第 3 章　多边形的 Voronoi 图及其应用

本章介绍多边形的 Voronoi 图，包括其概念、性质、构造方法及若干应用。

3.1　定义与性质

3.1.1　定义

在离散点集的 Voronoi 图中，站点只有离散点，Voronoi 边只来自于两个离散点之间的中分线。但在多边形中，站点包括多边形的顶点和边，它们之间的中分线主要存在以下四种情况。

（1）点与点：其中分线是它们之间连线的垂直平分线，该线将平面一分为二，分别是距离这两个点的最近点的集合。

（2）点与边（点是该边的一个顶点）：其中分线是过所给点且垂直于给定边的直线，该线也将平面一分为二，分别是距离给定点和边的最近点的集合。

这里，点到边的距离是指：如果该点的垂足在边上，则该点到边的距离为该点到其垂足的距离，否则，该点到边的距离为该点到边的顶点中最近的那个距离，如图 3.1 所示。

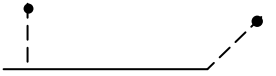


图 3.1　点到边的距离

（3）点与边（点不在该边上）：中分线是一条双曲线，该线上的点到给定点和边所在直线的距离相等。

（4）边与边：中分线是两边所形成夹角的角平分线。特殊情况下，若两边平行，则其中分线为平行于这两条边且距离相等的中线。

由此可见，多边形的两个站点 p_1 和 p_2 的平分线为到 p_1 和 p_2 距离相等的点轨迹，用 $b(p_1, p_2)$表示。半平面 $h(p_1, p_2)$是指平面上到站点 p_1 比 p_2 的距离更近的所有点的集合。

令 P 是多边形的所有站点集合，对于任意站点 $p_i \in P$，其 Voronoi 区域是指平面上到 p_i 的距离比 P 中任何其他站点距离更近的所有点的集合。它等价于平面 $h(p_i, p_j)$的交，其中 $p_j \in P-\{p_i\}$。因此，其可表示为：

$$\mathrm{VR}(p_i)=\cap \{h(p_i, p_j) \mid p_j \in P-\{p_i\}\}$$

令 P 是多边形的所有站点的集合，多边形的 Voronoi 图表示为：

$$\mathrm{VD}(P)=\cup \{\mathrm{VR}(p) \mid p \in P\}$$

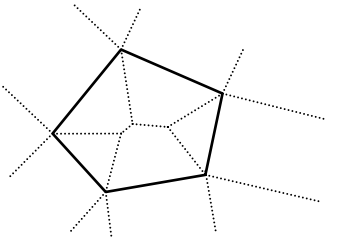


图 3.2　多边形的站点形成的 Voronoi 图，其中虚线是 Voronoi 边

上面的定义实际上是多边形的顶点和边形成的平面直线图的 Voronoi 图（如图 3.2 中虚线部分）。在实际应用中，常用到的多边形的 Voronoi 图实际上是对多边形域中的 Voronoi 剖分。在第 1 章中我们定义了两类多边界多边形。

（1）对于一类多边界多边形，有一个最外面的边界，称之为外边界；其他边界都在它的内部，称之为内边界（如图 3.3（a）中实线部分）。

（2）对于另一类多边界多边形，它不存在包含内边界的外边界（如图 3.3（b）中实线部）。

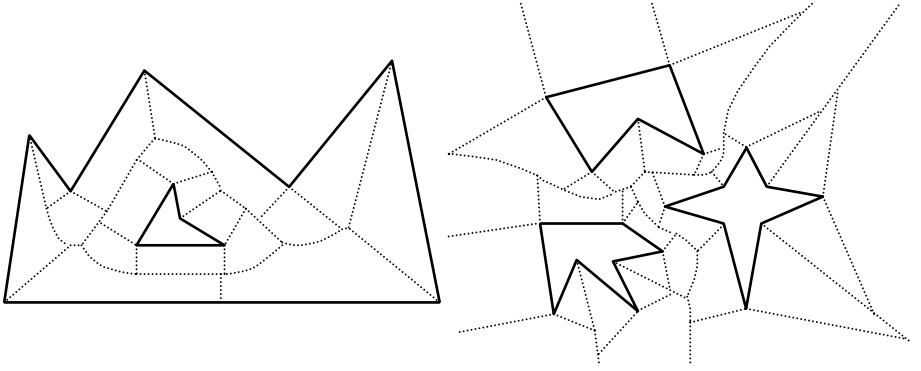
对于这两类多边形，假设外边界是逆时针（顺时针）的，内边界是顺时针（逆时针）的，如果一个人沿这些边界走动，多边形总是位于每条边界的左侧（右侧）。

相应地，对于多边界多边形的 Voronoi 图也分为两类。

（1）对于第一类多边界多边形，其 Voronoi 图位于外边界的内部和所有内边界的外部，并将这部分平面区域分割成许多单元。我们将这类多边界多边形的 Voronoi 图称为内部 Voronoi 图。

（2）对于第二类多边界多边形，其 Voronoi 图位于所有边界的外部，并将这部分平面区域分割成许多单元。我们将这类多边界多边形的 Voronoi 图称为外部 Voronoi 图。

图 3.3 给出了这两类多边界多边形（实线部分）和它们的 Voronoi 图（虚线部分）。在这两类多边界多边形的 Voronoi 图中，只有边和内尖点的 Voronoi 区域非空。因此，我们提及这两类多边界多边形的 Voronoi 图时，所说的站点不包含凸顶点。



(a) 第一类多边形与内部 Voronoi 图　　(b) 第二类多边形与外部 Voronoi 图

图 3.3　多边形及其 Voronoi 图

3.1.2 性质

本节介绍多边形的 Voronoi 图的性质[YangCL2005，YangCL2006c]。为便于描述，下面给出一些符号表示：

- h 表示多边形 P 所含的内边界的数目；
- n、k、s 分别是 P 的所有顶点、内尖点和凸顶点的数目，其中 $n=k+s$；
- t、r 分别是位于 P 的凸包上的所有顶点和边数；
- m、e 分别是 VD(P)的所有 Voronoi 顶点和边数；
- a_v、a_e 分别表示每个 Voronoi 区域包含的 Voronoi 顶点和 Voronoi 边的平均数。

1. 多边形的内部 Voronoi 图的性质

引理 3.1 令 j、i 分别是一棵树中的非叶结点和叶结点的数目。如果每个非叶结点至少有 2 个儿子，且根结点至少有 3 个儿子，则：

$$j \leqslant i-2$$

特别地，如果每个非叶结点有且只有 2 个儿子，且根结点有且只有 3 个儿子，则有：

$$j=i-2$$

引理 3.2 对于多边形 P 的一条内边界 B，VD(P)中必存在一个围绕 B 的环。

证明：VD(P)中必存在一条包围 B 的封闭的线，P 中在该线内侧的点距离 B 比距离 P 中其他任何站点更近。这条封闭的线便是 VD(P)中围绕 B 的环。证毕。

引理 3.3 对于一个有 $h(h \geqslant 0)$个内边界的多边形，其内部 Voronoi 图有

h 个环。

证明：一方面，假设内部 Voronoi 图中的环多于 h 个。设其中一个环的内部没有内边界，且这个环没有边或顶点与多边形的站点相关联。由于这个环所包围的区域与这个多边形的任何边界站点都不关联，所以这与多边形的 Voronoi 图的定义矛盾。另一方面，根据引理 3.2 知内部 Voronoi 图中的环不会少于 h 个。命题成立。证毕。

定理 3.1　对于多边形的内部 Voronoi 图，有：

$$m \leqslant n+k-2+2h$$

证明：如果多边形有内尖点或内边界，则其内部 Voronoi 图存在一些环。为了由其内部 Voronoi 图构造一棵树，我们首先将每个内尖点分成两个顶点，并添加一条短线段连接它们，分别作为修改后的多边形的顶点和边。这样就可以消除内部 Voronoi 图在内尖点处的环。然后，在每个环的一条 Voronoi 边的中点处将其剪断，且在剪断处得到两个剪断点。如图 3.4 所示，分别剪断相应的三条 Voronoi 边，得到剪断点 p_{11} 与 p_{12}、p_{21} 与 p_{22}、p_{31} 与 p_{32}。如果适当的选 h 条 Voronoi 边进行剪断，根据引理 3.3，多边形的 Voronoi 图变成了一棵树。

令一个 Voronoi 顶点为树的根结点，其他 Voronoi 顶点为非叶结点，修改后的多边形的顶点和所有剪断点为叶结点，且 Voronoi 边为树的边。这种情况下，该树有 $n+k+2h$ 个叶结点、m 个非叶结点、$e+h$ 条边。

由于每个 Voronoi 顶点的度至少是 3[Held1991]，所以树的根结点至少有 3 个儿子，非叶结点至少有 2 个儿子。根据引理 3.1，得到

$$m \leqslant n+k-2+2h$$

证毕。

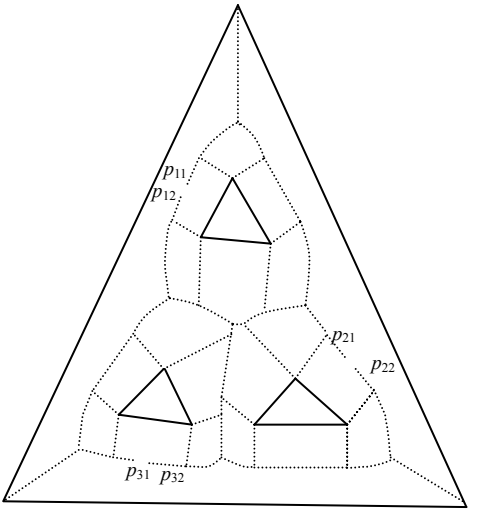


图 3.4　内部 Voronoi 图和被剪断的边

定理 3.2　对于多边形的内部 Voronoi 图，有：

$$e \leqslant 2(n+k)+3h-3$$

证明：根据定理 3.1 的证明，修改后的多边形的内部 Voronoi 图形成一棵树，其共有 $n+k+2h$ 个叶结点、m 个非叶结点和 $e+h$ 条边。由于一棵树的结点数比边数大 1，所以，有

$$e + h \leqslant (m+n+k+2h) - 1$$

$$e \leqslant m+n+k+h-1$$

根据定理 3.1，有

$$e \leqslant (n+k-2+2h) + n + k + h - 1$$

$$e \leqslant 2(n+k)+3h-3$$

证毕。

推论 3.1　对于多边形 P 的内部 Voronoi 图的每个 Voronoi 区域，有

（1）包含 Voronoi 边的平均数 a_e 小于 5；

（2）包含 Voronoi 顶点的平均数 a_v 小于 4。

证明：在多边形的内部 Voronoi 图中，对于一个边站点 o，必然存在一个 VR(o)；对于一个顶点站点 o，只有 o 是内尖点时才存在一个 VR(o)。因此，VD(P)有 $n+k$ 个 Voronoi 区域，且两个相邻的 Voronoi 区域共享一条 Voronoi 边。我们得到，$(n+k)\,a_e = 2e$。

根据定理 3.2，有

$$(n+k)a_e \leqslant 2(2(n+k)+3h-3) \leqslant 4n+4k+6h-6$$

$$a_e \leqslant 4+6h/(n+k)-6/(n+k)$$

在多边形中，每一条内边界的顶点数大于或等于 3，且内尖点数也大于或等于 3。所以，有 $n+k>6h$。

我们得到，$a_e < 4+(n+k)/(n+k)-6/(n+k) < 5-6/(n+k)$。

令 a_e 是一个整数，我们得到 $a_e<5$。

在一个 Voronoi 区域中，Voronoi 顶点数等于 Voronoi 边数减 1。我们得到 $a_v<4$。

证毕。

定理 3.3　对于单边界多边形 P 的内部 Voronoi 图，有：

(1) $m \leqslant n+k-2$

(2) $e \leqslant 2(n+k)-3$

推论 3.2　对于单边界多边形 P 的内部 Voronoi 图的每个 Voronoi 区域，有：

（1）包含 Voronoi 边的平均数 a_e 小于 4；

（2）包含 Voronoi 顶点的平均数 a_v 小于 3。

证明：根据定理 3.3，有：

$$(n+k)\,a_e = 2e \leqslant 2(2(n+k)-3) \leqslant 4n+4k-6$$

$$a_e \leqslant 4-6/(n+k)$$

得 $a_e < 4$，且 $a_v < 3$。证毕。

引理 3.4 令 j、i 分别是一棵树中的非叶结点和叶结点的数目。如果每个非叶结点至少有 2 个儿子，则：

$$j=i-1-\Sigma(\text{每个非叶结点儿子数目}-2)$$

如果考虑每一个 Voronoi 顶点的度，根据引理 3.4 可以得到如定理 3.4 所示的精确公式。

定理 3.4 对于多边形的内部 Voronoi 图，有：

（1）$m \leqslant n+k-1+2h-\Sigma$（每个非叶结点儿子数目−2）

（2）$e \leqslant 2(n-k)-2+3h-\Sigma$（每个非叶结点儿子数目−2）

根据引理 3.4，采用类似定理 3.1 和定理 3.2 的方法很容易证明定理 3.4。

2. 多边形的外部 Voronoi 图的性质

由第 2 章我们知道，对于点集 P 的 Voronoi 图，点 p 是点集 P 凸包上的点，当且仅当 p 的 Voronoi 区域是非封闭的。对于多边形的外部 Voronoi 图，也有类似的性质。

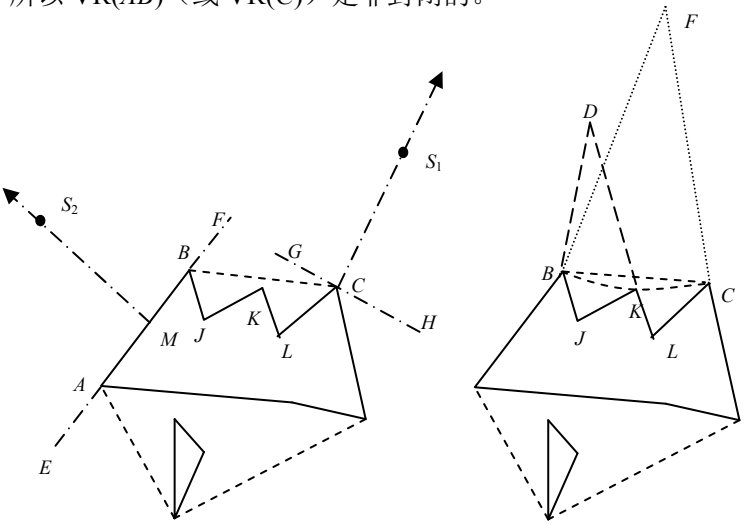
引理 3.5 多边形凸包上的顶点必是多边形的凸顶点。

定理 3.5 对于多边形 P 的站点 p，VR(p)是非封闭的当且仅当 p 在 P 的凸包上。

证明：

（1）**必要性**：如图 3.5（a）所示，令 $p=AB$（或 C）分别是 P 的凸包上

的站点。则可通过 *AB*（或 *C*）作直线 *EF*（或 *GH*），使 *P* 的顶点和边都在 *EF*（或 *GH*）的同侧。过 *AB* 上的任意一点 *M*（或 *C*）作 *EF*（或 *GH*）的垂线 MS_2（或 CS_1）。显然，从 MS_2（或 CS_1）上的任意一点 S_2（或 S_1）到 *P* 上任一其他站点的距离都大于 d(S_2，*M*)（或 d(S_1，*C*)）。因此，S_2（或 S_1）在 VR(*AB*)（或 VR(*C*)）中。由于 S_2（或 S_1）可以是射线 MS_2（或 CS_1）上的任意一点，所以 VR(*AB*)（或 VR(*C*)）是非封闭的。



