arXiv:2305.01467v1 [cs.CG] 2 May 2023

将多边形边界上的每个点折叠到一个点

Nattawut Phetmak* Jittat Fakcharoenphol†

摘要

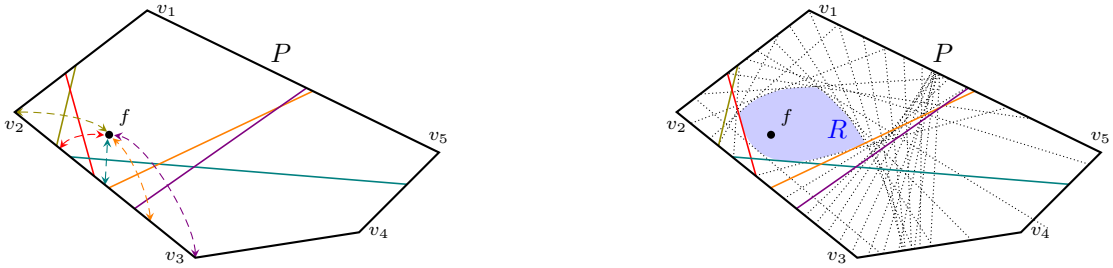
我们研究计算折纸中的一个问题。给定一张凸多边形P形状的纸，以及位于其内部的一个点 f，将 P 边界上的每个点折叠到 f，并计算一个免受折叠影响的区域，即没有折痕的区域。这个问题扩展了 Akitaya、Ballinger、Demaine、Hull 和 Schmidt [在 CCCG'21] 提出的问题；他们的问题只将多边形的角点折叠起来。为找到该区域，我们证明了抛物线界定区域的交点的结构性质，并利用这些性质设计出一种线性时间算法。我们还证明了一项关于安全区域复杂度的结构性结果，该复杂度随点 f 的位置而变化；也就是说，可以利用多边形 P的直骨架确定安全区域中弧的数量。

1 引言

纸张折叠蕴含着丰富的计算几何问题，并有许多现实应用 [20]。这一领域通常称为计算折纸或折纸数学 [10, 18], ，研究可行性问题和结构性问题 [3, 4, 1] ，旨在阐明物理结构 / 问题与数学几何对象之间的联系（例如参见 [9, 11]）。利用直尺与圆规进行几何作图，展现了代数与几何之间优雅的联系；而纸张折叠可视为加入了额外操作的几何作图，或许能揭示值得研究的优美结构性质（例如参见 [10, 17]）。

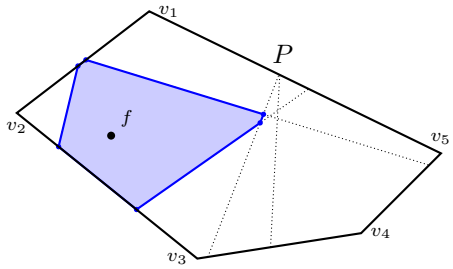
Akitaya、Ballinger、Demaine、Hull 和 Schmidt [3] 此前研究了凸纸片 P上的两个折叠问题。给定其中的一个查询点 p，他们的第一个问题最初由 Haga [14, 15, 16], 提出，称为点到点问题：将 P的每个角点折叠并展开到 p后，求出一个包含 p 且由折痕围成的区域。第二个问题称为线到线问题，其输入是位于 P内的一条直线 ℓ；目标是将 P 的每条边折叠并展开到 ℓ后，求出一个包含 ℓ 且由折痕围成的区域。尽管这两个问题结构相似，他们发现其结果却大相径庭。也就是说，第一个问题得到的区域类似于一个沃罗诺伊单元 [6] ，其种子点为内部点和角点；而第二个问题得到的区域则取决于纸片的直骨架。

*电子邮箱：nattawut.p@ku.th。泰国曼谷泰国农业大学计算机工程系。
†电子邮箱：jittat@gmail.com。泰国曼谷泰国农业大学计算机工程系。获泰国研究基金资助，项目编号 RSA-6180074。

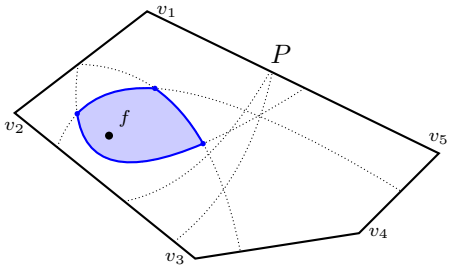


(a) 从一条边得到的示例折痕 $\overline{v_2v_3}$　　（b）安全区域 R

图1：将多边形边界上的每个点折叠到一个点的问题 P f



(a) 在 [3]中进行角折叠



(b) 本文中的边界折叠

图2：问题设定的比较

我们研究了一个风格相似的折叠问题变体：给定纸张 P 内部的一个查询点 f，我们希望在将纸张 δP 边界上的每个点折叠并展开到 f后，求出一个包含 f 且由折痕围成的区域。我们将所得区域称为安全区域 R，因为其中的每个点 $p \in R$ 都不会受到这一折叠过程的影响。我们的贡献包括分析 R的形状、提出一种高效算法来求出 R，以及研究安全区域相对于任意查询点 f的复杂度。

我们首先证明，尽管我们会折叠 δP上的无穷多个点，但结果区域 R 可以用有限方式描述。众所周知，将一条直线上的多个点折叠并展开到一个点上，会形成一条抛物线的包络线。因此，我们将 R 的每条“边”视为抛物线弧，而非传统的直线段。相关分析见第2节。

