多边形中的点可见性快速算法

赵海森[1,2)]，杨承磊[1,2)*]，吕　琳[1,2)]，王筱婷[1,2)]，杨义军[1,2)]，孟祥旭[1,2)]
[1)](山东大学计算机科学与技术学院　济南　250101)
[2)](山东省软件工程重点实验室　济南　250101)
(chl_yang@sdu.edu.cn)

摘　要：针对点的可见性计算这一计算几何中的基础问题，提出一种支持任意查询点的可见多边形快速计算的基于多边形 Voronoi 图的点可见性算法. 以与 Voronoi 骨架路径对应的 Voronoi 通道概念，以及相应的局部最短路径概念为基础，按照深度优先策略对 Voronoi 图进行遍历，在计算 Voronoi 骨架路径的同时计算局部最短路径，并基于局部最短路径计算所遍历的多边形边的可见部分. 该算法可以处理“带洞”多边形，而且只对多边形进行局部访问；对于“带洞”多边形，由于该算法的数据结构比较简单、剖分空间合理且易于实现，因此仅需 $O(n)$ 空间和 $O(n\lg n)$ 预处理时间. 最后给出了在三维室内虚拟场景设计与漫游系统中的应用实例，结果表明文中算法是实际可行，且运行时间与点的可见多边形的边数和多边形的边数均呈线性关系.

关键词：“带洞”多边形；可见多边形；Voronoi 图；最短路径
中图法分类号：TP391

An Algorithm for Visibility Computation of Points in Polygon

Zhao Haisen[1,2)], Yang Chenglei[1,2)*], Lu Lin[1,2)], Wang Xiaoting[1,2)], Yang Yijun[1,2)], and Meng Xiangxu[1,2)]
[1)](*School of Computer Science and Technology, Shandong University, Ji'nan* 250101)
[2)](*Shandong Provincial Key Laboratory of Software Engineering, Ji'nan* 250101)

Abstract: Visibility computation is a fundamental problem in computational geometry. This paper presents a Voronoi diagram-based algorithm for computing the visibility polygon of any point in the polygon. First, the definitions of Voronoi corridors and local shortest paths in Voronoi corridors are proposed. Then the depth-first search for traversing the Voronoi diagram is performed to compute Voronoi skeleton paths and the local shortest paths from a point to the vertices of the polygon. Afterwards, the visibility portions of the polygon edges are computed. Compared with the popular visibility algorithms based on triangulation, the proposed algorithm can deal with the polygon with holes and partially visiting the polygon only. Compared with algorithms based on visual cell subdivision or visible graphs, the proposed algorithm has a simpler data structure, more reasonable space subdivision, easier implementation, and only takes $O(n)$ space and $O(n\lg n)$ preprocessing time. Examples show that the algorithm is of practical effectiveness, and the running time is approximately linear to the number of the edges of the visibility polygon of a point and the number of the edges of the polygon.

Key words: polygon with holes; visibility polygon; Voronoi diagram; shortest path

---

收稿日期：2012-06-03；修回日期：2012-07-19. **基金项目**：国家自然科学基金(61272243,61202146,61202147)；山东省优秀中青年科学家基金(BS2012DX014)；山东省自然科学基金(ZR2012FQ026)；山东大学自主创新基金(2010HW010). **赵海森**(1988—)，男，硕士研究生，CCF 会员，主要研究方向为计算几何等；**杨承磊**(1972—)，男，博士，教授，CCF 会员，论文通讯作者，主要研究方向为计算几何等；**吕　琳**(1981—)，女，博士，副教授，CCF 会员，主要研究方向为计算机图形学等；**王筱婷**(1976—)，女，硕士，工程师，CCF 会员，主要研究方向为计算机图形学；**杨义军**(1979—)，男，博士，副教授，CCF 会员，主要研究方向为 CAD；**孟祥旭**(1962—)，男，博士，教授，博士生导师，CCF 会员，主要研究方向为人机交互等.

可见性计算是计算几何领域重要的研究内容之一,广泛应用于计算机图形学、计算机游戏、地理信息系统和机器人等领域.其中,点的可见性计算是最基础的问题,其典型应用包括艺术画廊中的警卫问题、机械模型浇注、无线传感网络中的覆盖问题等[1-3];在计算机图形学、计算机游戏和虚拟现实等领域,其作为实时绘制中的一种关键技术,根据视点的位置计算可见区域,并将可见区域中的对象加入图形绘制流水线,提高绘制效率.目前,已有一些关于虚拟场景可见性计算的文献[4-7].

本文工作背景之一是面向三维虚拟室内场景的交互设计与漫游等的具体应用,其中包括数据组织、协同设计、可见性计算、路径规划、碰撞检测以及点定位等多个环节.通常,在三维场景中,为了有效组织数据并提高计算/查询效率,区域划分是广泛采用的方法,包括常用于加速可见性计算.目前已有很多自动区域划分算法[8-11],如基于 BSP 树等;然而,它们对于任意场景空间的自动区域划分依然比较困难,通常都需要较长的预处理时间和较多的存储空间,并且划分的结果也不尽如人意.Voronoi 图作为一种有效的区域划分方法,能够保持所划分区域的拓扑关系,并具有最近性和邻接性等性质,可作为空间划分结构,在三维虚拟室内场景的设计与漫游的各个环节中进行应用,因此近年得到了人们的关注[12].

本文主要解决的问题是:给定一个有 $h\geqslant 0$ 个"洞"的多边形 P(可看作是室内场景的二维投影)及其 Voronoi 图,如何快速计算任意点 $q\in P$ 的可见多边形.为此,本文基于 Voronoi 图的几何特性,提出一种基于多边形 Voronoi 图的快速计算点的可见性算法.

1 相关工作

对于点可见多边形计算问题,计算几何领域有很多相关的工作[13-15].

对于简单多边形,文献[16]提出了一种基于三角化的点的可见多边形算法.它首先对简单多边形进行三角化,并基于三角化的结果构造查询点的最短路径树(该点到多边形所有顶点的最短路径的集合);然后基于该最短路径树计算多边形的每条边相对于查询点的可见部分,最后得到点的可见多边形,算法时间复杂度为 $O(n)$.该算法由于易于理解与实现,已成为计算简单多边形内点的可见性的常用算法,而且最短路径树也成为计算可见性的常用数据结构.文献[17-18]分别给出了时间复杂度为 $O(|V(q)|+\lg n)$ 的简单多边形内点的可见性计算方法($|V(q)|$ 为点 q 的可见多边形 $V(q)$ 的边的数目),其在预处理阶段需要将多边形区域划分成一个个可见区域,同一区域内的所有点的可见多边形的结构相同.基于该划分构造持久化的数据结构,以支持点的可见多边形的快速查询;该方法需要 $O(n^3)$ 的空间和 $O(n^3\lg n)$ 预处理时间.文献[19]将空间和预处理时间分别降低为 $O(n^2)$ 和 $O(n^2\lg n)$,但其算法复杂度为 $O(|V(q)|+\lg^2 n)$.近年来,又出现了一些基于可见区域分割或可见图计算"带洞"多边形中点的可见多边形的算法[14-15,20-22].这些算法的数据结构一般比较复杂且构造方法较困难,消耗较多的预处理时间和空间;其中,文献[15]给出的是计算移动点的可见多边形算法.表 1 所示为这些算法的时间和空间复杂度.表 1 中"S"表示简单多边形,"M"

