

基于采样代理与动态规划的高效点到网格近似距离查询

王鹏飞, 郭清昊, 曾琼, 陈双敏, 辛士庆, 屠长河

引用本文

王鹏飞, 郭清昊, 曾琼, 陈双敏, 辛士庆, 屠长河. [基于采样代理与动态规划的高效点到网格近似距离查询](#) [J]. 计算机科学, 2026, 53(9): 240-248.

WANG Pengfei, GUO Qinghao, ZENG Qiong, CHEN Shuangmin, XIN Shiqing, TU Changhe. [Efficient Approximate Point-to-Mesh Distance Query via Sampling Proxies and Dynamic Programming](#) [J]. Computer Science, 2026, 53(9): 240-248.

相似文章推荐 (请使用火狐或 IE 浏览器查看文章)

Similar articles recommended (Please use Firefox or IE to view the article)

[无线可充电传感器网络中异构感知的限时移动充电调度](#)

Time-constrained Mobile Charging Scheduling for Heterogeneous Sensing in Wireless Rechargeable Sensor Networks

计算机科学, 2025, 52(6): 355-364. <https://doi.org/10.11896/jsjcx.240400186>

[面向深度学习编译器TVM的算子融合优化](#)

Operator Fusion Optimization for Deep Learning Compiler TVM

计算机科学, 2025, 52(5): 58-66. <https://doi.org/10.11896/jsjcx.240100018>

[云计算环境下多截止期工作调度算法研究](#)

Scheduling Jobs with Multiple Deadlines in Cloud

计算机科学, 2024, 51(11A): 240100120-7. <https://doi.org/10.11896/jsjcx.240100120>

[多天线无线充电器的安全布置研究](#)

Safe Placement of Multi-antenna Wireless Chargers

计算机科学, 2024, 51(8): 345-353. <https://doi.org/10.11896/jsjcx.240400156>

[面向最优直方图求解的监督学习模型研究](#)

Study on Supervised Learning Model for Optimal Histogram Solution

计算机科学, 2023, 50(9): 145-151. <https://doi.org/10.11896/jsjcx.230300065>

基于采样代理与动态规划的高效点到网格近似距离查询

王鹏飞¹ 郭清昊¹ 曾 琮¹ 陈双敏² 辛士庆¹ 屠长河¹

1 山东大学计算机科学与技术学院 山东 青岛 266237

2 青岛科技大学信息科学技术学院 山东 青岛 266061

(pengfei1998@foxmail.com)

摘 要 点到网格表面的最近距离查询是计算机图形学领域的基础任务,广泛应用于碰撞检测、物理模拟及曲面配准等场景。然而,传统基于空间划分(如 KD 树、八叉树)或包围体层级(BVH)的加速结构,常无法有效过滤冗余分支导致查询性能显著退化,在查询点远离模型表面时尤为明显。近年来,基于增量 Delaunay 构造的动态规划最近点查询算法(DP-NNS)在点云近邻查询中展现出显著优势,然而其查询表的构建依赖点与点之间的 Voronoi cell 邻接关系,三角网格广义 Voronoi 图的计算复杂且缺乏实用工具,使得该算法无法直接推广至点到网格的距离查询场景。为此,提出一种基于 DP-NNS 框架的近似点到网格距离查询算法。三角网格由顶点、边、面 3 类基本几何元素构成,为每类元素配置代理点以近似其广义 Voronoi 胞元:顶点直接作为代理点插入,边与面则基于统一的密度参数自适应采样代理点,将 3 类元素的邻接关系统一转化为代理点之间的 Delaunay 邻接关系,从而将 DP-NNS 框架近似延伸至几何元素粒度,构建几何元素级查询表。查询阶段针对 3 类元素分别设计距离计算策略,对面片结合包围球估计进行高效剪枝,仅在必要时计算精确距离。在多个标准数据集上的实验结果表明,与 FCPW, Embree, Libigl 等方法相比,所提方法的查询速度提升 2 至 10 倍,在查询点远离模型表面时优势尤为显著,同时保持了较高的近似精度。

关键词: 点到网格距离查询;近似算法;动态规划;Delaunay 三角化;广义 Voronoi 图

中图分类号 TP391.41

Efficient Approximate Point-to-Mesh Distance Query via Sampling Proxies and Dynamic Programming

WANG Pengfei¹, GUO Qinghao¹, ZENG Qiong¹, CHEN Shuangmin², XIN Shiqing¹ and TU Changhe¹

1 College of Computer Science and Technology, Shandong University, Qingdao, Shandong 266237, China

2 College of Information Science and Technology, Qingdao University of Science and Technology, Qingdao, Shandong 266061, China

Abstract Nearest distance query from a point to a mesh surface is a fundamental operation in computer graphics, with broad applications in collision detection, physical simulation, and surface registration. However, traditional acceleration structures based on spatial partitioning (e. g., KD-tree, octree) or bounding volume hierarchies (BVH) often suffer from performance degradation due to their inability to effectively prune redundant branches, particularly when query points are distant from the model surface. Recently, the dynamic programming-based nearest neighbor search algorithm (DP-NNS), built upon incremental Delaunay construction, has demonstrated remarkable advantages in point cloud nearest neighbor queries. Nevertheless, its query table construction relies on Voronoi cell adjacency relationships between points, and the generalized Voronoi diagram of triangle meshes is computationally intractable in practice, preventing DP-NNS from being directly extended to point-to-mesh distance queries. To address this issue, this paper proposes an approximate point-to-mesh distance query algorithm built upon the DP-NNS framework. A triangle mesh consists of three types of geometric primitives: vertices, edges, and faces. Corresponding proxy points are assigned to approximate its generalized Voronoi cells. Vertices are directly inserted as proxy points, while edges and faces are adaptively sampled based on a unified density parameter. The adjacency relationships among all three primitive types are thereby unified into Delaunay adjacency relationships between proxy points, extending the DP-NNS framework to the geometric primitive level and enabling the construction of a primitive-level query table. During querying, tailored distance computation strategies are applied to each primitive type, with bounding sphere-based pruning employed for faces to avoid unnecessary exact distance computations. Experiments on multiple standard datasets demonstrate that the proposed method achieves a query speedup of 2 to 10 times over

收稿日期:2026-03-12 返修日期:2026-06-17

基金项目:山东省自然科学基金面上项目(ZR2025MS986)

This work was supported by the General Program of Shandong Provincial Natural Science Foundation(ZR2025MS986).

通信作者:陈双敏(csmqq@163.com)

FCPW, Embree and Libigl, with particularly pronounced advantages when query points are distant from the model surface, while maintaining competitive approximation accuracy.

