

凸/非凸多边形的明可夫斯基和隐式算法及其可视化

方涵斌 周洽屹 汪亦衡

1. 背景与应用

1.1. 选题背景

简单多边形的明可夫斯基和（Minkowski Sum）是计算几何中的一个基本概念。它描述了两个简单几何对象（通常是二维的多边形或凸集）组合的方式，设有表示两个简单几何对象的两个集合 A 和 B ，它们是欧几里得空间中的子集（比如平面中的凸多边形、线段等），明可夫斯基和是这两个集合所包含的所有可能的向量和的集合。具体地，明可夫斯基和定义为：

$$A \oplus B = \{a + b | a \in A, b \in B\}$$

简单地说，明可夫斯基和可以视为两个图形的“卷积”。从另一个角度看，从 B 中任一点到 A 中任一点的有向距离的集合 C 也可以表示为：

$$C = A \oplus (-B)$$

则 C 刻画了 A 和 B 之间的相对空间关系。

1.2. 选题意义

由于明可夫斯基和的数学意义非常重要，机器人学、计算机图形学、自动驾驶、碰撞检测和路径规划等领域都对其有所涉及。特别是在处理复杂的形状和物体交互时，明可夫斯基和提供了一种直接的数学工具，帮助我们进行形状的合成、物体的位移和碰撞检测等操作。因此，研究如何高效地计算明可夫斯基和是一个十分重要的问题。

明可夫斯基和有以下应用场景：

- **机器人学和自动驾驶**：处理控制体与环境的交互，特别是在进行路径规划，与目标互动和避让障碍物时。
- **碰撞检测**：通过计算物体的明可夫斯基和，可以快速判断两个物体是否发生碰撞，尤其是在复杂环境中。此外，其还可以做嵌入距离的检测。
- **计算机图形学**：帮助在虚拟环境中模拟物体的组合和运动。

现实场景中所涉及的物体大多属于或可近似归约为简单多边形及其组合的范畴。因此，本篇工作主要专注于简单多边形（主要是一个凸多边形和一个非凸简单多边形）的明可夫斯基和算法及其可视化。虽然在数学概念上，明可夫斯基和的表示较为简约，但在工程计算中，对于凸/非凸多边形的明可夫斯基和问题，其空间复杂度对于多边形顶点的数量呈 $O(n * m)$ 的增长，这使得常规算法不可能在低于平方时间内将其求解。这也是我们本文主要研究的问题。

我们参考的原文主要是 *Computing the Intersection-Depth of Polyhedra*, 作者是 David Dobkin 等，1993 年发表于 *Algorithmica*。这篇文章创造性地提出了一种基于“隐式”表示明可夫斯基和的算法，来降低计算明可夫斯基和的过程在时间和空间上的复杂度。简单来说，对于最多可能具有 $O(n * m)$ 个顶点的明可夫斯基和，本文所介绍的算法可以在 $O(n \log^2(m) \log(n) + m)$ 时间内将其隐式求解。也即，**我们所介绍的算法时间复杂度小于答案真正的空间复杂度**。

1.3. 本文工作

在本文中，我们完成了以下工作：

1. 补充了原文遗漏的理论证明和算法实现：本文不是对参考原文的翻译。本文补充和修正了参考原文中多处涉及到算法整体复杂性的数学定理的证明遗漏和不完善证明，同时对原文中完全省略的合并算法也进行了实现细节和复杂度证明上的补充。主要补充的证明包括**引理 3.1.1**，**事实 3.2.1 和 3.2.2**，以及**3.3 凸弧求交算法 的全部内容**。

原文除了创造性地提出了“隐式表示”这一概念以证明该算法可以在亚平方时间内求解以外，并没有展示算法的详细细节及其在时间复杂度上的严格推导。而本文在其基础上进行了一定的延伸，可以作为一个完善的说明手册，帮助一个零基础的计算几何爱好者从逻辑上完全理解“凸多边形和非凸简单多边形的明可夫斯基和”这个可能具有**平方**复杂度的结构如何得以**亚平方**时间内被隐式地求解。

2. 二维多边形的明可夫斯基和计算：开发一个高效的算法，能够计算任意两个二维简单多边形的明可夫斯基和。特别的，我们对凸包对非凸包的明可夫斯基和进行优化。

3. 可视化展示：为计算结果提供可视化，可操作的界面，用户能够直观地看到两个多边形计算明可夫斯基和的过程和最后的结果。

本文的工作在以下方面具有较优的表现：

- **时间复杂度**：能够在多边形顶点数较多时，依然保持较低的计算时间。
- **空间复杂度**：在进行大规模计算时，内存占用应保持在合理范围内，不影响整体性能。
- **可视化交互**：可视化模块能够清晰地展示多边形和明可夫斯基和的关系，并支持用户交互式操作，包括设计一对任意的多边形和后续的明可夫斯基和运算过程。

2. 研究综述

目前，在计算简单多边形明可夫斯基和的过程中，已有多种方法和算法。根据对象的不同，这些方法主要可以分为以下两类：

- **凸包的算法：**对于二维凸包，明可夫斯基和的计算通常较为简单，因为凸集具有良好的几何性质。在**凸包算法**中，两个凸包A,B的明可夫斯基和也是凸包，其拥有A,B全部的边，按极角排序。
- **基于分解的方法：**对于一般的两个多边形A和B（不一定是凸集），一种直接的计算方法是把A, B分解为凸多边形的组合，计算A, B内任意一对凸多边形对的明可夫斯基和，然后将所有的明可夫斯基和并为一个简单多边形。对于A, B都是非凸多边形的情形，分解算法是一种**暴力方法**。

现有的方法在处理具有复杂形状或较大数据量的情况下，仍然面临着计算效率和精度的挑战。特别是在实时应用中，如何有效地优化计算并降低计算成本，仍然是一个关键问题。

在第三章中，将详细展示本工作所依托的理论背景，包括具体的定义，关键公式和引理以及必要的算法细节。我们的理论部分主要参考 David Dobkin 等在1993年的一篇工作《Computing the Intersection-Depth of Polyhedra》。

3. 理论部分

我们已经知道了明可夫斯基和的定义：

$$A \oplus B = \{a + b | a \in A, b \in B\}$$

但是仅仅根据定义我们无法用离散的方式描述明可夫斯基和。我们必须将其转换为点或线段的表示方式，以支持计算机的存储和计算。

3.1. 凸多边形的明可夫斯基和

- 事实 3.1.1：

给定两个凸多边形 P 和 Q ，分别有 n 和 m 个顶点，我们可以在 $O(n + m)$ 的时间内计算二者的明可夫斯基和 $P \oplus Q$ 。

实际上这个形状很好想象。但为了相对严谨地证明，相关引理的文字表述如下：

- 引理 3.1.1. 对于分别具有 n 和 m 个顶点与边的凸多边形 P 和 Q ，其明可夫斯基和 $P \oplus Q$ 为具有 $n + m$ 个顶点的凸包。

由于 P 和 Q 都具有凸性，即 $Ch(P) = P$, $Ch(Q) = Q$ (Ch 为凸包算符)，而向量加法对凸性封闭，其明可夫斯基和自然为凸包。明可夫斯基和也可以表示为：

$$P \oplus Q = Ch(bd(P)) \oplus Ch(bd(Q))$$

可见，凸包 P, Q 的明可夫斯基和也可以视作仅与 $bd(P)$ 和 $bd(Q)$ 有关。进一步，对于任意的 $a_1, a_2, \dots, a_n \in bd(P), b_1, b_2, \dots, b_n \in bd(Q)$ ，以及任意的 a_{n+1}, b_{n+1} 分别作为 $a_1 \dots a_n$ 和 $b_1 \dots b_n$ 的凸组合，即

$$\begin{aligned} a_{n+1} &= R_a(a_1 \dots a_n) \\ b_{n+1} &= R_b(b_1 \dots b_n) \end{aligned}$$

其中 $a_1 \dots a_n$ 和 $b_1 \dots b_n$ 对应的系数 R_{ai}, R_{bi} 都为正数，且系数和为 1。因此，对于在 $bd(P) \oplus bd(Q)$ 里找到 n^2 组组合 $(a_i + b_j)$, $1 \leq i, j \leq n$ ，我们需要计算出每个组合对应的系数 c_{ij} ，使得 $a_{n+1} + b_{n+1}$ 为这些组合的凸组合。进一步分析，我们有 n^2 个未知数， $2n$ 个方程（对于每个 a_i, b_i ， $R_{ai} = \sum_i c_{ij}$ ， $R_{bi} = \sum_j c_{ij}$ ），因此 $n \geq 2$ 时必然可解， $n \geq 3$ 时非退化情况下有无穷多解。一个快速计算的技巧是将 c_{ij} 想象为一个 $n \times n$ 矩阵，而且是秩-1 矩阵。其中行向量与 R_a 线性相关，列向量与 R_b 线性相关。只要归一化条件满足，即所有元素之和为 1。

也就是说，对于任意的 a_{n+1}, b_{n+1} ：

$$a_{n+1} + b_{n+1} \in Ch(bd(P) \oplus bd(Q))$$

必然成立。

也就是说：

$$P \oplus Q = Ch(bd(P) \oplus bd(Q))$$

成立。终于，这个式子说明我们可以通过只计算点或线段的明可夫斯基和来得到总体的明可夫斯基和（此时 \oplus 运算只针对于点或线段，这是可以在计算几何中直接计算的）。

