Memory-based Adapters for Online 3D Scene Perception

摘要

在本文中，我们提出了一种新的用于在线 3D 场景感知（online 3D scene perception）的框架。传统的 3D 场景感知方法通常是离线的，即以已经重建的 3D 场景几何结构作为输入。然而，这种方法在机器人应用中并不适用，因为输入数据是连续的 RGB-D 视频流，而不是从预先收集的 RGB-D 视频中重建的完整 3D 场景。

为了处理在线 3D 场景感知任务（即数据收集与感知需同时进行），模型需要能够逐帧处理 3D 场景并利用时间信息。为此，我们为 3D 场景感知模型的主干网络（backbone）设计了一个基于适配器（adapter）的“即插即用”模块。该模块通过构建内存机制，缓存并聚合提取的 RGB-D 特征，从而赋予离线模型时间学习的能力。

具体来说，我们提出了一种队列式记忆机制（queued memory mechanism），用于缓存支持点云和图像特征。然后，我们设计了聚合模块，直接在记忆上进行操作，将时间信息传递给当前帧。此外，我们还提出了 3D 到 2D 的适配器（3D-to-2D adapter），以通过强大的全局上下文增强图像特征。

我们的适配器可以轻松插入到针对不同任务的主流离线架构中，并在在线任务中显著提升其性能。通过简单微调现有的离线模型，而无需进行任何特定于模型或任务的设计，我们的方法在 ScanNet 和 SceneNN 数据集上的实验表明，在三项 3D 场景感知任务中，相较于最先进的在线方法，我们的方法实现了领先的性能表现。

在实验复现中，我们进一步构建了与 CUDA、MinkowskiEngine 等软件的版本适配表，解决了 Python 环境配置、数据集获取和源码兼容性问题，优化了环境配置步骤。针对 ScanNet 和 SceneNN 数据集的 3D 语义分割、目标检测和实例分割任务，最终结果与原论文相比差距极小，证明了方法的可复现性。

本文也给出了对原项目的优缺点分析，在未来工作中，可以探索动态记忆机制以提升时间信息的利用效率，进一步优化高效微调技术（如 LoRA）以降低硬件需求。此外，通过跨模态深度融合点云与图像特征，有望提升对复杂场景的感知能力。在机器人导航、增强现实（AR）等应用中，本文方法展现了较大的潜力。

关键字：在线 3D 感知　RGB-D 视频　队列式记忆机制　3D-to-2D 适配器　特征聚合

一、 任务介绍

1.1 问题背景与研究意义

三维场景感知，包括语义分割、目标检测、实例分割等任务，是机器人或 AR/VR 应用的基本能力。自从 PointNet [1] 提出首个直接处理点云的模型以来，得益于精确且高效的架构设计，近年来 3D 场景感知 [2–6] 取得了巨大进步。

三维场景感知，包括语义分割、目标检测、实例分割等任务，近年来一直是一个受关注的研究方向，以室内感知为例，在 ScanNet 数据集上刷榜的文章已然数不胜数。然而，在具身智能越发火热的当下，我们却在各种操纵（manipulation）、巡航（navigation）的工作中难以发现这些感知方法的身影。其核心原因，在于绝大多数感知方法的 setting 都是离线的，难以直接应用于实际的机器人任务中。

然而，传统的 3D 场景感知方法通常是离线（offline）的，即预先通过收集的 RGB-D 视频，重建出不包含时间信息的 3D 场景几何结构，以此作为模型输入。而在大多数机器人应用中，例如巡航（navigation） [7, 8] 和操纵（manipulation） [9] 的工作中，代理（agent）通常是在未知环境中初始化的，输入数据是连续的 RGB-D 视频，并且场景感知需要与数据收集同步进行，以指导代理如何探索。因此，需要一种具有时间建模能力的在线 3D 场景感知模型，它以 RGB-D 视频流为输入，连续输出当前观测到的 3D 场景的感知结果。

目前也有一些针对特定架构和任务设计的在线 3D 场景感知方法 [10–14]。然而，这些方法仅关注单一模态的时间信息聚合，未能充分利用图像和点云特征之间的时间关系，因此性能并不理想。

本文中，我们提出了一种通用框架用于在线 3D 场景感知。不同于以往基于特定架构和任务设计的在线感知方法，并从零开始训练 RGB-D 视频模型的方法，我们通过简单地插入“即插即用”（plug and play）模块并进行微调，可以将现有的离线 3D 感知模型转化为在线模型。受到适配器 [15, 16] 的启发，这些适配器通过额外的参数调优，将图像主干适配到下游任务，我们提出了一种基于内存的适配器，以复用前帧提取的特征赋予 3D 感知模型主干时间建模能力。

具体来说，我们提出了一种队列式内存机制，用于缓存当前时间 RGB-D 帧的支持点云和图像特征。基于内存结构，我们设计了直接在内存上操作的聚合模块，将时间信息传递到当前帧。由于图像特征的全局上下文有限，我们进一步提出了 3D 到 2D 适配器，通过 3D 内存投影和 2D 稀疏聚合增强图像特征。通过这种方式，我们可以利用现有的主流 3D 场景感知模型库，简单地插入适配器并进行微调，从而获得一系列在线模型。

我们在 ScanNet [17] 和 SceneNN [18] 数据集上进行了三项在线感知任务的广泛实验。我们的方法在所有任务和数据集上都达到了领先性能，且不需要额外的损失函数和特殊的预测融合策略。总结来说，我们的贡献包括：

- 我们提出了一种新的在线 3D 场景感知框架，通过适配器将现有的离线模型扩展为在线模型，无需模型和任务特定设计。
- 我们提出了通用的基于内存的适配器，用于图像和点云主干，能够缓存和聚合提取的特征，以建模帧间的时间关系。
- 配备我们适配器的离线模型能够在三项任务中相比最先进的在线模型取得领先性能。

1.2 相关工作

3D 场景感知： 3D 场景感知在计算机视觉领域得到了广泛研究，主要分为三大任务：语义分割 [1, 2, 19, 20]、目标检测 [3, 4, 21, 22] 和实例分割 [5, 6, 23–25]。由于本文的研究重点是 3D 场景特征提取，因此主要讨论 3D 场景感知网络的主干设计。由于点云数据无序的特性，将点云体素化并在 3D 网格上应用卷积是一种自然的解决方案 [26, 27]。然而，体素分辨率的提高会导致计算成本和内存需求呈立方增长，这种方法效率较低。PointNet [1] 是首个直接从原始点云中提取特征表示的开创性工作。PointNet++ 在 PointNet 的基础上引入了集合抽象和特征传播操作，从而帮助学习更详细的局部几何信息。然而，由于 PointNet++ 中的最远点采样操作非常耗时，PV-CNN [28] 通过将点云转化为低分辨率体素并应用 3D 卷积来高效聚合局部特征。另一种提取高质量 3D 特征的方法是稀疏卷积 [29, 30]，它将点云体素化，但仅对非空体素应用 3D 卷积。为了进一步提高稀疏卷积神经网络的效率，提出了子流形稀疏卷积 [2, 20]，该方法仅在卷积核中心滑过活动位置时进行卷积，并在网络中保持相同水平的稀疏性。然而，这些方法是为离线 3D 场景感知设计的，无法实时处理流式 RGB-D 视频。

