arXiv:2301.02562v1 [cs.CV] 5 Jan 2023

超稀疏三维目标检测

Lue Fan、Yuxue Yang、Feng Wang、Naiyan Wang 和 Zhaoxiang Zhang

摘要—随着激光雷达感知范围的扩大，基于激光雷达的三维目标检测对自动驾驶中的远距离感知发挥着越来越重要的作用。主流三维目标检测器通常会构建稠密特征图，其计算成本随感知范围呈二次增长，因此难以扩展到远距离设置。为实现高效的远距离检测，我们首先提出一种全稀疏目标检测器，称为 FSD。FSD 基于通用稀疏体素编码器和一种新颖的稀疏实例识别（SIR）模块构建。SIR 将点分组成实例，并进行高效的逐实例特征提取。逐实例分组规避了中心特征缺失的问题，而该问题阻碍了全稀疏架构的设计。为进一步发挥全稀疏特性的优势，我们利用时序信息消除数据冗余，并提出一种名为 FSD++的超稀疏检测器。FSD++ 首先生成残差点，用于表示连续帧之间的点变化。残差点与少量先前的前景点共同构成超稀疏输入数据，大幅减少了数据冗余和计算开销。我们在大规模 Waymo 开放数据集上对该方法进行了全面分析，并取得了当前最先进的性能。为展示该方法在远距离检测方面的优越性，我们还在 Argoverse 2 数据集上开展了实验，其感知范围（$200m$）远大于 Waymo 开放数据集（$75m$）。代码已开源，地址为 https://github.com/tusen-ai/SST。

索引词—三维目标检测、激光雷达、自动驾驶、稀疏、Waymo 开放数据集、实例分割、时序融合、点云聚类。

1 引言

自动UTONOMOUS 驾驶系统迫切需要高效的远距离感知，尤其是在高速场景中。当前基于激光雷达的三维目标检测器通常会将稀疏特征转换为稠密特征图，以进行后续特征提取和预测，我们将其称为**稠密检测器**。稠密检测器在当前流行的基准数据集上表现良好 [1], [2], [3], ，这些数据集的感知范围相对较短（不足 75 米）。然而，将稠密检测器扩展到远距离场景并不切实际（超过 200 米，图1）。在此类场景下，稠密特征图的计算和空间复杂度与感知范围的平方成正比。幸运的是，随着感知范围扩大，激光雷达点云的稀疏性也会增加（见图1），而对未占用区域进行计算实际上并无必要。鉴于其固有的稀疏性，实现高效远距离检测的关键方案是去除稠密特征图，并使网络架构全稀疏。

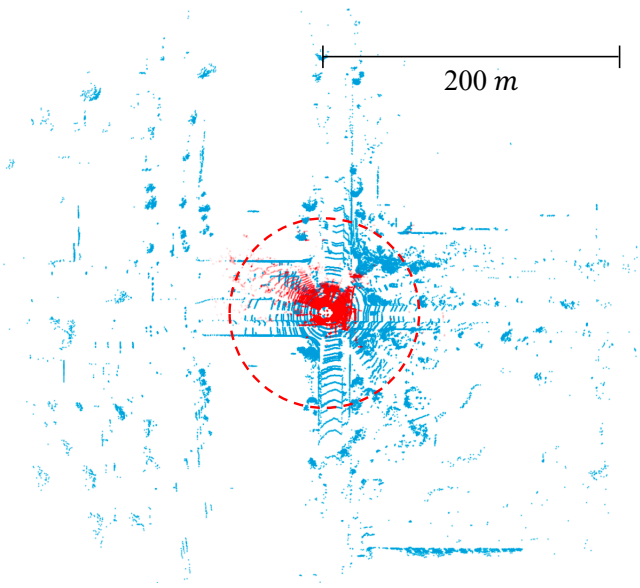


图 1. 近距离点云（红色，来自 KITTI [2]）与远距离点云（蓝色，来自 Argoverse 2 [4]）。红色圆圈的半径为 75 米。随着探测范围扩大，点云的稀疏程度迅速增加。

然而，移除稠密特征图并非易事，因为它在当前设计中发挥着不可或缺的作用。常用的稀疏体素编码器 [5], [6], [7] 仅提取非空体素上的特征。没有稠密特征图时，目标中心通常为空，尤其是对于大型物体。我们将这一问题称为**“中心特征缺失（CFM）”**（图 2）。CFM 会显著削弱中心体素的表征能力，在某些极端情况下（例如超大型车辆），甚至会使中心特征为空。然而，几乎所有流行的基于体素或柱体的检测器 [5], [6], [8], [9], [10] 都采用基于中心的方法分配，并依赖中心特征，因为它是对整个物体的理想表征。因此，他们必须先在稀疏体素编码器之后，将稀疏体素转换为鸟瞰图中的稠密特征图。随后，他们通过在稠密特征图上应用卷积，将特征扩散到实例中心，从而解决 CFM 问题；我们将此方法称为**特征扩散**（图 2）。

为了彻底消除稠密特征图，我们研究了纯基于点的检测器，因为它们天然是全稀疏的。然而，基于点的方法的应用受到两个缺点的限制。(1) 耗时的

• *Lue Fan*、*Yuxue Yang* 和 *Zhaoxiang Zhang* 均来自中国科学院（*CASIA*）自动化研究所智能感知与计算研究中心（*CRIPAC*）、模式识别国家重点实验室（*NLPR*），地址：中国北京市 *100190*。电子邮箱：{*fanlue2019*、*yangyuxue2023*、*zhaoxiang.zhang*}*@ia.ac.cn*。
• *Feng Wang* 和 *Naiyan Wang* 就职于图森未来，地址为中国北京 *100020*。电子邮箱：{*feng.wff, winsty*}*@gmail.com*。

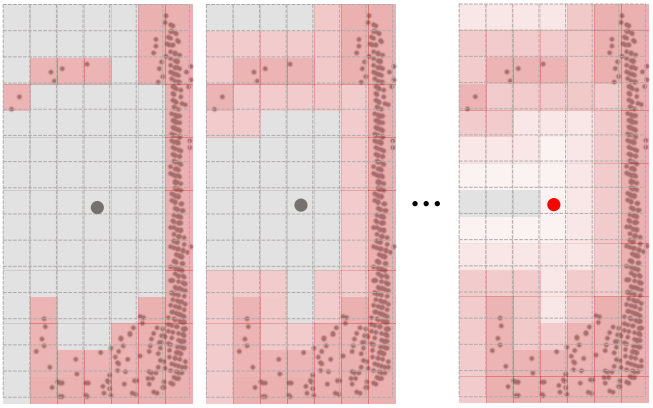


图2. 鸟瞰图稠密特征图上的**中心特征缺失**与**特征扩散**示意图。经过数次卷积后，空的实例中心（红点）由从被占用体素（含有激光雷达点）扩散而来的特征填充。

邻域查询 [11] 是将其应用于大规模点云（超过 100K 个点）时长期存在的难题。(2) 为了降低计算开销，基于点的方法会对整个场景进行激进下采样，将点数减少到固定数量。激进下采样不可避免地导致信息损失，并使前景物体的召回率不足 [12], [13], ，尤其是较小的物体。因此，在近期使用大规模点云的基准数据集中，纯基于点的检测器中只有极少数达到了最先进的性能。

本文首先提出**全稀疏检测器（FSD）**，以避开中心特征缺失问题。FSD 基于通用稀疏体素编码器 [5], [6], [7]构建，用于体素/点特征提取。随后，FSD 将点分组成实例，进一步提取实例级特征，并通过一种新颖的**稀疏实例识别（SIR）**模块，从整合后的实例特征中预测单个边界框。通过这种方式，预测基于整个实例特征，而非较弱或缺失的中心特征。作为一种基于点的模块，SIR 具有若干理想特性：（1）与以往的基于点的模块不同，SIR 仅将实例视为组，不会通过耗时的邻域查询进行进一步分组。（2）与动态体素化 [14],类似，SIR 利用动态广播/池化进行张量操作，从而避免点采样或填充。（3）由于 SIR 中的组覆盖整个实例，因此无论实例的物理尺寸如何，都能构建足够大的感受野。

为充分释放 FSD 的潜力，我们进一步利用时序信息，提出一种**超稀疏三维物体检测器**，称为 FSD++。FSD++ 的设计灵感来源于人类的视觉行为：人类对物理世界中的动态部分较为敏感，并会将注意力集中于这些部分。具体而言，FSD ++ 利用自运动移除包含大量时间冗余的静态部分，同时仅保留具有信息量的动态部分。我们将检测到的动态部分称为残差点，因为这一过程类似于计算帧之间的差分。通过这种方式，我们构建了一个由残差点和少量来自历史预测的历史前景点组成的超稀疏点云。随后，FSD++ 以该超稀疏点云作为输入，实现了一个非常高效的结合时序融合的检测框架。高效率得益于全稀疏特性与超稀疏输入之间的协同作用。我们的贡献总结如下。

- 我们提出了全稀疏检测器（FSD）的概念，这是实现高效远距离激光雷达检测的关键方案。我们进一步提出稀疏实例识别（SIR），以绕开稀疏特征图中的中心特征缺失（CFM）问题。将 SIR 与通用稀疏体素编码器相结合，我们开发出一种高效且有效的 FSD 实现。

- 基于 FSD，我们进一步提出 FSD++ 框架，该框架聚合多帧中的超稀疏点云作为输入，同时消除点云的时间冗余。所提出的框架揭示了稀疏架构尚未开发的潜力。我们希望我们的工作能引起社区对全稀疏架构的关注。

- FSD 在竞争激烈的 Waymo 开放数据集上取得了最先进的性能。此外，我们进一步将该方法应用于近期发布的 Argoverse 2 数据集，以展示 FSD 在远距离检测方面的优势；与稠密方法相比，FSD 的效率高得多。与单帧输入相比，FSD++仅需极少的额外开销，便能达到与主流最先进多帧检测器相当的性能。

2 相关工作

回顾基于激光雷达的三维目标检测器的发展历程，先前的方法可根据其空间稀疏性分为三类：稠密检测器、稀疏检测器和半稠密检测器。下面，我们根据空间稀疏性简要回顾以往的研究。

2.1 基于体素的稠密检测器

开创性工作 3DFCN [15] 和 VoxelNet [16] 采用稠密卷积进行体素特征提取。它们将卷积神经网络引入基于激光雷达的三维目标检测领域，并在当时取得了有竞争力的结果。然而，将稠密卷积应用于三维体素表示的效率较低。MV3D [17], 、PIXOR [18],和 PointPillars [19] 采用鸟瞰图（BEV）特征图上的二维稠密卷积，显著提升了效率。我们将这类检测器称为**稠密检测器**，因为它们会将稀疏点云转换为稠密特征图。

2.2 基于点的稀疏检测器

自从 PointNet [20] 和 PointNet++ [11] 为三维点集的深度学习研究带来启发后，一系列基于点的检测器便相继涌现。这些纯粹基于点的检测器天生就是全稀疏的。PointRCNN [21]是这一研究方向的开创性工作。3DSSD [12] 通过移除特征传播层和细化模块，加速了基于点的方法。VoteNet [22] 首先进行中心投票，再根据投票得到的中心生成候选区域，从而取得更高的准确率。尽管许多方法 [12], [13],[23] 都尝试加速基于点的方法，

在大规模点云（每个场景超过 10 万个点）中，耗时的点采样和邻域查询仍然难以承受。因此，当前的基准数据集 [1], [3] 中，大规模点云仍由基于体素的稠密／半稠密检测器主导 [10], [24], [25]。

2.3 半稠密检测器

与稠密检测器不同，**半稠密检测器**同时采用稀疏特征和稠密特征。SECOND [5]采用稀疏卷积提取三维空间中的稀疏体素特征，随后将其转换为鸟瞰图中的稠密特征图，以扩大感受野并与二维检测头集成 [26], [27], [28]。基于 SECOND 风格的半稠密检测器，一系列工作 [29],[30], [31] 进一步改进了单阶段范式。其他方法则增加了第二阶段，以进行细粒度特征提取和候选框细化 [7],[8], [10], [32]，从而取得更优的性能。尽管半稠密检测器已在学术界和工业界占据主导地位，但相关研究在此陷入停滞，因为正如我们在第1节所讨论的，半稠密检测器无法轻易转变为全稀疏检测器。

