arXiv:2304.04079v1 [cs.CG] 8 Apr 2023

凸包表面映射至球面

Ben Kenwright*

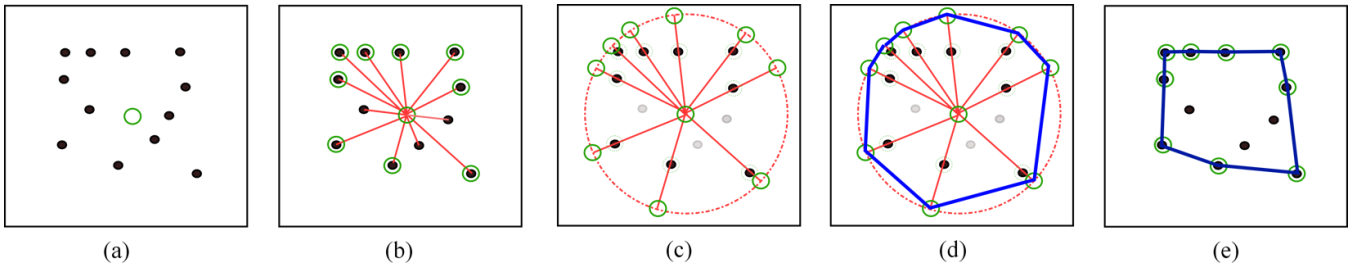

图 1： 处理共面三角形的迭代方法——*(a)* 计算形状的质心；*(b)* 使用支撑映射确定外部表面点（即从质心指向每个点的法线）；*(c)* 将表面点投影到单位球面上（即质心到表面点的距离固定）；*(d)* 使用投影后的表面点寻找相互连接的凸包三角形；*(e)* 每个投影表面点的索引构成凸包。

摘要

编写一种简单、稳健且可扩展的三维凸包算法，既具有挑战性，也存在诸多难题。这包括共面和共线问题、数值精度、性能与复杂度之间的权衡。尽管有多种基于几何计算（例如点间距离）来寻找凸包的方法，但这些方法并未解决实现可用方案时面临的技术挑战（例如数值问题和退化的点云）。我们将阐释一些常见的算法陷阱，并介绍克服和解决这些局限的工程改进。我们提出一种新颖的迭代方法，通过支撑映射和表面投影，构建一种简单而稳健的二维和三维凸包算法。

1 引言

问题 凸包算法是一项至关重要的跨学科技术，在计算机图形学、模式识别、医学分析和设计自动化等多个领域都发挥着重要作用[Barber 等人 1996年；Bentley 和 Shamos 1978年；Gregorius 2014年；Graham 和 Frances Yao 1983]。尽管已有多种方法（例如礼品包装法 [Chand 和 Kapur 1970] 以及分治法 [Preparataand Hong 1977]），但编写稳定、健壮的三维实现仍然困难且具有挑战性。在二维中实现算法或许很容易，但在三维中却并非如此 [Avis and Bremner 1995]。我们综述了多种技术，并探讨了凸包算法在实际应用中的常见问题。实际应用中会出现高度复杂和退化的顶点，使得难以轻松生成可靠的凸包。我们提出了一种新颖的方法，将凸表面点投影到球形边界上，以解决数值敏感性问题，从而为二维和三维问题提供简单可靠的解决方案。

动机 从一组点生成凸包的数学原理已有明确的定义，但尚不存在事实上的标准算法或实现。已有许多创新且有趣的概念被发表，用于解决这一问题，但要实现稳健的三维凸包算法，仍面临诸多技术挑战 [Avis and Bremner 1995]。本文着重介绍一种新颖而直接的算法，用于生成稳定的实现，并且可以轻松应用于二维和三维。

*电子邮件：bkenwright@ieee.org（2014年12月）

挑战 尽管已经提出了许多创新且独特的凸包算法，但它们并未解决实际应用中的不足。通常，算法会在简单测试用例的背景下进行介绍，例如二维情况，以解释其工作原理。我们专注于实际应用实现所需的工程改进。我们创建了各种测试用例，根据具体问题评估实现的成功与失败（见图 3）。我们的解决方案并不追求最优答案，而是专注于一种简单易行的方法，能够可靠地生成准确的凸包。

