基于 Delaunay 三角网的等值线绘制算法 *

蒋 瑜, 杜 斌, 卢 军, 王 鹏
(成都信息工程学院 软件工程学院, 成都 610225)

摘 要: 提出了一种快速构建 Delaunay 三角网算法(QGDTN)。在每次迭代中,该算法从点集 P 最左边的两点中,选取离凸边中点距离最近的一点与凸边构成 Delaunay 三角形,并加入三角网中,算法实现简单,且时间复杂度为 $O(n)$。基于 Delaunay 三角网,根据三角形的各边上是否有等值点,用内插值法求出等值点坐标,跟踪、连接等值点生成等值线;最后,采用三次方 Bezier 曲线平滑等值线。实验证明,基于 Delaunay 三角网的等值线绘制算法是高效的,并且具有一定的实用价值。

关键词: 等值线; Delaunay 三角网; LOP 优化; Bezier 曲线
中图分类号: TP391; P207 **文献标志码**: A **文章编号**: 1001-3695(2010)01-0101-03
doi:10.3969/j.issn.1001-3695.2010.01.030

Algorithm of drawing isoline based on Delaunay triangle net

JIANG Yu, DU Bin, LU Jun, WANG Peng
(*College of Software Engineering, Chengdu University of Information Technology, Chengdu* 610225, *China*)

Abstract: This paper proposed a new algorithm for quick generation Delaunay triangle net. In iterations, this algorithm selected a point from the leftmost two points in point set P, and the distance between this point and midpoint of convex edge was minimal. This point and convex edge constructed new Delaunay triangle, and added them to Delaunay triangle net. The average time complexity of the algorithm was $O(n)$. Based on Delaunay triangle net, computed the coordinate of equivalent points according to using interpolating method if there were equivalent points in each edge of triangles. Tracing and drawing equivalent points created isolines. At last, smoothed isolines based on cubic Bezier curve. Experiments results show the algorithm of drawing isoline based on Delaunay triangle net are high efficiency, and have some practical value.

Key words: isoline; Delaunay triangle net; LOP optimizing; Bezier curve

0 引言

等值线研究是科学计算可视化的一个基础而重要的内容。由于等值线图(如温度、降水分布图等)看起来直观、形象,它在地球科学、气象学、医学等许多学科中有着广泛的应用。等值线绘制算法主要由绘制等值线、平滑等值线两部分组成。而绘制等值线是实现等值线绘制算法的基础和重点。在实际应用中,基于三角网和规则格网绘制等值线的方法使用较多,其中三角网方法具有高精度、高效率和易于处理地性线(如等高线)等特点,并且规则格网数据的生成一般是在构建三角网的基础上经过内插得到,因此,构建三角网是绘制等值线的关键。其中,由于 Delaunay 三角网具有良好的形态,在表达地表形态方面表现较为出色,得到普遍认可。

如何快速、高效地构建 Delaunay 三角网,一直是众多学者研究和关注的焦点。迄今为止出现了不少成熟的算法,如分割—合并算法[1~3]、逐点插入法[4~6]及三角网生长法[7~10]等。其中三角网生长算法由于算法效率较低,目前较少采用;逐点插入法虽然实现较简单,占用内存较小,但其时间复杂度差,效率较低;分割—合并算法最为高效,但相对复杂,由于其深度递归,对内存要求较高。它们的平均时间复杂度分别有 $O(n)$、$O(n\log n)$和 $O(n^2)$。

