3

多边形三角剖分：画廊看守

面对出自名家手笔的绘画作品，怦然心动的可不只是艺术爱好者，罪犯们也是如此。这类作品价值不菲、易于运输，而且很显然，不愁出不了手。正因为此，艺术画廊都必须对其拥有的作品严加看管。白天，可以由值班人员担负起看守的任务，然而到了晚上，这项任务就落到了摄像机的肩上。通常，这些摄像机都被挂在天花板上，绕着某个垂直的轴旋转。由摄像机采集到的图像，将被传送到守夜值班室的电视屏幕上。显然，眼睛同时要盯住的屏幕数量越少，守夜员就可以更轻松一些，因此，总是希望能够尽可能地减少摄像机的数目。摄像机数目更少的另外一个好处还在于，相应的保安系统可以成本更低。另一方面，摄像机的数目也不可能过少——因为，画廊内的每一个角落，都必须被落在至少一台摄像机的视野之内。

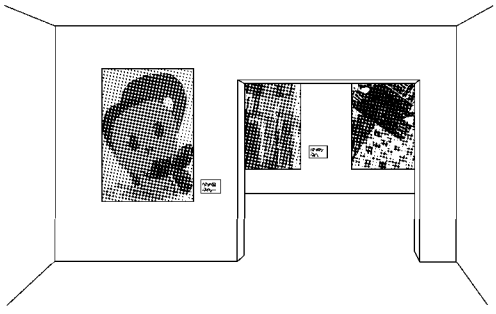

图 3-1　艺术画廊

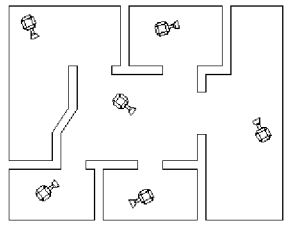

图 3-2　监视画廊的一组摄像机

因此，摄像机的安装位置很有讲究。我们的目标是：使每台摄像机都能在画廊中照应到更大的范围。这样，就出现了通常所谓的艺术画廊问题（art gallery problem）：如图 3-2 所示，给定一个画廊，需要多少台摄像机？应该将它们分别安装在什么位置？

3.1 看守与三角剖分

为了更确切地对艺术画廊问题做一定义，必须首先将画廊的概念做形式化处理。自然，每个画廊都是一个三维空间，然而通过它的平面结构图，我们就可以获得足够的信息来确定摄像机的安放位置。因此，可以利用平面多边形的模型来表示一个画廊。我们还进一步做出限制，要求画廊的模型应是简单多边形（simple polygon）——即由单个不自交的、封闭的多边形链所围出的区域[1]。这样，就不允许出现空洞。一台摄像机在画廊中的位置，对应于多边形中的一个点。如图 3-3 所示，对于多边形内部的任何一点，只要联接于它与某台摄像机之间的开线段完全落在多边形的内部，它就能被这台摄像机监视到。

为了看守一个简单多边形，需要多少台摄像机呢？显然，这要取决于具体的多边形：多边形越是复杂，需要的摄像机就越多。因此，我们将根据多边形的顶点数目 n，来界定所需摄像机的数量。然而，即使是顶点数目相等的两个多边形，其中的一个，也可能比另一个更容易看守。为了保险起见，我们所考虑的将是最坏的情况——也就是说，将要给出的只是一个上界（upper bound），该上界适用于由 n 个顶点组成的所有简单多边形（如果能够为特定的某个多边形，找到其所需摄像机的最小数目，而不是一个笼统的最坏上限，那自然是再好不过的了。但不幸的是，“计算出特定多边形所需摄像机的最小数目”这一问题，是NP-难的（NP-hard）[2]）。

设 $\mathcal{P}$ 为包含 n 个顶点的简单多边形。在确定为看守 $\mathcal{P}$ 所需摄像机的最小数目时，由于 $\mathcal{P}$ 的形状可能极为复杂，所以我们似乎无从下手。首先将 $\mathcal{P}$ 分解为很多块，每一块都很容易看守——具体而言，这里的“块”就是三角形。为了完成这种分解，需要添加一些对角线（diagonal），将某些顶点对联接起来。所谓对角线是一条开的线段，它联接于 $\mathcal{P}$ 的某两个顶点之间，而且完全落在 $\mathcal{P}$ 的内部[3]。通过极大的[4]一组互不相交的对角线，可将一个多边形分解为多个三角形——称为该多边形的三角剖分（triangulation），如图 3-4 所示（由于这一互不相交的对角线集合必须满足最大化的条件，故可以保证任何三角形各边的内部，都不包含多边形的任何顶点。如果多边形中存在三个共线的顶点，就可能会出现这种情况）。通常，简单多边形的三角剖分不是唯一的。例如，图 3-4 所示的这个多边形，就有多种不同的三角剖分方案。给定 $\mathcal{P}$ 的一个三角剖分 $\mathcal{T}_{\mathcal{P}}$，只要在（由此确定的）每个三角形中放置