基于对安全区域性质的分析，给定一张纸 P，并将其 n 个角点按逆时针顺序排列成有序列表，我们给出一种线性时间算法来求出区域 R。我们的算法与求凸包的 Graham 扫描算法类似。具体而言，我们每次将 P 的一条边作为抛物线弧加入，同时维护一个用于约束区域 R的循环不变式。在每次迭代中，我们可能会删除一些先前加入的抛物线弧。关键在于，被删除的抛物线弧必定是最接近新加入的抛物线区域的那一条。因此，我们可以摊销删除操作的成本，从而使整体算法达到线性时间。第3节将详细介绍该算法。

最后，给定任意一个潜在的查询点 f，我们计算区域 R中抛物线弧的精确数量。结果表明，这些弧的数量取决于一组内切圆，其中每个圆都以 P 的直骨架上的一个节点为圆心。第 4 节将专门探讨这一性质。

我们希望这个问题能丰富折叠与展开折纸问题这一类问题。不过，它也可以成为连接 [3], 中前两个问题的桥梁，因为我们的问题陈述与第一个问题相似，而我们的结果与第二个问题相似。

2 预备知识

本节将对问题进行正式定义。给定一个多边形 P，其顶点按逆时针方向排列成有序列表 $V(P) = [v_1, v_2, \ldots, v_n]$。我们将该列表视为循环列表。给定两个点 a 和 b，我们将端点为 a 和 b 的线段称为线段 $\overline{ab}$。等价地，多边形 P 也可以表示为边的列表 $E(P) = [e_1, e_2, \ldots, e_n]$，其中 $e_i = \overline{v_i v_{i+1}}$ 是连接 v_i 和 v_{i+1} 的线段。我们将其边界记为 δP，即 $\delta P = \bigcup E(P)$。

我们还给定一个严格位于 P内部的点 f。考虑一个点 $u \in \delta P$，将 u 折叠到 f 上，会形成一条直线 L，称为折痕线。该直线经过线段$\overline{uf}$ 的中点，并且垂直于 $\overline{uf}$。我们关注将$e_i = \overline{v_i v_{i+1}}$ 上的每个点折叠到 f 上时所得到的折痕线集合。这族折痕线的包络线是一条抛物线，其焦点为f，准线则是将线段 e_i 延长成直线后得到的直线。我们将这条抛物线称为p_i。见图3。

过点 u作垂直于准线的直线 L'，求 $u' = L \cap L'$。我们有 $|\overline{u'f}| = |\overline{u'u}|$，这确实符合抛物线的定义。换言之，$u'$ 是抛物线曲线上的一点。此外，对于满足 $|\overline{vf}| < |\overline{vu}|$ 的 $v \in L'$，点 v 不会受到从这条准线上的其他点逐一进行折叠与展开所产生的其他折痕的影响。

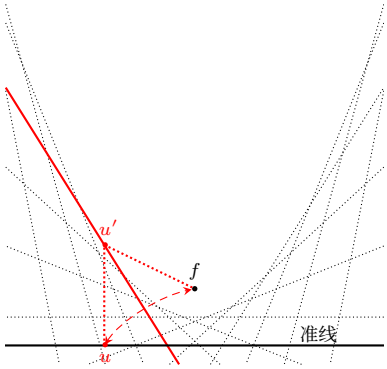


图3：多次将直线上的点折叠到一个固定点，再展开，从而形成抛物线的包络线

由于 p_i 将平面分成两个抛物线半平面，我们关注的是包含 f的区域。我们将包含 f的半平面定义为半空间 $H(p_i)$。更形式化地说， $H(p_i)$包含所有满足$|\overline{vf}| \leq |\overline{vu}|$的点 v，其中 u 是 v 在 e_i的直线延长线上的正交投影。

我们的目标是找到

$$R = \bigcap_{i=1}^{n} H(p_i),$$

定义为安全区域。当关注点 f 时，我们有时将相对于点 f 的安全区域称为 R_f。

We 在第3节中描述该算法。本节稍后将陈述相关的几何事实。

2.1 抛物线作为圆锥曲线的投影

抛物线可以看作圆锥曲线，即圆锥表面上的一条曲线与一个倾斜角度等于圆锥倾角的截平面相交所形成的曲线。从解析角度来看，我们考虑欧几里得空间 $\mathbb{R}^3$。圆锥是满足以下方程的曲面

$$x^2 + y^2 - z^2 = 0$$

圆锥的顶点位于原点。截平面可以用以下方程定义

$$x\cos\theta + y\sin\theta - z = r,$$

其中， r 是从圆锥顶点到截平面最近点在 xy 平面上的距离， θ 是该点在 xy 平面上的方向角。根据此定义，我们得到一条位于倾斜截平面上的抛物线。见图4a和图4b。

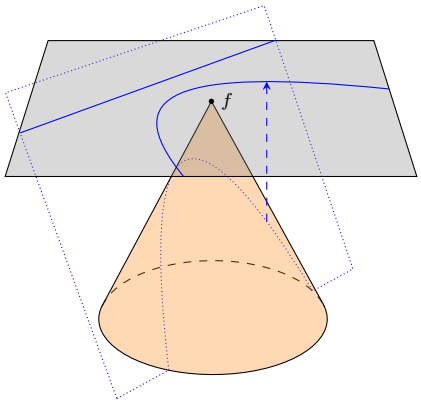
投影抛物线是倾斜抛物线在 xy 平面上的正交投影，它本身也是一条抛物线。投影抛物线的焦点是圆锥的顶点，其准线是截平面与 xy 平面的交线。这种投影视角在 [12] 的末尾曾被简要提及。本文将投影抛物线简称为抛物线。