(a) 必要性　　(b) 充分性

图 3.5　定理 3.5 的证明图例

（2）**充分性**：假定 *P* 的站点 *p* 不在 *P* 的凸包上，现证明 VR(*p*)是封闭的。如图 3.5（b）所示，设给定站点 *p* 是边 *JK*（或顶点 *K*），可找到 *P* 的凸包上边 *BC*，且 *P* 的边界上 *B* 到 *C* 的链中包含边 *JK*（或 *K*），*JK*（或 *K*）在 *CB* 的左侧。令 L_1 表示 *P* 的边界上 *B* 到 *C* 的链。过 *B* 和 *C* 作一个圆弧，该圆弧除了与 L_1 的一些站点相切外不和它们相交。*F* 是圆心，可知该圆弧的半径是有限的。易证 *P* 的外部及三角形△*FBC* 的外部任一点（例如 *D*），其到 *B* 或 *C* 的距离小于 *D* 到 L_1 上的任意站点 *p* 的距离，也就是 VR(*p*)的任意一点必然落在 *CFB* 和 L_1 围成的多边形内。即 VR(*p*)是有界的，也是封闭的。证毕。

由引理 3.5 和定理 3.5，可以得到：$r \leqslant t \leqslant s \leqslant n$，且 $r \geqslant 0$，$t \geqslant 3$，$s \geqslant 3$。

定理 3.6　对于多边形的外部 Voronoi 图，有

$$m \leqslant n + s + 2h - r - t - 2$$

证明：作一个圆 C，使多边形 P 的所有顶点和 VD(P)的所有顶点都包含在它内部（见图 3.6）。根据定理 3.5，只要这个圆足够大，C 只与 VD(P)的所有非封闭 Voronoi 区域相交，并被分成 $t + r$ 条弧。由 C 上的 $t + r$ 条弧和 VD(P)组成的图 $G = (V, E)$表示如下：

$V = V_1 \cup V_2 \cup V_3$，

$E = E_1 \cup E_2$。

其中，

V_1 = {Voronoi 顶点}，

V_2 = {C 与 Voronoi 边的交点}，

V_3 = {多边形 P 的顶点}，

E_1 = {Voronoi 边}，

E_2 = {C 被所有非封闭 Voronoi 区域分成的 $t + r$ 条弧}，

$|V| = |V_1| + |V_2| + |V_3| = m + t + r + n$，

$|E| = |E_1| + |E_2| = e + r + t$。

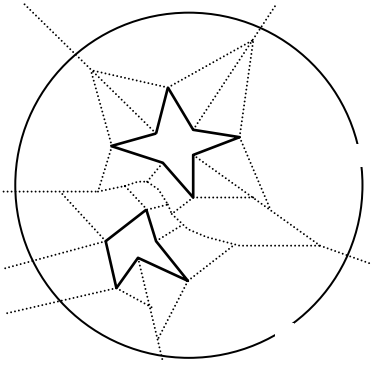


图 3.6　定理 3.6 的证明图例

但是，图 $G=(V，E)$具有 $s+h$ 个环，其中包括 s 个凸顶点所在的环和包围边界的 h 个环。为了得到一棵树，我们做如下处理。

首先，将 V_3 中的 P 的每个凸顶点分成 P 的两个顶点，并在两顶点之间连线，作为 P 的一条边。则有：

$G'=(V'，E)$，

$V'=V_1\cup V_2\cup V_3'$，

$V_3'=V_3\cup\{P$ 的每个凸顶点新分成的一个顶点$\}$，

$|V'|=|V_1|+|V_2|+|V_3'|=m+t+r+n+s$，

$|E|=e+r+t$。

然后，在每一个环上，我们在 C 或 VD(P)上适当的选择 h 条圆弧或 Voronoi 边，分别在中点处把它们剪断，这些中点分别变成 2 个剪断点，圆弧或 Voronoi 边由一条变成了两条（见图 3.6）。我们得到：

$G''=(V''，E')$，

$V''=V_1\cup V_2\cup V_3'\cup V_4$，

$E_1'=E_1'\cup E_2'$，

$V_4=\{$新生成的剪断点$\}$，

$E_1'=\{$未被剪断的 Voronoi 边$\}\cup\{$由被剪断的 Voronoi 边生成的边$\}$，

$E_2'=\{$未被剪断的圆弧$\}\cup\{$由被剪断的圆弧生成的边$\}$，

$|V''|=|V_1|+|V_2|+|V_3'|+|V_4|=m+t+r+n+s+2h$，

$|E'|=e+r+t+h$。

在图 $G''=(V''，E')$中取一个度为 3 的非叶结点作为根结点，这样，就得

到一棵树。该树共有 $e+r+t+h$ 条边、$m+r+t$ 个非叶结点、$n+s+2h$ 个叶结点。根据引理 3.1，有：

$$m+r+t \leqslant n+s+2h-2$$

$$m \leqslant n+s+2h-r-t-2$$

证毕。

定理 3.7 对于多边形的外部 Voronoi 图，有

$$e \leqslant 2n+2s+3h-r-t-3$$

证明：由于一棵树的结点数比边数多 1 个。根据定理 3.6 的证明，有：

$$e+r+t+h+1=n+s+2h+m+r+t$$

$$e=n+s+m+h-1$$

根据定理 3.6，有：

$$e \leqslant 2n+2s+3h-r-t-3$$

证毕。

推论 3.3 对于多边形的外部 Voronoi 图 VD(P)中的每一个 Voronoi 区域，有：

（1）其边界上的 Voronoi 边的平均数小于 4；

（2）其边界上的 Voronoi 顶点的平均数小于 3。

证明：令 a_e 和 a_v 分别表示多边形 P 的外部 Voronoi 图 VD(P)的一个 Voronoi 区域的边界上的 Voronoi 边和顶点的平均数。由于 VD(P)有 $n+s$ 个 Voronoi 区域，且两个相邻的 Voronoi 区域共有一条 Voronoi 边，所以有

$$a_e(n+s)=2e \leqslant 2(2n+2s+3h-r-t-3) \leqslant 4(n+s)+6h-2(r+t)-6$$

$$a_e \leqslant 4+6h/(n+s)-2(r+t)/(n+s)-6/(n+s)$$

$$a_e < 5 - 2(r + t) / (n + s) - 6 / (n + s)$$

由于 a_e 是一个整数，所以，$a_e < 5$。

由于在一个 Voronoi 区域的边界上，有 $a_v < a_e - 1$，所以，$a_v < 4$。

证毕。

3. 凸多边形的外部 Voronoi 图的性质

凸多边形是一个单边界多边形，只有一条封闭的边界。对于凸多边形，由于其顶点和边都在其凸包上，即有 $r = t = s = n$，且 $h = 1$，所以它的外部 Voronoi 图的性质存在一定的特殊性（如图 3.7 所示）。

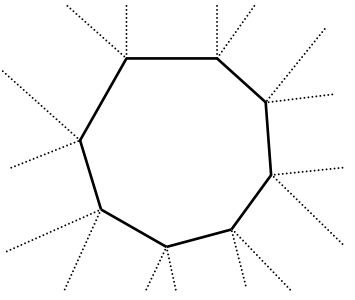


图 3.7　凸多边形（实线）及其外部 Voronoi 图（虚线）

推论 3.4　对于凸多边形的每个站点的外部 Voronoi 区域，有：

（1）是非封闭的；

（2）只有 2 条 Voronoi 边；

（3）有 0 个 Voronoi 顶点。

推论 3.5　对于凸多边形的外部 Voronoi 图，有：

（1） $m = 0$；

（2） $e = 2n$。

推论 3.4 和推论 3.5 是比较容易证明的，此处略。

3.2 构 造 方 法

Voronoi 图的计算方法有许多，主要有分而治之算法、扫描线算法、增量算法等。

分而治之算法主要是将站点划分为两部分，递归计算每一部分的 Voronoi 图，再将它们合并，最终得到整个站点集合的 Voronoi 图。文献[Kirkpatrick1979，Lee1982，Yap1987]分别给出了在 $O(n\log n)$时间内计算 PSLG 的 Voronoi 图的分而治之算法。对于直线段为站点的情况，分割生成两个站点子集比点为站点的情况复杂得多，而且连接两个站点子集的 Voronoi 图的步骤也是比较困难且难以实现的。

扫描线算法主要使用扫描线在平面上逐渐扫描生成站点，并不断更新 Voronoi 图的信息，最后得到整个站点集合的 Voronoi 图。扫描线算法可以看作一种特殊的增量算法。文献[Fortune1986]给出了可在 $O(n\log n)$时间内构造直线段 Voronoi 图的扫描线算法。

增量算法通过每次添加一个生成站点并不断在局部范围内修改已有 Voronoi 图来获取新 Voronoi 图的方式来计算整个站点集合的 Voronoi 图。文献[Boissonnat1992，Klein1993，Held2001]给出了在 $O(n\log n)$时间内构造 PSLG 的 Voronoi 图的增量算法。文献[Held2001]以文献[Sugihara2000]中的拓扑向导（Topology-Oriented）方法为基础，实现了一种计算点和线段的 Voronoi 图的增量算法。该算法从处理站点的空集开始，Voronoi 图也相应为空；然后每次向已处理的站点集合中添加一个新的站点，并对已有 Voronoi 图做相应修改，最终得到整个站点集合的 Voronoi 图。每次对已有 Voronoi 图做相应修改生成新的 Voronoi 图时，依据 Voronoi 图本身固有的拓扑属性，删除原来的 Voronoi 顶点，增加新的 Voronoi 图顶点。

文献[Aggarwal1989]给出了计算凸多边形的 Voronoi 图的 $O(n)$算法。文献[Klein1995，Chin1995，Chin1998]则给出了计算单边界多边形的 Voronoi 图的 $O(n)$算法。文献[Chin1995]首先将简单多边形分解为正则直方图

（Pseudonormal Histograms），然后通过分解为影响区域直方图（Influence Histograms），最后转化为 *xy* 单调的直方图（*xy* Monotone Histograms）；计算 *xy* 单调的直方图的 Voronoi 图，将其合并得到整个多边形的 Voronoi 图。相关的详细综述信息可阅文献[Sack2000，Held2001]等。

下面给出一种比较简单的多边形 Voronoi 图构造算法，该算法虽然复杂度较高，但易于理解与实现。

算法采用半平面法构造多边形的 Voronoi 图，为多边形每个站点独立地计算其 Voronoi 区域，即计算其 Voronoi 区域边界的各条 Voronoi 边。

算法 3.1　半平面法构造多边形的 Voronoi 图

（1）为多边形各站点编号，计算两两站点间的平分线；

（2）为多边形每个站点计算其 Voronoi 区域：

① 计算当前站点与其前后相邻站点的平分线；

② 设定当前站点与其前一个相邻站点的平分线为当前平分线；

③ 计算当前站点其他相关平分线与当前平分线的有效交点；

④ 按照与当前平分线起始点的距离大小对这些交点进行排序；

⑤ 将最近交点作为当前平分线的终点，输出一条 Voronoi 边；

⑥ 选择形成最近交点的平分线作为当前平分线；

⑦ 若当前平分线为当前站点与后一个相邻站点的平分线，转到⑧，否则转③；

⑧ 完成对该站点 Voronoi 区域的计算；

（3）输出多边形的 Voronoi 图。

3.3 应 用 实 例

本节主要介绍多边形的 Voronoi 图的应用，包括凸多边形间的求交与距离计算，可见性计算以及路径规划等，并介绍这些算法在虚拟博物馆中的集成应用。

3.3.1 两个凸多边形的求交计算

凸多边形间的求交计算在计算机游戏、计算机动画、虚拟现实、运动规划、图形学等方面应用广泛。目前已有很多算法，最著名的是 O'Rourke 的追逐算法[O'Rourke1994]，其时间复杂度为 $O(n)$。这里，我们介绍一种基于外部 Voronoi 图的凸多边形间的求交算法，时间复杂度仍为 $O(n)$，但算法效率和鲁棒性更好[YangCL2006a]。

由 3.2 节可知，凸多边形的外部 Voronoi 图非常特殊，每条 Voronoi 边都垂直于凸多边形的一条边。显然，对于一个凸多边形来说，在逆时针方向上顶点和边的 Voronoi 图是交替出现的，如图 3.7 所示。

算法的主要思想是：让两凸多边形 P 和 Q 的并集和交集的边界上的两边 a 和 b 相互追逐，并计算它们的交点，直到找到了两多边形的所有交点，或者是成功地判定了两多边形的空间位置（包含或分离）。在追逐过程中，b 沿着一个多边形（不妨假设为 Q，这时遍历的 P 和 Q 的并集的边界在 Q 上）的边界连续移动并穿过另一个多边形（不妨假设为 P）的外部 Voronoi 区域，a 同时在 P 的边界上相应地移动，直到 a 和 b 相交于一点。这时，a 和 b 互换，b 沿着 P 的边界移动并穿过 Q 的外部 Voronoi 区域，a 同时在 Q 的边界相应地移动。因此，b 总是在 a 所在多边形的外部。

算法的主要步骤是判断 b 和另一个凸多边形的外部 Voronoi 图的关系，据此决定是沿着原来的多边形的边界继续遍历，还是更换到另一个多边形的边界上进行遍历。如图 3.8 所示，a 是 P 的边 p_jp_{j+1}；b 是 Q 的边，s 是 b 的起点，也是 Q 的顶点，其位于 VR(a)内。b 与 VR(a)存在以下三种可能的关系。

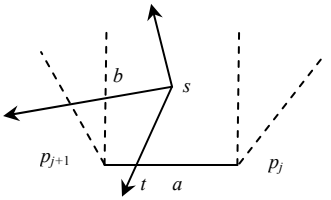


图 3.8　被遍历的边与另一多边形的外部 Voronoi 图的关系

- b 与 VR(a)的边界不相交：这时，令 s 为 b 的另一端点，即令 b 在 Q 上前进一条边，继续新的遍历。
- b 与 VR(a)的 Voronoi 边相交：这时，由于 b 首先穿过 VR(p_{j+1})的边界，取 $a=p_{j+1}$，继续新的遍历。
- b 与 a 相交：设 a 与 b 相交于点 t，令 $a = b$，$b = t\,p_{j+1}$，$s = t$，继续新的遍历。这里，t 为找到的 P 与 Q 的一个交点。

通过上述三种情况的处理，保证了 b 总是在 $P \cup Q$ 的边界上遍历，a 总是在 $P \cap Q$ 的边界上遍历，a 和 b 一旦相遇，即可得到 P 与 Q 的一个交点，这时需要交换 a 和 b 后继续遍历。

算法 3.2　两个凸多边形的求交计算

（1）找出两凸多边形的所有顶点中的最低的顶点（如果存在多个，取最左侧的那个），将其作为整个遍历的起点，记为 s_0。令 $s=s_0$。用 Q 表示 s_0 所在的凸多边形，P 表示另一个凸多边形。

（2）判断 s 所在的 P 的 Voronoi 区域。设 s 在 P 的站点 $p_j p_{j+1}$（顶点可以看做特殊的边）的 Voronoi 区域中，令 b 为以 s 为起点的 Q 的边（逆时针方向），$a=p_j p_{j+1}$。

（3）令 L={}。//用于记录 P 与 Q 的交点。

（4）如果 b 与 VR(a)的边界不相交，则令 s 为 b 的另一端点，b 为 s 起点的 Q 的边（逆时针方向），转（7）。

（5）如果 b 与 VR(a)的 Voronoi 边相交，则 $a=p_{j+1}$，$j=j+1$，转（7）。

（6）如果 b 与 a 相交，则计算 a 与 b 的交点 t，令 $a=b$，$b=t\,p_{j+1}$，$p_j=s$，$s=t$，$L=L\cup\{t\}$，转（7）。

（7）如果 $s==s_0$，转（8）；否则，转（4）。

（8）如果 L!={}，则 L 中的点为 P 与 Q 的所有交点，依次输出它们。

（9）否则，P 与 Q 没有交点。

上述算法最后可以给出 P 与 Q 是否相交的结果。如果它们不相交，可进一步判断它们的空间位置关系，即是分离还是包含关系：如果整个算法过程中，a 和 b 未发生交换，且只访问了 P 的部分顶点，则 P 与 Q 是分离的，否则 Q 包含 P。

