

文章编号:1001-9081(2010)01-0075-03

基于 Delaunay 三角剖分生成 Voronoi 图算法

孙继忠¹, 胡 艳², 马永强¹

(1. 西南交通大学 信息科学与技术学院, 成都 610031; 2. 西南交通大学 理学院, 成都 610031)

(jizhongsun@qq.com)

摘要:针对 Delaunay 三角网生长算法和间接生成 Voronoi 图算法构造效率不高的问题,提出了一种 Delaunay 三角网生长法间接生成 Voronoi 图的改进算法。该算法以点集凸壳上一边快速生成种子三角形,定义了半封闭边界点的概念,在三角形扩展过程中动态删除封闭点及半封闭边界点,加快 Delaunay 三角网生成速度。然后又定义了有序目标三角形的概念,该算法能迅速查找点的有序目标三角形,生成无射线的 Voronoi 图;考虑凸壳上点的特性,借助三个无穷点生成带射线的 Voronoi 图。通过实验结果分析表明,改进的算法执行效率有了很大提高。

关键词: Delaunay 三角剖分; Voronoi 图; 凸壳; 计算几何

中图分类号: TP391.41 **文献标志码:** A

Voronoi diagram generation algorithm based on Delaunay triangulation

SUN Ji-zhong¹, HU Yan², MA Yong-qiang¹

(1. School of Information Science and Technology, Southwest Jiaotong University, Chengdu Sichuan 610031, China;

2. Department of Mathematics, Southwest Jiaotong University, Chengdu Sichuan 610031, China)

Abstract: Considering the problem that the algorithm of building auto-connected Delaunay triangulation and indirectly building Voronoi diagram is of low efficiency, an improved Voronoi generation algorithm based on auto-connected Delaunay triangulation was presented. The seed triangle was rapidly generated by one side of the convex hull. The notion of half closed-border-point was proposed. The algorithm removed closed-points and half closed-border-points in the process of expanding triangle and improved the speed of generating Delaunay triangulation. Then, the notion of ordered target triangle was defined. It quickly found ordered target triangles and generated the non-ray Voronoi diagram. Considering the characteristics of convex hull, a ray Voronoi diagram was generated by three infinite points. The experimental results show that the efficiency of the improved algorithm is obviously improved.

Key words: Delaunay triangulation; Voronoi diagram; convex hull; computational geometry

0 引言

Voronoi 图与 Delaunay 三角形、Convex hull 并列为计算几何的三大支柱,它已成功解决了找最近点、求最大空圆、求 N 个点的凸包、求最小生成树等问题,在计算机图形学、地理信息系统、模式识别等有广泛的应用^[1-2]。Voronoi 图与 Delaunay 三角剖分互为对偶图,则可以通过 Delaunay 三角剖分间接生成 Voronoi 图。Delaunay 三角剖分具有最大化最小角、外接圆准则等重要性质。文献[3]根据实现过程,把生成 D-三角网的各种算法分为三类:分治算法、逐点插入法、三角网生长法。而三角生长法基本上都在寻找“第三点”上进行改进。例如 McCullagh 和 Ross^[4]通过把点集分块和排序改进了点搜索方法,减少了搜索时间。文献[5]中提出了封闭点概念对三角网生长算法作出了一些改进,如果凸壳上的点数过多,影响第三点的搜索速度,算法效率会降低。由 Delaunay 图构造 Voronoi 图,通常做法是遍历所有三角形,找到与该点相关的三角形。如果点集数比较多,每次都遍历一次所有三角形,搜索效率比较低;其次不能有效区分对凸壳内部点与凸壳边界点生成 Voronoi 图。为此,针对以上问题提出了一种 Delaunay 三角剖分间接生成 Voronoi 图的改进算法,该算法能提高构造 Delaunay 三角剖分速度,对凸壳内部点与凸壳边界

点进行有效识别与处理,同时提高构造 Voronoi 图的速度。

1 基本概念

在构造 Delaunay 图过程中,动态剔除封闭点和半封闭边界点,可以减少第三点的搜索时间,提高生成新三角形的速度。所以在文献[5]封闭点概念的基础上,定义半封闭边界点:

定义 1 不共线点集 $S = \{v_0, v_1, v_2, \dots, v_n \mid n \geq 2\}$, 假设 v 为点集 S 凸壳上的一点,与 v 相连接的点集合 $E(v) = \{p_1, p_2, \dots, p_m\}$ 。如果存在 $E(v)$ 不为空, $p_1vp_2, p_2vp_3, \dots, p_{m-1}vp_m$ 按空间顺时针或逆时针均已形成三角形,且 p_1, p_m 为凸壳上的点及 vp_1, vp_m 为凸壳上的边,则 v 为半封闭边界点。