如果通过旋转和平移 xy 平面，抛物线 p 具有 $y = tx^2$ 的解析形式，其中 $t > 0$，我们就称其为正立形式。

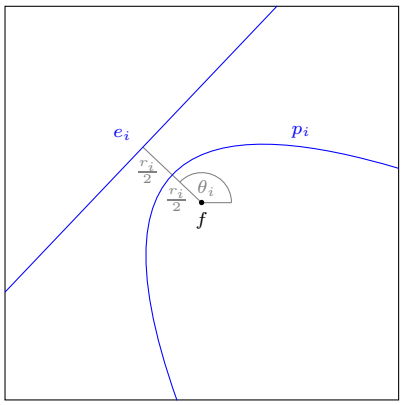
2.2 两条抛物线

我们将具有相同焦点、但准线的方向角不必相同的一族抛物线定义为半共焦抛物线。由此可知，半共焦抛物线是由多个截平面截取同一个圆锥所得到的投影抛物线。我们陈述关于两条抛物线交点的两个重要事实；从圆锥曲线解释的角度来看，这些事实显而易见。

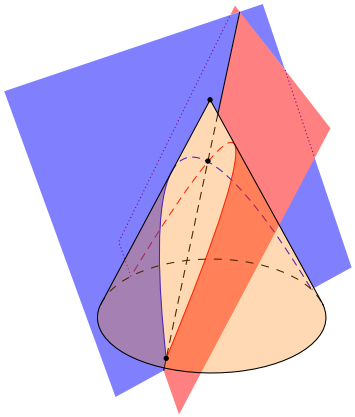
Lemma 2.1. 两条半共焦抛物线不相交，当且仅当它们的方向角相同 $e.$



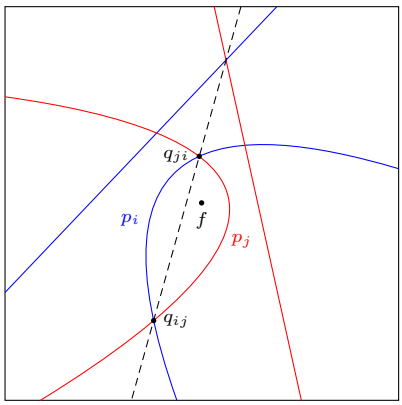
(a) 将倾斜抛物线映射到 xy-平面



(b) 抛物线在 xy-平面上的投影



(c) 两个截平面



(d) 角平分线与两个交点

图4：将抛物线解释为圆锥曲线的投影

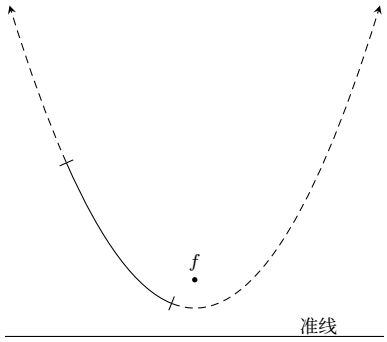


图5：抛物线的弧分解

证明。方向角相同的两条抛物线由圆锥曲线视角中的两个平行截平面产生，而这两个截平面永不相交。□

引理 2.2。如果两条半共焦抛物线相交，则它们会在两个点处相交，并且这两个点还位于其准线的角平分线上。

证明。采用圆锥曲线解释。它们在圆锥面上的曲线也必定位于各自的截平面上，而这两个平面的交线是一条直线。这条直线恰好穿过圆锥两次。见图4c和4d。□

由引理 2.2 可知，两条相交的半共焦抛物线 p_i、p_j 产生一个安全区域$R = H(p_i) \cap H(p_j)$，该区域是一个有界凸区域。此外，由于我们只处理准线来自凸多边形边界边的抛物线，因此不相交的情况永远不会发生。

给定两条抛物线 p_i 和 p_j，我们可以通过解析方法在 $O(1)$时间内按如下方式求出它们的交点。我们将求两条抛物线交点的问题转化为求一条抛物线与一条直线交点的问题。不失一般性，我们考虑正立形式的 p_i 。根据引理 2.2，我们找到一条角平分线 b ，使其平分边 e_i 和e_j之间的内角。然后，我们用二次方程求出 p_i 和 b 的交点。

我们特别说明安全区域 R的结构。

考虑正立形式的每条抛物线 p_i 。我们可以利用两个交点将这条抛物线划分为 3 段弧：左弧、中弧和右弧。其中，左弧对应于向 $-\infty$ 无界延伸的半抛物线，右弧对应于向$+\infty$无界延伸的半抛物线，而中弧位于两个交点之间。（见图 5。）按照这一记号，我们注意到， p_i 的左弧仅与 p_j 相交一次，该交点也是它与 p_j的右弧的交点，反之亦然。

在分析中，设有抛物线 $p_1, p_2, \ldots, p_n$，我们将抛物线 p_i 和 p_j 之间的两个交点分别称为 q_{ij} 和 q_{ji}。为了区分这两个点，设想沿着 $H(p_i) \cap H(p_j)$的边界逆时针行进，会依次看到一段 p_i、交点 q_{ij}、一段 p_j ，然后是交点 q_{ji}。这些点的逆时针定义对于引理 3.1 的证明至关重要。见图4d。

2.3 多条抛物线

本节我们分析 k 半共焦抛物线的交集结构，推广上一节的结果。

设 $p_1, p_2, \ldots, p_k$ 为方向角各不相同的 k 条半共焦抛物线。我们将考察这些抛物线的安全区域，并证明以下引理。

引理 2.3。每条抛物线至多与安全区域的一段弧相切。

证明。我们对 k 进行归纳法。当 $k = 2$ 时，由引理 2.2 可得结论。考虑 k 条抛物线。令 $R' = \bigcap_{i=1}^{k-1} H(p_i)$。根据归纳假设，每条抛物线 p_1、p_2、.. .、p_{k-1} 至多与 R' 的一段弧相切。我们