Keywords Point-to-mesh distance query, Approximation algorithm, Dynamic programming, Delaunay triangulation, Generalized Voronoi diagram

1 引言

点到网格表面的最近距离查询是计算机图形学与计算机辅助设计领域中的核心基础操作,在碰撞检测、物理仿真、曲面配准、有限元分析等诸多应用场景中均扮演着重要角色^[1-4]。给定一个查询点与一个三角网格模型,该任务要求高效地计算查询点到网格表面的最近距离及其对应的最近点位置。随着高精度三维扫描技术与几何处理需求的快速发展,模型的面片数量持续增长,如何在大规模网格上实现快速、准确的距离查询,成为该领域的重要研究课题之一。

传统方法通常通过构建空间加速结构来提升查询效率,其中以 KD 树^[5]、八叉树等空间划分结构,以及包围体层级(BVH)最为经典。点到网格距离查询在实际应用中往往是高频操作——在物理仿真的每一个时间步、碰撞检测的每一帧处理中,查询操作都需要被反复密集地执行,因此查询速度的提升具有重要的实际意义,也是该领域长期追求的目标。另一方面,上述传统加速结构在查询点位于模型表面附近时表现良好,但当查询点距离模型表面较远时,剪枝效率会有所下降——较远的查询点使得更多候选节点的距离下界难以超过当前最优距离,导致需要访问更多的树节点。P2M 方法^[6]通过预先构建 Voronoi 图与拦截表,在一定程度上缓解了上述问题,但其预处理开销较大。

近年来,Wang 等^[7]提出了基于增量 Delaunay 构造的动态规划最近点查询算法(DP-NNS),在点云近邻查询任务中展现出显著优势,尤其在二维流形数据和远场查询场景下,其查询速度较 KD 树和 R 树提升 1 至 10 倍。DP-NNS 的核心思想是:在点云的增量 Voronoi 构造过程中,记录每个点的 Voronoi 胞元被后续插入点侵占的历史,形成查询表,从而将最近点查询转化为沿查询表的高效遍历。然而,DP-NNS 的查询表构建完全依赖点与点之间的 Voronoi 邻接关系,而点到网格查询的基本对象是三角面片。然而三角形的 Voronoi 图在三维空间中的精确计算极为复杂,缺乏实用的构造工具,使得 DP-NNS 无法直接推广至点到网格的距离查询场景。

本文提出一种基于 DP-NNS 框架的近似点到网格距离查询算法。三角网格由顶点、边、面 3 类基本几何元素构成,本文为每类元素配置代理点以近似其广义 Voronoi 胞元:顶点直接作为代理点插入,边与面则基于统一的密度参数自适应采样代理点,将 3 类元素的邻接关系统一转化为代理点之间的 Delaunay 邻接关系,从而将 DP-NNS 的增量构造框架近似延伸至几何元素粒度,构建几何元素级查询表。查询阶段针对 3 类元素分别设计距离计算策略,对面片结合包围球估计进行高效剪枝,仅在必要时计算精确距离,从而在保持较高近似精度的同时大幅提升查询效率。

在 Bunny, Dragon, Armadillo 等多个经典三维模型上的

实验结果表明,本文方法的查询速度相比 FCPW^[8], Embree^[9]和 Libigl^[10]等方法提升 2 至 10 倍,在查询点远离模型表面的场景下优势突出,同时在绝大多数实际场景中均能保持较高的近似精度,满足工程应用的实际需求。

2 相关工作

2.1 点云最近邻查询

给定点云数据,最近邻查询旨在对于任意查询点,高效找到点云中与其距离最近的点,是计算机图形学与计算机视觉中最基础的几何操作之一。根据组织方式的不同,现有方法大致可分为 3 类:基于空间划分的方法、基于数据划分的方法,以及基于 Voronoi 图的方法。

基于空间划分的方法通过将三维空间递归地分割为若干子区域来加速查询。KD 树^[5]是最具代表性的结构,它沿坐标轴交替递归地划分空间,构造复杂度为 $O(n \log n)$,平均查询复杂度为 $O(\log n)$ ^[11]。在实际应用中,nanoflann^[12]能高效实现查询而被广泛采用。八叉树将三维空间均匀递归地划分为 8 个子立方体,每个内部节点对应一个立方体区域,其子节点对应 8 个等大的子立方体,定位叶节点的计算代价为 $O(\text{depth}) \approx O(\log n)$ ^[13]。网格方法则将空间覆盖划分为均匀网格,查询时检查包含查询点的网格单元及其相邻单元,在均匀分布的数据上效率较高,但在数据分布不均或查询点远离数据集时,性能显著退化。上述空间划分方法的共同局限在于:当查询点远离数据集或数据呈二维流形分布时,大量候选节点的距离下界均难以超过真实最近距离,导致需要访问更多节点,使查询效率下降。

基于数据划分的方法以 R 树^[14]及其变体为代表。R 树基于 B 树结构,以数据驱动的方式将点集组织为层次化的最小包围矩形,在非均匀分布场景下具有较好的适应性。R* 树^[15]在 R 树基础上引入了面积、边界与重叠的综合优化策略,并通过强制重插入机制动态调整结构,在多种数据分布与查询类型下均表现出优于其他 R 树变体的性能。BD 树^[16]在 KD 树基础上引入了缩减域的概念,通过同时记录包含域和排除域来改善最坏情况下的查询性能。

基于 Voronoi 图的方法理论上具有非常理想的特性,查询点 q 的最近邻恰好是 q 所在 Voronoi 胞元的生成元。然而,Voronoi 图缺乏层次化表示,点定位问题在三维空间中缺乏高效的实用算法,使得其在大规模点云上的直接应用受到较大限制。

Wang 等^[7]提出的 DP-NNS 算法将动态规划思想与增量 Delaunay 构造相结合,通过记录 Voronoi 胞元在增量点插入过程中的演变历史,为每个点构建查询列表,将最近邻查询转化为沿查询列表的高效遍历过程。对于分布在二维流形上的点云,以及查询点远离数据集的情形,该算法的查询速度可达

KD 树与 R 树的 1 至 10 倍,同时支持在有序点云中高效查询前 k 个点的最近邻,这一特性在密度峰值聚类等业务中具有独特价值。然而,DP-NNS 的查询表构建完全依赖点与点之间的 Voronoi 邻接关系,无法直接处理三角面片等更复杂的几何元素,限制了其在点到网格距离查询中的直接应用。

2.2 点到网格距离查询

点到网格距离查询与点云最近邻查询的本质区别在于:查询的基本几何元素由单纯的点集扩展为包含顶点、边、面的复合结构。对于网格模型,最近点可能落在任意顶点、边上的某处或某个三角面片的内部。这一复合性使得直接套用点云查询方法会产生错误结果——仅依赖 KD 树找到最近顶点,并不能保证该顶点就是真正意义上的最近几何元素,最近点很可能位于某条边或某个面的内部。因此,点到网格距离查询需要专门设计能够处理复合几何元素的查询结构与算法。

包围体层级(BVH)是目前点到网格距离查询最主流的解决方案。BVH 通过对三角面片构建层次化的包围体树来加速查询,其基本思想是:若查询点到某节点包围体的距离下界已超过当前最优距离,则该节点所包含的所有三角面片均可被安全剪枝。常用的包围体类型包括轴对齐包围盒(AABB)、包围球、有向包围盒(OBB)以及 K-DOP 等,不同类型在包围紧密度与相交测试计算代价之间各有取舍:AABB 计算简单,但与几何体的贴合较松;OBB 贴合更紧,但计算代价更高;包围球在旋转不变性上具有优势,但紧密度较差。

