arXiv:2512.19743v1 [cs.LG] 17 Dec 2025

# 从理论到吞吐量：面向大批次三维学习的 CUDA 优化 APML

Sasan Sharifipour*、Constantino ´Alvarez Casado*、Manuel Lage Ca˜nellas、Miguel Bordallo L´opez机器视觉与信号分析中心 *(CMVS)*，奥卢大学，芬兰

摘要—**损失函数对于学习准确的三维点云模型至关重要，但常用方法需要在几何保真度与计算成本之间取舍。倒角距离计算高效，但允许多对一对应关系；推土机距离更能体现一对一传输，但计算成本高昂。APML 使用可微分的 Sinkhorn 迭代和解析推导的温度来近似传输，但其稠密形式的内存开销随规模呈平方增长。我们提出 CUDA-APML，这是一种稀疏 GPU 实现，通过阈值筛除可忽略的分配，并直接在 COO 格式中执行自适应 softmax、双向对称化和 Sinkhorn 归一化。这使内存开销近似线性增长，并保留存储的支撑集上的梯度；不过，当前实现中的成对距离计算仍呈平方级开销。在 ShapeNet 和 MM-Fi 上，CUDA-APML 的结果与稠密 APML 的差异在较小容差范围内，同时将峰值GPU内存降低了 99.9%。代码见：https://github.com/Multimodal-Sensing-Lab/apml**

索引词—**点云学习、最优传输、自适应概率匹配损失、CUDA 优化、稀疏 Sinkhorn、三维重建**

## I. 引言

三维点云表示来自激光雷达、深度相机和结构光扫描仪的几何信息，也出现在从非视觉测量中推断三维结构的跨模态管线中。点云学习应用于机器人技术、增强现实和虚拟现实、无线人体感知以及数字孪生流水线 [1]–[4]。在有监督场景中，模型输出无序点集，因此训练需要在点数较大时仍保持稳定的置换不变损失。

匹配质量与可扩展性往往难以兼顾。最近邻目标函数效率较高，但允许多个点映射到同一点，从而可能降低几何覆盖率；而最优传输损失更能体现一对一匹配，但在大规模训练中代价高昂。这一成本限制了批大小和点数，减少了固定计算预算下可行的训练次数，并增加了 GPU 训练和调优过程中的能耗与经济成本。APML [5] 通过温度缩放相似度和 Sinkhorn 归一化构造可微的传输计划，并使用解析推导的温度，从而缩小了这一差距。然而，其稠密形式需要存储成对矩阵，内存占用随规模呈二次增长，可能会大量消耗训练资源。例如，[5] 报告称，训练 FoldingNet 时 GPU 内存接近 300 GB 在 ShapeNet-55 [7] 上。此类需求限制了其在更高分辨率场景中的应用，而这类场景常见于大型场景和反复更新中，包括映射和数字孪生流程[8]–[13]。

本研究得到芬兰研究委员会 6G 旗舰计划（资助编号 346208）、欧盟地平线计划 CONVERGE（资助编号 101094831）和芬兰商业署 WISECOM（资助编号 3630/31/2024）的支持。*这些作者贡献相同。

我们提出 CUDA-APML，这是一种稀疏 GPU 实现的 APML，可利用自适应 softmax 阈值化后传输计划呈现的经验稀疏性。CUDA-APML 直接以 COO 格式构建双向指派，在设备上对稀疏支撑集执行对称化和 Sinkhorn 归一化，并计算损失和梯度，而无需实例化稠密矩阵。在 ShapeNet 和 MM-Fi 上，CUDA-APML 的结果与稠密 APML 的差异在较小容差范围内，同时将峰值GPU内存降低多达 99.9%，从而改善了基于传输的监督在点云学习和跨模态三维学习中的实际成本表现。

## II. 相关工作

点云学习需要使用置换不变损失，以比较无序的预测点集与参考点集。倒角距离（CD）因将比较简化为双向计算的最近邻距离而被广泛采用 [2], [14]。然而，其多对一分配可能导致密集区域中的点聚类，以及稀疏区域中的覆盖不足 [15]。密度感知倒角距离（DCD） [16], 、HyperCD [15], 和 InfoCD[17] 等扩展方法缓解了其中一些影响，但仍依赖最近邻匹配，并且仍容易受到采样密度和局部歧义的影响。相比之下，推土机距离（EMD）强制执行一对一传输，能够更好地反映结构对齐 [18],，但其计算成本以及常见的等基数约束限制了其在训练中的应用。其他方法包括切片 Wasserstein 距离等投影距离，以及近似传输求解器 [19], [20], ；这些方法通常会引入额外的设计选择，并且在异质点分布上可能无法匹配完整的传输行为。