3 FSD：全稀疏 3D 目标检测

3.1 总体架构

遵循将实例视为组这一动机，我们通过四个步骤构建全稀疏检测器（FSD）：1）首先，我们使用稀疏体素编码器 [5], [6], [7] 提取体素特征，并对目标中心进行投票（第3.2节）。2）实例点分组根据投票结果，将前景点分组成实例（第 3.2节）。3）根据分组结果，稀疏实例识别（SIR）模块提取实例/点特征并生成候选区域（第 3.3节）。4）利用这些候选区域修正点分组，并迭代地优化候选区域（第 3.4节）。

3.2 实例点分组

分类与投票 首先，我们使用稀疏体素编码器从点云中提取体素特征，例如 SST 中的稀疏注意力块 [6]或稀疏卷积编码器。随后，我们将体素特征与点到其对应体素中心的偏移量拼接，构建点特征。这些点特征被输入两个分支，用于前景分类和中心投票。投票方式与 VoteNet 类似 [22]，，模型会预测前景点到对应目标中心的偏移量。我们采用 L1损失 [27] 和焦点损失 [33] 作为投票损失 L_{vote}和语义分类损失 L_{sem}。

连通分量标记（CCL）为了将点分组为实例，我们将所有预测的中心（图 3 中的红点）视为图中的顶点。如果两个顶点之间的距离小于某个阈值，则将它们连接起来。这样，该图中的一个连通分量就可以视为一个实例，所有投票到该连通分量的点共享一个组 ID。与 VoteNet 中的球查询不同，我们基于 CCL 的分组方法在大多数情况下能够避免实例碎片化。尽管存在许多精心设计的实例分组方法 [34], [35], [36]，，但我们选择简单的 CCL，因为它已足以满足我们的设计需求，并且可以通过高效的并查集算法 [37]并行实现。

3.3 稀疏实例识别

3.3.1 预备知识：动态广播/池化

给定属于 M 个组的 N 个点，我们将其对应的组 ID 数组定义为 I，其形状为 [N,]；将其特征数组定义为 F，其形状为 [N,C],，其中 C 为特征维度。$F^{(i)}$ 是属于第 i 个组的点的特征数组。动态池化将每个 F（i）聚合为一个组特征 $\boldsymbol{g}_i$，其形状为 [C,]。因此，我们有 $\boldsymbol{g}_i = p\left(F^{(i)}\right)$，其中 p 是一个对称池化函数。对所有组特征 G 进行动态池化，其形状为 [M,C]，可表示为 $G = p\ (F,\ I)$。动态广播可以视为动态池化的逆操作，它将 $\boldsymbol{g}_i$ 广播到第 i 个组中的所有点。由于广播本质上是一种索引操作，我们使用索引表示法 [] 将其表示为 $G[I]$，其形状为 [N,C]。动态广播/池化的效率非常高，因为它可以在现代设备上以高度并行的方式实现，并且非常适合具有动态大小的稀疏数据。我们在附录 A 中提供了高效的实现和运行时间评估。

动态广播/池化的前提是每个点唯一地 属于一个组，即各组之间不应重叠。由于真实三维世界中的实例彼此不重叠，因此各组自然也不会相互重叠。

3.3.2 稀疏实例识别的公式化

在第3.2节将点分组成实例后，我们可以直接通过一些基础的基于点的网络提取实例特征，例如 PointNet、DGCNN 等。定义一个基础的基于点的模块包含三个要素：组中心、成对特征和组特征聚合。

组中心组中心是一个组的代表点。例如，在球查询中，它是球体的局部原点。在 SIR 中，组中心定义为组内所有投票中心的质心。

成对特征 定义了如何配对组中心和邻域点，以提取组感知的邻域点特征。SIR采用两种成对特征：1）组中心与每个点之间的相对坐标；2）组特征与每个点特征的拼接。以特征拼接为例，并使用3.3.1中的符号，成对特征可表示为 $\mathrm{CAT}(F, G[I])$，其中 CAT 表示通道拼接。

组特征聚合 在一个组中，使用池化函数聚合邻域特征。SIR 采用动态池化来聚合特征数组 F。根据3.3.1中的符号，我们有 $G = p(F, I)$，其中 G 是聚合后的组特征。

整合 结合这三个基本元素，我们可以构建出许多基于点的算子变体，例如 Point- Net [20],DGCNN [38], Meta-Kernel [39], 等。图 4 阐释了如何利用动态广播/池化构建实例级点算子的基本思路。在我们的设计中，我们采用 VFE [16] 作为 SIR 层的基本结构，它本质上是一个两层的 PointNet。在 SIR 模块的第 l层中，给定输入的逐点特征数组

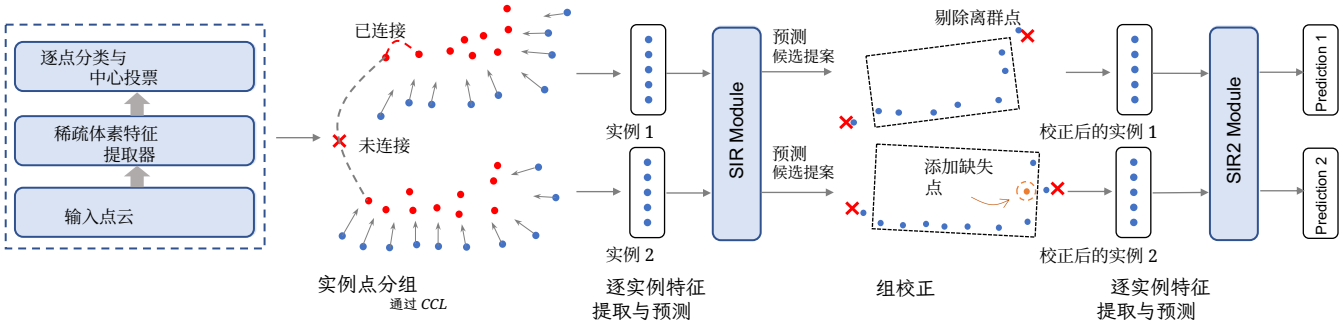


图3. FSD的总体架构。为简单起见，我们仅使用两个实例来说明这一流程。红色点是由每个激光雷达点投票得到的中心（蓝色点）。SIR模块和SIR2模块均包含3个SIR层。

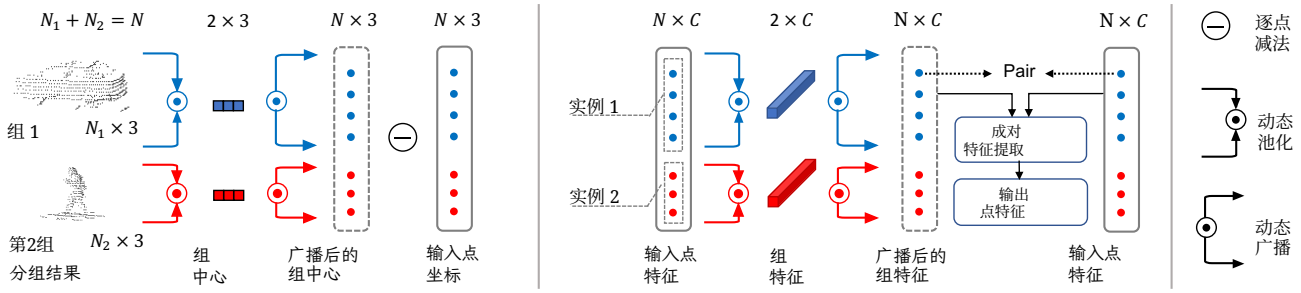


图4. 使用动态广播/池化构建实例级点算子的示意图。建议彩色查看。左：根据原始点云计算中心到邻居的偏移量。右：更新点特征。请注意，该操作在所有实例间并行进行。

F_l、点坐标数组 X、投票得到的中心 X' 和组 ID 数组 I，第 l 层的输出可表示为：

$$F'_l = \texttt{LinNormAct}\left(\texttt{CAT}\left(F_l, X - p_{\text{avg}}(X', I)[I]\right)\right), \quad (1)$$

$$F_{l+1} = \texttt{LinNormAct}\left(\texttt{CAT}\left(F'_l, p_{\max}(F'_l, I)[I]\right)\right), \quad (2)$$

其中，LinNormAct 是一个全连接层，后接一个归一化层 [40] 和一个激活函数 [41]。p_{avg} 和 $p_{\max}$ 分别是平均池化和最大池化函数。输出 F_{l+1} 还可作为下一个SIR层的输入，因此我们的SIR模块由若干个基本SIR层堆叠而成。

3.3.3 稀疏预测

根据公式 1 和公式 2，SIR 并行动态地提取所有实例的特征。随后，SIR 对所有组进行**稀疏预测**。与两阶段稀疏预测不同，我们的候选区域（即组）彼此不重叠。与单阶段密集预测不同，我们只对每个组生成一次预测，这显著降低了预测头的开销。值得注意的是，全稀疏架构可能面临严重的不平衡问题：近距离物体包含的点远多于远距离目标。一些方法 [39], [42] 使用手工设计的归一化因子来缓解不平衡问题。相比之下，SIR 不会产生不平衡问题，因为无论组内点的数量多少，它都只对每个组生成一次预测。在大多数情况下，一个组仅对应一个真实框。

具体而言，对于每个 SIR 层，都有一个 $G_l = p_{max}(F'_l,\ I)$，如公式 2 所示，可将其视为组特征。我们沿通道维度拼接每个 SIR 层中的所有 G_l，并使用拼接后的组特征，通过多层感知机预测边界框和类别标签。所有中心落在真实标注框内的组都是正样本。对于正样本，回归分支预测从组中心到真实标注框中心的偏移量，以及物体尺寸和方向。采用 L1损失 [27] 和焦点损失 [33] 分别作为回归损失 L_{reg} 和分类损失 L_{cls}。

3.4 组校正

实例点分组模块中不可避免地会出现错误分组。例如，某些前景点可能会被遗漏，或者某些组可能受到背景杂波的污染。因此，我们利用 SIR 生成的边界框提议来校正分组。候选提案内的点无论先前的组ID是什么，都属于校正后的组。由于少数点可能落入多个候选提案，我们只需复制这些点及其特征，并将不同的副本分配给不同的候选提案。校正后，我们对这些新组额外应用一个 SIR。为与第一个 SIR 模块区分开来，我们将这个额外的 SIR 模块记作 SIR2。

SIR2 按照许多两阶段检测器的做法，预测候选提案到其对应真实标注框的边界框残差。为了让 SIR2 感知候选提案的大小和位置，我们参照 [43]，将内部点到提议边界的偏移量作为额外的点特征。回归损失记为 $L_{res} = L1(\Delta_{res}, \Delta_{res})$，其中 Δ_{res} 是 $\widehat{\quad}$ 真实残差，Δ_{res} 是预测残差。按照以往方法，[7], [8], 候选提案与真实标注之间的三维交并比（IoU）用作 SIR2 中的软分类标签。具体而言，软标签q定义为 $q = \min(1, \max(0, 2IoU - 0.5))$，其中 IoU 是

候选区域与对应真实标注之间的交并比（IoU）面积。随后采用交叉熵损失训练分类分支，记为 L_{iou}。综合考虑分组（第 3.2 节）和稀疏预测中的所有损失函数，可得

$$L_{total} = L_{sem} + L_{vote} + L_{reg} + L_{cls} + L_{res} + L_{iou}, \quad (3)$$

为简洁起见，我们省略了每一项的权重。

3.5 讨论

FSD中的中心投票受到VoteNet的启发 [22], ，但FSD与VoteNet有两个本质区别。

• 投票后，VoteNet仅对投票中心周围的特征进行聚合，不再进一步提取特征。FSD则更进一步，利用动态广播/池化构建了一个高效的SIR模块，从而能够进一步提取实例级特征。因此，FSD提取的实例特征更具表现力，这一点已在第5.5节通过实验验证。

• VoteNet是一种典型的基于点的方法。正如我们在第1节所述，它会大幅下采样整个场景，将点数缩减至固定数量以提高效率，但这不可避免地会造成信息损失。相比之下，SIR的动态特性和高效性使其能够从任意数量的输入点中提取精细的点特征，而无需进行任何下采样。在第5.5节中，我们展示了该设计在处理大规模点云时的效率，以及精细点表示的优势。