表 1 已有文献算法的时间和空间复杂度

文献	多边形	预处理		查询 $V(q)$ 的时间复杂度
		时间复杂度	空间复杂度	
[16]	S	$O(n)$	$O(n)$	$O(n)$
[17-18]	S	$O(n^3\lg n)$	$O(n^3)$	$O(\|V(q)\|+\lg n)$
[19]	S	$O(n^2\lg n)$	$O(n^2)$	$O(\|V(q)\|+\lg^2 n)$
[14]	M	$O(n^4\lg n)$	$O(n^4)$	$O(\|V(q)\|+\lg n)$
[20]	M	$O(n^3\lg n)$	$O(n^3)$	$O(\|V(q)\|+(1+\min(h,\|V(q)\|))\lg n)$
[21]	M	$O(n^2\lg n)$	$O(n^2)$	$O(\|V(q)\|+h+(1+\min(h,\|V(q)\|))\lg^2 n)$
[15]	M	$O(n^2+\lg n)$	$O(n^2)$	移动点的初始位置:$O(\|V(q)\|*\lg n)$ 移动点的其他位置:$O(\|V(q)\|)$
[22]	M	$O(n^2\lg n)$	$O(n^2)$	$O\left(\|V(q)\|+h+h\lg\left(\frac{n}{h}\right)+\lg^2 n\right)$

表示“带洞”多边形. 另外，文献[23]提出了一种基于 Voronoi 图计算潜在可见集的算法，其基于该数据结构可以查询一般点、移动点、线段或区域的可见多边形；但需要用计算点的可见多边形的算法[24]来构造潜在可见集，需要 $O(n^3)$ 预处理时间和 $O(n^2)$ 空间. 文献[15]也用文献[24]算法计算移动点的初始位置处的可见多边形；文献[24]算法的优点是不需要预处理，但其时间复杂度为 $O(n\lg n)$.

对于简单多边形中最短路径的计算，基于 Voronoi 图的计算任意两点之间的最短路径的算法[25]的基本思想为：由于在简单多边形中任意两点之间只存在一条 Voronoi 骨架路径，最短路径上的点能够通过顺序遍历其 Voronoi 骨架路径关联的多边形顶点求得. 确定最短路径的算法有 3 个步骤：首先确定两点所在的 Voronoi 区域；然后确定 Voronoi 骨架路径；最后顺序遍历 Voronoi 骨架路确定最短路径. 本文将用到该算法.

2　相关概念

本文规定，简单多边形是指只有一条边界的多边形；“带洞”多边形含多条边界，其中有一条外边界；外边界是顺时针的，内边界是逆时针的. 多边形内的每一个凹顶点或每一条边为一个站点(site)，用 o 表示；它们在多边形内部都有其 Voronoi 区域(Voronoi region, VR)，站点 o 的 VR 用 VR(o)表示. 对于任意站点 o，其 VR 是指 P 中到 o 的距离比到 P 中任何其他站点距离更近的所有点的集合. 多边形 P 的 Voronoi 图(Voronoi diagram, VD)则是 P 中所有站点的 VR 的并集，用 VD(P)表示. 本文将多边形中的凹顶点看作特殊的边，即它的 2 个端点是重合的. 这样，简单多边形的 VD 是一棵树，其叶节点是多边形的顶点；“带洞”多边形的 VD 是一个图，其中度数为 1 的节点为多边形的顶点.

多边形 P 中，两点 p 和 q 是可见的当且仅当线段 pq 和 P 的交仍为 pq. P 中一点 q 的可见多边形 $V(q)$是 P 中 q 的所有可见点的集合. $V(q)$有 2 种类型的边：一类是原始边，是 P 的边或边的一部分，称其为 P 的边相对于 q 的可见部分；另一类为构造边，其端点在 P 的边界上，除端点外的部分则在 P 的内部. 对于简单多边形，每条边相对于 q 最多只有一个可见部分；对于“带洞”多边形，每条边相对于 q 可能有多个可见部分.

由于 $V(q)$是一个星形多边形，因此如果逆时针沿着 $V(q)$的边界依次遍历，q 和 $V(q)$的各顶点的连线与水平轴之间的有向角是从小到大依次排列的.

本文算法将沿着多边形的 VD 进行深度优先搜索，访问多边形中的边，并计算被访问边相对于 q 的可见部分；其中会用到 q 到其 2 个顶点的 Voronoi 骨架路径和局部最短路径的概念.

对于多边形 P 中一点 q，假设其位于边 e 的 Voronoi 区域 VR(e)中，过 q 作 e 的垂线，可以证明该垂线只与 VR(e)边界上的一条 Voronoi 边相交，不妨设交点为 a. 将 a 看作一个节点，qa 看作一条边加入 VD(P)，如图 1 所示.

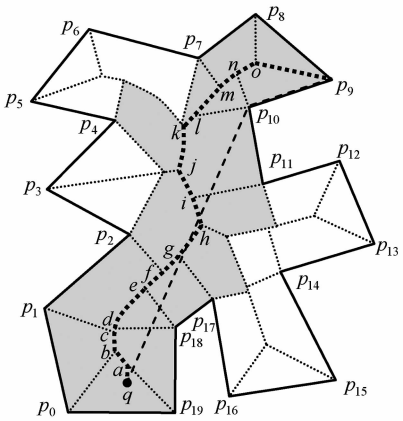


图 1　简单多边形

定义 1. 多边形 P 中的一个点 q 到 P 的一个顶点 p_i 的 Voronoi 骨架路径(Voronoi skeleton path, VSP)是一条由系列 Voronoi 边首尾依次连接而成的路径，用 VSP(q, p_i)表示.

图 1 中粗虚线所示 VSP(q, p_9) = {q, a, …, o, p_9}.

定义 2. 若 Voronoi 边 e 为 2 个 Voronoi 区域 VR(o_1)和 VR(o_2)的边界所共有，则称 VR(o_1)和 VR(o_2)为 e 的关联 VR，o_1 和 o_2 为 e 的关联边. 假设 a 和 b 分别为 e 的起点和终点，若 o_1 在有向边 ab 的左侧(右侧)，则称 o_1 为 ab 的左侧(右侧)关联边.

