一种改进的 Voronoi 图增量构造算法

孟 雷 张俊伟 王筱婷 杨承磊
(山东大学计算机科学与技术学院, 济南 250101)

摘 要 Voronoi 图是计算几何中的一种重要几何结构,也是计算几何的重要研究内容之一,如今已经在图形学、地理信息系统、机械工程、机器人等领域得到广泛应用。增量法是最常用的构造 Voronoi 图的方法,但一般实现方法中点的定位时间比较长。扫描线算法可以视为一种特殊的增量法,时间复杂度为 $O(n\log n)$,但需要构造比较复杂的数据结构。为了更有效地构建 Voronoi 图,提出了一种改进的 Voronoi 图增量构造算法,该算法是通过对已有的生成 Voronoi 图的增量法进行分析,并结合它们的优点,采用扫描线的方式,通过右凸链的结构来定位新插入的点,实现了 Voronoi 图的逐步构造。和扫描线算法类似,其时间复杂度为 $O(n\log n)$,但算法更简洁,且便于理解和编程实现。

关键词 Voronoi 图 Delaunay 三角化 增量法 扫描线法 右凸链
中图法分类号: TP391.41 **文献标志码**: A **文章编号**: 1006-8961(2010)06-978-07

An Improved Incremental Algorithm for Voronoi Diagram

MENG Lei, ZHANG Junwei, WANG Xiaoting, YANG Chenglei
(*School of Computer Science and Technology, Shandong University, Ji'nan* 250101)

Abstract Voronoi diagram(VD) is a very important geometry structure and an important research topic in computational geometry. It has been widely applied in computer graphics, GIS, machine engineering and robotics and so on. Incremental algorithm is one of the most popular algorithm to construct VD. To find the location of a new insertion site is a key problem and usually costs lots of time. Sweep-line algorithm can be seen as a special incremental algorithm, which spends $O(n\log n)$ time to locate a insertion point. In this paper, an improved incremental algorithm for constructing the VD of a planar point set is presented. A new data structure named right convex hull chain is used to find the location of a new input site in $O(n\log n)$ time. Compared with other incremental algorithms, this algorithm can also run in $O(n\log n)$ time, but it's simpler, more comprehensible and easier to implement.

Keywords Voronoi diagram(VD), Delaunay triangulation, incremental algorithm, sweep-line algorithm, right convex hull chain

0 引 言

Voronoi 图(VD)是计算几何中的一种重要的几何结构,其可按照对象(如点、线段等)集合中元素的最近属性将空间划分成许多单元区域。如今,VD 及其对偶图 Delaunay 三角网格(DT)(见图 1)已在图形学、地理信息系统、机械工程、机器人等领域得到广泛应用。

目前已开展了很多构造 VD 的工作[1],但构造 VD 的算法时间复杂度的下限为 $O(n\log n)$,其中 n 为给定的平面上离散点集的数目。第 1 个 $O(n\log n)$算法为分而治之算法[2]。但由于该算法实现细节比较复杂等原因[3],因而应用较少。增量法是最常用的构造 *VD* 的方法,但一般实现方法中点的定位时间比较长。扫描线算法可以视为一种特

基金项目:国家自然科学基金项目(60703028,60573181)
收稿日期:2009-07-01;**改回日期**:2009-10-30
第一作者简介:孟 雷(1987—),男。大学本科学生,本科科研助手。研究方向为计算机图形学。E-mail: menglei.thunder@gmail.com

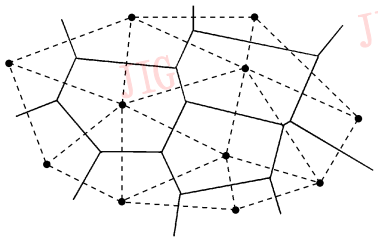


图 1　平面点集(黑点)的 VD(实线)及其 Delaunay 三角化(虚线)

Fig. 1　The VD(solid lines) and Delaunay triangulation(dash lines) of a set of sites(black dots)

殊的增量法,虽然其时间复杂度为 $O(n\log n)$,但需要构造比较复杂的数据结构。

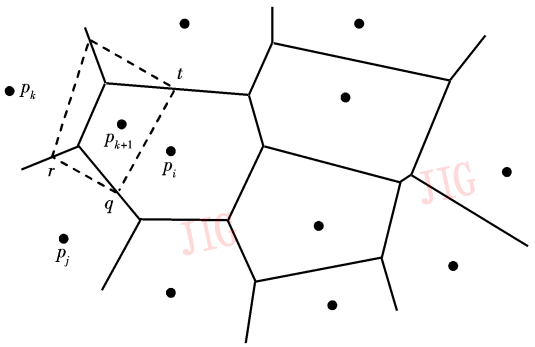
本文通过对已有的生成 VD 的增量法进行分析,结合它们的优点,采用扫描线的方式,通过本文提出的右凸链的结构来定位新插入的点,从而实现了 VD 的逐步构造。和扫描线算法类似,其时间复杂度为 $O(n\log n)$,但本文算法更简洁,且便于理解和编程实现。