4 FSD++：采用超稀疏输入的 FSD

众所周知，将多帧聚合为输入有助于提升性能。然而，朴素的聚合可能会导致点云变得更稠密，从而显著拖慢算法，尤其是在采用稀疏运算的架构中。因此，我们尝试从原始点云流中移除时间冗余，以获得更稀疏的输入数据。得益于全稀疏特性，移除冗余后稀疏度的提升能够显著改善全稀疏模型的性能。因此，一个自然的问题随之而来：如何在预先保留有信息量的部分的同时移除冗余？连续帧点云之间的相似性为解决这一问题提供了一种可能的方案。

具体而言，点的空间分布在一个序列中连续且平滑地变化。我们将连续帧之间发生变化的点称为残差点。残差点包含丰富信息，因为它们代表了某个时间步中的新观测。结合残差点与历史预测，检测器便掌握了足够的信息来推断当前物体。在这一范式下，残差点与先前的前景点共同构成超稀疏点云。FSD 可以直接将其作为输入，从而更高效地进行目标检测。

4.1 残余点探测

激光雷达传感器在每个时间步都会捕获大量新观测到的前景点。这些点主要来自两个方面：（1）物体移动到新的位置；（2）遮挡区域变得可见。这些新观测到的点称为残差点。残差点对于定位运动物体和检测最近出现的物体至关重要。正如我们之前提到的，可以通过点的空间分布变化来检测残差点。

残余点检测算法必须满足两个关键要求。(1) 算法应能抵御点的微小扰动，这些扰动可能由传感噪声或微小的自运动估计误差引起。此类点扰动不应被检测为残差点。(2) 算法应具备很高的效率，以处理来自多帧的数百万个点。具体而言，WOD 中的每帧最多包含 200,000 个点。

有几种直接的方法可以满足第一个要求，即球查询或将数据体素化为稠密占用图。如果球查询定义的邻域内没有落入任何先前帧中的点，则可将该点视为残差点。也可以通过对两个稠密占用图进行简单差分来检测残差点。尽管合适的球查询半径或体素大小能够抵御点扰动，但这些方法仍存在计算复杂度较高（$O(N^2)$）或内存占用巨大的问题。

为满足上述两项需求，我们采用哈希方法，并设计了算法，如算法1所示，命名为残余点探测（*RPP*）。RPP 包含两个步骤。(1) 首先将点坐标量化为整数。量化粒度决定了算法对点扰动的鲁棒性。(2) 对于每个点，RPP 通过哈希探测来验证它是否为残差点。具体而言，RPP 首先根据先前量化的点构建一个哈希表。哈希表的键集合记为 $K \subset \mathbb{Z}^3$，即量化后的整数坐标；值集合记为 $V = \{1, 0\}$，其中1 表示槽已被占用，0 表示槽未被占用。随后，RPP 使用当前量化坐标探测哈希表。如果当前点命中未被占用的槽，则将其视为残差点。RPP 隐含一个假设：如果两个点量化后位于同一体素中，我们就认为它们是同一个点。我们采用广为人知的开放定址法进行探测，并采用双重哈希作为哈希函数，以减少哈希冲突。

算法 1： 高效残余点探测

输入： 当前点 P_{cur}，上一帧点 P_{pre}，
体素大小 s，负载因子 α，哈希函数 h
输出： 当前帧的残差点 ΔP_{cur}

$\widehat{P}_{cur} \leftarrow$ 量化(P_{cur}, s);
$\widehat{P}_{pre} \leftarrow$ 量化(P_{pre}, s);
初始化长度为 $|\widehat{P}_{pre}|/\alpha$的空哈希表 T;
初始化空的残差点集合 ΔP_{cur};
对于 $\widehat{P}_{pre}$ 中的每个 $\widehat{p}_i$ **执行**
  $slot_i \leftarrow$ 探测($T, h(\widehat{p}_i)$);
  如果 $slot_i$ 未被占用**则**
    $slot_i \leftarrow$占用标志;
对每个 $\widehat{P}_{cur}$ 中的 $\widehat{p}_i$ **执行**
  $slot_i \leftarrow$ 探测(T, $h(\widehat{p}_i)$);
  如果 $slot_i$ 未被占用**则**
    将 p_i 添加到 ΔP_{cur};
返回 ΔP_{cur}

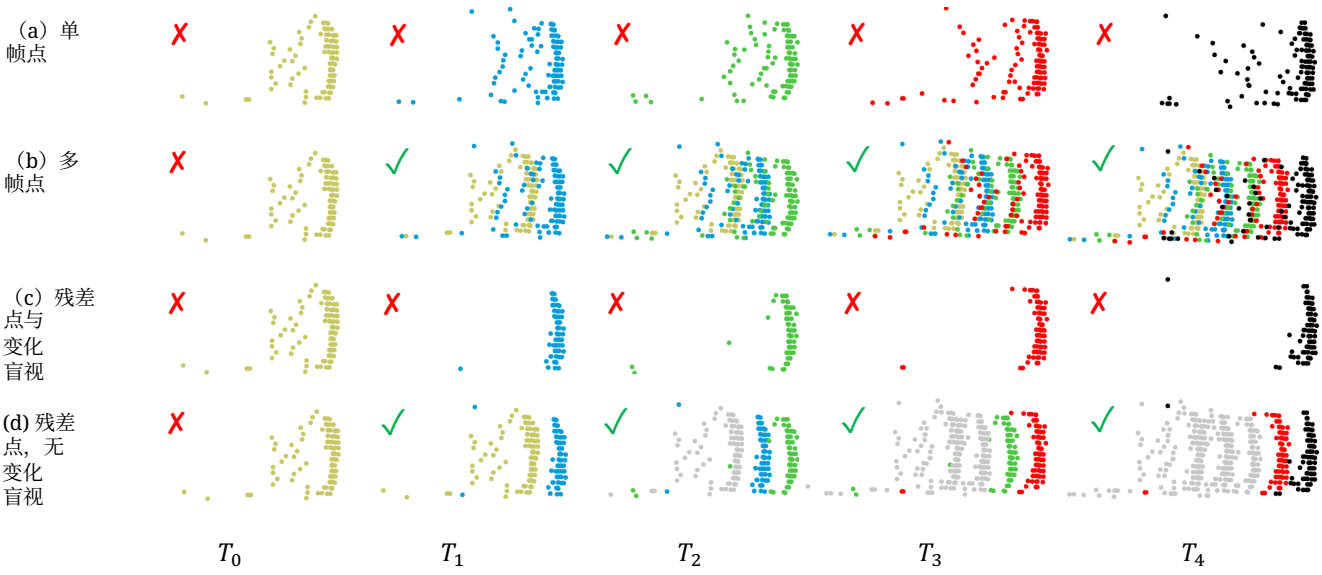


图5. 时序点操作示意图。不同颜色表示来自不同时间步的点。 灰色 点是采样得到的骨架点。 红色叉号 ✗ 表示观测太弱，无法识别为前景物体（实际上并非如此，仅为示意）。 绿色勾号 ✓ 表示观测足够强。(a) 单帧中的点太弱。(b) 经过多帧点聚合后，检测器从时间步 T_1生成真阳性。(c) 单帧残差点受到变化盲视的影响。由于检测器在 T_0 无法通过骨架点采样生成真阳性，因此在 T_1中仍然只有较弱的残差点。因此，检测器始终输出假阴性。(d) 检测器利用来自两帧的残差点（最大年龄为2），在 T_1 作出了正确预测。因此，在 T_2中，检测器可以利用 T_1 中的预测进行骨架点采样，从而缓解变化盲视。

残差点探测在内存和速度方面都很高效。例如，假设一个点云片段中有 N 个唯一的量化坐标。如果我们希望碰撞率低于 α，哈希表的长度应为 N/α。根据经验，在 WOD 的 5 帧点云中， N 约为 500,000。一个槽状态可以用单个位表示，并令 α 等于 0.1。由此可知，该哈希表的内存开销约为 0.6 MB。此外，每个点的探测彼此独立，因此能够在 GPU 上实现高度并行。

形式上，我们将 RPP 过程表示如下：

$$\Delta P_t = P_t - \bigcup_{i=1}^{B} P_{t-i}, \tag{4}$$

其中， ΔP_t 是时间步 t检测到的残差点， P_t 是时间步 t的原始点。符号“X –Y”表示从 X中移除 X 与 Y 的交集，等价于 $X \setminus (X \cap Y)$。并集表示点云拼接。 B 是 RPP 中使用的前序帧数，我们称之为基准帧。

4.2 骨架点采样

由于残差点仅包含当前时间步的新观测，检测器需要从先前帧获取额外信息，以获得充足的输入。为纳入这些历史数据，我们使用先前预测的边界框裁剪先前的前景点，同时丢弃边界框之外的其他点。经过自运动补偿后，将裁剪出的点放入当前帧。然而，来自多个先前帧的前景点本质上仍存在冗余。尤其是，多个帧中近距离物体上的大量点可能造成不必要的开销。

为了减少多帧前景点带来的冗余，我们进一步对这些裁剪出的点进行采样。直观地说，我们期望采样后的点包含模型做出恰当预测所需的最少信息。从这个意义上说，我们将裁剪出的点的这一最小子集称为骨架点，因为它们描绘了物体的基本结构或“骨架”。具体而言，我们尝试了三种采样方法：随机采样、最远点采样和体素采样。所有采样方法均在先前预测的边界框内进行。对于随机采样和最远点采样，我们采用预先定义的最大点数阈值 N_T。对于包含点数超过N_T 的边界框，我们在框内采样 N_T 个点。对于体素采样，我们采用动态体素化 [14]对点进行体素化。落入同一体素的所有点通过平均池化归并为一个点。

4.3 应对变化盲视

理论上，通过结合骨架点和残差点，模型能够对当前帧作出预测。然而，一种被称为“变化盲视”的现象可能会影响性能。变化盲视指的是人类视觉系统容易忽略场景中逐渐发生的细微变化，即使多个时间步累积起来的变化十分显著。我们的情况也可能出现类似问题。设想在时间步 t中，一辆车辆几乎进入激光雷达的感知范围，此时只能观测到车辆的一小部分。检测器很可能会将其识别为背景，因此 RPP 会在时间步 $t+1$ 移除这些点，只保留车辆的少量新点作为残差点。这样一来，如果车辆缓慢出现，检测器可能永远无法识别它。图5 展示了变化盲视现象。

为了解决变化盲视问题，我们为残差点引入最大保留时间M。换言之，检测器最多将来自 M 个时间步的残差点作为输入。形式上，在时间步 t，检测器将 $\bigcup_{i=0}^{M-1} \Delta P_{t-i}$ 作为累积残差点输入。

4.4 集成式 SuperSparseInput

输入点云由两部分组成：先前的骨架点，以及来自多个时间步的残差点。具体而言，对于一个 N 帧 FSD++ 检测器，时间步 t 的最终输入点如下：

$$P_t^{in} = \left(\bigcup_{i=1}^{N} P_{t-i}^{s} \right) \cup \left(\bigcup_{i=0}^{M-1} \Delta P_{t-i} \right), \tag{5}$$

其中，P_t^s表示时间步 t的骨架点。P_t^{in}比原始点云稀疏得多，并直接输入 FSD 检测器。图 7展示了 P^s、ΔP和 P^{in}的示例。

4.5 训练与推理流程

FSD++的训练与推理流程因使用历史预测和时序信息而有别于标准方法。图 6总结了整体流程。

4.5.1 训练

为了利用历史预测，输入点云流必须按照时间顺序排列。然而，由于缺少数据打乱，有序的输入流会影响模型训练。另一个问题是，训练初期的历史预测并不可靠。针对这两个问题，我们使用训练良好的 FSD 检测器为整个训练集生成离线预测。在训练过程中，对于每个样本，我们将其与此前的离线预测一同加载，以采样骨架点。真实框似乎可以作为离线预测的替代方案。然而，训练中使用的真实框与推理中使用的预测框之间存在较大的分布差异。因此，我们采用离线预测，而不是真实标注框。

4.5.2 推理

在在线推理阶段，输入点云流自然按照时间顺序排列。对于一个点云序列，其第一帧的预测来自训练良好的 FSD 检测器。这些预测被视为第一帧的“先前预测”，称为序列的 种子预测 。我们维护多个队列，用于缓存一些可以重复使用的历史数据。例如，在 N 帧的 FSD++ 流水线中，时间步 t 的原始点和骨架点可以从时间步 $t+1$ 重复使用到 $t+N-1$。