在 BVH 的具体实现方面,PQP 利用 OBB 层级树实现了高效的最近距离查询与碰撞检测,支持点到网格、网格到网格等多种查询类型,在工业界与学术界均得到广泛应用。FCPW^[8]通过宽 BVH(Wide BVH)结合 SIMD 向量化遍历,将多个包围体的相交测试并行化,在 CPU 上实现了显著的查询加速,是目前 BVH 类方法中单线程性能最优的实现之一。Libigl 作为几何处理领域的基础工具库,提供了基于 AABB 树的点到网格查询接口,使用便捷,常作为基准方法参与性能评估。Embree^[9]由 Intel 开发,针对光线追踪场景设计,同样支持点到网格的距离查询,并充分利用了 AVX 等 SIMD 指令集的计算能力。此外,基于体素化的方法通过将查询空间离散化为规则网格,预先计算每个体素到网格表面的近似距离,在查询阶段通过插值快速得到近似结果,适合对精度要求不高但需要极高吞吐量的场景。总体而言,BVH 类方法在查询点位于模型表面附近时表现出色,层次化结构能够有效剪枝大量不相关面片;但在远场查询场景下,由于更多候选节点的距离下界难以超过当前最优距离,剪枝效率会有所下降,需要访问更多的树节点。

P2M 提出了一种与 BVH 截然不同的算法范式,其出发点是:网格顶点集所定义的 Voronoi 图与由顶点、边、面共同定义的广义 Voronoi 图之间,存在一种可以预先计算的“拦截”关系。具体而言,若某顶点 v 的 Voronoi 胞元 $Cell(v; \mathcal{V}_V)$ 与某边 e (或某面 f)的广义 Voronoi 胞元 $Cell(e; \mathcal{V}_{V,E,F})$ 的交集非空,则称 e (或 f)拦截了 v ,即 v 是 e (或 f)的截距顶点。在预处理阶段,P2M 通过高效的洪泛式拦截检测算法,利用凸多面体松弛与快速过滤规则,在较短时间内完

成所有拦截对的计算,并构建拦截表。查询时,先用 KD 树找到距查询点最近的网格顶点,再通过该顶点的拦截表检索出其所拦截的边与面,从中确定真正的最近几何元素,整体查询速度远超 BVH 类方法,是目前精确点到网格查询的最快方法之一。然而,P2M 的预处理阶段需要精确计算广义 Voronoi 图的相交结构并完成大规模拦截检测,实现复杂度较高,因此预处理开销较大。

2.3 Voronoi 图与 Delaunay 三角化

给定域 Ω 中的一组生成元 $P = \{p_i\}_{i=1}^n$,Voronoi 图将 Ω 划分为 n 个胞元,其中每个胞元 $V(p_i)$ 定义为距生成元 p_i 最近的区域:

$$V(p_i) = \{x \in \Omega \mid d(x, p_i) \leq d(x, p_j), \forall j \neq i\}$$
 (1)

这一基于邻近性的空间分解天然编码了最近邻关系:查询点 q 落在 $V(p_i)$ 内,当且仅当 p_i 是 q 在 P 中的最近邻。Voronoi 图可以在欧氏空间、黎曼流形等不同度量空间下进行推广,在计算机图形学、几何建模等领域有着广泛应用^[17-19]。

Delaunay 三角化是 Voronoi 图的几何对偶结构,通过边的连接关系显式编码了相邻 Voronoi 胞元之间的邻接关系——若两个生成元的 Voronoi 胞元共享边界,则在 Delaunay 三角化中对应顶点之间存在一条边。在二维空间中,Delaunay 三角化由三角形构成;在三维空间中,其由四面体构成。Delaunay 三角化满足空球性质:任意单纯形的外接球内部不包含其他生成元,这一性质保证了三角化的良好几何质量。Bowyer-Watson 算法^[20-21]是构造 Delaunay 三角化的经典增量方法,通过依次插入点并局部更新受影响的单纯形来完成构造,平均时间复杂度为 $O(n \log n)$ 。目前,CGAL^[22],TetGen^[23]等工具库提供了高效稳健的 Delaunay 三角化实现,被广泛应用于几何处理领域。本文正是借助增量 Delaunay 构造过程中所产生的邻接信息,构建面片层级的查询表,从而实现高效的点到网格近似距离查询。

3 DP-NNS 框架概述

本文方法建立在基于动态规划的最近点查询算法 DP-NNS 之上。该框架的核心思想是:在点云的增量 Voronoi 构造过程中,预先计算并存储每个点的 Voronoi 胞元被后续插入点侵占的历史信息,从而将最近邻查询转化为沿预计算结构的高效遍历。该框架的核心思想如图 1 所示。

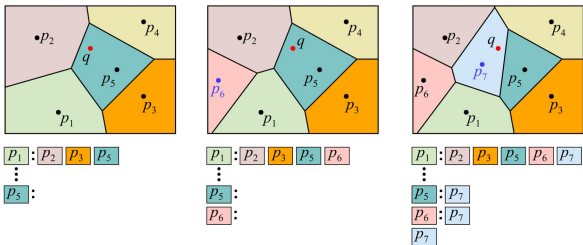


图 1 DP-NNS 核心观察:增量插入过程中查询点最近邻的演化及查询列表构建

Fig. 1 Core observation of DP-NNS:nearest-neighbor evolution and query list construction during incremental insertion

3.1 动态规划递推

设点云 $P = \{p_i\}_{i=1}^n$ 按插入顺序编号, 查询点为 $q \in \mathbb{R}^3$ 。用 $\Phi_m(q)$ 表示 q 在前 m 个点构成的子集 $\{p_i\}_{i=1}^m$ 中的最近邻, 则有如下递推关系:

$$\Phi_m(q) = \begin{cases} p_1, & m=1 \\ \Phi_{m-1}(q), & m>1 \text{ 且 } \|q - \Phi_{m-1}(q)\| \leq \|q - p_m\| \\ p_m, & m>1 \text{ 且 } \|q - \Phi_{m-1}(q)\| > \|q - p_m\| \end{cases}$$

全局最近邻即为 $\Phi_n(q)$ 。朴素实现的查询复杂度为 $O(n)$, DP-NNS 的目标是通过预计算来加速这一过程。

3.2 核心几何观察

设 $\mathcal{V}_m$ 为由前 m 个点构造的 Voronoi 图, $Cell(p; \mathcal{V}_m)$ 表示点 p 在 $\mathcal{V}_m$ 中所对应的 Voronoi 胞元。根据 Voronoi 图的定义, q 始终位于其当前最近邻的 Voronoi 胞元内, 即 $q \in Cell(\Phi_m(q); \mathcal{V}_m)$ 。DP-NNS 的关键几何观察是: $\Phi_m(q)$ 与 $\Phi_{m-1}(q)$ 不同, 意味着新插入点 p_m 必然侵占(缩减)了当前最近邻的 Voronoi 胞元; 反之, 若 p_m 侵占了该胞元, 则 $\Phi_m(q)$ 不一定改变, 还需满足 p_m 比当前最近邻更近才会发生切换。其形式化地表示为:

$$p_m \not\hookrightarrow Cell(\Phi_{m-1}(q); \mathcal{V}_{m-1}) \Rightarrow \Phi_m(q) = \Phi_{m-1}(q)$$

