

Memory-based Adapters for Online 3D Scene Perception

Xiuwei Xu^{1*}, Chong Xia^{1*}, Ziwei Wang², Linqing Zhao³, Yueqi Duan¹, Jie Zhou¹, Jiwen Lu^{1†}
¹Tsinghua University, ²Carnegie Mellon University, ³Tianjin University

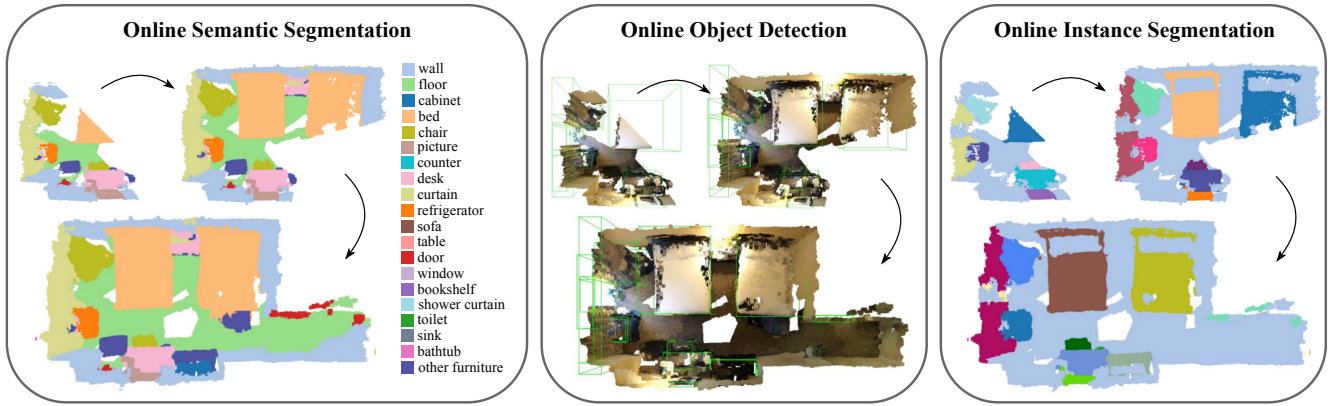


Figure 1. We propose a general framework for online 3D scene perception. With the presented memory-based adapters, we empower existing offline models in different tasks with online perception ability, which is valuable for robotics applications.

Abstract

In this paper, we propose a new framework for online 3D scene perception. Conventional 3D scene perception methods are offline, i.e., take an already reconstructed 3D scene geometry as input, which is not applicable in robotic applications where the input data is streaming RGB-D videos rather than a complete 3D scene reconstructed from pre-collected RGB-D videos. To deal with online 3D scene perception tasks where data collection and perception should be performed simultaneously, the model should be able to process 3D scenes frame by frame and make use of the temporal information. To this end, we propose an adapter-based plug-and-play module for the backbone of 3D scene perception model, which constructs memory to cache and aggregate the extracted RGB-D features to empower offline models with temporal learning ability. Specifically, we propose a queued memory mechanism to cache the supporting point cloud and image features. Then we devise aggregation modules which directly perform on the memory and pass temporal information to current frame. We further propose 3D-to-2D adapter to enhance image features with strong global context. Our adapters can be easily inserted into mainstream offline architectures of different tasks and significantly boost their performance on online tasks. Extensive experiments on ScanNet and SceneNN datasets demon-

strate our approach achieves leading performance on three 3D scene perception tasks compared with state-of-the-art online methods by simply finetuning existing offline models, without any model and task-specific designs. [Project page.](#)

1. Introduction

3D scene perception aims to parse a 3D scene into semantic or object-level entities, mainly including semantic segmentation, object detection and instance segmentation, which is the fundamental ability for robotics or AR/VR applications. Since PointNet [32] proposes the first model that directly process point clouds, great improvement on 3D scene perception [6, 38, 40, 42, 43] has been achieved in recent years by accurate and efficient architecture design.