1　相关工作

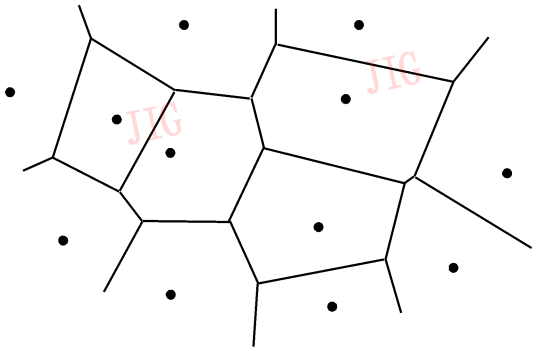
增量法是最常用的一种构造 VD 的算法。该算法的基本思想是:在已构造的输入了种子点(site)的 VD 的基础上,逐步加入新的种子点,再利用局部特性,通过局部修改已有的 VD 来快速生成新的 VD,即对于平面离散的种子点集合 $\boldsymbol{P}=\{p_1,p_2,\cdots,p_n\}$,在种子点集合 $\boldsymbol{P}_k=\{p_1,p_2,\cdots,p_k\}$ $(k<n)$ 的 VD 的基础上,再加入新的种子点 p_{k+1},然后通过局部修改来构造 $\boldsymbol{P}_{k+1}=\{p_1,p_2,\cdots,p_k,p_{k+1}\}$ 的 VD。如此不断加入新的种子点,最后即可得到点集 $\boldsymbol{P}$ 的 Voronoi 图 $\boldsymbol{VD}(\boldsymbol{P})$。

增量法主要涉及以下两个步骤:

1) 定位　查询新加入的点 $\boldsymbol{p}_{k+1}$ 在 $\boldsymbol{VD}(\boldsymbol{P}_k)$ 中的位置,即点 p_{k+1} 所属的 Voronoi 区域 $\boldsymbol{VR}_p(p_i)$ $(1\leqslant i\leqslant k)$,其中,$\boldsymbol{VR}_p(p_i)$ 表示在 $\boldsymbol{VD}(\boldsymbol{P}_k)$ 中种子点 p_i 所对应的 Voronoi 区域(VR)。如图 2(a)所示,可通过将点 p_{k+1} 与 $\boldsymbol{P}_k=\{p_1,p_2,\cdots,p_k\}$ 中每个种子点的距离进行比较,若可以找到离其最近的种子点 p_i,即可确定点 p_{k+1} 落在 $\boldsymbol{VR}_p(p_i)$ 中。也可从任一种子点和点 p_{k+1} 相连,然后顺着连线向点 p_{k+1} 方向沿一个个的 VR 找过去,直至找到点 p_{k+1} 所在的 $\boldsymbol{VR}_p(p_i)$ 为止。另一种方法是从任一 VD 的顶点开始,顺着指向最靠近点 p_{k+1} 的 Voronoi 边,一条接一条边地前进并向点 p_{k+1} 靠拢,直至走到包含点 p_{k+1} 的 $\boldsymbol{VR}_p(p_i)$ 的边为止。



(a) 定位 p_{k+1},计算其 VR



(b) 修改后的 VR

图 2　加入一个新种子点 p_{k+1} 后,更新 VD 的过程

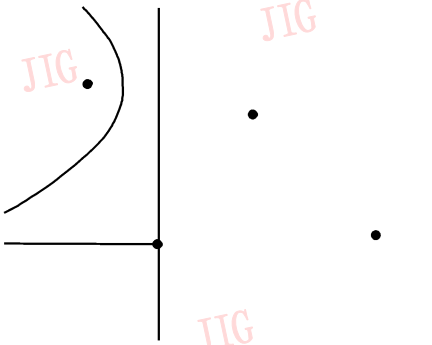
Fig. 2　After adding p_{k+1}, compute new VD