其中, $\hookrightarrow$ 表示新插入点的 Voronoi 胞元侵占了已有点的 Voronoi 胞元。进一步地, 若连续若干次插入均未侵占当前最近邻的胞元, 则最近邻保持不变:

$$\begin{aligned} p_{m+j} \not\hookrightarrow Cell(\Phi_m(q); \mathcal{V}_{m+j-1}), \forall 1 \leq j \leq k \\ \Downarrow \\ \Phi_{m+k}(q) = \Phi_{m+k-1}(q) = \dots = \Phi_m(q) \end{aligned} \quad (2)$$

3.3 查询表构建与查询过程

基于上述观察, DP-NNS 为每个点 p_i 维护一个查询列表 L_i 。在增量构造 Voronoi 图的过程中, 每当新插入点 p_j 侵占了 p_i 的 Voronoi 胞元(等价地, p_j 与 p_i 在对应的 Delaunay 三角化中互为邻居), 则将 p_j 追加至 L_i 的末尾。由于 Delaunay 三角化是 Voronoi 图的几何对偶, 胞元的邻接关系可直接通过 Delaunay 的边连接关系读取, 避免了显式构造 Voronoi 图。所有点插入完毕后, 查询表构建完成, 其构建复杂度与 Delaunay 三角化的构造复杂度一致, 平均为 $O(n \log n)$ 。

查询阶段从 p_i 出发, 依次遍历当前点的查询列表。若列表中某点 p_j 满足 $\|q - p_j\| < \|q - p_{\text{curr}}\|$, 则将当前候选更新为 p_j , 并跳转至 L_j 继续搜索; 否则继续遍历当前列表的下一个元素, 直至列表遍历完毕, 返回当前候选作为最终结果。从几何角度理解, 查询路径实际上重现了 Voronoi 图增量构造过程中 q 所在胞元的生成元随插入顺序演变的序列, 平均查询复杂度为 $O(\log n)$ 。

4 本文方法

本文的核心目标是将 DP-NNS 框架从点云最近邻查询扩展至点到网格距离查询。回顾 DP-NNS 的本质: 其查询表编码了 Voronoi 胞元在增量构造过程中的演变历史, 而这一历史恰好与最近邻的动态变化一一对应。将这一思想推广至三角网格的关键障碍在于: 点的 Voronoi 图具有简洁的多面

体结构, 而三角网格的广义 Voronoi 图的胞元边界由曲面构成, 其在三维空间中的精确计算至今缺乏实用工具。因此, 直接以面片为生成元构造广义 Voronoi 图的路线在实践中难以实现: 一方面, 目前尚无成熟的工具库支持三维空间中广义 Voronoi 图的稳健构造; 另一方面, 即便构造成功, 其计算复杂度也远超点 Voronoi 图的, 使得预处理开销难以承受。

为此, 本文的核心思想是以离散代理点的 Delaunay 邻接近似刻画几何元素间的广义 Voronoi 邻接: 通过为顶点、边、面分别配置代理点, 将原本难以计算的连续曲面邻接关系, 转化为由成熟 Delaunay 工具高效求解的离散点邻接关系, 从而绕开广义 Voronoi 图的直接构造。三角网格由顶点、边、面 3 类基本几何元素构成, 空间中任意查询点的最近点可能落在其中任意一类元素上, 因此 3 类元素均需参与查询表的构建。具体地, 所有顶点直接作为代理点插入; 对于边与面, 则分别按其几何尺度(长度与面积)自适应地采样若干代理点, 以点代线, 以点代面, 将各元素层级的邻接关系统一转换为点之间的 Delaunay 邻接关系。在此近似意义下, 两个几何元素之间的广义 Voronoi 邻接关系, 可由其代理点之间的 Delaunay 邻接关系近似捕获, 从而使 DP-NNS 的增量构造框架得以在几何元素粒度上近似复现。

4.1 自适应采样

三角网格的 3 类基本几何元素——顶点、边、面——在广义 Voronoi 图中各自占据独立的胞元, 空间中任意查询点的最近点可能落在其中任意一类元素上。本文的核心思路是: 为每类几何元素配置若干代理点, 以代理点的 Voronoi 胞元之并集近似覆盖对应几何元素的广义 Voronoi 胞元。在此近似意义下, 若两个几何元素各自的代理点在 Delaunay 三角化中互为邻居, 则认为这两个元素之间存在广义 Voronoi 邻接关系, 并将其记录至查询表中。因此, 代理点的空间覆盖质量直接决定了邻接关系的捕获完整性, 进而影响了查询的近似精度。

3 类元素的代理点采样策略各有不同, 需根据其几何特性分别设计。顶点本身即为点, 无须额外采样, 网格中所有顶点直接作为代理点参与 Delaunay 构造。对于边与面, 本文引入统一的用户参数 N , 其物理含义为边与面上代理点总数的近似量级, 用于整体控制采样的疏密程度。基于 N 与网格总面积 $A_{\text{total}} = \sum_{i=1}^m A_i$, 采样密度自动计算为:

$$\rho = \sqrt{\frac{A_{\text{total}}}{N}} \quad (3)$$

其中, ρ 的物理意义是代理点的平均空间间距。基于此, 边 e_i (长度为 l_i) 的目标采样点数定义为:

$$n_{e_i} = \max\left(1, \left\lfloor \frac{l_i}{\rho} \right\rfloor\right) \quad (4)$$

面片 T_i (面积为 A_i) 的目标采样点数定义为:

$$n_{T_i} = \max\left(1, \left\lfloor \frac{A_i}{\rho^2} \right\rfloor\right) \quad (5)$$

两者均保证至少有一个代理点, 确保每条边与每个面片均参与查询表的构建。需要注意的是, 由于 $\max(1, \cdot)$ 的截断效应, 实际代理点总数与 N 之间存在一定偏差, N 在此仅

作为控制采样粒度的近似量级参数,而非严格意义上的精确总数。

在每条边的内部,代理点沿边均匀分布;在每个面片的内部,代理点通过蓝噪声采样生成,以最大化点集的最小点间距,确保代理点在面片内部的空间覆盖均匀性。

上述采样策略具有良好的几何一致性:由 ρ 的定义可知,对于满足 $n_{T_i} > 1$ 的面片,其平均代理点间距约为 $\sqrt{A_i/n_{T_i}} \approx \rho$,与面片面积无关;对于边,相邻代理点间距约为 $l_i/n_{e_i} \approx \rho$,与边长无关。这意味着无论几何元素的尺度如何,其代理点的空间覆盖密度在统计意义下是一致的,从而保证了不同尺度元素之间邻接关系捕获能力的均匀性。 N 越大, ρ 越小,代理点对各元素广义 Voronoi 胞元边界的覆盖越完整,近似精度越高,但预处理开销也会随之增加。