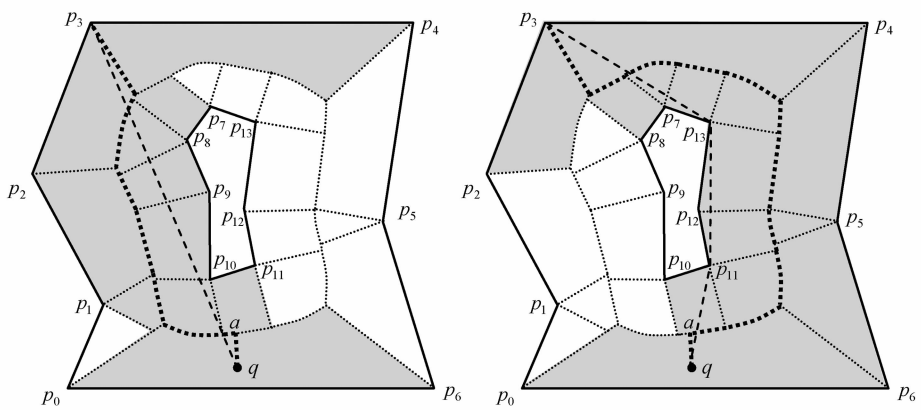
如图 1 所示，边 p_1p_2 为有向边 de 的左侧关联边，凹顶点(特殊边)p_{18}为 de 的右侧关联边. 需注意的是 p_1p_2 为 ed 的右侧关联边，p_{18}为 ed 的左侧关联边.

定义 3. 多边形 P 中的一个点 q 到 P 的一个顶点 p_i 的 Voronoi 骨架路径 VSP(q, p_i)的每条 Voronoi 边所关联的 Voronoi 区域形成 q 到 p_i 的一个通道，称之为 VSP(q, p_i)对应的 Voronoi 通道 $C(q, p_i)$. q 到 2 个顶点 p_i 和 p_{i+1}的 Voronoi 通道

$C(q,p_i)$ 和 $C(q,p_{i+1})$ 组成了 q 到边 p_ip_{i+1} 的 Voronoi 通道 $C(q,p_ip_{i+1})$. 这里，$VSP(q,p_i)$ 和 $VSP(q,p_{i+1})$ 是在沿 $VD(P)$ 深度优先搜索时依次遍历得到的，它们除了所含有的 $VR(p_ip_{i+1})$ 的 Voronoi 边不同，其他边都是相同的.

定义 4. 在一个 $C(q,p_i)$ 中，将 $VSP(q,p_i)$ 看作一条绳子，如果用力将其拉紧，则可得到在该 $C(q,p_i)$ 中 q 到 p_i 的一条最短路径，本文中将其称之为局部最短路径(local shortest past, SP)，用 $SP_c(q,p_i)$ 表示.

如图 1 中灰色区域为 $VSP(q,p_9)$ 所对应的 Voronoi 通道 $C(q,p_9)$，图 2 中灰色区域为 q 到 p_3 的 2 条 Voronoi 骨架路径分别对应的 Voronoi 通道. 如图 1 中，$VSP(q,p_8)=\{q,a,\cdots,o,p_8\}$，$VSP(q,p_9)=\{q,a,\cdots,o,p_9\}$，它们不同的部分是 op_8 和 op_9，都是 $VR(p_8p_9)$ 的 Voronoi 边. 在 $C(q,p_8p_9)$ 中，q 到 p_8 的局部最短路径为 $SP_c(q,p_8)=\{q,p_8\}$，q 到 p_9 的局部最短路径为 $SP_c(q,p_9)=\{q,p_{10},p_9\}$. 需要注意的是：在“带洞”多边形中，$q$ 到 p_i 存在多个 Voronoi 通道，因此存在多个局部最短路径. 如图 2 a 所示灰色区域的 Voronoi 通道 $C(q,p_3)$ 中，$SP_c(q,p_3)=\{q,p_3\}$；在图 2 b 所示灰色区域还存在一个 Voronoi 通道，其中 $SP_c(q,p_3)=\{q,p_{11},p_{13},p_3\}$.



a 左侧通道　　b 右侧通道

图 2 “带洞”多边形

3 算法描述

本节首先讨论多边形内计算一条边相对于给定查询点的可见部分的计算算法.

引理 1. 对于一个多边形 P 及其内部一点 q，在 q 到 p_i 的一个 Voronoi 通道中，q 到 p_i 的局部最短路径 $SP_c(q,p_i)$ 能够通过顺序遍历 $VSP(q,p_i)$ 得到，且 $SP_c(q,p_i)$ 上的顶点必为 $VSP(q,p_i)$ 所关联的凹顶点.

由于对于一个多边形 P 及其内部一点 q，q 到 P 的边 p_ip_{i+1} 的一个 Voronoi 通道可以看作一个简单多边形，因此引理 1 的证明可参考[25]，此处略之.

引理 2. 对于一个多边形 P 及其内部一点 q，在 q 到 P 的边 p_ip_{i+1} 的一个 Voronoi 通道中，p_ip_{i+1} 相对于 q 存在可见部分，当且仅当 q 到 p_i 的局部最短路径 $SP_c(q,p_i)$ 是一条在 P 内的凸多边形链且在 qp_i 的右侧，且 q 到 p_{i+1} 的局部最短路径 $SP_c(q,p_{i+1})$ 也是一条在 P 内的凸多边形链且在 qp_{i+1} 的左侧.

证明.

充分性. 由引理 1 知，$SP_c(q,p_i)$ 上的点为 $VSP(q,p_i)$ 所关联的凹顶点. 如果只用 $VSP(q,p_i)$ 的左侧关联凹顶点计算 $SP_c(q,p_i)$，则 $SP_c(q,p_i)$ 为 $VSP(q,p_i)$ 的左侧关联凹顶点的凸多边形链 L_CP. 在 p_ip_{i+1} 相对于 q 存在可见部分的情况下，若 $SP_c(q,p_i)\neq L\_CP$，即 $SP_c(q,p_i)$ 非凸，则存在 $SP_c(q,p_i)=\{q,l_1,\cdots,l_l,p_i\}$ 中一个顶点 $l_j(1\leqslant j\leqslant l)$ 为 $VSP(q,p_i)$ 的右侧关联凹顶点，则 $SP_c(q,p_i)$ 和 $SP_c(q,p_{i+1})$ 有交错部分，p_ip_{i+1} 相对于 q 不可见，存在矛盾. 因此，$SP_c(q,p_i)=L\_CP$，即 $SP_c(q,p_i)$ 是一条在 P 内的凸多边形链. 显然，该链在 qp_i 的右侧；同理可证明 $SP_c(q,p_{i+1})$ 也是一条在 P 内的凸多边形链且在 qp_{i+1} 的左侧.

必要性. 设 $SP_c(q,p_i)=\{q,l_1,l_2,\cdots,l_l,p_i\}$，并且 $SP_c(q,p_{i+1})=\{q,r_1,r_2,\cdots,r_r,p_{i+1}\}$，显然，$SP_c(q,p_i)$ 在 $VSP(q,p_i)$ 的左侧，$SP_c(q,p_{i+1})$ 在 $VSP(q,p_{i+1})$ 的右侧. 由于 $VSP(q,p_i)$ 在 $VSP(q,p_{i+1})$ 的左侧，因此 $SP_c(q,p_i)$ 在 $SP_c(q,p_{i+1})$ 的左

侧. 由于 $SP_c(q,p_i)$是一条凸多边形链且在 qp_i 的右侧，且 $SP_c(q,p_{i+1})$也是一条凸多边形链且在 qp_{i+1} 的左侧，它们形成一个漏斗形状，因此 p_ip_{i+1}相对于点 q 有可见部分. 证毕.