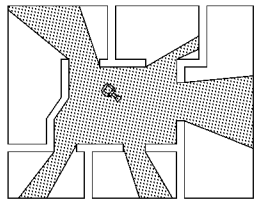

图 3-3 单台摄像机所能看守的区域

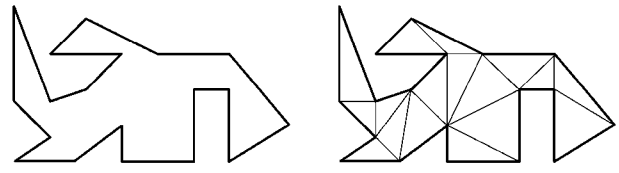

图 3-4 一个简单多边形，及其可能的一个三角剖分

1 更准确而简洁地，简单多边形的边界是一条约当曲线（Jordan curve）——译者。

2 参见[246]——译者。

3 请注意：根据这一定义，多边形的边不是对角线，而且对角线不能通过任何顶点，更不能与任何边有局部重合——译者。

4 所谓“极大”，指的是再增加任何一条对角线，都会与其中原有的某条对角线相交——译者。

一台摄像机，就可以实现对整个多边形的看守。然而，是否每个简单多边形总存在一个三角剖分呢？如果存在，其中三角形的数目又是多少呢？下面这则定理回答了这些问题。

【定理 3.1】

任何简单多边形都存在（至少）一个三角剖分；若其顶点数目为 n，则它的每个三角剖分都恰好包含 $n-2$ 个三角形。

【证明】

我们通过对 n 进行归纳来证明。若 $n=3$，则多边形本身就是一个三角形，此时是定理的平凡形式，显然成立。现假设 $n>3$，而且该定理对所有 $m<n$ 都成立。任取一个有 n 个顶点的简单多边形 $\mathcal{P}$。我们将首先证明，在 $\mathcal{P}$ 中存在（至少）一条对角线。

令 v 为在 $\mathcal{P}$ 中最靠左的顶点（如果这种顶点有多个，只选用其中的最低者）。沿 $\mathcal{P}$ 的边界，考虑与 v 相邻的那两个顶点，分别记为 u 和 w。若开线段 $\overline{uw}$ 完全落在 $\mathcal{P}$ 的内部（如图 3-5 所示），则它就是一条对角线。

否则，如图 3-6 所示，在由 u、v 和 w 所确定的三角形内部，必然存在至少一个顶点（当然，它也可能正好落在开线段[1] $\overline{uw}$ 上）。在这些顶点中，令 v' 为与 $\overline{uw}$ 相距最远者。现在，联接 v' 和 v 的那条（开）线段，不可能与 $\mathcal{P}$ 的任何边相交——否则，这条边的两个端点中，必有其一会落在这个三角形内部，而且该端点到 $\overline{uw}$ 的距离要比 v' 更远。因此，$\overline{vv'}$ 就是一条对角线[2]。

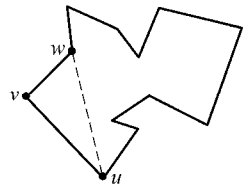

图 3-5 情况 1：线段 $\overline{uw}$ 完全落在 $\mathcal{P}$ 的内部

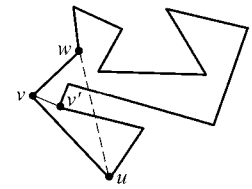

图 3-6 情况 2：线段 $\overline{uw}$ 不完全落在 $\mathcal{P}$ 的内部

既然对角线必定存在，那么这样一条对角线就会将 $\mathcal{P}$ 切分为两个子多边形 $\mathcal{P}_1$ 和 $\mathcal{P}_2$。分别记 $\mathcal{P}_1$、$\mathcal{P}_2$ 的顶点数目为 m_1、m_2，则 m_1 和 m_2 都小于 n。这样，根据归纳假设，$\mathcal{P}_1$ 和 $\mathcal{P}_2$ 都能够被三角化。于是，$\mathcal{P}$ 也能够被三角化。

现在只需要再说明：$\mathcal{P}$ 的任何三角剖分都必然由 $n-2$ 个三角形构成。为此，任取 $\mathcal{T}_\mathcal{P}$ 中的一条对角线。这条对角线将 $\mathcal{P}$ 切分为两个子多边形，设它们各有 m_1、m_2 个顶点。除了该对角线的两个端点之外，其他的每个顶点只归属于这两个子多边形中的某一个。于是就有 $m_1+m_2=n+2$。根据归纳假设，$\mathcal{P}_i$ 的任何三角剖分都由 m_i-2 个三角形组成（$i=1$ 或 2），故 $\mathcal{T}_\mathcal{P}$ 包含的三角形数目为 $(m_1-2)+(m_2-2)=n-2$。 ❑