此前，正如预期，凸包问题在过去几十年中得到了深入研究，并产生了多种解决方案（见图2）。用于解决凸包问题的算法主要分为两类：插入算法和枢轴算法 [Avisand Bremner 1995]。我们的实现属于插入算法这一类。一个成功的插入方法示例是 qhull [Barber 等人 1996年；Barber 1995-2012] 算法，它通过合并面片来解决共面点导致的精度问题。这包括将点合并到共面面片中、合并凹面片、合并重复棱以及合并翻转面片。类似地，开源 Bullet物理引擎 [Coumans2012], 所采用的枢轴算法实现能够为碰撞检测问题生成可靠的凸包。Bullet 凸包实现基于 Preparata and Hong [Preparata and Hong 1977]方法，其时间复杂度为 $O(nlogn)$。此外，为了降低算法对舍入误差的敏感性，所有计算均使用整数运算。该算法能够处理退化情形，包括任意平坦面和平行面。我们的方法则基于 [Eddy 1977](QuickHull)，但通过识别外部凸包点并将其投影到共同的球面上，减少了对极其敏感的共线和共面问题的计算。

贡献 我们的主要贡献包括：定义了一种新颖的凸包算法，该算法具有较低的数值敏感性，并能够处理共线和共面顶点。本文介绍的算法提供了许多理想的优点：

✓ 易于实现（即同时适用于二维和三维）✓ 终止条件✓ 可在扩展步骤中使用容差以实现自动简化✓ 处理“共面”和“退化”的输入数据✓ 输出网格完全由输入顶点构成

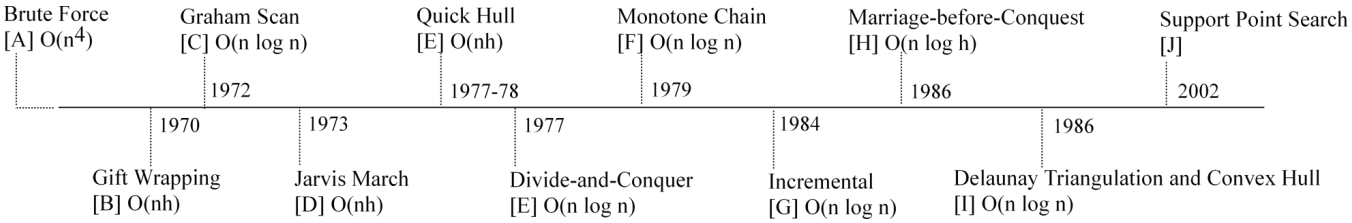

图 2：时间线——令 n 为输入集合中的点数，h 为输出凸包上的顶点数。 *[A] 蛮力法 [B][Chand 和 Kapur 1970] [C] [Graham 1972] [D] [Jarvis 1973] [E] [Eddy 1977年；Bykat 1978] [F] [Preparata and Hong 1977] [G] [Andrew 1979] [H] [Kallay 1984] [I] [Kirkpatrick and Seidel 1986] [J] EPA——扩展多胞体算法 [Kim 等人 2002] [K] [Maus 1984]*

✓ 可应用于现实世界的复杂模型，而不仅仅是点云

我们的方法是在工程层面改进将表面点映射到均匀球面上的方案，以解决一些技术缺陷（例如，用户干预（调整）以及共面面和共线面中常见的数值问题）。

2 相关工作

凸包问题在计算几何领域受到了广泛关注 [Gregorius 2014；Graham 和 Frances Yao 1983；Kim 等人 2002年；Maus 1984]。计算凸包（或简称“包”）是最早的一类复杂几何算法之一，而且有许多变体。这种算法最常见的形式是确定包含一个离散点集的最小凸集（称为“凸包”）。该算法也适用于多边形或任意线段集，其凸包与其顶点集的凸包相同。凸包有诸多应用，例如碰撞规避、隐藏物体判定和形状分析。