更进一步说，用凸多边形的线段边来代替凸多边形本身来表示其明可夫斯基和是合理的。明可夫斯基和的边界始终可以仅由凸多边形的边界构成。

现在，我们来想象 $bd(P) \oplus bd(Q)$ 的形状。不失一般性，假设原点都位于二者的内部（否则可以引入一个确定的偏置来修正），将 $bd(P)$ 的原点对准 $bd(Q)$ ，沿着扫过一周，当其扫过 $bd(Q)$ 上的每条边时，我们都能确定其相对于 Q “外侧”的方向，有一条与当前扫过的 $bd(Q)$ 的边平行，等长的新边，即新的凸多边形必然拥有和 $bd(Q)$ 对应的边。同理，新的凸多边形也必然拥有所有 $bd(P)$ 上对应的边。而因为这个多边形最多有 $n+m$ 条边，因此我们完全确定了这个凸多边形的形状。

现在我们已经知道了这个新的凸多边形具有原来两个凸多边形的每一条边，我们便可以根据边排列的方式来构建这个新的凸多边形。具体而言，我们已知 P, Q 按顺序排列的顶点，其可视为某条边的有向起始端点。假设起始点为 $P_0 + Q_0$ ，记录这两个对应点在 P, Q 中对应的参考点。任选 P_0 或 Q_0 在 P, Q 中对应的有向边，得到明可夫斯基和的下一个点，同时在 P 或 Q 中更新当前的参考点。从 P, Q 的当前参考点所对应的下一个候选边中选择按极角排序较小(或较大)的那个，并更新对应的点。这样，最终可以在线性时间内遍历 P, Q 的所有边，并构建出明可夫斯基和的边界。引理3.1得证。

3.2. 一个凸多边形和一个非凸（简单）多边形的明可夫斯基和

首先对其进行分析。假设 P 是一个简单非凸多边形，具有 n 个顶点，而 Q 是一个具有 m 个顶点的凸多边形。明可夫斯基和 $P \oplus Q$ 仍然是一个多边形，然而，和凸的情形不同的是， $P \oplus Q$ 的边界可能具有 $\Omega(nm)$ 个顶点的最坏情况，而不是线性的 $\Omega(n+m)$ 。这即意味着此类问题显然是输出敏感型的，其时间复杂度不会低于空间复杂度 $O(mn)$ ，如果我们要将答案完整地输出的话。因此，显式地构建明可夫斯基和的代价过余昂贵。然而，我们可以通过创造一种隐式构建明可夫斯基和的方法来回避 $O(mn)$ 的复杂度。隐式构建的明可夫斯基和可以在 $O(n \log^2(n+m) + m)$ 的时间内构建出来，并仅需 $O(n+m)$ 的空间来存储。我们的方法主要分为以下三步：

- 步骤1. 对 P 进行三角剖分。令 T_1, T_2, \dots, T_s 是最终的三角剖分中的三角形 ($s = n - 2$)
- 步骤2. 计算每一个三角形和 Q 的明可夫斯基和的隐式表示 $R_i = T_i \oplus Q$, for $i = 1, 2, \dots, s$.
- 步骤3. 对于每一个三角形的明可夫斯基和 R_i ，计算总的并集 $R = \bigcup R_i$

接下来我们会解释这种构建方法的细节。在开始之前，我们做一个简单的假设：我们把参考笛卡尔坐标系固定在 Q 所在的平面空间中，并使坐标原点与 Q 的参考原点对齐。

步骤1可以容易地在 $O(n \log n)$ 甚至 $O(n)$ 时间内实现。课堂上已经证明了一种 $O(n \log n)$ 的方法。首先可以在 $O(n \log n)$ 时间内将 P 分割为数个单调多边形，再对每个单调多边形进行线性时间内的三角剖分。由于这个算法不是本工作的核心部分，在这里不再对此展开更多介绍。

我们着重介绍步骤2和3。步骤2计算了一个三角形 T_i 和凸多边形 Q 的隐式明可夫斯基和。这一步的关键是观察到由于相对简单的三角形 T_i ，这个明可夫斯基和与 Q

具有非常相似的形式，可以仅根据 T_i 和 Q 在 $\log(m)$ 时间内确定。回忆一下凸多边形对的明可夫斯基和的计算技巧：我们实际上是按照边的极角顺序融合了所有的边。对于这个问题也一样。我们需要将三角形 T_i 的三条边按极角顺序插入到 Q 中，如图1所示。

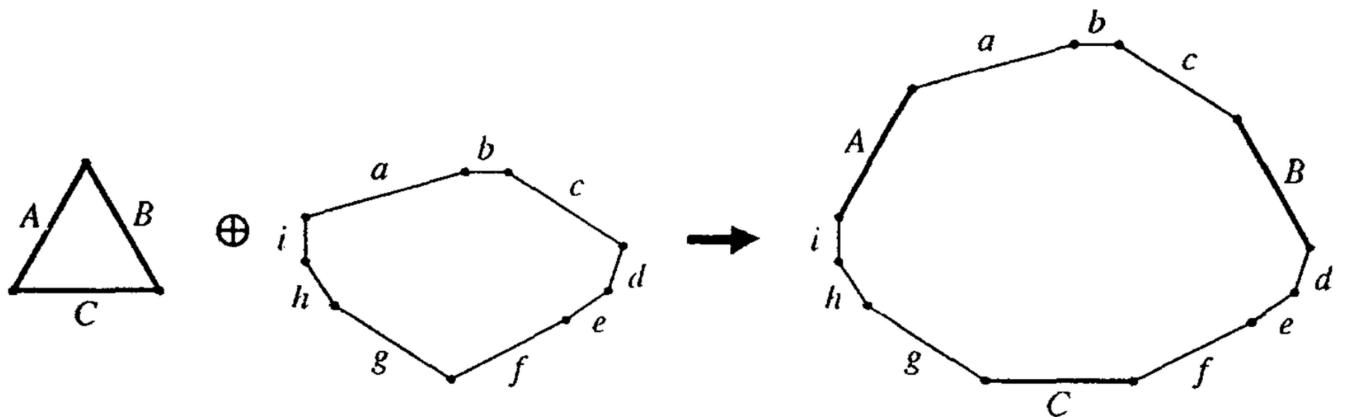


图1. 三角形 T_i 和凸多边形 Q 构成的明可夫斯基和 R_i 的直观展示。注意到另一个性质：明可夫斯基和 R_i 上每一段来自于 Q 的连续凸弧的首边和末边的极角变化量小于 180° 。

注意！在实际的算法中，为了节省空间存储的开支，我们不必真的这样做，而只是存储三角形 T_i 以及在 Q 中定位 T_i 的三条边所应当插入的位置。这就是所谓的**明可夫斯基和的隐式表示**。计算一下：如果我们对每一个三角剖分得到的 T_i 都这么做，那么我们最终只需要线性存储空间，主要是存储 Q 和每一个由 P 剖分而来的三角形。如果我们预先定义了 Q 中每条边的序号，则对于每一个具体的 T_i ，这个插入位置也可以由一个序号来表示。这一步骤需要对 T_i 的每条边查找其在 Q 中的插入位置，由于 Q 中的边是极角有序的，因此可以以二分查找的方式在 $\log(m)$ 时间内完成。

假设我们仅仅保存了三角形 T_i 中，某个顶点的绝对位置 r_i 以及剩余两个顶点的相对位置向量 a 和 b ，以及对应于 Q 中关于三角形的三条边待插的序号 i, j, k ，则我们可以根据这些数据还原明可夫斯基和按极角排序的每个顶点：

$$q_{i-1}, q_i, (q_i + a), (q_{i+1} + a), \dots, (q_j + a), (q_j + b), (q_{j+1} + b), \dots, (q_k + b), q_k, q_{k+1}, \dots, q_{i-1}$$

其中 q_i 是 Q 的第 i 个顶点的坐标，然后对每个点施加 r_i 的绝对位置偏置。这样就算完全还原了这个明可夫斯基和 R_i 。

这意味着，我们通过隐式表示能够完全存储明可夫斯基和的全部信息，而不必占用 $O(mn)$ 的存储空间。步骤2论述完毕。

步骤3即需要将我们隐式计算得到的所有 $s = n - 2$ 个凸包明可夫斯基和求并，即计算 $R = \bigcup_{i=1}^s R_i$ 。本质上，计算任意两个简单多边形求并所需的操作就是查询图形之间的所有交点对，并按照交点对的位置删去两个图形在交点之间的线段。所以，总体交点的数量确定了算法复杂度的下限。然而，为了令最终的计算复杂度低于 $O(mn)$ ，我们还需要先证明：这 s 个明可夫斯基和的并，即我们最终所求的明可夫斯基和，在其所有的顶点中，那些同时是某些小一些的明可夫斯基和的交点的数量，或称之为局部非凸的点的数量，是线性的。

如果确实是这样，我们便可以用二分归并的方法，在 $\log(s)$ ，或者也可以说 $\log(n)$ 层内，将所有 s 个 R_i 两两合并起来。每一层中，我们将每两个明可夫斯基和并为一个，处理的交点数量之和都不超过线性。那么算法的复杂度下限应该是 $(n+m)\log(n)$ （注意，实际的算法复杂度高于这个值）。

有以下两个重要的事实：

- 事实 3.2.1：

令 A_1 和 A_2 为两个不存在相交内部的凸多边形，并令 B 为另一个凸多边形。则 $A_1 \oplus B$ 和 $A_2 \oplus B$ 至多有两个交点。(原文，有一定歧义)

- 事实 3.2.2：

令 $A_i, i = 1, 2, \dots, s$ 是一个互相没有相交内部的凸多边形的集合，并且令 B 为另一个凸集。则 $R_i = \cup_{i=1}^s (A_i \oplus B)$ 拥有最多 $6s - 12$ 个局部非凸性的点(交点)。(原文，有一定歧义)