1 Delaunay 三角网构件算法

1.1 算法基本思路

本算法的基本思路是向三角网中不断添加新三角形,在添加三角形的同时,采用 Lawson 提出的 LOP 方法优化与新三角形相邻三角形组合成的四边形,使优化后的三角形满足空外接圆、最小角最大化的属性。本文中三角网构建方法如下:按照点在二维空间上的位置从左到右、从上到下将点连入三角剖分,每连入一个点就生成与该点相关的左侧三角形。在剩余点集的最左侧两点中,选择与三角形剖分的凸包边中点距离最短的点连入,即只有那些满足当前点位于其右侧的凸包边才可能与当前点生成三角形。假定点集中最左侧的若干个点已连成一个外部边界为凸包的三角剖分 T,且所有三角形均为逆时针方向(如图 1 所示,三角形 $\triangle ACB$ 的三个顶点逆时针顺序为 A、B、C),则三角剖分的外边界 D 也为逆时针方向的边序列。再假设下一个待连入的点为 P(需保证 P 为剩余点中位于最左侧的两点之一),可以证明边界线中只有满足某个条件的边与点 P 构成的三角形才能不相交于 T 中的任何三角形。这里需满足的条件为点 P 必须位于感兴趣边界线的右侧,且点 P 到该感兴趣边界线中点的距离最短。因此,只要待连入点 P 与所有满足条件的边界线连成三角形,则所有三角形构成的三角剖分形成了这些点所在空间的一个覆盖,且其外边

收稿日期: 2009-04-07; **修回日期**: 2009-05-25 **基金项目**: 国家自然科学基金资助项目(60702075);成都信息工程学院院选自然科学基金资助项目(CSRF200702)

作者简介:蒋瑜(1980-),男,四川邻水人,讲师,硕士,主要研究方向为数字图像处理、数据挖掘与粗糙集理论(jiangyu@cuit.edu.cn);杜斌,男,副教授,主要研究方向为软件工程;卢军,男,副教授,主要研究方向为软件工程;王鹏,男,副教授,主要研究方向为并行计算和数据挖掘.

界必为凸边。

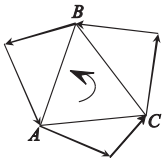

图1　三角形△*ACB*顶点逆时针顺序

1.2　算法的具体实现

QGDTN 算法的具体实现步骤如下：

a)先将待构网的离散点集按横坐标升序、纵坐标升序排列，这时各点在点集中的顺序即是空间上从左到右、从上到下的排列。

b)接下来由最左侧的两个点 *A*、*B*，连成一条边 *AB*。

c)选择剩余点集中最左边的两个点，取到边中点距离最短的点，假设为 *C* 点，从而点 *C* 与边 *AB* 构成三角形，并确保它的顶点为逆时针排列(假设为 *A*、*B*、*C*)。

d)将凸包边(为有向边，其指向与三角形顶点的排列顺序一致，如边 *BC*、*CA*)存储在一个集合中。

e)取出剩余点集中最左侧的两点 *P* 和 *Q*，遍历凸包边集合，找出符合条件的一条边，这里的条件为 *P* 或 *Q* 位于边的右侧。

f)当 *P* 和 *Q* 同时位于边的右侧时，选择距边中点最近的一点；否则，选择位于该边右侧的一点。不妨设这步选择的点为 *R*，该边与点 *R* 构成新三角形，LOP 优化该边所关联的两三角形，且从凸包边集合中删去该边，因为它已不再是凸包边集中的边；该边的两顶点与 *R* 的连线形成了两条新边，如果是凸包边(即该边的反向边不在凸包边集合中)，则加入到凸包边集合中，否则，若凸包边集合中含有该新边的反向边，这时不仅新边不加入凸包边集合，还要从集合中删去其反向边。

g)基于点 *R*，遍历凸包边集合，找出符合条件的边，这里的条件为 *R* 位于边的右侧。

h)将 g)中找出的所有边与点 *R* 构成新三角形，LOP 优化每边所关联的两三角形，且从凸包边集合中删去这些边；每条边的两顶点与 *R* 的连线分别形成了两条新边，如果是凸包边则加入到凸包边集合中，否则，凸包边集合中含有与新边的反向边，从集合中删去其反向边。

i)重复步骤 e)～h)中，直至点集为空。

1.3　算法事例分析

基于 QGDTN 算法，给定如图2 所示的 *a*，*b*，*c*，*d*，*e*，*f*，*g* 七个点，Delaunay 三角网构建过程如下：

a)将 *a*、*b*、*c*、*d*、*e*、*f*、*g* 七个点按横坐标升序、纵坐标升序排列，如图 2(a)所示。

b)连接 *a*、*b* 两点如图 2(b)所示。

c)选取 *c*、*d* 两点，由于 *c* 点到边 *ba* 中点距离最短，*c* 点与 *ba* 构成新三角形(如图 2(c)所示)，且边 *ab*、*bc* 和 *ca* 加入凸包边集合。