考虑 $R = R' \cap H(p_k)$，我们将证明 p_k 至多与 R 的一段弧相切。若 $R' \subseteq H(p_k)$，则 $R = R'$ 与 p_k 不相切于 R；因此引理成立。接下来假设 $R' \not\subseteq H(p_k)$。

显然， p_k 与 $R = R' \cap H(p_k)$的一段弧相切。将这段弧记为 a_k。 a_k 的每个端点都属于 R'的某一段弧。

分为两种情况。

情况 1： a_k 的两个端点都属于 p_i的同一段弧。在这种情况下， p_k 与 R' 的一段弧恰好相交两次。因此，安全区域 R 是两条抛物线的交集，即$R = H(p_i) \cap H(p_k)$。由此，根据引理 2.2 可得该引理。

情况 2： a_k 的一个端点属于 p_i ，另一个端点属于 p_j。我们证明，在这种情况下， p_k 恰好与两段弧相交，从而得出该引理。我们先考虑 p_i ，也就是说，考察交集$R_i = H(p_k) \cap H(p_i)$。设 q_i 为 p_i 与 p_k 的交点。我们可以将 p_k 划分为三段弧；设 b_k 为从 q_i出发的无界弧。根据引理 2.2，我们知道 b_k 只与 R_i 相交一次。由于 $R \subseteq R_i$，因此 b_k 在 q_i处与 R 至多相交一次。对p_j，我们采用相同的论证。因此，正如所声称的， p_k 仅与 R' 相交两次。 □

3 算法

我们用于求安全区域的算法与用于求凸包的 Graham 扫描算法类似。 [13] 我们在算法 1 中用伪代码简要描述了该算法。本节后续部分将解释该算法并证明其正确性。

算法 1：我们的安全区域求解算法

```
函数 SafeRegion(P : Polygon, f : Point)
    [e_1, e_2, e_3, ..., e_n] ← E(P)
    p_1 ← Parabola(f, e_1)
    p_2 ← Parabola(f, e_2)
    R ← DoublyLinkedList(Arc(p_1, q_21, q_12) Arc(p_2, q_12, q_21))
    对 i ∈ [3, 4, 5, ..., n] 执行
        p_i ← Parabola(f, e_i)
        (p_H, ℓ_H, r_H) ← Head(R)
        (p_T, ℓ_T, r_T) ← Tail(R)
        如果 q_iH ∈ LeftArc(p_H, ℓ_H) ∧ q_Ti ∈ RightArc(p_T, r_T)，则
            继续                                          // 跳过本次迭代
        当 q_iH ∈ RightArc(p_H, r_H) 时
            R ← RemoveHead(R)
            (p_H, ℓ_H, r_H) ← Head(R)                     // 也重新计算 q_iH
        当 q_Ti ∈ LeftArc(p_T, ℓ_T) 时
            R ← RemoveTail(R)
            (p_T, ℓ_T, ) ← Tail(R)                        // 也重新计算 q_Ti
        R ← UpdateHead(R, Arc(p_H, q_iH, r_H))
        R ← UpdateTail(R, Arc(p_T, ℓ_T, q_Ti))
        R ← AppendTail(R, Arc(p_i, q_Ti, q_iH))
    返回 R
```

由于我们的目标是找到一个安全区域

$$R = \bigcap_{i=1}^{n} H(p_i),$$

我们遍历抛物线 p_i，生成一个部分解 R_i，使得

$$R_i = \bigcap_{j=1}^{i} H(p_j),$$

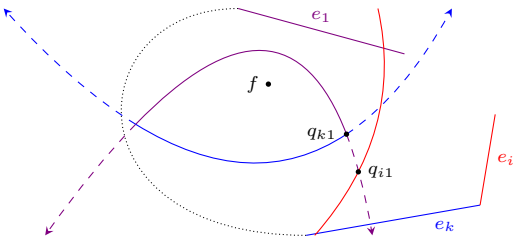


图 6：引理 3.1 的证明

即，R_i 是前 i 条抛物线的安全区域。我们将 R_i 维护为一个抛物线的循环列表 ar cs

$$a_1, a_2, \ldots, a_k,$$

其中，每条弧 a_j 都是一个三元组（p，ℓ，r），它保存了对抛物线 $a_j.p$、其左端点$a_j.\ell$和右端点 $a_j.r$ 的引用。我们注意到，使用这种表示法，$a_j.r = a_{j+1}.\ell$ 对于 $1 \le j < k$，且 $a_k.r = a_1.\ell$。我们还注意到，使用第 2.2 节中定义的记号，$a_j.r$ 是 $q_{j(j+1)}$，且 $a_j.\ell$ 是$q_{(j-1)j}$。

最初，我们从 $R_2 = H（p_1）\cap H（p_2）$开始。我们将部分安全区域编码为一个弧的有序列表 $A（R_2）= [a_1, a_2]$，其中

$$a_1 = (p_1, q_{21}, q_{12})$$
$$a_2 = (p_2, q_{12}, q_{21})$$

对于每次迭代 $i > 2$，我们考虑将 $H（p_i）$添加到 R_{i-1}，以生成 $R_i = R_{i-1} \cap H（p_i）$。共有 3 种情况：

- 情况 1：p_i 不会改变区域，即 $R_{i-1} \cap H(p_i) = R_{i-1}$，因此我们可以舍弃 p_i，
- 情况 2：p_i 裁剪区域，即 R_{i-1} 中的所有抛物线弧仍构成 $R_{i-1} \cap H(p_i)$ 的边界，或者
- 情况 3：p_i 遮蔽区域中的其他抛物线，即某条抛物线弧完全位于 $R_{i-1} \cap H(p_i)$ 外部。