为了提高 Voronoi 图的构造效率,定义有序目标三角形。

定义 2 p_0 为目标点,如果三角形 T 的三个顶点中包含点 p_0 , 则称 T 与 p_0 相关,记为 $R(p_0, T)$ 。显然若多个三角形 T_i 与点 p_0 相关,记为 $R(p_0, T_0, T_1, \dots, T_n)$ 。把 T_0, T_1, \dots, T_n 按空间位置顺时针或逆时针排序为 T_0', T_1', \dots, T_n' , 则称 $R(p_0, T_0', T_1', \dots, T_n')$ 为关于 p_0 的有序目标三角形。

由于算法的实现和绘图的需要,定义如下数据结构。

定义 3 用于存储平面上散乱点的数组为 pointArray, 构成 Delaunay 三角网的初始化点集链表为 PointList, 构成

收稿日期:2009-07-18;修回日期:2009-08-24。

作者简介: 孙继忠(1982-),男,河南驻马店人,硕士研究生,主要研究方向:计算机网络; 胡艳(1984-),女,河南驻马店人,硕士研究生,主要研究方向:智能信息处理; 马永强(1963-),男,福建福州人,教授,博士,主要研究方向:计算机网络、信息处理、并行计算、图像检测。

Voronoi 图点集链表为 PointList_V。定义了如下三个类:1)点类,包括点编号 ID、点坐标 (x, y) 、是否为凸壳上的点 huSign、该点使用次数是否为 2 clSign、与该点相关的边集 edgeList;2)边类,与该点相连的有向边终点编号 ID、边的使用次数 nCount;3)三角形类,三角形的编号 ID、点指针数组 * index [3]、三角形是否扩展 enlarge。

2 Delaunay 三角剖分的生成

改进算法构造 Delaunay 三角剖分的基本思想:先快速生成包含点集的凸壳,利用凸壳上的一边快速创建满足三角剖分的种子三角形,然后对该种子三角形的一边进行扩展,生成新的三角形,循环扩展所有新三角形新生成的两边,同时删除封闭点和半封闭边界点,直到三角形都扩展完毕。其过程如图 1 所示。

算法的关键是:1)如何快速创建凸壳。2)快速创建种子三角形。选凸壳上的一边进行扩展可省去确定第二点的时间。3)如何快速搜索第三点,搜索第三点关键在于删除封闭点和半封闭边界点,减少点的搜索时间。

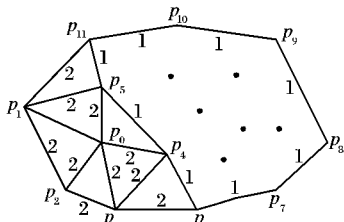


图1 三角剖分构成过程

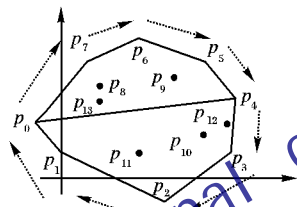


图2 点集凸壳生成过程

2.1 点集的预处理

初始化点集 S 的数组 pointArray、点集 S 的链表 PointList、三角形链表 TriList、Delaunay 三角剖分后点集 S 的链表 PointList_V、三角形个数 $Tricount = 0$ 。

2.2 凸壳的生成

构建散乱点集 S 的凸壳 $CH(S)$ 的方法,通常可以分为两类:一类是基于对点集排序的基础上,时间复杂度主要取决于排序时间,如格雷厄姆扫描法、卷包裹法。另一类不用预先进行排序,如增点递推算法,其时间复杂度为 $O(2hn)^{[7]}$ 。本文凸壳算法引用文献[6],不需要排序,时间复杂性为线性,执行效率比较高。