d)选取 *d*、*e* 两点，查找到凸边 *ca*，且 *d*、*e* 都位于凸边右侧，由于 *d* 点距离凸边 *ca* 中心距离最短，*d* 点与 *ca* 构成新三角形(如图 2(d)所示)，且 LOP 优化 *bca* 和 *cda* 两三角形，在凸边集合中删除 *ca*，且边 *cd*、*da* 加入凸边集合中。

e)基于点 *d*，搜索凸边集合，边 *bc* 满足点 *d* 位于其右侧，点 *d* 与边 *bc* 构成新三角形(如图 2(e)所示)，且 LOP 优化 *bca* 和 *bdc* 两三角形，在凸边集合中删除边 *bc*，且边 *bd* 加入凸边集合中，由于边 *dc* 在凸边集合中有其反向边 *cd* 存在，*dc* 不加入凸边集合，且从凸边集合中删除 *cd* 边。

f)选择点 *e*、*f*，搜索凸边集合，边 *bd* 满足条件，且 *e*、*f* 都位于边 *bd* 的右边，且点 *e* 到边 *bd* 中点距离最短，点 *e* 与边 *bd* 构成新三角形(如图 2(f)所示)，且 LOP 优化 *bdc* 和 *bed* 两三角形(如图 2(g)所示)，在凸边集合中删除边 *bd*，且边 *be*、*ed* 加入凸边集合中。

g)选择点 *f*、*g*，搜索图边集合，边 *be* 满足条件，且点 *f*、*g* 都位于边 *be* 的右边，且点 *g* 到边 *be* 中点距离最短，点 *g* 与边 *be* 构成新三角形(如图 2(h)所示)，且 LOP 优化 *bed* 和 *bge* 两三角形，在凸边集合中删除边 *be*，且边 *bg*、*ge* 加入凸边集合中。

h)选择点 *f*，搜索凸边集合，边 *bg* 满足条件，即点 *f* 位于边 *bg* 右边，点 *f* 与边 *bg* 构成新三角形(如图 2(i)所示)，且 LOP 优化 *bge* 和 *bfg* 两三角形，在凸边集合中删除边 *bg*，且边 *bf*、*fg* 加入凸边集合中。

i)点集为空，算法结束。

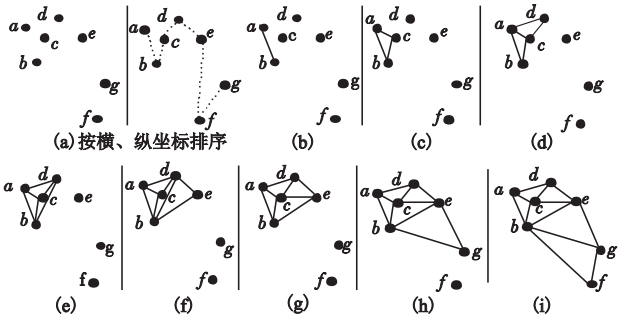

图2　离散的七个点

1.4　算法及算法复杂度分析

本算法在实现过程中有两大优点：a)在候选的两个点集中，选择离满足条件的第一条凸包边中心距离最短的一点，这样有力地保证了最小角最大化的属性，有效减少了 LOP 优化次数；b)在新生成三角形的同时，采用 LOP 优化该凸包边所关联的两三角形，这就避免了以往算法在生成三角网后采用 LOP 优化的搜索过程。