5 实验

5.1 设置

5.1.1 数据集

Waymo 开放数据集（WOD）在我们的实验中，我们使用 WOD [1] 作为主要数据集来评估我们所提出方法的性能。WOD 是基于激光雷达的三维目标检测领域最值得信赖的基准。WOD 拥有 1150 个序列和超过 200,000 帧，是目前同类数据集中规模最大的一个。其中，798 个序列用于训练，202 个用于验证，150 个用于测试。WOD 的检测范围为 75 米（覆盖区域为$150m \times 150m$）。

Argoverse 2 (AV2) 我们进一步在近期发布的 Argoverse 2 数据集上开展远距离实验 [4]，以展示 FSD 在远距离检测方面的优势。AV2 的规模与 WOD 相近，共包含 1000 个序列，其中 700 个用于训练、150 个用于验证、150 个用于测试。除平均精度（AP）外，AV2 还采用综合得分作为评估指标，同时考虑 AP 和定位误差。AV2 的感知范围为 200 米（覆盖区域为 $400m \times 400m$），远大于 WOD。如此大的感知范围会导致稠密检测器的内存占用极高。

5.1.2 模型配置

为展示 SIR 的通用性，我们构建了两个 FSD 变体。FSD$_{sst}$ 采用新兴的单步长稀疏 Transformer [6] 作为稀疏体素特征提取器。FSD^{spconv} 基于 PartA2 [7]中的稀疏卷积 U-Net 构建。在 FSD 实验中，除非另有说明，我们使用 FSD$_{sst}$。在 FSD++实验中，我们使用 FSD^{spconv} 作为检测器，因为 SpConv 高度优化的工程实现使其在多帧输入时比 SST 骨干网络更高效。

5.1.3 实现细节

我们的实现基于广泛使用的 MMDetection3D（v0.15） [44]。在 FSD$_{sst}$中，我们使用 4 个稀疏区域注意力块 [6] 作为体素特征提取器。SIR 模块和 SIR2 模块分别由 3 层和 6 层 SIR 层组成。SIR 层由公式 1 和公式 2定义。我们基于 SST 的模型收敛速度远快于 SST，因此消融研究中模型训练 6 个训练轮次，而非 SST 中的 2×计划（24 个训练轮次）。对于 FSD^{spconv}，除 6 轮次训练计划外，我们还采用更长的训练计划（12 轮次）以获得更好的性能。与 MMDetection3D 中的默认设置不同，我们减少了 CopyPaste 数据增强中粘贴实例的数量。在 FSD 中，一些稀有类别（如骑行者）容易因粘贴实例过多而过拟合。Argoverse 2 数据集上的所有实验均采用 12 轮次训练计划。用于性能分析的模型（第 5.3 ～ 节 5.6）使用 8 张 RTX 2080Ti GPU 训练，批量大小为 2。表 1 中的模型使用 8 张 RTX 3090 GPU 训练，批量大小为 2。更多细节请参见我们发布的代码。

5.2 FSD 和 FSD++ 的主要结果

我们首先在表1 和表 2中，将 FSD 与最先进的检测器及我们的基线进行比较。在验证集划分中，FSD/FSD++ 在单帧/多帧设置下分别取得了最先进的平均表现（L2 mAPH）。在测试集划分中，FSD 在所有单帧检测器中，各类别的表现均最佳。同时，就平均指标而言，采用 7帧输入的 FSD++超过了所有采用最多 100 帧输入的检测器。

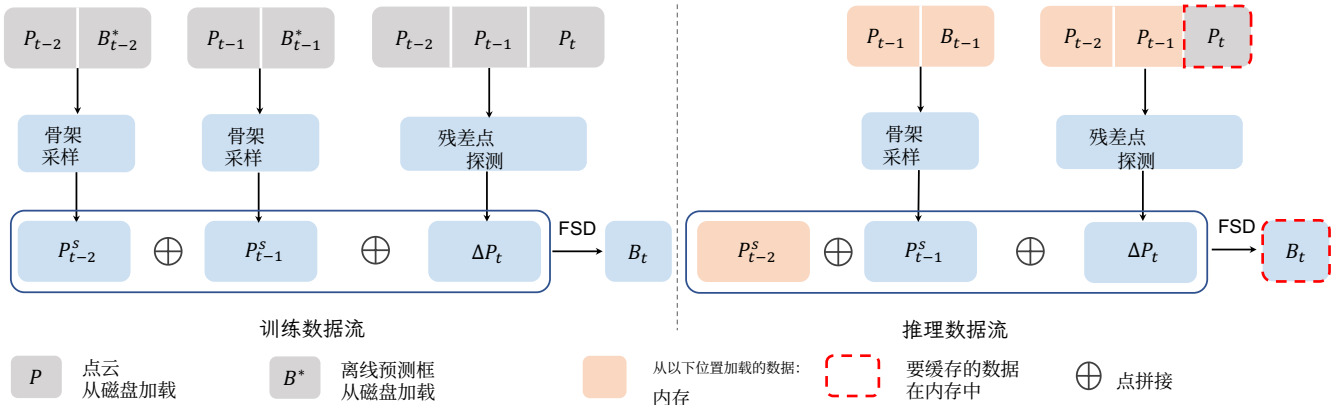
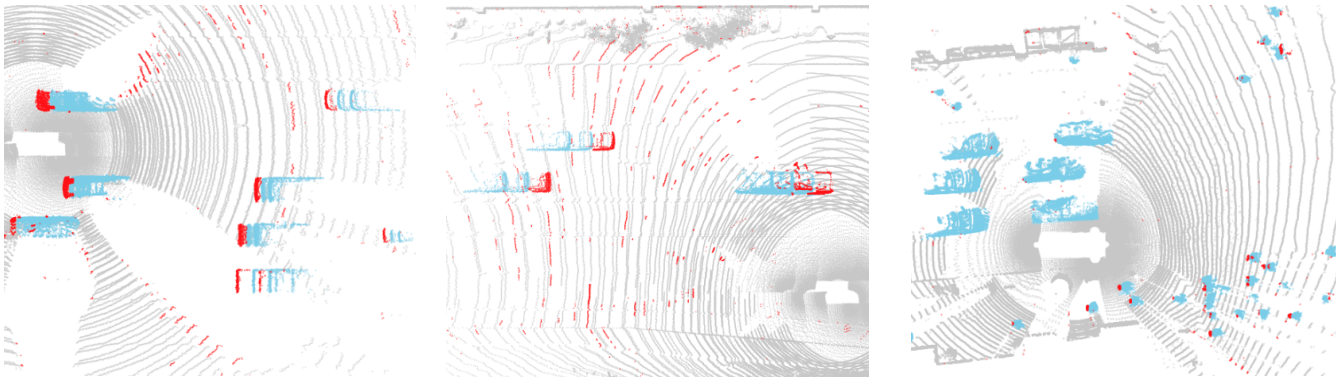


图6。FSD++的总体架构。训练时，我们采用离线预测来近似历史预测。推理时，检测器使用先前的在线预测进行骨架点采样。当最大年龄大于1时，还会有一些来自时间步$t-i$的其他 ΔP_{t-i}。为简单起见，我们这里只展示 ΔP_t。



（a）移动的汽车与静止的自车　（b）移动的汽车与移动的自车　（c）静止的汽车与移动的行人

图7。超稀疏输入点云示例。残差点以红色标示。先前的前景点以蓝色标示。灰色点不会被送入检测器。(a) 移动的汽车会产生明显的残差点。由于汽车移动，地平面上一些被遮挡的点也会显现出来。(b) 自车正在移动，导致地平面上的一些点被检测为残差点。(c) 残差点检测在移动的行人上，而非静止的汽车上。

还值得注意的是，FSD 和 FSD++ 的效率远高于以往大多数方法，尤其是在多帧设置和远距离设置下。我们将在第5.4节和第5.7节对此进行详细说明。

5.3 中心特征缺失的处理研究

5.3.1 定量实验

在下文中，我们在 WOD 上开展实验，以研究**中心特征缺失（CFM）**问题。首先，我们构建了若干具有不同特性的模型。需要注意的是，以下所有模型均采用相同的体素化分辨率，因此它们在初始阶段面临相同程度的中心特征缺失。

- **FSD**^{plain}：在稀疏体素编码器之后，FSD^{plain} 直接根据每个体素预测边界框。真实框内部的体素被分配为正样本。尽管 FSD^{plain} 采用了处理中心特征缺失最直接的方案，但它受到回归目标方差较大以及信息丰富体素产生的低质量预测的影响。
- **SST**$_{center}$：它将 SST 中基于锚框的检测头替换为 CenterHead [9], [28]。基于稀疏体素编码器，SST$_{center}$ 将稀疏体素转换为稠密特征图，并应用若干卷积，将特征扩散至空的目标中心，如图 2所示。随后，它根据扩散后的中心特征进行预测。
- **FSD**^{nogc}：它移除了 FSD 中的组校正和 SIR2 模块。
- **CenterPoint-PP**：它不采用任何稀疏体素编码器，而是在体素化后不久应用多个稠密卷积进行特征扩散，大幅消除化 CFM。它还配备了 CenterHead，以避免回归目标出现较大方差。

大型车辆的中心周围通常有相当大的未占用区域。因此，大型车辆的性能是衡量 CFM 效果的合适指标。于是，我们构建了一个定制评估工具，按照 COCO 评估对目标长度进行细分 [59]。接着，我们使用该工具评估不同长度车辆的性能。表 3展示了结果，我们将发现总结如下。

- 与 SST^{plain} 相比，FSD$_{center}$使用了相同的基于注意力的稀疏体素编码器。然而，就车辆尺寸而言，两者的趋势完全相反。采用特征扩散后，

方法	#. 帧	平均精度均值/平均精度航向均值 L2	车辆 3D AP/APH L1	L2	行人3D AP/APH L1	L2	骑行者 3D AP/APH L1	L2
SECOND [5]	1	61.0/57.2	72.3/71.7	63.9/63.3	68.7/58.2	60.7/51.3	60.6/59.3	58.3/57.0
MVF [14]	1	-/-	62.9/-	-/-	65.3/-	-/-	-/-	-/-
AFDet [45]	1	-/-	63.7/-	-/-	-/-	-/-	-/-	-/-
Pillar-OD [46]	1	-/-	69.8/-	-/-	72.5/-	-/-	-/-	-/-
RangeDet [39]	1	65.0/63.2	72.9/72.3	64.0/63.6	75.9/71.9	67.6/63.9	65.7/64.4	63.3/62.1
PointPillars [19]	1	62.8/57.8	72.1/71.5	63.6/63.1	70.6/56.7	62.8/50.3	64.4/62.3	61.9/59.9
Voxel RCNN [32]	1	-/-	75.6/-	66.6/-	-/-	-/-	-/-	-/-
RCD [42]	1	-/-	69.0/68.5	-/-	-/-	-/-	-/-	-/-
VoTr-TSD [47]	1	-/-	74.9/74.3	65.9/65.3	-/-	-/-	-/-	-/-
LiDAR-RCNN [43]	1	65.8/61.3	76.0/75.5	68.3/67.9	71.2/58.7	63.1/51.7	68.6/66.9	66.1/64.4
金字塔 RCNN [48]	1	-/-	76.3/75.7	67.2/66.7	-/-	-/-	-/-	-/-
体素到点 [49]	1	-/-	77.2/-	69.8/-	-/-	-/-	-/-	-/-
3D-MAN [50]	16	-/-	74.5/74.0	67.6/67.1	71.7/67.7	62.6/59.0	-/-	-/-
M3DETR [51]	1	61.8/58.7	75.7/75.1	66.6/66.0	65.0/56.4	56.0/48.4	65.4/64.2	62.7/61.5
Part-A2-Net [7]	1	66.9/63.8	77.1/76.5	68.5/68.0	75.2/66.9	66.2/58.6	68.6/67.4	66.1/64.9
CenterPoint-Pillar [9]	1	-/-	76.1/75.5	68.0/67.5	76.1/65.1	68.1/57.9	-/-	-/-
CenterPoint-Voxel [9]	1	69.8/67.6	76.6/76.0	68.9/68.4	79.0/73.4	71.0/65.8	72.1/71.0	69.5/68.5
IA-SSD [13]	1	62.3/58.1	70.5/69.7	61.6/61.0	69.4/58.5	60.3/50.7	67.7/65.3	65.0/62.7
PV-RCNN [8]	1	66.8/63.3	77.5/76.9	69.0/68.4	75.0/65.6	66.0/57.6	67.8/66.4	65.4/64.0
RSN [52]	1	-/-	75.1/74.6	66.0/65.5	77.8/72.7	68.3/63.7	-/-	-/-
SST_TS [6]	1	-/-	76.2/75.8	68.0/67.6	81.4/74.0	72.8/65.9	-/-	-/-
SST [6]	1	67.8/64.6	74.2/73.8	65.5/65.1	78.7/69.6	70.0/61.7	70.7/69.6	68.0/66.9
AFDetV2 [24]	1	71.0/68.8	77.6/77.1	69.7/69.2	80.2/74.6	72.2/67.0	73.7/72.7	71.0/70.1
PillarNet-34 [53]	1	71.0/68.5	79.1/78.6	70.9 / 7	80.6/74.0	72.3/66.2	72.3/71.2	69.7/68.7
PV-RCNN++ [10]	1	68.4/64.9	78.8/78.2	70.3/69.7	76.7/67.2	68.5/59.7	69.0/67.6	66.5/65.2
PV-RCNN++（中心） [10]	1	71.7/69.5	79.3 / 7	70.6/70.2	81.3/76.3	73.2/68.0	73.7/72.7	71.2/70.2
CenterFormer [54]	8	75.1/73.7	78.8/78.3	74.3/73.8	82.1/79.3	77.8/75.0	75.2/74.4	73.2/72.3
INT [55]	10	-/73.6	-/-	-/73.3	-/-	-/71.9	-/-	-/75.6
MPPNet [56]	16	75.6/74.9	82.7 / 8	75.4 / 7	84.7/82.3	77.4/75.1	77.3/76.7	75.1/74.5
FSD$_{spconv}$ （我们的方法）	1	71.9/69.7	77.8/77.3	68.9/68.5	81.9/76.4	73.2/68.0	76.5/75.2	73.8/72.5
FSD$_{sst}$ （我们的方法）	1	71.5/69.2	76.8/76.3	67.9/67.5	81.3/75.3	72.5/67.0	77.2/76.0	74.4/73.2
FSD$_{spconv}$ (我们的方法)†	1	72.9 / 7	79.2/78.8	70.5/70.1	82.6 / 7	73.9 / 6	77.1 / 7	74.4 / 7
FSD++ （我们的方法）†	7	76.8 / 7	81.4/80.9	73.3/72.9	85.1 / 8	78.2 / 7	81.2 / 8	78.9 / 7