熵正则化最优传输提供了 EMD 的可微近似，可通过 Sinkhorn 归一化进行计算 [21], [22]。基于 Sinkhorn 的损失已用于配准和特征匹配等软对应问题 [23], [24],，但通常需要调整正则化强度，以平衡尖锐度和稳定性。APML [5]根据成对距离构建温度缩放相似度矩阵，并应用 Sinkhorn 归一化，以获得边缘分布近似为一对一的软对应关系。其温度源自局部

距离间隔，以强制达到最小分配概率，从而减少手动调整；但其稠密形式会存储并更新成对矩阵，导致内存呈二次增长，这在实践中限制了批大小和点数 [5]。

缩放最优传输已通过稀疏或结构化支撑集限制以及低秩近似进行了研究[25]–[27]。与此同时，内核级设计表明，移除稠密中间结果可以改变二次算子的实际性能上限，例如 FlashAttention [28] 和 KeOps [29]。在三维感知领域，稀疏张量引擎和基于点的库为不规则稀疏模式提供 GPU 支持，以改善内存行为并提升吞吐量 [30], [31]。这些方向促使我们实现稀疏且面向 GPU 的 APML。在本工作中，我们利用 APML 传输计划经过自适应 softmax 阈值化后的经验稀疏性，并在稀疏支撑集上执行对称化和 Sinkhorn 归一化，以减少稠密矩阵占用，同时将目标函数保持在可控的近似范围内。

## III. 所提出的方法

本节介绍 CUDA-APML，这是一种 APML 的稀疏 GPU 实现 [5]。其目标是在前向和反向传播期间避免使用稠密 $N\times M$张量，同时保留 APML 的公式及其在稀疏支撑集上的可微性。CUDA-APML 直接以稀疏坐标（COO）形式构建按行和按列的自适应 softmax 分配，在设备上对其进行对称化，在稀疏支撑集上应用 Sinkhorn 归一化，并仅使用已存储的非零项计算损失。

### *A.* 背景：稠密 *APML*

设 $X=\{x_i\}_{i=1}^{N}$ 为预测点，$Y=\{y_j\}_{j=1}^{M}$ 为参考点，且 $x_i,\ y_j\in\mathbb{R}^d$。APML 从成对代价矩阵 $C\in\mathbb{R}^{N\times M}$ 开始，其中 $C_{ij}=\|x_i-y_j\|_2$。对于一般的代价向量 $c\in\mathbb{R}^K$，APML 将其平移为 $\tilde{c}_k=c_k-\min_\ell c_\ell$，使得 $\min_k \tilde{c}_k=0$。设 $\tilde{c}^{(2)}$ 表示 $\tilde{c}$ 中第二小的值。局部间隔定义为 $g=\tilde{c}^{(2)}+\delta$，其中 $\delta>0$。APML 计算温度 $T$，在 [5], 中给出的近似下，确保最小值对应的条目具有最小概率质量 $p_{\min}$：

$$T=\frac{-\log\left(\frac{1-p_{\min}}{(K-1)p_{\min}}\right)}{g},\qquad 0<p_{\min}<1,\quad K>1. \tag{1}$$

当 $\tilde{c}^{(2)}<\epsilon_g$ 时，APML 会应用均匀回退分布以确保数值稳定性。

APML 在 $C$ 上按行和按列应用自适应温度 softmax，生成 $P_{\text{row}}$ 和 $P_{\text{col}}$，并将$P^{(0)}=\frac{1}{2}(P_{\text{row}}+P_{\text{col}})$ 初始化。随后，Sinkhorn 归一化以稳定性常数 $\epsilon_{\text{stab}}$ [5] 交替执行列缩放和行缩放，共进行 $L_{\text{iter}}$ 次迭代。APML 目标函数是 Frobenius 内积 $\mathcal{L}_{\text{APML}}=\langle P,\ C\rangle_F=\sum_{i=1}^{N}\sum_{j=1}^{M}P_{ij}C_{ij}$。稠密形式存储并更新 $C$ 和 $P$，从而产生$O(NM)$ 的内存开销。