QGDTN 算法的时间复杂度主要由两次搜索和 LOP 优化的时间构成。第一次搜索即点在横、纵向上的排序，第二次搜索为三角网的生成过程。设 n 为离散点个数，则用于排序上的平均时间复杂度为 $O(n\log_2 n)$(快速排序)；LOP 优化是伴随三角网生成的过程，而 LOP 优化仅仅是个判断过程，不需要搜索。设 LOP 优化一个三角形的时间复杂度为 $O(r)$，与三角网的生成算法相比，排序和 LOP 优化只有比较操作，耗时较少。下面主要讨论用于三角网生成过程的平均时间复杂度。设单位横向距离内纵向排列的离散点数为 m(近似为常数)，这样每个新点与凸包边构造三角形的个数不超过 m(新生边数为 $3m$)，即生成新三角形和边的时间复杂度为 $O(m)$，LOP 优化的时间复杂度为 $O(r\times m)$；设凸包边集中边数为 N，在构网过程中每生成一个三角形还需要搜索凸包边集一趟，搜索过程的平均时间复杂度为 $O(N/2)$，且随着三角网的增大，N 也会缓慢增大。所以用于三角网生成过程和 LOP 优化过程的最大运算次数分别约为 $m\times(N/2)\times n$ 和 $m\times r\times n$。当点数 n 很大时，N、m 和 r 远小于 n，近似为常数，所以三角网生成过程平均时间复杂度为 $O(n)$。QGDTN 算法总的平均时间复杂度为 $O(n)$，并且由上面的讨论可知，本算法最坏情况下的时间复杂度也为 $O(n)$。

1.5　算法效率测试

基于 Microsoft Visual C ++ 6.0 平台，实现了本文 QCDTN 算法，并在 CPU 为 Pentium 1.7 GHz、内存为 256 MB 的笔记本中，基于随机离散点构建了 Delaunay 三角网。表1 中列出了本文算法及近年来发表的 Delaunay 三角网多种快速生成算法的执行效率比较。

图3　基于QGDTN算法构建的Delaunay三角网

表 1　几种 Delaunay 三角网算法效率比较

算法	硬件平台	语言	离散点数/s
QCDTN	Pentium 1.7 GHz,256 MB	C ++	约 9 500
文献[10]	Pentium Ⅳ 2 GHz,256 MB	C ++	约 9 000
文献[11]	Pentium Ⅲ 733 MHz,256 MB	C ++	约 5 000
文献[12]	Pentium Ⅲ 1.7 GHz,256 MB	C ++	约 1 000

2　等值线的绘制及平滑

2.1　绘制等值线

基于 QGDTN 算法所构建的三角网,在三角网中三角形的边上进行插值。根据三角形的各边上是否有等值点,用内插法求出等值点的坐标。插值公式为

$$\frac{x_1-x}{x_1-x_2}=\frac{val_1-val}{val_1-val_2}\Rightarrow x=x_1-(x_1-x_2)\times\frac{val_1-val}{val_1-val_2}$$

$$\frac{y_1-y}{y_1-y_2}=\frac{val_1-val}{val_1-val_2}\Rightarrow y=y_1-(y_1-y_2)\times\frac{val_1-val}{val_1-val_2}$$

其中:(x_1,y_1)、(x_2,y_2)为三角形一边上的两个顶点坐标;val_1为(x_1,y_1)点的数据资料值;val 为等值线线值;(x,y)即为所求点的坐标。经过插值后得到等值点及其相对应的坐标,通过计算可以得出相对于某个值总共有多少个等值点,然后根据所连接的三角网中找出等值点连接(图 4)。

2.2　Bezier 曲线平滑等值线

基于插值法所连成的等值线,其实是一条条折线, 为了使一条条等值线看起来连续而平滑。本文采用三次方 Bezier 曲线平滑等值线(图 5)。三次方 Bezier 曲线表述如下:

P_0、P_1、P_2、P_3 四个点在平面或在三维空间中定义了三次方 Bezier 曲线。曲线起始于 P_0 走向 P_1,并从 P_2 的方向来到 P_3,一般不会经过 P_1 或 P_2,这两个点只是在那里提供方向资讯。P_0 和 P_1 之间的间距,决定了曲线在转而趋近 P_3 之前,走向 P_2 方向的长度有多长。曲线的参数形式为