为了区分这些情况，我们的基本步骤是测试某个点是否位于 $H(p_i)$ 中。抛物线的逆时针排序确保 p_i 会影响两组弧序列，即按顺时针方向

$$a_k, a_{k-1}, a_{k-2}, \ldots,$$

称为 p_i 左侧的邻居，并按逆时针方向，

$$a_1, a_2, a_3, \ldots,$$

称为 p_i 右侧的邻居，

我们首先考虑点 $q_{k1} = a_k.r$（它也是 $a_1.\ell$）。下面的引理 3.1 保证，当 $q_{k1} \in H(p_i)$ 时，我们属于情况 1。参见图 6。

否则，R_{i-1} 的某些部分位于抛物线 p_i 下方。接下来，我们依次考虑这些点

$$a_{k-1}.r, a_{k-2}.r, \ldots,$$

从 p_i 左侧的邻居开始，并找出满足 $a_j \cap H(p_i) \neq \emptyset$ 的最大索引 j。在这种情况下，抛物线 $a_k.p$、$a_{k-1}.p$、.....、$a_{j+1}.p$ 被 p_i 遮挡。

我们也以类似的方式处理 p_i 右侧的邻居，找出满足 $a_{j'} \cap H(p_i) \neq \emptyset$ 的最小索引 j'，并得到被遮挡弧的序列 a_1、a_2、.....、$a_{j'-1}$。我们注意到，当只有一条弧在 p_i 遮挡后仍然保留时，可能出现 $j = j'$ 的情况。

为构造 R_i，我们舍弃被遮挡弧，添加一条表示 p_i 的新弧 a_i，并计算

- 左交点 $a_i.\ell = a_j.r$ 即 p_i 与 $a_j.p$的交点，并且

• 右交点 $a_i.r = a_{j'}.\ell$ 即 p_i 与 $a_{j'}.p$的交点。

最后，我们对 R_i 中的弧重新编号。简要地说，这一过程可以看作一种“双头”Graham 扫描。

下面的引理表明这一过程是正确的。

引理 3.1. 若 $q_{k1} \in H(p_i)$，则 $R_{i-1} \subseteq H(p_i)$ 且 $R_i = R_{i-1}$。

*Pro*of。由于 $R_{i-1} \subseteq H(p_1) \cap H(p_k)$，我们的目标是在这种情况下证明 $H(p_1) \cap H(p_k) \subseteq H(p_i)$ e.

考虑 p_1 与 p_k 的交集。回想一下，两个交点 q_{k1} 和 q_{1k} 将两条抛物线都分成左弧、中弧和右弧。我们将它们分别称为 a_k^ℓ、a_k^c、a_k^r 和 a_1^ℓ、a_1^c、a_1^r。

现在考虑 p_1 与 p_i 的交集。我们要证明，q_{i1}，即 p_i 与 p_1 的交点，位于 p_1 的左弧 a_1^ℓ 上。为此，我们先旋转平面，使 p_i 呈正立形式。然后找到一个区域 $r = H(p_1) \cap H(p_i)$。由于 $q_{k1} \in p_1$，且 $q_{k1} \in H(p_i)$，因此 q_{k1} 位于 $H(p_1) \cap H(p_i) = r$ 的边界上。同样，由于 $q_{k1} \in p_1$，且也位于 r 的边界上，从 r 的边界上的 q_{k1} 出发，相对于 f 顺时针方向行进将到达 q_{i1}；根据 q_{i1} 的定义，结论得证。

通过相同的论证，我们可以证明 q_{ki} 位于 p_k的右弧 a_k^r 中。

利用 q_{i1} 和 q_{ki}，我们将 p_i 划分为 3 条弧：a_1、a_2和 a_3，使得 a_1 是一条以 q_{i1}为端点的无界曲线，a_2 是一个以 q_{i1} 和 q_{ki} 为端点的有界部分，最后 a_3 是一条以 q_{ki} 为端点的无界曲线。（见图 6）

根据引理 2.2 中的结构，我们知道 $a_1 \cup a_2$ 仅在 $q_{ki} \in a_k^r$处与 p_k 相交。因此，p_i不与 $a_k^c \cup a_k^\ell$相交。

此外，我们知道 $a_2 \cup a_3$ 仅在 $q_{i1} \in a$处与 p_1 相交 ℓ，这意味着 p_i 不与 a相交 $_1^r \cup a^c$。我们可以得出结论，p_i 不与 R_{i-1}相交，因为 R_{i-1} 位于无界曲线 $a_k^c \cup a_k^\ell$ 和 $a_1^r \cup a_1^c$之间。

□

我们还有一个简单的逆否命题。

推论 3.1.1. 如果 $R_{i-1} \not\subseteq H(p_i)$，那么 $q_{k1} \notin H(p_i)$。

设 $\hat{R} = \bigcap_{j=1}^{i} H(p_j)$ 为正确更新后的解，我们想证明上文构造的 R_i ˆ等于 R。我们指出，第 i 条抛物线 p_i 对应于在所有其他对 R_{i-1}有贡献的边之后按逆时针方向排列的边 e_i。

引理 3.2. 如果 $R_{i-1} \not\subseteq H(p_i)$，则 R_{i-1}的边界上不属于 ˆR 的弧构成一个连续序列

$$a_j, a_{j+1}, \ldots, a_k, a_1, a_2, \ldots, a_{j'}.$$

ˆ 证明。引理 2.3 保证，若 p_i 与 R_{i-1}的边界相交，则 p_i 与 R的一段弧至多相切一次，该相交结果为 $H(p_i)$ 和 R_{i-1}。这意味着 R_{i-1} 中不属于 的弧构成一个（圆形）连续序列。为说明这一点，反设并注意，在这种情况下，p_i ˆ 会与 R的多段弧相交。