另外，对上述算法稍作修改，可让 L 记录 P 与 Q 交集的所有顶点。类似地，也可以记录 P 与 Q 并集的所有顶点。

3.3.2 两个分离凸多边形的距离计算

本节主要讨论两个分离凸多边形间的距离计算问题。目前解决这个问题的最优算法的时间复杂度为 $O(\log m+\log n)$。本节对两个分离凸多边形的外部 Voronoi 图之间的关系进行深入分析，并介绍基于 Voronoi 图来计算两凸多边形间距离的 $O(\log m+\log n)$ 算法[YangCL2006b]。

1. 算法思想

用 P 和 Q 表示欧氏几何空间中的两个分离的凸多边形。P 和 Q 之间的距离为：

$d(P，Q)$=min{$d(p，q)|p$，q 分别是 P 和 Q 的点，$d(p，q)$为 p 与 q 间的欧氏距离}

如果有 P 和 Q 的点 p^*和 q^*，满足 $d(p^*，q^*)=d(P，Q)$，则称<p^*，q^*>为 P 和 Q 的一个最短距离实现点对，向量 $\boldsymbol{v}^*=q^*-p^*$为 P 和 Q 的一个最短距离实

现向量。对于两个分离的凸多边形，其最短距离实现点对不一定是唯一的，但最短距离实现向量是唯一的。由于 P 和 Q 是凸多边形，可知 p^*和 q^*分别在 P 和 Q 的边界上。

分别用 o_p 和 o_q 表示 P 和 Q 边界上的边或顶点，如果满足 $d(o_p, o_q)=d(P, Q)$，则称$<o_p, o_q>$为 P 和 Q 的一个最短距离站点对。$<o_p, o_q>$有如下几种类型。

（1）<顶点，顶点>

o_p、o_q 是顶点。其只包含一个最短距离实现点对$<p^*, q^*>$，其中 $p^*=o_p$，$q^*=o_q$；

（2）<顶点，边>

o_p 是顶点，o_q 是边。其只包含一个最短距离实现点对$<p^*, q^*>$，其中 $p^*=o_p$，$q^*=p(o_p, o_q)$，$p(o_p, o_q)$为 o_p 在边 o_q 上的投影（垂心）。此时，q^*必在线段 o_q 上，且一定有 $q^* \in \mathrm{VR}(o_p)$。

类似地，可以 o_p 是边，o_q 是顶点。

（3）<边，边>

o_p，o_q 都是边，且 o_p 与 o_q 平行。这时，存在多个最短距离实现点对(但最短距离实现向量是唯一的)。这种情况下，其最短距离实现点必存在于 o_p，o_q 及其顶点形成的<顶点，顶点>或<顶点，边>情况中。因此，我们只考虑对类型（1）和（2）进行处理即可。

通过上述分析知，如果已知 P 和 Q 之间的最短距离站点对，可根据其类型在 $O(1)$时间内计算出 p^*，q^*以及 P，Q 之间的最短距离：$d(P, Q)=d(p^*, q^*)$。

定理 3.8　对于任意两个站点 $o_p \in P$ 和 $o_q \in Q$，p^*和 q^*为表示 o_p 和 o_q 之间最短距离的一对点。o_p 和 o_q 是 P 和 Q 的一个最短距离站点对，当且仅当 $p^* \in \mathrm{VR}(o_q)$且 $q^* \in \mathrm{VR}(o_p)$[Lin1991]。

证明：分别过 p^*和 q^*作线段 p^*q^*的垂线 l_1 和 l_2（如图 3.9、图 3.10 所示）。

易证 p^*和 q^*是 P 和 Q 之间最短距离的一对点的充要条件为 P 在垂线 l_1 的左侧，且 Q 在垂线 l_2 的右侧。而 P 在垂线 l_1 左侧及 Q 在垂线 l_2 右侧的充要条件为线段 p^*q^*属于 VR(oq)及 VR(op)。证毕。

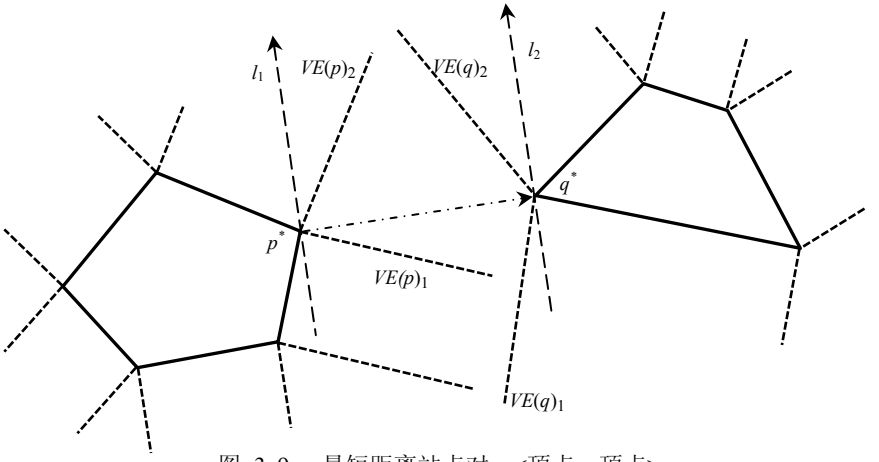


图 3.9　最短距离站点对：<顶点，顶点>

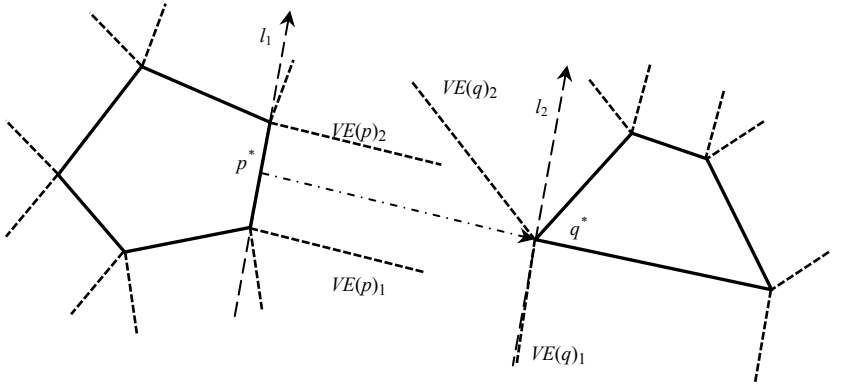


图 3.10　最短距离站点对：<顶点，边>

根据定理 3.8 知，对于分别属于两个凸多边形的任意两个站点，如果表示它们之间的最短距离的一对点中的每个点属于另一个站点的 Voronoi 区域，则这两个站点之间的最短距离就是两个凸多边形之间的最短距离。

为了计算P和Q之间最短距离站点对<o_p，o_q>，我们首先采用二分法查找包含o_p，o_q的初始搜索范围，然后仍采用二分法逐渐缩小搜索范围，直到查找到<o_p，o_q>。我们用C(p_1，p_2)表示多边形边界上从顶点p_1沿逆时针方向到顶点p_2的多边形链。结合图3.11，算法思想简单叙述如下。

（1）首先，在P和Q中确定两个初始搜索范围P'=C(p'，p'')，Q'=(q''，q')，这里要保证P'和Q'包含P和Q的一个最短距离站点对< o_p，o_q >(后面会具体介绍如何保证这一点)。

（2）然后，分别取P'、Q'中间的站点p_a、q_b，它们将P'、Q'分成 P_1''=C(p'，p_a)和 P_2''=C(p_a，p'')， Q_1''=C(q''，q_b)和 Q_2''=C(q_b，q')。

（3）再根据p_a、q_b、VR(p_a)和VR(q_b)的位置关系，确定 P_1'' 、 P_2'' 、 Q_1'' 和 Q_2'' 中有一个或两个肯定不包含P和Q的最短距离站点对。将肯定不包含P和Q的最短距离站点对的多边形链删除，得到新的搜索范围。例如，如果只确定删除 P_1''，则新的搜索范围为 P_2'' 和Q'；如果确定删除 P_1'' 和 Q_1''，则新的搜索范围为 P_2'' 和 Q_2''。

（4）重复上面的搜索过程，直到最终得到最短距离站点对<o_p，o_q>。最后，根据<o_p，o_q>的类型，计算P和Q的距离。

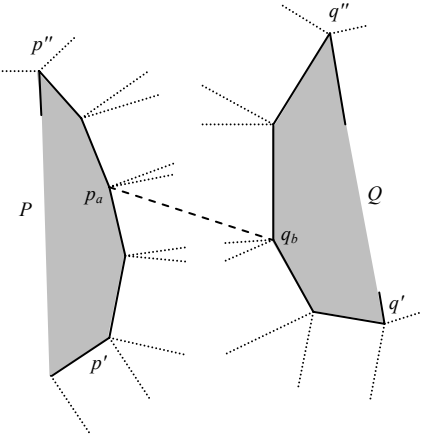


图 3.11　用二分法计算最短距离站点对

算法要在 $O(\log n+\log m)$时间内找到最短距离站点对，须解决以下两个关键问题。

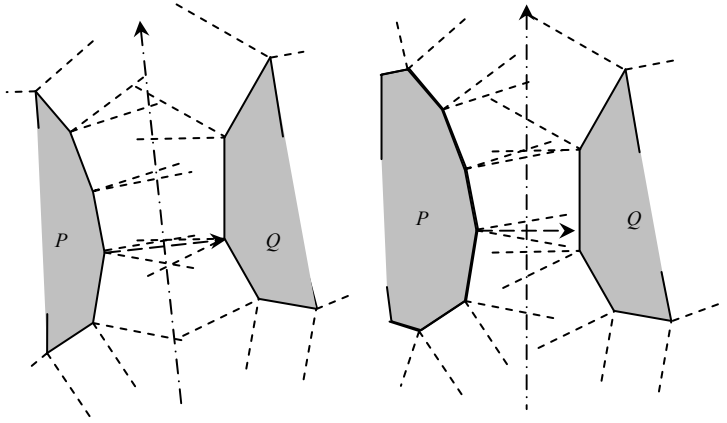
（1）在 $O(\log n+\log m)$时间内计算出初始搜索范围 P'和 Q'。其中 P'和 Q'应包含最短距离站点对<o_p，o_q>，且 P'和 Q'包含的站点越少越好。

（2）在 $O(1)$时间内根据 p_a、q_b、VR(p_a)和 VR(q_b)的位置关系，决定删除 P_1''、P_2''、Q_1''和 Q_2''中的哪一个。

2. 确定初始搜索范围

设 P 的顶点按逆时针方向依次为 p_1，p_2，…，p_n，多边形 Q 的顶点按逆时针依次为 q_1，q_2，…，q_m，其中 n、m 分别为 P 和 Q 的顶点数，如果下标超过 n 或 m，则进行求模运算。

对于一条将 P 和 Q 左、右分开的有向直线，我们称之为 P 和 Q 的一条分离线。对于 P 和 Q 的一个最短距离站点对<o_p，o_q>，p^*q^*表示 o_p，o_q之间距离最短的一对点，则线段 p^*q^*的中垂线必为 P 和 Q 的一条分离线（如图 3.12 所示）。我们称这样的分离线为 P 和 Q 的中分线，其为有向直线，P 和 Q 分别在其左右侧，用 l 表示。这里需要说明的是，本节中中分线 l 只是用于分析两个分离的凸多边形间的位置关系，在算法中并不需要计算。



(a) <顶点，顶点>　　(b) <顶点，边>

图 3.12　P 和 Q 间的中分线

设 p'、p''($p'\neq p''$)为多边形 P 上两个顶点，分别过 p'，p''作中分线 l 的垂线 l_1、l_2。如果 P 完全在 l_1、l_2 之间，且 l_1 与 P 只交于 p'，l_2 与 P 只交于 p''，则 p'、p''将 P 分成两条多边形链 P_1=C(p'，p'')和 P_2=C(p''，p')。如图 3.13 所示，选择 p'、p''可使 P_1 关于中分线 l 是严格单调的，P_2 关于中分线 l 是单调的[O′Rourke1994]。

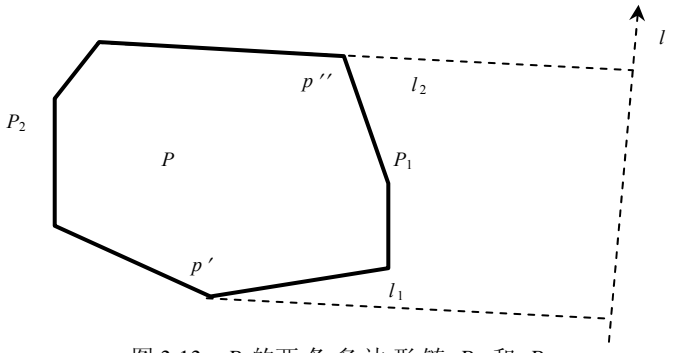


图 3.13　P 的两条多边形链 P_1 和 P_2

对于凸多边形 P 上的任意顶点站点 p_i，用 $\mathrm{VE}(p_i)_1$ 和 $\mathrm{VE}(p_i)_2$ 分别表示 p_i 的两条 Voronoi 边，其中 $\mathrm{VE}(p_i)_1$ 是起点为 p_i 且垂直于 $p_{i-1}p_i$ 的一条射线，$(p_{i-1}p_i\times \mathrm{VE}(p_i)_1)\cdot k<0$；$\mathrm{VE}(p_i)_2$ 是起点为 p_i 且垂直于 p_ip_{i+1} 的一条射线，$(p_ip_{i+1}\times \mathrm{VE}(p_i)_2)\cdot k<0$，其中 k 为 z 轴方向的单位向量。

下面用 l_u，l_{i1}，l_{i2}，l_0，l_1 和 v_u^* 分别表示 l，$\mathrm{VE}(p_i)_1$，$\mathrm{VE}(p_i)_2$，$\mathrm{VE}(p')_2$，$\mathrm{VE}(p'')_1$ 和最短距离实现向量的单位向量。

性质 3.1　对于多边形链 P_1 上的任意顶点站点 p_i，有下列性质：

(1) 如果 p_i=p'，则 $VE(p_i)_1$ 与 l 不相交，$VE(p_i)_2$ 与 l 相交；

(2) 如果 p_i=p″，则 $VE(p_i)_2$ 与 l 不相交，$VE(p_i)_1$ 与 l 相交；

(3) 如果 $p_i\neq p'$ 且 $p_i\neq p''$，则 $VE(p_i)_1$、$VE(p_i)_2$ 都和 l 相交，且 $(\mathrm{v}_u^*\times l_0)\cdot k<(\mathrm{v}_u^*\times l_{i1})\cdot k<(\mathrm{v}_u^*\times l_{i2})\cdot k<(\mathrm{v}_u^*\times l_1)\cdot k$。

性质 3.2　对于多边形链 P_2 上的任意顶点站点 p_i，有下列性质：

(1) 如果 $p_i=p'$，则 $VE(p_i)_1$ 与 l 不相交，$VE(p_i)_2$ 与 l 相交；

(2) 如果 $p_i=p''$，则 $VE(p_i)_2$ 与 l 不相交，$VE(p_i)_1$ 与 l 相交；

(3) 如果 $p_i\neq p'$ 且 $p_i\neq p''$，则 $VE(p_i)_1$、$VE(p_i)_2$ 都不和 l 相交。

因此，可以证明 P、Q 的最短距离站点对$<o_p, o_q>$中的 o_p 在 P_1 中，其将 P_1 分成三部分：上部分 P_u，中间部分 P_m，下部分 P_d。

性质 3.3 对于多边形链 P_1 上的任意顶点站点 p_i，有下列性质：

(1) $\forall p_i\in P_u$，有 $l_{i1}\bullet l_u>l_{i2}\bullet l_u>0$，且$(\mathrm{v}_u^*\times l_{i2})\bullet k>(\mathrm{v}_u^*\times l_{i1})\bullet k>0$;