2) 局部修改　由于点 p_{k+1} 的加入,在它周围形成了一个由离点 p_{k+1} 最近的点组成的 Voronoi 区域 $\boldsymbol{VR}_p(p_{k+1})$,它是从附近的 VR 中分割出来的(如图 2(a)的虚线多边形)。因点 p_{k+1} 落在 Voronoi 区域 $\boldsymbol{VR}_p(p_i)$ 中,所以分割就从 $\boldsymbol{VR}_p(p_i)$ 开始。用点 p_{k+1} 和点 p_i 的垂直平分线把 $\boldsymbol{VR}_p(p_i)$ 分成两部分,使每部分的点分别离点 p_{k+1} 和点 p_i 最近,它和 $\boldsymbol{VR}_p(p_i)$ 的边界交于 t 和 q 两点。在点 q 处开始进入 Voronoi 区域 $\boldsymbol{VR}_p(p_j)$,这时 $\boldsymbol{VR}_p(p_j)$ 中也有一部分点和点 p_{k+1} 最近,因此可用点 p_{k+1} 和点 p_j 的垂直平分线把 $\boldsymbol{VR}_p(p_j)$ 分成两部分,该线与 $\boldsymbol{VR}_p(p_j)$ 的边界交于 r 和 q 两点。用点 r 代替点 q 重复上述工作,直到有一条垂直平分线通过点 t,形成一个多边形为止(参见图 2(a)的虚线多边形)。再把该多边形中间的边和顶点删除,便形成了新的 Voronoi 区域 $\boldsymbol{VR}(p_{k+1})$(如图 2(b)所示)。

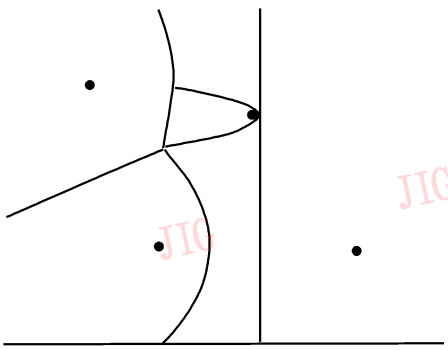
上述第 1 步,点的定位时间比较长,可高

达$O(n)$。

扫描线算法可以视为一种增量法，可在$O(n\log n)$时间和$O(n)$空间内构造出整个 VD[4-6]。扫描线算法主要用一条垂直扫描线l自左而右扫过平面上的点集，扫描过程中不断修改维护扫描线l左侧种子点的 VD(如图 3 所示)。当扫描完所有种子点时，则最后可得到整个种子点集的 VD。该算法的难点在于：$\boldsymbol{VD}(\boldsymbol{P})$位于扫描线$l$左侧部分的结构，不仅取决于扫描线$l$左侧的种子点，而且也取决于扫描线$l$右侧的某些种子点。由于扫描线$l$到达某个种子点$p_i$要晚于到达其 VD，即扫描线$l$到达$\boldsymbol{VR}_p(p_i)$最左端的$VD$的顶点时，还未探测到点$p_i$的存在，所以，在应用扫描线方法时，需要借助比较复杂的数据结构。另外，对于给定的平面上一致分布的点集，文献[7]给出了$O(n)$时间的构造 VD 的算法。



(a) 访问到第 2 个种子点时的 VD



(b) 访问到第 3 个种子点时的 VD

图 3 扫描线法构造 VD 过程中，访问到第 2 个种子点和刚刚访问过第 3 个种子点时 VD 的状态

Fig. 3 The constructed VD when the second site is scanned and after the third one has been scanned

本文对增量法进行了改进，给出一种简单、更易于理解和编程的方法。它主要也是由点定位和局部修改 VD 两部分组成。在定位过程中，引入了右凸链概念，基于此可简单、快速定位插入点的位置。

2 右凸链与点定位

2.1 右凸链

设$\boldsymbol{Q}=\{q_1,q_2,\cdots,q_n\}$为一个凸多边形，$l$是位于$\boldsymbol{Q}$右侧的一条垂直扫描线(如图 4 所示)。$\boldsymbol{Q}$的外部 VD 如图 4 中虚线部分所示，是一个特殊的 VD：其 Voronoi 边位于多边形的外部，且和其相关联的凸多边形$\boldsymbol{Q}$的边垂直于该边的顶点；多边形的一条边或一个顶点的 VR 只有两条边，且该区域是开放的[8]。

需要注意的是，本文中提到两类 VD，一类是本文要构造的点集的 VD，在构造过程中，会不断生成新的 VD；另一类是凸多边形的 VD，用于点的定位，起辅助作用。本文中的凸多边形是当前已扫描点集的凸包，其上面的点p_i的外部 VR 用$\boldsymbol{VR}_o(p_i)$表示(下角 o 代表 out)。

凸多边形$\boldsymbol{Q}$中，那些外部 VR 和扫描线l相交的顶点和边构成的多边形链，本文称之为关于扫描线l的右凸链(RCHC)。如图 4 所示，多边形链$\{q_3,q_4,q_5\}$为关于扫描线l的一条右凸链。显然，右凸链关于l是严格单调的，即垂直于l的任一直线与右凸链只有一个交点。

