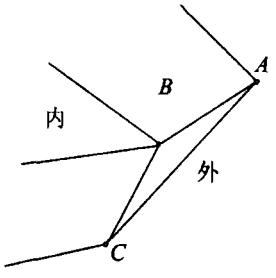

图2 两右凸壳相离
Fig.2 The two convex hull are separate

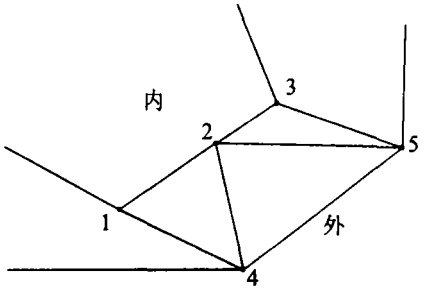

图3 内壳多点(三点)共线
Fig.3 The vertexes of the inner convex hull are collinear

③ convexhullinner中含两个点或共线的多点

设 P_1, P_2 是线的两个端点,P_L 是 convexhullouter中位于 P_1P_2 左侧的点集,P_R 是 convexhullbuter中位于 P_1P_2 右侧的点集.连接 P_L 和 P_1P_2 以及 P_1P_2 之间的点形成左凸壳,连接 P_R 和 P_2P_1 以及 P_2P_1 之间的点形成右凸壳,并参照②处理左右两个凸壳相交、相离的情况.最后分别剖分左右凸壳并优化.

利用凸壳建 TIN的过程,实际上就是从里到外不断的进行凸壳三角剖分并优化的过程.不论是分治算法、逐点插入法还是三角网生长算法,在寻找三角形顶点上面,需要对点集进行大量的判断,从而影响到建 TIN的速度.用凸壳建 TIN,利用的是从总体到局部,先控制后细部的思想[1],提高了点集建 TIN的速度;而优化条件的应用,正好与之相反先局部后整体,可以融入到建 TIN的步骤之中.这正是利用凸壳快速建各种三角网(如 D-三角网,最小权三角网等)的优势所在.

2 格网数据筛选法

通过分析不难看出,当点集扩大一倍,如以建凸壳时间复杂度为 $N^{\log}N$ 计算,时间会增加约 2倍;反之,如果将点集"分而治之",减少每次参与建凸壳的点数,可大大缩短时间.在这一思想的指导下,作者将点集用格网划分成 $M \times N$ 的数据块,每次求凸壳时,确保只有部分网格中的数据弹出到建凸壳的点集中.这样不断的从外往里缩进,直到每个网格中的数据均被弹出.这种将某些不可能成为凸壳顶点的数据弃之不用的方法,作者称之为格网数据筛选法.同时规定,如果网格中的点有可能是点集的凸壳点,那么这个网格就是临界网格.

2.1 格网划分策略

与分治算法中的数据分块目的类似,这里的格网划分是为了便于找到包含源数据凸壳的子数据集.很显然,子数据集的大小与网格的宽度和高度息息相关,而网格的宽度和高度受源数据的横向密度和纵向密度的影响.从理论上讲,平面点集越大,建凸壳所需时间越长.所以数据块划分的越小,子数据集就会越小,建凸壳就会越快;但是实际上,相反的用于临界网格搜索的时间就会越长,存储数据块相关信息的数据结构也会占用更大的内存空间,从而影响整体速度.所以如何确定网格的大小尤为重要.

本文根据地形图测量规范的要求,采用了标准十字格网划分的方法.以 1:500地形图为例,规范要求数据采集的平均间隔为 10m,那么在 50×50的标准网格中平均数据量是 25个,在实列测试中较多的采用了这种格网划分方法.当然,我们可以采用任意大小进行网格的划分.

2.2 临界网格搜索策略

根据前面临界网格的定义, 如图 4, 5所示: 很明显, 第一行, 最末行, 第一列, 最后一列中有数据的网格, 都是临界网格. 为了描述算法, 设有变量如表 1所示.