However, conventional 3D scene perception methods are offline, i.e., they take an already reconstructed 3D scene geometry from pre-collected RGB-D videos without temporal information as input. While in most robotic application such as navigation [4, 47] and manipulation [25] where the agent is usually initialized in an unknown environment, the input data is streaming RGB-D videos and scene perception should be performed synchronously with data collection to guide the agent how to explore. Therefore, online 3D scene perception model with temporal modeling ability is required, which takes in streaming RGB-D video and con-

*Equal contribution. †Corresponding author.

tinuously outputs the perception of the currently observed 3D scene. There are also a few online 3D scene perception methods [16, 22, 24, 26, 46] designed for special architecture and task. As these methods only focus on temporal aggregation for single modality, they cannot fully exploit temporal relations between image and point cloud features and thus their performances are not satisfactory.

In this paper, we propose a new general framework for online 3D scene perception. Different from previous works which design online perception approaches based on specific architecture and task and train models on RGB-D videos from scratch, we convert existing offline 3D perception models to online ones by simply inserting plug-and-play modules and finetuning. Taking inspiration from adapters [5, 29] which adapt image backbones to downstream tasks by additional parameter tuning, we propose memory-based adapters to empower the backbone of 3D perception model with temporal modeling ability by reusing the extracted features from previous frames. Specifically, we propose a queued memory mechanism to cache the supporting point cloud and image features for the RGB-D frame at current time. Based on the structure of memory, we devise aggregation modules which directly operate on the memory and pass temporal information to current frame. As the global context of image features is limited, we further propose 3D-to-2D adapter to enhance image features with 3D memory projection and 2D sparse aggregation. In this way, we can make use of the existing mainstream 3D scene perception model zoo to acquire a series of online models, with simple inserting and finetuning. We conduct extensive experiments on three online perception tasks on ScanNet [7] and SceneNN [15] datasets. Our approach achieves leading performance on all tasks and datasets without any additional loss function and special prediction fusion strategy. To summarize, our contributions include:

- We propose a new framework for online 3D scene perception, which extends existing offline models to online ones by adapter without model and task-specific design.
- We propose general memory-based adapters for image and point cloud backbones, which cache and aggregate extracted features to model the temporal relations between frames.
- Equipped with our adapters, offline models are able to achieve leading performance on three tasks compared with state-of-the-art online models.

2. Related Work

3D Scene Perception: 3D scene perception is widely studied in computer vision, which can be divided into three mainstream tasks: semantic segmentation [6, 11, 32, 33], object detection [13, 34, 38, 43] and instance segmentation [17, 18, 40, 42, 45]. As we focus on the feature extraction of 3D scene in this work, we mainly discuss the

backbone of 3D scene perception networks. Due to the unordered property of point cloud data, voxelizing the points and applying convolution on 3D grids is a natural solution [3, 31]. However, the computational cost and memory requirement both increase cubically with voxel resolution, which is inefficient. PointNet [32] is the pioneer work which directly extracts feature representations from raw point clouds. PointNet++ proposes set abstraction and feature propagation operation based on PointNet, which helps learning more detailed local geometric information. As the furthest point sampling operation in PointNet++ is time consuming, PV-CNN [23] converts point clouds to low-resolution voxels and apply 3D convolution to efficiently aggregate local features. Another way to extract high-quality 3D features is sparse convolution [9, 10], which voxelizes the point clouds but only apply 3D convolution on non-empty voxels. To further improve the efficiency of sparse CNNs, submanifold sparse convolution [6, 11] is introduced, which only conducts convolution when the center of kernel slides over active sites and keeps the same level of sparsity throughout the network. However, these methods are designed for offline 3D scene perception, which is not able to process a streaming RGB-D video at real time.