如图 3 所示，对于边 p_4p_5，有局部最短路径 $SP_c(q,p_4)=\{qp_3p_4\}$，$SP_c(q,p_5)=\{qp_6p_5\}$，符合引理 2 的条件，可以断定 p_4p_5 相对 q 有可见部分. 而对于边 p_3p_4，有局部最短路径 $SP_c(q,p_3)=\{qp_3\}$，$SP_c(q,p_4)=\{qp_3p_4\}$，不符合引理 2 的条件，所以可以断定 p_3p_4 相对于 q 不可见(严格地说，点 p_3 相对于 q 可见，在本文中定义类似 p_3p_4 这样的边不可见，不会影响算法输出结果).

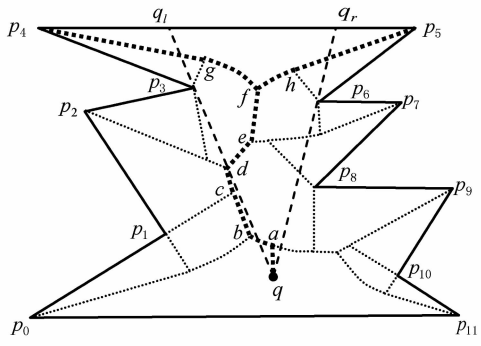


图 3　计算边的可见部分

由引理 1 和引理 2 容易得到定理 1.

定理 1. 对于一个多边形 P 及其内部一点 q，在 q 到 P 的边 p_ip_{i+1}的一个 Voronoi 通道中，有 $SP_c(q,p_i)=\{q,l_1,\cdots,l_l,p_i\}$，$SP_c(q,p_{i+1})=\{q,r_1,\cdots,r_r,p_{i+1}\}$. p_ip_{i+1}相对于 q 存在可见部分，当且仅当 ql_1 在 qr_1 的左侧；且当前 Voronoi 通道中 p_ip_{i+1}相对于 q 的可见部分可通过计算射线 ql_1 和 qr_1 与 p_ip_{i+1}的交点得到.

如图 3 所示，qp_3 在 qp_6 的左侧，根据定理 1 知，p_4p_5 相对于 q 存在可见部分. 通过计算射线 qp_3，qp_6 与 p_4p_5 的交点即可计算出 p_4p_5 相对于 q 的可见部分为 q_lq_r. 这里，射线 ql_1 和 qr_1 之间的区域成椎体状，称为可见椎体(visibility centrum, VC)，用 $VC(ql_1,qr_1)$表示.

为了计算整个 $VP(q)$，将沿着 $VD(P)$进行深度优先搜索，访问每个叶节点，即访问 P 中的每个顶点. 由于访问中得到的相邻 2 个顶点属于同一条边，所以可以在前一顶点的 VSP 基础上增加或减少新访问的 Voronoi 边，即可得到下一顶点的 VSP；并在计算新的 VSP 的同时，在前一顶点的 SP_c 的基础上计算下一顶点的 SP_c；这样可减少计算量.

在沿着 $VD(P)$进行深度优先搜索时，需要考虑每次遍历的终止条件. 原因在于：

1) 如图 4 a 所示灰色区域 $V(q)$只是多边形的一部分，在遍历到 Voronoi 顶点 q_2 和 q_3 时，都没有必要再深度搜索下去，因为它们后面的 Voronoi 边关联的多边形的边肯定相对于 q 不可见；

2) 如图 4 b 所示“带洞”多边形，如果不设终止条件，会陷入死循环. 其实，在沿每个“洞”的一侧遍历时，当遍历到某个 Voronoi 顶点(如 q_1，q_2 和 q_3)后，后面的 Voronoi 边对应的边都相对于 q 不可见，所以可以停止这个方向的遍历.

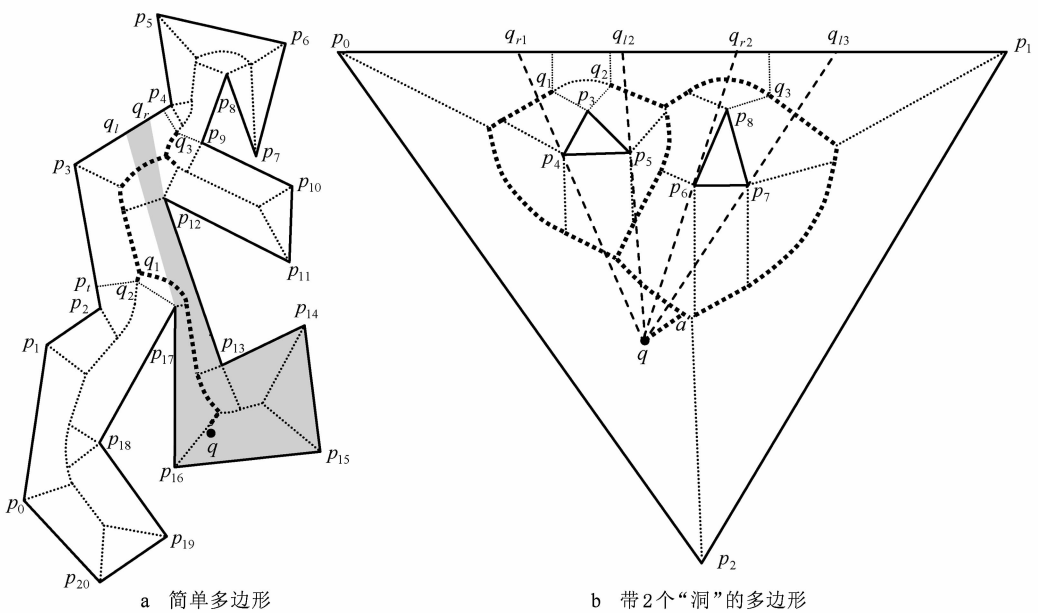


图 4　终止条件图例

下面讨论如何设定终止条件.

容易错误的认为:根据定理 1,如果 ql_1 不在 qr_1 的左侧,则可以停止继续往下遍历并开始回溯. 但如图 5 所示,在沿"洞"H_1 左侧遍历到顶点 p_3 时,$ql_1=qp_1$ 不在 $qr_1=qp_1$ 的左侧,但仍需继续遍历,并经过点 q_3,q_2,q_4,直至到达 q_6 才能停止;否则,多边形的某些边(如 p_4p_5,p_5p_6)相对于 q 的可见部分就会漏掉.