### *B. APML* 初始化的稀疏性

CUDA-APML 基于 [5] 中报告并在我们的测量中得到确认的一项经验性质。在自适应 softmax 阶段之后，未归一化相似度矩阵中的大多数条目都接近于零。对于固定行 $i$，未归一化相似度可写为 $s_{ij}^{\text{row}}=\exp(-T_i(C_{ij}-C_{i\min}))$，其中 $C_{i\min}=\min_j C_{ij}$，其中 $T_i$ 通过公式 (1) 中的行间隙计算得到。相应的归一化概率定义如下：

$$p_{ij}^{\text{row}}=\frac{s_{ij}^{\text{row}}}{\sum_{k=1}^{M}s_{ik}^{\text{row}}}. \tag{2}$$

列方向也有类似的定义，得到 $p_{ij}^{\text{col}}$。

CUDA-APML 引入剪枝阈值 $\tau$，应用于未归一化相似度，并仅保留支撑集中的索引对，定义为 $\Omega_\tau^{\text{row}}=\{(i,\ j):\ s_{ij}^{\text{row}}\ge\tau\}$，以及 $\Omega_\tau^{\text{col}}=\{(i,\ j):\ s_{ij}^{\text{col}}\ge\tau\}$。稀疏初始化保留并集支撑集 $\Omega_\tau=\Omega_\tau^{\text{row}}\cup\Omega_\tau^{\text{col}}$，并仅以 COO 格式存储这些条目。目标是在保留传输计划中占主导地位的概率质量的同时，减少内存流量和存储开销。

### *C.* 稀 疏 *COO* 自适应 *softmax* 与 *GPU* 上的 *Sinkhorn* 上

每行和每列保留的条目数量取决于数据，因此 CUDA-APML 会在每个方向执行两轮 GPU 遍历来构建 COO 缓冲区。第一轮中，每个线程块扫描一行或一列，实时计算代价 $C_{ij}$，跟踪最小值和次小值以获得局部温度，并统计满足 $\Omega_\tau^{\text{row}}$ 或 $\Omega_\tau^{\text{col}}$ 中剪枝规则的条目数。独占前缀和将这些计数转换为写入偏移量，并得出非零元素总数。第二轮中，内核重新扫描并重新计算未归一化相似度（在取指数之前减去 $C_{i\min}$），写入保留的 COO 三元组（**i**，**j**，**v**），并根据相应支撑集上保留相似度之和，在每行或每列内进行归一化。如果触发均匀回退，内核会在相应的行或列上写入均匀分布。由此得到 $\Omega_\tau^{\text{row}}$ 上的 $P_{\text{row}}$ 和 $\Omega_\tau^{\text{col}}$ 上的 $P_{\text{col}}$。

为在不进行稠密化的情况下实现对称化，CUDA- APML 将 COO 流拼接起来，将每个 $(i,j)$ 映射为一个 64 位键（例如键 $=i\cdot M+j$），在设备上按键排序，并通过对值取平均来归并重复项，从而在 $\Omega_\tau$ 上生成单个 COO 方案 $P^{(0)}$。随后，Sinkhorn 归一化在该稀疏支撑集上运行。设 $P$ 以 COO 格式存储，其条目为 $\{(i_t,j_t,v_t)\}_{t=1}^{\text{nnz}}$。一次迭代交替执行列缩放，

$$v_t^{(\ell+\frac{1}{2})}=\frac{v_t^{(\ell)}}{\sum_{t':\,j_{t'}=j_t}v_{t'}^{(\ell)}+\epsilon_{\text{stab}}}, \tag{3}$$

以及行缩放，

$$v_t^{(\ell+1)}=\frac{v_t^{(\ell+\frac{1}{2})}}{\sum_{t':\,i_{t'}=i_t}v_{t'}^{(\ell+\frac{1}{2})}+\epsilon_{\text{stab}}}, \tag{4}$$

通过累加内核实现（将原子加法应用于 denselength-$M$ 或 length-$N$ 缓冲区），然后进行缩放

内核会更新所有 COO 值。这样既能保持计划稀疏，也能避免使用任何 $N \times M$ 张量。我们像稠密 APML[5] 中那样使用 $L_{\text{iter}}$。