我们的过程找出从 q_{k1} 开始的连续序列，即 p_k 和 p_1 的交点，然后遍历其他连续的点。因此，只要起始点正确，该过程就是正确的，ˆ也就是说，我们从 R 外部的某个交点开始。确实如此，因为推论 3.1.1 保证，当 $R_{i-1} \not\subseteq H(p_i)$ 时，q_{k1} 位于 ˆR 外部。

□

最后，我们给出主要的正确性定理。

ˆ 定理 3.3。我们的更新过程是正确的，即 $R_{i+1} = R$，并且该算法在线性时间内计算出安全区域。

证明。关于更新过程，我们需要处理 3 种可能的情况。引理 3.1 确保了情况 1 的判定条件是正确的。在其他情况下，引理 3.2 表明删除弧的过程是正确的。根据对 n 的归纳法，该算法因此能够生成所需的安全区域。

为了分析运行时间，我们首先注意到，除两个内部 while 循环外，对于每个 i，算法都在 $O(1)$ 时间内运行。为计算内部 while 循环的运行时间，注意到循环的每次迭代都会从列表中移除一条抛物线。由于列表中最多插入 n 条抛物线，因此删除操作最多进行 n 次，这意味着循环的总运行时间为 $O(n)$。

□



(a) 查询点为 f_2 和 f_3的安全区域　　(b) 事件圆决定弧的数量

图 7：含有 2 条和 3 条抛物线弧的安全区域

4 安全区域的弧数

在本节中，我们研究安全区域边界的复杂度；换句话说，我们计算安全区域中弧的确切数量。由前面的章节可知，安全区域的一条边是以点 f 为焦点的抛物线弧。我们还看到，P 的某些边可能不会对最终的安全区域作出贡献，也就是说，与这些边相关联的抛物线不会接触安全区域。自然会提出这样的问题：安全区域有多少条弧。

假设多边形 P 固定，则安全区域的弧数取决于焦点 f。我们用 R_f 明确表示以点 f 为焦点的安全区域。与 [3], 一样，本节分析安全区域 R_f的弧数，即 $|A(R_f)|$，其中 $A(R_f)$是 R_f的弧集。图 7(a) 展示了两个安全区域：R_{f_2} 的查询点为 f_2，R_{f_3} 的查询点为 f_3。

Akitaya 等人 [3] 研究了同一问题在以下情形下的情况：将 P 的每条边折叠到一条直线上。他们证明，P 的直骨架在确定所得区域的边数方面发挥着重要作用。我们的情形也同样如此。例如，图 7(b) 展示了一个内切圆 C，它可以利用直骨架和先前展示的两个安全区域确定。我们指出，$f_2 \notin C$，但 $f_3 \in C$。

我们首先定义与直骨架和事件圆相关的有用记号。多边形 P的直骨架 [2]，记作 $S(P)$，是 P 的一个子集，满足：对于每个点 $u \in S(P)$，在 δP 上至少存在两个点，它们到 u的距离相同。更直观地说，我们可以将直骨架视为线段的沃罗诺伊图，其中每个生成元都是多边形的一条边。直骨架$S(P)$将 P 划分为若干区域，称为面。因此，在 Voronoi 解释下，每个面恰好由一条多边形边界定，因为 $S(P)$的其他边并不构成其边界。我们将由多边形边 e_i 界定的面称为面 F_i。我们还注意到，面也是一个凸多边形。示意图见图 8。

直骨架可以看作一棵树，其中每个非叶节点保证在 δP 上至少存在三个等距点。$S(P)$ 的非叶节点称为事件点。以事件点为圆心、与多边形最近的边相切的圆称为 $S(P)$ 的事件圆。令 $\mathcal{C}$ 为 $S(P)$ 的所有事件圆的集合，并令 $\mathcal{C}_e \subseteq \mathcal{C}$ 为与边 e 相切的事件圆集合。我们还用 int(C) 表示圆 C 的内部，即集合 $\{(x, y) : (x - x_0)^2 + (y - y_0)^2 < r^2\}$，其中对于圆心为 (x_0, y_0)、半径为 r 的圆。

本节旨在说明，在给定多边形 P 的情况下，$|A(R_f)|$ 的结构由 P 的直骨架 $S(P)$ 的事件圆所决定。

我们首先分析安全区域与骨架面相交的情形。以下引理直接由直骨架的 Voronoi 解释得到。

引理 4.1. 对于每个带有事件点 c 的事件圆 C，c 与面 F_i 相邻，当且仅当其对应的多边形边 e_i 与 C 相切。

以下两个引理给出了本节主要定理的基本性质。

对于特定焦点 f，int(R_f) 与和边 e_i 相邻的骨架面 F_i 相交，当且仅当相应的抛物线 p_i 是 R_f 的弧的一部分。

证明。($\Rightarrow$) 假设 int(R_f) 与 F_i 相交。由于 F_i 由一条多边形边界定，且 R_f 包含于 P 中，我们知道 R_f 的边界上有一些部分与 F_i 相交。考虑 F_i 内 R_f 边界上的任意一点 u。显然，u 必定位于某条抛物线的弧上，即我们有

$$\min_j |\overline{uu_j}| = |\overline{uf}|,$$

其中， u_j 是 u 在 e_j 上的正交投影。由于 F_i 中的所有点到 e_i 的距离都小于到其他边的距离，因此上述项的最小值在 u_i 处取得；于是， u 也必定位于 p_i 上，即 p_i 是 R_f 的弧的一部分。