数据流分析： 由于在线 3D 场景感知模型的输入是 RGB-D 视频流，我们回顾了图像和点云领域的流数据分析方法。在 2D 视觉中，许多工作 [31–33] 将因果卷积 [34] 扩展到流视频，其中设计了流缓冲区来缓存之前的帧，并通过 3D 因果卷积单向地聚合时空信息。TSM [35] 提出了一种更高效的通道移位机制，将前一帧图像特征的一部分通道移位到下一帧，然后通过 2D 卷积高效聚合时空信息。我们的图像适配器也利用了通道移位进行高效的时间建模。与 TSM 不同的是，TSM 从零开始训练，网络可以根据移位比例学习如何建模时间信息；而我们通过重新组织通道并采用即插即用的适配器中的通道移位来增强图像主干的时间建模能力。然而，2D 流式视频对机器人导航 [7, 8] 和操作 [9] 等实际应用提供的信息有限，因此 3D 流式 RGB-D 视频分析受到了越来越多的关注。

一种自然的解决方案是先处理 2D 图像，然后将预测结果投影到 3D 点云中，随后

通过融合步骤合并来自不同帧的预测结果 [10, 11]。例如，Fusion-aware 3D-Conv [12] 和 SVCNN [13] 在 3D 空间中保留先前帧的信息，并基于点卷积融合 3D 特征以进行语义分割。INS-CONV [14] 扩展了稀疏卷积到增量 CNN 以高效提取全局 3D 特征，用于语义和实例分割。与这些方法不同，我们的方法通过图像和点云基于记忆的适配器赋予离线模型在线感知能力，从而充分利用多模态时间关系。

二、 符号说明

符号	说明	形状
I_t	t 时刻的图像	$H \times W \times 3$
P_t	t 时刻的点云	$N \times 3$
M_t	位姿转换参数	3×4
s	放大参数	标量
R_1/R_2	空间转换矩阵	$C \times C'$
$M_I(\cdot)$	图像特征	$H \times W \times C$
$M_P(\cdot)$	点云特征	-
m_t^P	t 时刻前的点云记忆	-
m_t^I	t 时刻前的图像记忆	-

三、 模型的建立和求解

在本节中，首先介绍在线 3D 场景感知的定义，并解释我们使用基于记忆的适配器的动机。然后描述如何为点云和图像分别构建记忆并通过适配器细化主干特征。

3.1 在线 3D 场景感知

设 $\mathcal{X}_t = \{x_1, x_2, ..., x_t\}$ 为一个有位姿的 RGB-D 流式视频，这意味着视频是由传感器移动时采集的，而不是预先采集完成的视频。我们有：

$$x_t = (I_t, P_t, M_t),\ I_t \in \mathbb{R}^{H\times W\times 3},\ P_t \in \mathbb{R}^{N\times 3},\ M_t \in \mathbb{R}^{3\times 4} \tag{1}$$

其中 I_t 和 P_t 分别表示一帧 RGB-D 图像的图像部分和点云部分。通过利用位姿参数 M_t 将深度图像转换到世界坐标系中，从而得到点云 P_t ，其中 M_t 可以借助视觉里程计测量得到 [36, 37]。在 t 时刻，在线感知模型的输入为 $\mathcal{X}_t$，输出为当前观察到的 3D 场景 $S_t = \bigcup_{i=1}^{t} P_i$ 的预测结果，如边界框或语义/实例掩码。一些工作还会结合在线感知任务进行 3D 重建 [14] ，以获得高质量点云或网格，但这不是必要的 [7, 8, 38]。在本文中，我们直接以 RGB-D 流式视频作为输入，而不依赖 3D 重建，这是一种更通用的设置.

尽管现有的 3D 感知模型在设计上取得了很大进展，但它们大多只关注两种场景 (1) 场景重建 [17, 39]：模型 $\mathcal{M}_{Rec}$ 通过从 RGB-D 视频重建的完整场景点云进行训练。(2) 单视角场景 [40, 41]: 模型 $\mathcal{M}_{SV}$ 使用从 RGB-D 图像反投影得到的单视角点云进行训练。然而，$\mathcal{M}_{Rec}$ 需要完整的场景输入，这在实时任务中无法实现；而 $\mathcal{M}_{SV}$ 能逐帧处理 RGB-D 视频，但无法利用时间信息。因此，现有的 3D 模型尚未准备好用于更实际的在线场景感知。因此，现有的 3D 模型尚未准备好用于更实际的在线场景感知。

为此，我们的目标是设计一个即插即用的时间学习模块，可以插入任何单视角感知模型 $\mathcal{M}_{SV}$ ，赋予其时间建模能力。注意,$\mathcal{M}_{SV}$ 本质上是一个 3D 感知模型，我们通过早期融合图像特征和点云 [42] 扩展为 RGB-D 感知模型：

$$
\begin{aligned}
p_t &= \mathcal{M}_{SV}(P'_t), \\
P'_t &= P_t \oplus \mathcal{S}(\mathcal{M}_I(I_t), P_t, M_t),\ \mathcal{S}(\cdot) \in \mathbb{R}^{N \times C}
\end{aligned} \tag{2}
$$

其中，p_t 是时间 t 的预测结果，$\mathcal{M}_I$ 是在相同感知任务上预训练的图像主干网络，$\mathcal{S}$ 是通过位姿 M_t 将点云 P_t 映射到图像坐标系，并采样对应的 2D 特征以丰富点云特征。我们将 $\mathcal{M}_{SV}$ 分为点云特征提取主干网络 $\mathcal{M}_P$ 和任务特定头 $\mathcal{M}_H$ 。本文的目标是为图像特征 $\mathcal{M}_I(I_t)$ 构建基于记忆的适配器，并为点云特征 $\mathcal{M}_P(P'_t)$ 构建相应的适配器，以存储和重用提取的主干特征进行时间建模：

$$
\begin{aligned}
\mathcal{M}_I(I_t), m_t^I &= \mathcal{A}_I(\mathcal{M}_I(I_t), m_{t-1}^I, m_{t-1}^P), \\
\mathcal{M}_P(P'_t), m_t^P &= \mathcal{A}_P(\mathcal{M}_P(P'_t), m_{t-1}^P)
\end{aligned} \tag{3}
$$

其中，基于记忆的适配器 $\mathcal{A}$ 通过当前特征更新记忆 m_t ，并重用记忆优化当前特征。我们充分利用跨模态和模态内的关系：$\mathcal{M}_{SV}$ 将图像特征融合到点云中，$\mathcal{A}_P$ 重用之前的点云特征优化当前点云特征，$\mathcal{A}_I$ 重用之前的图像特征优化当前图像特征。正如图 2 所示，我们遵循相同的设计范式，为图像和点云主干网络设计模块，从而实现时间建模。