### D. 损失、梯度和稳定性

经过 Sinkhorn 归一化后，CUDA-APML 在 COO 支持集上计算目标函数，如下所示： $\mathcal{L}_{\text{CUDA-APML}} = \sum_{t=1}^{\text{nnz}} v_t \, \|x_{i_t} - y_{j_t}\|_2$. 不会形成稠密代价矩阵 $C$，前向传播仅计算已存储索引对的距离。反向传播则通过同一个稀疏计算图进行求导。具体而言，距离项会在已存储的索引对上贡献标准的成对梯度:

$$\frac{\partial}{\partial x_{i_t}} \|x_{i_t} - y_{j_t}\|_2 = \frac{x_{i_t} - y_{j_t}}{\|x_{i_t} - y_{j_t}\|_2 + \epsilon_{\text{dist}}}, \tag{5}$$

其中， $\epsilon_{\text{dist}} > 0$ 可避免两个点重合时发生除零。传输权重 $v_t$ 相对于 $X$ 和 $Y$ 的梯度，通过稀疏支撑集上的自适应 softmax 和 Sinkhorn 更新的融合内核进行计算。剪枝条目被视为零，不参与前向传播或反向传播，从而抑制较弱的竞争对应关系，并增强主导梯度信号。

CUDA-APML 遵循 APML [5]的稳定性条件。通过将局部间隔截断为 max(gap$^{\epsilon_g}$)，而不是采用均匀回退，确保了稳定性。Sinkhorn 归一化使用 $\epsilon_{\text{stab}}$，稀疏性则由剪枝阈值 $\tau$控制。

### E. 复杂度、内存与实现细节

算法 1 概述了单个批次的 CUDA-APML 流水线，并强调了避免使用稠密张量的环节。

**算法 1**单个批次的 CUDA-APML**要求：** 预测 $X \in \mathbb{R}^{B\times N\times d}$、参考 $Y \in \mathbb{R}^{B\times M\times d}$、$p_{\min}$、 $\tau$、 $L_{\text{iter}}$1:**行方向：** 扫描每一行，计算最小值和温度，统计相似度为 $\geq \tau$ 的配对数，通过排他扫描获取偏移量，再次扫描以写入并归一化 COO 2: **列方向：** 交换行列角色并重复操作，以获得 $P_{\text{col}}$ 3: **对称化：** 拼接 COO，按 64 位键（$i$， $j$）排序，通过平均合并重复项以获得 $P^{(0)}$4: **对于** $\ell = 1$ 到 $L_{\text{iter}}$ **执行**5: 使用稠密列和缓冲区和 $\epsilon_{\text{stab}}$在 COO 中进行列缩放6: 使用稠密行和缓冲区和 $\epsilon_{\text{stab}}$在 COO 中进行行缩放7: **结束循环**8: **损失与反向传播：** 在 COO 配对上计算损失，并通过稀疏算子进行反向传播

设 $B$ 为批大小， $d$ 为点维度，nnz $= |\Omega_\tau|$ 为对称化后存储的 COO 条目数。稠密 APML 将代价矩阵$C$ 和传输计划 $P$ 存储为 $B \times N \times M$ 张量，因此需要 $O(BNM)$ 内存。使用单精度时，仅存储这两者至少需要 $2 \cdot 4 \cdot BNM$ 字节，不包括逐行和逐列阶段的中间缓冲区。计算开销主要来自成对距离计算和稠密 Sinkhorn 更新，其复杂度为 $O(BNMd + BNML_{\text{iter}})$。CUDA-APML 不会将 $C$ 和 $P$ 实例化为稠密张量，而是以 COO 格式存储传输计划。使用 32 位索引（i,j）和 32 位数值 v时，传输计划占用 12 nnz 字节，此外还需要用于对称化和 Sinkhorn 的临时缓冲区。对称化会为每个条目增加一个 64 位键和排序工作区。Sinkhorn 使用两个稠密一维累加缓冲区，长度分别为 $N$ 和$M$ ，用于计算行和与列和。总体而言，内存复杂度为$O(B(\text{nnz} + N + M))$，另需排序和扫描工作区带来的常数开销。

在当前实现中，自适应 softmax 阶段仍会扫描所有点对以确定最小值并进行阈值处理，因此距离评估仍为 $O(BNMd)$。运行时间的减少得益于内存流量降低，以及将 Sinkhorn 更新和损失评估限制在 nnz 个条目上。Sinkhorn 迭代的开销为 $O(B$ nnz $L_{\text{iter}})$，而对称化主要受排序影响，在比较模型中的开销为 $O$ (nnz log nnz)。除非另有说明，我们使用 $\tau = 10^{-8}$、 $L_{\text{iter}} = 10$ 和 $\epsilon_{\text{stab}} = 10^{-8}$，并按照 [5] 设置 $p_{\min}$。