(2) $\forall p_i\in P_d$，有 $l_{i1}\bullet l_u<l_{i2}\bullet l_u<0$，且$(\mathrm{v}_u^*\times l_{i1})\bullet k<(\mathrm{v}_u^*\times l_{i2})\bullet k<0$;

(3) P_1 的中间部分 P_m 只包含最短距离站点对$<o_p, o_q>$中的站点 o_p，且：

(a) 如果 o_p 是边，则 $l_{i1}\bullet l_u=l_{i2}\bullet l_u=0$，且$(\mathrm{v}_u^*\times l_{i1})\bullet k=(\mathrm{v}_u^*\times l_{i2})\bullet k=0$;

(b) 如果 o_p 是顶点，则 $l_{i1}\bullet l_u<0$，$l_{i2}\bullet l_u>0$，且$(\mathrm{v}_u^*\times l_{i1})\bullet k<0$，$(\mathrm{v}_u^*\times l_{i2})\bullet k>0$。

类似地，凸多边形 Q 上也可分成两条多边形链 Q_1 和 Q_2，且 Q_1 和 Q_2 仍然保持逆时针方向。Q_1 关于中分线 l 是严格单调的，Q_2 关于中分线 l 是严格单调的或单调的。Q_1、Q_2 中任意站点的 Voronoi 边，也分别具有类似上面所介绍的 P_1、P_2 中站点的 Voronoi 边的性质。P、Q 间的最短距离站点对$<o_p, o_q>$中的 o_q 也将 Q_1 分为三部分：上部分 Q_u，中间部分 Q_m，下部分 Q_d。

对于多边形 Q 上的任意顶点站点 q_j，用 $VE(q_j)_1$ 和 $VE(q_j)_2$ 分别表示 q_j 的两条 Voronoi 边，其中 $\mathrm{VE}(q_j)_1$ 是起点为 q_j 且垂直于 $q_{j-1}q_j$ 的一条射线，$(q_{j-1}p_i\times\mathrm{VE}(q_j)_1)\sqcup k<0$；$\mathrm{VE}(q_j)_2$ 是起点为 q_j 且垂直于 q_jq_{j+1} 的一条射线，$(q_jq_{j+1}\times\mathrm{VE}(q_j)_2)\sqcup k<0$。用 l_{j1} 和 l_{j2} 分别表示 $\mathrm{VE}(q_j)_1$ 和 $\mathrm{VE}(q_j)_2$ 方向的单位向量。

性质 3.4 对于多边形链 Q_1 上的任意顶点站点 q_j，有下列性质：

（1）$\forall q_j\in Q_u$，有 $l_{j2}\sqcup l_u>l_{j1}\sqcup l_u>0$，且$(\mathrm{v}_u^*\times l_{j1})\bullet k>(\mathrm{v}_u^*\times l_{j2})\bullet k>0$;

（2）$\forall q_j \in Q_d$，有 $l_{j2} \bullet l_u < l_{j1} \bullet l_u < 0$，且 $(\mathrm{v}_u^* \times l_{j2}) \bullet k < (\mathrm{v}_u^* \times l_{j1}) \bullet k < 0$；

（3）Q_1 的中间部分 Q_m 只包含最短距离站点对 $<o_p, o_q>$ 中的站点 q，且：

（a）如果 o_q 是边，则 $l_{j1} \bullet l_u = l_{j2} \bullet l_u = 0$，且 $(\mathrm{v}_u^* \times l_{j1}) \bullet k = (\mathrm{v}_u^* \times l_{j2}) \bullet k = 0$；

（b）如果 o_q 是顶点，则 $l_{j2} \bullet l_u < 0$，$l_{j1} \bullet l_u > 0$，且 $(\mathrm{v}_u^* \times l_{j2}) \bullet k < 0$，$(\mathrm{v}_u^* \times l_{j1}) \bullet k > 0$。

显然，P 和 Q 的最短距离站点对 o_p，o_q 分别位于 P_1，Q_1 中。我们将 P_1，Q_1 中的站点称为候选站点。后面的算法 3.3 可计算出 P 和 Q 的初始搜索范围 P' 和 Q'，其中 P' 和 Q' 分别只是 P_1，Q_1 的一部分，但包含有最短距离站点对。在介绍该算法前，先介绍几个定理。

引理 3.6　对于任意一点 $u \in P$，存在站点 $v \in Q_1$，满足 $u \in \mathrm{VR}(v)$。

引理 3.7　对于任意一点 $u \notin P$，站点 $p' \in P$，$p'' \in P$，$p' \neq p''$，$u \notin \mathrm{VR}(p')$ 且 $u \notin \mathrm{VR}(p'')$，以及任意顶点 $v \in \mathrm{C}(p', p'')$，$v \neq p'$ 且 $v \neq p''$。如果 $uv \cap \mathrm{VE}(p')_2 = \varnothing$，且 $uv \cap p'p'' = \varnothing$，且 $uv \cap \mathrm{VE}(p'')_1 = \varnothing$，则存在一个站点 $o \in \mathrm{C}(p', p'')$，满足 $u \in \mathrm{VR}(o)$（如图 3.14 所示）。

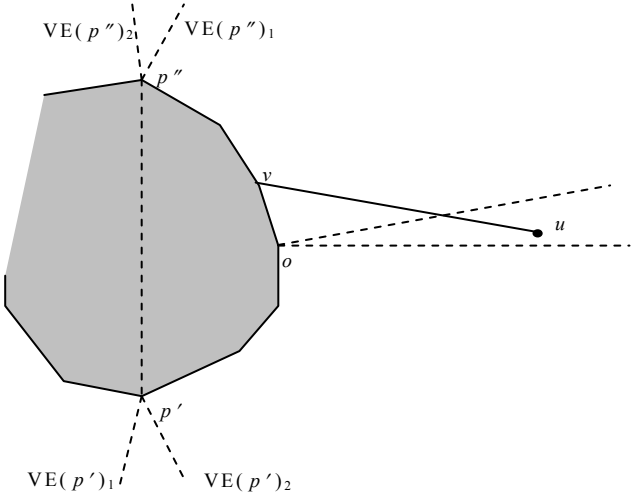


图 3.14　P 外一点 u 与多边形链 C(p'，p'')的关系

对于任意一点 $u\notin P$，我们首先取 $p'=p_0$，$p''=p_{n/2}$，然后根据引理 3.7 判断 u 是在 C(p'，p'')中还是在 C(p''，p')中，然后继续在 C(p'，p'')或 C(p''，p')中用二分法查找，最后可查找出 P 中离 u 最近的站点。由引理 3.7 知，对于任意一点 $u\notin P$ 和站点 $p'\in P$，$p''\in P$，可以在 $O(1)$时间内判断出是否存在站点 $o\in$C(p'，p'')，满足 $u\in$VR(o)。因此，对于任意一点 $u\notin P$，可在 $O(\log n)$时间内找到站点 $o\in P$，$u\in$VR(o)。因此，根据引理 3.6，对于任意一点 $u\in P$，可在 $O(\log n)$时间内找到站点 $v\in Q_1$，满足 $u\in$VR(v)。

引理 3.8 给定一个顶点 $o\in Q_d$ 和任意顶点 $o'\in Q$，$o'\neq o$。有 $o'\in Q_u$，或 $o'\in Q_m$，如果其同时满足下列四个条件：

（1） $(\mathrm{VE}(o)_1\times\mathrm{VE}(o')_1)\bullet k<0$；

（2） $(\mathrm{VE}(o)_1\times\mathrm{VE}(o')_2)\bullet k<0$；

（3） $\mathrm{VE}(o)_1\bullet\mathrm{VE}(o')_1>0$；

（4） $\mathrm{VE}(o)_1\bullet\mathrm{VE}(o')_2<0$。

证明：假设 o'同时满足四个条件，但 $o'\in Q_2$。由于 $o'\in Q_2$，存在 $(\mathrm{VE}(o)_1\times\mathrm{VE}(o')_1)\bullet k>0$ 且 $(\mathrm{VE}(o)_1\times\mathrm{VE}(o')_2)\bullet k>0$，或 $\mathrm{VE}(o)_1\bullet\mathrm{VE}(o')_1<0$ 且 $\mathrm{VE}(o)_1\bullet\mathrm{VE}(o')_2<0$。矛盾，假设不成立。

假设 o'同时满足四个条件，但 $o'\in Q_d$。由于 $o'\in Q_d$，所以 $\mathrm{VE}(o)_1\bullet\mathrm{VE}(o')_1>0$ 且 $\mathrm{VE}(o)_1\bullet\mathrm{VE}(o')_2>0$。矛盾，假设不成立。

因此，原命题成立。证毕。

类似地，可得到引理 3.9。

引理 3.9 给定一个顶点 $o\in Q_u$ 和任意顶点 $o'\in Q$，$o'\neq o$。有 $o'\in Q_d$ 或 $o'\in Q_m$，如果其同时满足下列四个条件：

（1） $(\mathrm{VE}(o)_2\times\mathrm{VE}(o')_1)\bullet k>0$；

（2） $(\mathrm{VE}(o)_2\times\mathrm{VE}(o')_2)\bullet k>0$；

（3）$VE(o)_2 \bullet VE(o')_1 > 0$；

（4）$VE(o)_2 \bullet VE(o')_2 < 0$。

定理 3.9　设 p_i、q_j 分别是 P_1、Q_1 的顶点，$p_i \notin VR(q_j)$ 且 $q_j \in VR(p_i)$，则有：

（1）如果 p_i 在 $VE(q_j)_2$ 的左侧，则 $p_i \notin P_d$，$q_j \notin Q_d$；

（2）如果 p_i 在 $VE(q_j)_1$ 的右侧，则 $p_i \notin p_u$，$q_j \notin Q_u$。

证明（反证法）：

（1）p_i 在 $VE(q_j)2$ 的左侧。

（a）假设 $p_i \in P_d$。由于 $q_j \in VR(p_i)$，所以有 $(\mathrm{v}_u^* \times l_{i1}) \bullet k < (\mathrm{v}_u^* \times p_i q_j) \bullet k < (\mathrm{v}_u^* \times l_{i2}) \bullet k < 0$，即 $q_j \in Q_d$。过 q_j 作 p^*q^* 的平行线 l'，则 p_i 位于 l' 的左侧。由于 $q_j \in Q_d$，所以 $VR(q_j)_1$ 和 $VE(q_j)_2$ 在 l' 的右侧，因而 p_i 在 $VE(q_j)_2$ 的右侧，矛盾。故 $p_i \notin P_d$。

（b）假设 $q_j \in Q_d$。由（a）知 $p_i \notin P_d$，即 $p_i \in P_m$ 或 $p_i \in P_u$。由于 $q_j \in VR(p_i)$，且 $q_j \in Q_d$，所以有 $(\mathrm{v}_u^* \times l_{i1}) \bullet k < (\mathrm{v}_u^* \times p_i q_j) \bullet k < 0$。过 q_j 作 p^*q^* 的平行线 l'，则 p_i 位于 l' 的左侧。由于 $q_j \in Q_d$，所以 $VE(q_j)_1$ 和 $VE(q_j)_2$ 在 l' 的右侧，因而 p_i 在 $VE(q_j)_2$ 的右侧，矛盾。故 $q_j \notin Q_d$。

（2）证明方法类似（1）。证毕。

如果 p_i 是 P_1 的一个边，则根据顶点 q_j 在 p_i 边上的投影 $p\ (q_j, p_i)$ 与 $VE\ (q_j)_2$ 和 $VE\ (q_j)_1$ 的位置关系确定 p_i 与 q_j 的位置。

定理 3.10　设站点 $o_1 \in P$，站点 $o_2 \in Q$，有 $o_2 \in VR(o_1)$。

（1）如果 $o_1 \in P_d$，则 $o_2 \in Q_d$；

（2）如果 $o_1 \in P_u$，则 $o_2 \in Q_u$；

（3）如果 $o_2 \in Q_d$，则 $o_1 \in P_d$ 或 P_m；

（4） 如果 $o_2 \in Q_u$，则 $o_1 \in P_u$ 或 P_m。

证明：（1）假设 $o_1 \in P_d$，$o_2 \notin Q_d$。令 o_1*，o_2*表示 o_1，o_2 的最短距离点对，$o^*=o_2^*-o_1^*$为向量，则$(v^* \times o^*) \bullet k>0$。根据性质 3.3，$o_2$ 在 $\mathrm{VE}(o_1)_2$ 的左侧，$o_2 \notin \mathrm{VR}(o_1)$。矛盾。所以，如果 $o_1 \in P_d$，则 $o_2 \in Q_d$。

类似的方法可证明（2）、（3）、（4）也成立。证毕。

算法 3.3 计算 P 和 Q 的初始搜索范围 P'和 Q'

（1）取 P 的第一个顶点 p_1，根据引理 3.7，用二分法在 Q 中计算站点 o_1，其中 $p_1 \in \mathrm{VR}(o_1)$；

（1.1）如果 o_1 是边，取 q_{i2} 为 o_1 的一个端点；如果 o_1 是顶点，$q_{i2}=o_1$；

（1.2）根据引理 3.6 知，$q_{i2} \in Q_1$。

（2）类似地，用二分法在 P 中计算站点 o_2，其中 $q_{i2} \in \mathrm{VR}(o_2)$；

如果 o_2 是顶点且 $o_2 \in \mathrm{VR}(q_{i2})$，或 o_2 是边且 $p(q_{i2}, o_2) \in \mathrm{VR}(q_{i2})$，则 o_2、q_{i2} 为最短距离站点对，整个算法结束，否则转（3）。

（3）根据定理 3.9，确定 q_{i2} 的位置：

（3.1）如果 o_2 在 $\mathrm{VE}(q_{i2})_2$ 的左侧，则 $o_2 \notin P_d$，$q_{i2} \notin Q_d$；

（3.2）如果 o_2 在 $\mathrm{VE}(q_{i2})_1$ 的右侧，则 $o_2 \notin P_u$，$q_{i2} \notin Q_u$。

（4）根据引理 3.8 和引理 3.9，用二分法在 Q 中计算 q_{j2}：

（4.1）如果 $q_{i2} \notin Q_d$：

（4.1.1）如果 $q_{(i2+2^k)} \in Q_d$或 Q_m，则 $q_{j2}=q_{(i2+2^k)}$（k 为整数，初值为 0）；

（4.1.2）如果 $q_{(i2+2^k)} \in Q_u$，则继续判断 $q_{(i2+2^{k+1})}$的位置；

（4.1.3）否则，在 $q_{(i2+2^k)}$与 $q_{(i2+2^{k+1})}$之间，用类似的方法（索引每次增加 2、4、8…）进行查找，可得到 $q_{j2} \in Q_d$ 或 Q_m。

（4.2）如果 $q_{i2}\notin Q_u$，可沿顺时针方向用上面类似的方法确定 $q_{j2}\in Q_u$ 或 Q_m。

（5）用二分法在 P 中计算 o_4，其中 $q_{j2}\in \mathrm{VR}(o_4)$；

如果 o_4 是顶点且 $o_4\in \mathrm{VR}(q_{j2})$，或 o_4 是边且 $p(q_{j2}, o_4)\in \mathrm{VR}(q_{j2})$，则 o_4、q_{j2} 为最短距离特征对，整个算法结束。

（6）根据定理 3.10 和 q_{i2}、q_{j2} 的位置，确定 p_{i1}、p_{j1} 分别取 o_2、o_4 的哪个端点，并确定 p_{i1}，p_{i2} 的位置。

p_{i1}、p_{j1} 间的站点组成 P'（逆时针），q_{i2}、q_{j2} 间的站点组成 Q'（逆时针）。根据引理 3.8、引理 3.9 以及定理 3.10，P'和 Q'即为包含有最短距离站点对的初始搜索范围。这里，P'和 Q'分别只是 P_1、Q_1 的一部分。

由前面的分析知，算法 3.3 计算 q_{i2} 的时间复杂度为 $O(\log m)$，计算 p_{i1} 的时间复杂度为 $O(\log n)$；根据引理 3.9，计算 q_{j2} 的时间复杂度为 $O(\log m)$，计算 p_{j1} 的时间复杂度为 $O(\log n)$。因此，算法 3.3 中确定初始搜索范围的时间复杂度为 $O(\log n+\log m)$。