4.2 查询表构建

完成代理点采样后,预处理阶段的核心任务是构建几何元素级查询表。设网格的顶点集、边集、面集分别为 $\mathcal{V}, \mathcal{E}, \mathcal{F}$, 查询表 $\mathcal{L}$ 是一个全局数据结构,为网格中每个几何元素 g 维护一个对应的查询列表 L_g ,即 $\mathcal{L} = \{L_g | g \in \mathcal{V} \cup \mathcal{E} \cup \mathcal{F}\}$,其中每个 L_g 都是一个动态数组,记录在增量 Delaunay 构造过程中与 g 建立邻接关系的后续几何元素及其类型标记。为阐明查询表构建的理论基础,首先回顾 DP-NNS 中查询列表的语义:点 p_j 出现在 p_i 的查询列表中,当且仅当在插入顺序为时序的增量 Voronoi 构造过程中, p_j 的插入导致 p_i 的 Voronoi 胞元收缩,即 $p_j \hookrightarrow \text{Cell}(p_i; \mathcal{V}_{j-1})$ 。由于 Delaunay 三角化与 Voronoi 图的对偶性,上述条件等价于 p_j 与 p_i 在插入 p_j 后的 Delaunay 三角化中互为邻居。

本文将上述语义近似推广至三类几何元素。用 $\text{Proxy}(g)$ 表示几何元素 g (顶点、边或面片) 的代理点集合,则几何元素 g_k 出现在几何元素 g_j ($j < k$) 的查询列表 L_j 中,当且仅当 g_k 的某个代理点与 g_j 的某个代理点在插入 g_k 的代理点批次后的 Delaunay 三角化中互为邻居。形式化地,若存在代理点对 (s, s') 满足:

$$s \in \text{Proxy}(g_k), s' \in \text{Proxy}(g_j), (s, s') \in \mathcal{D} \quad (6)$$

其中, $\mathcal{D}$ 为将所有代理点按批次增量插入后构建的三维 Delaunay 三角化, $(s, s') \in \mathcal{D}$ 表示 s 与 s' 在 $\mathcal{D}$ 中由一条 Delaunay 边直接相连,即互为 Delaunay 邻居。若该条件成立,则将 g_k 连同其元素类型标记一并追加至 L_j 。类型标记用于在查询阶段区分顶点、边、面 3 类元素,以调用对应的精确距离计算函数。这一判定条件在几何上意味着, g_k 的代理点 Voronoi 胞元与 g_j 的代理点 Voronoi 胞元存在公共边界,从而近似反映了两个几何元素在广义 Voronoi 意义下的相互侵占关系。

将代理点按顶点、边、面的顺序分批插入 Delaunay 三角化结构 $\mathcal{D}$: 首先,按随机顺序依次将每个顶点插入;随后,按随机顺序依次将每条边的代理点作为一个批次插入;最后,按随机顺序依次将每个面片的代理点作为一个批次插入。在构建过程中,同一几何元素内部代理点之间的 Delaunay 边对查询表无贡献,处理时直接跳过。由于同一几何元素 g_k 可能通过

多个不同的代理点对与 g_j 建立 Delaunay 邻接,为避免重复记录,对每个 g_j 维护已记录几何元素的标记集合,保证每个有序元素对 (g_j, g_k) 在 L_j 中至多出现一次。整体构建过程的伪代码如算法 1 所示。

算法 1 几何元素级查询表构建

输入: 顶点集 $\mathcal{V}$, 边集 $\mathcal{E}$ (随机排列), 面集 $\mathcal{F}$ (随机排列), 各元素代理点集合 $\text{Proxy}(\cdot)$

输出: 查询表 $\mathcal{L}$

1. 初始化空查询表 $\mathcal{L}$, 初始化 Delaunay $\mathcal{D}$;
2. 按顺序 $\mathcal{V} \rightarrow \mathcal{E} \rightarrow \mathcal{F}$ 依次处理几何元素 g_k ;
3. for 每个几何元素 g_k do
4. 将 $\text{Proxy}(g_k)$ 中所有代理点插入 $\mathcal{D}$;
5. for 每个新插入的代理点 s do
6. for 每个 Delaunay 邻居点 s' do
7. 设 g_j 为 s' 所属几何元素;
8. if $g_j \neq g_k$ 且 g_k 尚未记录至 L_{g_j} then
9. 将 g_k 追加至 L_{g_j} ;
10. end
11. end
12. end
13. end

4.3 查询阶段

给定查询点 q , 查询阶段的目标是在几何元素级查询表上高效地找到距 q 最近的几何元素 (顶点、边或面片) 及对应的最近距离。设 $\Psi_k(q)$ 表示 q 在前 k 个几何元素中的最近元素, 随着 k 的递增, $\Psi_k(q)$ 按如下递推关系动态更新:

$$\Psi_k(q) = \begin{cases} g_1, & k=1 \\ \Psi_{k-1}(q), & d(q, g_k) \geq d(q, \Psi_{k-1}(q)) \\ g_k, & d(q, g_k) < d(q, \Psi_{k-1}(q)) \end{cases} \quad (7)$$

其中, $\Psi_{|\mathcal{V} \cup \mathcal{E} \cup \mathcal{F}|}(q)$ 为最终的近似最近几何元素。查询表的作用在于: g_k 出现在 L_{g_j} 中意味着 g_k 可能侵占了 g_j 对应的空间区域, 对于落在 g_j 附近的查询点 q , g_k 才有可能成为比 g_j 更近的候选。因此, 查询过程只需沿查询表遍历那些真正可能改变当前最优的候选元素, 而无须遍历所有元素, 从而实现有效剪枝。

由于 3 类几何元素的几何特性不同, 其距离计算策略也有所区别。对于顶点 v , 其到查询点的距离直接计算为 $d(q, v) = \|q - v\|$ 。对于边 e , 计算点到线段的距离; 若最近点落在线段端点处, 则直接跳过, 因为端点已由对应顶点负责, 仅当最近点落在线段内部时才考虑切换至该边。对于面片 T_i , 在预处理阶段为其预计算最小包围球 $B_i = (c_i, r_i)$, 查询时首先利用包围球对距离进行快速估计, 仅在必要时才计算精确的点到三角形的距离, 从而减少不必要的精确距离计算开销。