事实 3.2.1 比较容易理解，虽然其实它表述有误，没有考虑到边重合的情况。这个结论其实可以推广到更一般的描述，即把凸多边形替换为任意凸的几何形状。

下面简单地论证 3.2.1：

我们知道 A_1 和 A_2 初始时无内部交点。想象一个连续函数 $f(t)$ 描述了一个从实数 t 到几何域 $A_1 \oplus tB, A_2 \oplus tB$ 的连续映射。随着 t 的增大，这种映射必然令 $A_1 \oplus tB$ 和 $A_2 \oplus tB$ “扩张”，并且首先会使得 $A_1 \oplus tB$ 和 $A_2 \oplus tB$ 相交于至少一个点或一条边，然后分裂为两个交点(因为二者的相交集合是一个扩张的凸集)。记录初始的两个交点随着 t 增大而移动的轨迹，由于这必然是一个连续的轨迹，因此我们可以编号并追踪这些交点。

假设存在 $\epsilon_0 = \sup \epsilon$ ，其中 ϵ 满足 $\forall 0 \leq \epsilon' < \epsilon, \#|bd(A_1 \oplus \epsilon' B) \cap bd(A_2 \oplus \epsilon' B)| \leq 2$ ，也就是说 $t = \epsilon_0$ 时交点数不小于 3。设 $t = \epsilon_0$ 。如果此时边界上得到了一个新生的交点 P 为第三个交点，有两种可能：1. 交点 P 是局部凸的，则其拥有 $A_1 \oplus tB$ 和 $A_2 \oplus tB$ 的公共支撑平面；2. 交点 P 是局部非凸的，即 $bd(A_1 \oplus tB)$ 和 $bd(A_2 \oplus tB)$ 发生了交叉。注意，对于第二种情况来说不存在一条直线可以沿 P 完全分割 $A_1 \oplus tB$ 和 $A_2 \oplus tB$ 的内部，否则交点 P 必定是第一个交点。

此外，如果在点 P 的局部范围内 A_1 和 A_2 只在 P 点重合，那么易知 $(A_1 \oplus tB)$ 和 $(A_2 \oplus tB)$ 仅仅相交于点 P 一点，和假设矛盾。

对于第一种可能，可以反过来推算出交点

$P \in bd(A_1 \oplus tB) \cap bd(A_2 \oplus tB)$ 中所对应的 $p = a_1 + tb_1$ 和 $p = a_2 + tb_2$ 中的 a, b ，这是由于明可夫斯基和在公共支撑平面上所具有的特性所决定的。

- 对于 P 上的任意一个公共支撑平面 L ，必然有 $b_1, b_2 \in bd(B)$ 且 b_1, b_2 为 B 在 L 法向上投影最小值的点。如果 L 不与 $bd(B)$ 中的任意一条边平行（一般而言很容易做到，因为一般一个具有局部凸性的交点 P 上存在无穷多的公共支撑平面），则这个点 b 必定有 $b \in bd(B)$ 且唯一。从而我们确定了 $b_1 = b_2 = b$ ，进而确定了 $a_1 = a_2$ 。但这也意味着一个相同的 a 同时属于 A_1, A_2 ，且它在 A_1 和 A_2 中也对应相同的支撑平面 L 。这也意味着 a 是同时属于 $bd(A_1)$ 和 $bd(A_2)$ 中在支撑平面 L 的法向上投影最小值的点，且必然可以为某个角点。然而，这样的角点在 $t = 0$ 时就已经导致了相交，因此它必定是第一个或第二个交点，而不可能是新生的交点，与假设矛盾。

对于第二种可能，如果 P 是局部非凸的交点(交叉的交点)，显然在足够小的 δ 范围内，任意 $t \in (\epsilon_0 - \delta, \epsilon_0 + \delta)$ 都不会导致这个交点在拓扑学的意义上消失。则这同样与假设 $\epsilon_0 = \sup \epsilon$ 矛盾。因此只存在两种可能：一种可能是不存在这样的 ϵ_0 ，另一种是 ϵ_0 上无交点。但这其实是同一种可能性。因此，我们证明了任意两个无内部相交的凸图形分别与另外的同一个凸图形的明可夫斯基和的边界最多有两个交点。

如果事实 3.2.1 是正确的，那么 3.2.2乍一看会比较令人疑惑。因为 3.2.1 可以视为 3.2.2 在 $s = 2$ 上的一个推论，但两者的结论似乎冲突了，因为 $6s - 12$ 在 $s = 2$ 时交点数应该为 0。事实上，这个公式在表达上不够精准。其更精细也更一般的正确表述如下：

- 事实 3.2.3 (3.2.2 的再表述)：

假设 $\Gamma = \{\gamma_i\}_{i=1}^n$ 是一个弱可接受的 (weakly admissible) 简单闭合的若尔当曲线的集合。定义 $I(\Gamma) = \cup_{i \neq j} (\gamma_i \cap \gamma_j)$ ，以及 $E(\Gamma) = I(\Gamma) \cap bdK(\Gamma)$ 。如果 $n \geq 3$ ，则 $\#E(\Gamma) \leq 6n - 12$ 。更进一步，对于任意的 $n \geq 3$ ，存在这样的可接受的集合 Γ_n 使得 $\#E(\Gamma_n) = 6n - 12$ 。这里 $K(\Gamma) = \cup_{i=1}^n K(\gamma_i)$ 表示对于一个闭合曲线的集合施加运算，即对集合中每个元素的内部取闭包的并集。

需要说明：“弱可接受”的意思是任意两个简单闭合的若尔当曲线之间的交点数不大于 2。这里的“弱可接受的简单闭合的若尔当曲线的集合”可以视为一群相互无内部相交的凸多边形的更一般性情况。这些凸多边形的凸性意味着任意两个凸多边形的交点数量不大于 2。 $\#E(\Gamma)$ 是 $bdK(\Gamma)$ 上的交点数量， $\#$ 代表求集合的元素数量。

在证明这个事实之前，我们先回顾一下我们的目标，是为了证明总的明可夫斯基和中非凸的点，也即是那些由小一些的明可夫斯基和相交并且保留到最后而得到的顶点的数量，是线性的。注意，这并不意味着全体的 R_i 的交点数量是线性的。容易举出一些具体的例子证明在最坏情况下全体 R_i 的交点是 $O(n^2)$ 的。

接下来我们证明 3.2.3。

不失一般性，假设我们的若尔当闭合曲线 γ_i 具有一般性的位置，即，它们没有三个或以上的曲线段交于同一点。否则，我们总可以稍稍改变一些 γ_i 的形状，使得在不改变 n 的情况下让 $\#E(\Gamma)$ 增大。

我们声称，一个闭合曲线 γ_i 是多余(redundant)的，当 $\gamma_i \subset \cup_{j \neq i} K(\gamma_j)$ ，即 γ_i 完全被其他的曲线的并的内部包裹住了。令

$$\begin{aligned} r_1(\Gamma) &= \#\{\text{redundant curves in } \Gamma\} \\ r_2(\Gamma) &= \#\{(i, j) | i \neq j, \gamma_i \cap \gamma_j \neq \emptyset\} \\ r_3(\Gamma) &= \#\{(i, j, k) | i, j, k \text{ are distinct}, K(\gamma_i) \cap K(\gamma_j) \cap K(\gamma_k) \neq \emptyset\} \end{aligned}$$

以 Γ 的字典序引入上述四元组(为什么不是三元组?) $r(\Gamma) = [r_i(\Gamma)]_{i=1}^3$ 来继续我们的证明。我们假设下面讨论的, 在给定的 n 下, 能使得 $E(\Gamma)$ 最大化的 Γ 满足仅有 $r_2(\Gamma)$ 非 0。

这是因为, 对于 $r_1(\Gamma)$ 或 $r_3(\Gamma)$ 非 0 的情形, 显然 Γ 中有一些多余的 γ_i (对应 $r_1(\Gamma) \neq 0$), 或者有三个或以上的 $\gamma_{i,j,k}$ 具有共同的交点(对应 $r_3(\Gamma) \neq 0$)。任意一种情形都可以在删除或退化一些 γ_i 的情况下, 使得新的 Γ' 满足在字典序中满足 $r(\Gamma') < r(\Gamma)$, 而总的 $\#E(\Gamma') \geq r(\Gamma)$ 。换句话说, 删除或调整掉一些不必要的结构, 不会减少最终的 $\#E(\Gamma)$ 。在 Γ 中, 任何多余的 γ_i 或者 三个以上图形的共同交对于 $\#E(\Gamma)$ 都没有任何贡献。

接下来, 在证明 3.2.3 之前, 我们先证明其更特殊的一个版本:

- 命题 3.2.4:

假设 $\Gamma = \{\gamma_i\}_{i=1}^n$ 是一个具有一般性位置的可接受的曲线集合, 并且 $r_1(\Gamma) = r_3(\Gamma) = 0$ 。则 $n \geq 2$ 时 $\#E(\Gamma) \leq 6n - 12$ 。