由【定理 3.1】可得出推论：包含 n 个顶点的任一简单多边形，都可用 $n-2$ 台摄像机来看守。然而，为每个三角形配备一台摄像机，不免有些浪费。比如，只要将一台摄像机安装在任何一条对角线上，就可以看守住（与该对角线关联的）两个三角形——也就是说，

1 此处原文为“对角线”。与前面的定义不合——译者。

2 更严密地，还应该说明“开线段 $\overline{vv'}$ 完全落在多边形内部”。这可以反证——否则，$\overline{vv'}$ 必然与 $\mathcal{P}$ 的某条边相交——译者。

如果精心挑选出若干对角线，然后在那些位置安装摄像机，就可能将所需摄像机的总数减少到大约 $n/2$。而似乎更好的策略是，将摄像机安装在（多边形的）顶点上——毕竟，一个顶点可能同时与更多的三角形相关联，这样，只需一台摄像机，就可以将与之相关联的所有三角形都看守住。这样，就导出了下面的方法。

令 $\mathcal{T}_{\mathcal{P}}$ 为 $\mathcal{P}$ 的一个三角剖分。选出 $\mathcal{P}$ 的部分顶点组成一个子集，使得 $\mathcal{T}_{\mathcal{P}}$ 中的每个三角形，都有至少一个顶点来自于该子集；然后，在被挑选出的每个顶点处，分别放置一台摄像机。为了找出这样一个子集，可以使用白、灰和黑三种颜色，给 $\mathcal{P}$ 的所有顶点染色（如图 3-7 所示）。我们的染色方案必须满足：由任何边或者对角线联接的两个顶点，所染的颜色不能相同——称为“对经过三角剖分后的多边形的 3-染色（3-coloring）”。三角剖分后的多边形经过如此染色，其中每个三角形都有（且仅有）一个白色、灰色和黑色的顶点。因此，只要在同色（比如灰色）的各顶点处分别放置一台摄像机，就必然可以看守整个多边形。进一步地，若选用点数最少的那一类同色顶点，并为它们配备摄像机，则只需不超过 $\left\lfloor \frac{n}{3} \right\rfloor$ 台摄像机，即可看守住 $\mathcal{P}$。

然而，3-染色方案是否总是存在？答案是肯定的。为了理解这一结论，让我们来看看所谓“$\mathcal{T}_{\mathcal{P}}$ 的对偶图”（如图 3-8 所示）——记之为 $\mathcal{G}(\mathcal{T}_{\mathcal{P}})$。对应于 $\mathcal{T}_{\mathcal{P}}$ 中的每个三角形，$\mathcal{G}(\mathcal{T}_{\mathcal{P}})$ 都有一个顶点。将对应于顶点 v 的三角形记作 $t(v)$。若 $t(v)$ 与 $t(u)$ 共用一条对角线，则在 v 和 u 之间就设置一条弧。这样，$\mathcal{G}(\mathcal{T}_{\mathcal{P}})$ 中的各条弧就分别对应于 $\mathcal{T}_{\mathcal{P}}$ 中的各条对角线。任何一条对角线都会将 $\mathcal{P}$ 一分为二，故移去 $\mathcal{G}(\mathcal{T}_{\mathcal{P}})$ 的任意一条弧，$\mathcal{G}(\mathcal{T}_{\mathcal{P}})$ 都会分裂成两个（各自连通的）部分。因此，$\mathcal{G}(\mathcal{T}_{\mathcal{P}})$ 必然是一棵树——当然，如果允许多边形内含有空洞，这个结论就不一定成立）。这样，只要对该图进行一次（比如，深度优先）遍历（traverse），就可以得到一种 3-染色的方案。以下介绍具体的做法。在深度优先遍历的过程中，始终都保证这样一点：已经访问过的三角形的所有顶点，都已被染上了白色、灰色或黑色；而且，任何一对（通过对角线或边）相互联接的顶点，颜色互异。由此可以保证：在访问完所有的三角形之后，可得到一个 3-染色的方案。深度优先遍历可从 $\mathcal{G}(\mathcal{T}_{\mathcal{P}})$ 的任一顶点开始；第一个被访问的三角形，其三个顶点将分别被染上白色、灰色或黑色（次序无所谓）。现在，假设从 $\mathcal{G}$ 的一个顶点 u 到达另一个顶点 v。既然如此，$t(v)$ 和 $t(u)$ 之间肯定存在一条公共对角线。由于 $t(u)$ 的三个顶点都已经被染上了互异的颜色，所以 $t(v)$ 的三个顶点中只有一个顶点需要染色。而且，只有一种颜色可供它使用——准确地，就是 $t(v)$ 与 $t(u)$ 之间公共对角线所没有用到的那种颜色。$\mathcal{G}(\mathcal{T}_{\mathcal{P}})$ 是一棵树，故在此时，与 v 相邻（除 u 之外）的其他顶点都尚未访问到，因此的确可以将剩下的这一颜色赋给这个顶点。