3. 计算候选站点的位置

下面讨论关键问题 2 的解决方法。

设两个顶点 $p_i\in P_1$，$q_j\in Q_1$。p_i、q_j 关于 $\mathrm{VR}(p_i)$、$\mathrm{VR}(q_j)$的位置关系可以分为以下几种情况。

情况 1：q_j 在 $\mathrm{VE}(p_i)_1$ 的右侧，又分为以下三种子情况：

情况 1.1：p_i 在 $\mathrm{VE}(q_j)_2$ 的左侧；

情况 1.2：$p_i\in \mathrm{VR}(q_j)$，即 p_i 在 $\mathrm{VE}(q_j)_1$ 的左侧，且在 $\mathrm{VE}(q_j)_2$ 的右侧；

情况 1.3：p_i 在 $\mathrm{VE}(q_j)_1$ 的右侧。

情况 2：$q_j\in \mathrm{VR}(p_i)$，即 q_j 在 $\mathrm{VE}(p_i)_1$ 的左侧且在 $\mathrm{VE}(p_i)_2$ 的右侧，又分为以下三种子情况：

情况 2.1：p_i 在 $\mathrm{VE}(q_j)_2$ 的左侧；

情况 2.2：$p_i \in \mathrm{VR}(q_j)$，即 p_i 在 $\mathrm{VE}(q_j)_1$ 的左侧，且在 $\mathrm{VE}(q_j)_2$ 的右侧；

情况 2.3：p_i 在 $\mathrm{VE}(q_j)_2$ 的右侧。

情况 3：q_j 在 $\mathrm{VE}(p_i)_2$ 的左侧，又分为以下三种子情况：

情况 3.1：p_i 在 $\mathrm{VE}(q_j)_1$ 的右侧；

情况 3.2：$p_i \in \mathrm{VR}(q_j)$，即 p_i 在 $\mathrm{VE}(q_j)_1$ 的左侧，且在 $\mathrm{VE}(q_i)_2$ 的右侧；

情况 3.3：p_i 在 $\mathrm{VE}(q_j)_2$ 的左侧。

如图 3.15 所示，q_7 在 $\mathrm{VE}(p_1)_1$ 的右侧，q_3、q_4、q_5、q_6、q_7 在 $\mathrm{VE}(p_2)_1$ 的右侧，q_1、q_2、q_3、q_4、q_5 在 $\mathrm{VE}(p_1)_2$ 的左侧，q_1 在 $\mathrm{VE}(p_2)_2$ 的左侧，q_6 在 $\mathrm{VR}(p_1)$ 中，q_2 在 $\mathrm{VR}(p_2)$中，p_2 在 $\mathrm{VE}(q_1)_2$、$\mathrm{VE}(q_2)_2$、$\mathrm{VE}(q_3)_2$ 的左侧，p_2 在 $\mathrm{VR}(q_4)$ 中，p_1、p_2 在 $\mathrm{VE}(q_5)_1$、$\mathrm{VE}(q_6)_1$、$\mathrm{VE}(q_7)_1$ 的右侧，p_1 在 $\mathrm{VE}(q_1)_2$、$\mathrm{VE}(q_2)_2$、$\mathrm{VE}(q_3)_2$、$\mathrm{VE}(q_4)_2$ 的左侧，$<p^*, q^*>$ 为 P 和 Q 的一个最短距离实现点对。

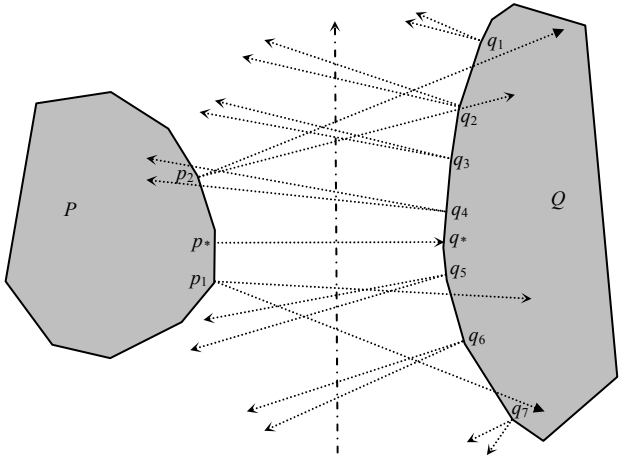


图 3.15　候选站点间的位置关系

定理 3.11　设 q_j 在 $\mathrm{VE}(p_i)_1$ 的右侧，p_i 在 $\mathrm{VE}(q_j)_1$ 的右侧，则有：

（1）如果$(\mathrm{VE}(p_i)_1 \times \mathrm{VE}(q_j)_1) \bullet k \geqslant 0$，则 $p_i \in P_u$；

（2）如果$(\mathrm{VE}(p_i)_1\times\mathrm{VE}(q_j)_1)\bullet k<0$，则 $q_j\in Q_d$。

证明：（1）假设$(\mathrm{VE}(p_i)_1\times\mathrm{VE}(q_j)_1)\bullet k\geqslant 0$，但 $p_i\in P_d$。由于 q_j 在 $\mathrm{VE}(p_i)_1$ 的右侧，p_i 在 $\mathrm{VE}(q_j)_1$ 的右侧，所以 $q_j\in Q_d$。根据性质 3.4，有$(\mathrm{VE}(p_i)_1\times\mathrm{VE}(q_j)_1)\bullet k<0$。矛盾。所以原命题成立。类似地，可证明（2）。证毕。

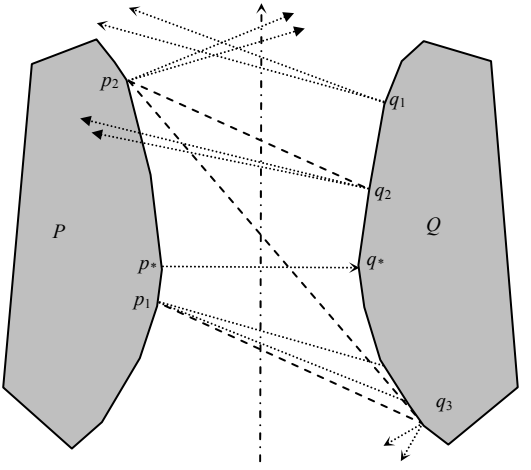


图 3.16　情况 1.3 中候选站点间的位置关系

对于情况 1.3（q_j 在 $\mathrm{VE}(p_i)_1$ 的右侧，p_i 在 $\mathrm{VE}(q_j)_1$ 的右侧）：当 $p_i\in P_u$ 时，$q_j\in Q_u, Q_m$ 或 Q_d（如图 3.16 中的 p_2 与 q_2，p_2 与 q_3，其中 $p_2\in P_u, q_2\in Q_u, q_3\in Q_d$）。当 $p_i\in P_m$ 或 P_d 时，$q_j\in Q_d$（如图 3.16 中的 p_1 与 q_3，其中 $p_1\in P_d$，$q_3\in Q_d$）。如果$(\mathrm{VE}(p_2)_1\times\mathrm{VE}(q_2)_1)\bullet k\geqslant 0$，则可排除 $p_i\in P_m$ 且 $q_j\in Q_d$，$p_i\in P_d$ 且 $q_j\in Q_d$ 的情况（如图 3.16 中的 p_1 与 q_3），因此必有 $p_i\in P_u$（如图 3.16 中的 p_2 与 q_2，p_2 与 q_3）。如果$(\mathrm{VE}(p_i)_1\times\mathrm{VE}(q_j)_1)\bullet k<0$，则可排除 $q_j\in Q_u$ 和 $q_j\in Q_m$ 的情况，因此必有 $q_j\in Q_d$（如图 3.16 中的 p_1 与 q_3）。

定理 3.12　设 q_j 在 $\mathrm{VE}(p_i)_2$ 的左侧，p_i 在 $\mathrm{VE}(q_j)_2$ 的左侧，则有：

（1）如果$(\mathrm{VE}(q_j)_2\times\mathrm{VE}(p_i)_2)\bullet k\leqslant 0$，则 $q_j\in Q_u$；

（2）如果$(\mathrm{VE}(q_j)2\times\mathrm{VE}(p_i)_2)\bullet k>0$，则 $p_i\in P_d$。

证明方法类似定理 3.11。

对于情况 3.3（q_j 在 VE(p_i)$_2$ 的左侧，p_i 在 VE(q_j)$_2$ 的左侧）：当 $p_i \in P_u$ 或 P_m 时，$q_j \in Q_u$（如图 3.17 中的 p_2 与 q_1，其中 $p_2 \in P_u$，$q_1 \in Q_u$）。当 $p_i \in P_d$ 时，$q_j \in Q_u$，Q_m 或 Q_d（如图 3.17 中的 p_1 与 q_1，p_1 与 q_2，其中 $p_1 \in P_d$，$q_1 \in Q_u$，$q_2 \in Q_d$）。如果(VE(q_j)$_2$×VE(p_i)$_2$)•$k \leqslant 0$，可排除 $q_j \in Q_u$ 和 $q_j \in Q_m$，必有 $q_j \in Q_u$。类似地，如果(VE(q_j)$_2$×VE(p_i)$_2$)•$k > 0$，可排除 $p_i \in P_u$ 和 $p_i \in P_m$，则 $p_i \in P_d$。

针对上面的每一种情况，我们给出判断 p_i，q_j 关于有向线段 p^*q^* 的位置的算法。

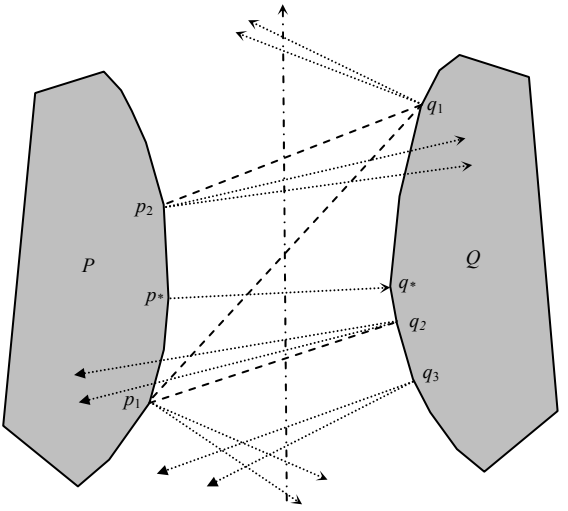


图 3.17　情况 3.3 中候选站点间的位置关系

算法 3.4　判断 p_i、q_j 关于有向线段 p^*q^* 的位置

（1）如果满足情况 1：

（1.1）如果满足情况 1.1，则 $p_i \in P_u$，$q_j \in Q_u$；

（1.2）如果满足情况 1.2，则 $p_i \in P_u$，$q_j \in Q_u$；

（1.3）如果满足情况 1.3，根据定理 3.11，有：

（a）如果$(\mathrm{VE}(p_i)_1\times\mathrm{VE}(q_j)_1)\bullet k\geqslant 0$，则 $p_i\in P_u$；

（b）如果$(\mathrm{VE}(p_i)_1\times\mathrm{VE}(q_j)_1)\bullet k<0$，则 $q_j\in Q_d$。

（2）如果满足情况 2：

（2.1）如果满足情况 2.1，则 $p_i\in P_u$ 或 $p_i\in P_m$，$q_j\in Q_u$；

（2.2）如果满足情况 2.2，则 $p_i\in P_m$，$q_j\in Q_m$；

（2.3）如果满足情况 2.3，则 $p_i\in P_d$ 或 $p_i\in P_m$，$q_j\in Q_d$。

（3）如果满足情况 3：

（3.1）如果满足情况 3.1，则 $p_i\in P_d$，$q_j\in Q_d$；

（3.2）如果满足情况 3.2，则 $p_i\in P_d$，$q_j\in Q_d$；

（3.3）如果满足情况 3.3，根据定理 3.12，有：

（a）如果$(\mathrm{VE}(q_j)_2\times\mathrm{VE}(p_i)_2)\bullet k>0$，则 $p_i\in P_d$。

（b）如果$(\mathrm{VE}(q_j)_2\times\mathrm{VE}(p_i)_2)\bullet k\leqslant 0$，则 $q_j\in Q_u$。

根据算法 3.4，可以确定 p_i 或 q_j 关于 p^*q^*的位置，然后可以确定新的搜索范围。算法 3.4 判断 $p_i\in P_1$ 和 $q_j\in Q_1$ 位置的时间复杂度为 $O(1)$。

4. 计算最短距离站点对

为便于描述，设初始搜索范围 P'的起点 $p'\in P_d$，终点 $p''\in P_u$；Q'的终点 $q'\in Q_d$，起点 $q''\in Q_u$。

算法 3.5　查找最短距离站点对

（1）如果 p'与 p''为相邻的两个顶点，则可用二分法在 Q'中计算 q。

（1.1）如果 q 是顶点，满足 $o^*\in\mathrm{VR}(q)$且 $q\in\mathrm{VR}(o)$，其中 o 为站点 p'、p''或 $p'p''$，o^*为 q 到 p'、p''或 $p'p''$的投影；则算法结束。

（1.2）如果 q 是边，满足 $p'\in$VR(q)且 $q^*\in$VR(p')，或 $p''\in$VR(q)且 $q^*\in$VR(p'')，其中 q^*为 p'或 p''到 q 的投影；则算法结束。

（2）如果 q'与 q''为相邻的两个顶点，则可用二分法在 P'中计算 p。

（2.1）如果 p 是顶点，满足 $o^*\in$VR(p)且 $p\in$VR(o)，其中 o 为站点 q'、q''或 $q'q''$，o^*为 p 到 q'、q''或 $q'q''$的投影；则算法结束。

（2.2）如果 p 是边，满足 $q'\in$VR(p)且 $p^*\in$VR(q')，或 $q''\in$VR(p)且 $p^*\in$VR(q'')，其中 p^*为 q'或 q''到 p 的投影；则算法结束。

（3）分别取 P'、Q'中间的顶点站点 p_i、q_j，它们将 P'，Q'分成子多边形链 P_1''和 P_2''、Q_1''和 Q_2''。

（4）用算法 3.4 判断 p_i和 q_j位置。

（5）确定新的搜索范围。

（5.1）如果 $p_i\in P_m$且 $q_j\in Q_m$，则<p_i，q_j>为最短距离站点对；算法结束；

（5.2）如果 $p_i\in P_m$且 $q_j\notin Q_m$，则找到最短距离站点对中的一个 $p=p_i$，然后在 Q'中用二分法计算 q，其中 $p\in$VR(q)且 $q^*\in$VR(p)，算法结束；

（5.3）如果 $q_j\in Q_m$且 $p_i\notin P_m$，则找到最短距离站点对中的一个 $q=q_j$，然后在 P'中用二分法计算 p，其中 $p^*\in$VR(q)且 $q\in$VR(p)，算法结束；

（5.4）如果 $p_i\in P_u$，则 $P'=P_1''$，转（1）；

（5.5）如果 $p_i\in P_d$，则 $P'=P_2''$，转（1）；

（5.6）如果 $q_j\in Q_u$，则 $Q'=Q_1''$，转（1）；

（5.7）如果 $q_j\in Q_d$，则 $Q'=Q_2''$，转（1）。

算法 3.5 的时间复杂度为 $O(\log n+\log m)$。

算法 3.6　计算两个分离凸多边形间的距离

（1）用算法 3.3 计算 P 和 Q 的初始搜索范围 P'和 Q'；

（2）用算法 3.5 在 P'和 Q'中查找一个最短距离站点对<o_p，o_q>；

（3）根据<o_p，o_q>的类型计算 P 和 Q 的距离：$d(P, Q)=d(o_p, o_q)$。

由推论 3.4 可知，对于凸多边形的每个站点的外部 Voronoi 区域只有 2 条 Voronoi 边，且每条 Voronoi 边都和多边形的边垂直。因此，计算一个站点的 Voronoi 边的时间为 $O(1)$，这种计算只在计算最短距离站点对的过程中进行，不需要预先计算与存储。由上面对算法 3.3 和算法 3.5 的分析知，计算初始搜索范围及在其中查找一个最短距离站点对的时间复杂度都为 $O(\log n+\log m)$。另外，根据最短距离站点对计算多边形的距离的时间为 $O(1)$。因此，整个算法的时间复杂度为 $O(\log n+\log m)$。