## IV. 评估方法

### A. 数据集、模型和基线

我们在点云补全和跨模态生成任务上评估 CUDA-APML。对于点云补全，我们使用 ShapeNet-34 和 ShapeNet-55 [7] ，采用标准的部分到完整配对数据及官方划分，并使用 PCN [32]（2,048 点的部分输入和 16,384 点的参考点云）。对于跨模态生成，我们使用 MM-Fi [3], [33], ，其将 WiFi CSI 与基于激光雷达的 3D 人体点云配对；我们遵循 CSI2PC 手册_中的划分协议，以测试模型对未见过的受试者和环境的泛化能力 [3]。我们将损失集成到用于点云补全的 Fold- ingNet [6], 、PCN [32], 和 PoinTr [34] ，以及用于 MM-Fi 的 CSI2PC [3] 中。我们将 CUDA-APML 与 CD [14], 、DCD [16], 、HyperCD [15], 、InfoCD [17], 和稠密 APML [5] 进行比较。

### B. 指标与协议

我们报告 $\ell_1$ 和 $\ell_2$ 变体下的 CD [14],EMD$\times$100 [18], 以及阈值为 0.01 [32] 时的 F1 分数。效率通过每个训练轮次的实际运行时间和峰值GPU内存来衡量。每种配置使用不同的随机种子训练三次，并报告平均值。ShapeNet 结果采用标准测试集划分 [7] 以及各骨干网络的默认训练协议，而 MM-Fi 使用手动_划分 [3]。为支持第 III 节中的稀疏表示，我们还报告：(i) 在合成随机点集上，COO 条目经对称化后的非零数量随点数的变化，每种点数进行 500 次试验；(ii) 在相同的合成设置下，稠密 APML 和 CUDA-APML 的每样本峰值内存。

### C. 实现细节与测量

所有实验均使用 PyTorch 2.5 和 CUDA 12.8，并在 NVIDIA V100（32GB）和 RTX 4070 GPU 上运行。除非

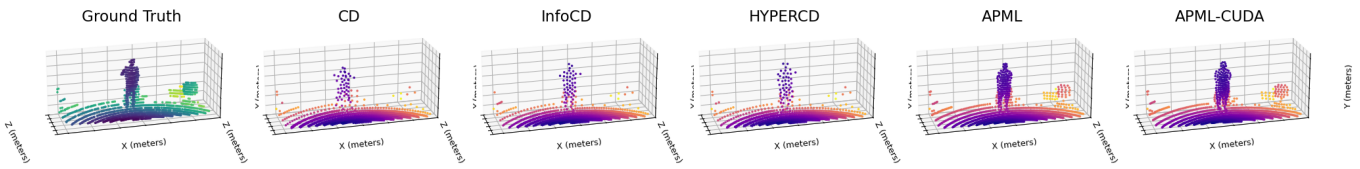


图1. MM-Fi 样本的定性比较。从左到右：真实值，以及使用 CD、InfoCD、HyperCD、APML 和 CUDA-APML 得到的预测结果。

另有说明，我们使用 Adam，学习率为 $10^{-4}$，批大小为 32，并遵循各骨干网络的官方训练计划和预处理流程。对于 CUDA-APML，COO 构造在设备上运行，Sinkhorn 使用 $L_{\text{iter}} = 10$ 和$\epsilon_{\text{stab}} = 10^{-8}$，除非另有说明，剪枝阈值为 $\tau = 10^{-8}$。每轮运行时间使用 PyTorch分析器测量，峰值 GPU内存则使用 PyTorch CUDA内存统计和 NVIDIA 工具（包括 nvidia-smi 和 CUDA 分析器接口）测量，并报告每次运行中观测到的最大值。

## V. 实验结果

### A. *ShapeNet* 和 *MM-Fi* 上的定量结果

表 I 报告了主要基准测试结果（三次运行的均值）。在使用 FoldingNet 的 ShapeNet-34 上，Dense-APML 和 CUDA-APML 的结果相近；与 CD、HyperCD 和 InfoCD 相比，它们改善了 EMD，并取得最高的 F1 分数。在使用 CSI2PC 的 MM-Fi 上，CUDA-APML 相较于 DCD 改善了 EMD，且结果略优于 Dense-APML。