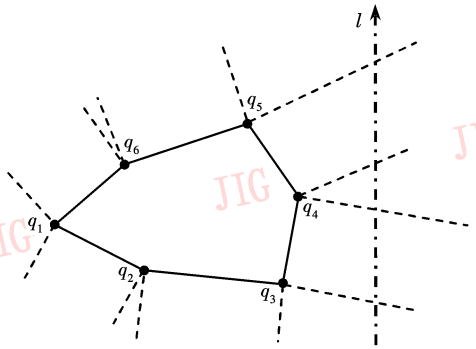


图 4 $\boldsymbol{Q}=\{q_1,q_2,\cdots,q_6\}$的关于$l$的右凸链

Fig. 4 A RCHC of $\boldsymbol{Q}=\{q_1,q_2,\cdots,q_6\}$ about l

2.2 点的定位

本文定义的 RCHC 主要用于快速定位新扫描到的种子点q的位置，即对于已扫描种子点集的 VD，q属于哪一个 VR。由于 RCHC 的顶点或边的外部 VR 的边是依次和扫描线l相交的(如图 4 所示)，所以可以用二分法找到一个顶点p_i(或一条边p_ip_{i+1})，点q包含在$\boldsymbol{VR}_o(p_i)$(或$\boldsymbol{VR}_o(p_ip_{i+1})$)中。这个过程用时为$O(\log n)$。

图 5 中的多边形链$p_1p_2\cdots p_6$是关于扫描线l的

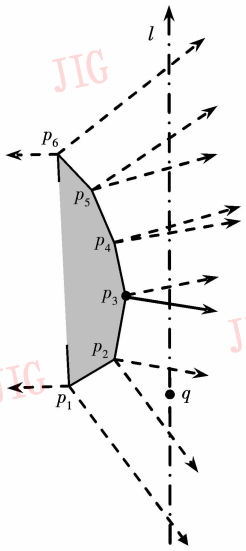


图 5　基于右凸链查找包含 q 的 Voronoi 区域

Fig. 5　Find the VR containting q based on the RCHC

右凸链,点 q 是扫描线 l 上的一点。由于点 q 在 $p_1p_2\cdots p_6$ 的中点 p_3 的外部 Voronoi 区域 $\boldsymbol{VR}_o(p_3)$ 的边的下面,因此接下来只需继续用二分法搜索多边形链 $p_1p_2p_3$,最后即可发现,点 q 位于 $\boldsymbol{VR}_o(p_2)$ 中。

下面需要证明的是,如果点 q 位于 $\boldsymbol{VR}_o(p_2)$ 中,则点 q 必位于 $\boldsymbol{VR}_p(p_2)$ 中。

假设点集 $\boldsymbol{P}=\{p_1,p_2,\cdots,p_n\}$ $(n\geqslant 3)$ 位于垂直扫描线 l 的左侧,其 Voronoi 图 $\boldsymbol{VD}(\boldsymbol{P})$ 已构造出来。另设点 $v_1,v_2,\cdots,v_m$ $(m\leqslant n)$ 是点集 $\boldsymbol{P}$ 的凸包上的顶点,且多边形链 $v_1v_2\cdots v_m$ $(m\leqslant n)$ 是关于垂直扫描线 l 的右凸链;点 q 是扫描线 l 上一点。

定理 1　如果点 q 位于 $\boldsymbol{VR}_o(v_j)$ $(1\leqslant j\leqslant m)$ 中,则点 q 必位于 $\boldsymbol{VR}_p(v_j)$ 中。

证明　在 RCHC $v_1v_2\cdots v_m$ $(m\leqslant n)$ 的外部 VD 中,由于点 q 位于 $\boldsymbol{VR}_o(v_j)$ $(1\leqslant j\leqslant m)$ 中,所以点 v_j 是 RCHC $v_1v_2\cdots v_m$ 中离点 q 最近的点。

由于点 v_{j-1},v_j,v_{j+1} 是 $v_1v_2\cdots v_m$ $(m\leqslant n)$ 上 3 个连续的顶点,$v_1v_2\cdots v_m$ 是点集 $\boldsymbol{P}$ 的凸包的一部分,所以在整个点集 $\boldsymbol{P}$ 的 VD 中,点 v_{j-1} 与点 v_j,点 v_j 与点 v_{j+1} 之间都存在 Voronoi 边。其中,点 v_{j-1} 与点 v_j 之间的 Voronoi 边与 $\boldsymbol{VR}_o(v_j)$ 的一条垂直于 $v_{j-1}v_j$ 的 Voronoi 边平行,点 v_{j+1} 与点 v_j 之间的 Voronoi 边与 $\boldsymbol{VR}_o(v_j)$ 的一条垂直于 $v_{j+1}v_j$ 的 Voronoi 边平行。由于点 v_j 是点集 $\boldsymbol{P}$ 的凸包的一顶点,且 $\boldsymbol{VR}_p(v_j)$ 是开放的,因此可知,点 q 必位于 $\boldsymbol{VR}_p(v_j)$ 中(如图 6(b)所示)。证毕。