表 1 Waymo 开放数据集验证集划分上的性能。所有报告结果均来自未使用任何测试时增强的单模型。†：更长的训练计划（12 个训练轮次）。我们分别用灰色框和青色框标出最佳单帧结果和多帧结果。

在大型车辆上的表现远逊于 FSD^{plain}。这表明，对于大型物体而言，特征扩散并非 CFM 的理想方案。对于这些大型物体，特征可能无法扩散到中心，或者扩散后的特征过弱，无法进行准确预测。

- 然而，在常规尺寸车辆上，FSD^{plain} 的性能在所有检测器中最差。需要注意的是，CFM 问题对常规尺寸车辆的影响较小。因此，在这种情况下，SST$_{center}$ 中的基于中心的分配优于 FSD^{plain}中的分配。这表明，即使 FSD^{plain} 在大型物体上取得了更好的性能，其解决 CFM 的方案仍非最优。

- 对比 FSD^{nogc} 与 SST$_{center}$，两者采用相同的稀疏体素编码器，而 FSD^{nogc} 用 SIR 替换了 SST$_{center}$ 中的稠密部分。FSD^{nogc} 在大型车辆上取得的显著提升，充分表明 SIR 能有效解决 CFM，且优于特征扩散。
- CenterPoint-PP 从 CFM 中受到的影响要小得多，因为它从网络一开始就利用稠密特征图。此外，它还采用了先进的基于中心的分配策略。即便如此，FSD^{nogc} 和 FSD 仍优于 CenterPoint-PP，尤其是在大型车辆上。

5.3.2 定性分析

除了定量实验外，我们还展示了 CFM 及其处理方法的定性效果，如图 8所示。图 8展示了 FSD 的投票中心以及

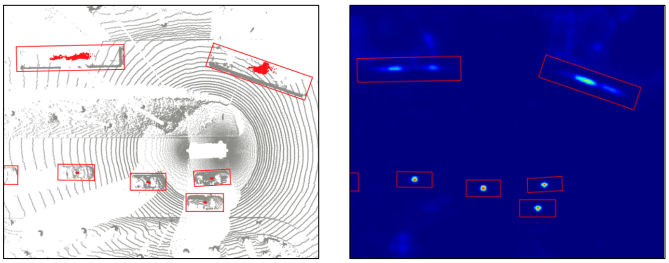


图 8. 中心特征缺失的直观示意图。**左侧**：FSD 的投票中心。**右侧：** SST$_{center}$ 的预测热力图。

基于中心的 SST 的预测热力图。对于正常尺寸的车辆，它们都能产生高质量的预测，但对于大型车辆，它们的预测（投票）通常存在歧义。基于中心的稠密检测器根据此类存在歧义的热力图进行预测，因此容易产生有缺陷的最终预测。尽管 FSD 对大型车辆的中心投票同样表现平平，但 FSD 仅利用这些投票来获取点组（即实例分割），这并不

方法	#. 帧	平均精度均值/平均精度航向均值 L2	车辆3D AP/APH L1	车辆3D AP/APH L2	行人3D AP/APH L1	行人3D AP/APH L2	骑行者3D AP/APH L1	骑行者3D AP/APH L2
CenterPoint [9]	1	-/69.0	-/-	-/71.9	-/-	-/67.0	-/-	-/68.2
AFDetV2-lite [24]	1	72.2/70.0	80.5/80.0	73.0/72.6	79.8/74.3	73.7/68.6	72.4/71.2	69.8/69.7
PV-RCNN [8]	1	71.3/68.8	80.6/80.1	72.8/72.4	78.2/72.0	71.8/66.0	71.8/70.4	69.1/67.8
PV-RCNN++ [10]	1	72.4/70.2	81.6/81.2	73.9/73.5	80.4/75.0	74.1/69.0	71.9/70.8	69.3/68.2
Graph R-CNN [57]	1	73.8/71.6	83.6 / 8	76.0 / 7	81.9/76.5	75.6/70.5	72.5/71.3	69.8/68.7
AFDetV2 [24]	2	74.6/73.1	81.7/81.2	74.3/73.9	81.3/78.0	75.5/72.4	76.4/75.4	74.1/73.0
CenterPoint++ [9]	3	74.2/72.8	82.8/82.3	75.5/75.1	81.0/78.2	75.1/72.4	74.4/73.3	72.0/71.0
BEVFusion* [58]	3	77.7/76.3	85.0 / 8	77.9/77.5	84.7 / 8	79.1/76.4	78.5/77.5	76.0/75.1
DeepFusion* [25]	5	76.9/75.5	83.3/82.8	76.1/75.6	84.6/81.8	79.2/76.4	77.8/76.8	75.5/74.5
CenterFormer [54]	16	76.9/75.6	84.7/84.4	78.1 / 7	84.6/81.8	79.4 / 7	75.5/74.5	73.3/72.4
INT [55]	100	76.6/75.2	84.7/84.3	78.0/77.6	82.4/79.7	76.6/74.0	77.4/76.3	75.2/74.1
MPPNet [56]	16	76.9/75.7	84.3/83.9	77.3/76.9	84.1/81.5	78.4/75.9	77.1/76.4	74.9/74.2
FSD$_{spconv}$ （我们的方法）†	1	74.4 / 7	82.7/82.3	74.4/74.1	82.9 / 7	75.9 / 7	75.6 / 7	72.9 / 7
FSD++ （我们的方法）†	7	78.4 / 7	84.5/84.1	77.1/76.7	84.5/81.7	79.0/76.2	81.4 / 8	79.2 / 7

表 2 Waymo 开放数据集测试集划分上的性能。所有结果均采用单模型设置，未使用集成或测试时增强。 *：使用摄像头信息的多模态方法。我们分别用灰色框和青色框标出最佳单帧结果和多帧结果。†：12 轮次训练计划。

方法	车辆长度（米） [0, 4)	[4, 8)	[8, 12)	[12, +∞)	官方*
CenterPoint-PP†	34.3	69.3	42.0	43.6	66.2
FSD^{plain}	32.2	64.6	41.3	42.2	62.3
SST$_{center}$ [6]	36.0	69.4	33.7	30.5	66.3
FSD^{nogc}	33.5 ↓ 2.5	68.2 ↓ 1.2	47.7 ↑ 14.0	47.9 ↑ 17.4	65.2 ↓ 1.1
FSD	36.7 ↑ 0.7	71.0 ↑ 1.6	51.3 ↑ 17.6	53.7 ↑ 23.2	69.3 ↑ 3.0

表 3 车辆检测及车辆长度细分。†：由我们自行重新实现。 *：Waymo 官方 L2 总体指标。箭头表示相较于 SST$_{center}$ 的性能变化。

要求完美的中心投票。FSD 的最终预测
是从完整的点组而非较弱的中心特征中得出的，从而避开了 CFM 的问题。

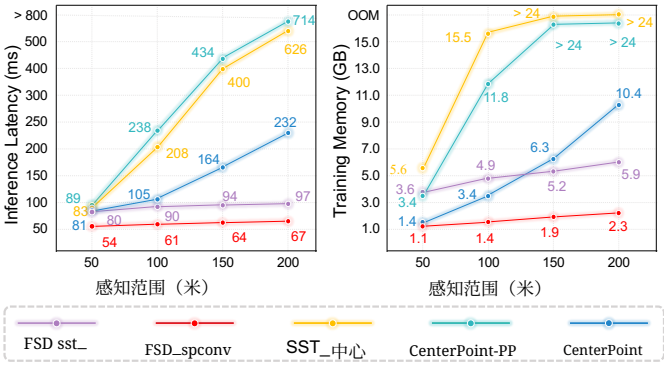


图 9. 不同感知范围下的内存占用和推理延迟。统计数据是在单张 3090 GPU 上以批量大小为 1 获得的。推理延迟通过 MMDetection3D 中的标准基准测试脚本进行评估，未采用任何测试时优化。CenterPoint-PP 和 SST$_{center}$ 定义见第 5.3 节。彩色效果更佳。

5.4 远距离检测

多个广泛采用的三维检测基准 [1], [2], [3]的感知范围相对较短。为了释放 FSD 的潜力，我们在远距离检测方面开展实验实验使用了最近发布的 Argoverse 2 数据集（AV2）。AV2 的感知范围可达 200 米，是测试我们方法的理想平台。此外，AV2 包含 30 个类别的物体，呈现长尾分布，这也是 FSD 面临的另一项挑战。

5.4.1 主要结果

我们首先在表 4 中列出 FSD 在 AV2 上的主要结果。AV2 的作者提供了一个基线 CenterPoint 模型，但其结果表现平平。为了进行公平比较，我们在 AV2数据集上重新实现了一个更强的 CenterPoint 模型。重新实现的 CenterPoint 采用了与 FSD 相同的训练方案，包括使用真实框采样来缓解长尾问题。FSD 在平均指标上优于 CenterPoint。值得注意的是，FSD 在一些微小物体（例如行人、施工锥桶）以及一些尺寸极大的物体（例如铰接式公交车、校车）上显著优于 CenterPoint。这得益于 SIR 中实例级细粒度特征提取的优势。

5.4.2 范围缩放

为了展示 FSD 在远距离检测中的效率，我们在图 9 中描绘了三种检测器在感知范围增大时训练内存和推理延迟的变化趋势。图9 显示，随着感知范围扩大，稠密检测器的延迟和内存占用会大幅增加。FSD 采用全稀疏设计，其所需资源与输入点数量大致呈线性关系，因此随着感知范围扩大，其内存占用和延迟仅略有增加。

5.5 FSD 性能评估

5.5.1 组件有效性

除了 FSD^{plain} 和 FSD^{nogc} （第 5.3 节）之外，我们还将 FSD 降级为 FSD^{agg} ，以深入了解其机制。在 FSD^{agg}中，我们在实例点分组后通过动态池化聚合分组后的点特征，然后直接