表 I ShapeNet-34（补全）和 MM-Fi（生成）上的损失函数比较。指标：F1 分数（↑）和 EMD×100（↓）。

| 数据集 / 模型 | Loss | F1 分数 ↑ | EMD×100 ↓ |
|---|---|---|---|
| ShapeNet-34 / FoldingNet | CD | 0.11 | 27.7 |
| | HyperCD | 0.19 | 16.5 |
| | InfoCD | 0.18 | 22.3 |
| | Dense-APML | **0.20** | **9.5** |
| | CUDA-APML | **0.20** | **9.5** |
| MM-Fi / CSI2PC | DCD | – | 25.7 |
| | 稠密 APML | – | 16.3 |
| | CUDA-APML | – | **16.1** |

### B. 效率与扩展行为

表II展示了在 ShapeNet-55 上使用 FoldingNet（批大小为 32）时的峰值 GPU 内存。CUDA-APML 在所有测试点数下均能降低内存占用。图 2 展示了合成扩展性测试结果：左图显示对称化后 COO 非零元素的数量（每个 $N$ 进行 500 次试验，最多至 262，144），右图则显示在相同设置下，稠密 APML 和 CUDA-APML 每个样本的损失侧峰值内存。每轮运行时间和峰值 GPU 内存的测量方法如第 IV-C 节所述。

### C. 定性结果

图1对比了真实点云与 CD、InfoCD、HyperCD、APML 和 CUDA-APML 的预测结果。在所示的 MM-F 样本中，基于倒角距离的损失

表II FoldingNet 在 ShapeNet-55 上的峰值GPU内存。

| 点数 | 稠密 APML | CUDA-APML |
|---|---|---|
| 1k | 50 MB | 0.39 MB |
| 8k | 1.6 GB | 2.6 MB |
| 32k | 18.5 GB | 8.6 MB |
| 65k | 68 GB | 17.1 MB |

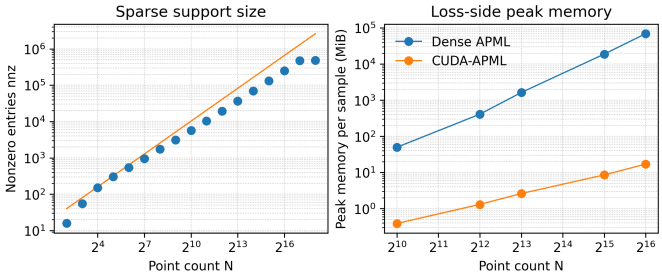


图 2. CUDA-APML 在合成点集上的扩展性测量。左：对称化后非零 COO 条目的数量随点数变化的情况（每个 $N$ 进行 500 次试验）。右：在相同设置下，稠密 APML 和 CUDA-APML 的每样本峰值内存。

会导致人体轮廓中出现更多孤立点和空隙。APML 和 CUDA-APML 生成的覆盖更连贯，缺失区域也更少。

## VI. 结论

我们提出了 CUDA-APML，这是一种 APML 的稀疏 CUDA 实现，可消除点云监督中稠密传输计划构建带来的二次内存瓶颈。CUDA-APML 以 COO 格式构建按行和按列的自适应 softmax 分配，在设备上进行对称化，并在稀疏支撑集上应用 Sinkhorn 归一化，而无需实例化稠密 $N \times M$ 张量。在 ShapeNet 和 MM-Fi 上，CUDA-APML 保持了稠密 APML 的准确率趋势，同时将峰值 GPU 内存降低了 99.9% 以上；在合成测试中，其存储的支撑集和损失侧内存呈现近线性扩展，因此当稠密传输损失受内存限制时，APML 也能切实可用。

一个局限在于，自适应Softmax阶段仍会扫描所有成对距离，因此距离计算的复杂度仍随点数呈二次增长。未来的工作将限制候选支撑集，例如通过邻域剪枝或近似最近邻，并在更大规模的场景级和数字孪生更新设置中评估该方法。

致谢

本研究获得了芬兰研究理事会（6G旗舰计划，资助编号 346208）、欧盟地平线计划 CON- VERGE（资助编号 101094831）以及芬兰商业署 WISECOM（资助编号 3630/31/2024）的支持。Sasan Sharifipour 感谢芬兰人工智能博士项目网络 AI-DOC 提供的资助（决定编号 VN/3137/2024- OKM-6）。该项目由芬兰教育与文化部支持，并由芬兰人工智能中心（FCAI）主办。作者感谢芬兰 CSC–IT科学中心提供计算资源。