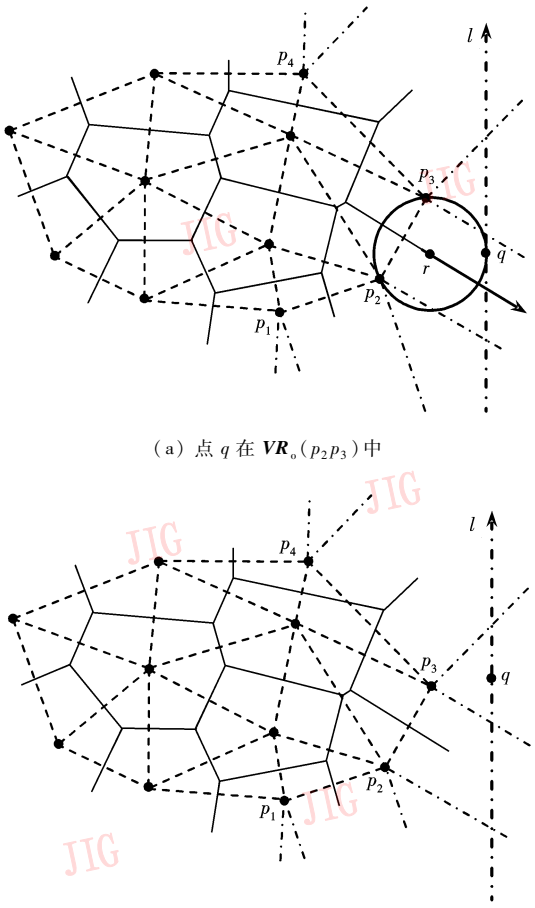


图 6　VD($\boldsymbol{VD}(\boldsymbol{P}\cup q)$)中与 q 有关的 Voronoi 边

Fig. 6　Voronoi edges about q in $\boldsymbol{VD}(\boldsymbol{P}\cup q)$

如果一旦确定点 q 位于 $\boldsymbol{VR}_p(v_j)$ 中,则可知在 $\boldsymbol{P}\cup q$ 的 VD 中,点 q 和点 v_j 之间必存在 Voronoi 边。

定理 2　如果点 q 位于 $\boldsymbol{VR}_o(v_jv_{j+1})$ $(1\leqslant j<m)$ 中,在 $\boldsymbol{P}\cup q$ 的 VD 中,点 q 和点 v_j,点 q 和点 v_{j+1} 之间分别存在 Voronoi 边。

证明　设 $\Delta v_iv_jv_{j+1}$ 是 $\boldsymbol{VD}(\boldsymbol{P})$ 的对偶图—Delaunay 三角网格 $\boldsymbol{DT}(\boldsymbol{P})$ 中的一个三角形,垂直扫描线 l 上的点 q 位于 $\boldsymbol{VR}_o(v_jv_{j+1})$ $(1\leqslant j<m)$ 中。如果点 q 在 $\Delta v_iv_jv_{j+1}$ 的外接圆内,那么为了得到 $\boldsymbol{DT}(\boldsymbol{P}\cup q)$,就需要将 v_jv_{j+1} 用 v_iq 替代,用来构造两个新的三角形,才能满足 Delaunay 原则。qv_j 与 qv_{j+1} 是 Delaunay 三角网格 $\boldsymbol{DT}(\boldsymbol{P}\cup q)$ 中的两条边,因此在 $\boldsymbol{VD}(\boldsymbol{P}\cup q)$ 中,点 q 与点 v_j,点 q 与点 v_{j+1} 之间分别存在 Voronoi 边。否则,如果点 q 在 $\Delta v_iv_jv_{j+1}$ 的外接圆外部,令点 r 是 $\Delta v_iv_jv_{j+1}$ 的外接圆的圆心,则点 r 必位于线段 v_jv_{j+1} 的垂直平分线上。如果沿 v_jv_{j+1} 的垂直平分线拉动点 r 朝凸包外的方向运动,

则以点 r 为圆心、过点 v_j 和点 v_{j+1} 两顶点的圆逐渐变大,且保持空圆特性(即该圆内部没有其他种子点),直到遇到点 q(如图 6(a)所示)。这时,由于 qv_j 和 qv_{j+1} 必是 Delaunay 三角网格 $\boldsymbol{DT}(\boldsymbol{P}\cup q)$ 的两条边,因而在 $\boldsymbol{P}\cup q$ 的 VD 中,点 q 和点 v_j,点 q 和点 v_{j+1} 之间分别存在 Voronoi 边。证毕。