$$B(t)=P_0(1-t)^3+3P_1t(1-t)^2+3P_2t^2(1-t)+P_3t^3,t\in[0,1]$$

图4　基于QGDTN算法构建的三角网,采用插值法所绘制的等值线

图5　三次方Bezier曲线平滑等值线图

三次 Bezier 曲线的 C 语言代码如下:

```
/*cp 在此为四个元素的点阵:cp[0]为起点,或上边的 P0、cp[1]为第一控制点,或上边的 P1,cp[2]为第二控制点,或上边的 P2、cp[3]为结束点,或上边的 P3、t 为参数值,0⇐t⇐1,Point2D 为有两个整型变量的结构体*/
Point2D PointOnCubicBezier( Point2D* cp, float t )
{
   float ax, bx, cx;
   float ay, by, cy;
   float tSquared, tCubed;
     Point2D result;
     //计算多项式系数
   cx = 3.0 * (cp[1].x - cp[0].x);
   bx = 3.0 * (cp[2].x - cp[1].x) - cx;
   ax = cp[3].x - cp[0].x - cx - bx;
   cy = 3.0 * (cp[1].y - cp[0].y);
   by = 3.0 * (cp[2].y - cp[1].y) - cy;
   ay = cp[3].y - cp[0].y - cy - by;
   //计算位于参数值 t 的曲线点
   tSquared = t * t;
   tCubed = tSquared * t;
   result.x = (ax * tCubed) + (bx * tSquared) + (cx * t) + cp[0].x;
   result.y = (ay * tCubed) + (by * tSquared) + (cy * t) + cp[0].y;
   return result;
}
void ComputeBezier ( Point2D * cp, int numberOfPoints, Point2D * curve )
{
   float dt;
   int i;
   dt = 1.0 / ( numberOfPoints - 1 );
   for( i = 0; i < numberOfPoints; i++)
     curve[i] = PointOnCubicBezier( cp, i*dt );
}
```

3　结束语

本文提出了一种新的构建 Delaunay 三角网算法——QGDTN(quick generation Delaunay triangle net)算法,其时间复杂度接近线性。该算法思路简单,易于实现,避免了现有算法的不足。基于该算法所构建的 Delaunay 三角网,在三角网中三角形的边上进行插值。根据三角形的各边上是否有等值点,用内插法求出等值点的坐标,跟踪、连接等值点生成等值线;最后,采用三次方 Bezier 曲线平滑等值线,使等值线连续而平滑。

参考文献:

[1] GUIBAS L J, STOLFI J. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams[J]. ACM Trans on Graphics, 1985,4(2):74-123.

[2] KATAJAINEN J, KOPPINEN M. Constructing Delaunay triangulations by merging buckets in quad tree order[J]. Ann Soc Math Polon Ser Ⅳ Fund Inform , 1988,11(3):275-288.

[3] 胡金星,潘懋,马照亭,等. 高效构建 Delaunay 三角网数字地形模型算法研究[J]. 北京大学学报:自然科学版,2003,39(5):736-741.

[4] 武晓波,王世新,肖春生. 一种生成 Delaunay 三角网的合成算法[J]. 遥感学报,2000,4(1):32-35.

[5] 刘学军,符锌砂,赵建三. 三角网数字地面模型快速构建算法研究[J]. 中国公路学报,2000,13(2):31-36.

[6] 宋占峰,蒲浩,詹振炎. 快速构建 Delaunay 三角网算法研究[J]. 铁道学报, 2001,23(5):85-91.

[7] 祝志恒,傅鹤林,蒲浩,等. 构件 Delaunay 三角网的一种新型生长法——壳外插入法[J]. 铁道科学与工程学报,2007,4(6):67-72.

[8] 徐青,常歌,杨力. 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, 2000,5A(6):461-465.

[9] 刘永和,王燕平,齐永安. 一种快速生成平面 Delaunay 三角网的横向扩张法[J]. 地球信息科学,2008,10(1):20-25.

[10] 何俊,戴浩,谢永强,等. 一种改进的快速 Delaunay 三角剖分算法[J]. 系统仿真学报,2006,18(11):3055-3057.

[11] 曾闽山,田冬玲,郭吉民. 一种基于格网划分的高效 Delaunay 三角网格化算法[J]. 微计算机信息, 2006,22(9):127-129.

[12] 方勇,刘鹏,胡海彦. 一种 Delaunay 三角网的快速生成算法[J]. 测绘科学与工程, 2006,26(3):1-4.