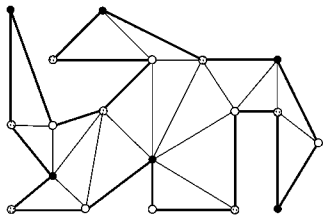

图 3-7 根据三角剖分对顶点进行 3-染色

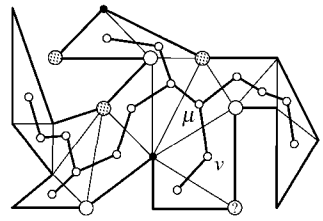

图 3-8 3-染色方案必然存在

总而言之，对于经过三角剖分的任意简单多边形，都能够（对其顶点）实施 3-染色。于是，只需$\left\lfloor \frac{n}{3} \right\rfloor$台摄像机，就可以看守住任何一个（包含 n 个顶点的） 简单多边形。不过，我们的成本还可能更低。毕竟，放置在顶点处的一台摄像机，其能够看守的范围，可能不止是与之相关联的那些三角形。然而不幸的是，对任何 $n \geqslant 3$，都存在一个（包含 n 个顶点的）简单多边形，它的确需要$\left\lfloor \frac{n}{3} \right\rfloor$台摄像机。这样的一个例子就是所谓的“梳状多边形”（comb-shaped polygon）：如图 3-9 所示，它有一条长长的水平基边，以及$\left\lfloor \frac{n}{3} \right\rfloor$个分别由两条边形成的“梳齿”。任何两个相邻的梳齿之间，由一条水平边相联。只要适当地安排各顶点的位置，就总能够保证：单台摄像机无论放置在多边形内的什么位置，都不可能同时看到两个梳齿。因此，我们不能指望能够依靠某种策略，每次都找到少于$\left\lfloor \frac{n}{3} \right\rfloor$台摄像机。换而言之，就最坏情况来而言，上述 3-染色的方法已经是最优的了。

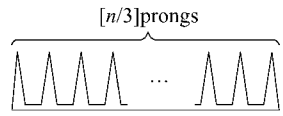

图 3-9 梳状 n 边形需要$\left\lfloor \frac{n}{3} \right\rfloor$台摄像机

以上就证明了组合几何学（combinatorial geometry）的一个经典结果：

【定理 3.2（艺术画廊定理）】

包含 n 个顶点的任何简单多边形，只需（放置在适当位置的）$\left\lfloor \frac{n}{3} \right\rfloor$台摄像机就能保证：其中任何一点都可见于至少一台摄像机。有的时候，的确需要这样多台摄像机。

现在我们已经知道，$\left\lfloor \frac{n}{3} \right\rfloor$台摄像机总是够用的。然而，我们还没有有效的算法，以计算出各台摄像机的具体位置。为此，需要一个快速的算法，以实现对任何简单多边形的三角剖分。同时，通过该算法，还应该能够导出一个合理的数据结构（比方说，双向链接边表），来表示三角剖分后的结果——这样，（在遍历时）只需常数时间，就可以从一个三角形转到它的一个邻居。一旦已经得到了这种形式的结构表示，就可以在线性时间内，按照上述方法——深度优先遍历对偶图，完成 3-染色，按照颜色将所有顶点分为三类，取出数量最少的一类顶点，并在这类顶点处放置摄像机——确定总数不超过$\left\lfloor \frac{n}{3} \right\rfloor$台摄像机的具体位置。接下来的一节，将介绍如何在 $O(n\log n)$时间内构造一个三角剖分。提前借用这一结果，就可以得出有关多边形看守的如下最后结论。

【定理 3.3】

任给一个包含 n 个顶点的简单多边形 $\mathcal{P}$。总可以在 $O(n\log n)$时间内，在 $\mathcal{P}$ 中确定$\left\lfloor \frac{n}{3} \right\rfloor$台摄像机的位置，使得 $\mathcal{P}$ 中的任何一点都可见于其中的至少一台摄像机。

