清华大学

综合论文训练

题目：Folding and 1-Cut 问题的求解与可视化

系　　别：计算机科学与技术系

专　　业：计算机科学与技术

姓　　名：于　海

指导教师：邓俊辉　副教授

2001 年 6 月

摘　要

由于涉及到平面图骨架结构、三角化、凸分解等计算几何中重要的概念和方法，传统纸艺近年来一直是计算几何中的一个研究热点，甚至形成了一个称之为 Computational Origami 的专门研究方向. 在本文中，我们研究了 Computational Origami 中的一个著名问题 Folding and Cutting—的求解与可视化算法及其实现. 我们首先引入一种新的平面图骨架结构——直架，并介绍了它的 $O(n^2 \log n)$ 生成算法. 直架本身是不可折叠的，因此又在直架的基础上引入了垂线折痕. 经过适当的内外折分配，两者共同组成了折痕图. 为了模拟凸多边形的 3 维折叠过程，我们考察了折叠极基的问题，并由此得到了一个递推求解的模拟方法.

关键词

计算几何，纸艺，直架，垂线折痕，折痕图，可视化，极基

Abstract

Since it has deep relations with so many important concepts and methods in the computational geometry literature, such as skeletons of planar graph, triangulation and convex decomposition, traditional Origami has always been a hotspot in recent years, and even forms a special field called Computational Origami. In this paper, we study the solution and visualization algorithms for a famous problem in this field, Folding and Cutting. Firstly, we introduce a novel type of skeleton for planar graphs, Straight Skeleton, and also present an $O(n^2 \log n)$ algorithm for finding the straight skeletons of simple polygons. Straight skeleton by itself is not foldable. Therefore, another type of crease, perpendicular fold, is added to solve the problem. After proper mountain-valley assignment, straight skeleton and perpendicular folds build up the final crease pattern. By investigating the extreme bases, we get a recursive method to simulate the 3D folding process of convex polygons.

Keywords

Computational Geometry, Origami, Straight Skeleton, Perpendicular Folds, Crease Pattern, Visualization, Extreme Base

目　录

一、引　言

纸艺是一种源于日本的民间艺术，可以分为两类，一类称为折纸手工艺(Origami)，另一类称为剪纸手工艺(Kirigami)，两者也统称为纸艺．在 Kirigami 中有一个有趣的 Folding and cutting 问题：任意给定一张纸和一平面图形，如何通过一定方式将给定的纸折叠得到一个平面图案，在该图案上面沿着某一条直线剪一刀，把剩余的一部分展开得到的图形正是给定的图形．另外，Origami 中也有类似的 Folding flat silhouette 问题：任意给定一张纸和一平面图形，如何通过一定方式将给定的纸折叠，使所的图案的侧面(Silhouette)恰好为给定的图形．甚至还有更强的问题：任意给定一张正反具有不同颜色的纸和一个两色的平面图形，如何通过一定方式将给定的纸折叠，使所的图案的侧面恰好为给定的图形，并且两者颜色分布也相吻合．

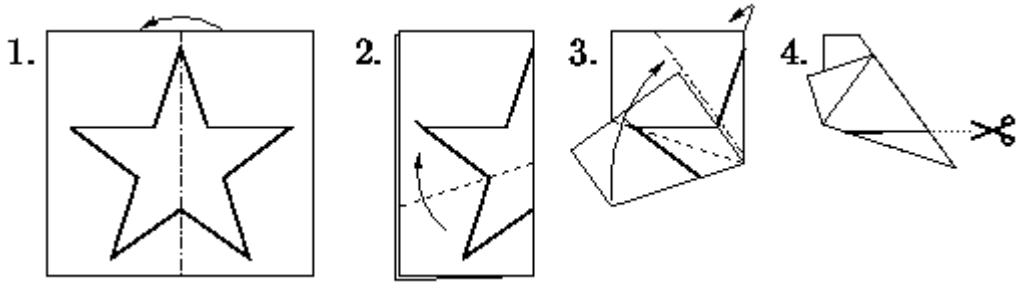


图 1.1. Folding and cutting

另外还有一个与 3D 模型有关的 3D folding & unfolding 问题．一个 3D 模型在表示时一般都使用三角面片的方法，如果对这些三角面片的边经过适当的剪辑，便可将所有的面平布在一个平面上形成以简单多边形，我们可以通过裁剪一张纸得到这样的多边形，从而也就能构建出一个真实的 3D 模型．

以上这些问题统称为Computational Origami．由于Computational Origami涉及到计算几何中多边形骨架结构(Straight Skeleton)、三角化(Triangulation)、凸分解

(Convex Decomposition)等重要的概念和方法，因此成为一个较为热点的研究领域，最近在这方面也有一系列新的研究成果[1,2,3,4,8,9,10].

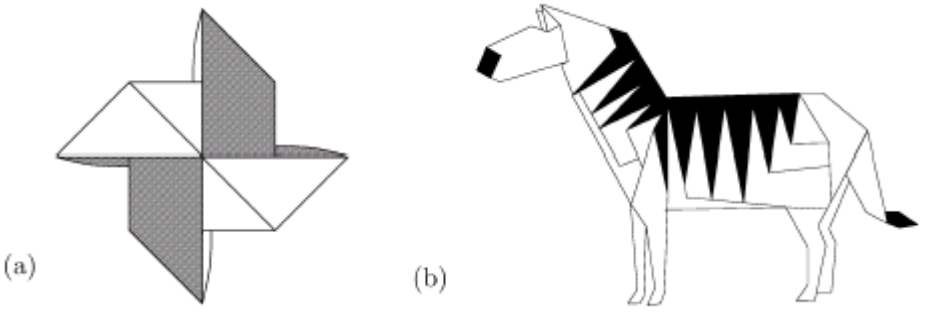


图 1.2. Folding flat silhouette with two colors

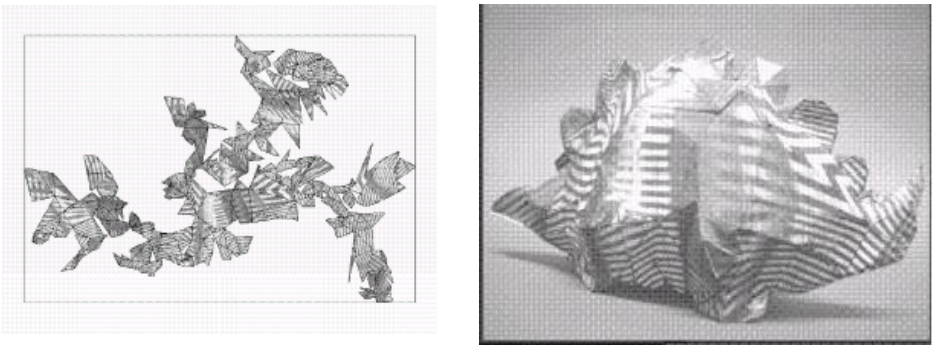


图 1.3. Paper unfolding and folding for 3D model

本文所要讨论的纸艺自动生成算法主要针对第一种类型的问题，即 Folding and Cutting Problem.

二、直架及其生成算法

2.1 直架

在本节，我们将介绍一种称为*直架*(Straight Skeleton)的几何结构，它构成了我们所要给出的折痕图的主干部分.

中轴(Medial Axis)是我们熟知的一种多边形的骨架结构，它的概念由平面点集的 Voronoi 图[17,18]推广而来. 形象地说，如果我们沿着多边形的边界点燃草原，那么中轴便是那些火苗相遇并因此而熄灭的地方. 直架也是一种多边形骨架结构，对直架来说，火苗并不是沿着圆形的方式蔓延，事实上，多边形内部的直架是通过一种收缩过程来定义的：均匀的收缩多边形，直到它变为非简单(non-simple)；然后，递归收缩每个子多边形，直到其面积变为 0，多边形和子多边形的顶点在这一过程中的轨迹便定义了直架. 显然，对于凸多边形 P，直架等价于中轴，但对于非凸多边形，两者往往是不同的. 图 2.1.1(a)中给出了一个递归收缩过程的例子. 当收缩为两个三角形时，原来的多边形变为两个子三角形. 继续收缩每个三角形，便继续产生一些线段，直到三角形消失(图 2.1.1(b)).