步骤 1 求点集 S 中, x 坐标最大时 y 坐标也最大、 x 坐标最小时 y 坐标也最小所对应的点,设为 $p_0 (x_{\min}, y_{\min})$ 、 $p_4 (x_{\max}, y_{\max})$ 如图 2 所示。对点集求区间 $L = (x_{\min}, x_{\max})/k (k = x_{\max} - x_{\min} \text{ 为正整数})$,划分点集 S 所在区域为 k 个长条,计算 S 中各点所在长条的编码,位于第 i 条的点构成 $S_i \circ S = \bigcup_{i=1}^k S_i$ 。在线段 $p_0 p_4$ 之间的点,计算 S_i 中点的 y 坐标最大、最小值的点,设为 $p_{y_{\max}}^i, p_{y_{\min}}^i, i = \overline{1, k}, S_i^* = \{p_x, p_{y_{\max}}^i, p_{y_{\min}}^i\}$ 。按 x 坐标值升序把 S_i^* 中 $p_{y_{\max}}^i$ 依次加入链表 LiskYmax。同理,按 x 坐标值降序把 S_i^* 中 $p_{y_{\min}}^i$ 依次加入链表 LiskYmin。

如图 2,经过处理后,除横坐标为 x_{\min}, x_{\max} 的点外,分为 $S_+ = \{p_5, p_6, p_7, p_8, p_9\}, S_- = \{p_1, p_2, p_3, p_{10}, p_{11}\}$ 。

步骤 2 把长条区间 S_i^* 中 $p_{y_{\min}}^i, p_{y_{\max}}^i$ 非重复点的依次加入链表 LiskHull,然后依次以 Listmax 中的点为极坐标顶点,求 Listmax 中此点以后基于 x 轴正方向斜率最大的点加入 ListHull 中,直到链表 LiskYmax 遍历到最后一个点为极坐标顶点,把最后一个点加入 ListHull 中。如图 2 所示,以 p_0 为极坐标顶点,求 p_0 以后链表 LiskYmax 中基于 x 轴正方向最大斜率的点为 p_7 ,把 p_7 加入 LiskHull。同理依次把 p_6, p_5, p_4 加入 ListHull 中。

步骤 3 把长条区间 S_i^* 中 y 坐标最小的点非 p_4 点加入链表 LiskHul。分别以链表 ListYmin 中的点为极坐标顶点,在链表 LiskYmin 中求最大斜率的点,直到链表 ListYmin 遍历到最后一个点为极坐标顶点,把最后一个点加入 ListHull 中。如图 2 所示,分别把 p_3, p_2, p_1 加入 LiskHull 中。

步骤 4 链表 LiskHull 即为点集凸壳上的点。

2.3 Delaunay 三角网生成

步骤 1 对凸壳上的点及边链表更新。凸壳上的点 huSign = TRUE,点的有向边链表使用次数赋值为 1。

步骤 2 创建种子三角形。如图 1 所示,以 $p_1 p_2$ 为一边,在链表 PointList 中寻找点 p_0 ,使角 $\angle p_1 p_0 p_2$ 最大,创建种子三角形,加入到链表 TriList 中,三角形计数 $Tricount$ 加 1,更新点的边链表及边的使用次数分别加 1。

步骤 3 扩展种子三角形第一边。如图 1 所示 $p_1 p_2$ 边,边的使用次数为 2 就不再扩展。

步骤 4 循环扩展所有三角形。循环扩展三角形的链表 TriList 中新生成的两条边,直到链表 TriList 中三角形遍历完为止,执行步骤 5。

寻找第三点应满足:1)与该点相关的边的使用次数不全为 2;2)与该边所对应该点的角度在未扩展点集中最大;3)当前扩展三角形的另一顶点分布在当前扩展边的两侧。

边的扩展:若边的使用次数小于 2 时扩展,等于 2 不扩展。

生成的新三角形加入到三角形链表 TriList 中, $Tricount$ 次数加 1,更新点的边链表以及边的使用次数加 1。而每个新生成三角形有向边为 6,均加 1。当点成为封闭点和半封闭边界点时,删除此点,加快第三点的搜索速度。

由文献[5]和定义 1 可以判断封闭点与半封闭边界点:如果点的边链表使用次数均为 2 且该点不是凸壳上的点,则该点为封闭点,如图 1 的 p_0 点。凸壳上的点且点的边链表使用次数均为 2 时,则该点为半封闭边界点,如图 1 p_1 点。所以删除封闭点和半封闭边界点判别条件统一为:判断点的边链表使用次数是否均为 2。

步骤 5 三角形链表 TriList 即为所求三角剖分。

3 Voronoi 图生成

3.1 间接生成 Voronoi 图分析

由 Delaunay 三角网构造 Voronoi 图关键寻找每个点的有序目标三角形。而寻找有序目标三角形定位的常规方法^[8]是遍历所有的三角形,找到与该目标点相关的所有三角形。对有序目标三角形的圆心按空间位置进行顺时针或逆时针排序,然后依次连接圆心生成该点 Voronoi 图。遍历所有点,就生成点集 S 的 Voronoi 图。如果点集 S 很大,则生成的三角形的总数很多,如果每个点都遍历一次三角形链表,然后排序生