参考文献

[1] Y. Guo、H. Wang、Q. Hu、H. Liu、L. Liu 和 M. Bennamoun，“三维点云深度学习：综述”，*IEEE*模式分析与机器智能汇刊，第43卷，第12期，第4338–4364页，2020年。[2] C. R. Qi、H. Su、K. Mo 和 L. J. Guibas，“PointNet：用于三维分类和分割的点集深度学习”，*arXiv*预印本 *arXiv:1612.00720*，2017年。[3] T. M¨a¨att¨a、S. Sharifipour、M. Bordallo L´opez 和 C. ´Alvarez Casado，“通过变换器网络从 WiFi-CSI 数据生成时空三维点云”，载于 *IEEE*第五届联合通信与感知国际研讨会（*JC&S*），2025年，第1–6页。[4] F. Xue、W. Lu、Z. Chen 和 C. J. Webster，“从激光雷达点云走向数字孪生城市：基于格式塔原理的城市对象聚类”，国际摄影测量与遥感杂志，第167卷，第418–431页，2020年。[5] S. Sharifipour、C. ´Alvarez Casado、M. Sabokrou 和 M. Bordallo L´opez，“APML：用于鲁棒三维点云重建的自适应概率匹配损失”，*arXiv*预印本 *arXiv:2509.08104*，2025年。[6] Y. Yang、C. Feng、Y. Shen 和 D. Tian，“FoldingNet：通过深度网格变形实现的点云自编码器”，载于 *IEEE*计算机视觉与模式识别会议（*CVPR*）论文集，2018年，第206–215页。[7] A. X. Chang、T. Funkhouser、L. J. Guibas、P. Hanrahan、Q. Huang、Z. Li、S. Savarese、M. Savva、S. Song、H. Su 等，“ShapeNet：信息丰富的三维模型库”，*arXiv*预印本 *arXiv:1512.03012*，2015年。[8] A. Dai、A. X. Chang、M. Savva、M. Halber、T. Funkhouser 和 M. Nießner，“ScanNet：带有丰富标注的室内场景三维重建”，载于 *IEEE*计算机视觉与模式识别会议论文集，2017年，第5828–5839页。[9] J. Behley、M. Garbade、A. Milioto、J. Quenzel、S. Behnke、C. Stachniss 和 J. Gall，“SemanticKITTI：用于激光雷达序列语义场景理解的数据集”，载于 *IEEE/CVF*国际计算机视觉会议论文集，2019年，第9297–9307页。[10] H. Caesar、V. Bankiti、A. H. Lang、S. Vora、V. E. Liong、Q. Xu、A. Krishnan、Y. Pan、G. Baldan 和 O. Beijbom，“nuScenes：用于自动驾驶的多模态数据集”，载于 *IEEE/CVF*计算机视觉与模式识别会议论文集，2020年，第11 621–11 631页。[11] T. Hackel、N. Savinov、L. Ladicky、J. D. Wegner、K. Schindler 和 M. Pollefeys，“Semantic3D.net：新的大规模点云三维分类基准”，*arXiv*预印本 *arXiv:1704.03847*，2017年。[12] L. Landrieu 和 M. Simonovsky，“利用超点图进行大规模点云语义分割”，载于 *IEEE*计算机视觉与模式识别会议论文集，2018年，第4558–4567页。[13] O. Wysocki、B. Schwab、M. K. Biswanath、M. Greza、Q. Zhang、J. Zhu、T. Froech、M. Heeramaglore、I. Hijazi、K. Kanna 等，“TUM2TWIN：介绍大规模多模态城市数字孪生基准数据集”，*arXiv*预印本 *arXiv:2505.07396*，2025年。[14] H. Fan、H. Su 和 L. J. Guibas，“用于从单张图像进行三维物体重建的点集生成网络”，载于 *IEEE*计算机视觉与模式识别会议（*CVPR*）论文集，2017年，第605–613页。[15] M. Lin、Y. Xu、Z. Wang 和 Y. Liu，“用于点云补全的双曲钱弗距离”，载于 *IEEE/CVF*计算机视觉与模式识别会议（*CVPR*）论文集，2023年，第19 176–19 185页。