在图 6 中,多边形链 $p_1p_2p_3p_4$ 是关于直线 l 的一条右凸链,点 q 是 l 上一点。图 6(b)中的点 q 在 $\boldsymbol{VR}_o(p_3)$ 中,也必在 $\boldsymbol{VR}_p(p_3)$ 中。p_3q 是 $\boldsymbol{DT}(\boldsymbol{P}\cup q)$ 中的一条 Delaunay 边,即在 $\boldsymbol{VD}(\boldsymbol{P}\cup q)$ 中,p_3,q 之间存在一条 Voronoi 边。图 6(a)中的点 q 在 $\boldsymbol{VR}_o(p_2p_3)$ 中。这时,p_2q 和 p_3q 都是 $\boldsymbol{DT}(\boldsymbol{P}\cup q)$ 中的 Delaunay 边,即在 $\boldsymbol{VD}(\boldsymbol{P}\cup q)$ 中,点 p_2 和点 q,点 p_3 和点 q 之间都存在 Voronoi 边。

因此,对于新扫描的一个点 q,可以先基于右凸链在 $O(\log n)$ 时间内定位点 q 所在的 VR,并找到一条属于 $\boldsymbol{VD}(\boldsymbol{P}\cup q)$ 中的、与点 q 相关的 Voronoi 边;然后执行第 2 节所介绍的增量方法的第 2 步计算出点 q 的 VR,即可得到 $\boldsymbol{VD}(\boldsymbol{P}\cup q)$。

2.3 右凸链的更新

设当前插入点 q 落入 $\boldsymbol{VR}_p(p_i)$ 中,并已经计算出 $\boldsymbol{VD}(\boldsymbol{P}\cup q)$,则根据前面的分析,$qp_i$ 为 $\boldsymbol{DT}(\boldsymbol{P}\cup q)$ 中的一条边,即点 q,p_i 之间存在 Voronoi 边。根据第 2 节所介绍的增量方法的第 2 步计算出点 q 的 VR 区域,即可以肯定 $\boldsymbol{VR}_p(q)$ 是开放的,且有两条开放的 Voronoi 边。设这两条开放的 Voronoi 边分别是点 q 和点 p_j,点 q 和点 p_k 之间的 Voronoi 边。容易证明,点 q 与点集 P 的凸包的切线必相切于点 p_j 和点 p_k(如图 7 中的点 p_2 和点 p_4)。因此,算法在计算 $\boldsymbol{VD}(\boldsymbol{P}\cup q)$ 时就可计算出新的右凸链。

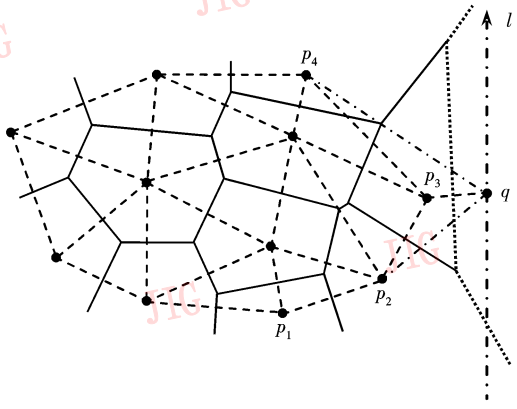


图 7 右凸链原为 $p_1p_2p_3p_4$,更新后为 p_2qp_4

Fig. 7 Renew RCHC. The old is $p_1p_2p_3p_4$, the new is p_2qp_4

3 算法描述

本文算法首先对给定的种子点按 x 方向由小到大排序,得到点 $p_1,p_2,\cdots,p_n$;然后取点 p_1,p_2,p_3,构造它们的 VD,并计算初始的右凸链 RCHC;接着依次插入后面的每一个种子点 p_k,并基于 RCHC 计算包含点 p_k 的 VR,设点 p_k 落在 $\boldsymbol{VR}_p(p_i)$ 中,则 p_kp_i 为 $\boldsymbol{DT}(\boldsymbol{P}\cup p_k)$ 的一条边;最后计算 $\boldsymbol{VD}(\boldsymbol{P}\cup p_k)$,并更新 RCHC。当所有种子点都处理完后,即得到整个输入种子点集的 VD。

下面给出整个算法的形式化描述。

算法 1 Create$\boldsymbol{VD}(\boldsymbol{P})$——改进的 VD 算法

输入:含有 $n\geqslant 3$ 个的平面离散点的集合 P。

输出:$\boldsymbol{VD}(\boldsymbol{P})$。

1) 如果 $n<3$,则返回。

2) 对给定的种子点按 x 方向由小到大排序,如果方向 x 相同,则按 y 方向排序,得到点 p_1,$p_2,\cdots,p_n$。

3) 构造点 p_1,p_2,p_3 的 VD。

4) 计算初始的右凸链 RCHC。

//RCHC 有可能是其中的 3 个点或 2 个点。

5) For($k=4;k\leqslant n;k++$){

6) 基于 RCHC 用二分法查找包含点 p_k 的 VR。

7) 如果点 p_k 在 $\boldsymbol{VR}_o(v_i)$ 中,则计算点 p_k 与点 v_i 之间的 $\boldsymbol{VD}(\boldsymbol{P}\cup p_k)$ 的 Voronoi 边。其中,点 v_i 为 RCHC 的一个顶点。