Streaming Data Analysis: As the input for online 3D scene perception model is streaming RGB-D video, we review the streaming data analysis methods for both image and point cloud domains. In 2D vision, many works [2, 8, 19] extend causal convolution [28] to streaming videos, where a stream buffer is devised to cache previous frames and 3D causal convolution is applied to unidirectionally aggregate spatial-temporal information. TSM [20] utilizes a more efficient shift mechanism, where a proportion of channels of previous image features are shifted to the next frame. Then the spatial-temporal information can be efficiently aggregated by 2D convolution. Our image adapter also utilizes channel shift for efficient temporal modeling. Differently, TSM is trained from scratch where the network can learn how to model temporal information according to the shift proportion. While we reorganize the channels and adopt channel shift in a plug-and-play adapter to empower image backbone with temporal modeling ability. However, as 2D streaming videos contain less helpful information for real world applications like robotic navigation [4, 47] and manipulation [25], increasing attention has been paid to 3D streaming RGB-D video analysis. A natural solution is to first process 2D images and then project the predictions to 3D point clouds, which is followed by a fusion step to merge the predictions from different frames [24, 26]. Fusion-aware 3D-Conv [46] and SVCNN [16] maintain the information of previous frames in 3D space and conduct point-based convolution to fuse the 3D features for semantic segmentation. INS-CONV [22] extends sparse convolution to incremental CNN to efficiently extract global 3D features

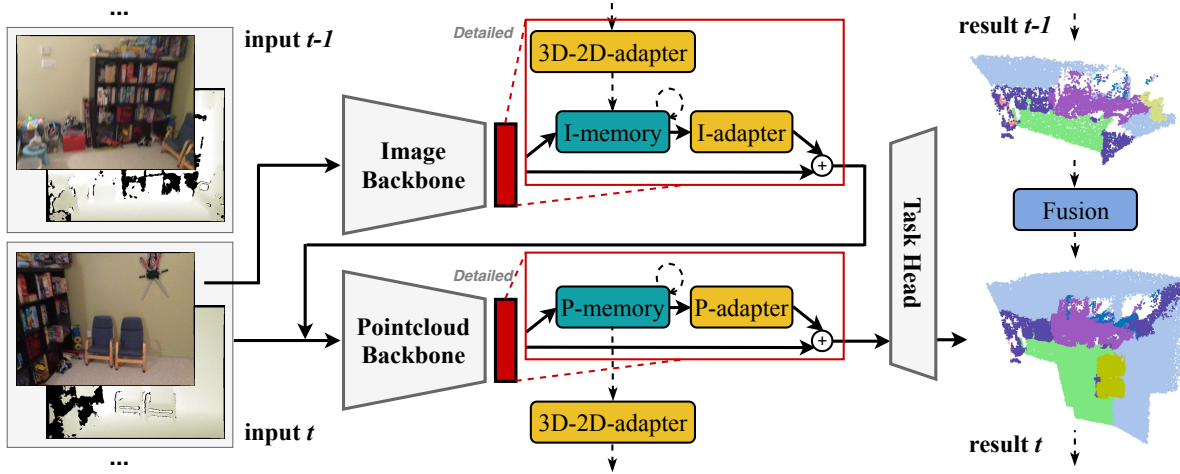


Figure 2. Overall architecture of our approach. We insert memory-based adapters after image and point cloud backbones, which cache the extracted features in memory over time and perform temporal aggregation. 3D-to-2D adapter is proposed to further exploit inter-modal temporal information. Solid lines indicate operations within a single frame, while dashed lines indicate temporal operations.

for semantic and instance segmentation. Differently, our approach empowers offline model with online perception ability by image and point cloud memory-based adapters, which fully exploits the multimodal temporal relations.

3. Approach

In this section, we first introduce the definition of online 3D scene perception and explain our motivation of memory-based adapter. Then we describe how to construct memory and refine the backbone features by adapter for point cloud and image respectively.

3.1. Online 3D Scene Perception

Let $\mathcal{X}_t = \{x_1, x_2, \dots, x_t\}$ be a posed RGB-D streaming video, which means the video is collected with the moving of sensor rather than a pre-collected video. We have:

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

where I_t and P_t refer to the image and point clouds for one RGB-D frame. P_t is acquired by lifting the depth image to world coordinate system with pose parameters M_t , where M_t can be estimated by visual odometry [30, 48]. At time t , the input to online perception model is \mathcal{X}_t and the output is predictions for the observed 3D scene $S_t = \bigcup_{i=1}^t P_i$, which can be bounding boxes and semantic/instance masks. Some works also perform 3D reconstruction along with online perception [22] to acquire high-quality point clouds or meshes, but this is not required [4, 35, 47]. In this work we do not rely on 3D reconstruction and directly take RGB-D streaming video as input, which is a more general setting.