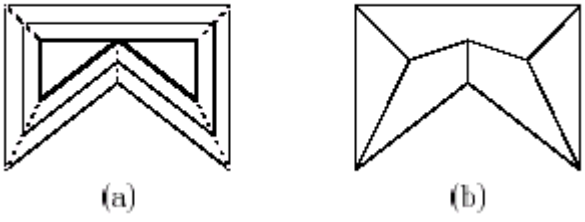


图 2.1.1. (a) 初始多边形的两个收缩了的多边形，第二个以粗线标出，是退化了的情况. 顶点的轨迹已用虚线标出. (b) 完整的直架

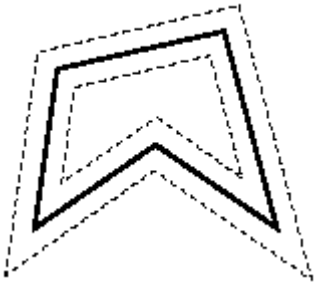
上面的收缩使我们得到了多边形内部的直架结构. 与此类似，我们也可以定义一种向外扩张的过程，从而便能得到多边形外部的直架结构. 两者结合起来便

是相应多边形完整的直架结构.

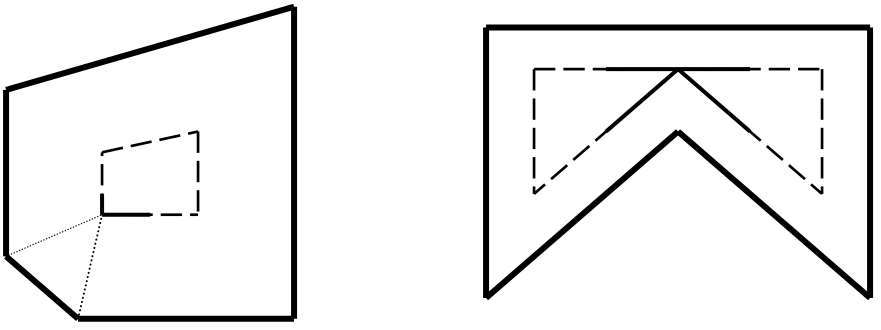
直架也可以通过波阵面(Wavefront)来定义. 多边形分别向内外两个方向产生波阵面(图 2.1.2(a)), 两者以恒定速率传播. 一般来说, 在传播过程中, 波阵面的形状可能会发生两种类型的改变(图 2.1.2(b)).

(1)边消失事件(Edge event): 波阵面的一条边退化为长度 0.

(2)边分裂事件(Split event): 由于其他部分的干涉, 波阵面的一条边因此而被分裂, 波阵面也因此分裂为两个子波阵面.



(a)多边形的起始波阵面



(b)边消失事件和边分裂事件

图 2.1.2. 波阵面

在事件发生后, 新的波阵面将由此产生, 它们将继续递归地传播下去. 最终, 各波阵面顶点的轨迹便是多边形的直架, 而直架的每一节点都对应于一个边消失

事件或边分裂事件.

目前，人们已经找到了生成中轴的线性算法[13]，但是对于生成直架却没有有效的方法，目前较好的结果为$O(n^{1+\varepsilon}+n^{8/11+\varepsilon}r^{9/11+\varepsilon})$ [11]，本节介绍的算法的复杂度为$O(n^2\log n)$ [12]. 尽管如此，中轴的描述一般会比直架复杂，因为中轴里可能含有曲线，而直架均是由线段或射线构成的，这是直架相对于中轴的一个优势，因而直架也在很多方面得到了应用[12]. 当然，两者的规模(节点数)仍都是线性的.

下面，我们用$SS(P)$或SS来表示简单多边形P的直架.

2.2 直架的一些基本性质

在波阵面的传播过程中，波阵面的每一条边e都会扫过一定的区域，我们称之为e的面. P的每一条边都会产生内外两条波阵面的边，因此也就会产生两个相应的面. 下面是这些面的一个基本的性质.

引理 2.2.1[12]. $SS(P)$的面(称为直架面(skeleton face))是单调多边形(monotone polygon).

证明. 考虑P的边e产生的一个面f. 在e的传播过程中，边消失事件和边分裂事件都不可能分割f，这表明f是一个连通集.

下面我们证明f在e方向上是单调的. 否则，如图 2.2.1，假设一条垂直于e的直线L在点x处离开f，此后又在离e更远的y点再次进入f. 这表明，在点x处由e产生的波阵面的边与另一条边e'产生的波阵面的边相遇，且e'不与L垂直，从而这个波阵面将早于e的波阵面传播到达y. 这与y的性质是矛盾的，因为y在

f 的边界上就表明 e 的波阵面将最先到达 y. ■

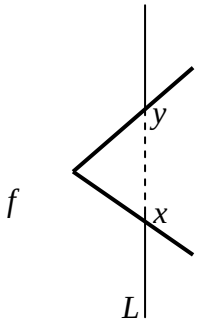


图 2.2.1. 引理 2.2.1 证明用图

令 $V(SS), L(SS), I(SS), E(SS)$ 分别表示 SS 的顶点集, 叶节点集, 内部节点(非叶节点)集和边集.

引理 2.2.2. SS 形成一棵树, $|E(SS)| = |V(SS)| - 1$.

引理 2.2.3. SS 将 P 分为 $|P|$ 个区域, 每个区域与 P 的不同的边相邻, 这些边称为该区域相应的切边(cut edge).

我们把整个过程中最后一次收缩 P 的子多边形得到的点作为直架的根节点 $r = root(SS) \in I(SS)$ (我们总假定 $root(SS)$ 是唯一的; 否则, 将有两个收缩过程同时结束, 这时认为 P 是一种退化情况). 以 r 为根很自然的就可以定义出直架的形结构中各节点之间的父子关系. 我们将在 4.2 节利用直架的这种以 r 为根的树形结构来描述凸多边形的折叠模拟算法.

另外, 凸多边形直架本身还有一些有益的性质, 这将在后文中适当地方谈到.

2.3 直架的生成算法

2.3.1 概述

直架的定义本身就向我们提供了一种直观的模拟生成方法. 取定一足够小的时间间隔 Δt，以此为时间片对波阵面的传播和干涉进行模拟，在每个时间片，求多边形的各个顶点，并以此判断是否有边消失时间和边分裂时间发生，如果有事件发生，就对波阵面的结构作相应改变，并重新计算波阵面各顶点的运动方向. 这样不断循环下去，直到不可能再有有效的事件发生为止. 在这一过程中记录下各顶点的运动轨迹，便可最终得到多边形的直架.

但是，这种模拟的方法不够精确，且开销较大. 不过，我们由此也可以看出症结所在: 因为直架的每一个节点都对应于一个边消失事件或边分裂事件，因此最关键的在于求出在波阵面传播过程中所有事件发生的时刻. [12]中提出的直架生成算法正是利用平面三角化方法巧妙的解决了这个问题.

算法的大致流程如下: 首先，对于包含简单多边形 P 的整个平面做一个有限制的三角剖分(在三角剖分时，多边形原有的边必须保留)；然后，将原始多边形链复制两条作为初始波阵面，一条上的所有顶点运动方向赋值为相应顶点的内角平分线方向(该链称为内向波阵面)，另一条上的所有顶点运动方向赋值为相应顶点的外角平分线方向(该链称为外向波阵面)；同时，我们还可以根据波阵面上各个顶点所在角的大小求出它们的运动速度，这样我们便得到了各顶点的运动矢量，每一个三角形的三个顶点的运动也就明确了；下面，求出每个三角形的消亡时间，并建立一个按照消亡时间递增排队的消亡队列，根据这个队列进行循环，对每个消失的三角形做相应处理，如记录新生成的直架节点，更新波阵面，更新各顶点的运动矢量，更新消亡队列等. 整个过程持续到消亡队列中的所有三角形都不消失为止. 该算法的复杂度为 $O(n^2 \log n)$.