成 Voronoi 图,则花费的时间较多,从而影响了 Voronoi 图的生成速度。针对以上问题,改进算法思想是快速定位每个点的有序目标三角形,顺序连接三角形外接圆的圆心,生成 Voronoi 图。该算法同时考虑凸壳边界点的特性,对生成带射线 Voronoi 图与不带射线 Voronoi 图进行详细分析。

3.2 快速确定有序目标三角形

由定义3可以得到点和边的类数据结构,加上点集数组 pointArray,我们可以快速确定有序目标三角形。如图3所示,点 p_0 的边链表可以确定点 $\{C_0, C_1, C_2, C_3, C_4, C_5\}$ 的ID号,在点的数组 pointArray 可以确定坐标,然后把 p_0 的边链表中的点按顺时针或逆时针排序,可以确定 p_0 的有序目标三角形 $R\{P_0, T_0, T_1, T_2, T_3, T_4, T_5\}$ 。

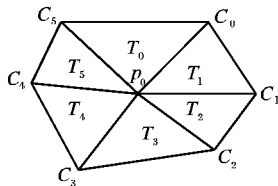


图3 有序目标三角形

3.3 带射线的 Voronoi 图与无射线的 Voronoi 图

带射线 Voronoi 图与无射线 Voronoi 图的区别关键是凸壳上的点的处理。对凸壳内部的点,二者处理都是一样。

只处理凸壳内部的点时,生成不带射线的 Voronoi 图,如图4所示:点 K_0, K_1, K_2, K_3 为初始点集 S , A_1, A_2, A_3 分别为构造 Delaunay 三角剖分后外接圆的圆心,虚线表示不带射线的 Voronoi 图。考虑 Delaunay 图与 Voronoi 图的关系,借助相对于点集所在区域中最大 X, Y 值无穷大以及最小 X, Y 值无穷小中的三个辅助点参与三角剖分,三个辅助点与其外接圆圆心相连彼此近似 120° 即可。显示时,凡是包含三个辅助点中任意一点的 Delaunay 三角剖分都不显示,而包含其中两个点的三角形的外接圆的圆心不参与 Voronoi 图的构成。如图5所示添加三个辅助点 w_1, w_2, w_3 ,构成 Voronoi 图的过程,得到红方框内的带射线的 Voronoi 图,如图6所示。