Although great improvement has been achieved in the

design of 3D perception models, most of them only focus on two scenarios: (1) Reconstructed scenes [1, 7]. The model \mathcal{M}_{Rec} is trained on point clouds of complete scenes reconstructed from RGB-D videos. (2) Single-view scenes [27, 41]. The model \mathcal{M}_{SV} is trained on single-view point clouds back-projected from RGB-D image. However, \mathcal{M}_{Rec} requires the input to be a complete scene, which is not accessible in real-time tasks. \mathcal{M}_{SV} is able to process RGB-D videos frame-by-frame, but fails to exploit temporal information. Therefore, previous 3D models are not ready for the more practical online scene perception.

To this end, we aim to devise a plug-and-play temporal learning module, which can be inserted into any single-view perception model \mathcal{M}_{SV} and empowers it with temporal modeling ability. Note that \mathcal{M}_{SV} is originally a 3D perception model. We can extend it to a RGB-D perception model by early fusing the image features to point clouds [39]:

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

where p_t is the prediction for input frame at time t . \mathcal{M}_I is a image backbone pretrained on the same perception task as \mathcal{M}_{SV} . \mathcal{S} projects P_t to image coordinate system by M_t and samples the corresponding 2D features to enrich the features of P_t . We divide \mathcal{M}_{SV} into a backbone \mathcal{M}_P for extracting features of point clouds and a task-specific head \mathcal{M}_H . The goal of this work is to construct an image memory-based adapter for $\mathcal{M}_I(I_t)$ and a point cloud memory-based adapter for $\mathcal{M}_P(P'_t)$ to store and reuse the

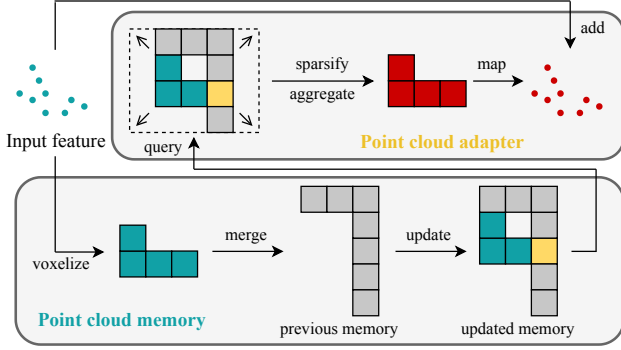


Figure 3. The architecture of the memory-based adapter for point cloud features. We cache and aggregate the features in a queue of 3D voxel grids. Gray, green, yellow and red block refer to previous, current, updated and aggregated voxel features.

extracted backbone features for temporal modeling:

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

where the memory-based adapter \mathcal{A} updates the memory m_t with current features and refines current features by reusing the memory. We fully exploit inter-modal and intra-modal relationships: \mathcal{M}_{SV} fuses image features to point clouds, \mathcal{A}_P reuses previous point cloud features to refine current point cloud features, and \mathcal{A}_I reuses previous features from both modal to refine the current image features. As shown in Figure 2, we follow the same paradigm to design both modules, which can be easily embedded into image and point cloud backbones to achieve temporal modeling.

3.2. Temporal Modeling for Point Clouds

Given $\{\mathcal{M}_P(P'_1), \mathcal{M}_P(P'_2), \dots, \mathcal{M}_P(P'_t)\}$ at time t , i.e., the sequence of point cloud features extracted from the backbone, we aim to enhance the current features $\mathcal{M}_P(P'_t)$ by exploiting the temporal relations within this sequence. Here we first construct a memory to efficiently cache the point cloud features from different timestamp. Then we aggregate the temporal information from the memory to features at current time t with a plug-and-play adapter.