3.2 点云的时间建模

在时间 t，给定点云主干网络提取的特征序列 $\{\mathcal{M}_P(P'_1), \mathcal{M}_P(P'_2), ..., \mathcal{M}_P(P'_t)\}$，我们的目标是通过利用此序列中的时间关系，增强当前帧特征 $\mathcal{M}_P(P'_t)$。具体来说，我们首先构建一个高效缓存点云特征的内存机制，然后通过即插即用的适配器聚合内存中的时间信息到当前帧特征中。

内存构建： 3D 场景的时间信息体现在更完整的几何结构中。单帧 RGB-D 数据可能无法包含大型物体或高层次场景的完整信息，因此来自前几帧的几何信息对于准确感知当前帧至关重要。因此，我们将提取的点云特征序列缓存到共享的 3D 空间中。一种简单的方法是直接以世界坐标系的形式存储点云特征。然而，这在存储和计算上效率低

下：（1）点云坐标是实数值，即使 RGB-D 相机不移动，存储开销仍会不断增加；（2）随着时间推移，点的数量会非常庞大，从而导致点云采样和特征聚合的计算开销大幅增加。为了解决这些问题，我们提出在量化坐标系中存储特征，即将点云体素化并存储在 3D 网格中。此外，当场景过大时，我们通过维护一个队列来减少内存开销。具体而言，首先将 $\mathcal{M}_P(P'_t)$ 体素化为体素网格 V_t，通过对落入同一网格的所有特征求平均完成体素化操作。这些体素会被标记时间戳 t。然后，我们通过最大池化将 V_t 合并到内存 m^P_{t-1} 中：

$$\begin{aligned} m^P_t &= \text{maxpooling}(V_t, m^P_{t-1}), \\ m^P_t &= \text{deq}(m^P_t, l) \;\; if \;\; N(m^P_t) > N_{max} \end{aligned} \tag{4}$$

其中，最大池化是指在每个体素网格上进行通道维度的最大池化操作，这会同时更新特征和时间戳。最大池化保留了时间上的最具判别性的特征，并且计算效率高，因为只需要处理时间 $t-1$ 和 t 的特征。此外，当内存中的体素数量 $N(m^P_t)$ 超过设定的最大值 N_{max} 时，通过 $\text{deq}(\cdot, l)$ 删除时间戳早于 $t-l+1$ 的体素.

基于内存的适配器： 在点云特征被缓存并更新到体素后，我们需要利用 m^P_t 的时间信息来增强当前帧的点云特征 $\mathcal{M}_P(P'_t)$。为了在充分利用内存丰富场景上下文的同时减少冗余计算，我们首先使用 V_t 的坐标查询邻域体素集合：

$$\mathcal{N}(V_t) = \{m^P_t[x][y][z] | (x, y, z) \in s * \mathcal{B}(V_t)\} \tag{5}$$

其中，$\mathcal{N}(V_t)$ 是 V_t 查询到的邻域体素集合，$m^P_t[x][y][z]$ 表示 m^P_t 中坐标为 (x, y, z) 的体素。$\mathcal{B}$ 是 V_t 的最小包围轴对齐边界框，s 是一个放缩因子，用于扩大边界框的范围。通过这种方式，提供当前帧支持几何信息的时间信息被收集到该体素集合中。

然后，我们将 $\mathcal{N}(V_t)$ 转换为稀疏张量 [2, 20]，并通过 3D 稀疏卷积模块 $\mathcal{A}_P$ 聚合上下文信息到 V_t 的位置。最终，我们通过以下两步更新 $\mathcal{M}_P(P'_t)$：（1）将聚合的特征映射回 $\mathcal{M}_P(P'_t)$ 的坐标，并通过残差连接将其添加到原始特征中。（2）将适配器模块 $\mathcal{A}_P$ 零初始化，从而在插入适配器后，微调训练能够从原始点平滑开始。

3.3 图像的时间建模

对于时间 t 的图像特征序列 $\{\mathcal{M}_I(I_1), ..., \mathcal{M}_I(I_t)\}$，我们采用与点云特征类似的范式，将其存储到内存中，并通过适配器将时间信息聚合到当前帧的图像特征 $\mathcal{M}_I(I_t)$ 中。

内存构建： 与点云数据不同，来自不同时间戳的点云可以存储在共享的 3D 空间中，而图像特征通常被堆叠为视频流数据。一种常见的处理方法 [33] 是维护一个队列，并沿时间维度执行因果卷积（单向卷积），以将前几帧的信息聚合到当前帧的图像特征中。然

而，这种视频分析方法的重点在于提取视频的全局信息，而我们在此只需要增强当前帧的图像特征 $\mathcal{M}_I(I_t)$。因此，直接在前几帧上执行因果卷积会引入大量的冗余计算。

此外，在在线 3D 场景感知中，图像特征最重要的信息是从多个视角对物体的观测。由于一个物体通常会出现在相邻的几帧中，维护一个短队列在大多数情况下已经足够。因此，我们进行了一种极简化：仅存储前一帧的图像特征，并利用时间 $t-1$ 和 t 之间的时间关系。

为了高效聚合相邻帧的信息，我们采用通道移位（channel shift）来缓存时间信息。具体来说，给定 $\mathcal{M}_I(I_t) \in \mathbb{R}^{H\times W\times C}$，我们学习一个线性变换 $R_1 \in \mathbb{R}^{C\times C'}$，将图像特征映射到另一个嵌入空间，其中前 $\frac{1}{\tau}$ 的通道包含与下一帧相关的丰富时间信息。因此，内存可以简单地通过移出这一部分通道构建：

$$m_t^I = (\mathcal{M}_I(I_t)\cdot R_1)_{[:,:,:\frac{C'}{\tau}]} \tag{6}$$

注意，这一操作是逐帧重复的，因此 m_{t-1}^I 包含与当前帧 $\mathcal{M}_I(I_t)$ 相关的时间信息。

基于内存的适配器： 在时间 t，在将一部分通道移入内存后，我们可以用之前的内存 m_{t-1}^I 填充空通道。通过这种方式，相邻两帧的时间信息被合并到一个单帧中，我们可以直接采用 2D 卷积来聚合特征：

$$F_t = \text{2D-Conv}(m_{t-1}^I \oplus (\mathcal{M}_I(I_t)\cdot R_1)_{[:,:,\frac{C'}{\tau}:]})\cdot R_2 \tag{7}$$

其中 $R_2 \in \mathbb{R}^{C'\times C}$ 是一个可学习的逆变换，用于将图像特征映射回原始嵌入空间。最终，我们通过残差连接将 F_t 添加到 $\mathcal{M}_I(I_t)$，以更新当前帧特征。同时，我们将 R_1, R_2 和 2D-Conv 卷积模块零初始化，以便在插入适配器后，微调能够从原始模型平滑开始。