最流行的凸包算法是“Graham扫描”算法[Graham 1972] 和“分治”算法 [Preparata and Hong 1977]。这两种算法的实现都很容易获得（参见 [O’Rourke 1998]）。两者都是 $O（nlogn）$ 时间算法，但 Graham在二维中的运行时计算开销较低。然而，Graham 算法无法推广到三维及更高维度，而分治算法则可以自然扩展。图2展示了凸包算法的时间线。

我们的工作基于插入算法的概念。在此过程中会创建一个初始的凸包近似（即三维情况下的起始四面体）。我们使用支撑映射（例如，参见扩展- 多胞体算法（EPA） [Kim 等人 2002]）和 QuickHull 方法，以迭代方式扩展并包围所有点，从而形成一个凸包。

哪种方法最好？ 这取决于您的需求，例如，您想要二维还是三维解决方案？您是否关注运行时速度，或数值稳定性和精度？

-可并行化 -内存开销- 复杂度- 计算速度 - 鲁棒性和数值敏感性 - 维度数量（二维或三维）

3 背景

凸包 凸包是指包围指定点集的最小凸区域。严格来说，它是包含这些点的最小凸集，可以将其想象成一根橡皮筋，环绕在最外侧的点周围（也就是说，所有其他点都必须位于这根橡皮筋以内 [Barber 等人 1996]）。

不同物体的凸包各不相同，因为它取决于每个物体的特征点。有关凸几何的详细说明，请参阅 Joseph O’Rourke [O’Rourke 1998]。一个点集的凸包是唯一的（共线情形除外）。我们的表面映射方法以及将点投影到球面上的做法，降低了敏感性和共线性歧义。

支撑映射支撑映射常用于物理学和碰撞检测 [Kim等人 2002]。给定一个方向，点云的支撑映射是沿该方向距离最远的点——这简单来说就是寻找点积最大的点（即dot(direction, point)）。任意方向上的支撑点都保证位于点云凸包的表面上。我们在算法中利用这一概念来高效地确定表面点。

无新增顶点 我们有一个点数组，希望找出如何使用三角形连接这些点，以形成一个凸包。不添加额外的点。顶点编号从 1 到 N，每个三角形由顶点数组中的三个索引组成。这样可以避免任何数值漂移。我们使用三角形和顶点索引，不生成任何新点。

质心有界凸多面体称为多胞体。凸多胞体的质心，即其顶点的质心，由方程 1。

$$\frac{1}{n}\sum_{i=1}^{n} p_i \tag{1}$$

其中，质心由一组点 p_1，$...p_n$ 组成。注意，质心将包含在凸包的相对内部中。

算法概述 我们的算法采用了广为人知的 QuickHull方法，但增加了预阶段剔除和顶点重新映射。算法首先计算质心并执行支撑映射阶段，以剔除内部顶点，随后使用质心将最终顶点映射到球面上。从点集中，我们使用四个点生成一个四面体（注意——由于支撑映射阶段，随着凸包的增长，我们不需要丢弃任何内部点）。然后，通过添加外部点来迭代细化多面体的面，并将与每个面关联的剩余点重新分配到其子面中。面的细化通过从其关联点中选取距离最远的点并生成三个子三角形来完成。我们不需要担心凹边交换或移除凹顶点。

4 Our Iterative Method

本文介绍的凸包算法专注于三维情况。我们主要关注计算能够解决任意点云中未预见问题的凸包，这些点云除了包含退化数据外，还可能包含模型场景几何体，例如复杂的三维几何模型（例如，见图 6、图5和图11）。