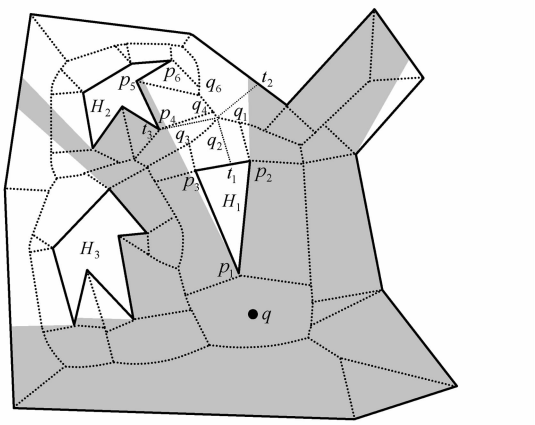


图 5 遍历方向不同,q_2 处的终止条件计算方法不同

不妨设当前遍历的骨架路径为 $q,a,a_1,a_2,\cdots,a_j$,当前的可见椎体为 $VC(ql_1,qr_1)$,进行深度搜索访问到的下一个 Voronoi 顶点为 a_{j+1}. 过 a_{j+1} 分别作 a_ja_{j+1} 的两侧关联边的垂足 t_1 和 t_2(需要注意的是:若另一个遍历是从 a_{j+1} 遍历到 a_j,应当过点 a_j 作 Voronoi 边 $a_{j+1}a_j$ 左右两侧关联的边的垂足. 如图 5 所示,当从点 q_3 遍历到 q_2,取垂足为 t_1 和 t_3;若从 q_1 遍历到 q_2,垂足为 t_1 和 t_2;若从 q_4 遍历到 q_2,垂足为 t_3 和 t_2).

定理 2. 在当前访问 Voronoi 通道中,若点 t_1 和 t_2 同时在 ql_1 的左侧或在 ql_1 上,或者同时在 qr_1 的右侧或在 qr_1 上,则沿着 VD(P)进行的当前遍历中,a_{j+1} 后面的 Voronoi 边 e 所关联的边都不在 $VC(ql_1,qr_1)$中.

证明. 假设命题不成立,则在当前访问的 Voronoi 通道中,假设 a_{j+1} 后面的 Voronoi 边 e 所关联的边上有一点 p 在 $VC(ql_1,qr_1)$中,即 p 相对于 q 可见,则有 $SP_c(q,p)=\{q,p\}$. 然而,由于 t_1 和 t_2 同时在 ql_1 左侧或在 ql_1 上,或者同时在 qr_1 右侧或在 qr_1 上,e 为沿着 VD(P)在 a_{j+1} 后遍历到的 Voronoi 边所关联的边,所以 p 必在远离 $VC(ql_1,qr_1)$的一侧. 因而可知 l_1 或 r_1 必为 q 到 p 的局部最短路径上一点,这与 $SP_c(q,p)=\{q,p\}$矛盾,定理 2 得证.

证毕.

定理 2 给出了沿 VD(P)进行深度遍历的终止条件. 根据定理 2(termination condition, TC)设计函数 TC(v),用于表示在当前访问的 Voronoi 通道中 Voronoi 顶点 v 是否满足终止条件. 如果 VC(v)=TRUE,则满足终止条件. 如图 4 a 所示,在当前访问的 Voronoi 通道中,p_{17} 和 p_t 为点 q_2 在边 q_1q_2 的两侧关联边的垂足;p_{17} 在 vp_{17} 上,p_i 在 vp_{17} 的左侧,因此 TC(q_2)=TRUE. 该多边形中,在当前访问的 Voronoi 通道中,多边形 $p_tp_2p_1p_0p_{20}p_{19}p_{18}p_{17}$ 内的点相对于视点 v 为不可见. 同理,在相应访问的 Voronoi 通道中,TC(q_3)=TRUE.

在图 4 b 中,在相应访问的 Voronoi 通道(粗虚线为相应的 Voronoi 骨架)中,TC(q_1),TC(q_2)和 TC(q_3)为真,其中点 q_3 是从不同的遍历方向(不同方向所访问的 Voronoi 通道)判断了 2 次终止条件;粗虚线为搜索过程中遍历到的 Voronoi 边. 在图 5 中,当从 H_1 的左侧向右侧遍历(从 q_3 朝 q_2 和 q_1 方向遍历)时,遍历到 q_3 时,TC(q_3)=FALSE;访问到 q_2 时,用到垂足 t_3 和 t_1,因为点 t_3 相对于点 q 可见,TC(q_2) 为假;只有遍历到 q_1 时有 TC(q_1)为真,此次遍历才终止. 当从 H_1 的右侧向左侧遍历(从 q_1 朝 q_2 和 q_3 方向遍历)时,遍历到 q_1 时,TC(q_1) 为假;遍历到 q_2 时,用到垂足 t_2 和 t_1,因为点 t_2 对于点 q 可见,同样 TC(q_2)=FLASE;只有遍历到 q_3 时有 TC(q_3)=TRUE,此次遍历才终止.

基于上面的描述,算法 1 给出了本文算法的大致过程描述. 这里,假设给定一个有 n 条边和 $h\geqslant 0$ 个"洞"的多边形 P 以及 P 中一点 q,在预处理阶段已计算出 P 的 Voronoi 图 VD(P).

算法 1. 计算点的可见多边形.

Step1. 判断 q 所在的 Voronoi 区域 VR(o). 假设 $o=p_ip_{i+1}$.

Step2. 计算 VSP(q,p_i)和 VSP(q,p_{i+1}),同时计算 $SP_c(q,p_i)$和 $SP_c(q,p_{i+1})$.

Step3. 计算 o 对于 q 的可见部分.

Step4. 取 VSP(q,p_{i+1})最后 2 个点 q_{r-1},q_r;q_r 为多边形的顶点.

Step5. 如果 TC(q_{r-1})=FALSE {

Step5.1. 取 $q_{r-1}q_r$ 右侧关联边为新的 p_ip_{i+1};

Step5.2. 如果 $p_ip_{i+1}=o$,算法结束;

Step5.3. 计算 VSP(q,p_i)和 VSP(q,p_{i+1}),同时计算 $SP_c(q,p_i)$和 $SP_c(q,p_{i+1})$ } 否则 {

Step5.4. 取 $q_{r-2}q_{r-1}$ 右侧关联边为新的 p_ip_{i+1}；

Step5.5. 计算 VSP(q,p_i)和 VSP(q,p_{i+1})，同时计算 SP$_c(q,p_i)$和 SP$_c(q,p_{i+1})$ }.

Step6. 转 Step3.

需要说明的是：算法 1 中的 Step2，Step5.3，Step5.5 计算 VSP(q,p_i)和 VSP(q,p_{i+1})的方法不同. 本文将在下一节分别对它们进行讨论.

4　计算 VSP 和 SP$_c$

4.1　初始 VSP 和 SP$_c$ 的计算

算法 1 的 Step1 首先确定给定点 q 所在的 Voronoi 区域 VR(o)，可通过逐个测试每个 VR 来实现，需要 $O(n)$时间. 算法 1 的 Step2 用于计算初始的 Voronoi 最短路径和 SP$_c$. VD(P)中加入点 a 和边 qa，令 VSP$(q,p_i)=\{q,a,q_k,\cdots,q_0,p_i\}$，VSP$(q,p_{i+1})=\{q,a,q_{k+1},\cdots,q_j,p_{i+1}\}$，其中 $q_1,q_2,\cdots,q_j$ 为 VR(o)的 Voronoi 顶点，a 在 q_k 和 q_{k+1}之间. 采用类似文献[25]方法，在遍历 VSP(q,p_i)(VSP(q,p_{i+1}))的同时，找到被访问 Voronoi 边的两侧的关联凹顶点，逐次加入到已生成的 SP$_c$ 中，修改生成新的 SP$_c$，最后可得到 SP$_c(q,p_i)$(SP$_c(q,p_{i+1})$)；如图 6 所示，VSP$(q,p_{13})=\{q,a,q_8,q_7,\cdots.q_0,p_{13}\}$，VSP$(q,p_0)=\{q,a,q_9,q_{10},\cdots,q_{14},p_0\}$. 在遍历 VSP$(q,p_{13})$(VSP$(q,p_0)$)的同时，计算 SP$_c$，求得 SP$_c(q,p_{13})=\{q,p_7,p_9,p_{13}\}$，(SP$_c(q,p_0)=\{q,p_4,p_2,p_0\}$). 由于 qp_7 在 qp_4 的左侧，因此边 p_0p_{13} 相对于点 q 的可见部分为 q_rq_l，其中 q_rq_l 为 qp_7，qp_4 与 p_0p_{13} 的交.