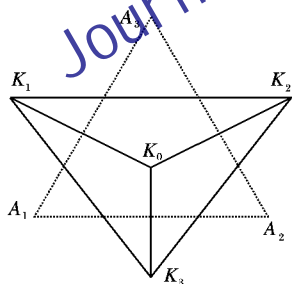


图4 不带射线的 Voronoi 图

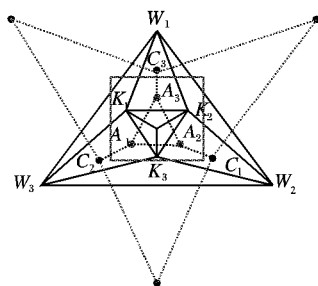


图5 添加三个无穷点构成 Voronoi 图过程

3.4 改进的间接生成 Voronoi 图算法

这里只阐述不带射线的 Voronoi 图生成过程,带射线 Voronoi 图需要添加三个辅助点参与 Delaunay 图构成即可。

步骤1 在 Delaunay 生成过程中,用点集链表 PointList_

V 添加删除的封闭点和半封闭边界点,然后把链表 PointList 中剩余的点添加到 PointList_V 中。

步骤2 遍历点链表 PointList_V。

判断当前点为 p_0 是否为空,若不空执行1),否则结束链表的遍历。

1)若 p_0 为凸壳上的点(半封闭边界点),不参与 Voronoi 图构成,执行5),否则执行2)。

2)凸壳内部的点(封闭点),遍历该点的边链表 edgeList,边链表保存有向边终点ID号,利用点集数组 pointArray,求得各点坐标。

3)把各点坐标按顺时针或逆时针排序,排序后生成有序目标三角形 $R(p_0, T_0, T_1, \dots, T_n)$,计算各三角形的外接圆的圆心。

4)顺时针或逆时针依次连接圆心生成该点 Voronoi 图。

5)遍历下一个节点。

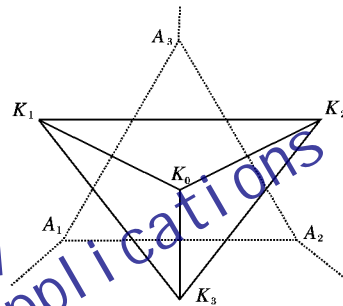


图6 带射线的 Voronoi 图

4 实现的结果与分析

本文算法采用C++语言编程,用VC++6.0编译环境分别实现了原算法及改进的算法。计算机的配置为 Intel CPU 1.80 GHz(2 CPUs),1024 MB 内存。在随机产生没有约束的1000个离散点后,生成凸包,如图7所示。生成细线为 Delaunay 三角剖分和粗线为不带射线 Voronoi 图,如图8所示。

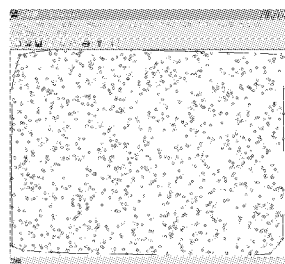


图7 初始化点集和生成凸壳

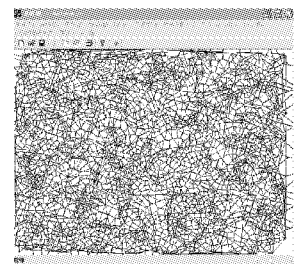


图8 Delaunay 图和 Voronoi 图

为了比较算法改进前后的效率,分别用1000、2000、3000、4000、5000和10000个随机离散点,在没有约束的情况下,分别对生成 Delaunay 三角网和生成不带射线 Voronoi 图分阶段进行构网,实验结果对比如表1、2所示。

表1 Delaunay 三角网算法改进前后对比

点数	凸壳上 点数	三角形数	耗时/s	
			文献[5]算法	改进后算法
1000	17	1981	0.538	0.451
2000	24	3970	1.582	1.473
3000	26	5965	3.396	3.076
4000	21	7975	5.939	5.043
5000	23	9971	9.059	7.078
10000	29	19960	35.091	26.822

(下转第97页)

Contourlet 变换在捕获丰富的纹理方向方面具有更强的优势。

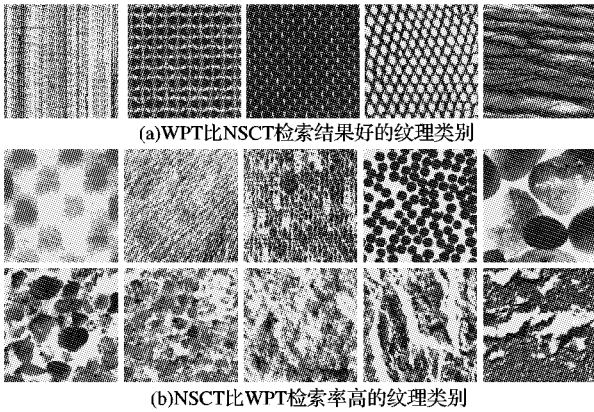


图 5 基于 NSCT 和 WPT 的纹理检索结果比较

5 结语

本文提出了一种新的纹理图像检索方法,主要思想为利用非下采样 Contourlet 变换对纹理图像进行分解,提取变换系数的均值和方差作为特征向量,采用不同的相似度计算方法计算待查询图像和数据库图像之间的相似程度,建立了基于示例查询图像的纹理图像检索系统。为验证本文方法的有效性,本文分别采用了小波包变换和非下采样 Contourlet 变换两种特征提取方法,Euclidean 距离和 Bhattacharyya 距离两种相似度计算方法,进行了对比实验。实验结果表明,利用非下采样 Contourlet 变换提取纹理特征使平均检索正确率有了明显提升,采取不同的相似度计算公式也会给检索结果带来影响。然而,利用 NSCT 变换的纹理特征提取速度较慢,下一步的工作是研究如何提高运算速度。同时,如何更有效地表征纹理图像之间的相似度还有待进一步研究。

参考文献:

[1] 庄越挺,潘云鹤,吴飞.网上多媒体信息分析与检索[M].北京:

清华大学出版社,2002:27-48.