2.3.2 三角剖分与维护

由于初始的三角剖分是对整个平面进行的，这不可避免的需要引入无限三角形(unbounded triangle)，如图 2.3.2.1 所示．引入无限点(infinite point)后，无限三角形也就具有了三个顶点．对有限三角形的消失比较容易理解，当它的三个顶点运动到一条直线上时，该有限三角形便消失了；对无限三角形，注意只有当两个有限的顶点运动到一点时，该无限三角形才认为消失，这是因为它的两条无限边(射线)实际上是虚拟的，可以朝向任何方向．

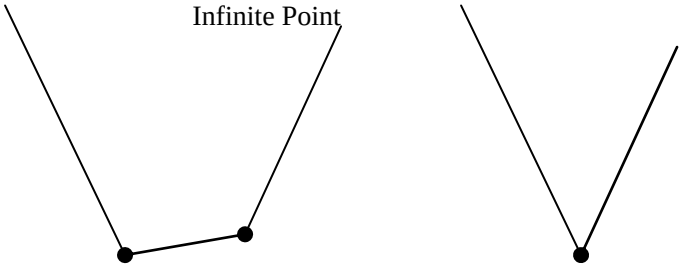


图 2.3.2.1．无限三角形与它的消亡

为了对整个平面进行剖分，我们需要先求出初始多边形的凸包(这可以使用经典的Graham扫描算法[17,18])．对凸包外的部分，我们用无限三角形进行剖分，而对于凸包内的部分，则可以使用多种已知的三角剖分算法(参见[15,16])．目前已有三角剖分的线性算法(B.Chazelle，1991)，但由于直架生成算法本身复杂度就达 $O(n^2 \log n)$，因此对三角剖分的复杂度并没有太高要求．图 2.3.2.2 提供了一个完整的三角剖分的例子．

内外向波阵面在传播过程中的形态会发生改变，有时甚至类型也会发生变化，如，外向波阵面在发生边分裂事件后，便会分裂为两个新的波阵面，其中处于外圈的仍然是外向波阵面，但处于内圈的却转化为了内向波阵面．在整个过程中，始终只有一个外向波阵面，而内向波阵面却可以产生很多个．对于外向波阵面以

外区域，我们始终使用无限三角形(外向波阵面凸包以外部分)和有限三角形(外向波阵面凸包以内部分)来维护对它的三角剖分；对于不同的内向波阵面以内区域，我们均保持它们各自的三角剖分．也就是说，在整个过程中，我们不断维护三角剖分的结果，让它始终是对波阵面未到达的区域的三角剖分，而对于波阵面已经扫过的区域，我们便不再关心了．具体的维护方法将在 2.3.4 中陈述．

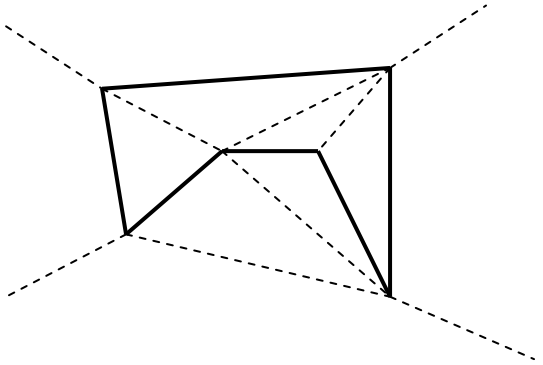


图 2.3.2.2. 三角剖分实例

直架的节点都是由波阵面传播过程中发生的边消失事件和边分裂事件产生的．在进行了三角剖分后，这两种事件都会触发某个或某几个三角形的消亡，这是显而易见的．因此，只要找出所有的三角形消亡事件，我们便能找到所有的边消失事件和边分裂事件，从而也就找到了直架所有的节点，而正如我们要在下一节中看到的，求出各个三角形消亡的时刻十分简单．这正是该算法最大的成功之处．

2.3.3 三角形的消亡

算法的核心是三角形消亡队列．它是将当前所有三角形按照消亡的先后顺序进行排列的．上文已经指出，对无限三角形，只有当它两个有限的顶点运动到同一点时，该无限三角形才认为消失，因此对无限三角形的消亡时刻的计算比较简单．对于有限三角形，只有当三点运动到共线的状态时才会消失．

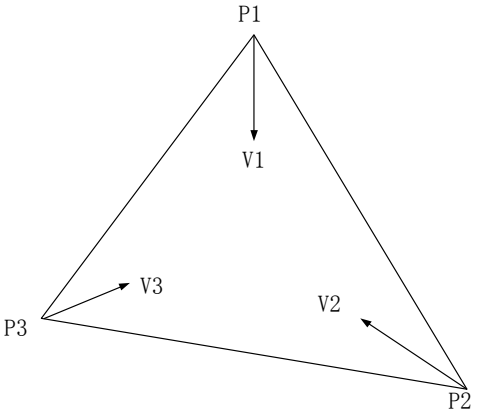


图 2.3.3.1. 有限三角形消亡时刻分析

如图 2.3.3.1, 顶点 P_i 的运动方程为: $P_i^{'} = P_i + t * V_i$, 其中, P_i 为顶点的初始位置, t 为时间, V_i 为顶点 P_i 的运动矢量. 当且仅当三个顶点共线时, 三角形消失, 此时三角形面积为 0; 如果无论怎样运动, 三点都无法共线, 则该三角形永不消失. 由三角形面积公式, 我们可以列出如下方程($\det(A)$ 表示矩阵 A 的行列式值):

$$\det(\begin{vmatrix} P_1' & P_2' & P_3' \\ 1 & 1 & 1 \end{vmatrix}) = 0$$

亦即:

$$\det(\begin{vmatrix} P_1 & P_2 & P_3 \\ 1 & 1 & 1 \end{vmatrix} + t \times \begin{vmatrix} V_1 & V_2 & V_3 \\ 1 & 1 & 1 \end{vmatrix}) = 0 \qquad (*)$$

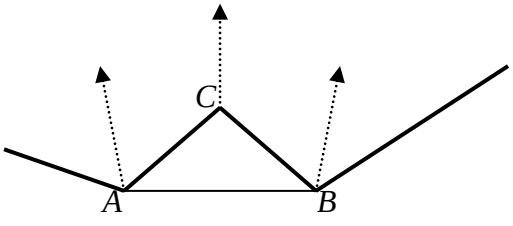


图 2.3.3.2. 在当前运动状态下, ΔABC 永不消失.

图中粗线条表示波阵面, 箭头表示波阵面顶点运动矢量.

(*)式是关于时间t的二次方程，通过分析该二次方程解的情况就可以判断三角形的消亡情况并精确的求出消亡的时间.

在某一时刻我们所维护的三角形中，按照当前的运动状态，未必所有的有限三角形都会消失，也就是说，方程(*)可能无正实数解. 图 2.3.3.2 给出了这样一个例子. 但是，我们仍然有如下结论.

引理 2.3.3.1. 对于内向波阵面，其内部区域的任意一个三角剖分中，按照当前的顶点运动状态(运动方向和速度)，至少有一个三角形会消亡.

证明. 设该波阵面为n边形，其顶点按逆时针依次为$P_0, P_1, \cdots, P_{n-1}$，并记$P_n = P_0$. 再令$P_i$的内向角平分线左右侧与对应的边的夹角分别为$\angle P_i^-, \angle P_i^+$ $(i = 0,1,\cdots,n-1)$，则必$\exists k, \angle P_k^+ + \angle P_{k+1}^- < \pi$，否则便有

$$\sum_{i=0}^{n-1} \angle P_i = \sum_{i=0}^{n-1} (\angle P_i^- + \angle P_i^+) = \sum_{i=0}^{n-1} (\angle P_i^+ + \angle P_{i+1}^-) \geq n\pi,$$

这与n边形内角和为$(n-2)\pi$矛盾. 设边$\overline{P_k P_{k+1}}$所在的三角形为T，则显然按照当前的顶点运动状态，T会消亡. ■

当算法结束时，所有留在消亡队列中的三角形都是对外向波阵面外部的剖分三角形. 也就是说，在整个过程中出现的内向波阵面都将消失. 这是因为，算法结束时，所有留在消亡队列中的三角形均是永不消失的三角形，此时如果还存在内向波阵面，那么由上述引理，我们所维护的它的三角剖分中的某个三角形必定仍会消失，从而导出矛盾.

至于为何按照当前的顶点运动状态不会消失的有限三角形可能会在以后的时刻消失，那是因为当前及此后消失的三角形改变了波阵面的形状，从而改变了顶点的运动矢量所造成的.

2.3.4 三种消亡事件及处理方法

三角形消亡时，它的三个顶点必定共线. 这时有两种情况：一是三点中至少有两点已经重合为一点；二是三点仅仅是共线，而没有两者重合的情况，此时，设三点 A,B,C 依次排列于某条直线上，则按 $\overline{AC}$ 是否为波阵面上的一条边又可分为两种情况. 综上，三角形的消亡有三种类型(参见图 2.3.4.1):

(1) 边消失事件：当波阵面上的两个顶点运动到同一个点时发生该事件.

处理方法：产生新的直架节点(即它们重合的点)，将波阵面上的这两个点合并即可.

(2) 边交换事件：当波阵面上的某个顶点穿越一条三角形的边 s 并且该边不是波阵面上的边，则发生该事件.

处理方法：将 s 移走，加入边 t，这种情况不产生新的直架节点，只是改变三角剖分方式而已.