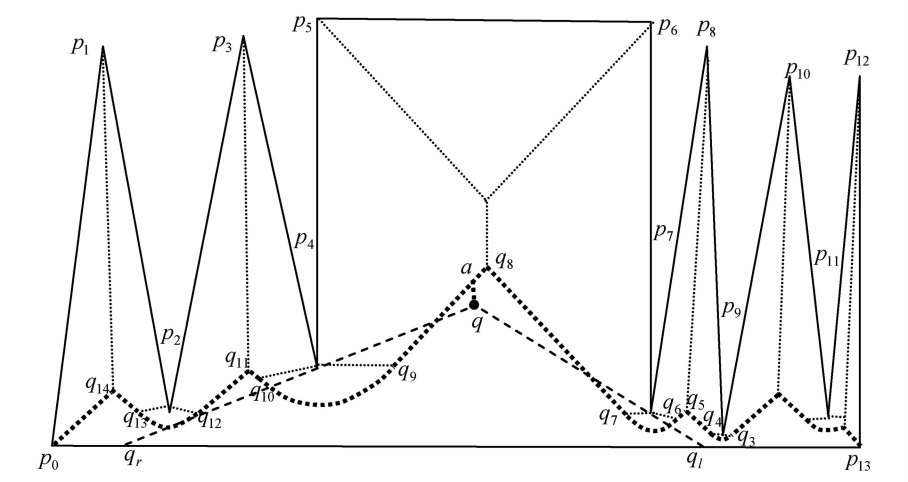


图 6　计算初始 Voronoi 骨架路径

注意到，在从 a 到 p_{13}的访问过程中，当访问到 Voronoi 顶点 q_4 时，TC(q_4)=TRUE，这时没必要继续访问 q_3,q_2,q_1 和 q_0. 这时得到的 SP$_c$ 的第一条边 vp_7 就是最后用于计算 $p_{13}p_0$ 相对于 q 的可见部分的那条射线. 不难证明，即使遍历完 q_3,q_2,q_1 和 q_0，所得到的 q 到 p_{13}的最短路径的第一条边仍为 qp_7；由于 q_3,q_2,q_1 和 q_0 在 qp_7 左侧，因此，这里可以测试终止条件，减少计算.

4.2　后续 VSP 和 SP$_c$ 的计算

在已计算的 VSP(q,p_{i+1})和 SP$_c(q,p_{i+1})$的基础上，考虑如何快速确定下一条边对应的 2 条 VSP 和 SP$_c$，这取决于 TC(q_{r-1})的值：

1) TC(q_{r-1})=FALSE 时，可以继续往前遍历. 这时，取 VSP(q,p_{i+1})中的最后 2 个点 q_{r-1}和 q_r. 当前已知 q_r 为多边形的顶点 p_{i+1}，此时，Voronoi 边 $q_{r-1}q_r$ 的右侧关联边为 $p_{i+1}p_{i+2}$，VSP(q,p_{i+1})和 SP$_c(q,p_{i+1})$已知，只需计算 VSP(q,p_{i+2})和 SP$_c(q,p_{i+2})$. 令 VSP(q,p_{i+2})=VSP(q,p_{i+1})，SP$_c(q,p_{i+2})$=SP$_c(q,p_{i+1})$；对 VD(P)进行遍历直到到达 p_{i+2}对应的叶节点，同时对于这个过程中所遍历的 Voronoi 边，如果该边原来被遍历过，则其已在 VSP(q,p_{i+2})中，需要将其从 VSP(q,p_{i+2})中删除；否则，将其追加到 VSP(q,p_{i+2})中，同时修改 SP$_c(q,p_{i+2})$.

如图 3 中，已知 VSP$(q,p_4)=\{qabcdefgp_4\}$，需要在其基础上计算 VSP(q,p_5). VSP(q,p_5)初始值为$\{qabcdefgp_4\}$，在对 VD(P)进行遍历到达 p_5 对应的叶节点过程中，需要从 VSP(q,p_5)中删除 gp_4 和 fg，加入 fh 和 hp_5，得到 VSP$(q,p_5)=\{qabcdefhp_5\}$. 在这个过程中，同时修改 SP$_c(q,p_5)$，最后得到 SP$_c(q,p_5)=\{qp_6p_5\}$.

类似于第 4.1 节中计算初始 Voronoi 骨架路径

所做的分析和处理，在朝 p_{i+2} 遍历的过程中，同时进行终止条件的测试，遇到满足终止条件的情况就停止遍历，这时得到的 SP_c 的第一条边可用于边的可见部分的计算. 遇到这种情况时，计算完边的可见部分后，需转算法 1 的 Step5. 4.

2) $TC(q_{r-1})=TRUE$ 时，不用继续遍历. 这时，取 $VSP(q,p_{i+1})$ 中的最后 3 个点 q_{r-2},q_{r-1},q_r. 当前已知 q_r 为多边形的顶点 p_{i+1}，此时，设 Voronoi 边 $q_{r-2}q_{r-1}$ 的右侧关联边为 p_jp_{j+1}；进行如下处理：

1) 令初始 $VSP(q,p_j)$ 为 $VSP(q,p_{i+1})$，初始 $SP_c(q,p_j)$ 为 $SP_c(q,p_{i+1})$. 沿 $VR(p_jp_{j+1})$ 朝 p_j 方向遍历，将遍历的 Voronoi 边加入 $VSP(q,p_j)$，并同时计算 $SP_c(q,p_j)$；

2) 令初始 $VSP(q,p_{j+1})$ 为 $VSP(q,p_{i+1})$，初始 $SP_c(q,p_{j+1})$ 为 $SP_c(q,p_{i+1})$. 沿 $VR(p_jp_{j+1})$ 朝 p_{j+1} 方向遍历，将遍历的 Voronoi 边从 $VSP(q,p_{j+1})$ 中删除或加入，并同时计算 $SP_c(q,p_{j+1})$.

然后令 $i=j$，继续新的处理. 上述处理过程中，也都可以考虑进行终止条件的判断.