[16] M. Wang、X. Luo 和 Y. Li，“密度感知倒角距离：点云补全的综合度量”，载于 IEEE/CVF国际计算机视觉会议（ICCV），2021年，第13 014–13 023页。[17] F. Lin、Y. Yue、Z. Zhang、S. Hou、K. Yamada、V. Kolachalama 和 V. Saligrama，“InfoCD：用于点云补全的对比倒角距离损失”，载于神经信息处理系统进展（NeurIPS），第36卷，2023年，第76 960–76 973页。[18] Y. Rubner、C. Tomasi 和 L. J. Guibas，“推土机距离作为图像检索的度量”，国际计算机视觉期刊，第40卷，第2期，第99–121页，2000年。[19] W. Wu、Z. Qi 和 L. Fuxin，“PointConv：三维点云上的深度卷积网络”，载于 IEEE计算机视觉与模式识别会议（CVPR），2019年，包含用于监督的切片 Wasserstein 距离变体。[20] A. Genevay、G. Peyr´e 和 M. Cuturi，“使用 Sinkhorn 散度学习生成模型”，载于国际人工智能与统计会议（AISTATS），2018年，第1608–1617页。[21] M. Cuturi，“Sinkhorn距离：最优传输的高速计算”，载于神经信息处理系统进展（NeurIPS），第26卷，2013年。[22] G. Peyr´e 和 M. Cuturi，《计算最优传输：及其在数据科学中的应用》。机器学习基础与趋势，2019年。[23] P. E. Sarlin、D. DeTone、T. Malisiewicz 和 A. Rabinovich，“Superglue：利用图神经网络学习特征匹配”，载于 IEEE/CVF计算机视觉与模式识别会议（CVPR），2020年，第4938–4947页。[24] Z. Shen、J. Feydy、P. Liu、A. H. Curiale 和 M. Niethammer，“利用鲁棒最优传输实现精确点云配准”，载于神经信息处理系统进展（NeurIPS），2021年。[25] M. Blondel、V. Seguy 和 A. Rolet，“平滑稀疏最优传输”，载于国际人工智能与统计会议（AISTATS），2018年，第880–889页。[26] M. Li、J. Yu、T. Li 和 C. Meng，“Sinkhorn算法的重要性稀疏化”，机器学习研究期刊，第24卷，第247期，第1–44页，2023年。[27] X. Tang、M. Shavlovsky、H. Rahmanian、E. Tardini、K. K. Thekumpara-mpil、T. Xiao 和 L. Ying，“利用稀疏牛顿迭代加速 Sinkhorn 算法”，arXiv预印本 arXiv:2401.12253，2024年。[28] T. Dao、D. Fu、S. Ermon、A. Rudra 和 C. R´e，“FlashAttention：具有 IO感知性的快速且节省内存的精确注意力”，神经信息处理系统进展，第35卷，第16 344–16 359页，2022年。[29] B. Charlier、J. Feydy、J. A. Glaunes、F.-D. Collin 和 G. Durif，“GPU上的内核运算：支持自动微分且不会发生内存溢出”，机器学习研究期刊，第22卷，第74期，第1–6页，2021年。[30] C. Choy、J. Gwak 和 S. Savarese，“四维时空卷积网络：Minkowski卷积神经网络”，载于 IEEE/CVF计算机视觉与模式识别会议，2019年，第3075–3084页。[31] H. Tang、Z. Liu、X. Li、Y. Lin 和 S. Han，“Torchsparse：高效的点云推理引擎”，机器学习与系统会议论文集，第4卷，第302–315页，2022年。[32] W. Yuan、T. Khot、D. Held、C. Mertz 和 M. Hebert，“PCN：点云补全网络”，载于 IEEE计算机视觉与模式识别会议（CVPR），2018年，第774–783页。[33] J. Yang、H. Huang、Y. Zhou、X. Chen、Y. Xu、S. Yuan、H. Zou、C. X. Lu 和 L. Xie，“MM-Fi：用于多样化无线感知的多模态非侵入式四维人体数据集”，载于神经信息处理系统进展（NeurIPS），2024年。[34] X. Yu、Y. Rao、Z. Wang、Z. Liu、J. Lu 和 J. Zhou，“PoinTr：利用几何感知变换器实现多样化点云补全”，载于 IEEE/CVF国际计算机视觉会议（ICCV），2021年，第12 498–12 507页。