8) 如果点 p_k 在 $\boldsymbol{VR}_o(v_iv_{i+1})$ 中,计算点 p_k 与点 v_i,点 p_k 与点 v_{i+1} 之间的 $\boldsymbol{VD}(\boldsymbol{P}\cup p_k)$ 的 Voronoi 边。

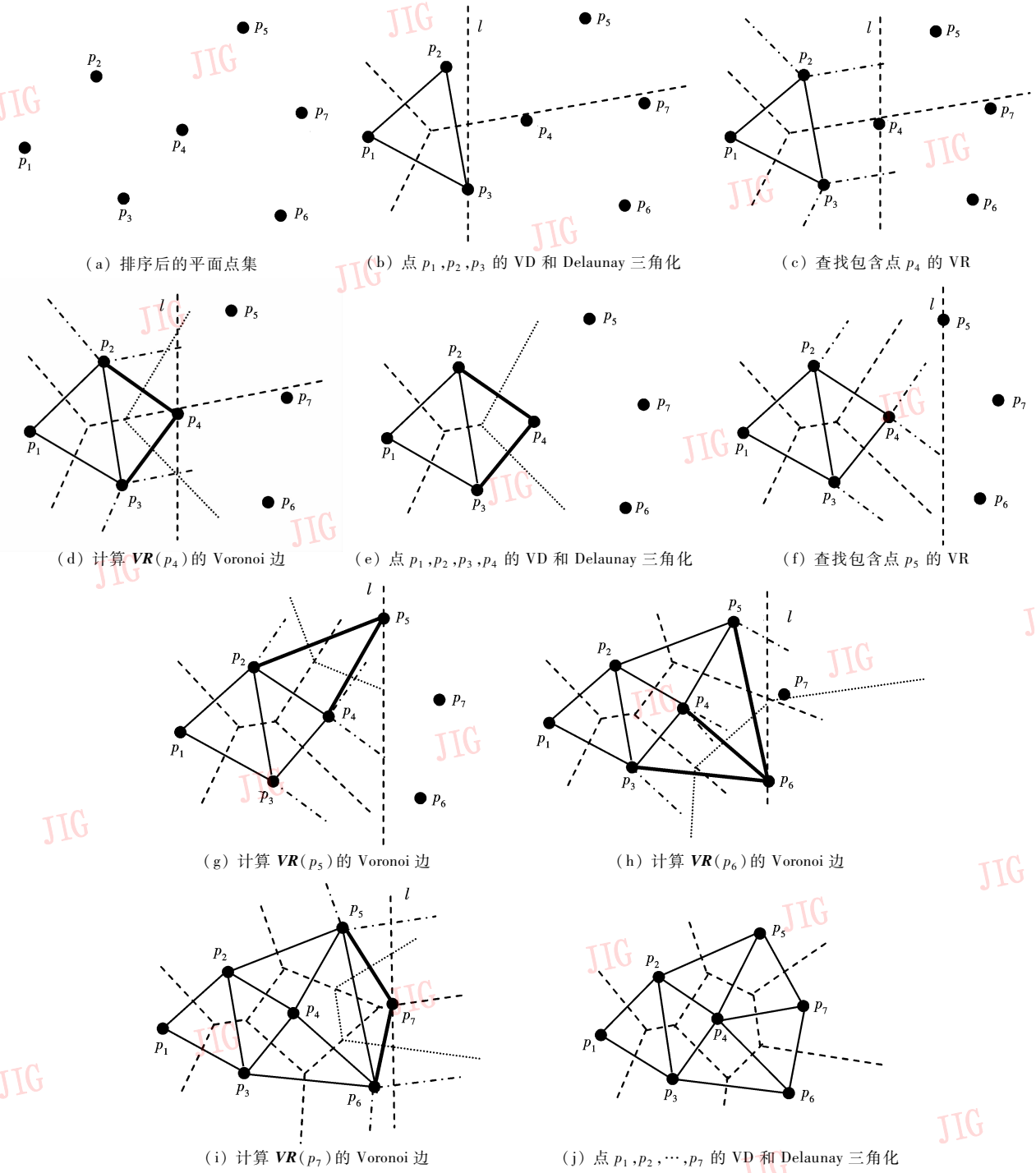
9) 根据上面的 Voronoi 边,采用第 2 节所介绍的增量法的第 2 步中的方法来计算 $\boldsymbol{VR}_p(p_k)$,并计算得到 $\boldsymbol{VD}(\boldsymbol{P}\cup p_k)$。