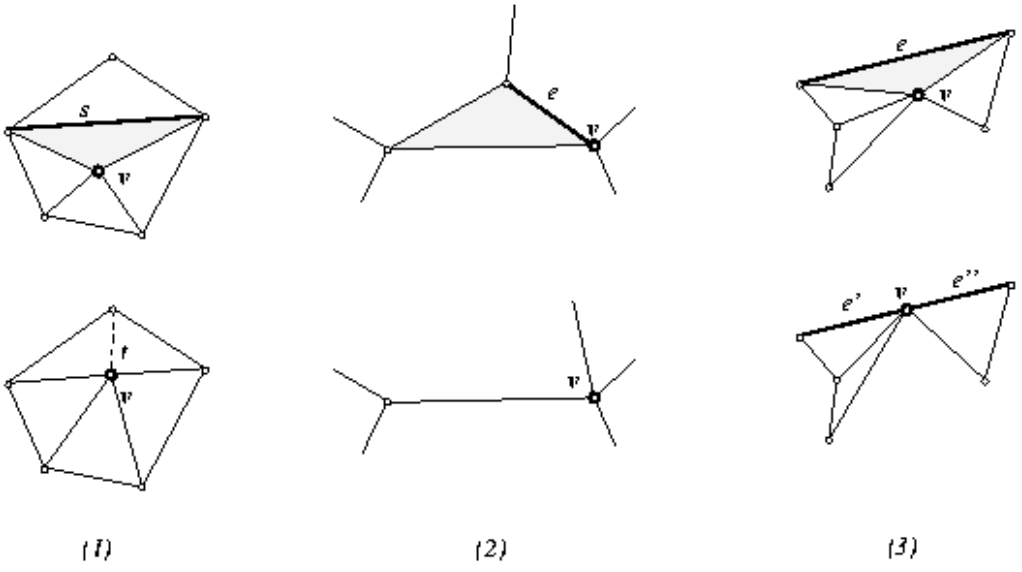


图 2.3.4.1 三种三角形消亡事件

(1)Flip Event (2)Edge Event (3)Split Event

(3) 边分裂事件：当波阵面上某顶点穿越波阵面上的一条边 e，则发生该事件.

处理方法：将被切分的边切分成 e', e''，原波阵面也因此而分裂为两段.

依次分析三角形消亡队列中的三角形消失情况，并对波阵面做相应的改变，同时按照新的波阵面更新三角形消亡队列，由以一步步产生直架的节点，直至消亡队列中的所有三角形都不消失. 这样，我们便得到了完整的直架结构.

2.3.5 算法运行实例

我们给出几个根据上述算法计算出的多边形直架结构的实例.

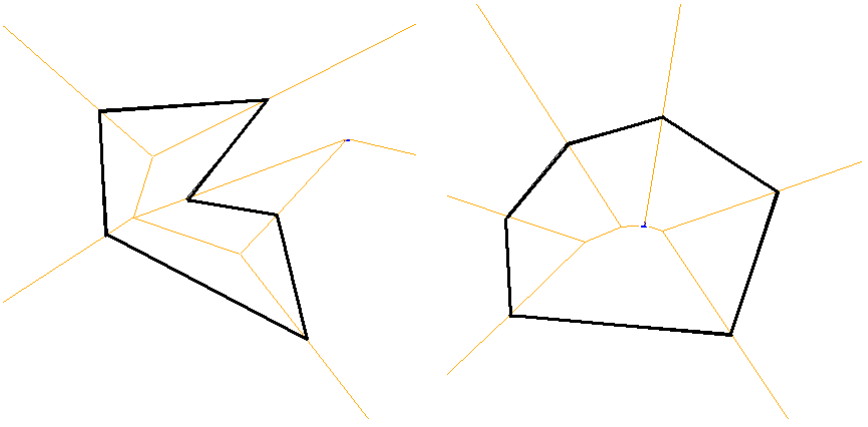


图 2.3.5.1. 一些算法运行结果实例

三、折痕图

3.1 垂线折痕

*折痕图*是指将平面中的一个简单多边形 P 分为有限个子多边形(称为*面*), 其中每一条边都对应于一条折痕. 直架本身是不可折叠的 因此除了直架, 我们所要求的折痕图中还包含*垂线折痕*(perpendicular fold), 将在这一节描述.

在得到直架之后, 对于平面内的任意一点 p , 都可以求出与它相关的垂线折痕, 它由一组线段和射线构成,其中的每一条线段或射线称为一条*垂线边*, 每一条垂线边都与一个直架面相对应. 这组垂线边可以递归的定义如下: 对于点 p 所在的每一个直架面 f , 令 l 为过点 p 且与 f 的切边垂直的直线, 令 m 为 l 与 f 求交所得的部分(可能是 p 自身, 一条线段, 一条射线等), 那么与点 p 相关的垂线折痕由 m 以及和 m 的端点相关的垂线折痕构成.

我们关心的垂线折痕是所谓的*实垂线*(real perpendicular), 它是与每一个直架节点相关的垂线折痕. 正如我们看到的, 垂线折痕的定义是一个递推性质的定义, 因此我们可以采用常规的递推算法来求出所有的实垂线.

直架节点数为 $O(n)$ ([12]), 因此实垂线也是 $O(n)$ 的, 可实垂线中所包含的垂线边却可能是无限的. 在 Folding and Cutting 问题中, 我们只需要实垂线落在纸张内的部分, 以它们作为垂线折痕. 下面的引理给出了递推算法中的停止条件.

引理 3.1.1.[19] 使用递推算法计算一条实垂线时, 如果先后两次计算一个直架面内得到的垂线边, 以及这过程中计算过的直架面内得到的垂线边, 都没有和设定的

纸张 R 相交，并且，后计算得到的垂线边在先计算得到的垂线边的外侧(相对于切边)，则以后计算的直架面内得到的垂线边，也不会和 R 相交.

垂线折痕不会互相碰撞，因为，垂线折痕与和它相交的多边形的边相互垂直. 因此，垂直折痕在 SS 的任何区域中都是相互平行的. 两条垂线折痕可能完全重合，但我们只认为它是一种退化的情况而不予考虑.

3.2 内外折分配

直架和垂线折痕构成了折痕图的所有折痕边，下面需要确定折痕的折叠方式，即向外折，向内折或者不折.

所有的直架边都是必须要折的. 由[3]，对于凸顶点所发出的边都定义为外折，对于凹顶点所发出的边都定义为内折；对完全在多边形内部的边定义为外折，完全在多边形外部的边则定义为内折.

实垂线的折叠方式比较复杂. 观察图 3.2.1 中的树形结构，如果将每个实垂线看成树中一个节点的话，那么相邻两条实垂线所夹的走廊就是对应两个节点的边，图 3.2.1 中，共有 A, B, C…J, K 共 11 条实垂线，对应于右边树模型中 11 个结点，左图中 H 和 G 之间的走廊用灰色标志，对应于树中的 HG 边. 事实上，如果我们定义纸的平面为 xy 平面，那么我们将左图折叠成三维模型后，从 z 轴方向所看到的就应该是右图中的树模型，也就是说，同一条实垂线在空间中应该折成一条垂线，这样从 z 方向看就是一个点，而对应的走廊也就被折成一条边，双边的走廊对应于树中一条线段，单边的走廊对应于一条射线，如右图中 A, E, F, K 四个点向外发出的箭头.

上述的树结构，实际上就是下文第四部分将要提到的阴影树(shadow tree).

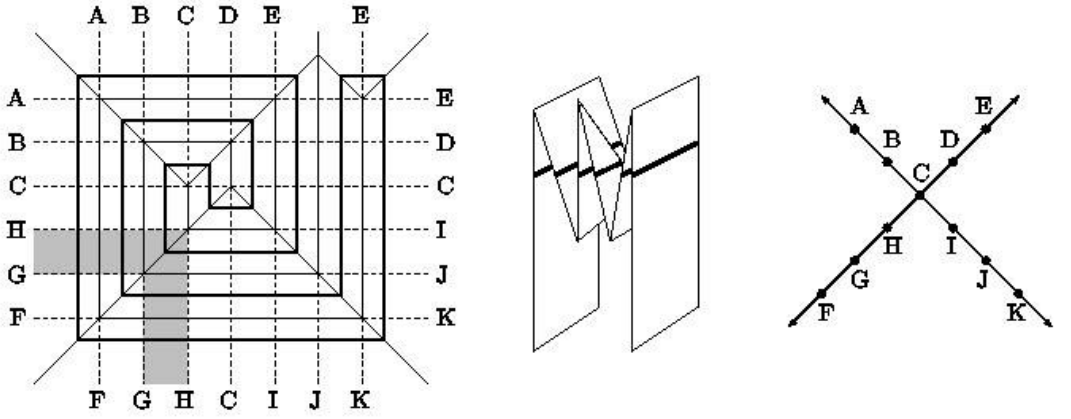


图 3.2.1. 实垂线的树模型

有了这个树的模型，我们就可以直观地得出折叠方法，即在平面上将一个树折成一条直线，我们可以以任意一点为根结点，然后向下拉直. 例如：我们以 A 为根，那么 C 就有三个子树，只要把这三个子树压成一条直线即可，因此我们需要将角 HCI 和角 ICD 变成零，而对应于左图就是要把走廊 HC 和 CI 之间的一部分实垂线进行折叠. 可以看出，选择不同的根结点所得出的折叠方法是不同的.