Memory construction: The temporal information for 3D scenes is reflected in a more complete geometry. As a single RGB-D frame may not contain a complete large objects or high-level scene context, the geometric information from previous frames are important for accurate perception of current frame. Therefore, we can cache the sequence of extracted point cloud features in a shared 3D space. A simple way is to directly store the features in terms of point clouds in the world coordinate system. However, this is inefficient in both storage and computation: (1) as the coordinates of the point cloud are real values, the storage overhead will keep growing even if the RGB-D camera is not

moving; (2) the number of points will be very large over time, so point-based sampling and feature aggregate take up high computation overhead. To this end, we propose to store the features in a quantized coordinate system, where point clouds are voxelized and stored in 3D grids. We also maintain the voxels in a queue to reduce the memory footprint when the scene is too large. Specifically, $\mathcal{M}_P(P'_t)$ is first voxelized into the voxel grids V_t by averaging all features whose coordinates falls into the same grid. These voxels are tagged with timestamp t . Then we merge V_t to the memory m_{t-1}^P by maxpooling:

$$\begin{aligned} m_t^P &= \text{maxpooling}(V_t, m_{t-1}^P), \\ m_t^P &= \text{deq}(m_t^P, l) \text{ if } N(m_t^P) > N_{max} \end{aligned} \quad (4)$$

where maxpooling refers to channel-wise maxpooling conducted on each voxel grid, which will update both features and timestamps. $\text{deq}(\cdot, l)$ means removing voxels with timestamp earlier than $t - l + 1$ from the memory. $N(m_t^P)$ is the number of voxels in the memory. We utilize maxpooling as it preserves the most discriminative features over time, which is also efficient to compute as only features at time $t - 1$ and t are required.

Memory-based adapter: After caching and updating the point cloud features in voxels, we need to make use of m_t^P to enhance $\mathcal{M}_P(P'_t)$ with temporal information. To exploit rich scene context from the memory m_t^P while reduce redundant computation, we first use the coordinates of V_t to query a neighbor voxel set:

$$\mathcal{N}(V_t) = \{m_t^P[x][y][z] | (x, y, z) \in s * \mathcal{B}(V_t)\} \quad (5)$$

where $\mathcal{N}(V_t)$ is the queired neighborhood of V_t . $m_t^P[x][y][z]$ refers to the voxel in m_t^P at coordinate (x, y, z) . \mathcal{B} is the minimum enclosing axis-aligned bounding box of V_t and s is a scaling factor to enlarge the size of box. In this way, the temporal information which provides supporting geometric information for current frame is collected into this voxel set.

We then convert $\mathcal{N}(V_t)$ to a sparse tensor [6, 11], which is followed by a 3D sparse convolutional module \mathcal{A}_P to aggregate the context information within $\mathcal{N}(V_t)$ to locations of V_t . Finally we update $\mathcal{M}_P(P'_t)$ in an adapter-manner: (1) We map the aggregated features back to the coordinates of $\mathcal{M}_P(P'_t)$ and then add it to the original features with residual connection; (2) The adapter module \mathcal{A}_P is zero-initialized. Therefore after inserting the adapter, finetuning will smoothly start from the original point.

3.3. Temporal Modeling for Images

For the sequence of image features $\{\mathcal{M}_I(I_1), \dots, \mathcal{M}_I(I_t)\}$ at time t , we follow the same paradigm as the point cloud counterpart to store it with a memory and aggregate temporal information to current frame $\mathcal{M}_I(I_t)$ with an adapter.

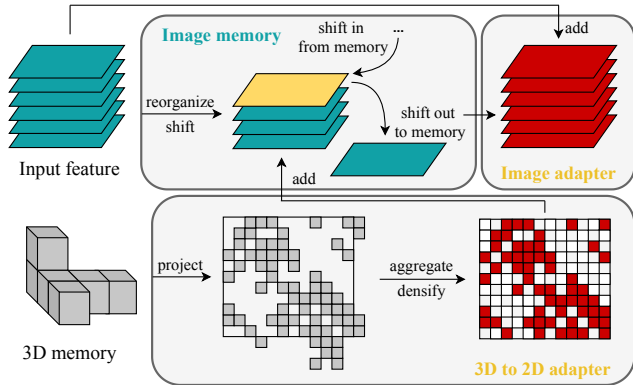


Figure 4. The architecture of the memory-based adapter for image features. We reorganize the input features and shift out a proportion of channels into the memory, while shifting in previous memory and aggregating temporal information by 2D convolution. We also resort to the 3D memory for more global context.