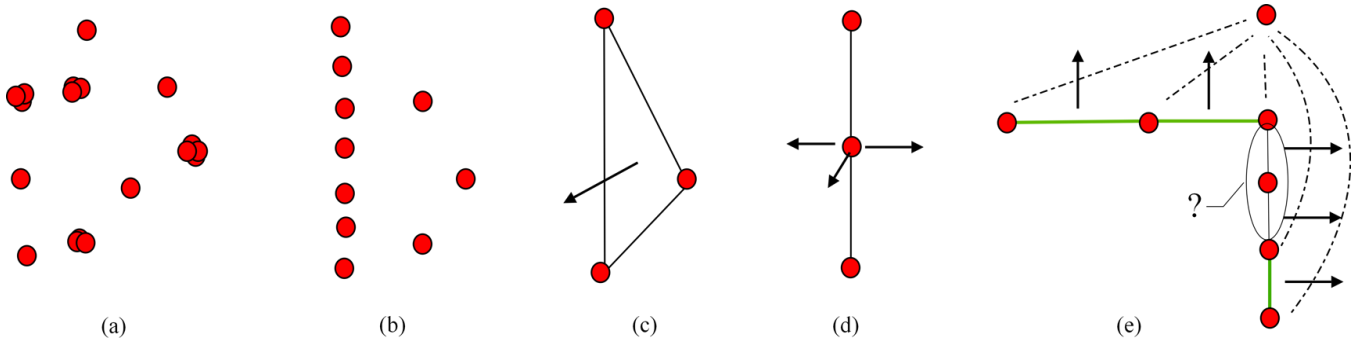

图3： 实现注意事项 -示例问题，例如：*(a)*多个重叠点，*(b)* 共面点，*(c)* 有效三角形，*(d)* 退化三角形（即，细长形状在计算面法线时会导致问题——狭长三角形），以及*(e)*共面点——导致为扩展凸多面体错误地选择面（即，重叠三角形导致凹陷错误，如图9所示）。

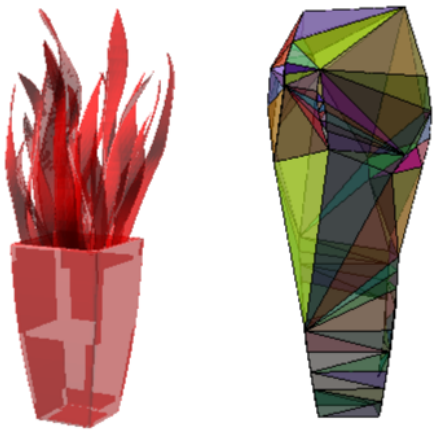

图4： 复杂三维模型 - 植物的复杂模型（*19370* 个面）网格及生成的凸包壳（*198* 个面）。

预处理阶段会先执行，以筛除并预处理点。算法从一组点开始。我们移除重复点（即彼此之间的距离在预定义容差范围内的点）。我们计算这组点的质心。我们计算从质心指向每个点的法线，找出支撑顶点，并将其添加到列表中。列表将包含一组位于凸包表面上的点。以质心为球心，我们将每个表面点投影到球面上。

迭代阶段首先取前四个顶点，并将它们连接起来，以形成尽可能小的起始封闭网格（即一个四面体）。设置好四面体后，我们通过迭代扩展凸包，将其包围住其余的表面点，并创建凸包。我们遍历每个表面点，并选择该点可见的三角形面，以此扩展凸包。移除可见面，并使用未共享的边和新的表面点添加一组新的面。关键阶段如算法 1。

- **点位于凸形状内部还是外部**- 我们可以通过遍历所有面并检查点是否位于平面内侧（即点积），轻松确定一个点是否位于凸包内。这对于在开发实现时自动检查算法是否失败非常有用。

- **点可以看到的面**——我们通过计算面法线与点的点积，找出该点可见的所有面（即当 $n \bullet p > n \bullet v$ 时面朝前，其中 n 是面法线，v 是一个面顶点，p 是测试点）。
- **从一组面中提取边** -我们有一个包含所有已找到三角形的边的数组，任何共享的边（即$count > 1$）都会被丢弃。然后使用剩余的边来创建新的三角形（即边和新点）。