方法	*Average*	Vehicle	Bus	*Pedestrian*	*Stop Sign*	Box Truck	Bollard	*C-Barrel*	*Motorcyclist*	MPC-Sign	*Motorcycle*	Bicycle	*A-Bus*	*School Bus*	Truck Cab	*C-Cone*	V-Trailer	Sign	Large Vehicle	*Stroller*	*Bicyclist*
精度																					
CenterPoint† [9]	13.5	61.0	36.0	33.0	28.0	26.0	25.0	22.5	16.0	16.0	12.5	9.5	8.5	7.5	8.0	8.0	7.0	6.5	3.0	2.0	14
CenterPoint*	22.0	67.6	38.9	46.5	16.9	37.4	40.1	32.2	28.6	27.4	33.4	24.5	8.7	25.8	22.6	29.5	22.4	6.3	3.9	0.5	20.1
FSD	24.0	67.1	39.8	57.4	21.3	38.3	38.3	38.1	30.0	23.6	38.1	25.5	15.6	30.0	20.1	38.9	23.9	7.9	5.1	5.7	27.0
FSD‡	28.2	68.1	40.9	59.0	29.0	38.5	41.8	42.6	39.7	26.2	49.0	38.6	20.4	30.5	14.8	41.2	26.9	11.9	5.9	13.8	33.4
综合得分																					
CenterPoint*	17.6	57.2	32.0	35.7	13.2	31.0	28.9	25.6	22.2	19.1	28.2	19.6	6.8	22.5	17.4	22.4	17.2	4.8	3.0	0.4	16.7
FSD	19.1	56.0	33.0	45.7	16.7	31.6	27.7	30.4	23.8	16.4	31.9	20.5	12.0	25.6	15.9	29.2	18.1	6.4	3.8	4.5	22.1
FSD‡	22.7	57.7	34.2	47.5	23.4	31.7	30.9	34.4	32.3	18.0	41.4	32.0	15.9	26.1	11.0	30.7	20.5	9.5	4.4	11.5	28.0

表 4 Argoverse 2 验证集划分上的性能。†：由 AV2数据集 的作者提供。‡：用于防止过拟合的弱CopyPaste增强（每类一个实例）。 ∗：由我们重新实现。C-Barrel：施工路障桶。MPC-Sign：移动式人行横道标志。A-Bus：铰接式公交车。C-Cone：施工锥桶。V-Trailer：车辆拖车。由于狗、轮椅和信息板拖车这几个类别中的实例极少，因此省略了它们的结果。平均结果涵盖所有类别，包括省略的类别。我们将提升显著的类别标记为粗体。

从池化特征中进行预测。FSD^{agg} 的方式与 VoteNet 中的方式 [22]类似，正如我们在第 3.5 节中所讨论的那样。因此，FSD^{agg} 可以显式利用实例级特征，而不仅仅是 FSD 中的点级特征^{plain}，从而缓解 CFM 的问题。然而，FSD^{agg} 无法利用 SIR 中进一步的点特征提取。正如表5所示，如果只应用分组而不使用 SIR，改进幅度有限。另一方面，将分组与 SIR 结合则能带来显著的改进。

体素大小	AP CC	AP 路桩	AP 骑自行车者	AP 停车标志	延迟（毫秒）†
30cm	35.4	36.5	24.6	18.3	3.5
20cm	37.3	37.3	26.4	20.0	4.1
10cm	38.9	38.3	27.0	21.3	4.5
点	39.3	38.6	27.1	21.5	6.3

表6 不同表示粒度下的性能。†：SIR模块的延迟。

	分组	SIR	组校正	L2三维平均精度 车辆	行人	骑行者
FSD$_{plain}$				62.29	64.31	64.49
FSD^{agg}	✓			63.13	65.13	64.52
FSD$_{nogc}$	✓	✓		65.20	67.39	67.78
FSD	✓	✓	✓	69.30	69.30	69.60

表 5 SIR 中设计因素的消融实验。性能在 Waymo 验证集划分上进行评估。

	FSD Mem.	FSD 延迟（毫秒）	FSD mAP	CenterPoint Mem.	CenterPoint 延迟（毫秒）	CenterPoint mAP
all	5.9	97	24.0	10.4	232	22.0
仅感兴趣区域†	3.2 ↓ 45.8%	81↓ 16.5%	23.2	9.9↓ 4.8%	227↓ 2.2%	21.5
不含地面	2.3 ↓ 61.0%	74↓ 25.8%	21.0	9.7↓ 6.7%	217↓ 6.4%	19.8

表7 不同检测区域下的性能。†：感兴趣区域由AV2数据集中的高清地图定义。

5.5.2 SIR 中的下采样

SIR 的高效性使得无需对点进行任何下采样也能提取细粒度点特征。这也是 FSD 与 VoteNet 之间的另一项显著差异。为展示其优势，我们在 SIR 模块之前对原始点进行体素化，并将体素的质心视为下采样后的点。我们在 AV2数据集上进行实验，因为其中包含一些尺寸极小的类别，这些类别可能对下采样较为敏感。正如预期，采用下采样会导致小物体的性能明显下降，我们在表6中列出了其中一些类别。我们还评估了 SIR 模块在 3090 GPU 上的推理延迟，其效率非常高。

5.5.3 高清地图辅助检测

Argoverse 2 数据集提供了高度可靠的高清地图，可将其用作先验信息，以移除非关注区域，使场景更加稀疏。因此，我们通过实验移除部分非关注区域，以展示 FSD 在更稀疏场景中的优势。结果汇总于表7中。移除非关注区域后，FSD 的内存占用和延迟显著降低，同时精度仍在可接受范围内。相比之下，CenterPoint 的效率提升幅度较小。这表明，FSD 从数据稀疏性的增加中获益更多，这也是全稀疏架构的另一项优势。

5.6 FSD++的综合分析

5.6.1 FSD++ 分析的初步设置

在本节中，我们将进行大量实验，以揭示 FSD++的内部工作机制。首先，我们介绍本节所用基线模型的设置，该设置与表++ 1 和表2中的最佳 FSD模型略有不同。除非另有说明，第++ 5.6节中所有 FSD模型的默认超参数均列于表8的第一列。

本节报告的模型延迟是在单块 RTX 3090 GPU 上，以小批量大小为 1、采用 float32 精度测得的。为确保准确性，所有评估仅考虑模型本身的延迟，不包括输入输出的延迟；在多帧设置下，输入输出延迟可能不稳定。

值得注意的是，我们观察到骑行者类别的性能在不同次运行之间存在差异，这可能是由于数据集中该类别的数量较少。因此，我们在

	基线 FSD++	最佳 FSD++	多帧 FSD
计划	6 个训练轮次	12 训练轮次	6 训练轮次
SPS†	随机	随机	-
#. 帧	6	7	6
最大年龄	2	2	-
骨干网络*	SpUNet-base	SpUNet-large	SpUNet-base
SIR2 中的层数	3	3	3
RPP 尺寸	(0.25, 0.25, 0.4)	(0.25, 0.25, 0.4)	-

表 8 本节（第 5.6 节）所采用模型的基本超参数设置 †：SPS 表示骨架点采样。 ∗：SpUNet-large 比 SpUNet-base 多一个阶段 [7]，且其第一个阶段的通道数加倍。

#. 帧	L2 3D AP/APH				延迟
	Mean	车辆	行人	骑行者	(ms)
2	73.39/71.83	69.54/69.12	74.68/71.35	75.94/75.02	66.1
3	75.20/73.74	70.95/70.52	76.13/73.09	78.51/77.62	67.0
4	75.44/74.03	71.67/71.21	76.48/73.50	78.18/77.37	67.3
5	75.13/73.72	71.50/71.04	76.71/73.83	77.17/76.29	68.7
6	76.10/74.73	72.19/71.74	76.92/74.11	79.20/78.33	68.7
6（FSD）†	75.65/74.28	71.54/71.07	78.04/75.22	77.37/76.54	116.2

表10 不同帧数下 FSD++ 的性能。由于 FSD++ 使用先前的前景点，因此至少需要两帧。†：采用简单点拼接的多帧 FSD 模型。骑行者类别样本稀少，性能不稳定，因此我们将数值标为灰色。

一些实验中用灰色标记该类别的性能，以表示其结果可能不可靠。

5.6.2 骨架点采样

表 9 展示了采用不同骨架点采样策略时的性能。我们发现，所考察的三种策略之间没有显著差异。然而，采用随机骨架采样可大幅降低 FSD 的++ 延迟，同时不会牺牲性能。骨架采样还能持续提升骑行者类别的性能，这表明适当的采样能够缓解稀有类别的过拟合。对于行人类别，不进行采样时性能更好，这说明更多的点可能有助于检测小物体。实际应用中，尽管可以针对不同类别采用不同的采样策略，但为求简便且具有通用性，我们对所有类别都采用随机采样。

	L2 3D AP/APH				延迟
	Mean	车辆	行人	骑行者	(ms)
随机	76.10/74.73	72.20/71.74	76.93/74.11	79.20/78.33	68.7
物体帧率	75.56/74.21	72.06/71.59	76.94/74.14	77.68/76.88	72.3
体素采样	75.76/74.40	72.10/71.66	76.80/74.05	78.37/77.49	71.4
None	75.23/73.91	71.93/71.50	77.54/74.72	76.33/75.51	73.9

表9 不同骨架点采样方法的有效性。

5.6.3 不同帧数

FSD++ 从多个先前帧中采样骨架点。表10 展示了所用帧数对其性能的影响。由此可以得出两个有趣的发现。

- 随着帧数增加，性能会有所提升。同时，延迟并未显著增加。这要归功于残差点探测，它能够去除大部分背景。如果使用更多基准帧（公式 4），就能获得更多更干净的残差点。

- FSD++ 在车辆类别和骑行者类别上，使用相同帧数时的表现优于 FSD。我们也直观地认为，这要归功于 RPP，因为它去除了大部分背景杂波，减轻了分割的负担。行人类别的性能略低，这表明保留行人的所有点可能更好。不过，这种性能损失可以接受，因为 FSD++的平均性能更好，延迟也低得多。

5.6.4 漂移分析

鉴于 FSD++ 依赖历史预测，如果出现漂移误差，将是一个重大隐患。具体而言，如果检测器在时间步 t做出了不准确的预测，那么它很可能会在时间步 $t+1$表现得更差，因为 $t+1$ 中的预测依赖于来自 t（骨架点采样）的预测。

为防止潜在的漂移，我们在序列推理过程中定期插入一些关键帧。在关键帧处，我们使用标准 FSD 的预测进行骨架点采样，这可以视为对潜在漂移的校正。表格 11 显示，FSD++即使不使用任何关键帧，也能取得有竞争力的结果。第一行与最后一行之间的微小差距证实，FSD++ 的漂移可以忽略不计。

差距	L2 3D AP/APH			
关键帧	Mean	车辆	行人	骑行者
5	76.05/74.67	72.23/71.77	76.83/74.02	79.08/78.21
10	76.03/74.66	72.20/71.75	76.88/74.07	79.00/78.15
20	76.07/74.68	72.20/71.74	76.90/74.08	79.11/78.23
50	76.10/74.72	72.20/71.74	76.91/74.10	79.20/78.32
None	76.10/74.73	72.20/71.74	76.93/74.11	79.20/78.33

表11 不同关键帧间隔下的性能。“None”表示仅使用初始预测。

5.6.5 变化盲视消融

由于我们在第*4.3*节讨论的变化盲视，检测器可能会忽略新出现的物体。最大年龄旨在缓解变化盲视问题，表12展示了它的效果。我们发现，保留残差点两个时间步（最大年龄为 2）就足够了。

最大年龄	L2 3D AP/APH				延迟
	Mean	车辆 行人		骑行者	(ms)
1	75.17/73.82	71.38/70.94	76.74/73.97	77.40/76.56	65.3
2	76.10/74.73	72.20/71.74	76.93/74.11	79.20/78.33	68.7
3	**76.14/74.74**	**72.22/71.75**	**77.06/74.20**	79.13/78.27	70.2

表12 不同的残差点最大年龄。骑行者类别中的性能不稳定，因此我们将这些数字标为灰色。

为更深入地考察变化盲视问题，我们将原始 WOD 验证集中的物体划分为新出现的物体 和已有物体 ，以便进一步开展消融实验。新出现的物体是指在序列的第一帧中未出现、而在之后才出现的物体。结果如