至于具体的树模型构造算法，以及如何确定每段实垂线的内外折分配，可以参见[19]中的描述. 以被标记为内折的点为起点，沿折痕回溯，交替标记每段垂线边为内折，外折，内折，外折，直至该实垂线对应的直架节点结束；同样，以被标记为外折的点为起点，沿折痕回溯，交替标记每段垂线边为外折，内折，外折，内折，直至该实垂线对应的直架节点结束.

3.3 折痕图计算结果实例

关于折痕图的一个重要性质，我们将在 4.4 节陈述，这里先给出一个折痕图的计算结果实例.

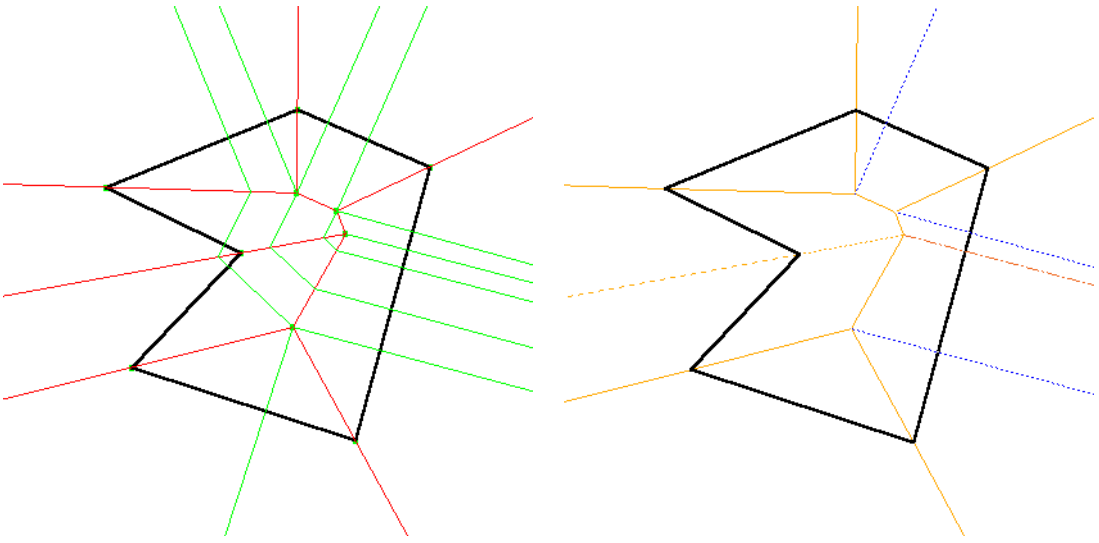


图 3.3.1. 折痕图计算结果实例. 左图为折痕边(直架和垂线折痕),

右边为进行内外折分配后的结果, 实线外折, 虚线内折, 不折的折痕边未画出

四、凸多边形折叠过程的模拟

4.1 极基

我们将首先介绍一个与 Folding and Cutting 十分类似的问题——极基. 对于 Folding and Cutting 来说, 纸一般是方形的, 它包含指定的多边形 P, 而极基问题中, 纸本身就是多边形 P. 后面会谈到, 极基实际上是一个比 Folding and Cutting 简单的问题, 但是对于凸多边形的过程模拟, 两者是等价的, 且用极基来说明会更加清楚.

给定一折痕图, 经过折叠后的纸称为*基*. 也就是说, 基是一个连续的分段全等的(piecewise-isometric)从 P 到三维空间的函数 b. 这里"分段全等"是指, b 限制在折痕图的任一面上是全等的.

单轴基(uniaxial base) b 定义如下. 首先, 对所有 $p \in P$, $b^z(p)$ 必须非负. 第二, 基到 xy 平面上的投影必须恰好是基和 xy 平面的交, 即应有

$$\{b(p) \mid p \in P, b^z(p) = 0\} = \{(b^{xy}(p), 0) \mid p \in P\}.$$

另外, 上述集合必须是一棵树的结构, 称之为阴影树. 从而, 折痕图的所有面经过 b 映射后是与 xy 平面垂直的. 最后, 每个面都只属于唯一的*折翼*(投影到阴影树中同一条边的所有面的集合 F), 并且, 每一个 $f \in F$ 的边界都能投影到这条边的一个或两个端点.

下面的一种更有用的对单轴的定义可以推广到平行于 xy 平面的任意平面. 我们迫使对任意 $c \geq 0$ 有

$$\{b(p) \mid p \in P, b^z(p) = c\} = \{(b^{xy}(p), c) \mid p \in P, b^z(p) \geq c\}.$$

也就是说，如果我们平行于 *xy* 平面剪切基(去掉剪切平面以下的部分)，则剩余部分仍然是上面所定义的单轴.

我们称一个基 b 是*极基*，如果它是单轴的，而且 $b^z(p)=0$ 当且仅当 p 在 P 的边界上. 事实上，任一呈多边形形状 P 的纸均可以折成一个极基(参见图 4.1.1 中给出的几个例子).

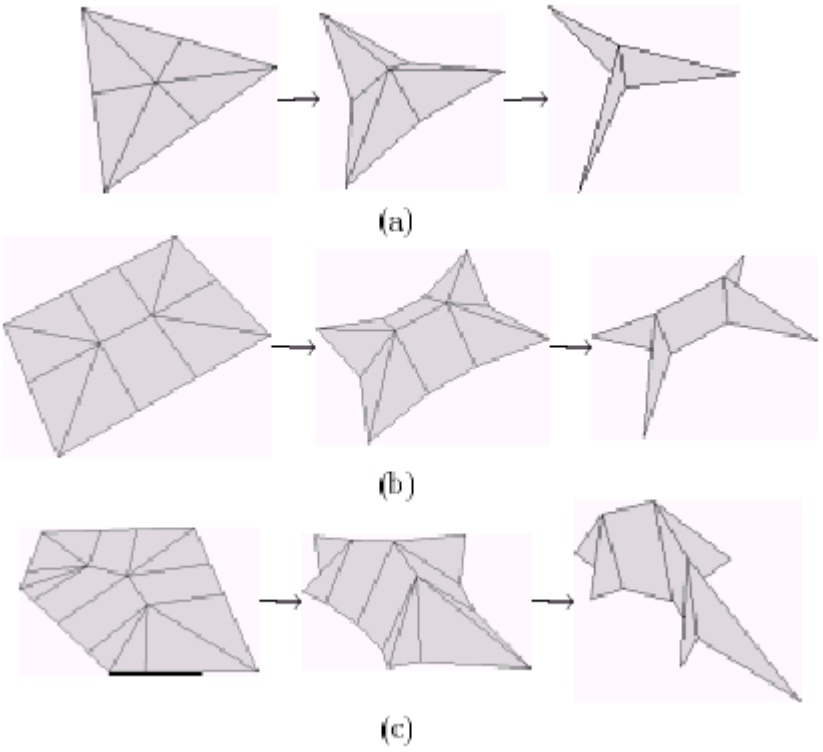


图 4.1.1 未折叠，部分折叠和完全折叠的极基. (a)三角形(b)凸四边形(c)凸五边形.

我们能针对 P 构造出一张纸的极基，事实上，极基的折痕图与第三部分中介绍的 Folding and Cutting 问题的多边形内的折痕图完全一致. 我们将在以后看到，它能够继续”压平”到一个平面，同时 P 的边界均处于一条直线. P 是凸的，我们

便能将处于 P 的顶点处的折痕(它们是角平分线)延长到方形纸的边界. 这里不会出现相交的情况, 因为凸多边形的角平分线都是"外向"的. 因此, 我们便得到了一个将 P 的边缘映射到同一条直线上的折平方案. 也就是说, 在一张纸上任意给定一凸多边形 P, 则这张纸能被折叠为一平面图案, 使得 P 的边界被映射到同一条直线上, 而其它部分都不会映射到该直线上. 因此, 对于凸多边形, 极基问题与 Folding and Cutting 两者没有本质差别. 而对于非凸多边形, 后者往往比前者复杂, 因为后者还需考虑纸张的处于多边形以外的部分如何进行折叠以配合多边形内折痕的问题.