3.4 跨模态时间建模

尽管维护一个短队列并采用通道移位可以有效聚合图像特征的时间信息，但其全局上下文信息仍然有限。这种限制在对象非常大或 RGB-D 相机停止移动时可能导致性能下降。然而，正如前文所述，从流式视频中直接提取丰富的全局上下文信息需要较高的内存和计算开销。为了解决这个问题，我们利用点云记忆来提取全局上下文，因为点云特征可以高效地缓存到共享的 3D 空间中，从而允许队列长度更长。

我们设计了一个 3D 到 2D 适配器 (3D-to-2D Adapter)，通过全局 3D 特征优化当前图像特征。首先，我们将 m_{t-1}^P 投影到离散的图像坐标系，使用 $\mathcal{S}$ 的逆函数完成该步骤。投影后的数据被转换为稀疏张量，以保持点云记忆的稀疏性并使 2D 特征具有几何感知能力; 接着在稀疏张量上应用 2D 稀疏卷积以聚合上下文信息，通过密化操作将特征限制在图像内部，其他像素填充为零。最后，将密化后的 2D 特征添加到 $\mathcal{M}_I(I_t)\cdot R_1$，从而增强其全局上下文信息。我们对 2D 稀疏卷积进行零初始化，以实现平滑的训练。

为了获得最终的在线感知结果，我们需要采用预测融合策略。由于本文的重点是图像和点云主干网络的时间学习模块，我们仅采用简单的后处理策略，将每一帧的预测结果融合为整体，具体细节在4.1节中说明。

四、 实验

在本节中，我们首先描述所使用的数据集和实现细节。随后，我们在房间级别和在线基准测试中将我们的方法与当前最先进的方法进行对比，以全面分析基于记忆的适配器的优势。最后，我们通过消融实验验证我们设计的有效性。

4.1 基准测试和实现细节

我们在两个数据集上评估了我们的方法：ScanNet [17] 和 SceneNN [18]。ScanNet 包含 1513 个扫描场景序列，其中我们使用 1201 个序列用于训练，312 个序列用于测试。SceneNN 是一个较小的数据集，包含 50 个高质量的场景序列。我们对其中的场景进行了筛选，选择了 12 个干净的序列用于测试。我们在 ScanNet 数据集上训练所有模型，并在 ScanNet 或 SceneNN 数据集上进行评估。

表 1　ScanNet 和 SceneNN 数据集上的 3D 语义分割结果。对于在线方法，我们将由带位姿的 RGB-D 图像拼接而成的点云预测结果映射到重建的点云上，以便与离线方法进行对比。

Method	Type	ScanNet		SceneNN	
		mIoU	mAcc	mIoU	mAcc
MkNet [2]	Offline	71.6	80.4	–	–
Fs-A [12]	Online	63.5	73.7	51.1	62.4
MkNet-SV	Online	68.8	77.7	48.4	61.2
MkNet-SV+Ours	Online	**72.7**	**84.1**	**56.7**	**70.1**

基准测试： 我们首先在房间级别的基准测试上比较不同的方法，即在重建的完整场景上的表现。对于语义分割任务，我们在 ScanNet 和 SceneNN 上比较了不同方法的性能。由于在线方法可能无法进行 3D 重建，我们将其对点云的预测（由 RGB-D 图像拼接而成）通过最近邻插值映射到重建的点云上。对于目标检测和实例分割任务，指标是基于每个对象单独计算的，而不是整个点云。因此，我们分别使用重建点云和 RGB-D 视频作为离线和在线方法的输入，并基于各自的输入计算指标。

为了更真实地评估在线感知任务的性能，我们参考 AnyView [43] 的方法，在 ScanNet 上组织了一个在线基准测试。在测试中，将每个房间的 RGB-D 视频划分为若干不重叠的序列，并将每个序列视为独立场景。通过设置不同的序列数量和长度，可以测试方法在场景不完整和规模变化条件下的泛化能力。这种设置更加贴近实际应用。在我们的实验中，我们将每个房间划分为 1/5/10 个序列或长度固定为 5/10/15 的序列，从而生成 6 个评估指标。

表 2　ScanNet 数据集上的 3D 目标检测和实例分割结果。离线方法和在线方法通过横线分隔。标记为 † 的方法表示 INS-Conv 需要额外的 3D 重建算法以获取高质量的点云或网格。

Detection			Insseg		
Method	mAP		Method	mAP	
	@25	@50		@25	@50
FCAF3D [3]	70.7	56.0	SoftGroup [6]	78.9	67.6
CAGroup3D [4]	74.5	60.3	TD3D [25]	81.3	71.1
AnyView [43]	60.4	36.0	INS-Conv† [14]	–	57.4
FCAF3D-SV	41.9	20.6	TD3D-SV	53.7	36.8
FCAF3D-SV+Ours	**70.5**	**49.9**	TD3D-SV+Ours	**71.3**	**60.5**

实现细节： 为了训练 $\mathcal{M}_{SV}$，我们首先按照 Pri3D [44] 的方法训练一个 2D 感知模型 $\mathcal{M}_I$。在语义分割任务中，我们使用 UNet [45]；在目标检测和实例分割任务中，我们使用 Faster-RCNN [46]（仅需要 ResNet [47] 和 FPN [48] 的主干网络）。然后，我们固定图像主干网络的参数，并在单视角的 RGB-D 数据集 ScanNet-25k [17] 上训练 $\mathcal{M}_{SV}$。对于在线感知任务，我们将基于记忆的适配器零初始化后，插入到 $\mathcal{M}_{SV}$ 中。接着，我们在 ScanNet 的 RGB-D 视频上训练新的模型。为了减少内存开销，我们在每次迭代中从每个场景中随机采样 8 个相邻的 RGB-D 帧。我们将基于记忆的适配器插入到主干网络和颈部网络之间。对于输出多层级特征的主干网络，我们将不同的适配器插入到不同的层级中。超参数设置如下：$l = 50, s = 2.5, \tau = 8, \delta = 0.03$。我们直接使用原始论文中针对离线训练设计的优化器配置来训练这些模型。

对于预测融合，不同任务采用不同的策略。**语义分割**: 将每一帧的预测结果拼接起来，拼接后的点数与 S_t 保持一致。我们采用 $2cm$ 的体素化方法，通过通道维度的最大池化操作统一位于相同体素网格内的点的预测结果。**目标检测**: 每帧的预测边界框通过 3D 非极大值抑制（NMS）进行合并。当不同时间帧的两个边界框在 NMS 过程中发生冲突时，我们为较新帧中的边界框的分类得分增加 δ。这是因为我们的方法可以确保主干网

络对较新帧提取更完整的几何特征。**实例分割**: 实例分割方法可以分为基于 transformer 的方法 [5]、基于分组的方式 [6, 24] 和基于检测的方式 [21, 23, 25]。前两类方法需要专门设计的掩码融合策略 (如 [14]) 以支持在线感知，因此我们选择基于检测的方式。该方法首先进行在线 3D 目标检测，然后利用边界框裁剪并分割存储在记忆中的点云特征。