实际上这个假设对应一种较为简单的拓扑情形, 让问题变得简化。

注意到, 在这个假设下, 我们实际上有 $E(\Gamma) = I(\Gamma)$, 这是因为任意三个或以上的若尔当闭合曲线都不存在交集时, 所有的曲线交点必然在 $Kbd(\Gamma)$ 的边界上。对于任意的 $i \neq j$, 定义集合 $I_{ij} = \gamma_i \cap K(\gamma_j)$ 必定是一个闭的, 联通的, 不与其他任何结构相交的一段完整的弧 (也可能是空集)。这是由“弱可接受性”所定义的。对于任意的 i , 令

$$D_i = K(\gamma_i) - \cup_{j \neq i} (int \gamma_j)$$

则 D_i 是非空集合, $bd(D_i)$ 是一个简单封闭的若尔当闭合曲线。事实上,

$$bdD_i = \gamma_i - (\cup_{j \neq i} I_{ij}) \cup (\cup_{j \neq i} I_{ji})$$

对于每一个 i , 选择一个任意的点 $p_i \in int D_i$, 且对于所有 $j \neq i$, 如果 $\gamma_j \cap \gamma_i \neq \emptyset$, 选择任意一个非端点的点 $q_{ij} \in I_{ji}$ 。然后我们可以用分段无相交的简单弧 α_{ij} 将 p_i 连接到所有的 q_{ij} , 并用存在于 $I_{ij} \cup I_{ji}$ 的内部的简单弧 $\beta_{ij} = \beta_{ji}$ 将 q_{ij} 和 q_{ji} 连接起来。最终的连接拓扑关系如图2所示。

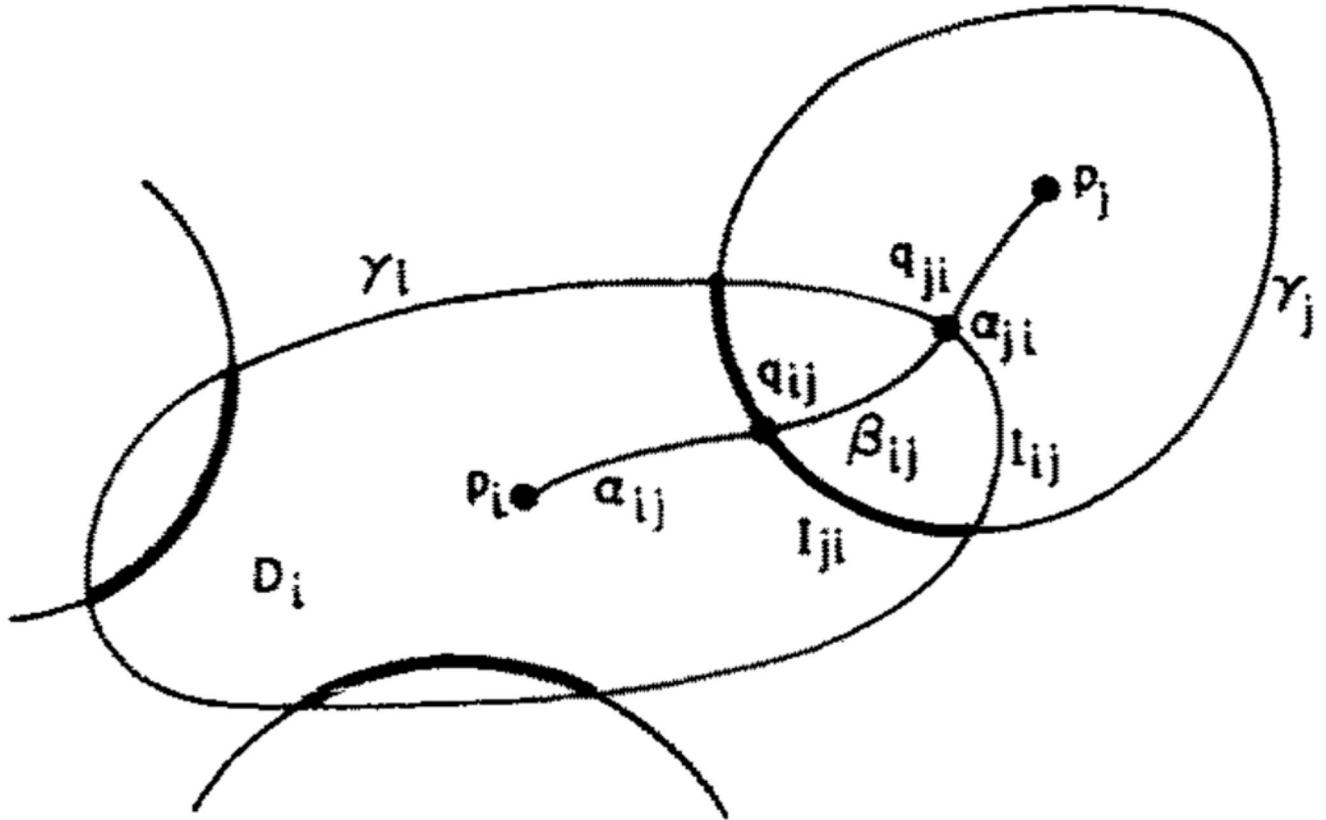


图2. $G(\Gamma)$ 中, p_i, p_j, q_{ij}, q_{ji} 之间的拓扑连接图示。

以这种方式, 我们获得了一个图 $G(\Gamma)$, 定义其顶点为所有满足

“所有的 $i \neq j$ 有 $\gamma_i \cap \gamma_j \neq \emptyset$ ” 的点 p_i 。 $G(\Gamma)$ 包含了边 $\alpha_{ij} \cup \beta_{ij} \cup \alpha_{ji}$, 其连接了顶点对 (p_i, p_j) 。可以看到, 边 $\alpha_{ij} \cup \beta_{ij} \cup \alpha_{ji}$ 依次经过点 p_i , 弧 I_{ji} , 弧 I_{ij} 和点 p_j 。显然, 所有这样的边 $\alpha_{ij} \cup \beta_{ij} \cup \alpha_{ji}$ 可以不会彼此相交, 因为对于同一个 i , 所有 α_{ij} 可以不必相交。而所有不同的 β_{ij} 必定不会互相相交或者与角标不同的 α_{ij} 相交, 因为它们位于互不相交的 $K(I_{ij} \cup I_{ji})$ 之中。

最后, 由于 弱可接受性 的定义, 任意 p_i, p_j 之间至多有一条边。

上述讨论是为了明确 $G(\Gamma)$ 是一个平面图。因此, 基于平面图的欧拉公式, 我们有:

$$e(G(\Gamma)) \leq 3v(G(\Gamma)) - 6$$

其中 $e(G(\Gamma))$ 和 $v(G(\Gamma))$ 分别为 $G(\Gamma)$ 中边和点的数量。注意到 $v(G(\Gamma)) = n$ 和 $e(G(\Gamma)) = \frac{1}{2}\#E(\Gamma)$ ，则命题3.2.4成立。

现在我们继续证明3.2.3。实际上之前在引入字典序时，我们已经论述过，对于 n 个若尔当闭合曲线构成的集合 Γ ，在

$$r_1(\Gamma) \neq 0 \text{ 或 } r_3(\Gamma) \neq 0$$

时，总能够通过删除或变形一些 γ_i 的方式，在字典序中找到一些 Γ' ，使得 $\#E(\Gamma') \geq \#E(\Gamma)$ 同时 $r(\Gamma') < r(\Gamma)$ 。通过这样的方式得到的 Γ' 对应的若尔当闭合曲线数量 $n' \leq n$ （如果 $n' < n$ ，我们也能够对 Γ' 任意添加一些若尔当曲线使得 $n' = n$ ）。也就是说，对于固定 n 下，使得 $\#E(\Gamma)$ 最大的 Γ 必然满足 $r_1(\Gamma) = r_3(\Gamma) = 0$ 。则可以由3.2.4 推出3.2.3。从而完成了这部分的证明。

最后，我们证明归并算法所涉及到的交点数目的整体复杂性。此复杂性上限与任意一层合并算法中输入的任意一个多边形的大小及其是否带孔无关。

任意 s 个作为凸多边形的单位明可夫斯基和 R_1, R_2, \dots, R_s ，由于其原始对应的 P 的三角剖分 T_1, T_2, \dots, T_s 无内部交点，其最终的并，即 $\cup_{i=1}^s R_i$ 的局部非凸点的数量不大于 $6s - 12$ 。假设我们通过 $\log(s)$ 层归并对 R_1, R_2, \dots, R_s 求交，在第 $k + 1$ 层中，共有不超过 $\frac{s}{2^k}$ 个封闭图形，每个图形是 2^k 个不同的单位明可夫斯基和的并。**在这一层中，任意两个封闭图形之间求并而生成的交点必然属于这两个图形自身所对应的 $2 * 2^k$ 个单位明可夫斯基和的所有 边界上的交点，即交点总数不会多于 $6(2^k) - 12$ 个。** 因此每层中所有新产生的交点数量必然小于 $6s$ 个。进而，在 $\log(s)$ 次归并的过程中总体处理的交点数量不会高于 $6s * \log(s)$ 个。

3.3. 归并算法

回顾前文内容，我们已经介绍了凸包的明可夫斯基和及其计算复杂度，以及凸包与非凸包的明可夫斯基和中所需要处理的交点数量的最高复杂度。然而，原文中并未直接给出任何的算法步骤，以及完整证明算法本身的复杂度。我们将会补充这一点。回顾前述的算法步骤：

- 步骤1. 对 P 进行三角剖分。令 T_1, T_2, \dots, T_s 是最终的三角剖分中的三角形 ($s = n - 2$)
- 步骤2. 计算每一个三角形和 Q 的明可夫斯基和的隐式表示 $R_i = T_i \oplus Q$, for $i = 1, 2, \dots, s$.
- 步骤3. 对于每一个三角形的明可夫斯基和 R_i ，计算总的并集 $R = \cup R_i$ 。