(⇐) 我们通过反证法证明。假设 int(R_f) 与 F_i 不相交，但 p_i 是 R_f 边界的一部分。考虑边界上的任意点 $u \in p_i$。由于 int(R_f) 与 F_i 不相交， u 严格位于某个面 F_j 内。在这种情况下，我们有

$$|\overline{uf}| = |\overline{uu_i}| > |\overline{uu_j}|,$$

其中， u_i 和 u_j 分别是将 u 正交投影到 e_i 和 e_j 上所得的投影点，这意味着 $u \notin H(p_j)$，矛盾。

□

引理 4.3. 对于带有事件点 c 的 $C \in \mathcal{C}$ ，安全区域 R_f 严格包含事件点 c，即，$c \in$ int(R_f) 当且仅当 $f \in$ int(C)。

证明。令 r 为事件圆 C 的半径。将 c 正交投影到每条边 e_i 的直线延长线上，并将所得投影点命名为 c_i。

(⇐) 假设 $f \in$ int(C)，即 $|\overline{cf}| < r$。我们证明，对于与多边形边 e_i 相关联的每条抛物线 p_i ，都有 $r \in H(p_i)$。当 r 严格比其他每条边 e_i 更靠近 f 时，情况便是如此。考虑每条与 C 相切的边 e_i ，我们有 $|\overline{cf}| < r = |\overline{cc_i}|$。对于不与 C 相切的边 e_i ，有 $|\overline{cc_i}| > r$；因此， $|\overline{cf}| < r < |\overline{cc_i}|$。所以， c 位于安全区域 R_f 内。

(⇒) 假设 $f \notin$ int(C)。在这种情况下，我们有 $|\overline{cf}| \geq r$。由于 C 是一个事件圆，存在边 e_i 与 C相切。对于这条特定的边，我们有 $|\overline{cc_i}| = r$。因此， $|\overline{cf}| \geq r = |\overline{cc_i}|$，而且 c 并未严格包含在安全区域 R_f内。

□

下述定理给出了 R_f 的边界弧数量与包含 f的事件圆数量之间的函数关系。

定理 4.4. 若 f 严格位于某个事件圆内部，即对于某个 $C \in \mathcal{C}$，有 $f \in$int(C)，则

$$|A(R_f)| = |\{e_i \in E(P) : \textit{there exists } C \in \mathcal{C}_{e_i} \textit{ s.t. } f \in \mathrm{int}(C)\}|$$

否则， $|A(R_f)| = 2$。

证明。首先假设对于某个事件圆 C，有 $f \in$int(C)。考虑每个带有事件点 c的事件圆 C ，使得 $f \in$ int(C)。由引理 4.3 可知， $c \in$ int(R_f)。这也意味着，int(R_f)与每个邻接事件点 c的面 F_i 相交。引理 4.1 保证这些面 F_i对应于与 C相切的边 e_i；这些边的集合为 e_i ，且满足 $C \in \mathcal{C}_{e_i}$。由于引理 4.2 保证，对于每个与 R_f 相交且邻接边 e_i 的面 F_i ，抛物线 p_i 会作为 R_f的一段弧出现，因此对于 C的每条切边 e_i ，其抛物线 p_i 都会作为 R_f中的一段弧出现。在这种情况下，只需取所有事件圆 $C \in \mathcal{C}$ 的边界边的并集，其中 f 严格位于其内部，即可得证。

另一方面，如果 f 没有严格包含在任何事件圆 $C \in \mathcal{C}$内，根据引理 2.2，安全区域必定与两条抛物线相切。

□

图 9 展示了定理 4.4 的一个应用。

另一种理解方式是从圆锥曲线解释的角度来看定理 4.4。输入多边形 P 会产生 n 个截平面，投影到 xy 平面后形成直骨架及其对应的面；点 f 则表示为一个圆锥，其顶点位于 f。

本节的结构性结果还提供了另一种在线性时间内寻找安全区域的算法：首先使用 [8], , 以 $O(n)$ 的时间复杂度求出直骨架；然后计算事件圆；最后利用这些信息找出构成安全区域各弧的边集。不过，我们认为本节的结果主要有助于从结构上理解这一问题，也可为解决更难的问题，尤其是该问题的非凸情形，提供指导。我们将在第 5.1 节讨论这一点。

5 结论与开放问题

我们给出一种线性时间算法，用于为将凸多边形边界上的每个点 P 折叠到点 $f \in P$ 寻找安全区域。我们还给出了与弧的数量相关的结构性质，涉及

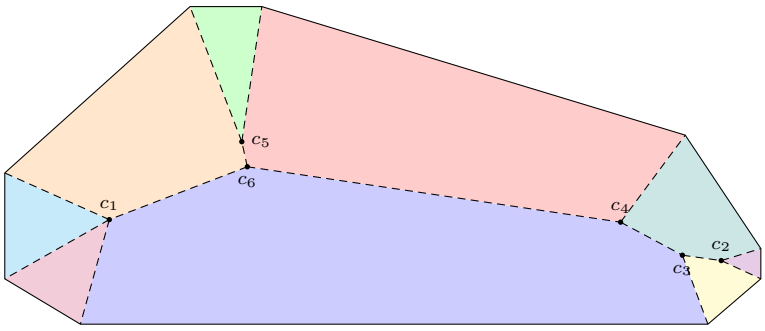


图 8：多边形 P 的骨架，各个面以不同颜色表示

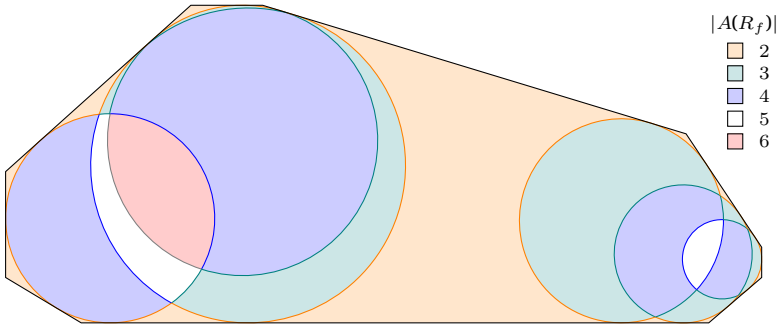


图 9：由直骨架的事件圆确定 $|A(R_f)|$ （与图中相同的多边形 8)