动态性 我们的方法能够处理未知的点集。此外，由于我们算法的迭代特性，我们能够处理不断变化的凹包，其中的点可以动态地添加或删除。我们的凸包算法能够在每次插入/删除操作后轻松更新网格。

修改 该算法具有灵活性，可以通过不同方式进行修改，以应对这一问题。例如，我们不必像当前实现那样迭代选择列表中的每个顶点，而是可以进一步利用支撑映射概念，遍历每个面，并选择距离该面最远的点，以迭代方式扩展凸多面体。我们的算法依赖一种简单的局部几何点面测试来确定点相对于平面的位置，并利用该测试选择与该点合并的三角形。由于我们已经在支撑映射阶段剔除了内部点，因此支撑映射阶段之后的所有点都会用于创建凸包表面。将点投影到球面上可以减少共面和共线问题。由于采用了这种方法，无需进行任何交换操作来解决折叠和自相交问题，而这些问题可能会使点面测试复杂化，并破坏其局部性 [Stein 等人 2012]。

优化 我们没有专注于任何优化，而是致力于提出一种新颖的解决方案，以提供一种稳健且易于实现的方法，解决常见问题（即减少共面点和共线点）。不过，我们的实现的计时结果如图7所示，展示了顶点数量与经过时间之间的关系。我们的迭代算法具有时间复杂度 $O(n \log n)$，因为它建立在 Clarkson 和 Shor 提出的概念之上 [Clarkson 和 Shor 1989]该算法通过迭代添加外部点来扩展凸多面体，直到剩余的点集变为空集。

5 实验结果

我们在 Visual Studio 2013 和 Windows 7 中使用浮点精度，在 C++ 中实现了该算法。我们通过多种测试场景评估了我们的实现：

- 各种复杂三维模型（例如枪、兔子、茶壶）- 图 11 6，程序化测试用例（例如闵可夫斯基形状）- 图5 - 这样还有一个额外优势：能够生成各种各样的点数据（例如预加载网格模型中可能不会自然出现的退化情形）

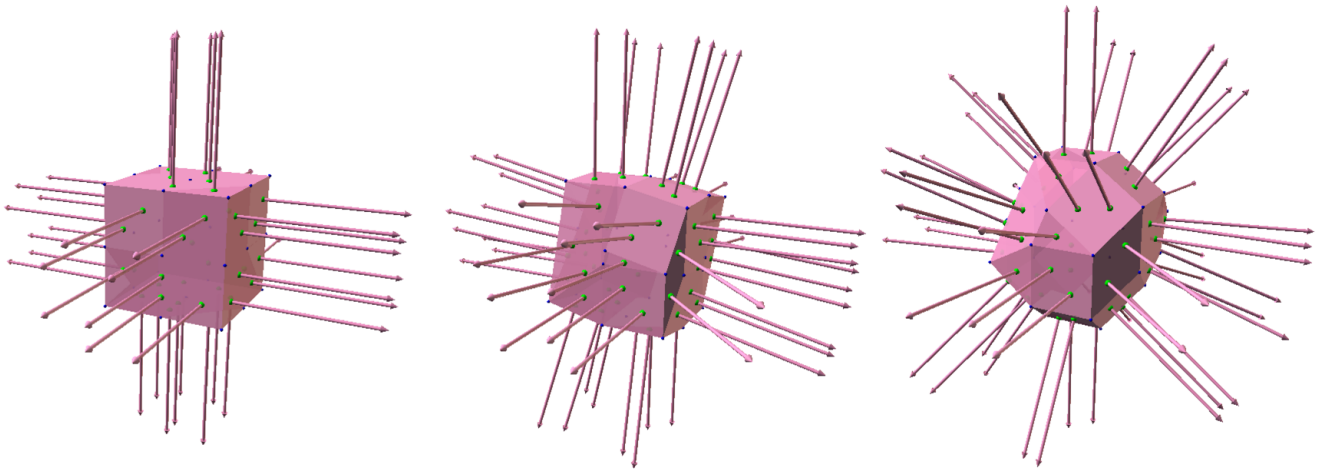

图 5： 闵可夫斯基和 - 对两个立方体模型求闵可夫斯基和，以在实际场景中评估凸包实现。绘制每个三角形的法线，便可轻松识别凸包网格。（图中所示的闵可夫斯基和测试用例中，一个立方体旋转，另一个保持固定。）