步骤1是三角剖分，可在 $n\log(n)$ 时间内求解。步骤2对每个小三角形计算 $R_i = T_i \oplus Q$ 的隐式表示，总体也可在 $n\log(m)$ 时间内完成。对于步骤3，我们仅证明了可以通过 $\log(n)$ 层求并实现，且每层求并的总体交点数为 $6s$ 个， $s \leq n$ 。但我们并未证明，每一层中，完成求并算法的整体复杂度。接下来我们来介绍这种算法。

扫描线算法

本文中使用的扫描线算法是一种修改以适应凸弧作为segment的情形的 Ottman 扫描线算法的特殊变体。我们知道，对于一个具有 n 个线段、 m 个交点的问题，用于寻找所有交点的 Ottman 扫描线算法可以在 $(n + m)\log(n)$ 时间内求解。这是因为在扫描线扫描过程中，将每个线段的两个端点以及两个线段的交点作为事件。在每个事件上查找可能的交点，并通过平衡树结构重新排列空间上相邻线段的相对位置。平衡树的维护需要 $\log(n)$ 的时间。

因此，如果我们需要令扫描线算法能够处理具有凸弧的情形，首先需要该凸弧对于扫描线移动方向（如x轴）能够构成单调关系。即，凸弧沿自身方向在扫描线移动方向的投影必须是一一映射的。这同样是在说，一段凸弧，默认从左到右方向，要么完全凸，要么完全凹。

实现这一点可以在常数时间内完成。因为我们知道所有的凸弧都来自于凸多边形 Q 。我们可以用线性时间 $\log(m)$ 对 Q 做预处理，来查找其最左和最右的端点，设为 q_l 和 q_r 。因此，任意给一段隐式表示的 Q 上凸弧段，我们可以在常数时间内判断 q_l 和 q_r 是否是该凸弧的内点。若不是，则这段凸弧对x轴本来就是单调的。否则，则可以从 q_l 或 q_r 将其分解为两个对x轴单调的凸弧段。此外，预处理需要记录 Q 上每个端点的坐标，从而令我们可以在常数时间内知道任意一段凸弧在扫描线前进方向上的投影长度。**注意，我们下面所说的凸弧都默认是扫描线前进方向(x轴)上的函数。**

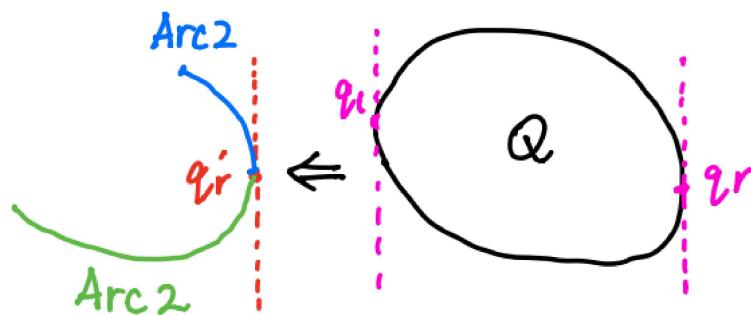


图3. 任意一段来自于凸包 Q 上的凸弧可以在常数时间内分解为至多2段子弧，使得所有子弧可以视作x方向上的函数。

如果所有的segment(即凸弧和线段)都可表示为x轴上的函数，那么我们仍然可以完全沿用扫描线算法的结构，维护一个包含凸弧和线段的平衡树来表示所有segment之间的相对位置。注意以下细节：在每个“事件”位置上，可能多条凸弧会被扫描线截为左右两侧，但我们只会处理那些与当前“事件”有关的凸弧（不超过两个），截取其扫描线右侧的部分，在 $\log(m)$ 时间内完成。

平衡树的维护代价仍然是 $\log(n)$ ，但和原版不同的是，在此时计算两个segment交点所需的复杂度不再是线段求交的常数复杂度。

- 事实 3.3.1：在变体的 Ottman 扫描线算法中，计算指定一段凸弧和一条线段的交点的时间复杂度为 $\log(m)$ 。

- 事实 3.3.2: 在变体的 Ottman 扫描线算法中, 计算指定两段凸弧的交点的时间复杂度(最多)为 $\log^2(m)$ 。

事实 3.3.1 的证明。需要注意的是, 一段凸弧和一条线段最多可能有两个交点, 但原版Ottman 扫描线算法仅关注前进方向上的第一个交点。这不会对算法带来本质上的影响, 我们可以通过一些简单的标记来避免后续的重复运算。假设简化这个问题, 即不考虑一个segment刚好截断另一个, 或二者“相切”的情况。

不失一般性, 假设凸弧在当前扫描线上位于指定线段的下方。

- 3.3.1.1 凸弧开口向上: 最多只能有一个交点。由于我们可以在常数时间内计算任意一段凸弧在x轴上的定义域, 因此我们可以通过二分查找, 在最多 $\log(m)$ 时间内确定凸弧和线段在x轴上的公共定义域。如果确定了公共定义域, 那么这个问题就变成了简化的问题:
 - 在一个公共定义域上, 判断并求解一个 C^0 连续函数的零点, 这个连续函数为凸弧与指定线段在y方向上的差。已知最多存在一个零点。
 这个问题非常简单, 由连续性定理, 首先计算函数两端的正负性。若同号则必然无交点, 若异号则必然有一交点。然后用二分方法查找交点即可, 总体时间复杂度为 $\log(m)$ 。
- 3.3.1.2 凸弧开口向下: 可能有0个, 1个或2个交点。类似地, 我们可以先在 $\log(m)$ 时间内确定凸弧和线段在x轴上的公共定义域。这个问题可以不经任何处理, 简化为另一个问题:
 - 在一个公共定义域上, 判断并求解一个一阶差分单调的 C^0 连续函数的零点, 这个连续函数为凸弧与指定线段在y方向上的差。已知存在0个, 1个或2个零点。

由于这个函数一阶差分单调的性质, 我们可以在 $\log(m)$ 时间内求出这个函数的最高点或最低点。由于这个函数在起始位置小于0 (因为凸弧在指定线段下方), 如果最高点大于0, 则存在1-2个交点。此时将凸弧从最高点对应的x坐标处分为左右两段, 则左侧凸弧与线段必然有1交点, 右侧凸弧与线段可能有0个或1个交点。然后, 类似通过 3.3.2.1 中的方法, 在 $\log(m)$ 的时间内分别判断和计算两侧的交点。如果最高点等于0, 则凸弧和线段刚好相切或截断。如果最高点小于0, 则无交点。后二者都可以在常数时间内结束运算。

从而, 我们证明了计算指定一段凸弧和一条线段的交点的时间复杂度为 $\log(m)$ 。

事实 3.3.2 的证明。一般而言, 两段复杂度都为 m 的凸弧至少需要线性时间来判断交点, 但在这个问题中是特殊的情况。注意到, 在计算任意两个明可夫斯基和的求并过程中, 这两个几何形上的“凸弧”都完全来自于原始的三角形的明可夫斯基和 R_i , 进一步说, 都属于凸包 Q 的一部分。也就是说, 我们计算的凸弧和凸弧求交问题, 并不是任意的凸弧和凸弧求交的问题, 而是如下的特殊问题:

- 已知凸多边形 Q 上所有端点的坐标及其排列顺序。给定任意两段用首末点序号表示的凸弧段 \widehat{AB} 和 \widehat{CD} , 其二者都是由 Q 上的两条已知凸链分别施加线性位移 r_{ab} 和 r_{cd} 得到的, 且随x轴单调。判断并计算其交点位置。

首先, 由于 $\widehat{AB} \in R_i$ 和 $\widehat{CD} \in R_j$, 根据事实 3.2.1, 我们知道 \widehat{AB} 和 \widehat{CD} 最多只有两个交点。类似于凸弧和线段求交的情况, 这里也需要分情况讨论。假设 \widehat{AB} 和 \widehat{CD} 分别对应于原 Q 上的非空凸链 $Arc1$ 和 $Arc2$ 。有3种可能性:

1. $Arc1 \subseteq Arc2$ 或 $Arc2 \subseteq Arc1$
2. $Arc1 \cap Arc2 = \emptyset$
3. $Arc1 \cap Arc2 \neq \emptyset$, $(Arc1 \cap Arc2) \subset Arc1$, $(Arc1 \cap Arc2) \subset Arc2$

对这三种可能性的判断可在常数时间内完成, 因为我们完全了解凸包 Q 的信息。类比一下, 凸包 Q 就像一个时钟, $Arc1$ 和 $Arc2$ 是用始末时刻分别表示的两个时间段。我们可以在常数时间内判断和计算两个时间段的彼此包含关系。对于第3种情况, 我们可以在常数时间内将 $Arc1$ 以 $Arc2$ 的某个端点为切分点分割为两段 $Arc3$ 和 $Arc4$, 使得 $Arc2$ 与 $Arc3$, $Arc2$ 与 $Arc4$ 的相对关系必然满足情况1或2。因此, 我们实际上只需要关注前两种情况。

对于情况1或2, 我们仍然可以参考前面的方法, 在 $\log(m)$ 时间内先求出两段凸弧的公共x轴定义域。

- $Arc1 \subseteq Arc2$ 或 $Arc2 \subseteq Arc1$ 。则这两段凸弧必然具有相同的凸性。不妨设 $Arc1 \subseteq Arc2$ 。请想象一下, 一个与x轴具有一一映射关系的凸链 \widehat{AB} , 与经过某个已知线性位移 $-r_{ab} + r_{cd}$ 的其自身的某个部分 \widehat{CD} , 最多只能有一个交点。更广泛来说, 对一个有限定义域上的连续凸函数 $f(x)$, 构造新函数 $f(x+a)+b$, 则二者之差 $f(x+a)+b-f(x)$ 具有单调性, 对应于几何关系中至多一个交点的特性。
我们可以通过判断定义域两端二者之差来预先确定是否存在交点。如果两端异号, 必然存在一个交点, 如果两端同号则必然无交点。存在交点时, 可以用二分法查找交点的位置。