3.2　多边形的单调块划分

任给一个包含 n 个顶点的简单多边形 $\mathcal{P}$。根据【定理 3.1】，$\mathcal{P}$ 的三角剖分总是存在。那个定理的证明本身就是构造式的，故马上就可以由此导出一个递归的三角剖分算法：找到一条对角线，将原多边形切分为两个子多边形，然后递归地对两个子多边形实施三角剖分。为了找到这样一条对角线，我们找出 $\mathcal{P}$ 中最靠左的顶点 v，然后试着将与 v 相邻的两个顶点 u 和 w 联接起来；如果不能直接联接这两个顶点，就在由 u、v 和 w 确定的三角形内，找出距离 $\overline{uw}$ 最远的那个顶点，然后将它与 v 联接起来。按照这种方法，需要花费线性的时间才能找到一条对角线。而且，（在最坏情况下）这条对角线将 $\mathcal{P}$ 切分为一个三角形，以及一个含有 n–1 个顶点的多边形。我们的确可能一直都是联接 u 和 w——这就是最坏情况。在这种最坏情况下，上述三角剖分算法需要运行平方量级的时间。能否更快呢？对于某些类型的多边形，的确可以更快。

比如凸多边形（convex polygon）就很容易：如图 3-10 所示，取出多边形的任何一个顶点，除了它的两个邻居之外，在这个顶点与其他的所有顶点之间分别联接一条对角线。整个过程只需要线性的时间。因此，对非凸多边形进行三角剖分的一种可能的方法就是：首先将 $\mathcal{P}$ 划分为多个凸块，然后分别对每块做三角剖分。然而不幸的是，将多边形划分为凸块的难度，与对它做三角剖分是一样的[1]。因此，我们将把 $\mathcal{P}$ 划分为所谓的“单调块”（monotone piece）——这项工作要容易得多。

一个简单多边形称为“关于某条直线 l 单调”（monotone with respect to a line l），如果对任何一条垂直于 l 的直线 l'，l' 与该多边形的交都是连通的。换而言之，它们的交或者是一条线段，或者是一个点，也可能是空集。

如果一个多边形关于 y 坐标轴单调（如图 3-11 所示），则称它是 y 单调的（y-monotone）。下面这个性质，是 y 单调多边形（y monotone polygon）的一个特征：在沿着多边形的左（右）边界，从最高顶点走向最低顶点的过程中，我们始终都是朝下方（或者水平）运动，而绝不会向上。

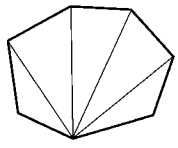

图 3-10　凸多边形的三角剖分可以在线性时间内构造出来

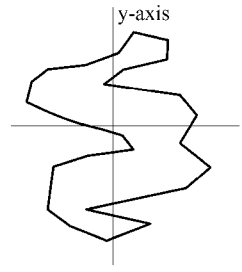

图 3-11　单调多边形

我们对多边形 $\mathcal{P}$ 进行三角剖分的策略是：首先将 $\mathcal{P}$ 划分成若干个 y 单调块，然后再对

1 比如前面所提到的梳状多边形，对它的任何凸分解，都需要划分出 $\left\lfloor \frac{n}{3} \right\rfloor = \Omega(n)$ 个三角形——译者。

每块分别进行三角剖分。可以按照下面的方法，将一个多边形划分成单调块。设想我们沿着 $\mathcal{P}$ 的左或右边界，从其最高顶点走向最低顶点。在某些顶点处，我们的行进方向可能会从向下转成向上，或者从向上转成向下——这些位置称为拐点（turn vertex）。为了将 $\mathcal{P}$ 划分成多个 y 单调块，就必须消除这些拐点。为此可以引入对角线。如图 3-12 所示，若在某个拐点 v 处，与之关联的两条边都朝下[1]，而且在此局部，多边形的内部位于 v 的上方，那么就必须构造一条从 v 出发、向上联接的对角线。

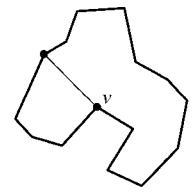

图 3-12 通过引入对角线消除拐点

这条对角线将原多边形一分为二，而且在划分出来的两块中，顶点 v 都会出现。此外，在其中的任何一块中，与 v 相关联的两条边，必然有一条朝下（具体讲，就是从原多边形中继承下来的那条边），而另一条则朝上（也就是所引入的对角线）[2]。也就是说，在两个子块中，v 都不再是一个拐点。如果与 v 相关联的两条边都朝上，而且在此局部，多边形的内部位于 v 的下方[3]，那么就需要构造一条从 v 出发、向下联接的对角线。显然，拐点有多种不同类型，故需要更加准确地加以区分。