算法1：迭代三维表面映射到球面的凸包算法 数据：点数组 结果：表示点集凸包的三角形数组 1 移除重复点（即位于预定义距离容差内的点） 2 计算质心 3 清空数组 spherepoints 4 for i=*1* to 点数 do 5 从质心到点(i)的法线 = 6 supportpoint =FindSupportPoint(Normal, 点) 7 将表面点投影到球面表面 8 将 supportpoint 添加到 spherepoints 9 end 10 数组 hull 11 使用 spherepoints 构造一个四面体（例如，前四个点）并添加到数组 hull 12 for i=*4* to spherepoints 数量 do 13 找到所有对 spherepoint(i) 可见的三角形（即三角形的正面） 14 从数组 hull 中移除找到的三角形 15 找到被移除三角形的非共享边 16 使用 spherepoint(i) 和非共享边添加新三角形 17 end

- 随机点簇——即用于提供计算成本与点数之间关系的近似性能指标——图 7

我们还强调了失败案例，如图 8、图 9和图 10所示。这些情况是由于未加入额外的球面投影阶段来减少共线和共面问题而导致的。需要注意的是，我们的方法通过将点投影到公共球面格点上，降低了不同曲面的精度约束（即数值敏感性）。虽然我们可以使用支撑映射来外推表面顶点，但由于数值敏感性和共面三角形，错误地扩展初始四面体可能会生成包裹住表面的网格

顶点，但也可能是凹的，如图 9所示。

随机生成的点集测试包含不同规模的点集，范围从 500 个点到 10000 个点。对于每种规模，我们生成 5 个不同的数据集，并计算其运行时间的平均值。图 7 展示了运行时间的详细情况，并表明我们的算法为 $O(n\ log\ n)$。



图 6： 复杂三维模型 - *(a)* 枪支的复杂模型（*6174* 个点或 *2058* 个面），*(b)* 线框视图，以及 *(c)* 生成的凸包壳（*342* 个点或 *114* 个面）。

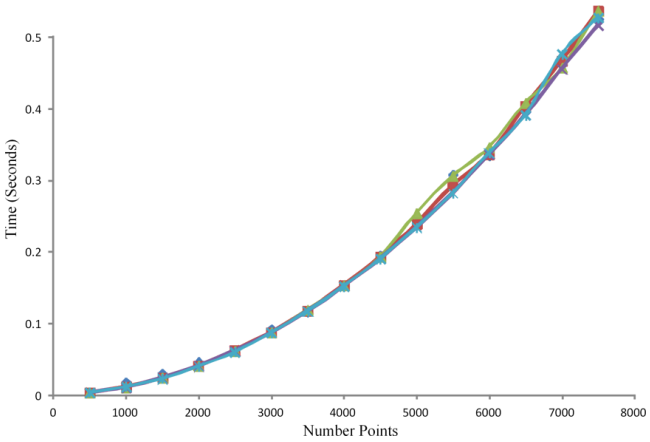

图 7： 随机三维点云- 衡量生成凸包所需时间的近似性能指标，所用随机生成的三维点集各不相同（即从 *500* 到 *10000* 个顶点）。

6 Discussion

需要考虑许多因素，例如计算速度和鲁棒性。从表面上看，凸算法可能具备.....