查询过程从第一个插入的几何元素 g_1 出发 (由插入顺序可知 g_1 为某一点), 维护当前最优元素 g_{curr} 及最优距离估计 d_{best} , 初始令 $d_{\text{best}} = \|q - g_1\|$ 。遍历 g_{curr} 的查询列表时, 对每个候选元素 g_j 根据其类型计算距离, 若 $d(q, g_j) < d_{\text{best}}$, 则切换 $g_{\text{curr}} \leftarrow g_j$ 并更新 d_{best} 。每当 g_{curr} 发生切换, 算法递归地进入新的 $L_{g_{\text{curr}}}$ 继续遍历, 直至当前列表遍历完毕且无更优

候选,最终返回 g_{curr} 及对应的最近距离与最近点。查询过程的伪代码如下算法 2 所示。

算法 2 近似点到网格距离查询

输入:查询表 $\mathcal{L}$,查询点 q

输出:最近几何元素 g^* 及最近距离 d^*

```

1.  $g_{curr} \leftarrow g_1, d_{best} \leftarrow \|q - g_1\|$ ;
2. FunctionQuery( $g_{curr}$ ):
3.   for 每个  $g_j \in \mathcal{L}_{g_{curr}}$  do
4.     计算  $d(q, g_j)$ ;
5.     if  $d(q, g_j) < d_{best}$  then
6.        $d_{best} \leftarrow d(q, g_j), g_{curr} \leftarrow g_j$ ;
7.       Query( $g_j$ );
8.     break;
9.   end
10. end
11. Query( $g_1$ );
12. return  $g_{curr}, d_{best}$ ;
```

4.4 复杂度与近似性质分析

预处理阶段的主要开销集中于增量 Delaunay 三角化的构建。设网格的面片总数为 m ,总代理点数量为 N ,Delaunay 构造的平均时间复杂度为 $O(N \log N)$,最坏情况下的复杂度为 $O(N^2)$,但后者在实际数据中极少出现。查询表的构建随 Delaunay 构造同步完成,每次点插入后提取其 Delaunay 邻居并更新相应的查询列表,单次操作的额外开销为常数级,不改变整体复杂度。各面片包围球的预计算开销为 $O(m)$,因此预处理阶段的总时间复杂度为 $O(N \log N)$ 。

查询阶段的复杂度取决于遍历过程中需要对其计算距离的候选面片数目。以三维 Delaunay 三角化中每个面片平均邻接面片数为常数 c 的已知性质, T_k 与当前最优面片存在 Delaunay 邻接的概率约为 c/k ,因此需要计算距离的候选面片期望数目为:

$$O\left(\sum_{k=1}^m \frac{c}{k}\right) = O(\log m) \quad (8)$$

即平均查询复杂度为 $O(\log m)$ 。最坏情况下,若所有面片形成链式侵占结构,查询复杂度退化为 $O(m)$,但在实际网格数据中极为罕见。算法的内存开销主要来自 Delaunay 三角化结构、查询表以及各面片球包信息,整体内存复杂度为 $O(N)$ 。

本文方法的近似误差本质上源于代理点对几何元素广义 Voronoi 胞元的覆盖不完整:当某个几何元素的代理点未能覆盖到靠近空间元素一侧的区域时,两者之间便无法在 Delaunay 三角化中建立邻接关系,从而在查询表中产生遗漏。在三维空间中,任意两个相邻几何元素 g_i 与 g_j 的广义 Voronoi 分界面是由平面、抛物柱面、双曲面等二次曲面片拼接而成的连续曲面:

$$\mathcal{B}_{ij} = \{x \in \mathbb{R}^3 | d(x, g_i) = d(x, g_j)\} \quad (9)$$

元素间的跳转逻辑能否被正确捕获,取决于 $Proxy(g_i)$ 与 $Proxy(g_j)$ 中是否存在一对代理点在离散点集诱导的 Voronoi 图中互为邻居。误差的产生来源于两类几何退化情

形。1)采样稀疏导致的胞元未产生邻接:几何元素 g_i 的势力范围在本文方法中被近似为代理点 Voronoi 胞元的并集 $\bigcup_{s \in Proxy(g_i)} Cell(s; V_P)$,当采样间距 δ 较大时,分布在 V_{ij} 两侧的代理点因距离过远,其对应的点 Voronoi 胞元无法在空间中相互接触,从而无法生成对偶的 Delaunay 边,导致 (g_i, g_j) 的邻接关系被遗漏。2)窄带几何退化:对于极度狭长的三角形或极短的边,其广义 Voronoi 胞元在空间中呈现出极薄的片状或带状结构,即便平均采样密度较高,代理点仍可能无法精准落在薄胞元与其邻居交界的窄带区域内,从而产生拓扑盲区。

对于误差收敛性,由自适应采样策略可知,全局平均采样间距 $\delta \approx \rho = \sqrt{A_{total}/N}$,当 δ 显著小于各几何元素的最小局部特征尺寸时,代理点集能够以极高概率穿透复杂的广义 Voronoi 分界面并建立邻接连接。因此,近似误差随 N 的增大单调收敛。在实际应用中,当 N 设置为适当量级时,绝大多数查询均能返回精确的最近几何元素。

5 实验对比

本文在多个经典三维模型上对所提方法进行了系统评估,并与当前先进的点到网格距离查询方法进行了对比。所有实验均在配备 Apple M4 处理器与 16GB 内存的 Mac mini 上运行,操作系统为 macOS。为保证对比的公平性,所有参与评估的方法均未手动开启向量化编译选项,所有方法在相同的编译配置下运行。参与对比的方法包括 FCPW, Embree 与 Libigl,三者均为目前点到网格距离查询领域广泛使用的高性能实现。查询点在模型轴对齐包围盒各维度 5 倍缩放后的空间内随机生成,实际查询空间体积为原包围盒的 125 倍,以覆盖近场与远场两类典型查询场景。图 2 给出了实验所用的经典三维模型,包括 Bunny, Dragon, Armadillo 等,这些模型在面片数量与几何复杂度上各有不同,能够较为全面地反映算法在不同几何特征下的性能表现。在所有实验中,总代理点数量 N 均设置为面片数目的 3.5 倍左右,在精度与效率之间取得了良好的平衡。



图 2 本文实验所用的经典模型展示

Fig. 2 Showcase of classic models used in experiments

5.1 查询精度分析

作为一种近似算法,精度是衡量本文方法实用价值的核心指标之一。尽管本文方法以牺牲少量精度换取更高的查询效率,但在实际应用中,近似结果与精确结果之间的偏差必须

控制在可接受的范围之内,方能满足工程应用的需求。为此,本文在计算机图形学领域广泛使用的标准模型上对算法的近似精度进行了系统评估,将所提方法返回的近似距离与精确的点到网格距离进行逐查询点比较,统计正确返回最近距离比例作为精度指标。

表 1 列出了各标准模型上的精度统计结果。其中 FCPW, Embree, Libigl 为精确算法,其准确率恒为 100%,在此一并列出作为参照。可以看到,在所有测试模型上,本文方法的查询准确率均在 99.5% 以上,绝大多数查询均能返回正确的最近距离,验证了其在实际应用中的可靠性。