4.2 一些符号和约定

下文会对将纸折为基 b 的过程 f 进行讨论. 为此, 我们引入取值于区间 $[0,1]$ 上的时间变量 t. 折叠过程是从 $P\times[0,1]$ 到三维空间中的一个连续函数(参见图 4.1.1). 初始时间 $t=0$ 表示开始时的那张形状为 P 的平面纸(即未折叠状态), 终止时间 $t=1$ 表示最后折叠得到的基(即完全折叠状态). 也就是说, 对任意 $p\in P$, $f(p,0)=(p,0)$ 且 $f(p,1)=b(p)$.

另外, 我们约定折叠过程中的某些拓扑性质. 首先, f 不允许纸张弯曲, 这等价于 f 必须在任一给定时刻均是分段全等的, 即 f 限制在任意一个面和任一时刻 $0\le t\le 1$ 上是全等变换. 第二, f 不能引起纸张与自身相交, 如果 f 对任一给定的 $t<1$ 均是一一映射时, 这个要求便自然满足了.

下面再为凸多边形 P 的折痕图定义一些记号, 此时, 垂线折痕均为线段(也就是说, 它们不会发生碰撞). 这时, 我们通常将垂线折痕与它们的端点(即它们所到达的多边形边界上的点)形成对应. 我们用 $perps(v)$ 表示和 $v\in V(SS)$ 相邻的垂足

集合，用 $edge(v_1)$ 表示 $v_1 \in perps(v)$ 所在的 P 的边．不同的 $v_1, v_2 \in perps(v)$ 确定了一个邻点 $w \in V(SS)$，记作 $nei(v_1, v_2)$．如果 $w \in I(SS)$，则存在 $w_1, w_2 \in perps(w)$ (分别记作 $perp(w, v_1)$ 和 $perp(w, v_2)$)与 v_1, v_2 相邻．(这里，“相邻”是指在 v_1, w_1 或 v_2, w_2 之间再无其他垂足).

另外，易知对所有 $v_1 \in perps(v)$，$|v - v_1|$ 均相同．我们用 $plength(v)$ 来记这个共同的长度．$plength(root(SS)) = \max\{ plength(v) \mid v \in I(SS)\}$.

我们用到的大部分运动均由圆来定义，下面再引入一些关于圆上的点的符号．如果 p, q 是二维空间中两不同的点，我们用 $\angle(p, q)$ 表示从 p 到 q 的有向线段到 x 轴的角度．如果 p 是三维空间中的一点，用 $c[p, \theta, r]$ 表示三维空间中垂直于 xy 平面，中心为 p，半径为 r，和 xz 平面夹 θ 角的圆．因此，$c[p, \theta, r]$ 上一点可以通过角度 φ 来定义，即 $c[p, \theta, r](\varphi) = p + r(\cos\varphi\cos\theta, \cos\varphi\sin\theta, \sin\varphi)$.

4.3 凸多边形的折叠过程

我们现在考虑凸多边形 P 的折叠过程 f，此时垂线折痕不会发生碰撞．而且，我们只考虑 P 非退化的情况．因为 f 必须在折痕图的每一面上都是全等的，我们只需在折痕图的顶点(也就是直架的顶点和垂线折痕的端点)处定义它就够了．沿着每条边和每个面进行线性插值就可得到 f 的完整定义．

在 2.2 中，我们已经说明，整个过程中最后一次收缩 P 的子多边形得到的点作为直架的根节点 $r = root(SS) \in I(SS)$．该节点具有如下性质(证明略):

引理 4.3.1. $b^z(r) = \max_{p \in P} b^z(p)$.

我们将递归地在各顶点处定义 f，从 $root(SS)$ 开始到 $v \in L(SS)$ 结束. 上述引理表明，在极基上，$b(r)$ 是最高点. 递归从 $root(SS)$ 开始也正是出于对这一点的考虑. 现在，我们假定 $f(root(SS),t)$ 的定义已经给出.

假设已知一顶点 $v \in I(SS)$ (比如，$v = root(SS)$) 的运动方程，令 $v_1 \in perps(v)$. 我们迫使 v_1 沿着以 v 为圆心，$plength(v)$ 为半径的圆运动，从角度 0 到角度 $-\pi/2$，当然同时 v 也会运动. 因此，我们定义

$$f(v_1,t) = c[f(v,t), \angle(v,v_1), plength(v)](-t\pi/2).$$

假设 $v_1, v_2 \in perps(v)$，$v \in I(SS)$ 且 $f(v,t)$ 已定义，另外，设 $parent(w) = v$，其中 $w = neigh(v_1,v_2)$. 我们定义 $f(w,t) = g[v_1,v_2,w](t)$. 下面我们考虑 g.

假设对某一 $v \in I(SS)$，$v_1, v_2 \in perps(v)$ 不同，$parent(neigh(v_1,v_2)) = v$，$f(v,t)$ 已定义(因此 $f(v_1,t), f(v_2,t)$ 已定义). 令 b 表示 $edge(v_1), edge(v_2)$ 的平分线，即指向 $neigh(v_1,v_2)$ 的 v_1, v, v_2 的角平分线(图 4.3.1). 显然，$v, neigh(v_1,v_2) \in b$. 我们将给出任意一点 $w \in b - \{v\}$ 的运动方程. 注意到，$-\pi/2 \le \angle w,v,v_1 = \angle w,v,v_2 \le \pi/2$，这是因为 b 的指向决定的. 这样的点在以 v 为圆心的某个圆上运动，即

$$g[v_1,v_2,w](t) = c[f(v,t), \angle(v,w), |v-w|](h(t)),$$

其中 $h(t)$ 待定，它确定了点沿圆的"运动速度".

我们迫使 $h(0) = 0$，即 $g[v_1,v_2,w](0) = (w,0)$. 下面我们只需在定义 $h(t)$ 时保证

$|f(v_1,t)-g[v_1,v_2,w](t)|$为恒定值.

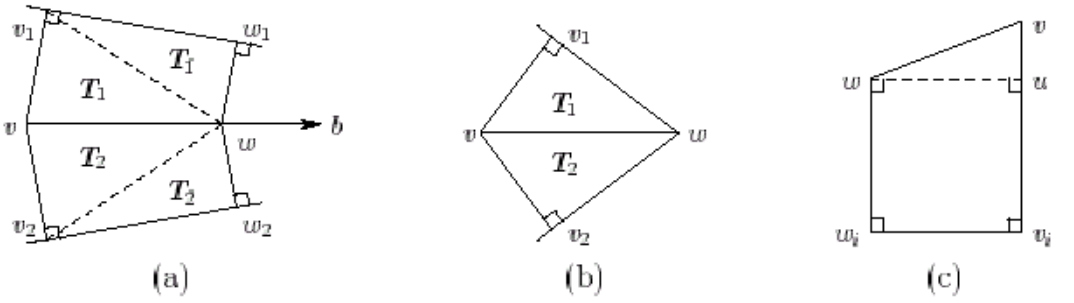


图 4.3.1. (a-b)全等保持. w 是$\in b-\{v\}$ 的任一点.

有两种情况, (a) $w \in I(SS)$ *(b)* $w \in L(SS)$

(c) $f^z(v,t)-f^z(w,t)=|v-u|=|v-v_i|-|w-w_i|=plength(v)-plength(w)$

引理 4.3.2.[5] 令$c_1=c[p,\theta_1,r_1], c_2=c[p,\theta_2,r_2]$. $\alpha=\theta_1-\theta_2$, $d=|c_1(0)-c_2(0)|$. 假设$\alpha\in[-\pi/2,\pi/2], \varphi_1,\varphi_2\in[-\pi/2,0]$. 则$|c_1(\varphi_1)-c_2(\varphi_2)|=d$ 当且仅当

$$\cos\varphi_2=\frac{|\sin\alpha|+\cos\varphi_1}{|\sin\alpha|\cos\varphi_1+1}$$

$$\sin\varphi_2=-\sqrt{1-\cos^2\varphi_2}$$

证明. 假设在三维空间中我们有两个垂直于 xy 平面的同心圆C_1, C_2, r_1, r_2 分别是它们的半径. 不失一般性, 通过平移可使圆心位于原点, 通过旋转可使C_1在 xz 平面上. 因此, C_1 上任一点具有坐标 $(r_1\cos\varphi_1, 0, r_1\sin\varphi_1)$, 其中 φ_1 是参数, $0\le\varphi_1<2\pi$.