表13。FSD 对新出现物体的召回率++(1)[1] 低于帧数相同的 FSD 6f。这表明变化盲视确实是 FSD 面临的问题++。然而，延长最大年龄后，FSD++ 在所有类别上的表现都优于 FSD 6f，这证明所提出的最大年龄能够有效缓解变化盲视。

方法	新出现物体的召回率			
	Mean	车辆	行人	骑行者
FSD	73.05	66.01	72.62	80.53
FSD 6f*	77.54	69.74	78.34	84.53
FSD++(1)	75.58	69.18	77.48	83.09
FSD++(2)	77.82	70.56	78.10	84.79
FSD++(3)	78.14	70.60	78.30	85.52

表13 新出现的物体的性能。∗：采用 6 帧拼接输入的 FSD。在 WOD 验证集划分中，新出现的物体数量分别约占车辆、行人和骑行者所有物体数量的 42.4%/37.1%/52.6%。

5.6.6 种子质量鲁棒性

在每个点云序列的推理过程中，FSD++都需要初始帧中的预测作为种子才能开始，正如我们在第 4.5.2 节所讨论的。这里我们探究种子预测的质量如何影响性能。具体而言，我们向种子预测中加入两种典型的随机噪声，包括随机框丢弃和随机框插入，以模拟假阴性和假阳性。所有实验均使用同一个训练好的 FSD++ 检测器。上述修改仅应用于推理阶段，不用于训练时数据增强。

表 14 显示 FSD++对这两种噪声都具有鲁棒性。特别是对于框丢弃，即使丢弃所有初始种子框，性能也仅略有下降。我们从两个方面解释这一令人惊讶的现象：(1) FSD++ 天生能够抵御运动物体被丢弃的影响。这是因为运动物体会产生大量残差点，而 FSD++ 能够根据这些残差点进行预测。(2) 由于视点变化，静态物体中仍会存在少量残差点。此外，最大年龄机制也有助于在静态物体上累积残差点。

对于框插入，FSD++ 几乎不受影响，因为这些框在 FSD 的分割阶段很容易被识别为背景。

噪声类型	L2 3D AP/APH			
	Mean	车辆	行人	骑行者
None	76.10/74.73	72.20/71.74	76.93/74.11	79.20/78.33
丢弃（10%）†	75.95/74.58	72.11/71.66	76.81/73.99	78.92/78.08
下降（50%）	75.76/74.39	71.86/71.41	76.63/73.82	78.78/77.95
下降（100%）	74.69/73.35	70.47/70.02	75.41/72.69	78.20/77.33
插入（10%）	76.00/74.62	72.16/71.70	76.82/73.99	79.01/78.16
插入（50%）	76.02/74.64	72.14/71.69	76.86/74.04	79.05/78.19
插入（100%）	75.98/74.61	72.16/71.71	76.84/74.02	78.95/78.10

表14 带噪种子预测的鲁棒性。†：括号中的百分比表示删除/插入实例的比例。

1. 括号中的数字表示最大 ag es.

5.6.7 残差点探测分析

量化大小和基准帧数量是 RPP 中两个重要的超参数。下面展示它们如何影响输出残差点和性能。

量化大小使 RPP 对轻微的点扰动具有鲁棒性。表15列出了不同量化大小下的结果。从“残差点比例”可以看出，量化大小越大，RPP 产生的残差点越少，从而使检测器效率更高，但性能也略有下降。

量化 size	L2 3D AP/APH			残差点比例†	延迟 (ms)
	车辆	行人	骑行者		
(0.15, 0.15, 0.4)	72.06/71.60	77.19/74.43	77.53/76.67	17.4%	72.3
(0.25, 0.25, 0.4)	72.20/71.74	76.93/74.11	79.20/78.33	9.6%	68.7
(0.35, 0.35, 0.4)	72.04/71.58	76.88/74.03	78.54/77.68	6.0%	65.2

表15 残差点探测中量化大小的有效性。†：残差点比例表示单帧中残差点数占总点数的平均比例。

基帧（见公式4）对 RPP 也有相当大的影响。纳入的基准帧越多，得到的残差点就越少，从而提高效率。此外，增加基准帧几乎不会影响性能。

#. RPP 基准帧	L2 3D AP/APH			残差点比例†	延迟 (ms)
	车辆	行人	骑行者		
3	72.48/72.03	77.16/74.35	77.60/76.78	14.4%	70.2
4	72.24/71.79	77.04/74.20	78.44/77.60	10.8%	69.0
5	72.20/71.74	76.93/74.11	79.20/78.33	9.6%	68.7
6	72.22/71.77	77.41/74.59	78.60/77.74	9.4%	68.5

表16 残差点探测中基帧数量的有效性。†：残差点比例表示单帧中残差点数占总点数的平均比例。

5.7 详细运行时评估

这里我们详细介绍 FSD 和 FSD++各组件的效率。所有评估模型均使用 SpUNet-large 作为骨干网络。评估在单张 RTX 3090 上以 FP32精度进行，且未采用任何测试时优化。我们仅记录使用 MMDetection3Dv0.15 实现的检测器的单样本前向传播延迟，不计点云的输入输出，因为其在多帧设置下不稳定。图10 展示了详细结果，这些结果是在验证集划分的前十个序列上评估所得的平均值。

从图中可以看出，分割器的延迟大幅降低。分割器由稀疏体素编码（即骨干网络）和分割头组成。因此，FSD++ 的速度与单帧 FSD 一样快，但性能优于 FSD 6f（表10）。值得强调的是，延迟中的“其他”部分通常由一些串行操作造成，例如类别专属检测头和类别专属非极大值抑制。在部署时，这部分延迟可以大幅降低。

6 结论

本文首次提出一种基于激光雷达的全稀疏三维目标检测框架，即 FSD。FSD 摒弃了

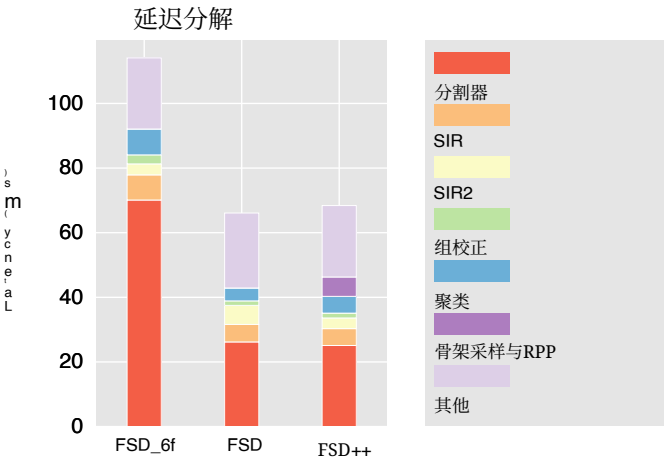


图10. 多帧 FSD、单帧 FSD 和 FSD++ 的延迟分解。

以往广泛采用的稠密鸟瞰图特征图，因为它阻碍了检测器实现全稀疏。FSD 转而由通用的稀疏体素编码器和高效的稀疏实例识别（SIR）模块组成。SIR 解决了中心特征缺失这一问题，而这正是全稀疏架构面临的核心难点。FSD 不仅在 Argoverse 2 数据集上实现了高效的远距离（最远 200 米）检测，还在竞争激烈的 Waymo 开放数据集上取得了最先进的性能。

为充分发挥 FSD 的潜力，我们提出利用时序信息来消除数据冗余。我们提出的骨架点采样和残差点探测，为 FSD 提供了一个超稀疏的输入点云，从而构成了 FSD++框架。FSD++ 在 Waymo 开放数据集的验证集划分和测试集划分上均取得了单模型最先进的性能，同时保持了较高的效率。我们希望这项工作能为未来基于激光雷达的点云识别研究指明方向。

附录 A 高效动态池化

所提出的 SIR 由三种基本操作组成：多层感知机、动态广播和动态池化。多层感知机已在主流深度学习框架中得到高度优化。动态广播本质上是一种索引操作，在现代图形处理器上具有高度并行性。SIR 的效率瓶颈在于动态池化。因此，本节将介绍一种高效的动态池化实现方法。

A.1 实现

PyTorch 实现的动态池化称为散射操作。在这种实现中，每个线程处理一个特征，并通过简单的原子操作进行特征归约。大量组中的密集原子操作会损害并行性。

我们从三个方面优化动态池化算子。(1) 将较大的组划分为固定大小的较小子组，以平衡工作负载。(2) 对

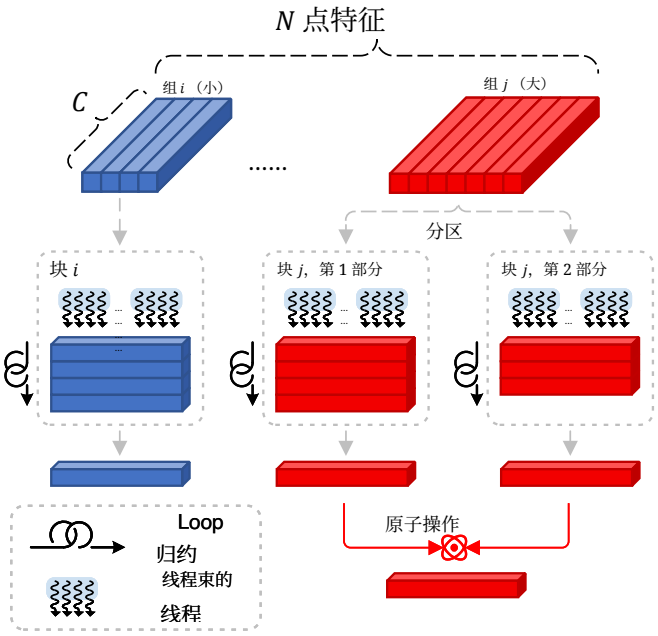


图 11. CUDA 中动态池化实现示意图。彩色观看效果最佳。我们以两个大小不同的组为例。动态池化会将每个组归约为单个特征。

具有相同组ID的特征进行排序，使其位于相邻位置，以增强内存局部性。需要注意的是，每个元素的组ID在整个SIR模块中保持不变，因此排序只需执行一次。(3) 将线程束中的线程分配到相邻通道，以实现合并内存访问并避免线程束分歧。

具体而言，每个线程对应一个特征维度，并通过循环对分区子组中的特征进行归约。此外，原子算子的调用次数从每个线程一次减少到每个线程块一次，从而提高了并行性。图 11 展示了我们的高效实现。

A.2 运行时评估

我们以动态最大池化为例，在不同情况下评估我们的实现与 *torch scatter* 的延迟，包括多种特征维度、多种组大小，以及组大小是否均衡。评估中的组总数为 100。结果见表17 和表 18。对于不同的数据规模，我们的实现实现了 2。48× 至 39。58× 倍加速比。而且，组大小不均衡时，加速效果更为显著。

特征维度	不同组大小下的延迟（毫秒）			
	$[10^0, 10^1)$	$[10^1, 10^2)$	$[10^2, 10^3)$	$[10^3, 10^4)$
64	0.06/0.16	0.06/0.16	0.08/1.49	0.24/14.05
256	0.06/0.18	0.06/0.33	0.14/3.52	0.72/20.53
1024	0.09/0.16	0.10/0.85	0.37/6.53	4.82/65.28
加速比	2.48×	5.56×	20.47×	30.53×

表17 在**均衡** 组大小的数据上进行动态最大池化的延迟。每个数值表示**我们的方法/torch scatter** 的延迟，单位为毫秒。

–

特征维度	不同组大小下的延迟（毫秒） $[10^0, 10^1)$*	$[10^1, 10^2)$*	$[10^2, 10^3)$*	$[10^3, 10^4)$*
64	0.06/0.16	0.06/0.32	0.09/2.60	0.36/20.87
256	0.06/0.15	0.08/0.78	0.20/5.44	1.19/32.75
1024	0.06/0.24	0.12/1.36	0.57/9.24	5.91/196.4
加速比	3.05×	8.81×	24.01×	39.58×

表18 不均衡组大小数据上的动态最大池化延迟。每项表示 ours/torch scatter 的延迟，单位为毫秒。 *：将十分之一组的大小扩大 10× ，以创建不均衡数据。