看似优雅而直观，但要实现得当可能很困难。例如，对于三维点云，该解决方案在处理较小凸包时可能会遇到数值问题，而在点数较多（10,000 个或更多顶点）时则可能出现计算瓶颈。尽管定点整数运算可能有助于提高鲁棒性，但使用精度有限的实数（例如单精度浮点数或双精度浮点数）可以使算法运行得快得多，但也要付出代价（例如稳定性和精度）。我们提出了一种新颖的通用方法，该方法能够有效处理实数，并且无需任何复杂的工程改进或用户干预，即可处理共面和共线表面。

我们的方法之所以有效，是因为凸包由其顶点定义。与其他方法相比，我们的方法的优势在于，表面顶点已在支撑映射阶段被找到，无需额外开销。在复杂度方面，生成凸表面时无需进行顶点剔除。此外，结合球面映射后，由于凹边和其他边导致的凸性问题，需要执行交换操作进行修复的情况也会减少。

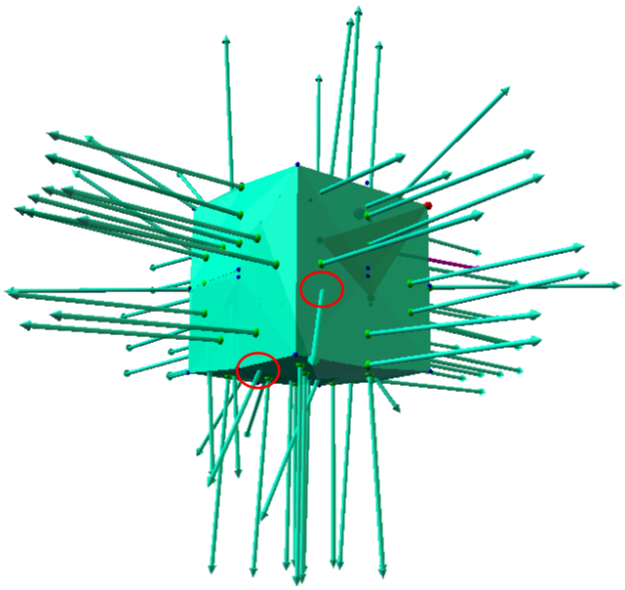

图 8： 失败案例***1***-需要谨慎对待- 尽管实现乍看之下可能在表面上是正确的，但退化且存在问题的三角形可能会被插入凸包内部（通过观察法向量，我们可以很容易地辨认出这些有问题的内部三角形）。

致谢

特别感谢审稿人抽出时间审阅本文，并提出富有洞察力的评论和建议，帮助提高本文的质量。

参考文献

Andrew, M. 1979年。另一种高效的二维凸包算法。《信息处理快报》*9*，5，216 2192– 。Avis，D.，和Bremner，D. 1995年。凸包算法有多好？载于第十一届计算几何年度研讨会论文集，ACM，20–28。1Barber，C. B.，Dobkin，D. P.，和Huhdanpaa，H. 1996年。用于求凸包的Quickhull算法。《*ACM*数学软件汇刊》（*TOMS*）*22*，4，469–483。1，2Barber，C.，1995-2012年。http://www.qhull.org/。1