表 3　不同 3D 场景 s 感知方法在 ScanNet 在线基准测试上的性能表现。我们分别报告语义分割、目标检测和实例分割任务的 mIoU / mAcc、mAP@25 / mAP@50，以及 mAP@25 / mAP@50 指标。

	Method	Type	Number of Sequence			Length of Sequence		
			1	5	10	5	10	15
Semseg	MkNet	Offline	63.7 / 73.5	62.7 / 72.8	58.9 / 69.4	59.3 / 69.8	63.0 / 73.0	63.5 / 73.7
	Fs-A	Online	62.0 / 72.8	60.6 / 71.7	60.0 / 71.3	60.1 / 71.3	60.7 / 71.8	61.0 / 72.0
	MkNet-SV	Online	63.3 / 74.3	63.3 / 74.3	63.3 / 74.3	63.3 / 74.3	63.3 / 74.3	63.3 / 74.3
	MkNet-SV+Ours	Online	**69.1** / **82.2**	**66.8** / **80.0**	**65.9** / **79.2**	**65.9** / **79.3**	**66.8** / **80.1**	**67.1** / **80.4**
Detection	FCAF3D	Offline	57.0 / 40.6	41.1 / 25.2	34.6 / 19.3	28.4 / 15.2	33.9 / 19.4	37.7 / 22.8
	AnyView	Online	60.4 / 36.0	48.8 / 25.3	43.1 / 20.5	36.6 / 16.5	42.0 / 20.7	45.6 / 23.8
	FCAF3D-SV	Online	41.9 / 20.6	29.8 / 13.3	27.0 / 11.5	24.4 / 10.1	26.2 / 11.0	27.6 / 12.1
	FCAF3D-SV+Ours	Online	**70.5** / **49.9**	**58.7** / **37.7**	**56.2** / **34.3**	**53.1** / **31.2**	**54.9** / **33.8**	**56.1** / **35.6**
Insseg	TD3D	Offline	64.0 / 50.8	61.6 / 49.7	59.4 / 48.4	59.0 / 47.9	61.4 / 49.8	61.7 / 49.8
	TD3D-SV	Online	53.7 / 36.8	54.2 / 41.6	57.0 / 46.3	56.4 / 45.5	53.9 / 40.9	52.6 / 39.5
	TD3D-SV+Ours	Online	**71.3** / **60.5**	**64.7** / **55.2**	**64.2** / **55.0**	**64.0** / **54.7**	**64.6** / **55.1**	**63.9** / **54.3**

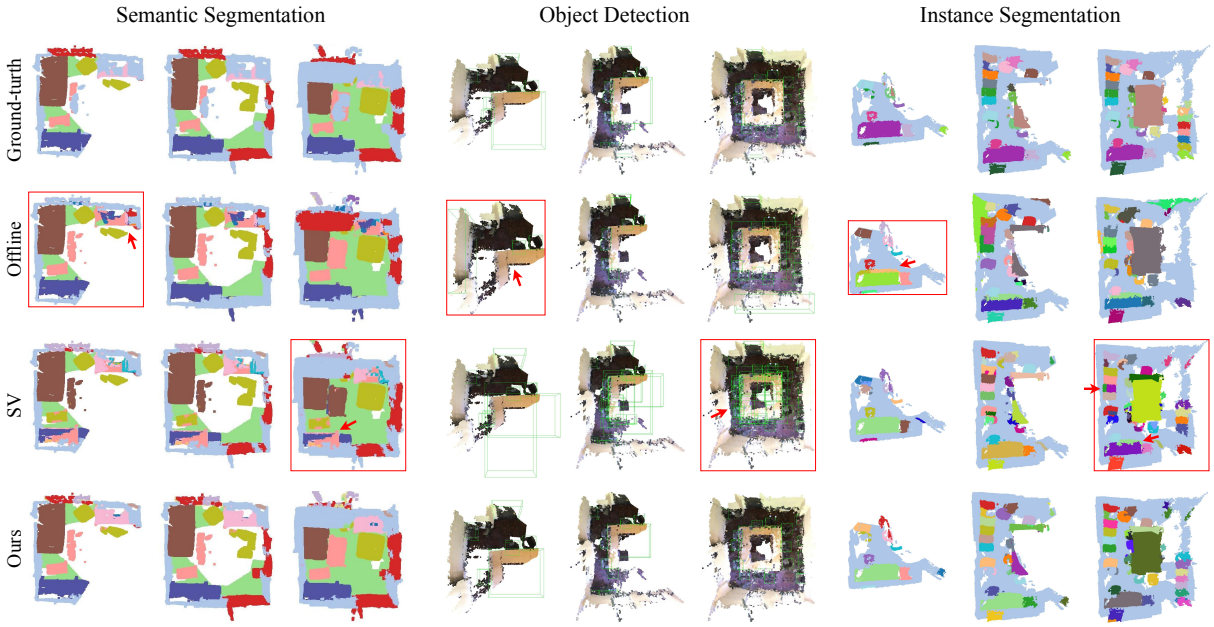

图 1　在线基准测试上的可视化结果。我们的预测在帧数变化时既准确又具有鲁棒性。请注意，由于存在噪声的 2D 标注，一些真实值掩码是不完整的，在这种情况下，我们的预测比真实值更加合理。

4.2 与最新技术的比较

我们将我们的方法与性能最佳的离线和在线 3D 感知模型进行了比较。离线模型指的是第 3.1 节中描述的 $\mathcal{M}_{Rec}$，该模型在重建的点云上进行训练。带有后缀 “-SV” 的模型指的是 $\mathcal{M}_{SV}$，该模型在单视角 RGB-D 图像上进行训练。

房间级别基准测试： 默认情况下，离线方法使用重建的点云作为输入，而在线方法使用带位姿的 RGB-D 视频，不进行 3D 重建。特殊情况用 † 标记。需要注意的是，与离线方法相比，在线方法面临更大的挑战，因为离线方法直接处理完整且干净的房间 3D 几何信息，而在线方法则需要处理部分且噪声较多的帧。根据表 1 和表 2，通过简单地将基于记忆的适配器插入到 $\mathcal{MSV}$ 中，我们显著提升了其在完整场景上的准确性，并在各任务中相较于为每个任务专门设计的最先进在线 3D 场景感知模型取得了更好的性能。我们观察到，在 3D 目标检测和实例分割任务中，相较于 $\mathcal{MSV}$ 的提升尤其显著。这是因为这些任务要求对每个物体生成完整的预测，而使用单帧 RGB-D 数据通常很难推断出大型物体的整体几何信息。

我们还注意到，在语义分割任务上，我们的方法甚至优于离线方法。由于语义分割任务更需要对局部几何的细致感知，而非全局上下文，我们的方法即使使用部分且噪声较多的输入，也能生成更精细的分割结果。