◦ 事实 3.3.3:

对于上述两段复杂度不超过 m 且存在一个交点的有序凸链, 给定算法可以在 $\log^2(m)$ 时间内求出交点位置。

在数学上, 上述两段凸链之差是一个最多 $2m$ 段的分段线性函数, 且具有保凸或保凹的特性和唯一零点。然而, 我们无法简单地在 $\log(m)$ 内将其求解的原因是我们无法在常数时间内获知一段凸链上任意x位置对应的y坐标 (除非我们预先设置了快速求值结构)。因此, 这里需要两重二分的策略。假设我们所求为满足上述性质的凸链 \widehat{AB} 和凸链 \widehat{CD} 的交点。

- 算法 3.3.4: 已知两段相同x轴定义域的凸弧有且仅有一个交点, 求其唯一交点。

- 给定查找域 $[l, r]$, 最多对数时间 $\log(m)$ 内判断 (不失一般性): 在 l 上 \widehat{AB} 低于 \widehat{CD} , 在 r 上凸链 \widehat{AB} 高于 \widehat{CD} 。
- 若当前查找域的x轴范围不能够被同时包含于凸链 \widehat{AB} 和凸链 \widehat{CD} 的一对边上, 重复:

- 常数时间内，查找 \widehat{AB} 的中间顶点的x,y坐标。
- 根据x坐标， $\log(m)$ 时间内，查找 \widehat{CD} 上范围包含给定x坐标的一条边。
- 在这条边上计算x坐标所对应的y。
- 判断：在此x坐标上，凸链 \widehat{AB} 和凸链 \widehat{CD} 的相对高低。
- 判断交点可能在x坐标的左侧还是右侧。将查找域减小为 $[l, x]$ 或 $[x, r]$ ，常数时间内将凸弧 \widehat{AB} ， \widehat{CD} 沿新的定义域裁剪。

上述算法调用了双层二分结构，因此具有 $\log^2(m)$ 的总体复杂度。当然，实际上内层循环的二分查找的对数复杂度在系数上一般要低于外层。

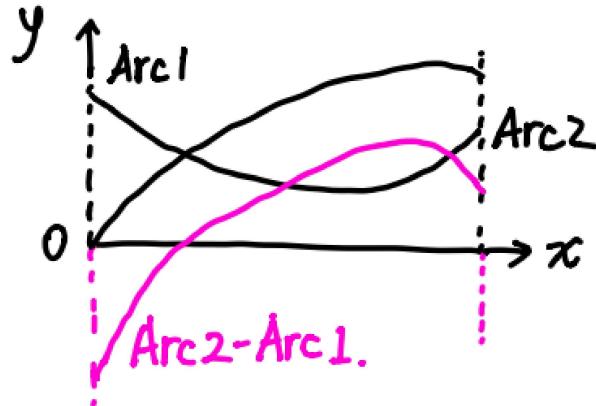


图4. 算法3.3.4。任意两段相同定义域的凸弧 Arc1 和 Arc2 有且仅有一个交点，则基于连续性，用二分查找的方式查找高度差的零点。注意到高度差只是一个隐含的表示，因此，在 Arc1 上二分查找下一个高度差的求值点的x坐标，获得求值点x坐标后，在 Arc2 上用二分查找来计算出该点的高度差。

- $Arc1 \cap Arc2 = \emptyset$ 。此时，两段凸链在x轴上的高度之差不再具有单调性。继续分为两类情形：

- $Arc1$ 和 $Arc2$ 具有相同凸性。
- $Arc1$ 和 $Arc2$ 具有相异凸性。

对于第一类情形，如果 $Arc1$ 和 $Arc2$ 具有相同凸性且二者在原始的凸包 Q 上没有任何交集，由于凸包 Q 自身的边必然是极角有序的，则 $Arc1$ 的所有边的极角必定全部大于或小于 $Arc2$ 的所有边的极角。也就是说，如果将 \widehat{AB} 和 \widehat{CD} 视为x轴上的函数 $AB(x)$ 和 $CD(x)$ ，则二者之差 $F(x) = AB(x) - CD(x)$ 在任意非节点上的导数都是同号的。也就是说两段弧的高度之差仍然是单调的，二者最多有一个交点，对应两段弧的两端高度差异号的情形。那么，我们仍然可以用算法 3.3.4 来求解这一问题。

对于第二类情形，那么 \widehat{AB} 和 \widehat{CD} 可能具有两个交点。不失一般性，假设 \widehat{AB} 开口向上， \widehat{CD} 开口向下。首先我们需要判断交点的数量。在二者共同定义域的左右端点上，常数时间内判断：

- 两端点上均有 \widehat{AB} 高于 \widehat{CD} ，可能有0个，1个或两个交点。1个交点必然对应相切情形。
- 在一侧 \widehat{AB} 高于 \widehat{CD} ，另一侧 \widehat{AB} 低于 \widehat{CD} 。只能有一个交点。
- 两端点上均有 \widehat{AB} 低于 \widehat{CD} ，无交点。

最后一种情况即不必求交点。第二种情况也可以通过算法 3.3.4 寻找交点。第一种情况下可能有两个交点，因此必须将其转化为只有最多一个交点的情形。

- 算法 3.3.5：相异凸性的凸弧分割问题。

- 给定两段凸性相异，完全排序的凸弧 X 和 Y 。二者都可视为x轴上的函数。将其分别分割为除顶点外不相交的子弧 $X = X_1 \cup X_2$, $Y = Y_1 \cup Y_2$ ，使得 X_1 与 Y_1 , X_2 与 Y_2 最多只有一个交点。

由于 X 和 Y 二者凸性相反，其在y方向上高度之差必定为凸函数。只要可以求这个凸函数的极值点，那么将凸弧 X 和 Y 各自在极值点上分开，两对子弧必然至多相交于一点。**因此，定义 $F(x)$ 为凸弧 X 和 Y 的高度差函数，但我们不知道其解析表达式。**以下算法即求函数 $F(x) = X(x) - Y(x)$ 的极值点。

不失一般性，假设 X 开口向上， Y 开口向下。目标即求凸函数 $F(x) = X(x) - Y(x)$ 的极小值点。仍然使用双重二分查找的方法：

- 给定查找域 $[l, r]$ ，计算 $F(x)$ 在公共定义域端点上的**广义梯度**，可以在常数时间内完成。
- 若广义梯度不包含0点，且当前查找域的x轴范围不能够被同时包含于凸链 \widehat{AB} 和凸链 \widehat{CD} 的一对边上，重复：
 - 在常数时间内查找凸弧 X 的中间顶点的x,y坐标。
 - 根据x坐标，在 $\log(m)$ 对数时间内获得凸弧 Y 在x处的值以及对应的边。计算 $F(x)$ 在该点上的广义梯度。

- 若x位置的广义梯度包含0，则x点为 $F(x)$ 的极小值点，记录x，结束。
- 若x位置的广义梯度不包含0，根据查找域左，右端点的广义梯度，判断极小值点可能在x的左侧或右侧。将查找域缩小为 $[l, x]$ 或 $[x, r]$ ，常数时间内将凸弧 \widehat{AB} , \widehat{CD} 沿新的定义域裁剪。

这样，我们可以在 $\log^2(m)$ 的双对数时间内得到凸函数 $F(x)$ 在公共定义域上的极值点（极值点可能在公共定义域的端点上）。通过常数时间，可判断是没有交点或是交于两点。若交于两点，将凸弧 X 和 Y 各自沿极值点分割，得到的每一对子弧 X_1 与 Y_1 , X_2 与 Y_2 有且仅有一个交点。然后，可以继续套用算法 3.3.4 来在双对数时间内求出交点的坐标。

综上，我们证明了一定条件下的两对凸弧求交的问题可以在 $\log^2(m)$ 的时间内求解。则对于本算法中以扫描线算法对任意两个各自具有最多 n 个凸弧和线段的多边形的求并问题，总体时间复杂度为：

$O(n + n)(\text{事件数量} + \text{交点数量}) * [O(\log^2(m)) (\text{计算交点的最高复杂度}) + O(\log(n)) (\text{维护平衡树的成本})]$ 。从而，经过 $\log(n)$ 次归并，求交算法的整体复杂度为 $O(n \log^2(m) \log(n))$ 。整体算法复杂度为 $O(n \log^2(m) \log(n) + m)$ ，这里 m 为对 Q 进行的预处理。

预填充快速求值结构

在我们所参考的论文中，对该算法的复杂度描述是 $O(n \log^2(m + n) + m)$ ，即比我们所推出的复杂度 $O(n \log^2(m) \log(n) + m)$ 要降低一个 m 的对数。这主要是由于该论文中不加证明地断言本文中关于两段凸弧相交的算法的复杂度为一次对数 $O(\log(m))$ ，而我们提出的算法 3.3.4 以及算法 3.3.5 的复杂度却高达 $O(\log^2(m))$ 。

这可能导致令人困惑的矛盾；但我们必须进一步说明，在通常情况下算法 3.3.4 的复杂度确实可以近似为一次对数 $O(\log(m))$ ，从而使得总体复杂度进一步下降。