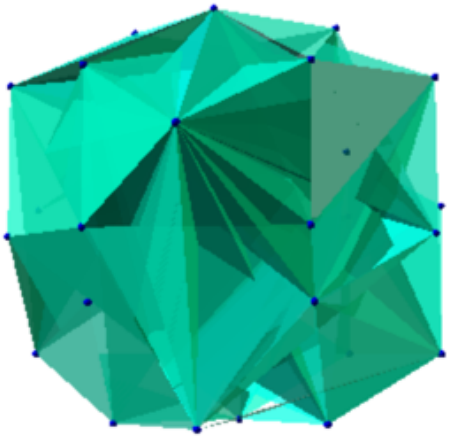

图9： 失败案例*2*- 迭代过程中错误地移除和添加三角形，可能会包围所有表面顶点，却生成凹网格（即非凸网格）。

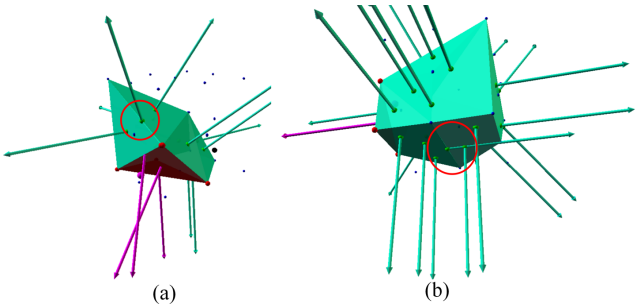

图10： 失败案例*3*- 退化三角形（即细长三角形）会导致凸包的迭代演化出现问题（例如法向量错误）。*(a)* 细长三角形；*(b)* 法线与表面平行的退化三角形。

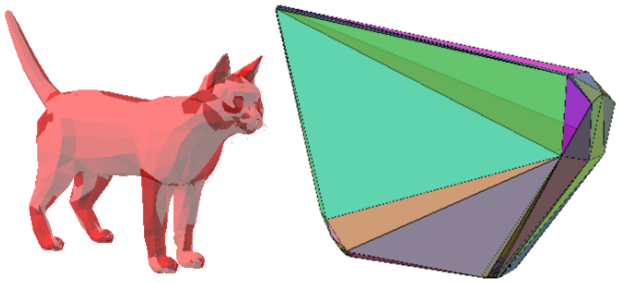

图11： 复杂三维模型 - *(a)* 猫的复杂模型（*11862* 个点或 *3954* 个面）；*(b)* 生成的凸包壳（*162* 个点或 *54* 个面）。

BENTLEY, J. L.，和 SHAMOS, M. I. 1978年。采用分治法实现线性期望时间。《信息处理快报》7，2，87 91 1– 。BYKAT, A. 1978年。二维有限点集的凸包。《信息处理快报》7，6，296–298。2 CHAND, D. R.，和 KAPUR, S. S. 1970年。一种用于凸多胞体的算法。《ACM学报》（JACM）17，1，78–86。1，2 CLARKSON, K. L.，和 SHOR, P. W. 1989年。随机采样在计算几何中的应用，II。《离散与计算几何》4，1，387–421。3 COUMANS, E.，2012年。Bullet物理SDK手册。1 EDDY, W. F. 1977年。一种用于平面点集的新型凸包算法。《ACM数学软件汇刊》（TOMS）3，4，398–403。1，2 GRAHAM, R. L.，和 FRANCES YAO, F. 1983年。求简单多边形的凸包。《算法学报》4，4，324–331。1，2 GRAHAM, R. L. 1972年。确定有限平面点集凸包的一种高效算法。《信息处理快报》1，4，132–133。2 GREGORIUS, D. 2014年。Quickhull的实现。旧金山的游戏开发者大会（Valve Software）。1，2 JARVIS, R. A. 1973年。关于识别平面上有限点集的凸包。《信息处理快报》2，1，18–21。2 KALLAY, M. 1984年。rd空间中增量凸包算法的复杂度。《信息处理快报》19，4，197。2 KIM, Y. J.，LIN, M. C.，和 MANOCHA, D. 2002 年。Deep：用于估计凸多胞体间穿透深度的对偶空间扩展。载于《机器人与自动化，2002年。论文集。ICRA’02。IEEE国际会议》，第1卷，IEEE，921–926。2 KIRKPATRICK, D. G.，和 SEIDEL, R. 1986年。终极平面凸包算法? 《SIAM计算杂志》15，1，287–299。2 MAUS, A. 1984年。在线性期望时间内对n个点进行Delaunay三角剖分并求其凸包。《BIT数值数学》24，2，151 163 2– 。O’ROURKE, J. 1998年。《C语言计算几何》。剑桥大学出版社。2 PREPARATA, F. P.，和 HONG, S. J. 1977年。二维和三维有限点集的凸包。《ACM通讯》20，2，87–93。1，2 STEIN, A.，GEVA, E.，和 EL-SANA, J. 2012年。Cudahull：图形处理器上的快速并行三维凸包。《计算机与图形学》36，4，265–271。