在线基准测试： 在该基准测试中，所有方法的输入均为带位姿的 RGB-D 序列。对于离线方法，我们将每个序列中的 RGB-D 帧的点云拼接成一个整体。由于 INS-Conv 的代码无法获取，我们未与其在该基准测试中进行比较。根据表 3，离线方法在处理部分且噪声较多的场景时表现出较差的泛化能力，尤其是当输入序列较短时。需要注意的是，离线方法在每个时间点都会处理完整的观察场景 S_t。在处理 S_{t+1} 时，对 S_t 提取的特征被浪费掉了。相比之下，在线方法在每个时间点只处理单帧数据 x_t，并融合每帧的预测结果，这在实时机器人任务中更加高效和实用。通过配备基于记忆的适配器，$\mathcal{MSV}$ 在所有任务和实验设置中均取得了相较于其他离线和在线方法的最佳性能。我们还观察到，输入序列越长，相较于 $\mathcal{MSV}$ 的提升越大，这验证了我们的模块可以有效聚合长期时间信息。我们在图 1 中可视化了不同方法的预测结果。可以看出，由于具备时间建模能力，我们的方法相比 $\mathcal{M}_{SV}$ 更加准确，同时在帧数变化时比离线方法更具鲁棒性。

4.3 消融实验

我们首先在 ScanNet 数据集上的 3D 语义分割任务中，分析了两种基于记忆的适配器的设计选择。此外，我们进一步展示了在微调适配器时，固定图像和点云主干网络的情况下，我们方法的性能表现。

表 4　关于点云和图像模块的消融实验结果。在 ScanNet 数据集上报告语义分割结果。图像模块的性能基于点云模块。

方法 (Method)	mIoU	mAcc
移除残差连接 (Remove residual connection)	64.6	77.9
随机初始化 (Random initialization)	66.2	78.6
移除体素最大池化 (Remove voxel maxpooling)	64.8	76.1
设置放缩因子 $s = 1$	65.3	78.4
设置放缩因子 $s = 5$	66.8	79.3
插入在颈部后 (Insert after neck)	66.0	78.8
最终点云模块 (Final point cloud module)	**66.9**	**79.3**
移除残差连接 (Remove residual connection)	67.1	79.8
随机初始化 (Random initialization)	68.7	81.7
设置通道移位比例 $\tau = 4$ (Set shift ratio $\tau = 4$)	68.9	82.1
设置通道移位比例 $\tau = 16$ (Set shift ratio $\tau = 16$)	68.7	81.9
移除 3D-to-2D 适配器 (Remove 3D to 2D adapter)	68.0	80.8
插入在颈部后 (Insert after neck)	68.4	81.6
最终图像模块 (Final image module)	**69.1**	**82.2**

点云和图像模块： 表 4 验证了我们设计的有效性。我们观察到，移除体素最大池化会显著降低性能，这表明更新记忆信息的重要性。随着参数 s 的增加，性能最初提升，随后趋于稳定，甚至略有下降。这表明邻域上下文信息对于时间学习非常重要，但过大的邻域体素集合会带来冗余特征。此外，较大的 s 值还会增加计算开销，因此我们选择 $s = 2.5$，以实现最佳的准确性与计算成本平衡。我们观察到参数 τ 的影响与 s 类似，因此选择适当的值对于获得高准确性和减少内存占用同样重要。这些实验还验证了“适配器范式”的有效性，该范式包括残差连接、零初始化以及在主干网络后插入适配器等设计。

表 5　在微调时固定图像和点云主干网络情况下，基于记忆的适配器的效果分析。

方法 (Method)	MkNet	FCAF3D	TD3D
固定图像模块 (Fix I)	69.1 / 82.2	70.5 / 49.9	71.3 / 60.5
固定点云和图像模块 (Fix P & I)	67.3 / 79.9	66.4 / 47.1	69.1 / 58.2

固定主干网络： 在微调我们的适配器时，我们固定了图像主干网络，仅微调其他参数。此外，我们还研究了在图像和点云主干网络均固定的情况下，我们方法的效果。如表 5 所示，即使在固定图像和点云主干网络的情况下，我们的方法仍然在所有三个在线任务中达到了最先进的性能。在这种情况下，我们可以进一步减少内存占用和训练时间，为用户提供更高效的准确性与效率权衡选项。

五、 部署复现

部署复现步骤主要过程按照原项目 Memory-based Adapters for Online 3D Scene Perception(github)给出的教程进行，但也遇到了不少问题，主要包括环境配置，数据集申请与下载，和源代码相关问题。将在下文中进行相关的详细分析。复现后的代码仓库地址为：https://github.com/Neo-0-Gu/Online3D。

5.1 环境配置

在配置 python 环境过程中，主要会遇到 python 软件包不匹配的状况。首先解决 cuda 版本和 pytorch 的匹配。然后，在下载后续的多个 python 库时，例如 mmdet3d，mmcv-full，和 mmsegmentation 在版本迭代后的相关矛盾。

表6是我自行搜索后的，软件包版本匹配表，主要涉及 mmdetection3D 插件的相关依托，以及与 cuda 版本的适配情况。原项目采用的 cuda 版本是 11.7，我复现时则采用 11.6，会有些许的版本区别。

MMDet3D 版本	PyTorch 版本	MMCV 版本	CUDA 版本	MinkowskiEngine 版本
1.1.0	1.10.0 - 1.12.1	1.4.0 - 1.7.1	10.2, 11.x	0.5.4
1.0.0	1.8.0 - 1.10.0	1.3.0 - 1.5.0	10.2, 11.0	0.5.4
0.18.0	1.7.0 - 1.8.1	1.3.0 - 1.4.0	10.1, 10.2	0.5.3
0.17.0	1.6.0 - 1.7.0	1.2.5 - 1.3.0	10.1, 10.2	0.5.2

表 6　版本匹配表

在下载 MinkowskiEngine 时，因为需要用到”install-option” 指令，而 pip 在 2020 年的 20.3 版本发布后，因为基于 pyproject.toml ，从而不再支持”–install-option”，故而在复现时，需要将 pip 进行版本降级。对于 MinkowskiEngine 的下载，需要将”blas_include_dirs” 更改为服务器中的 conda 所在文件夹路径，其中必须包含 openlab 相关库。

5.2 数据集申请与下载

该篇文章主要涉及室内感知工作，所以使用了 ScanNet 数据集，但 ScanNet 数据集的唯一正规获取方式，需要填写”ScanNet Terms of Use”申请表，并发送至官方邮箱来获取，但在申请过后，我并没有得到官方的下载许可回信，所以我参考了网上流出的下载 python 文件, 详见scannnet 数据集下载，这里可能涉及到侵权问题，此代码可见我 fork 原项目后的github 仓库中的 download-scannet.py.