3.3.3　简单多边形中的最短路径计算

本节介绍一种基于 Voronoi 图的计算简单多边形中任意两点 s 和 t 之间的最短路径 SP(s，t)的算法[YangCL2007]。

1. 概念和性质

在介绍具体算法之前，先介绍几个基本概念。

简单多边形 P 中两点 s 和 t 之间的 Voronoi 骨架路径 VSP(s，t)是一条由系列 Voronoi 边首尾依次连接而成的路径。如图 3.18 所示，其由三个部分组成，分别是中间骨架部分 $s_1v_1r_1...r_kv_2t_2$，起始部分 ss_1 以及结束部分 t_2t。其中，由 s 点作相应多边形边（s 所在 Voronoi 区域所对应的边）的垂线（如果 s 在内尖点 p 的 Voronoi 区域内，则直接连接 sp），交 Voronoi 边于 s_1，从而得到路径的起始部分 ss_1；由 t 点作相应多边形边的垂线（如果 t 在内尖点 q 的 Voronoi 区域内，则直接连接 qt），交 Voronoi 边于 t_2，得到路径的结束部分 t_2t。又如图 3.19 中粗虚线所示，VSP(v，p_9)={v，a，…，o，p_9}。

若 Voronoi 边 e 为两 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 Voronoi 区域，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。在图 3.19 中，边 $p_1\,p_2$ 为有向边 de 的左侧关联边，凹顶点(特殊边)p_{18} 为有向边 de 的右侧关联边。需要注意的是，$p_1\,p_2$ 为有向边 ed 的右侧关联边，p_{18} 为有向边 ed 的左侧关联边。

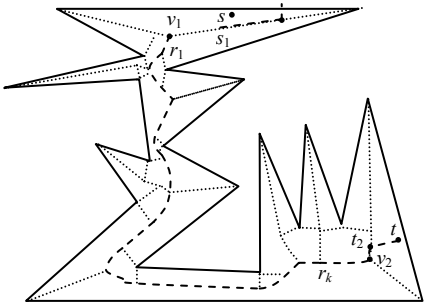


图 3.18　Voronoi 骨架路径

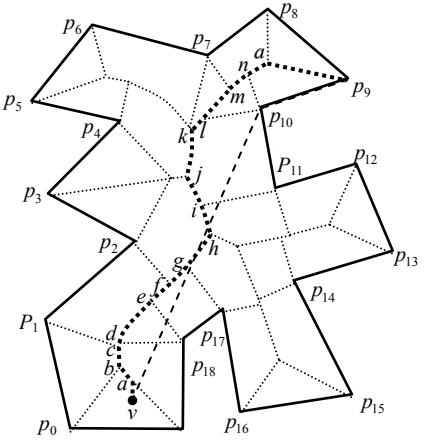


图 3.19　简单多边形中的 Voronoi 骨架路径(粗虚线)

如果一个内尖点 r 的 Voronoi 区域与一条 Voronoi 骨架路径有公共部分（Voronoi 边），则称 r 为该路径的关联内尖点。当我们沿着 Voronoi 骨架路径走过时，发现所有的关联内尖点会依次路过，那些位于路径左侧的关联内尖点称为左关联内尖点，相应地也可以定义右关联内尖点。

设 s、t 是多边形 P 内的两点，p_0，p_1，…，p_n(p_0=s，p_n=t)是最短路径 SP(s，t)上的所有点。可以看出最短路径是由左关联内尖点和右关联内尖点组成的。我们得到以下引理。

引理 3.10　SP(s，t)上的任意顶点 p_i(0<i<n)一定是路径 VSP(s，t)上的关联内尖点。

证明：图 3.20 中 *sABCDt* 是最短路径，存在且只存在一条 Voronoi 骨架路径连接 *s* 与 *t*。我们证明最短路径上的任意一点 *B* 都是该 Voronoi 骨架路径的一个关联内尖点。

假定 *BF* 是∠*CBA* 的平分线，*F* 是 *BF* 与 Voronoi 骨架路径 *EFGH* 的交点。从点 *F* 看，*KLBCMN* 的左侧可见部分在点画线 *ABCD* 之后或者上面。因为 *ABCD* 是一个凸包链，而点 *F* 到 *KLBCMN* 的最近点是 *B*，所以 *B* 点一定是一个关联内尖点。证毕。

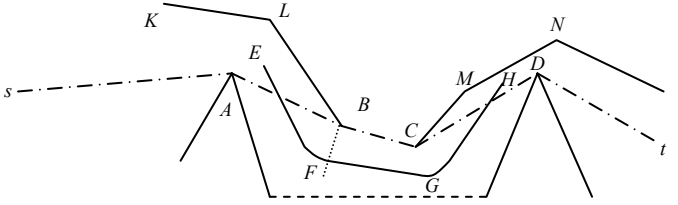


图 3.20　*s*、*t* 之间的最短路径（点画线）

设 q_1，q_2，…，q_m 是 VSP(*s*，*t*)的关联内尖点，是沿 VSP(*s*，*t*)访问依次得到的，而最短路径 SP(*s*，*t*)是由其中的点 q_{k1}，q_{k2}，…，q_{kn} 依次组成。由引理 3.10 知，q_{k1}，q_{k2}，…，q_{kn} 是 q_1，q_2，…，q_m 的子集。

引理 3.11　点 q_{k1}，q_{k2}，…，q_{kn} 在有序序列 q_1，q_2，…，q_m 中的顺序与在 SP(*s*，*t*)中的顺序一致。

证明：如果最短路径上的点都是左关联内尖点或者都是右关联内尖点，则证明显然成立。所以只需要证明最短路径上的点同时来自左、右内尖点时的情况，如图 3.21 中的点 *A*、*B*。

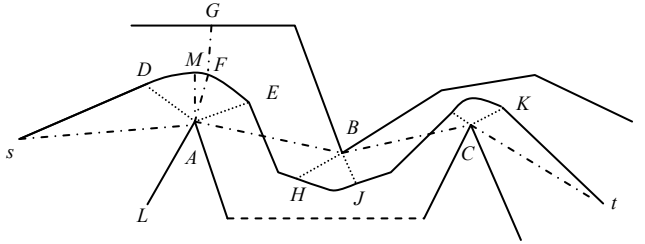


图 3.21　点画线是最短路径，细线是 Voronoi 边

在图 3.21 中，*sABCt* 是最短路径，细线 *sDFEHJKt* 是 Voronoi 骨架路径。点 *F* 是骨架上的一点，且 *AF*⊥*AB*。*G* 与 *A* 是距离 *F* 最近的两个点。显然，Voronoi 骨架路径 *sDFEHJKt* 仅能通过 *sDF* 一次。从点 *B* 看骨架 *sDF* 上的点一定经过 *AFG*。点 *B* 的 Voronoi 区域的边界不可能存在于 *sDF* 上，因为 *sDF* 上的点与 *B* 的距离均大于与 *A* 的距离。*M* 是 Voronoi 骨架路径上的点，且 *AM*⊥*As*，因为 *s* 从 *B* 点不可见，所以 *M* 在 *AF* 的左侧。区域 *AMF* 一定是点 *A* 的 Voronoi 区域的一部分。因此在 *sDF* 骨架上一定存在 *A* 的 Voronoi 区域的边界，而 *B* 的 Voronoi 区域的边界只能存在于 *FEHJKt* 上。这就证明了 Voronoi 骨架上的内尖点顺序与最短路径上的内尖点顺序一致。证毕。

通过引理 3.10 与引理 3.11 可以得到定理 3.13。

定理 3.13 SP(*s*，*t*)上的顶点可以沿着 VSP(*s*，*t*)顺序寻找，且一定是 VSP(*s*，*t*)的关联内尖点集合的子集。

由此可以判断，如果 VSP(*s*，*t*)中不包含内尖点，则 *s* 与 *t* 之间是可见的，SP(*s*，*t*)就是线段 *st*。否则，按照定理 3.13 设计算法来计算 SP(*s*，*t*)。

2. 算法思想与步骤

由于在简单多边形中 *s*、*t* 之间只存在一条 Voronoi 骨架路径 VSP(*s*，*t*)，SP(*s*，*t*)上的点能够通过顺序遍历 VSP(*s*，*t*)关联的多边形顶点求得。计算 SP(*s*，*t*)的算法有以下三个步骤。

（1）首先确定 *s*、*t* 所在的 Voronoi 区域；

（2）然后确定 VSP(*s*，*t*)；

（3）最后顺序遍历 VSP(*s*，*t*)，确定 SP(*s*，*t*)。

算法的第（1）步可以通过计算 *s* 或 *t* 到多边形的每个站点的距离，然后找出最短距离，即可确定 *s* 或 *t* 所在的 Voronoi 区域。

算法的第（2）步可以先通过 *s* 作相应多边形边的垂线（如果 *s* 在内尖点 *p* 的 Voronoi 区域内，则直接连接 *sp*），得到路径的起始部分 ss_1；同样计算出路径的结束部分 t_2t。然后通过遍历多边形的 Voronoi 图（是一棵树）得到

s_1 和 t_2 之间的 Voronoi 骨架路径，将这三部分连接起来即可得到 VSP(s，t)。

这里重点介绍如何实现第（3）步，即如何沿 VSP(s，t)计算 SP(s，t)。其计算方法是：首先，算法从 s 往 t 遍历 VSP(s，t)时，依次找出关联的左右内尖点。然后，计算已找到的左、右内尖点的凸包链 p_0，q_{l1}，q_{l2}，…及 p_0，q_{r1}，q_{r2}，…（见图 3.22），同时维护从当前点 p_0 出发的左、右切线（例如图 3.22 中的 $p_0\,q_{l2}$ 和 $p_0\,q_{r1}$）。接下来，检查右凸包链的切线 $p_0\,q_{r1}$ 是否与左侧的关联边相交，左凸包链的切线 $p_0\,q_{l2}$ 是否与右侧的关联边相交，沿左、右内尖点的交替顺序检查。如果切线 $p_0\,q_{l2}$ 与边 $v_{r4}\,q_{r4}$ 相交，则 q_{l2} 一定是 SP(s，t)上的一点。将当前点移动到 q_{l2}，继续检查工作。由于左侧切线与右关联边 $v_{r4}\,q_{r4}$ 也相交，则点 q_{l3} 也一定是 SP(s，t)上的点。将当前点移动到 q_{l3}，继续如上的计算。

我们注意到，在检查过程中所用的多边形的边一定是 VSP(s，t)的关联边。为了减少运算，可以事先将不关联的边用线段截去（例如图 3.23 中的 $v_{l2}\,v_{l3}$）。将所有的关联边以及用于截去非关联边的线段叫做检查边。

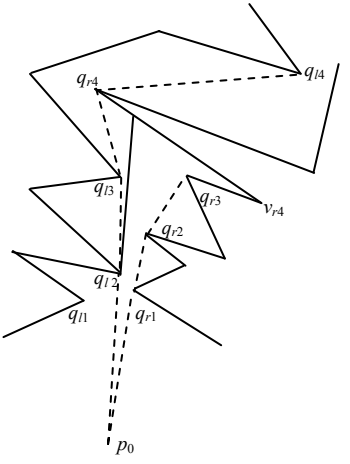


图 3.22　左切线与右侧多边形边相交

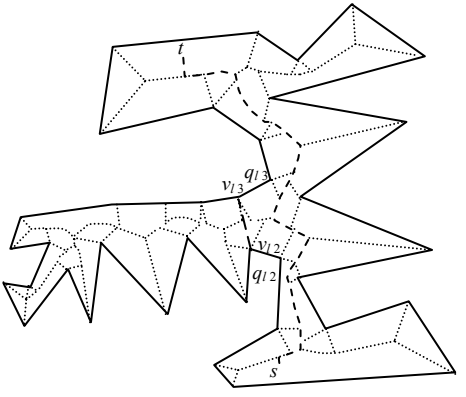
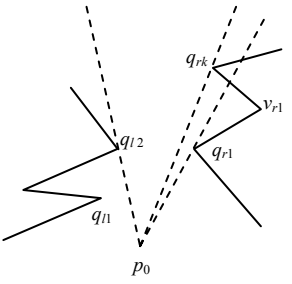
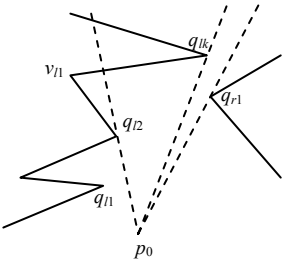


图 3.23　去掉与 Voronoi 骨架路径不关联的多边形的边

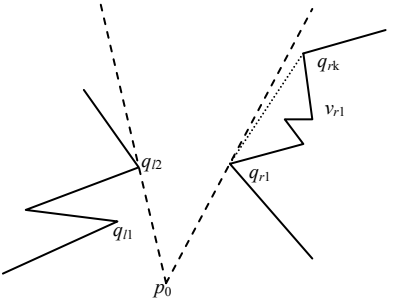
在上述计算过程中，当出现如图 3.24（$v_{r1}\,q_{rk}$、$v_{l1}\,q_{lk}$ 是当前的检查边，$p_0\,q_{l2}$、$p_0\,q_{r1}$ 是当前切线）所示的情况时，凸包链以及左、右切线需要重新计算。



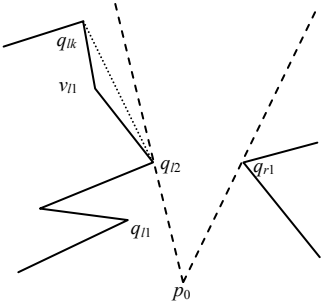
(a) q_{rk} 在 p_0q_{r1} 的左侧、p_0q_{l2} 的右侧



(b) q_{lk} 在 p_0q_{r1} 的左侧，p_0q_{l2} 的右侧



(c) q_{rk} 在 p_0q_{r1} 的右侧



(d) q_{lk} 在 p_0q_{l2} 的左侧

图 3.24　凸包链以及左、右切线需要重新计算

当出现图 3.25 中的情况时，q_{rk} 的两个相邻的顶点 v_{r1} 和 v_{r2} 不在 p_0q_{rk} 的同侧，这时候 q_{rk} 一定不是 SP(s，t)上的点。

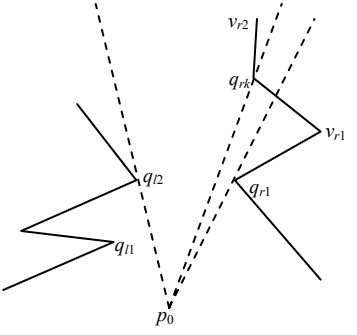


图 3.25　q_{rk} 在 p_0q_{r1} 左侧且在 p_0q_{l2} 右侧

重复上面的计算直至遇到终点 t。

下面考虑当计算遇到终点 t 时候的情况。

如果左、右凸包中没有内尖点了，则 t 处于 p_0q_{l2}、p_0q_{r1} 构成的视角之内，则 p_0、t 之间是可见的，计算结束（见图 3.26（a））。

否则，若左凸包不是空，且 t 在 p_0q_{l2} 的左侧，计算该凸包上的点与 t 形成的新凸包，新凸包上的点都是 SP(s，t)上的结点（见图 3.26（b））；如果右凸包不是空，且 t 在 p_0q_{r1} 的右侧，计算该凸包上的点与 t 形成的新凸包，新凸包上的点都是 SP(s，t)上的结点（见图 3.26（c））。整个计算结束。