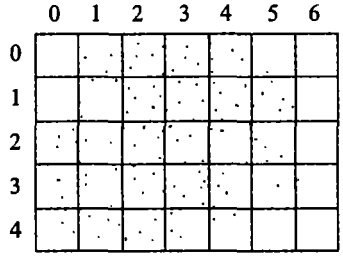

图4　格网划分数据示例1
Fig.4　The first sample of using girds to divide data

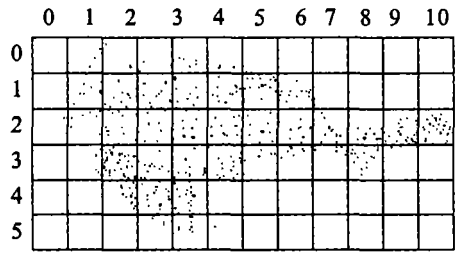

图5　格网划分数据示例2
Fig.5　The second sample of using grids to divide data

表 1　变量及其描述

Tab.1　The variables and their description

变量名	变量描述	变量名	变量描述
col_count	网格总列数	row_count	网格总行数
firstrow_first	起始行中第一个有数据的网格的列号	firstrow_last	起始行中最后一个有数据的网格的列号
lasttrow_first	终止行中第一个有数据的网格的列号	lastrow_last	终止行中最后一个有数据的网格的列号
firstcol_first	起始列中第一个有数据的网格的行号	firstcol_last	起始列中最后一个有数据的网格的行号
lasttcol_first	终止列中第一个有数据的网格的行号	lastcol_last	终止列中最后一个有数据的网格的行号

以图 4为例: col_count= 7, row_count= 5, firstrow_first= 1, firstrow_last= 5, lastrow_first= 0, lastrow_last= 4, firstcol_first= 2, firstcol_last= 4, lastcol_first= 3, lastcol_last= 3. 临界网格的搜索步骤如下:

第一步: 标记第一行, 最末行, 第一列, 最后一列中有数据的网格为临界网格.

第二步: 如果 lastrow_last小于 col_count- 1, 则在右下角搜索临界网格.

第三步: 如果 lastcol_first大于 0, 则在右上角搜索临界网格.

第四步: 如果 firstrow_first大于 0, 则在左上角搜索临界网格.

第五步: 如果 firstcol_last小于 row_count- 1, 则在左下角搜索临界网格.

下面以右下角搜索临界网格为例, 说明临界网格的搜索方法, 其它三个角上的搜索方法可依此类推. 搜索临界网格是一个循环过程, 以下用伪码语言说明其过程:

第一步: x_offset= col_count- lastrow_last- 1; y_offset= row_count- lastcol_last- 1; irow= row_count- 2; icol= lastrow_last;

第二步: if (y_offset= = 1) then标记 irow行中 icol及其后有数据网格为临界网格.

```
        else
        {
            if ( irow行中 icol及其后没有数据 ) then
            {
               int col_offset= x_offset/y_offset; x_offset-= col_offset;
               y_offset-= 1; irow-= 1;  icol+= col_offset;转步二;
            }
            else
            {
              令 ilastcol等于 irow行中有数据的最后一列的序号.
               for( i= icol; i<= ilastcol; i++ ) 标记网格 ( irow, i)为临界网格;
```

```
            x_offset--= ilastcol- icol; y_offset--= 1; icol= ilastcol;
            irow--= 1; 转步二;
        }
    }
```

如图4在右下角搜索临界网格为(3, 4)和(3, 5),在图5中为(3, 7), (3, 8), (3, 9), (2, 9)以及(2, 10). 从图上可以看出,在右下角搜索出的临界网格中,并不是每个网格中都包含凸壳的顶点,只有在某种特殊条件下才能出现右下角的临界网格都包含凸壳顶点的情形. 这种临界网格的搜索方法是一种最具包容性的临界网格法. 值得注意的是,当用以上算法对整个数据源进行第一次临界网格搜索后,后继临界网格的搜索略有不同,需要加入判断条件,以免临界网格的重复.

用格网数据筛选法进行数据的筛选过程,与凸壳的建立融为一体,且有一套严格的数据结构. 由于篇幅有限,这里不再赘述.

3 数据结构

算法的实现,借助了 Visual C++ 6.0开发环境和 AutoCAD 2000的 ObjectArx软件开发包,编译后的程序以 arx文件格式存在,能被AutoCAD 2000加载和调用. 其中数据结构参考了 Lawson 提出的标准三角网数据结构,本算法中采用的点、边、三角形数据结构如下:

点数据结构名称: CConvexhullVertex, 数据描述如表2所示.