原项目涉及到数据集为 ScanNet-SV,ScanNet-MV 和 Scenenn-MV 数据集，大小约为 280G 左右，所以在处理时候，需要一个较大的磁盘存储空间。此外，在下载点云数据集时，可通过修改传入 download-scannet.py 的终端参数来控制下载的具体文件，例如运行”python download_scannet.py -o path(存放数据集的路径) –type_vh_clean_2.ply” 下载干净的语义标注，其余同理。

5.3 源代码问题

在运行源代码时，会遇到不少问题，例如有关 pointnet2 库的 module 缺失。因为 pytorch 版本的更新，所以需要将”AT_CHECK” 修改为”TORCH_CHECK”, 此外需要将 pointnet2 中的 setup.py 的 headers 从相对路径修改为绝对路径, 以适应工作文件夹路径的更改。至于一些软件包不在 requirements.txt, 从而导致软件包缺失的情况，包括”fvcore”, ”scipy” 等. 最终主要软件包配置, 可见7.

5.4 模型运行和相关结果

在配置完环境和预处理好数据集后，开始进行模型训练和评估。参考原项目，使用”MinkUNet” 作为主干神经网络，以下，分别从在线 3D 语义分割，在线 3D 目标检测，

软件包	版本	软件包	版本
blas	1.0	h5py	3.9.0
icecream	2.1.3	matplotlib	3.6.3
mmcv-full	1.6.0	mmdet	2.28.1
mmdet3d	1.0.0rc6	mmengine	0.10.5
mmsegmentation	0.30.0	cuda	11.6
pytorch	1.12.1	MinkowskiEngine	0.5.4
Python	3.8	scikit-learn	1.2.2

表 7　软件包及对应版本

在线 3D 实例分割任务进行结果复现。

5.5 在线 3D 语义分割

语义分割任务, 参考原项目, 采用 MinkUNet 作为主干神经网络, 首先在 ScanNet-SV 数据集上进行不带适配器 (adapters) 的模型训练和模型评估，而后加入适配器（adapters）, 在 ScanNet-MV 数据集上，进行微调与评估。同时，也在 sceneNN-MV 数据集上进行评估。最终的复现结果比较，详见表8。

表 8　语义分割：原论文和复现结果比较

Method	Type	ScanNet		SceneNN	
		mIoU	mAcc	mIoU	mAcc
MkNet-SV(原论文)	Online	**72.7**	**84.1**	**56.7**	**70.1**
MkNet-SV(复现)	Online	**71.4**	**83.2**	**56.2**	**69.8**

5.6 在线 3D 目标检测

目标检测任务，参考原项目，采用 FCAF3D 作为主干神经网络，同上，首先在 ScanNet-SV 上进行训练和评估，而后加入适配器，在 ScanNet-MV 上进行微调和评估。

5.7 在线 3D 实例分割

在线 3D 实例分割任务，则采用 TD3D 作为主干神经网络，同样先在 ScanNet-SV 上进行训练和预训练，而后加入适配器，在 ScanNet-MV 上进行微调和评估。

表 9　3D 目标检测和实例分割：原论文和复现结果比较

Detection			Insseg		
Method	mAP		Method	mAP	
	@25	@50		@25	@50
FCAF3D-SV(原论文)	**70.5**	**49.9**	TD3D-SV(原论文)	**71.3**	**60.5**
FCAF3D-SV(复现)	**70.3**	**47.1**	TD3D-SV(复现)	**70.8**	**60.2**

六、 模型的评价与思考

6.1 模型的优点

• 适用性强：即插即用，可以直接用于扩展离线模型到在线场景感知任务，无需重新进行特定的模型或任务设计。

• 计算和存储效率高：采用体素化（voxelization）存储点云特征，降低了存储和计算开销；提出了点云到图像的适配器（3D-to-2D adapter），利用点云的全局上下文增强了图像特征。

6.2 模型的缺点

• 全局信息有限：在内存和性能的平衡中，为了缩减内存占用，使用了短队列设计，图像特征的全局上下文提取能力有限，在某些特定场景（如 RGB-D 相机移动速度较快时）性能可能下降。

• 插入复杂性：尽管适配器设计高效，但在模型微调阶段需要同时优化多个模块，可能增加训练复杂性。

6.3 思考

时间信息高效利用: 当前模型通过短队列机制和体素化存储点云特征，有效利用了时间信息，但在处理长期场景时（如机器人在较大空间中探索）可能存在信息丢失或冗余。可以考虑结合动态记忆机制（如 Transformer 中的注意力机制），动态调整存储的特征数量和权重，以选择性保留最关键的时间信息，从而减少冗余存储和计算。

微调: 可以引入高效微调技术（如 LoRA 或参数高效微调方法），以更少的计算开销在适配器中优化关键参数？这不仅能降低硬件要求，还可以缩短微调时间。

参考文献

[1] QI C R, SU H, MO K, et al. Pointnet: Deep learning on point sets for 3d classification and segmentation[C]//CVPR. [S.l.: s.n.], 2017: 652-660.

[2] CHOY C, GWAK J, SAVARESE S. 4d spatio-temporal convnets: Minkowski convolutional neural networks[C]//CVPR. [S.l.: s.n.], 2019: 3075-3084.

[3] RUKHOVICH D, VORONTSOVA A, KONUSHIN A. Fcaf3d: fully convolutional anchor-free 3d object detection[C]//ECCV. [S.l.]: Springer, 2022: 477-493.

[4] WANG H, DING L, DONG S, et al. Cagroup3d: Class-aware grouping for 3d object detection on point clouds[J]. arXiv preprint arXiv:2210.04264, 2022.

[5] SCHULT J, ENGELMANN F, HERMANS A, et al. Mask3d for 3d semantic instance segmentation[J]. arXiv preprint arXiv:2210.03105, 2022.

[6] VU T, KIM K, LUU T M, et al. Softgroup for 3d instance segmentation on point clouds [C]//CVPR. [S.l.: s.n.], 2022: 2708-2717.

[7] CHAPLOT D S, GANDHI D P, GUPTA A, et al. Object goal navigation using goal-oriented semantic exploration[J]. NeurIPS, 2020, 33:4247-4258.

[8] ZHANG J, DAI L, MENG F, et al. 3d-aware object goal navigation via simultaneous exploration and identification[C]//CVPR. [S.l.: s.n.], 2023: 6672-6682.

[9] MOUSAVIAN A, EPPNER C, FOX D. 6-dof graspnet: Variational grasp generation for object manipulation[C]//ICCV. [S.l.: s.n.], 2019: 2901-2910.

[10] MCCORMAC J, HANDA A, DAVISON A, et al. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks[C]//ICRA. [S.l.]: IEEE, 2017: 4628-4635.

[11] NARITA G, SENO T, ISHIKAWA T, et al. Panopticfusion: Online volumetric semantic mapping at the level of stuff and things[C]//IROS. [S.l.]: IEEE, 2019: 4205-4212.

[12] ZHANG J, ZHU C, ZHENG L, et al. Fusion-aware point convolution for online semantic 3d scene segmentation[C]//CVPR. [S.l.: s.n.], 2020: 4534-4543.