我们之所以在算法 3.3.4 和 3.3.5 中出现 $\log^2(m)$ 的复杂度，主要是由于对于两个指定复杂度为 m 的凸弧段 $Arc1$ 和 $Arc2$ ，其二者之差 $Arc1 - Arc2$ 作为一个x轴上的函数，在任意给定x坐标的情况下无法在常数时间内求解。但事实上，通过对 Q 的预处理，能使得这个求解过程近似于常数。问题的核心在于：

- 给定一个来自于 Q 上复杂度为 $O(m)$ 的凸弧段，其为 Q 的上弧段或下弧段(以 Q 的左右端点 q_l 和 q_r 为划分)，可作为x上具有分段解析表达式的函数。给定x，如何在常数时间内查找其从属的弧段中的边的序号。

对数复杂度主要体现在我们需要寻找坐标x所对应的边上。然而，如果该凸弧被x方向均匀分解为单独的线段，我们就可以常数时间内直接获取对应的线段表达式，然后计算高度。我们需要设计一个数组结构来加速分段的查找过程。下面我们提供一种可能的方案，读者也可以自己思考是否存在更优的方法。

查找表 LUT(Lookup Table)

- 我们只需要对 Q 中的上、下弧段(以 Q 的左右端点 q_l 和 q_r 为划分)分别做一次预处理即可。

类似于“网格加密”方法，将弧段沿x轴分割为总计 s 段，每段长度均为 h 来建立一个空间上均匀分布的查找表。这需要花费 $O(s)$ 的时间。需要保证以下要求成立：每个长度为 h 的查找段上，不得具有 Q 上数量超过常数 α 的端点。也就是说，每个长度为 h 的查找段不得对应 Q 的超过 $\alpha + 1$ 条以上的边。

如果 $\alpha = 1$ ，一般而言必然有 $s > m$ ，因为 h 基本要求与弧段中每条边对应的x轴宽度的最小值等长。在比较坏的情况下 s 数量级可能和 m 的平方相当。但如果我们将 α 适当放大， h 只需要不大于任意 α 个连续的凸弧边对应的x轴宽度即可。对于正 m 边形这类结构，如果设 $\alpha = [km] + 1$ ，其中 k 为一个非常小的系数，如 0.01，则查找表的段落数 s 基本可视为 m 的 $1/k$ 倍。

因此，通过构建这样一个查找表，我们计算任何一个给定坐标x在弧段上对应的高度 y 可以由三步达成：1，在常数时间内确定它在查找表中的分段；2，在 $\log(km)$ 时间内确定其具体属于哪一条边；3，根据边的解析式，在常数时间内求解高度 y 。如果能够获得一个足够小的 k ，则这三步可以近似视作在常数时间内完成。

4. 可视化程序

基于上述的理论推导，我们开发了一个基于 Qt 的交互式可视化工具，可以求解以下问题：

- 凸多边形 + 凸多边形 的明可夫斯基和计算
- 非凸多边形 + 凸多边形 的三角剖分 + 分别求和 + 合并（含空洞处理）
- 鼠标点击输入顶点，回车完成多边形

其中在非凸多边形 + 凸多边形 的合并任务中，调用了外部库来保障程序稳定性。虽然我们在理论部分 3.3 中提供的算法不存在复杂度上的问题，但在实际计算过程中，难以避免遇到非简单多边形（带空洞）的情况，这会导致更复杂的维护逻辑。下面几幅图展示了几个算例。