基于直骨架的每个焦点 f 的安全区域。我们指出，直骨架在我们的问题以及折纸设计中的其他问题里都发挥着关键作用，这一点可见于 [19]。未来一个有趣的研究方向是，利用直骨架作为关键工具，探究具有类似结构的问题。

正如引言中所述，我们也希望研究结果能展现 Akitaya 等人提出的两个问题之间有趣的联系 [3]。

5.1 关于非凸多边形的说明

第 4 节的结果为非凸情形提供了一些启示。不过，当前方法仍存在一些问题。处理非凸多边形时，有两个相关概念：直骨架 [2] 和中轴 [7] （另见 [8, 5]）。对于给定的多边形，中轴包含到边界上多个点距离相等的点 δP；而直骨架是一种沃罗诺伊图，其中每个边的直线延长线都是一个站点。二者在凸多边形中相同；然而，在非凸多边形中，其中的中轴包含曲线段。

在我们的应用中，由于我们只能折叠每条多边形边上的点，而不能折叠边的直线延长线上的点，因此考虑中轴是合理的。然而，中轴中的每个面可能由多条多边形边界定，这违背了我们的一个关键假设。我们将把这种方法应用于非凸多边形的研究留作一个重要的开放问题。

6 致谢

作者谨向 JCDCGGG 2022 的组织者和与会者致谢，尤其感谢 Hugo Akitaya 对本文的报告提出有益反馈，并指出了与该问题相关的重要事项。

引用

[1] Zachary Abel、Erik D. Demaine、Martin L. Demaine、David Eppstein、Anna Lubiw 和 Ryuhei Uehara。具有指定角度和边长的平面图平坦折叠。《计算几何杂志》，9(1):74–93，2018年。[2] Oswin Aichholzer 和 Franz Aurenhammer。平面上一类一般多边形图形的直骨架。载于 Jin-Yi Cai 和 Chak Kuen Wong 编，《计算与组合》，第117–126页，柏林、海德堡，1996年。Springer Berlin Heidelberg。[3] Hugo A. Akitaya、Brad Ballinger、Erik D. Demaine、Thomas C. Hull 和 Christiane Schmidt。点到点、线到线的折叠。载于第33届加拿大计算几何会议论文集（CCCG 2021），2021年8月10日至12日，加拿大新斯科舍省哈利法克斯达尔豪斯大学，第271–278页，2021年。[4] Hugo A. Akitaya、Erik D. Demaine 和 Jason S. Ku。简单折叠确实非常困难。《信息处理杂志》，25:580–589，2017年。[5] Dominique Attali、Jean-Daniel Boissonnat 和 Herbert Edelsbrunner。中轴的稳定性与计算——最新进展报告，第109–125页。Springer Berlin Heidelberg，柏林、海德堡，2009年。[6] Franz Aurenhammer、Rolf Klein 和 Der-Tsai Lee。沃罗诺伊图与德劳内三角剖分。世界科学出版社，新加坡，2013年。[7] H. Blum。一种用于提取形状新描述符的变换。载于 W. Wathen-Dunn 编，《语音感知与视觉形式模型》。麻省理工学院出版社，马萨诸塞州剑桥，1967年。[8] F. Chin、J. Snoeyink 和 C. A. Wang。在线性时间内求简单多边形的中轴。《离散与计算几何》，21(3):405–420，1999年。[9] Jana Dambrogio、Amanda Ghassaei、Daniel Starza Smith、Holly Jackson、Martin L. Demaine、Graham Davis、David Mills、Rebekah Ahrendt、Nadine Akkerman、David van der Linden 和 Erik D. Demaine。通过对X射线显微断层成像获得的密封文件进行自动虚拟展开，揭开历史的面纱。《自然-通讯》，12，2021年3月。[10] Erik D. Demaine 和 Joseph O’Rourke。《几何折叠算法：连杆机构、折纸、多面体》。剑桥大学出版社，美国，重印版，2008年。[11] S. Felton、M. Tolley、E. Demaine、D. Rus 和 R. Wood。一种构建自折叠机器的方法。《科学》，345(6197):644–646，2014年。[12] Steven Fortune。一种用于沃罗诺伊图的扫描线算法。载于第二届计算几何年度研讨会论文集，第313–322页，1986年。[13] R.L. Graham。一种确定有限平面点集凸包的高效算法。《信息处理快报》，1(4):132–133，1972年。[14] Kazuo Haga。提议将“origamics”一词用于塑料折纸——无需工作的科学折纸。载于第二届折纸科学与科学折纸国际会议，摘要 A-3，第29–32页，1994年。[15] Kazuo Haga。《数学折纸学：通过纸张折叠进行数学探索》。世界科学出版社，2008年。[16] T. Hull。《折纸计划：探索数学的活动》，第二版。A K Peters/CRC出版社，2013年。[17] Thomas C Hull。用折痕解三次方程：Beloch 和 Lill 的工作。《美国数学月刊》，118(4):307–315，2011年。[18] Thomas C Hull。《折纸几何学：纸张折叠中的数学方法》。剑桥大学出版社，2020年。

[19] Robert J Lang。《用于折纸设计的计算算法》。载于第十二届年度计算几何研讨会论文集，第 98–105 页，1996 年。[20] Robert J. Lang。《计算折纸：从扑翼鸟到太空望远镜》。载于第二十五届年度计算几何研讨会论文集，SCG '09，第 159–162 页，美国纽约州纽约市，2009 年。美国计算机协会。