表2 点结构数据类型及其描述

Tab. 2 The data type of point structure and their description

数据类型	变量名	用 途
双精度型数组	pt_3d	存储点的 X、Y、Z 坐标
指针数组	poslist_edge	分别指向以该点为起(终)点的边

表3 边结构数据类型及其描述

Tab. 3 The data type of edge structure and their description

数据类型	变量名	用 途
指针	pos_vfrom	指向边的起点
指针	pos_vto	指向边的终点
指针	pos_tinleft	指向边的左邻接三角形
指针	pos_tinright	指向边的右邻接三角形

边数据结构名称: CConvexhullEdge, 数据描述如表3所示.

三角形数据结构名称: CTriangle, 数据描述如表4所示.

表4 三角形结构数据类型及其描述

Tab. 4 The data type of triangle structure and their description

数据类型	变量名	用 途
指针	pos_vfirst	指向三角形的第一个顶点索引
指针	pos_vsecond	指向三角形的第二个顶点索引
指针	pos_vthree	指向三角形的第三个顶点索引
指针	pos_efirst	指向一、二两个顶点组成的边,该边为三角形的第一条边
指针	pos_esecond	指向一、三两个顶点组成的边,该边为三角形的第二条边
指针	pos_ethree	指向二、三两个顶点组成的边,该边为三角形的第三条边
指针	pos_adjtinfirst	指向与第一条边邻接的另一个三角形
指针	pos_adjtinsecond	指向与第二条边邻接的另一个三角形
指针	pos_adjtinthree	指向与第三条边邻接的另一个三角形

4 实例测试

为了考察本算法的实际运行效果,我们在 CPU为赛扬 2.4GHz,内存为 256M 的机器上,对有 6072和15 069个数据点的模拟数据集分别做了测试(据1:500地形图上 126个点的真实数据复制而得). 对应于不同的网格长宽,进行了多次运行,表5,表6分别列出了两组数据在不同情况下,运行所需要的时间,表7

则列出了两组数据在 CASS5. 0下运行所需要的时间.

表 5　第一组数据 (6072个点)在不同情况下的运行时间

Tab 5　Algorithm's performance under various grids (data included 6072 points)

网格大小 /m	建凸壳集 /s	剖分 /s	合计 /s	数据显示 /s	共计 /s	格网数据筛选法
100× 100	4	1. 5	5. 5	5. 5	11	√
50× 50	2	1	3	5. 5	8. 5	√
25× 25	2. 5	1. 5	4	5. 5	9. 5	√
30× 40	1. 5	1. 5	3	5. 5	8. 5	√
40× 30	1. 5	1. 5	3	5. 5	8. 5	√
	28	1. 5	29. 5	5. 5	35	×

表 6　第一组数据 (15069个点)在不同情况下的运行时间

Tab 6　Algorithm's performance under various grids (data included 15069 points)

网格大小 /m	建凸壳集 /s	剖分 /s	合计 /s	数据显示 /s	共计 /s	格网数据筛选法
100× 100	30	3. 5	33. 5	22	55. 5	√
50× 50	9	3. 5	12. 5	22. 5	35	√
25× 25	22	4	26	23	49	√
30× 40	9	3. 5	12. 5	22	34. 5	√
40× 30	9	3	12	22	34	√
	276	3. 5	279. 5	22	291. 5	×

表 7　两组数据在 CASS5 0下运行所需要的时间

Tab 7　The performance of the two array data in CASS5. 0

数　据	运行时间 /s	备　注
第一组 (6072)	15	运行时间包括建
第二组 (15069)	78	网和显示三角形

通过对比发现, 格网数据筛选法的应用, 大大改善了算法的时间性能, 说明这种方法在理论和实际上可行有效. 其次, 网格长宽与运行所需要的时间以某一点为转折, 分别成正负相关的关系. 在网格规格为 50× 50时, 效率较好, 这也间接说明上述的网格划分策略有一定的合理性. 当然, 同样的数据放在其它比例尺的图形中, 则上面的格网划分策略就不适用, 所以, 网格的长宽主要还是取决于源数据的密度, 上面的划分策略仅能参考. 最后, 通过与 CASS5. 0进行比较, 在数据量大时本算法的综合表现明显优于 CASS5. 0中采用的算法.