[13] HUANG S S, MA Z Y, MU T J, et al. Supervoxel convolution for online 3d semantic segmentation[J]. TOG, 2021, 40(3):1-15.

[14] LIU L, ZHENG T, LIN Y J, et al. Ins-conv: Incremental sparse convolution for online 3d segmentation[C]//CVPR. [S.l.: s.n.], 2022: 18975-18984.

[15] PAN J, LIN Z, ZHU X, et al. St-adapter: Parameter-efficient image-to-video transfer learning[J]. NeurIPS, 2022, 35:26462-26477.

[16] CHEN Z, DUAN Y, WANG W, et al. Vision transformer adapter for dense predictions[J]. arXiv preprint arXiv:2205.08534, 2022.

[17] DAI A, CHANG A X, SAVVA M, et al. Scannet: Richly-annotated 3d reconstructions of indoor scenes[C]//CVPR. [S.l.: s.n.], 2017: 5828–-5839.

[18] HUA B S, PHAM Q H, NGUYEN D T, et al. Scenenn: A scene meshes dataset with annotations[C]//3DV. [S.l.: s.n.], 2016: 92-101.

[19] QI C R, YI L, SU H, et al. Pointnet++: Deep hierarchical feature learning on point sets in a metric space[C]//NeurIPS. [S.l.: s.n.], 2017: 5099-5108.

[20] GRAHAM B, ENGELCKE M, VAN DER MAATEN L. 3d semantic segmentation with submanifold sparse convolutional networks[C]//CVPR. [S.l.: s.n.], 2018: 9224-9232.

[21] HOU J, DAI A, NIESSNER M. 3d-sis: 3d semantic instance segmentation of rgb-d scans [C]//CVPR. [S.l.: s.n.], 2019: 4421-4430.

[22] QI C R, LITANY O, HE K, et al. Deep hough voting for 3d object detection in point clouds[C]//ICCV. [S.l.: s.n.], 2019: 9277-9286.

[23] YI L, ZHAO W, WANG H, et al. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud[C]//CVPR. [S.l.: s.n.], 2019: 3947-3956.

[24] JIANG L, ZHAO H, SHI S, et al. Pointgroup: Dual-set point grouping for 3d instance segmentation[C]//CVPR. [S.l.: s.n.], 2020: 4867-4876.

[25] KOLODIAZHNYI M, RUKHOVICH D, VORONTSOVA A, et al. Top-down beats bottom-up in 3d instance segmentation[J]. arXiv preprint arXiv:2302.02871, 2023.

[26] CHANG A X, FUNKHOUSER T, GUIBAS L, et al. Shapenet: An information-rich 3d model repository[J]. arXiv preprint arXiv:1512.03012, 2015.

[27] QI C R, SU H, NIESSNER M, et al. Volumetric and multi-view cnns for object classification on 3d data[C]//CVPR. [S.l.: s.n.], 2016: 5648-5656.

[28] LIU Z, TANG H, LIN Y, et al. Point-voxel cnn for efficient 3d deep learning[C]//NeurIPS. [S.l.: s.n.], 2019: 963-973.

[29] GRAHAM B. Spatially-sparse convolutional neural networks[J]. arXiv preprint arXiv:1409.6070, 2014.

[30] ENGELCKE M, RAO D, WANG D Z, et al. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks[C]//ICRA. [S.l.: s.n.], 2017: 1355-1361.

[31] CARREIRA J, PATRAUCEAN V, MAZARE L, et al. Massively parallel video networks [C]//ECCV. [S.l.: s.n.], 2018: 649-666.

[32] DAI Z, YANG Z, YANG Y, et al. Transformer-xl: Attentive language models beyond a fixed-length context[J]. arXiv preprint arXiv:1901.02860, 2019.

[33] KONDRATYUK D, YUAN L, LI Y, et al. Movinets: Mobile video networks for efficient video recognition[C]//CVPR. [S.l.: s.n.], 2021: 16020-16030.

[34] OORD A V D, DIELEMAN S, ZEN H, et al. Wavenet: A generative model for raw audio [J]. arXiv preprint arXiv:1609.03499, 2016.

[35] LIN J, GAN C, HAN S. Tsm: Temporal shift module for efficient video understanding [C]//ICCV. [S.l.: s.n.], 2019: 7083-7093.

[36] PARK J, ZHOU Q Y, KOLTUN V. Colored point cloud registration revisited[C]//ICCV. [S.l.: s.n.], 2017: 143-152.

[37] ZHAO X, AGRAWAL H, BATRA D, et al. The surprising effectiveness of visual odometry techniques for embodied pointgoal navigation[C]//ICCV. [S.l.: s.n.], 2021: 16127-16136.

[38] RAMAKRISHNAN S K, CHAPLOT D S, AL-HALAH Z, et al. Poni: Potential functions for objectgoal navigation with interaction-free learning[C]//CVPR. [S.l.: s.n.], 2022: 18890-18900.

[39] ARMENI I, SENER O, ZAMIR A R, et al. 3d semantic parsing of large-scale indoor spaces[C]//ICCV. [S.l.: s.n.], 2016: 1534-1543.

[40] NATHAN SILBERMAN P K, Derek Hoiem, FERGUS R. Indoor segmentation and support inference from rgbd images[C]//ECCV. [S.l.: s.n.], 2012.

[41] SONG S, LICHTENBERG S P, XIAO J. Sun rgb-d: A rgb-d scene understanding benchmark suite[C]//CVPR. [S.l.: s.n.], 2015: 567-576.

[42] RUKHOVICH D, VORONTSOVA A, KONUSHIN A. Tr3d: Towards real-time indoor 3d object detection[J]. arXiv preprint arXiv:2302.02858, 2023.

[43] WU Z, XU X, WANG Z, et al. Anyview: generalizable indoor 3d object detection with variable frames[J]. arXiv preprint arXiv:2310.05346, 2022.

[44] HOU J, XIE S, GRAHAM B, et al. Pri3d: Can 3d priors help 2d representation learning? [C]//ICCV. [S.l.: s.n.], 2021: 5693-5702.

[45] RONNEBERGER O, FISCHER P, BROX T. U-net: Convolutional networks for biomedical image segmentation[C]//MICCAI. [S.l.]: Springer, 2015: 234-241.

[46] REN S, HE K, GIRSHICK R, et al. Faster r-cnn: towards real-time object detection with region proposal networks[J]. TPAMI, 2016, 39(6):1137-1149.

[47] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition[C]//CVPR. [S.l.: s.n.], 2016: 770-778.

[48] LIN T Y, DOLLÁR P, GIRSHICK R, et al. Feature pyramid networks for object detection [C]//CVPR. [S.l.: s.n.], 2017: 2117-2125.

附录 A　相关链接

原论文 arxiv 地址：https://arxiv.org/abs/2403.06974

原项目代码仓库：https://github.com/xuxw98/Online3D

复现代码仓库：https://github.com/xuxw98/Online3D