Memory construction: Different from 3D data where point clouds from different timestamps can be stored in a shared 3D space, for 2D data the image features are stacked into a streaming video. A common practice to process this kind of data [19] is to maintain a queue and perform causal convolution (unidirectional along temporal dimension) to aggregate the information from previous frames to current image features. However, video analysis methods focus on extracting the global information of the streaming video up to now, while in our case we only need to enhance the current image features $\mathcal{M}_I(I_t)$. So causal convolution on previous frames will bring a large amount of redundant computation. Moreover, in online 3D scene perception, the most important information for image features is the observation of objects from multiple perspectives. Since an object is usually observed in a few adjacent frames, maintaining a short queue is enough in most cases. To this end, we make an extreme simplification: we only store one frame of previous image and exploit temporal relations between frames at time $t-1$ and t . In order to efficiently aggregate the adjacent frames, we adopt channel shift to cache the temporal information. Formally, given $\mathcal{M}_I(I_t) \in \mathbb{R}^{H \times W \times C}$, we learn a linear transformation $R_1 \in \mathbb{R}^{C \times C'}$ to map the image features into another embedding space, where the first $\frac{1}{\tau}$ channels contain rich temporal information relevant to the next frame. Therefore the memory can be simply constructed by shifting out this part of channels:

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

Note that this operation is repeated frame-by-frame, thus m_{t-1}^I contains the temporal information relevant with current features $\mathcal{M}_I(I_t)$.

Memory-based adapter: At time t , after shifting out a proportion of channels to memory, we can pad the empty

channels with previous memory m_{t-1}^I . In this way, the temporal information of the adjacent two frames are merged into a single frame, for which we can directly adopt 2D convolution to aggregate the features:

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

where $R_2 \in \mathbb{R}^{C' \times C}$ is a learnable inverse transformation to map image features back to the original embedding space. Finally we update $\mathcal{M}_I(I_t)$ by adding F_t with a residual connection. We also zero-initialize R_1 , R_2 and 2D-Conv for smooth finetuning.

3.4. Inter-modal Temporal Modeling

Although maintaining a short queue and adopting channel shift are able to effectively aggregate temporal information for image features, the global context is limited. This will lead to a performance drop when an object is very large or the RGB-D camera stops moving. However, as analyzed before, extracting rich global context from a streaming video is really memory and computation consuming. To solve this problem, we resort to the point cloud memory for global context extraction, because the point cloud features are efficiently cached in a shared 3D space and thus the length of queue can be long. We devise a 3D-to-2D adapter to refine current image features with the global 3D features. Here we first project m_{t-1}^P to the discrete image coordinate system with the inverse function of \mathcal{S} , which are then converted to sparse tensor. In this way, we keep the sparsity of point cloud memory and make the 2D features geometry-aware. 2D sparse convolution is then applied on the sparse tensor to aggregate the context information, followed by a densify operation to keep features inside image and pad other pixels with zero. Finally we add the densified 2D features to enhance $\mathcal{M}_I(I_t) \cdot R_1$ with richer global context. We zero-initialize the 2D sparse convolution for smooth training.

To acquire the final online perception results, a prediction fusion strategy is needed. As the focus of our work is the temporal learning modules for image and point cloud backbones, we only adopt a simple post-processing strategy to fuse the predictions of each frame in a whole, which we detail in Section 4.1.

4. Experiment

In this section, we first describe our datasets and implementation details. Then we compare our method with state-of-the-art methods on both room-level and online benchmarks to comprehensively analyze the advantage of memory-based adapters. Finally we conduct ablation studies to validate the effectiveness of our design.

4.1. Benchmarks and Implementation Details

We evaluate our method on two datasets: ScanNet [7] and SceneNN [15]. ScanNet contains 1513 scanned scene se-

quences, out of which we use 1201 sequences for training and the rest 312 for testing. SceneNN is a smaller dataset which contains 50 high-quality scanned scene sequences with semantic label. After careful filtering, we select 12 clean sequences for testing. We train all models on ScanNet and evaluate them on ScanNet or SceneNN.