在对两组数据进行测试可知, 数据量越大, 建凸壳所用的时间越占主导地位, 如何快速建立点集的凸壳集, 是今后算法改进的重点. 另外, 在对第一组数据进行 5 000 × 5 000的网格划分时, 建凸壳集的过程非常缓慢, 时间远远大于不用格网数据筛选法时所用时间. 其原因是运用格网数据筛选法所用的辅助数据占用了巨大的内存空间导致总体性能降低. 这也从一个侧面反映了算法优劣的关键, 不能仅仅从理论时间复杂度来评价, 对时间和空间的合理综合应用[4]以及其在时间上的表现才是一个算法优劣的标志.

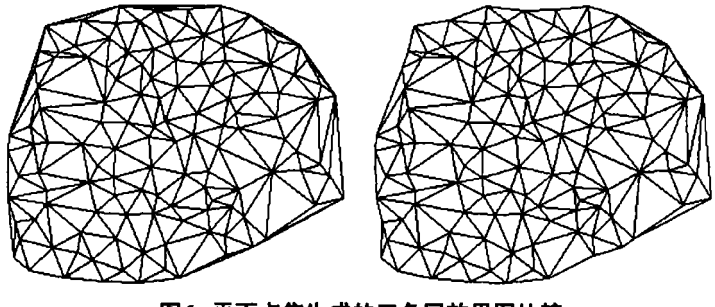

图6　平面点集生成的三角网效果图比较,
图左三角网基于本算法生成, 图右三角网由CASS5.0软件生成

Fig.6　The sample of TIN established by different algorithm.The left TIN is established based on convex shell and the right one is established in CASS5.0

(下转第 18页)

位的解决方案,在实践中也得到了实际应用,本文提出了一种不同于 Trimble的 VRS数学模型,虽然没有在实际中应用,但作为一种理论方法,可以从不同角度对 VRS进行深入研究,为 VRS的理论研究提供一定的参考.

参考文献:

[1] 刘大杰,施一民,过静珺.全球定位系统(GPS)的原理与数据处理[M].上海:同济大学出版社,1996:97-116.
[2] 刘基余.GPS卫星导航定位原理与方法[M].北京:科学出版社,2003:334-363.
[3] 武汉大学测量平差教研室.测量平差基础(第三版)[M].北京:测绘出版社,1996:83-86.
[4] 李成钢.基于多基站网络的 VRS技术及其误差分析与建模[M].成都:西南交通大学,2003:32-34.
[5] LIW ei,DAIE.A study on GPS/GLONASS Multiple Reference Station Technique for Precise Real-Time Carrier Phased-Based Positioning[M], University ofWales New South, Sydney, NSW, Australia,2000:2-6.
[6] 陈本富,李军杰,施昆.稳健估计在参心与 GIS地心坐标间相互转换的应用[J].昆明理工大学学报:理工版,2005,30(3):11-13.

(上接第 13页)

总的来说,本算法达到了快速建 TIN的目的.同时,通过对比 CASS 5.0,也表明建网结果正确.如图 6所示,图左边的三角网基于本算法所建,图右三角网由 CASS 5.0软件生成.可以看到,图左边的三角网过滤掉满足一定条件(如角度太小,最长边与最小边比值过大等)的三角形后与图右的三角网完全一致.

5 总结与讨论

由于 TIN,特别是 D-三角网的独特性质,使之成为 2.5维分析处理的主要工具.经过 20多年的研究与实践,各种算法的优劣得到了人们的共识.本文对用凸壳建 TIN的方法进行了研究,对凸壳剖分的算法进行了综合和改进,并基于“分而治之”的思想提出了一种格网数据筛选法,极大地提高了利用凸壳建 TIN的速度.经过测试,该算法表现出色,达到了快速建 TIN的目的.

本文所讨论的算法仅只针对一般三角网,对于没有约束条件(山脊线,谷地线)的三角网,有一定的参考和应用价值.

参考文献:

[1] 程效军,孙晋岳.基于凸壳的 TIN建立技术[J].东北测绘,2001,24(1):6-9.
[2] 周培德.平面点集三角剖分算法[J].计算机辅助设计与图形学,1996,8(4):259-264.
[3] 周培德.求凸壳顶点的一种算法[J].北京理工大学报,1993,13(1):69-72.
[4] 武晓波,王世新.Delaunay三角网的生成算法研究[J].测绘学报,1999,8(1):28-35.
[5] 徐青,常耿.基于自适应分块的 TIN三角网建立算法[J].中国图象图形学报,2000,05A(6):461-465.