为了更加仔细地对不同类型的拐点做出定义，需要特别注意那些 y 坐标相同的顶点。为此，要定义好“下方”和“上方”的概念：所谓“点 p 处于点 q 的下方”，是指 $p_y < q_y$，或者 $p_y = q_y$ 而 $p_x > q_x$；而所谓“点 p 处于点 q 的上方”，是指 $p_y > q_y$，或者 $p_y = q_y$ 而 $p_x < q_x$（你可以想象着相对于原来的坐标系，沿顺时针方向，将整个平面旋转“一丁点”——这样，任何两个点都不会具有相同的 y 坐标，而且上面所定义的上/下关系，在旋转后的平面上依然保持不变）。

$\mathcal{P}$ 的顶点可划分为五类（如图 3-13 所示）。其中四类都是拐点：起始顶点、分裂顶点、终止顶点以及汇合顶点。它们的定义如下。顶点 v 是一个起始顶点（start vertex），如果与它相邻的两个顶点的高度都比它低，而且在 v 处的内角小于π；如果该内角大于π，v 就是一个分裂顶点（split vertex）（注意，既然与 v 相邻的两个顶点都比 v 更低，此处的内角就不可能等于π）。顶点 v 是一个终止顶点（end vertex），如果与它相邻的两个顶点的高度都比它高，而且在 v 处的内角小于π；如果该内角大于π，v 就是一个汇合顶点（merge vertex）。这四类拐点以外的所有顶点，都是普通顶点（regular vertex）。也就是说，在每个普通顶点的两个相邻顶点中，必然有一个比它高，而另一个则比它低。之所以要给不同类型的顶点取这样的名字，是因为我们的算法要进行一次自上而下的平面扫描，在此过程中，要维护扫描线与多边形的交集。当扫描线触及一个分裂顶点时，交集中的某个（连通的）部分就要分裂；当扫描线触及一个汇合顶点时，则有两个（连通的）部分会汇合起来；诸如此类。

---

1 由于边是没有方向的，故准确地讲，应该是“与 v 相邻的两个顶点，y 坐标均低于 v”（后面的“朝上”也有这个问题）。如果再加上“在此局部多边形的内部位于其上方”的条件，则这种顶点也被称为石笋（stalagmite）——译者。

2 再次地，由于这里并没有定义边的方向，故准确的描述应该是：与 v 相邻的两个顶点中，必然有一个（的 y 坐标）低于 v（该顶点与 v 之间的边，来自原多边形），而另一个要高于 v（该顶点与 v 是所引入对角线的两个端点）——译者。

3 这种顶点也别称为钟乳石（stalactite）——译者。

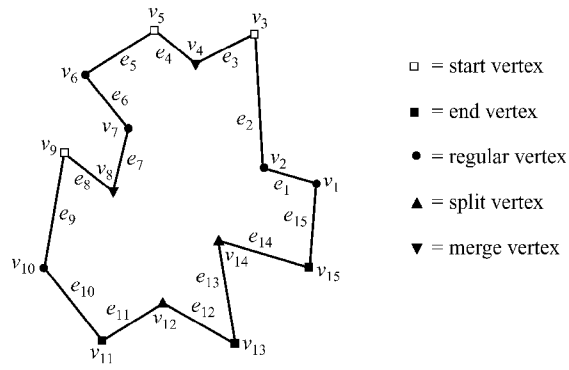

图 3-13　五种类型的顶点

多边形中局部的非单调性，正来自于这些分裂顶点和汇合顶点。而且反过来，下面这个命题看似更强，却也竟然是成立的：

【引理 3.4】
一个多边形若既不含分裂顶点，也不含汇合顶点，则必然是 y 单调的。

【证明】

假设 $\mathcal{P}$ 不是 y 单调的。我们来证明，$\mathcal{P}$ 中必然含有一个分裂顶点，或者一个汇合顶点。

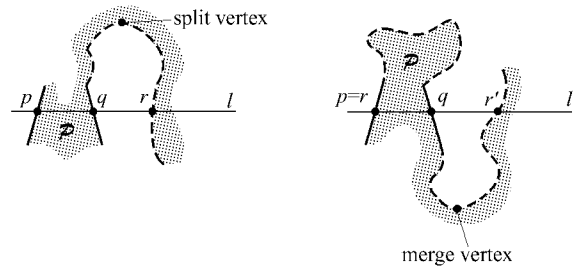

图 3-14　【引理 3.4】的证明中所涉及到的两种情况

既然 $\mathcal{P}$ 不单调，则根据定义必然存在某条水平线 l，它与 $\mathcal{P}$ 的交集含有（至少）两个（各自）连通子集。只要选取得当，总能找到一条这样的 l，其最左边的那个（连通）子集是一条线段，而不是一个点。分别令 p 和 q 为该线段的左、右端点。现在，从 q 开始，沿着 $\mathcal{P}$ 的边界行进——行进的方向要使得 $\mathcal{P}$（在任何局部都）居于左侧（也就是说，从 q 处出发向上而行）。在某一点（令其为 r）处，边界必将与 l 再次相交。若 $r \neq p$（如图 3-14(a)所示），则在从 q 通往 r 的沿途上，我们所遇到的位置最高的那个顶点必然是一个分裂顶点，此时引理成立。