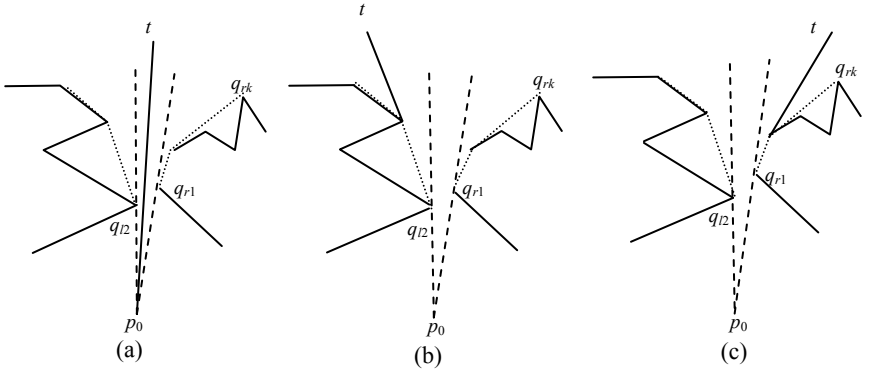


图 3.26　遇到 t 的三种情况

算法3.7　沿着Voronoi骨架路径VSP(s，t)计算最短路径SP(s，t)

令p是SP(s，t)上的当前顶点，q_k是VSP(s，t)上的当前关联内尖点，CH(q_l)是左关联内尖点q_l到q_k的左凸包链；CH(q_r)是右关联内尖点q_r到q_k的右凸包链，TL(q_l)是从p到CH(q_l)的左切线，切点为q_l，TL(q_r)是从p到CH(q_r)的右切线，切点为q_r。

（1）p=s；CH(q_l) = NULL；CH(q_r) = NULL；k =1；输出p。

（2）对于VSP(s，t)上的当前Voronoi边e_k，设$q_{k-1}q_k$是当前的检查边。

（2.1）如果$q_{k-1}q_k$是左检查边，则：

（2.1.1）如果$q_{k-1}q_k$与TL(q_r)相交，沿CH(q_r)移动p和q_r，输出p，转（2.1.1）；

（2.1.2）否则，重新计算CH(q_l)和TL(q_l)。

（2.2）如果$q_{k-1}q_k$是右检查边，则：

（2.2.1）如果$q_{k-1}q_k$与 TL(q_l)相交，沿CH(q_l)移动p和q_l，输出p，转（2.2.1）；

（2.2.2）否则，重新计算CH(q_r)和TL(q_r)。

（2.3）k++。

（3）如果CH(q_l)= NULL 且CH(q_r) = NULL，则输出t，返回。

（4）如果CH(q_l)!= NULL且t在TL(q_l)左侧，

（4.1）输出q_l；

（4.2）如果q_l是CH(q_l)的最后一个顶点，转（6）；

（4.3）设CH(q_l)上q_l后面的点为a，令$p=q_l$，$q_l=a$；转（4）。

（5）如果CH(q_r) != NULL且t在TL(q_r)右侧，

（5.1）输出q_r；

（5.2）如果q_l是CH(q_r)的最后一个顶点，转（6）；

（5.3）设CH(q_r)上q_r后面的点为b，令$p= q_r$，$q_r =b$；转（5）。

（6）输出t；返回。

算法分析：简单多边形的 Voronoi 图的顶点个数为 $n+k-2$，边数为$2(n+k)-3$，其中 n 是多边形的顶点数，$k(k<n)$是内尖点的数目，寻找骨架路径 VSP(s，t)的时间复杂度为 $O(n)$，在计算最短路径时是沿着 VSP(s，t)上关联内尖点顺序计算的，计算过程中每个关联内尖点的处理为常数次。所以通过 VSP(s，t)计算 SP(s，t)所用时间为 $O(n)$。由于找到起点与终点的步骤花费

的时间仅为 $O(n)$，所以该最短路径算法的总时间复杂度为 $O(n)$。

对于基于 Voronoi 图的复杂多边形中的最短路径计算，可参见文献[WangXT2011]。

3.3.4　复杂多边形中的可见性计算

如果多边形 P 内两点 p 和 q 之间的线段上没有多边形 P 外部的点，则 p 和 q 是相互可见的；多边形 P 中某点 p 的可见多边形 VP(p)是指 P 中所有从 p 看去可见的点的集合。计算多边形内一点 p 的可见多边形，即找到多边形中相对于 p 点可见的全部点的集合，是一系列多边形可见性计算问题中最基本的一个问题。

本节介绍一种基于多边形 Voronoi 图的点可见性算法[Zhao2013]。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。算法数据结构比较简单，剖分空间合理且易于实现，只需要 $O(n)$空间和 $O(n\log n)$预处理时间。应用实例与测试分析表明，该算法是一种实际可行的算法，且运行时间与|VP(v)|和 n 均呈线性关系，其中|VP(v)|为点 v 的可见多边形的边数，n 为多边形的边数。

本节规定，多边形的外边界是顺时针的，内边界是逆时针的。我们将多边形中的凹顶点看作特殊的边，即它的两个端点是重合的。这样，简单多边形的 Voronoi 图是一棵树，其叶结点是多边形的顶点；“带洞”多边形的 Voronoi 图是一个图，其中度数为 1 的结点为多边形的顶点。

1. 概念与性质

VP(v)有两种类型的边：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v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v最多只有一个可见部分；对于“带洞”多边形，每条边相对于v可能有多个可见部分。由于VP(v)是一个星形多边形，因此如果逆时针沿着VP(v)的边界依次遍历，v和VP(v)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。算法将沿着多边

形的Voronoi图进行深度优先搜索，访问多边形中的边，并计算被访问边的相对于*v*的可见部分，其中会用到*v*到其两个顶点的Voronoi骨架路径和局部最短路径的概念。

对于多边形 P 中一点 v，假设其位于边 e 的 Voronoi 区域 VR(e)中，我们过 v 作 e 的垂线，可以证明该垂线只与 VR(e)的边界上的一条 Voronoi 边相交，不妨设交点为 a。我们将 a 看作一个结点，va 看作一条边加入 VD(P)（如图 3.27 所示）。

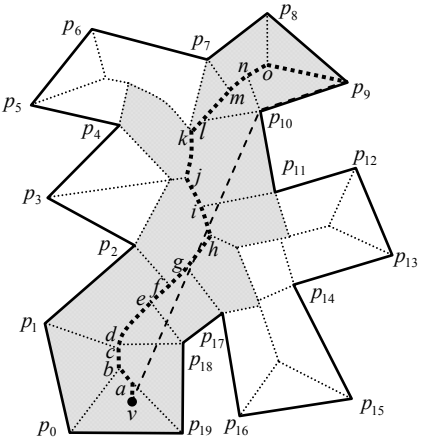


图 3.27　简单多边形中 Voronoi 骨架路径（粗虚线）和 Voronoi 通道（灰区域）

由 3.3.3 节可知，v 到 P 的一个顶点 p_i 的 Voronoi 骨架路径 VSP(v，p_i)是一条由系列 Voronoi 边首尾依次连接而成的路径。如图 3.27 中粗虚线所示，VSP(v，p_9)= {v，a，…，o，p_9}。边 p_1p_2 为有向边 de 的左侧关联边，凹顶点（特殊边）p_{18} 为有向边 de 的右侧关联边。需要注意的是，p_1p_2 为有向边 ed 的右侧关联边，p_{18} 为有向边 ed 的左侧关联边。

多边形 P 中的一个点 v 到 P 的一个顶点 p_i 的 Voronoi 骨架路径 VSP(v，p_i)的每条 Voronoi 边所关联的 Voronoi 区域形成 v 到 p_i 的一个通道，称之为 VSP(v，p_i)对应的 Voronoi 通道 C(v，p_i)。v 到两个顶点 p_i 、p_{i+1} 的 Voronoi 通道 C(v，p_i)和 C(v，p_{i+1})组成了 v 到边 p_ip_{i+1} 的 Voronoi 通道 C(v，p_i p_{i+1})。

这里，VSP(v，p_i)和 VSP(v，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($p_i p_{i+1}$)的 Voronoi 边不同，其他边都是相同的。

在一个 C(v，p_i)中，将 VSP(v，p_i) 看作一条绳子，如果用力将其拉紧，则可得到在该 C(v，p_i)中 v 到 p_i 的一条最短路径，我们将其称之为局部最短路径，用 $SP_c(v, p_i)$表示。

图 3.27 中，灰色区域为 VSP(v，p_9)所对应的 Voronoi 通道 C(v，p_9)，图 3.28 中灰色区域为 v 到 p_3 的两条 Voronoi 骨架路径分别对应的 Voronoi 通道。图 3.27 中，VSP(v，p_8)= {v，a，…，o，p_8}，VSP(v，p_9)= {v，a，…，o，p_9}，它们不同的部分是 op_8 和 op_9，都是 VR($p_8 p_9$)的 Voronoi 边。在 C(v，$p_8 p_9$)中，v 到 p_8 的局部最短路径为 $SP_c(v, p_8)$={v，p_8}，v 到 p_9 的局部最短路径为 $SP_c(v, p_9)$={v，p_{10}，p_9}。需要注意的是，在“带洞”多边形中，v 到 p_i 存在多个 Voronoi 通道，因此存在多个局部最短路径。如在图 3.28（a）中的 Voronoi 通道 C(v，p_3)（灰色区域）中，$SP_c(v, p_3)$={v，p_3}；在图 3.28（b）中，还存在一个 Voronoi 通道（灰色区域），其中 $SP_c(v, p_3)$={v，p_{11}，p_{13}，p_3}。

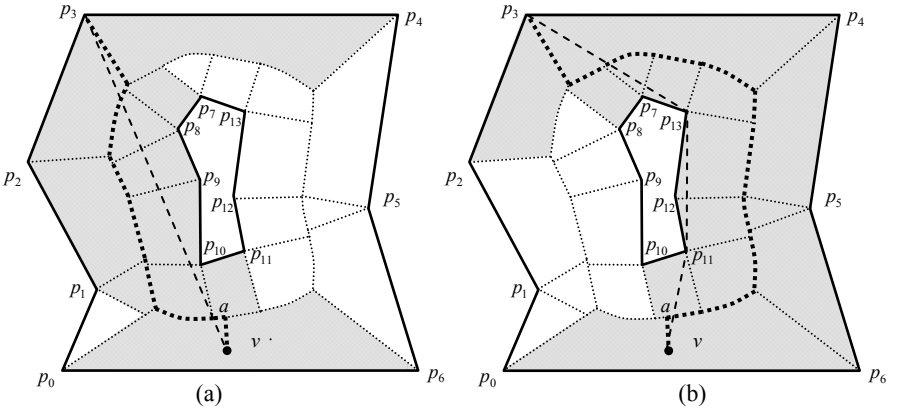


图 3.28　“带洞”多边形中 Voronoi 骨架路径（粗虚线）和 Voronoi 通道（灰区域）

2. 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方法。

由于对于一个多边形 P 及其内部一点 v，v 到 P 的顶点 p_i 的一个 Voronoi 通道可以看作一个简单多边形，由定理 3.13 可得引理 3.12。

引理 3.12 对于一个多边形 P 及其内部一点 v，在 v 到 p_i 的一个 Voronoi 通道中，v 到 p_i 的局部最短路径 $SP_c(v, p_i)$能够通过顺序遍历 $VSP(v, p_i)$得到，且 $SP_c(v, p_i)$上的顶点必为 $VSP(v, p_i)$所关联的内尖点。

引理 3.13 对于一个多边形 P 及其内部一点 v，在 v 到 P 的边 p_ip_{i+1} 的一个 Voronoi 通道中，p_ip_{i+1} 相对于 v 存在可见部分，当且仅当 v 到 p_i 的局部最短路径 $SP_c(v, p_i)$是一条在 P 内的凸多边形链且在 vp_i 的右侧，且 v 到 p_{i+1} 的局部最短路径 $SP_c(v, p_{i+1})$也是一条在 P 内的凸多边形链且在 vp_{i+1} 的左侧。

证明：充分性：由引理 3.12 知，$SP_c(v, p_i)$上的点为 $VSP(v, p_i)$所关联的内尖点。如果只用 $VSP(v, p_i)$的左侧关联内尖点计算 $SP_c(v, p_i)$，则 $SP_c(v, p_i)$为 $VSP(v, p_i)$的左侧关联内尖点的凸多边形链 L_CP。在 p_ip_{i+1} 相对于 v 存在可见部分的情况下，若 $SP_c(v, p_i)\neq$L_CP，即 $SP_c(v, p_i)$非凸，则存在 $SP_c(v, p_i)=\{v, l_1, \ldots, l_l, p_i\}$中一个顶点 $l_j(1\leqslant j\leqslant l)$为 $VSP(v, p_i)$的右侧关联内尖点，则 $SP_c(v, p_i)$和 $SP_c(v, p_{i+1})$有交错部分，p_ip_{i+1} 相对于 v 不可见，存在矛盾。因此，$SP_c(v, p_i)=$ L_CP，即 $SP_c(v, p_i)$是一条在 P 内的凸多边形链。显然，该链在 vp_i 的右侧。同理可证 $SP_c(v, p_{i+1})$也是一条在 P 内的凸多边形链且在 vp_{i+1} 的左侧。

必要性：设 $SP_c(v, p_i)=\{v, l_1, \cdots, l_l, p_i\}$，$SP_c(v, p_{i+1})=\{v, r_1, \cdots, r_r, p_{i+1}\}$，显然，$SP_c(v, p_i)$在 $VSP(v, p_i)$的左侧，$SP_c(v, p_{i+1})$在 $VSP(v, p_{i+1})$的右侧。由于 $VSP(v, p_i)$在 $VSP(v, p_{i+1})$的左侧，因此 $SP_c(v, p_i)$在 $SP_c(v, p_{i+1})$的左侧。由于 $SP_c(v, p_i)$是一条凸多边形链且在 vp_i 的右侧，且 $SP_c(v, p_{i+1})$也是一条凸多边形链且在 vp_{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 v 有可见部分。证毕。

如图 3.29 所示，对于边 p_4p_5，有局部最短路径 $SP_c(v, p_4)=\{vp_3p_4\}$，$SP_c(v, p_5)=\{vp_6p_5\}$，符合引理 3.13 的条件，可以断定 p_4p_5 相对 v 有可见部分。而对于边 p_3p_4，有局部最短路径 $SP_c(v, p_3)=\{vp_3\}$，$SP_c(v, p_4)=\{vp_3p_4\}$，

不符合引理 3.13 的条件，所以可以断定 p_3p_4 相对于 v 不可见（严格地说，点 p_3 相对于 v 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）。

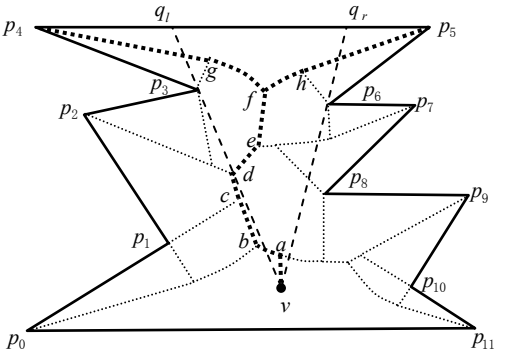


图 3.29　计算边的可见部分

由引理 3.12 和引理 3.13 容易得到定理 3.14。

定理 3.14　对于一个多边形 P 及其内部一点 v，在 v 到 P 的边 $p_i\,p_{i+1}$ 的一个 Voronoi 通道中，有 $SP_c(v, p_i)= \{v, l_1, \ldots, l_l, p_i\}$，$SP_c(v, p_{i+1})= \{v, r_1, \cdots, r_r, p_{i+1}\}$。$p_ip_{i+1}$ 相对于 v 存在可见部分，当且仅当 vl_1 在 vr_1 的左侧，而且当前 Voronoi 通道中 p_ip_{i+1} 相对于 v 的可见部分可通过计算射线 vl_1 和 vr_1 与 p_ip_{i+1} 的交点得到。