此外,为更直观地呈现误差的分布特征,图 3 进一步展示了在坐标归一化至 $[0,1]^3$ 的标准 Lucy 与 Armadillo 模型上,各误差区间内查询点的分布情况。结果表明,绝大多数查询的距离误差集中在极小范围内,最大误差量级约为 10^{-3} ,对于绝大多数图形学与 CAD 应用场景而言,这完全可以忽略。进一步地,考虑到在曲面配准、点画等应用中最近点的位置往往比距离值更重要,本文统计了所提方法返回的最近点与精确最近点之间的欧氏距离,并以模型包围盒对角线长度进行归一化。在全部查询中,归一化位置误差处于 10^{-1} 至 10^{-5} ,各量级的占比分别为 0.007%, 0.02%, 0.03%, 0.05% 与 0.1%,其余 99.8% 的查询位置误差均在 10^{-5} 以下。可见,绝大多数查询返回的最近点与精确解几乎重合,本文方法在最近距离与最近点位置上均能以极高的概率返回与精确解一致的结果。综合来看,本文方法在保持高查询效率的同时,能够以极高的概率返回精确结果,近似误差在实际应用中处于可控范围之内。

表 1 各三维模型上返回正确最近距离的查询占全部查询的比例
Table 1 Percentage of queries returning the correct nearest distance across 3D models

	Bunny	Camel	Dragon	Lucy	Sponza	Kitten	Armadillo
Faces	5 000	20 000	100 000	100 000	262 000	269 000	346 000
Libigl/%	100	100	100	100	100	100	100
Embree/%	100	100	100	100	100	100	100
FCPW/%	100	100	100	100	100	100	100
Ours/%	99.9	99.8	99.8	99.7	99.7	99.9	99.6

注:Faces 表示每个模型的三角形总数。

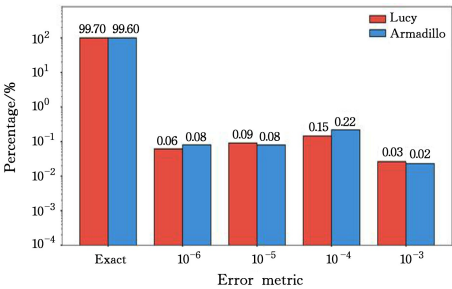


图 3 误差比例
Fig. 3 Error distribution

5.2 查询速度对比

查询效率是点到网格距离查询算法最核心的性能指标。表 2 列出了本文方法与 FCPW, Embree, Libigl 在多个经典三维模型上的平均单次查询时间。

表 2 不同模型的距离查询性能比较

Table 2	Performance comparison of distance queries across models						
	Bunny	Camel	Dragon	Lucy	Sponza	Kitten	Armadillo
Faces	5 000	20 000	100 000	100 000	262 000	269 000	346 000
Libigl/ μ s	2.35	3.46	3.79	3.19	1.34	19.28	9.92
Embree/ μ s	1.83	2.21	2.85	2.04	<u>0.83</u>	14.99	9.98
FCPW/ μ s	<u>1.37</u>	<u>1.78</u>	<u>2.03</u>	<u>1.59</u>	0.79	<u>10.1</u>	<u>6.67</u>
Ours/ μ s	0.57	0.65	0.65	0.61	0.97	1.27	1.07

注:最佳结果用黑体+下划线表示;次佳结果用下划线表示。

结果表明,在绝大多数测试模型上,本文方法的查询速度均优于对比方法,体现了在实际场景中的效率优势。这一优势在查询点远离模型表面时尤为显著,原因在于本文方法的剪枝能力不依赖于查询点与模型表面的距离,而传统 BVH 类方法在远场查询时需要访问更多的树节点。

为进一步分析本文方法在不同网格规模下的可扩展性,将 Lucy 模型重新网格化至不同的面片密度,图 4 给出了各方法的查询时间随面片数目变化的曲线。可以看到,随着网格密度的增加,各方法的查询时间均有所上升,这与面片数量增多导致候选集规模扩大的直觉相符。然而,本文方法在不同网格密度下始终保持明显的速度优势,且随着面片数目的增大,该优势并未出现明显收窄,这表明本文方法具有良好的可扩展性,适用于大规模网格模型的高频距离查询场景。

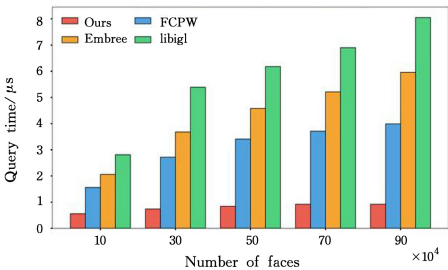


图 4 各方法在不同面片数目 Lucy 模型上的查询速度对比
Fig. 4 Comparison of query speed among different methods on the Lucy model at varying face counts

此外,为分析本文方法在不同查询距离下的表现,以 Lucy 模型为例,通过调整查询空间相对模型包围盒的缩放尺度(Scale)来控制查询点到表面的远近,Scale 越大,查询点越远离表面。各方法的平均查询时间随 Scale 的变化如图 5 所示。

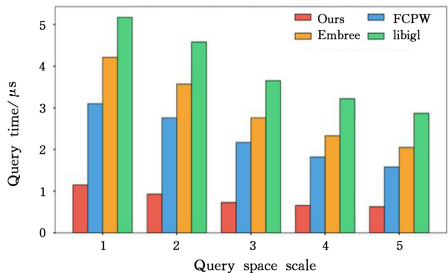


图 5 标准 Lucy 模型上,不同查询空间缩放尺度(Scale)下各方法的平均单次查询时间

Fig. 5 Average per-query time of each method under different query-space scaling factors(Scale) on standard Lucy model

可以看到,在该场景下,随着 Scale 增大,查询点远离模

型表面,各方法的查询时间均有所缩短;而在所有缩放尺度下,本文方法相对 FCPW,Embree 与 Libigl,均保持显著的速度优势,表明其在近场与远场场景下均具有良好的适用性。

5.3 预处理开销

预处理是算法整体性能的重要组成部分。由于本文方法需要在预处理阶段完成增量 Delaunay 三角化的构建与面片级查询表的生成,其预处理时间相较于基于 BVH 的方法有所增加,这也是本文方法与传统方法在设计权衡上的本质差异所在——以更充分的预处理换取更高效的查询性能。因此,本文方法更适合应用于查询密集型场景:当查询次数足够多时,查询阶段节省的时间将远超预处理阶段的额外开销,整体收益显著。表 3 列出了本文方法在多个经典模型上的预处理时间。图 6 进一步展示了在不同网格密度的 Lucy 模型上,预处理时间随面片数目的变化趋势,结果与第 4.4 节的理论分析一致,预处理时间随面片数目的增加呈现出平滑的增长态势,具有良好的可预测性。

表 3 预处理速度

Table 3 Preprocessing speed

	Bunny	Camel	Dragon	Lucy	Sponza	Kitten	Armadillo
Faces	5 000	20 000	100 000	100 000	262 000	269 000	346 000
Time/s	0.11	0.45	2.58	2.61	11.1	7.57	9.56