在图 4 a 中，算法会在 q_2 处停止继续往前遍历，将转向处理边 p_2p_3. 用上面方法计算得到 $SP_c(q,p_2)=\{q,p_{17},p_2\}$，$SP_c(q,p_3)=\{q,p_{17},p_3\}$，可知 p_2p_3 相对于 q 不可见. 算法也会在 q_3 处停止继续往前遍历，将转向处理凹顶点 p_9 开始回溯. 在图 4 b 中，算法会在 q_1 处停止继续往前遍历，将转向处理边 p_3p_4. 用上面方法计算得到 $SP_c(q,p_4)=\{q,p_4\}$，$SP_c(q,p_3)=\{q,p_4,p_3\}$，可知 p_3p_4 相对于 q 不可见.

5 算法分析与实例

本文算法在预处理阶段需要构造多边形的 Voronoi 图，一旦构造完成，其可用于任意点的可见多边形查询. 对于简单多边形，其 Voronoi 图可在 $O(n)$ 时间内构造出来[26]；对于“带洞”多边形的 Voronoi 图的构造，则需要 $O(n\lg n)$ 时间[27]. 对于多边形 P，因为有 $e\leqslant 2(n+k)+3h-3$，其中 e 为 $VD(P)$ 中 Voronoi 边的边数，n 为多边形的边数，k 为多边形中凹顶点的个数，h 为多边形内边界的个数，因此存储 $VD(P)$ 所需的空间为 $O(n)$[1]. 算法在遍历过程中，只需保存 2 条 VSP 和 SP_c，所以均只需要 $O(n)$ 的存储空间. 在多边形 P 中，点 q 的可见多边形 $V(q)$ 是一个星形多边形，其原始边的顶点或为 P 的顶点，或在 P 的边的内部. 后者也为 $V(q)$ 的构造边的一个顶点（远离 q），其可以看作是以 q 为视点的构造边的另一个顶点（靠近 q，是 P 的凹顶点）在 P 的边上的投影. 因此这类点的数目不会超过 P 的凹顶点数目；可得 $O(|V(q)|)=O(n)$；本文算法只需 $O(n\lg n)$ 预处理时间和 $O(n)$ 空间.

算法 1 可在 $O(n)$ 时间内确定点 q 所在的 Voronoi 区域，且确定初始 VSP 和 SP_c 需要 $O(n)$ 的时间；新的 VSP 和 SP_c 只需要对原来的更新即可. 这个过程中，最坏情况下，每条 Voronoi 边最多被访问 $2h$ 次，每个多边形顶点最多被处理 $4h$ 次，时间复杂度为 $O(hn)$. 故在已构造多边形 Voronoi 图的前提下，本文算法计算点的可见多边形的整体时间复杂度为 $O(hn)$.

但是，如图 7 所示的例子，灰色区域为 $V(q)$，红色虚线为本文算法计算 $V(q)$ 所访问到的 Voronoi 边，蓝色线为本文算法计算 $V(q)$ 所访问到的多边形的边；可以看出，本文算法主要在 $V(q)$ 附近进行遍历. 算法实验测试表明，本文算法访问到的 Voronoi 边的数目和 $V(q)$ 中的边的数目是呈近似线性关系（如图 8 所示，其中横轴方向为访问到的 Voronoi 边的数目，纵轴为 $V(q)$ 中边的数目）. 由于 $O(|V(q)|)=O(n)$，所以本文算法运行时间与 $|V(q)|$ 和 n 均呈线性关系，即它是一个近似线性算法且是一个近似输出敏感算法.

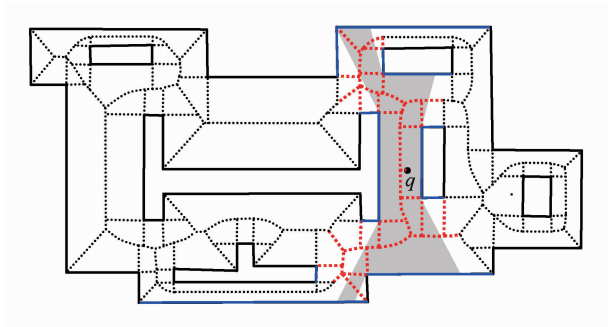


图 7 算法的一个实例

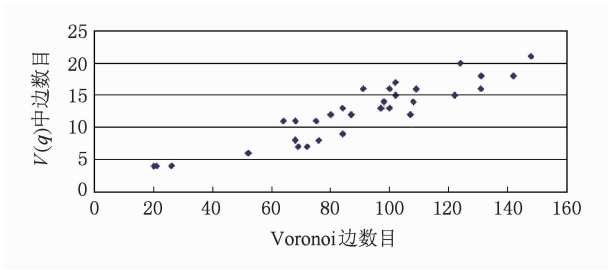


图 8 图 7 中给出的算法实例的分析

目前，本文算法已应用在我们研发的基于 Voronoi 图和多触控技术的三维虚拟博物馆设计与漫游系统中. 该系统以多边形的 Voronoi 图为基本数据结构，支持数据组织、协同设计、可见性计算、路径规划、碰撞检测以及点定位等，可有效地组织数据并提高计算/查询效率. 图 9 右下图所示为系统的硬件环境，

其中多触控屏幕用于二维设计和交互漫游，另一个显示器用于显示三维场景，系统也可通过手机辅助操作三维场景；左上图为多触控屏幕上的二维界面，在上面可以用手自然地进行交互设计并操作场景漫游，其中的灰色区域为当前视点所看到的区域（由视角限制，可在 VP(v)的基础上用二分法快速计算出来）；左下图是相应三维场景的俯视图；右上图则是当前视点所看到的三维场景.