C_2 上的点可被参数化定义为 $(r_2\cos\varphi_2\cos\alpha, r_2\cos\varphi_2\sin\alpha, r_2\sin\varphi_2)$, φ_2 是具有和φ_1相同的界的参数, α 是C_1, C_2所在平面的夹角.

对于给定的φ_1,φ_2，我们可以计算圆上相应两点的距离的平方:

$$\begin{aligned}&(r_1\cos\varphi_1 - r_2\cos\varphi_2\cos\alpha)^2 + (r_2\cos\varphi_2\sin\alpha)^2 + (r_1\sin\varphi_1 - r_2\sin\varphi_2)^2\\&= r_1^{\,2} + r_2^{\,2} - 2r_1r_2(\cos\varphi_1\cos\varphi_2\cos\alpha + \sin\varphi_1\sin\varphi_2)\end{aligned} \tag{4.3.1}$$

我们需要根据φ_1来确定φ_2以使这个距离是常数. 我们只关心$\varphi_1 \in [-\pi/2, 0]$的情况. 只需将$\varphi_1 = \varphi_2 = 0$代入(4.3.1)中便可以得到所需的距离平方$d^2 = r_1^{\,2} + r_2^{\,2} - 2r_1r_2\cos\alpha$.

因为(4.3.1)必须等于d^2，则有

$$\begin{aligned}&\cos\alpha = \cos\varphi_1\cos\varphi_2\cos\alpha + \sin\varphi_1\sin\varphi_2\\&\Rightarrow \cos^2\alpha(1-\cos\varphi_1\cos\varphi_2)^2 = (1-\cos^2\varphi_1)(1-\cos^2\varphi_2)\end{aligned}$$

用求根公式可求出

$$\begin{aligned}\cos\varphi_2 &= \frac{\cos^2\alpha\cos\varphi_1 \pm \sin^2\varphi_1\sin\alpha}{\cos^2\alpha\cos^2\varphi_1 + \sin^2\varphi_1}\\&= \frac{(\sin\alpha \pm \cos\varphi_1)(-\sin\alpha\cos\varphi_1 \pm 1)}{(1+\sin\alpha\cos\varphi_1)(1-\sin\alpha\cos\varphi_1)}\\&= \frac{\sin\alpha \pm \cos\varphi_1}{\sin\alpha\cos\varphi_1 \pm 1}\end{aligned} \tag{4.3.2}$$

因为我们知道$-\pi/2 \le \varphi_2 \le 0$，所以上式给出了$\varphi_2$关于$\varphi_1,\alpha$的关系. 为了确定符号，我们考察$\varphi_1 = -\pi/2$的情况，由(4.3.2)这意味着$\cos\varphi_2 = \pm\sin\alpha$. 因为$\varphi_2 \in [-\pi/2, 0]$，所以$\varphi_2 = -\left|\cos^{-1}(\pm\sin\alpha)\right| = -\left|\pi/2 \mp \alpha\right|$. 我们只关心$\alpha \in [-\pi/2, \pi/2]$的情况，也就是说，两圆分开不超过$\pi/2$. 因$\varphi_2 \in [-\pi/2, 0]$，如果$\alpha \in [0, \pi/2]$,则∓应取－，如果$\alpha \in [-\pi/2, 0]$，则∓应取＋. 也就是，我们选择∓为$\sin\alpha$的符号. 从而，

$$\cos\varphi_2 = \frac{|\sin\alpha| + \cos\varphi_1}{|\sin\alpha|\cos\varphi_1 + 1}. \qquad \blacksquare$$

在我们所考虑的具体问题中，φ_1, φ_2 分别表示 $-t\pi/2, h(t)$．这些方程完全用 φ_1(从而，t)定义了 $h(t)$，从而也就定义了 g．

到这里，我们已经知道如何递归，现在只需定义 $f(r,t)$ 就行了，其中 $r = root(SS)$．我们选择了如下定义，它能保证 r 的垂足落在 xy 平面上:

$$f(r,t) = (r, -plength(r)\sin(-t\pi/2)) = (r, plength(r)\sin(t\pi/2)).$$

前面已经提到过，对于凸多边形 P，我们能将处于 P 的顶点处的折痕(它们是角平分线)和它的垂线折痕自然延长到方形纸的边界．从而，上述极基的模拟方法完全可以方便地移植到 Folding and Cutting 问题中凸多边形的模拟中去，只不过适当地将各圆的半径按照延长后的长度做相应改变即可．这里便不赘述了．

最后，我们需要把极基折平．f 的基 b 可以看作是三维的"纸"，它是可以被折平的．$b(P)$ 的每个面均垂直于 xy 平面，因此这只是一个将一棵树(阴影树)折叠成线段的二维问题，那自然是很简单的．另外，从上面的模拟过程可以看到，阴影树具有和 $SS(P)$ 相同的结构和角度，只是各边的长度可能不同．

遗憾的是，对非凸多边形，对其折纸过程的描述存在种种困难，目前仍处于研究中，尚无较为满意的结果．

4.4 OpenGL 带来的问题

在 OpenGL 中，绘制多边形 P 的语句如下:

```
glBegin(GL_POLYGON);
glVertex3f(P1);
……
glVertex3f(Pn);
glEnd();
```

上述语句执行后，OpenGL 会绘出依次以 $P_1,\cdots,P_n$ 为顶点的多边形. 但是，这里有个限制，即所作的多边形必须是凸多边形，如果不是，OpenGL 将作出错误的图形. 这使得我们在绘制凹多边形时不得不先对其进行凸分解，然后再绘出分解所得的各个子凸多边形. 可以想见，这将非常麻烦.

由于我们已经约定在模拟过程中折痕图的各个面是刚性的，因此在实际编程过程中，对整张纸的作图是通过分别作出折痕图的各个面来实现的. 对于折痕图，利用引理 2.2.1，我们得到下面重要的结论.

引理 4.4.1. 对于任何简单多边形 P，其折痕图上的各个面均为凸多边形.

证明. 由文中第三部分知，折痕图由两部分组成，$SS(P)$ 与垂线折痕. 首先，垂线折痕均是线段，且每一条线段都完全落在 $SS(P)$ 分割纸张形成的的某个面内，因此，垂线折痕的引入不会带来任何新的凹顶点(也称之为优角顶点，即内角大于 180 度的顶点).

下面，我们说明 $SS(P)$ 原有的凹顶点是如何被后来增加的垂线折痕破坏的. 如图 4.4.1，设 $v\in V(SS)$ 是 $SS(P)$ 中由边 e 产生的面的任一凹顶点，显然 v 是内部节点.

现由 v 向 e 引垂线，则该垂线的方向必定不会落在区域 1，3 内部，否则，假设垂线的方向为 a_1，则我们可以作出一条与 a_1 平行的直线 L_1，它与 v 的两邻边 k_1,k_2

相交，且通过区域 4. 而 L 垂直于 e，这便与引理 2.2.1 矛盾.

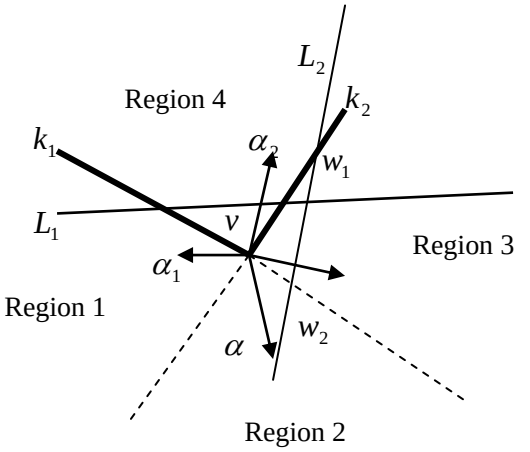


图 4.4.1. 证明引理 4.4.1 用图

垂线的方向也不可能落在区域 4 内部，否则假设垂线的方向为 a_2，则我们可以作出一条与 a_2 平行的直线 L_2，它与 v 的邻边 k_2 相交于 w_1，与 k_1 的内向延长线相交于 w_2，且 w_2 在面内. w_1 在面的边界上说明 e 的波阵面在此与另一边的波阵面相遇，由于后者并非沿着 L_2 传播，因此它将先期到达 w_2，矛盾.

最后，垂线的方向也不可能落在 k_1, k_2 或其延长线上(留给读者验证). 因此垂线的方向必落在区域 2 内部，如 α. 也就是说，在折痕图中，从 v 将引出一条沿 α 方向的垂线折痕. 这条折痕便破坏了 v 的凹性.

由上所述，$SS(P)$ 上原有的凹顶点都将被破坏，而新引进的垂线折痕并不带来新的凹顶点，故在最后的折痕图中没有凹顶点. 证毕. ■

根据上述引理，我们在绘制各个面的时候便不必对它进行凸分解后再进行绘制了.