如图 3.29 中，vp_3 在 vp_6 的左侧，根据定理 3.14 知，$p_4\,p_5$ 相对于 v 存在可见部分。通过计算射线 vp_3、vp_6 与 p_4p_5 的交点即可计算出 p_4p_5 相对于 v 的可见部分为 q_lq_r。这里，射线 vl_1 和 vr_1 之间的区域成椎体状，称为可见椎体，用 $VC(vl_1, vr_1)$表示。

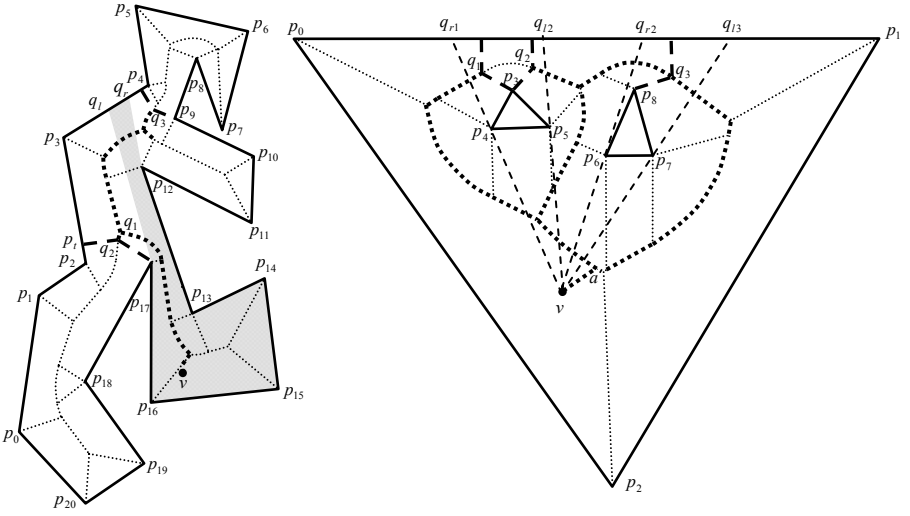
为了计算整个 VP(v)，我们将沿着 VD(P)进行深度优先搜索，访问每个叶结点，即访问 P 中的每个顶点。由于访问中得到的相邻两个顶点属于同一条边，所以可以在前一顶点的 Voronoi 骨架路径基础上，增加或减少新访问的 Voronoi 边，即可得到下一顶点的 Voronoi 骨架路径；并在计算新的 Voronoi 骨架路径的同时，在前一顶点的局部最短路径的基础上计算下一顶点的局部最短路径。这样可减少计算量。

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，原因在于以下两点。

（1）在图 3.30（a）中，VP(*v*)（灰色区域）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 *v* 不可见。

（2）在图 3.30（b）中的“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 *v* 不可见，所以可以停止这个方向的遍历。

下面讨论如何设定终止条件。很容易错误地认为：根据定理 3.14，如果 vl_1 不在 vr_1 的左侧，则可以停止继续往下遍历并开始回溯。但在图 3.31 中，在沿“洞”H_1 左侧遍历到顶点 p_3 时，vl_1=vp_1 不在 vr_1=vp_1 的左侧，但仍然需要继续遍历，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 *v* 的可见部分就会漏掉。



(a) 简单多边形　　(b) 带两个“洞”的多边形

图 3.30　终止条件示例（粗虚线）

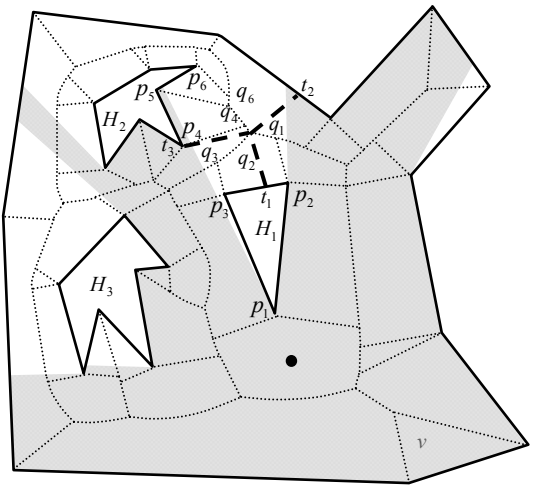


图 3.31　遍历方向不同，q_2 处的终止条件计算方法不同

不妨设当前遍历的骨架路径为 v，a，a_1，a_2，…，a_j，当前的可见椎体为 VC(vl_1, vr_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 $a_j a_{j+1}$ 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 3.31 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 3.15　在当前访问 Voronoi 通道中，若点 t_1 和 t_2 同时在 vl_1 的左侧或在 vl_1 上，或者同时在 vr_1 的右侧或在 vr_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(vl_1，vr_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(vl_1，vr_1)中，即 p 相对于 v 可见，则有 $SP_c(v, p)= (v, p)$。然而，由于 t_1 和 t_2 同时在 vl_1 左侧或在 vl_1 上，或者同时在 vr_1 右侧或在 vr_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(vl_1，vr_1)的一侧，因而可知 l_1 或 r_1 必为 v 到 p 的局部最短路径上一点，这与 $SP_c(v, p)= (v, p)$矛盾，定理 3.15 得证。

证毕。

定理 3.15 给出了沿 VD(P)进行深度遍历的终止条件。我们根据定理 3.15 设计函数 TC(q)，用于表示在当前访问的 Voronoi 通道中 Voronoi 顶点 q 是否满足终止条件。如果 TC(q)=True，则满足终止条件。如图 3.30（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_t 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)为真。在图 3.30（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）判断了两次终止条件。粗虚线为搜索过程中遍历到的 Voronoi 边。在图 3.31 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)为假；访问到 q_2 时，用到垂足 t_3 和 t_1，因为点 t_3 相对于点 v 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 v 可见，TC(q_2)同样为假；只有遍历到 q_3 时有 TC(q_3)为真，此次遍历才终止。

基于上面的描述，算法 3.8 给出了计算点的可见多边形的算法。这里假设给定一个有 n 条边和 $h(\geq 0)$个“洞”的多边形 P 以及 P 中一点 v，在预处理阶段已计算出 P 的 Voronoi 图 VD(P)。

算法 3.8　计算点的可见多边形

（1）判断 v 所在的 Voronoi 区域 VR(o)，假设 $o=p_ip_{i+1}$；

（2）计算 VSP(v，p_i)和 VSP(v，p_{i+1})，同时计算 $SP_c(v，p_i)$和 $SP_c(v，p_{i+1})$；

（3）计算 o 对于 v 的可见部分；

（4）取 $SP_c(v，p_{i+1})$最后两个点 q_{r-1}、q_r；//q_r 为多边形的顶点；

（5）如果 TC(q_{r-1})=False

（5.1）取 $q_{r-1}q_r$ 右侧关联边，作为新的 p_ip_{i+1}；

（5.2）如果 $p_ip_{i+1}=o$，算法结束；

（5.3）计算 VSP(v，p_i)和 VSP(v，p_{i+1})，同时计算 SP_c(v，p_i)和 SP_c(v，p_{i+1})；

（6）否则，

（6.1）取 $q_{r-2}q_{r-1}$ 右侧关联边，作为新的 p_ip_{i+1}；

（6.2）计算 VSP(v，p_i)和 VSP(v，p_{i+1})，同时计算 SP_c(v，p_i)和 SP_c(v，p_{i+1})；

（7）转（3）。

需要说明的是，算法 3.8 中的第（2）、（5.3）和（6.2）步中计算 VSP(v，p_i)和 VSP(v，p_{i+1})的方法不同。下面将分别对它们进行讨论。

3. Voronoi 骨架路径和局部最短路径的计算

算法 3.8 的第（1）步首先确定给定点 v 所在的 Voronoi 区域 VR(o)。我们可通过逐个测试每个 Voronoi 区域来实现，需要 $O(n)$时间。算法 3.8 的第（2）步用于计算初始的 Voronoi 最短路径和局部最短路径。在加入了点 a 和边 va 的 VD(P)中，令 VSP(v，p_i)={v，a，q_k，…，q_0，p_i}，VSP(v，p_{i+1})={v，a，q_{k+1}，…，q_j，p_{i+1}}，其中 q_0，q_1，…，q_j 为 VR(o)的 Voronoi 顶点，a 在 q_k 和 q_{k+1} 之间。采用类似 3.3.3 节中的方法，在遍历 VSP(v，p_i)(VSP(v，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局部最短路径中，修改生成新的局部最短路径，最后可得到 SP_c(v，p_i)(SP_c(v，p_{i+1}))。如图 3.32 所示，在加入了点 a 和边 va 的 VD(P)中，VSP(v，p_{13})={v，a，q_8，q_7，…，q_0，p_{13}}，VSP(v，p_0)={v，a，q_9，q_{10}，…，q_{14}，p_0}。在遍历 VSP(v，p_{13})(VSP(v，p_0))的同时，计算局部最短路径，最后可得到 SP_c(v，p_{13})={v，p_7，p_9，p_{13}}(SP_c(v，p_0)={v，p_4，p_2，p_0})。由于 vp_7 在 vp_4 的左侧，因此边 p_0 p_{13} 相对于点 v 的可见部分为 q_rq_l，其中 q_rq_l 为 vp_7、vp_4 与 p_0p_{13} 的交。

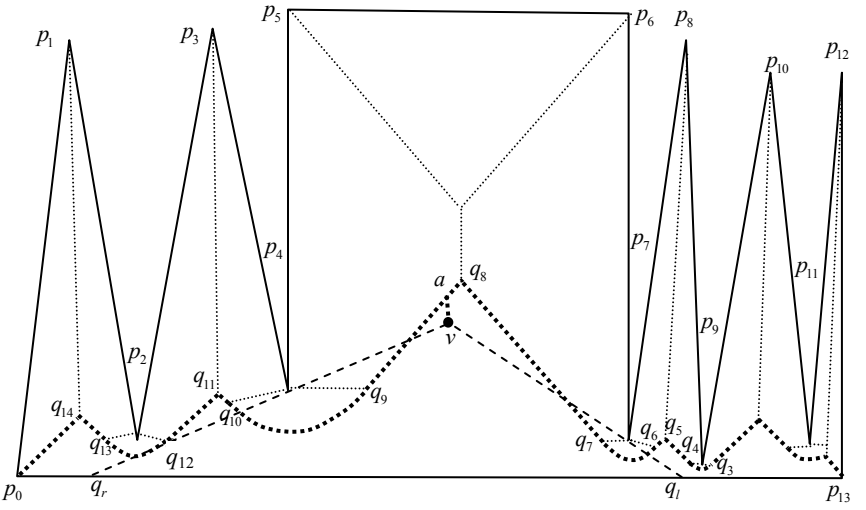


图 3.32　计算初始 Voronoi 骨架路径

我们注意到，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)为真，这时没必要继续访问 q_3、q_2、q_1 和 q_0。这时得到的局部最短路径的第一条边 vp_7 就是最后用于计算 $p_{13}p_0$ 相对于 v 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 v 到 p_{13} 的最短路径的第一条边仍为 vp_7，因为 q_3，q_2，q_1 和 q_0 在 vp_7 左侧。因此，这里可以测试终止条件，减少计算。

在已计算 VSP(v，p_{i+1})和 SP_c(v，p_{i+1})的基础上，考虑如何快速确定下一条边对应的两条 Voronoi 骨架路径和局部最短路径。这取决于 TC(q_{r-1})的值，如下所述。

（1）TC(q_{r-1})=False 时，可以继续往前遍历。这时，取 SP_c(v，p_{i+1})中的最后两个点 q_{r-1}、q_r。我们知道 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}。这时，VSP(v，p_{i+1})和 SP_c(v，p_{i+1})已知，只须计算 VSP(v，p_{i+2})和 SP_c(v，p_{i+2})。令 VSP(v，p_{i+2})= VSP(v，p_{i+1})，SP_c(v，p_{i+2})= SP_c(v，p_{i+1})。对 VD(P)进行遍历，直到到达 p_{i+2} 对应的叶结点，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(v，

p_{i+2})中，需要将其从 VSP(v，p_{i+2})中删除；否则，将其追加到 VSP(v，p_{i+2})中，同时修改 $SP_c(v, p_{i+2})$。

在图 3.29 中，已知 VSP(v,p_4)={$vabcdefgp_4$}，需要在其基础上计算 VSP(v，p_5)。VSP(v，p_5)初始值为{$vabcdefgp_4$}。在对 VD(P)进行遍历到达 p_5 对应的叶结点过程中，需要从 VSP(v,p_5)中删除 gp_4 和 fg，加入 fh 和 hp_5，得到 VSP(v，p_5)={$vabcdef\,hp_5$}。在这个过程中，同时修改 $SP_c(v, p_5)$，最后得到 $SP_c(v, p_5)$={vp_6p_5}。

类似前面计算初始 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继续往前遍历。这时得到的局部最短路径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，须转算法 3.8 的第（6）步。

（2）TC(q_{r-1})=True 时，不用继续往前遍历。这时，取 $SP_c(v, p_{i+1})$中的最后三个点 q_{r-2}、q_{r-1}、q_r。我们知道 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}。这时做如下处理。

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

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

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 3.30（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 $SP_c(v, p_2)$={v，p_{17}，p_2}，$SP_c(v, p_3)$={v，p_{17}，p_3}，可知 p_2p_3 相对于 v 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 3.30（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 $SP_c(v, p_4)$={v，p_4}，$SP_c(v, p_3)$={v，

p_4，p_3}，可知 p_3p_4 相对于 v 不可见。

3.3.5 虚拟室内场景设计与漫游系统

虚拟室内场景可以看作是将二维平面多边形垂直拉伸后得到三维墙体，然后再在其中布置物品。虚拟博物馆设计与漫游系统即为其中一种具体应用[WangL2006]。其场馆的基本数据结构是 2D 多边形及其 Voronoi 图，它们是协同设计、馆内物品组织、可见性计算、碰撞检测、路径规划等的基础。建立 3D 博物馆场景时，使用的方法是将 2D 多边形进行高度拉伸获得 3D 博物馆外墙；通过 2D 多边形的 Voronoi 图计算多边形的 Offset 曲线，偏移的距离由用户交互给出，将 Offset 曲线进行一定高度的拉伸，可以作为沿墙而设的展台；用户可以协同地往不同的 Voronoi 区域中放置各种形式的数字展品；场景中的数字展览品和虚拟替身通过 Voronoi 区域进行定位与管理；基于多边形的 Voronoi 图设计路径规划、可见性计算、碰撞检测等算法，用于加速漫游。图 3.33 显示了多边形及其 Voronoi 图的应用框架。本章介绍的多边形 Voronoi 图及其应用算法可用于此系统。

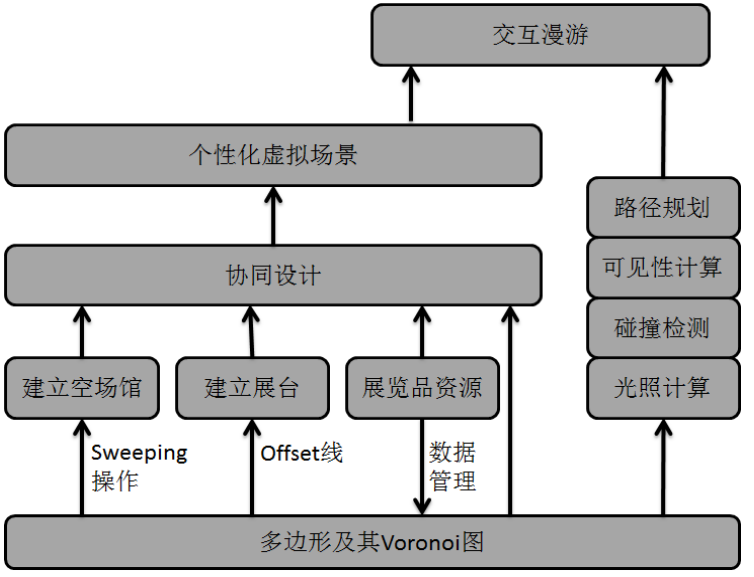


图 3.33　多边形及其 Voronoi 图的应用框架