- [2] MANJUNATH B S, MA W Y. Texture features for browsing and retrieval of image data [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1996, 18(8): 837-842.
- [3] LIAPIS S, TZIRITAS G. Color and texture image retrieval using chromaticity histograms and wavelet frames [J]. IEEE Transactions on Multimedia, 2004, 5(6): 676-686.
- [4] DO M N, VETTERLI M. Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance [J]. IEEE Transactions on Image Processing, 2002, 2(11): 146-158.
- [5] KOKARE M, BISWAS P K, CHATTERJI B N. Texture image retrieval using new rotated complex wavelet filters [J]. IEEE Transactions on Systems, Man, and Cybernetics, 2005, 35(6): 1168-1178.
- [6] LIVENS S, SCHEUNDERS P, van de WOUWER G, et al. Wavelets for texture analysis, an overview [C]// 6th International Conference on Image Processing and Its Applications. Dublin: Institution of Electrical Engineers, 1997, 2: 581-585.
- [7] PO D D-Y, DO M N. Directional multiscale modeling of images using the Contourlet transform [J]. IEEE Transactions on Image Processing, 2006, 15(6): 1610-1620.
- [8] 杨家红,许灿辉,王耀南.基于 Contourlet 广义高斯模型的纹理图像检索[J].中国图象图形学报,2007,12(4):691-694.
- [9] da CUNHA A L, ZHOU J P, DO M N. The nonsubsampled contourlet transform: Theory, design, and applications [J]. IEEE Transactions on Image Processing, 2006, 15(10): 3089-3101.
- [10] da CUNHA A L, ZHOU J, DO M N. Nonsubsampled contourlet transform: Filter design and applications in denoising [EB/OL]. [2009-04-10]. http://www.ifp.illinois.edu/~minhdo/publications/nsctdenoise_icip.pdf.
- [11] 梁栋,殷兵,于梅,等.基于非抽样 Contourlet 变换的彩色图像数字水印算法[J].光学学报,2008,28(8):1469-1474.
- [12] DO M N, VETTERLI M. Texture similarity measurement using Kullback-Leibler distance on wavelet subbands [C]// Proceedings of 2000 International Conference on Image Processing. Washington, DC: IEEE, 2000, 3: 730-733.

(上接第 77 页)

由表 1 可以看出改进的 Delaunay 构网算法,总体上效率高于文献[5]中的算法,点数较少时,二者相差很小,当点数较多时,改进算法优势较明显。由表 2 所示,在 Delaunay 三角剖分基础上生成 Voronoi 图,随着点数和三角形的增多,改进的算法较原算法的效率提高就越明显。表 2 的原算法之所以比表 1 生成 Delaunay 三角网的时间还要高,主要原因是随点数增多和生成 Delaunay 三角形增多,每次遍历寻找有序目标三角形,花费时间较大。

表 2 D-图直接生成 V-图算法改进前后对比

点数	三角形数	耗时/s	
		原算法	改进后算法
1 000	1 981	0.228	0.038
2 000	3 980	0.755	0.074
3 000	5 961	2.867	0.107
4 000	7 974	7.214	0.113
5 000	9 970	12.417	0.124
10 000	19 958	49.916	0.169

5 结语

Delaunay 三角剖分间接生成 Voronoi 图的改进算法,无论是在 Delaunay 三角网构造阶段还是在 Voronoi 图构造阶段,

效率较原算法均有所提高,特别是在寻找点集的有序目标三角形效率非常高,但间接构造 Voronoi 图算法仍依赖于 Delaunay 三角形的构网速度。

参考文献:

- [1] AURENHAMMER F. Voronoi diagrams: A survey of a fundamental geometric data structure [J]. ACM Computing Surveys, 1991, 23(3): 345-405.
- [2] 刘金义,刘爽. Voronoi 图应用综述 [J]. 工程图学学报, 2004, 25(2): 125-132.
- [3] BRASSEL K E, REIF D. Procedure to generate thissen polygons [J]. Geographical Analysis, 1979(11): 289-303.
- [4] McCULLAGH M J, ROSS G T. Delaunay triangulation of a random data set for is arithmetic mapping [J]. The Cartographic Journal, 1980(17): 93-99.
- [5] 凌海滨,吴兵,改进的自连接 Delaunay 三角网生成算法 [J]. 计算机应用, 1999, 19(12): 10-12.
- [6] 周培德. 计算几何: 算法设计与分析 [M]. 3 版. 北京: 清华大学出版社, 2008.
- [7] 张忠武,吴信才. 平面海量散乱点集凸壳算法 [J]. 计算机工程, 2009, 35(9): 43-45.
- [8] 刘少华,罗小龙,何幼彬,等. 基于 Delaunay 三角网的泰森多边形生成算法研究 [J]. 长江大学学报, 2007, 4(1): 100-103.