反之，若 $r = p$（如图 3-14(b)所示），则我们再一次沿着 $\mathcal{P}$ 的边界，从 q 出发行进——不同的是，此次行进的方向与上次相反。与上面同理，我们将再次与 l 相交。令该交点为 r'。我们断言，不可能有 $r' = p$——否则，$\mathcal{P}$ 的边界与 l 只相交两次，这与“l 和 $\mathcal{P}$ 相交出多于一个连通子集”的前提相矛盾。因此就有 $r' \neq p$，而这意味着，在

从 q 通往 r' 的沿途中，所遇到的位置最低的那个顶点，必然是一个汇合顶点。 ❑

根据【引理 3.4】，只要将其中的分裂顶点和汇合顶点都消除掉，也就完成了将 $\mathcal{P}$ 划分为多个 y 单调块的任务。为此，需要在每个分裂顶点处增加一条向上的对角线，也要在每个汇合顶点处增加一条向下的对角线[1]。当然，这些对角线必须互不相交。一旦这些工作完成，$\mathcal{P}$ 也就已经被划分为多个 y 单调块了。

首先来看看，在一个分裂顶点处应该如何引入一条对角线。这里采用平面扫描的方法。按照顺时针的方向，令 $\mathcal{P}$ 的所有顶点排列为 $v_1, v_2, \ldots, v_n$。再令 $\mathcal{P}$ 的各边为 $e_1, e_2, \cdots, e_n$，其中对任何的 $1 \leqslant i < n$，都有 $e_i = \overline{v_i v_{i+1}}$；另外，$e_n = \overline{v_n v_1}$。按照平面扫描算法，一条假想的水平扫描线 l 自上而下地扫过整个平面。在一些被称为事件点（event point）的位置，扫描线会稍做停留。就目前这一问题而言，这些事件点包括 $\mathcal{P}$ 的所有顶点；不过，在整个扫描的过程中，不会产生任何新的事件点。所有的事件点被组织成一个事件队列（event queue）$\mathcal{Q}$。该事件队列实际上是一个优先队列，各顶点的优先级就是其各自的 y 坐标[2]。如果两个顶点的 y 坐标相同，则居于左边（x 坐标更小）的那个顶点具有更高的优先级。这样，每次只需 $O(\log n)$ 时间，就可以找出下一待处理的顶点（既然在扫描过程中不会出现新的事件，不妨在扫描之前将所有顶点按照 y 坐标排一次序——经过这一预处理，每次只需 $O(1)$ 时间就可以确定下一事件点）。

扫描的目的，是为了将每个分裂顶点，与位于其上方的某个顶点联接起来，从而引入一条对角线。如图 3-15 所示，试考虑扫描线触及某个分裂顶点 v_i 的时刻。此时，应该将 v_i 与哪个顶点相连呢？与 v_i 相距较近的顶点，是一个不错的选择——这样，在将它与 v_i 联接起来之后，连线不与 $\mathcal{P}$ 的任何边相交的可能性更大。让我们更准确地做一解释。沿着当前的扫描线，令位于 v_i 的左侧、与之相邻的那条边为 e_j；令位于 v_i 的右侧、与之相邻的那条边为 e_k。

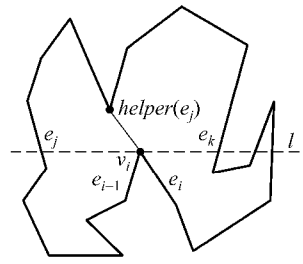

图 3-15 分裂顶点的处理

现在考虑介于 e_j 和 e_k 之间、位于 v_i 上方的那些顶点，若这些顶点至少存在一个，则总可以将其中最低的那个与 v_i 连接起来（构成一条合法的对角线）。若这类顶点根本不存在，则可将 v_i 与 e_j 或 e_k 的上端点连接起来。无论如何，我们都将这个顶点称为“e_j 的助手”（helper of e_j），记作 $helper(e_j)$。按照正式的定义，$helper(e_j)$ 应该是“在位于扫描线上方、通过一条完全落在 $\mathcal{P}$ 内部的水平线段[3]与 e_j 相连的那些顶点中，高度最低的那个顶点”。请注意，$helper(e_j)$ 可能就是 e_j 自己的上端点。