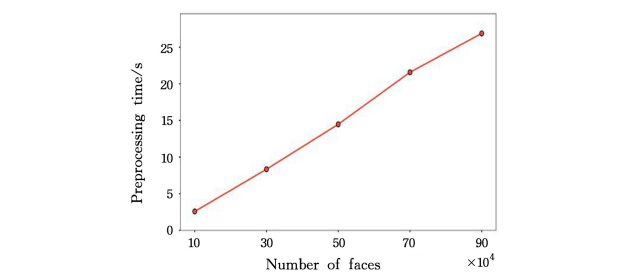


图 6 本文方法在不同面片数目 Lucy 模型上预处理时间变化

Fig. 6 Preprocessing time of our method on the Lucy model at varying face counts

结束语 本文提出了一种基于 DP-NNS 框架的近似点到网格距离查询算法。针对 DP-NNS 无法直接处理三角网格几何元素的核心障碍——三角网格的广义 Voronoi 图在三维空间中缺乏实用构造工具且计算复杂度过高,本文提出以采样代理点近似各几何元素的广义 Voronoi 胞元,将顶点、边、面 3 类几何元素的邻接关系统一转换为代理点之间的 Delaunay 邻接关系,从而将 DP-NNS 的增量构造框架近似延伸至几何元素粒度。其中,顶点直接作为代理点插入,边与面则基于统一的密度参数 ρ 自适应采样代理点,以保证不同尺度元素之间邻接关系捕获能力的一致性。查询阶段针对 3 类元素的几何特性分别设计距离计算策略,对面片结合包围球估计实现高效剪枝,在保持较高近似精度的同时实现了高效的候选元素筛选。

在多个标准三维模型上的实验结果表明,本文方法在查询精度上达到 99.5% 以上,最大距离误差维持在 10^{-3} 量级,能够满足绝大多数图形学与 CAD 应用场景的精度需求。在查询效率上,本文方法在经典模型及不同网格密度的测试中均优于 FCPW,Embree 和 Libigl 等现有方法,在查询密集型

场景下具有显著的实用价值。

目前,本文方法存在两个主要局限。1) 预处理阶段需要构建完整的增量 Delaunay 三角化与几何元素级查询表,预处理开销高于 BVH 类方法,在查询次数较少的场景下整体收益有限;2) 对于包含大量极狭长三角形或极短边的网格,广义 Voronoi 分界面附近的窄带区域难以被代理点充分覆盖,可能导致局部精度下降。未来工作将探索更高效的预处理策略以降低构建开销,并研究针对几何退化元素的自适应加密采样方案,进一步提升本文方法在复杂几何场景下的鲁棒性。

参 考 文 献

[1] REZAABBASIFARD M,GHAHREMANI B,NADERI H. A survey on nearest neighbor search methods[J]. International Journal of Computer Applications,2014,95:39-52.

[2] AUER S,WESTERMANN R. A semi-lagrangian closest point-method for deforming surfaces[J]. Computer Graphics Forum, 2013,32.

[3] GUÉZIEC A. ‘meshsweeper’: Dynamic point-to-polygonal-mesh distance and applications[J]. IEEE Transactions on Visualization and Computer Graphics,2001,7:47-61.

[4] WALD I,USHER W,MORRICAL N,et al. Rtx beyond ray tracing:exploring the use of hardware ray tracing cores for tet-mesh point location[C] // Proceedings of the Conference on High-Performance Graphics. 2019.

[5] BENTLEY J L. Multidimensional binary search trees used for associative searching[J]. Communications of the ACM,1975, 18:509-517.

[6] ZONG C,XU J C,SONG J T,et al. P2m: A fast solver for querying distance from point to mesh surface[J]. ACM Transactions on Graphics,2023,42(4):1-13.

[7] WANG P F,SONG J T,XIN S Q,et al. Efficient nearest neighbor search using dynamic programming[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence,2026,48(1):999-1014.

[8] SAWHNEY R. Fcpw: Fastest closest points in the west[EB/OL]. <https://github.com/rohan-sawhney/fcpw>.

[9] WALD I,WOOP S,BENTHIN C,et al. Embree: a kernel framework for efficient cpu ray tracing[J]. ACM Transactions on Graphics,2014,33(4):143.

[10] JACOBSON A,PANOZZO D. libigl: A simple C++ geometry processing library[EB/OL]. <https://libigl.github.io/>.

[11] FRIEDMAN J H,BENTLEY J L,FINKEL R A. An algorithm for finding best matches in logarithmic expected time[J]. ACM Transactions on Mathematical Software,1977,3:209-226.

[12] BLANCO J L,RAI P K. nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor(NN) with kd-trees[EB/OL]. <https://github.com/jlblancoc/nanoflann>.

[13] DROST B,ILIC S. Almost constant-time 3d nearest-neighbor lookup using implicit octrees[J]. Machine Vision and Applications,2017,29:299-311.

[14] GUTTMAN A. R-trees: a dynamic index structure for spatial searching[C] // ACM SIGMOD Conference. 1984.

[15] BECKMANN N,KRIEGEL H P,SCHNEIDER R,et al. The r^* -tree:an efficient and robust access method for points and rectangles [C]// ACM SIGMOD Conference. 1990.

[16] JAMES D L,PAI D K. Bd-tree:output-sensitive collision detection for reduced deformable models[J]. ACM Transactions on Graphics,2004,23(3):393-398.

[17] WU Z C,MA L,CHEN Y Y,et al. Instant Color Stippling with Incremental Voronoi Sequence[J]. Journal of Computer-Aided Design & Computer Graphics,2019,31(9):1502-1508.

[18] DU X Y,YAN D M,YE J T,et al. Quad Mesh Generation via Field-Aligned Centroidal Voronoi Tessellation[J]. Journal of Computer-Aided Design & Computer Graphics, 2018, 30 (5) : 764-771.

[19] CHEN Z G,OUYANG Y S,CAO J. Feature-Preserving Mosaic Image Generation [J]. Journal of Computer-Aided Design & Computer Graphics,2014,26(4):520-527.

[20] BOWYER A. Computing dirichlet tessellations * [J]. The Computer Journal,1981,24(2):162-166.

[21] WATSON D F. Computing the n-dimensional delaunay tessellation with application to voronoi polytopes * [J]. The Computer Journal,1981,24(2):167-172.

[22] HERT S,SEEL M. dD convex hulls and delaunay triangulations [M] // CGAL User and Reference Manual. CGAL Editorial Board,2026.

[23] SI H. Tetgen,a delaunay-based quality tetrahedral mesh generator[J]. ACM Transactions on Mathematical Software, 2015, 41(2):1-36.



WANG Pengfei, born in 1998, Ph.D candidate, is a member of CCF (No. A18958G). His main research interests include computational geometry and computer graphics.



CHEN Shuangmin, born in 1982, Ph.D, associate professor, is a member of CCF (No. J1285M). Her main research interests include computational geometry and computer graphics.

(责任编辑:何杨)