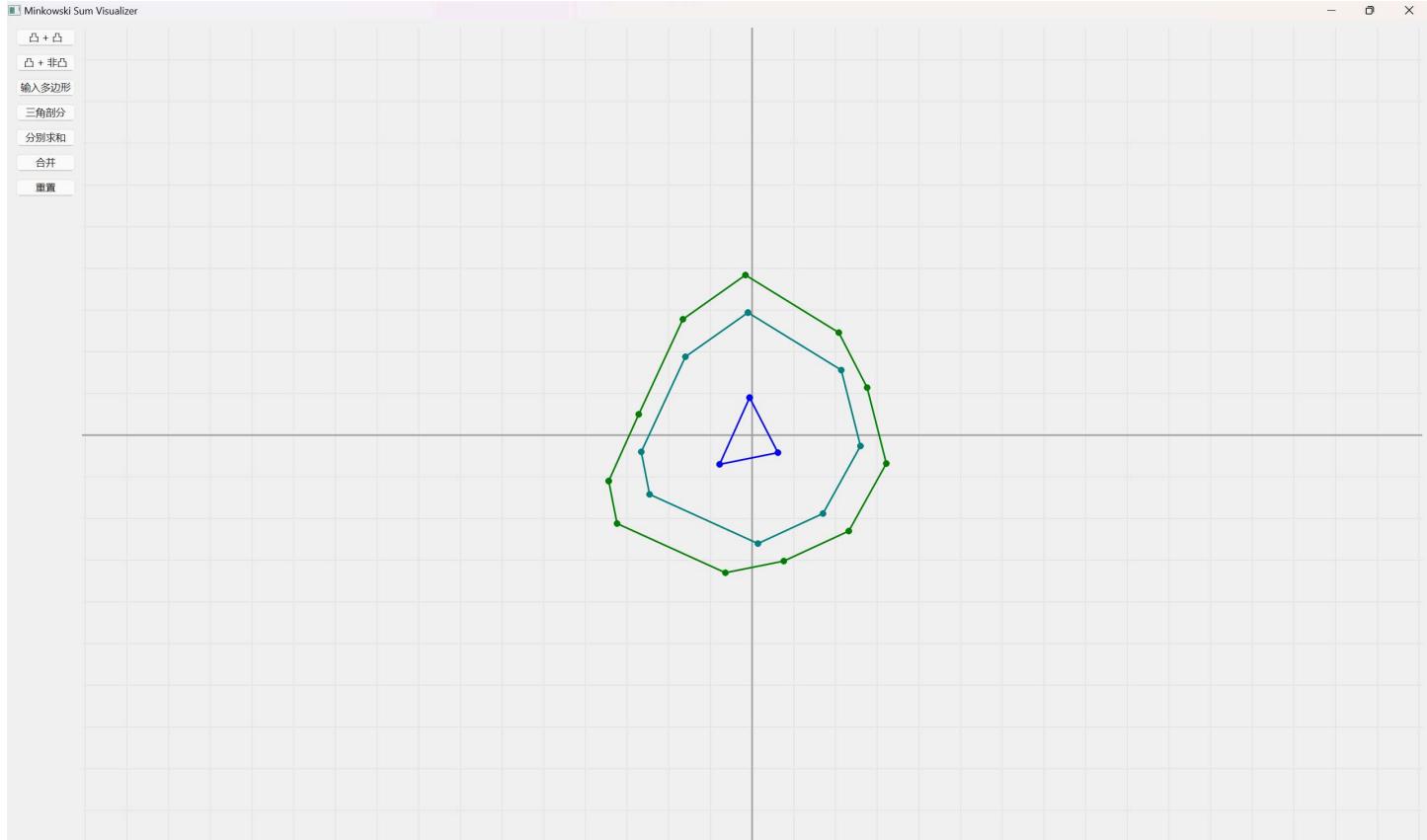


图5. 凸多边形(蓝色三角形)和凸多边形(青色八边形)的明可夫斯基和(绿色十一边形)。

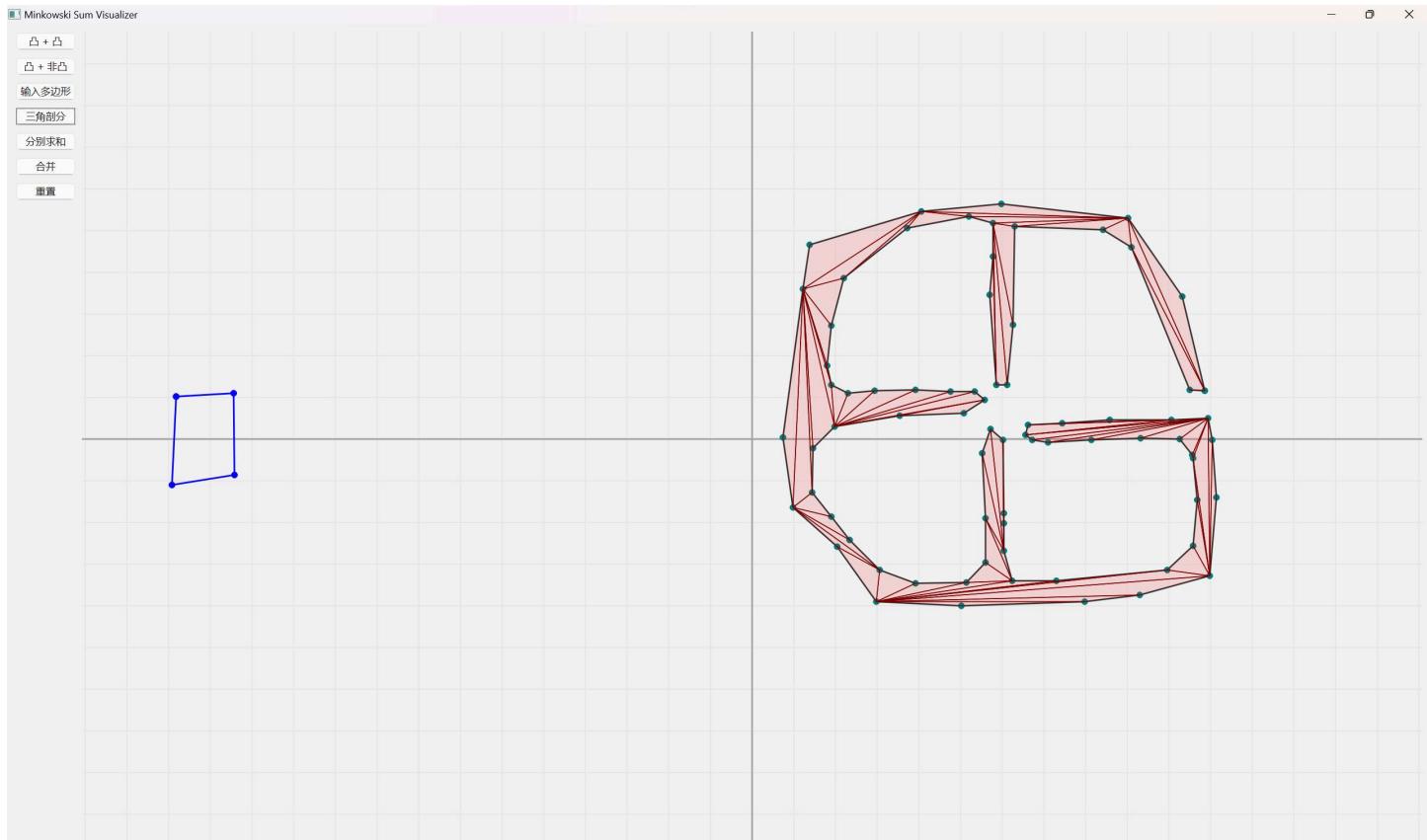


图6. 非凸多边形的三角剖分。

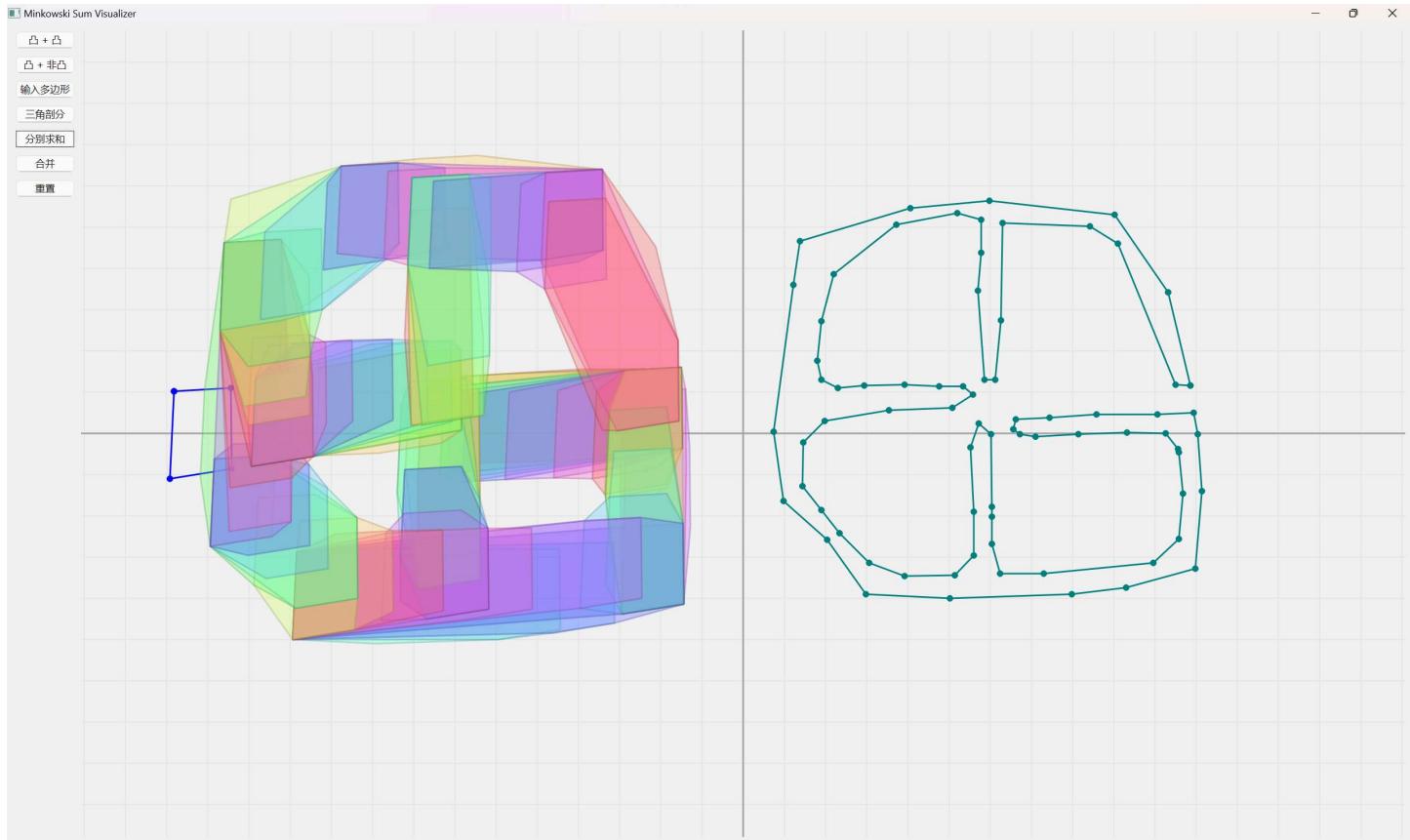


图7. 所有部分三角形和凸多边形各自的明可夫斯基和。

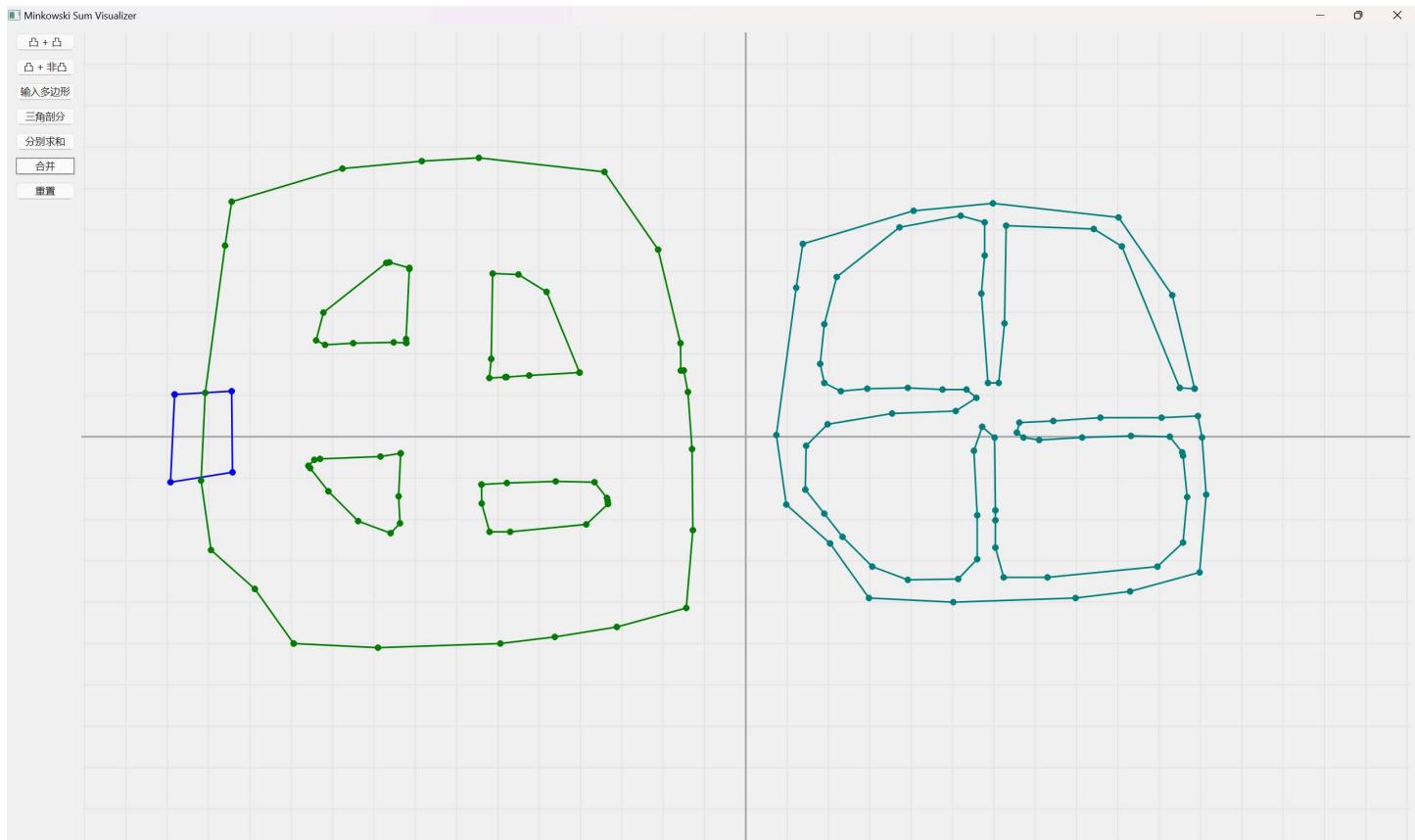


图8. 求并得到的最终明可夫斯基和。

参考文献

[1] Dobkin, D., Hershberger, J., Kirkpatrick, D., & Suri, S. (1993). Computing the intersection-depth of polyhedra. *Algorithmica*, 9(6), 518–533. <https://doi.org/10.1007/BF01190153>

[2] Kedem, K., Livne, R., Pach, J., & Sharir, M. (1986). On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete & Computational Geometry*, 1(1), 59–71. <https://doi.org/10.1007/BF02187683>