这样，我们就知道了消除分裂顶点的方法——分别将它们与各自左侧那条相邻边的助手相连。那么，汇合顶点呢？从表面上看，它们似乎更难以消除——因为，对称地，它们各自需要借助一个位置更低的顶点，才能引入一条对角线。然而，位于扫描线下面的那些部分尚未访问到，所以在遇到一个汇合顶点时，并不能参照上面的方法构造出一条对角线。幸运的是，该问题并不像乍看起来那样困难。试考虑扫描线刚刚触及某一汇合顶点 v_i 的时

1 意为：将分裂顶点（汇合顶点）与位于其上（下）方的另一个顶点相联接，形成一条对角线——译者。

2 y 坐标越大，优先级越高——译者。

3 其长度可能为零——译者。

刻。沿着扫描线的方向，令 e_j（e_k）为位于 v_i 左（右）侧、与之相邻的边。请注意以下事实：在到达 v_i 的时候，它也就成为了 e_j 的新助手。这样，就可以从介于 e_j 和 e_k 之间、位于当前扫描线下方的所有顶点中，选出其中的最高者，然后将 v_i 与之相连。这个过程，与处理分裂顶点的情况正好相反——在那里，我们是在从介于 e_j 和 e_k 之间、位于当前扫描线上方的所有顶点中，选出其中的最低者，然后将 v_i 与之相联。这也不值得奇怪——实际上，只要将上和下颠倒过来，汇合顶点也就相当于分裂顶点。当然，在扫描线触及 v_i 那一时刻，我们还不知道哪个才是位于扫描线下方的最高顶点。然而我们马上就会看到，这并不难判断出来。如图 3-16 所示，此后将遇到某个顶点 v_m，它将取代 v_i 的地位，成为 e_j 的新助手——这时，v_m 就是我们所寻找的顶点。因此，在每次更换某条边的助手时，都要通过检查以确认（被替换的）先前的助手是否为一个汇合顶点。如果是，就在新、老助手之间引入一条对角线。若新助手是一个分裂顶点，则这条对角线本来就需要被加入进来，以消除这一分裂顶点。若同时老助手是一个汇合顶点，则这条对角线将把一个分裂顶点和一个汇合顶点同时消除掉。还有一种可能：在扫描线越过 v_i 之后，e_j 的助手不再会被更换——在这种情况下，可以将 v_i 与 e_j 的下端点连接起来。

按照上述方法，还需要找出居于每个顶点左侧、与之紧邻的那条边。为此，可使用一棵动态二分查找树 $\mathcal{T}$，将 $\mathcal{P}$ 中与当前扫描线相交的所有边存放在该树的叶子中。$\mathcal{T}$ 中所有叶子从左到右的次序，对应于这些边从左到右的次序。既然我们只关心在左侧与各分裂顶点或汇合顶点紧邻的边，故在 $\mathcal{T}$ 中，只需存放 $\mathcal{P}$ 的内部（在局部）位于其右侧的那些边[1]。对 $\mathcal{T}$ 中的每一条边，我们都记录其对应的助手。树 $\mathcal{T}$ 以及所存储的各边的助手，构成了扫描线算法的状态（Status）。随着扫描线的推进，状态会相应地变化：有些边可能开始与扫描线相交，原来与扫描线相交的一些边可能不再相交，同时某条边原先的助手可能会被新助手替换掉。

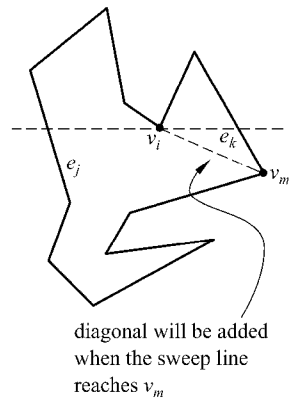

图 3-16 汇合顶点的消除：当扫描线扫过 v_m 时，将输出一条对角线

采用上述算法对 $\mathcal{P}$ 进行划分之后，得到的各个子多边形还必须经过后续的处理。为了能够方便地访问到这些子多边形，需要将由 $\mathcal{P}$ 导出的子区域划分（subdivision）存储起来，并且将所有的对角线加入到双向链接边表 $\mathcal{D}$ 之中。我们假定，$\mathcal{P}$ 原本就是以双向链接边表（doubly-connected edge list）形式给出的；否则——比如，仅表示为所有顶点的一个逆时针列表——就需要首先为 $\mathcal{P}$ 构造出一个双向链接边表。随后，为每个分裂顶点和汇合顶点引入的对角线，都必须加入到这个双向链接边表之中。为了访问该双向链接边表，需要将状态结构与双向链接边表中对应的各边通过指针链接起来。借助于指针的操作，可以在常数时间内引入一条对角线。这样，就得到了如下的主算法。

1 亦即所谓的“左边”——译者。