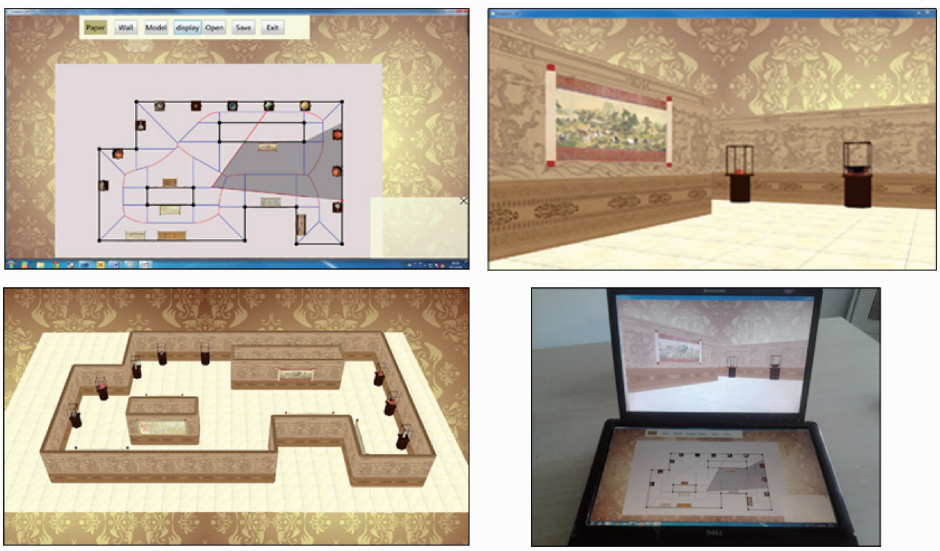


图 9　算法在虚拟博物馆中的一个应用实例

6　结　　论

本文讨论了有 n 条边和 $h \geqslant 0$ 个“洞”的多边形 P 中任意点的可见多边形计算问题，给出了一种基于 Voronoi 图的点的可见多边形计算方法. 本文算法对 Voronoi 图 VD(P)进行深度优先搜索，在计算 VSP 和局部最短路径的基础上计算给定查询点的可见多边形. 本文贡献主要在于：提出了与 VSP 对应的 Voronoi 通道概念，及其相应的局部最短路径概念；给出了计算局部最短路径的算法，以及对 Voronoi 图进行遍历时的终止条件，使得算法可以在“带洞”多边形中应用，并只在所给查询点的可见多边形周围的局部范围内进行遍历. 算法对于简单多边形预处理时间为 $O(n)$，对于带洞多边形预处理时间为 $O(n \lg n)$，空间复杂性都为 $O(n)$. 在运行阶段，对于简单多边形，算法时间复杂度为 $O(n)$；对于带洞多边形，最坏情况下时间复杂度为 $O(h * n)$. 但实际数据测试表明，本文算法运行时间与 $|V(q)|$ 和 n 均呈线性关系，即本文算法是一个近似线性算法且是一个近似输出敏感算法. 另外，本文算法数据结构简单、剖分空间合理且易于理解和实现，可以应用于如虚拟博物馆在内的三维虚拟场景，其中多边形的 Voronoi 图可以作为一种统一的数据结构用于数据组织，并进行可见性计算、路径规划、碰撞检测、点定位和光照计算等.

本文算法也可扩展到以曲线为边界的多边形，用于计算点的可见多边形或曲线的弱可见多边形，这是我们未来的一项工作.

参考文献(References)：

[1] Wang Jiaye, Wang Wenping, Tu Changhe, *et al*. Computational geometry and its applications [M]. Beijing: Science Press, 2011: 52-53 (in Chinese)
(汪嘉业，王文平，屠长河，等. 计算几何及应用[M]. 北京：科学出版社，2011：52-53)

[2] Lu L, Choi Y K, Wang W P. Visibility-based coverage of mobile sensors in non-convex domains [C] //Proceedings of the 8th International Symposium on Voronoi Diagrams in Science and Engineering. Washington D C: IEEE Computer Society Press, 2011: 105-111

[3] Wang Wencheng, Wei Feng, Wu Enhua. Visibility determination for rendering large scale scenes [J]. Journal of Computer-Aided Design & Computer Graphics, 2006, 18(2): 161-169 (in Chinese)
(王文成，魏　峰，吴恩华. 绘制大规模场景的可见性计算技术[J]. 计算机辅助设计与图形学学报，2006，18(2)：161-169)

[4] Cohen-Or D, Chrysanthou Y, Silva C T, et al. A survey of visibility for walkthrough application [J]. IEEE Transactions on Visualization and Computer Graphics, 2003, 9(3): 412-431

[5] Govindaraju N K. Efficient visibility-based algorithms for interactive walkthrough, shadow generation, and collision detection [D]. Chapel Hill: University of North Carolina at Chapel Hill, 2004

[6] Mattausch O, Bittner J, Wonka P, *et al*. Optimized subdivisions for preprocessed visibility [C] //Proceedings of Graphics Interface. New York: ACM Press, 2007: 335-342

[7] Li Sheng. Study on acceleration technique of rendering for large-scale terrain environments [D]. Beijing: Institute of Software of Chinese Academy of Sciences, 2005 (in Chinese)
(李 胜. 大规模室外地形场景加速绘制技术研究[D]. 北京: 中国科学院软件研究所, 2005)

[8] Lowe N, Datta A. A technique for rendering complex portals [J]. IEEE Transactions On Visualization And Computer Graphics, 2005, 11(1): 81-90

[9] Lefebvre S, Hornus S. Automatic cell-and-portal decomposition [R]. Nice: INRIA, 2003

[10] Lerner A, Chrysanthou Y, Cohen-Or D. Efficient cells-and-portals partitioning [J]. Computer Animation & Virtual Worlds, 2006, 17(1): 21-40

[11] Sunar M S, Azahar M A M, Mokhtar M K, *et al*. Crowd rendering optimization for virtual heritage system [J]. The International Journal of Virtual Reality, 2009, 8(3): 57-62

[12] Wang L, Yang C L, Qi M, *et al*. Design of a walkthrough system for virtual museum based on Voronoi diagram [C] // Proceedings of the 3rd International Symposium on Voronoi Diagrams in Science and Engineering. Washington D C: IEEE Computer Society Press, 2006: 258-263

[13] Ghosh S K. Visibility algorithms in the plane [M]. Cambridge: Cambridge University Press, 2007: 13-43

[14] Nouri M, Ghodsi M. Space-query-time tradeoff for computing the visibility polygon [C] //Proceedings of the 3d International Workshop on Frontiers in Algorithmics. Heidelberg: Springer, 2009: 120-131

[15] Zarei A, Ghodsi M. A practical approach for planar visibility maintenance [J]. Journal for Geometry and Graphics, 2009, 13(1): 1-16

[16] Guibas L, Hershberger J, Leven D, *et al*. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygon[J], Algoritmica, 1987, 2(1-4): 209-233

[17] Guibas L J, Motwani R, Raghavan P. The robot localization problem in two dimensions [C] //Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms. Pennsylvania: Society for Industrial and Applied Mathematics Philadelphia Press, 1992: 259-268

[18] Bose P, Lubiw S, Munro J I. Efficient visibility queries in simple polygons [J]. Computational Geometry: Theory and Applications, 2002,23 (3): 313-335

[19] Aronov B, Guibas L, Teichmann M, *et al*. Visibility queries and maintenance in simple polygons [J]. Discrete and Computational Geometry. 2002, 27(4): 461-483

[20] Zarei A, Ghodsi M. Query point visibility computation in polygons with holes [J]. Computational Geometry: Theory and Applications, 2008, 39(2): 78-90

[21] Inkulu R, Kapoor S. Visibility queries in a polygonal region [J]. Computational Geometry: Theory and Applications, 2009, 42(9): 852-864

[22] Lu L, Yang C L, Wang J Y. Point visibility computing in polygons with holes [J]. Journal of Information and Computational Science, 2011, 8(16): 4165-4173

[23] Lu L, Yang C L, Wang W Z, *et al*. Voronoi-based potentially visible set and visibility query algorithms [C] // Proceedings of the 8th International Symposium on Voronoi Diagrams in Science and Engineering. Washington D C: IEEE Computer Society Press, 2011: 234-240

[24] Asano T. An efficient algorithm for finding the visibility polygons for a polygonal region with holes [J]. IEICE Transactions, 1985, E68-E(9): 557-559

[25] Yang C L, Qi M, Wang J Y, *et al*. Shortest path queries in a simple polygon for 3D virtual museum [C] //Proceedings of the International Conference on Computational Science and Its Applications. Heidelberg: Springer, 2007: 110-121

[26] Chin F. Finding the constrained Delaunay Triangulation and constrained Voronoi Diagram of a simple polygon in linear time [J]. Siam Journal on Computing, 1998, 28(2): 471-486

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