10) 计算新的 RCHC。

11) }

12) 返回。

图 8 给出了一个计算平面点集的 VD 的例子。排序后的序列为:$p_1,p_2,\cdots,p_7$(如图 8(a)所示)。图 8(b)给出了点 p_1,p_2,p_3 的 VD 和 Delaunay 三角网格,其 RCHC 为 p_3p_2。图 8(c)—图 8(e)给出了计算点 p_1,p_2,p_3,p_4 的 VD 的过程,以及最后得到的 RCHC $p_3p_4p_2$。图 8(f)和图 8(g)给出了计算点 p_1,p_2,p_3,p_4,p_5 的 VD 的过程,以及最后得到的 RCHC $p_3p_4p_5$。图 8(h)给出了计算点 p_1,p_2,p_3,p_4,p_5,p_6 的



(a) 排序后的平面点集　(b) 点 p_1,p_2,p_3 的 VD 和 Delaunay 三角化　(c) 查找包含点 p_4 的 VR

(d) 计算 $\boldsymbol{VR}(p_4)$ 的 Voronoi 边　(e) 点 p_1,p_2,p_3,p_4 的 VD 和 Delaunay 三角化　(f) 查找包含点 p_5 的 VR

(g) 计算 $\boldsymbol{VR}(p_5)$ 的 Voronoi 边　(h) 计算 $\boldsymbol{VR}(p_6)$ 的 Voronoi 边

(i) 计算 $\boldsymbol{VR}(p_7)$ 的 Voronoi 边　(j) 点 $p_1,p_2,\cdots,p_7$ 的 VD 和 Delaunay 三角化

图 8　本文算法过程

Fig. 8　A snapshot of our algorithm

VD 的过程,以及最后得到的 RCHC p_6p_5。图 8(i)和图 8(j)给出了计算得到的点 $p_1,p_2,\cdots,p_7$ 的 VD。

4　结　论

本文算法所用的数据结构,一是采用双向链表来保存生成的 VD,一是采用平衡二叉树来保存右凸链。由于 VD 的大小与所输入的种子点的个数为线性关系,所以这两个数据结构占用的空间都是 $O(n)$,其中 n 是所输入的种子点的数目,即空间复杂度为 $O(n)$。

本文算法是对增量法的改进,主要是改进了点

定位部分。在前面也做了分析,由于基于右凸链可在 $O(\log n)$时间内定位新插入的点,所以总的定位查询时间为 $O(n\log n)$。由于右凸链的修改是和计算 VD 一块进行的,因此整个算法的时间复杂度与扫描线算法一样,时间复杂度也为 $O(n\log n)$,但本文算法更简洁,且便于理解和编程实现。

参考文献(References)

[1] Goodman Jacob E, O'Rourke Joseph. Handbook of Discrete and Computational Geometry(2nd edition)[M]. Boca Raton, FL, USA: Chapman & Hall/CRC, 2004.

[2] Shamos M I, Hoey D. Closest-point problems[C]//Proceedings 16th Annual Symposium on Foundations of Computer Science, New York, USA: lEEE Computer Society, 1975: 151-162.

[3] Franz Aurenhammer. Voronoi diagrams—A survey of a fundamental geometric data structure[J]. ACM Computing Surveys, 1991, 23(3): 345-405.

[4] De Berg M, Kreveld M, Van Overrmars M, et al. Computational Geometry: Algorithms and Applications(2nd Edition)[M]. New York, USA: Springer-Verlag Inc, 2000.

[5] Fortune S. A sweepline algorithm for Voronoi diagrams[J]. Algorithmic, 1987, 2(1-4): 153-174.

[6] Guibas L J, Stolfi J S. Ruler, compass, and computer: The design and analysis of geometric algorithms[C]//Earnshaw R A (ed): Proceedings Theoretical Foundations of Computer Graphics and CAD, Berlin, German; New York, USA: Springer-Veralag, 1988: 111-165.

[7] Fang Tsungpao, Piegl L A. Delaunay triangulation using a uniform grid[J]. IEEE Computer Graphics and Applications, 1993, 13(3): 36-47.

[8] Yang Chenglei, Wang Jiaye, Meng Xiangxu. Upper bounds of the numbers of vertices and edges in outer Voronoi diagram of polygon[J]. Journal of Computer Aided Design and Computer Graphics, 2005, 17(4): 689-693.[杨承磊, 汪嘉业, 孟祥旭. 多边形外部 Voronoi 图顶点和边数的上界[J]. 计算机辅助设计与图形学学报, 2005, 17(4): 689-693].