4.5 模拟结果实例

作为结束，我们给出三组模拟折叠的结果．它们有些类似于图 4.1.1，但图 4.1.1 是对极基问题的模拟，而这里给出的是对 Folding and Cutting 问题的模拟，前者的纸张形状为多边形，后者的纸张形状为方形．

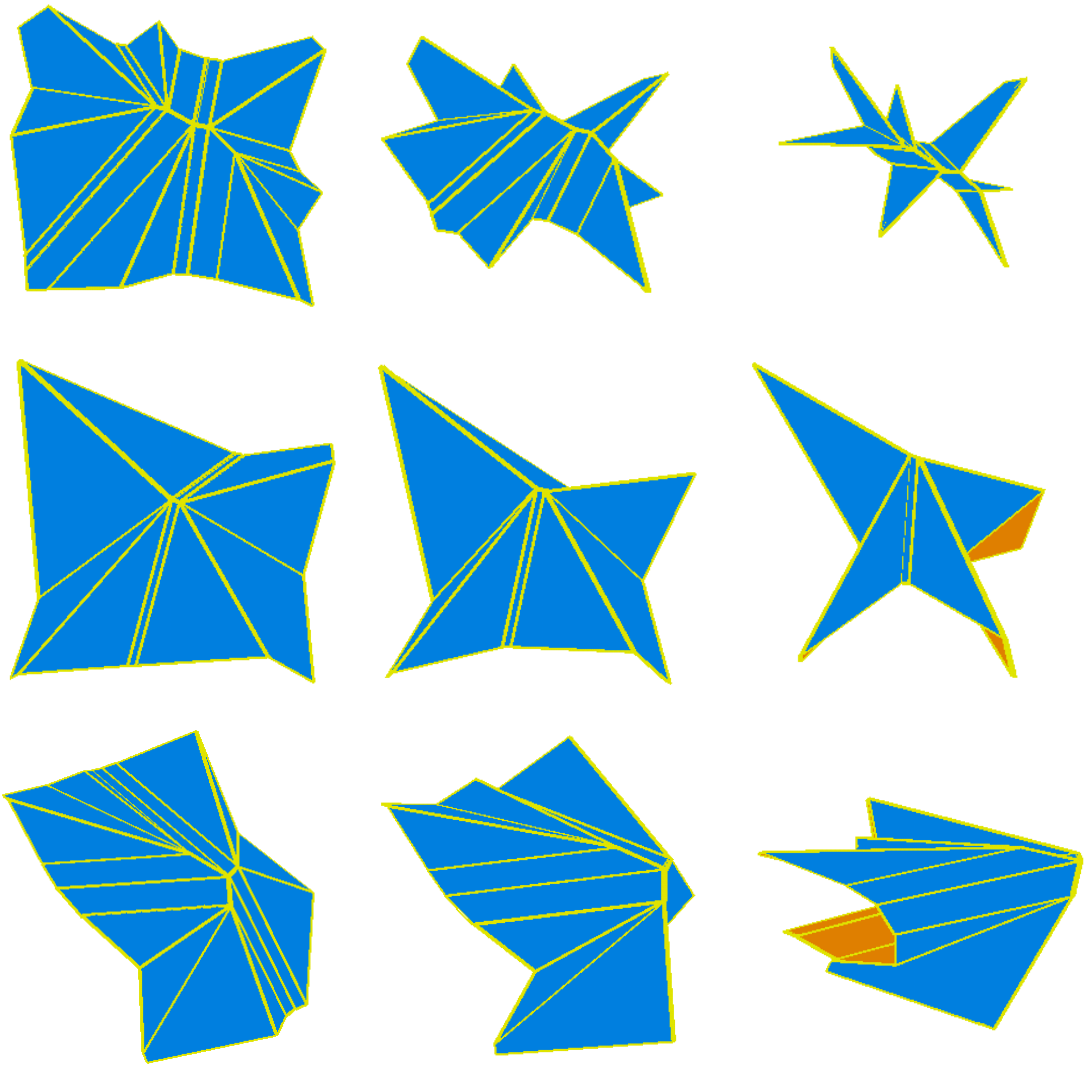


图 4.5.1. 三组模拟结果

参考文献

[1] E.D.Demaine, M.L.Demaine, Joseph S.B.Mitchell. Folding flat silhouettes and wrapping polyhedral packets: new results in computational origami. In *Computational Geometry*, pp 3-21, Vol.16, 2000.

[2] S.Bangay. From virtual to physical reality with paper folding. In *Computational Geometry*, pp 161-174, Vol.15, 2000.

[3] E.D.Demaine , M.L.Demaine and Lubiw. Folding and cutting paper . In *Proc. Japan Conf. Discrete and Computational Geometry* , Tokyo , 1998.

[4] M.Bern, E.Demaine, D.Eppstern and B.Hayes. A disk-packing algorithm for an origami magic trick. In *Proc. Int. Conf. Fun with Algorithms*, Italy, June 1998.

[5] E.D.Demaine and M.L.Demaine Computing extreme origami bases. Tech. Rep. CS-97-22, University of Waterloo, May 1997.

[6] R.J.Lang. A computational algorithm for origami design. In *Proc. 12th Symp. Computational Geometry*. pp 98—105, Philadelphia, May 1996.

[7] O.Aichholzer, D.Alberts, F.Aurenhammer and B.Gärtner. A novel type of skeleton for polygons. In *J. Universal Comput. Sci.* 1(1995), pp 752-761.

[8] E.D.Demaine, M.L.Demaine and Lubiw. Folding and one straight cut suffice. outline of Tech . Rep. CS-98-18, University of Waterloo, 1998.

[9] N.Kishi, Y.Fujii and T.College. Origami, Folding paper over the web.

[10] E.D.Demaine and M.L.Demaine. Planar drawings of origami polyhedra.

[11] D.Eppstein and J.Erickson. Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. Submitted to 14th ACM Symposium on Computational Geometry, Minneapolis/St. Paul, June, 1998.

[12] O.Aichholzer and F.Aurenhammer. Straight skeletons for general polygonal

figures in the plane. In *Proceedings of the 2nd Annual International Conference on Computing and Combinatorics, Vol. 1090 of Lecture Notes in Computer Science*, pp 117-126, Springer, 1996.

[13] F.Chin,J.Snoeyink and C.A.Wang. Finding the medial axis of a simple polygon in linear time. In *Proc. 6th Ann. Int. Symp. Algorithms and Computation (ISAAC 95)*, Lecture Notes in Computer Science 1004, pp 382-391, 1995.

[14] S.K.Kim, Chan-Su Shin and Tae-Cheon Yang. Placing two disks in a convex polygon. In *Information Processing Letters* 73(2000), pp 33-39.

[15] C.A.Wang, F.Y.Chin and B.Yang. Triangulations without minimum-weight drawing. In *Information Processing Letters* 74(2000), pp 183-189.

[16] P.Cignoni, C.Montani and R.Scopigno. DeWall: A fast divide and conquer Delaunay triangulation Algorithm in E^d. In *Computer Aided Design*, Vol.30, No.5, pp 333-341, 1998.

[17] 周培德. 计算几何——算法分析与设计. 清华大学出版社, 广西科学技术出版社, 2000.

[18] 邓俊辉. 清华大学 2000 计算几何课程讲义. http://cad3.cs.tsinghua.edu.cn/~djh/cg/handout/

[19] 丁俊勇, 朱旭平, 郭镔. Folding and one straight cut on simple polygon. 清华大学 2000 计算几何课程实验报告

[20] 白燕斌, 史惠康. OpenGL 三维图形库编程指南. 机械工业出版社, 1998

附录一　名词索引

Corridor　走廊

Crease Pattern　折痕图

Cut Edge　切边

Distance-Preserving　保距

Edge Event　边消失事件

Extreme Origami Base　折纸极基

Flap　折翼

Flat Foldable 可折平的

Flip Event　边交换事件

Inner Wavefront　内向波阵面

Medial Axis　中轴

Monotone Polygon　单调多边形

Mountain Fold　外折

Origami　折纸

Outer Wavefront　外向波阵面

Perpendicular Edge　垂线边

Perpendicular Fold　垂线折痕

Real Perpendicular　实垂线

Reflex Vertex　凹顶点(或优角顶点)

Shadow Tree　阴影树

Skeleton Face　直架面

Split Event　边分裂事件

Straight Skeleton　直架

Unbounded Triangle　无限三角形

Uniaxial Base　单轴基

Valley Fold　内折

Wavefront　波阵面

附录二　英文概述