参考文献

[1] P. Sun、H. Kretzschmar、X. Dotiwalla、A. Chouard、V. Patnaik、P. Tsui、J. Guo、Y. Zhou、Y. Chai、B. Caine 等人，“自动驾驶感知的可扩展性：Waymo 开放数据集”，发表于 CVPR，2020年。[2] A. Geiger、P. Lenz 和 R. Urtasun，“我们准备好自动驾驶了吗？KITTI视觉基准套件”，发表于2012年IEEE计算机视觉与模式识别会议。IEEE，2012年。[3] 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”，发表于 CVPR，2020年。[4] B. Wilson、W. Qi、T. Agarwal、J. Lambert、J. Singh、S. Khandelwal、B. Pan、R. Kumar、A. Hartnett、J. K. Pontes、D. Ramanan、P. Carr 和 J. Hays，“Argoverse 2：面向自动驾驶感知与预测的下一代数据集”，发表于2021年NeurIPS数据集与基准，2021年。[5] Y. Yan、Y. Mao 和 B. Li，“SECOND：稀疏嵌入卷积检测”，Sensors，第18卷，第10期，2018年。[6] L. Fan、Z. Pang、T. Zhang、Y.-X. Wang、H. Zhao、F. Wang、N. Wang 和 Z. Zhang，“借助稀疏变换器实现单步幅三维目标检测器”，发表于 CVPR，2022年。[7] S. Shi、Z. Wang、J. Shi、X. Wang 和 H. Li，“从点到部件：利用部件感知与部件聚合网络进行点云三维目标检测”，IEEE模式分析与机器智能汇刊，2020年。[8] S. Shi、C. Guo、L. Jiang、Z. Wang、J. Shi、X. Wang 和 H. Li，“PV-RCNN：用于三维目标检测的点-体素特征集抽象”，发表于 CVPR，2020年。[9] T. Yin、X. Zhou 和 P. Kr¨ahenb¨uhl，“基于中心的三维目标检测与跟踪”，arXiv 预印本 arXiv:2006.11275，2020年。[10] S. Shi、L. Jiang、J. Deng、Z. Wang、C. Guo、J. Shi、X. Wang 和 H. Li，“PV-RCNN++：利用局部向量表示进行点-体素特征集抽象，以实现三维目标检测”，arXiv 预印本 arXiv:2102.00463，2021年。[11] C. R. Qi、L. Yi、H. Su 和 L. J. Guibas，“PointNet++：度量空间中点集的深度层次特征学习”，发表于神经信息处理系统大会，2017年。[12] Z. Yang、Y. Sun、S. Liu 和 J. Jia，“3DSSD：基于点的三维单阶段目标检测器”，发表于 CVPR，2020年。[13] Y. Zhang、Q. Hu、G. Xu、Y. Ma、J. Wan 和 Y. Guo，“并非所有点都相同：为三维激光雷达点云学习高效的基于点的检测器”，发表于 CVPR，2022年。[14] Y. Zhou、P. Sun、Y. Zhang、D. Anguelov、J. Gao、T. Ouyang、J. Guo、J. Ngiam 和 V. Vasudevan，“激光雷达点云三维目标检测的端到端多视图融合”，发表于机器人学习大会，2020年。[15] B. Li，“用于点云车辆检测的三维全卷积网络”，发表于智能机器人与系统国际会议，2017年。[16] Y. Zhou 和 O. Tuzel，“VoxelNet：基于点云的三维目标检测端到端学习”，发表于 CVPR，2018年。[17] X. Chen、H. Ma、J. Wan、B. Li 和 T. Xia，“面向自动驾驶的多视图三维目标检测网络”，发表于 CVPR，2017 年。[18] B. Yang 、 W. Luo 和 R. Urtasun，“PIXOR：实时点云三维目标检测”，发表于 CVPR，2018年。[19] A. H. Lang、S. Vora、H. Caesar、L. Zhou、J. Yang 和 O. Beijbom，“PointPillars：用于点云目标检测的快速编码器”，发表于 CVPR，2019年。[20] C. R. Qi、H. Su、K. Mo 和 L. J. Guibas，“PointNet：点集的深度学习及三维分类与分割”，发表于 CVPR，2017年。[21] S. Shi、X. Wang 和 H. Li，“PointRCNN：点云三维目标候选生成与检测”，发表于 CVPR，2019年。

[22] C. R. Qi、O. Litany、K. He 和 L. J. Guibas，《用于点云三维目标检测的深度霍夫投票》，发表于国际计算机视觉大会，2019年。 [23] J. Yang、L. Song、S. Liu、Z. Li、X. Li、H. Sun、J. Sun 和 N. Zheng，《DBQ-SSD：用于高效三维目标检测的动态球查询》，arXiv预印本 arXiv:2207.10909，2022年。 [24] Y. Hu、Z. Ding、R. Ge、W. Shao、L. Huang、K. Li 和 Q. Liu，《AFDetV2：重新思考点云目标检测中第二阶段的必要性》，arXiv预印本 arXiv:2112.09205，2021年。 [25] Y. Li、A. W. Yu、T. Meng、B. Caine、J. Ngiam、D. Peng、J. Shen、B. Wu、Y. Lu、D. Zhou 等，《DeepFusion：用于多模态三维目标检测的激光雷达与相机深度融合》，发表于CVPR，2022年。 [26] W. Liu、D. Anguelov、D. Erhan、C. Szegedy、S. Reed、C.-Y. Fu 和 A. C. Berg，《SSD：单次多框检测器》，发表于ECCV，2016年。 [27] S. Ren、K. He、R. Girshick 和 J. Sun，《Faster R-CNN：借助区域建议网络实现实时目标检测》，神经信息处理系统大会，第28卷，2015年。 [28] X. Zhou、D. Wang 和 P. Kr¨ahenb¨uhl，《以点表示物体》，arXiv预印本 arXiv:1904.07850，2019年。 [29] W. Zheng 、 W. Tang 、 L. Jiang 和 C.-W. Fu，《SE-SSD：基于点云的自集成单阶段目标检测器》，发表于CVPR，2021年。 [30] W. Zheng、W. Tang、S. Chen、L. Jiang 和 C.-W. Fu，《Cia-ssd：置信度感知IoU的基于点云的单阶段目标检测器》，发表于美国人工智能协会年会，2021年。 [31] C. He、H. Zeng、J. Huang、X.-S. Hua 和 L. Zhang，《基于点云的结构感知单阶段三维目标检测》，发表于CVPR，2020年。 [32] J. Deng、S. Shi、P. Li、W. Zhou、Y. Zhang 和 H. Li，《Voxel R-CNN：迈向高性能基于体素的三维目标检测》，发表于美国人工智能协会年会，2021年。 [33] T.-Y. Lin、P. Goyal、R. Girshick、K. He 和 P. Doll´ar，《用于密集目标检测的焦点损失》，发表于国际计算机视觉大会，2017年。 [34] L. Jiang、H. Zhao、S. Shi、S. Liu、C.-W. Fu 和 J. Jia，《Pointgroup：用于三维实例分割的双集合点分组》，发表于CVPR，2020年。 [35] W. Wang、R. Yu、Q. Huang 和 U. Neumann，《SGPN：用于三维点云实例分割的相似性组提议网络》，发表于CVPR，2018年。 [36] F. Hong、H. Zhou、X. Zhu、H. Li 和 Z. Liu，《通过动态移位网络实现基于激光雷达的全景分割》，发表于CVPR，2021年。 [37] R. E. Tarjan，《一类需要非线性时间来维护不相交集合的算法》，《计算机与系统科学杂志》，1979年。 [38] Y. Wang、Y. Sun、Z. Liu、S. E. Sarma、M. M. Bronstein 和 J. M. Solomon，《用于点云学习的动态图卷积神经网络》，《ACM图形学汇刊》（TOG），第38卷，第5期，第1–12页，2019年。 [39] L. Fan、X. Xiong、F. Wang、N. Wang 和 Z. Zhang，《RangeDet：为范围视图在激光雷达三维目标检测中的应用辩护》，发表于国际计算机视觉大会，2021年。 [40] A. Vaswani、N. Shazeer、N. Parmar、J. Uszkoreit、L. Jones、A. N. Gomez、Ł. Kaiser 和 I. Polosukhin，《注意力就是一切》，发表于神经信息处理系统大会，2017年。 [41] D. Hendrycks 和 K. Gimpel，《高斯误差线性单元（GELUs）》，arXiv预印本 arXiv:1606.08415，2016年。 [42] A. Bewley、P. Sun、T. Mensink、D. Anguelov 和 C. Sminchisescu，《用于尺度不变三维目标检测的范围条件扩张卷积》，arXiv预印本 arXiv:2005.09927，2020年。 [43] Z. Li、F. Wang 和 N. Wang，《LiDAR R-CNN：高效通用三维目标检测器》，发表于CVPR，2021年。 [44] M. Contributors，《MMDetection3D：面向通用三维目标检测的OpenMMLab新一代平台》，https://github.com/open-mmlab/mmdetection3d，2020年。 [45] R. Ge、Z. Ding、Y. Hu、Y. Wang、S. Chen、L. Huang 和 Y. Li，《AFDet：无锚单阶段三维目标检测》，arXiv预印本 arXiv:2006.12671，2020年。 [46] Y. Wang、A. Fathi、A. Kundu、D. Ross、C. Pantofaru、T. Funkhouser 和 J. Solomon，《面向自动驾驶的基于柱体的目标检测》，发表于ECCV，2020年。 [47] J. Mao、Y. Xue、M. Niu、H. Bai、J. Feng、X. Liang、H. Xu 和 C. Xu，《用于三维目标检测的体素Transformer》，发表于国际计算机视觉大会，2021年。 [48] J. Mao、M. Niu、H. Bai、X. Liang、H. Xu 和 C. Xu，《Pyramid R-CNN：提升三维目标检测的性能与适应性》，发表于国际计算机视觉大会，2021年。 [49] J. Li、H. Dai、L. Shao 和 Y. Ding，《从体素到点：借助体素到点解码器实现面向点云的IoU引导三维目标检测》，发表于ACM多媒体会议，2021年。 [50] Z. Yang、Y. Zhou、Z. Chen 和 J. Ngiam，《3D-MAN：用于目标检测的三维多帧注意力网络》，发表于CVPR，2021年。

[51] T. Guan、J. Wang、S. Lan、R. Chandra、Z. Wu、L. Davis 和 D. Manocha，《M3DETR：基于Transformer的多表示、多尺度、互关系三维目标检测》，发表于*WACV*，2022年。 [52] P. Sun、W. Wang、Y. Chai、G. Elsayed、A. Bewley、X. Zhang、C. Sminchisescu 和 D. Anguelov，《RSN：用于高效、精准激光雷达三维目标检测的范围稀疏网络》，发表于*CVPR*，2021年。 [53] G. Shi、R. Li 和 C. Ma，《PillarNet：高性能基于柱体的三维目标检测》，*arXiv*预印本 *arXiv:2205.07403*，2022年。 [54] Z. Zhou、X. Zhao、Y. Wang、P. Wang 和 H. Foroosh，《Center-Former：基于中心的三维目标检测Transformer》，发表于*ECCV*。Springer，2022年。 [55] J. Xu、Z. Miao、D. Zhang、H. Pan、K. Liu、P. Hao、J. Zhu、Z. Sun、H. Li 和 X. Zhan，《INT：借助高效框架实现无限帧三维检测》，发表于*ECCV*。Springer，2022年。 [56] X. Chen、S. Shi、B. Zhu、K. C. Cheung、H. Xu 和 H. Li，《MPP-Net：利用代理点实现多帧特征交织，应用于三维时序目标检测》，发表于*ECCV*。Springer，2022年。 [57] H. Yang、Z. Liu、X. Wu、W. Wang、W. Qian、X. He 和 D. Cai，《Graph r-cnn：利用语义装饰的局部图实现精准三维目标检测》，发表于欧洲计算机视觉会议。Springer，2022年，第662–679页。 [58] Z. Liu、H. Tang、A. Amini、X. Yang、H. Mao、D. Rus 和 S. Han，《BEVFusion：基于统一鸟瞰图表示的多任务多传感器融合》，*arXiv*预印本 *arXiv:2205.13542*，2022年。 [59] T.-Y. Lin、M. Maire、S. Belongie、J. Hays、P. Perona、D. Ramanan、P. Doll´ar 和 C. L. Zitnick，《Microsoft COCO：常见情境中的常见物体》，发表于欧洲计算机视觉会议。Springer，2014年。