Benchmarks: We first compare different methods on room-level benchmarks, i.e., the performance on the reconstructed complete scenes. For semantic segmentation, we compare different methods on ScanNet and SceneNN. Since online methods may not perform 3D reconstruction, we map their predictions on point clouds concatenated from posed RGB-D images to the reconstructed point clouds with nearest neighbor interpolation. For object detection and instance segmentation, the metric is computed on each object rather than the whole point clouds. Therefore we use reconstructed point clouds and RGB-D videos as the inputs for offline and online methods respectively, and calculate metrics based on their respective inputs.

We also follow AnyView [44] to organize an online benchmark on ScanNet for more comprehensive evaluation. We divide the RGB-D video of each room into several non-overlapping sequences and regard each sequence as an independent scene, where the number or the length of each sequence can be set to different values. In this way, we can measure the generalization ability of different methods when the input scenes are incomplete and of variable scales, which is a more practical setting. In our experiments, we divide each room into 1/5/10 sequences or sequences with fixed length 5/10/15, resulting in 6 metrics.

Implementation details: To train \mathcal{M}_{SV} , we first train a 2D perception model \mathcal{M}_I following Pri3D [14]. We use UNet [37] for semantic segmentation and FasterRCNN [36] (only ResNet [12] and FPN [21] backbones are needed) for object detection and instance segmentation. Then we fix the image backbone and train \mathcal{M}_{SV} on ScanNet-25k [7], which is a single-view RGB-D dataset. For online perception, we zero-initialize the memory-based adapters and insert them into \mathcal{M}_{SV} . Then we train the new model on RGB-D videos from ScanNet. To reduce memory footprint, we randomly sample 8 adjacent RGB-D frames for each scene at every iteration. We insert our memory-based adapters between the backbone and neck. For backbones which output multi-level features, we insert different adapters to different levels. In terms of hyperparameters, we set $l = 50$, $s = 2.5$, $\tau = 8$ and $\delta = 0.03$. We simply use the same optimizer configurations to train the models as in their original paper (designed for offline training).

For prediction fusion, we adopt different strategies for different tasks. *Semantic segmentation:* The predictions for each frame are concatenated, which has the same point number with S_t . We use $2cm$ voxelization to unify the predictions for points inside the same voxel grid by channel-

Table 1. 3D semantic segmentation results on ScanNet and SceneNN datasets. For online methods, we map the predictions on point clouds concatenated from posed RGB-D images to the reconstructed point clouds to compare with offline method.

Method	Type	ScanNet		SceneNN	
		mIoU	mAcc	mIoU	mAcc
MkNet [6]	Offline	71.6	80.4	–	–
Fs-A [46]	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

Table 2. 3D object detection and instance segmentation results on ScanNet dataset. Offline and online methods are separated by horizontal line. [†] means INS-Conv requires an additional 3D reconstruction algorithm to acquire high-quality point clouds or meshes.

Detection			Insseg		
Method	mAP		Method	mAP	
	@25	@50		@25	@50
FCAF3D [38]	70.7	56.0	SoftGroup [42]	78.9	67.6
CAGroup3D [43]	74.5	60.3	TD3D [18]	81.3	71.1
AnyView [44]	60.4	36.0	INS-Conv [†] [22]	–	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

wise maxpooling. *Object detection:* The predicted bounding boxes for each frame are merged by 3D NMS. When two boxes of different time are colliding during NMS, we add δ to the classification scores of box in the newer frame. This is because our method ensures the backbone extracts more complete geometric features for the newer frame. *Instance segmentation:* Instance segmentation can be divided into transformer-based [40], grouping-based [17, 42] and detection-based [13, 18, 45]. As the first two kinds require a specially designed mask fusion strategy [22] for online perception, we opt for the detection-based manner, where we can first conduct online 3D object detection and then utilize the boxes to crop and segment the point cloud features stored in the memory.

4.2. Comparison with State-of-the-art

We compare our method with the top-performance offline and online 3D perception models. Offline models refer to \mathcal{M}_{Rec} described in Section 3.1, which is trained on reconstructed point clouds. Models with suffix "-SV" refer to \mathcal{M}_{SV} that is trained on single-view RGB-D images.

Room-level benchmarks: By default, offline methods take in reconstructed point clouds and online methods take in posed RGB-D videos without 3D reconstruction. Special case is denoted by [†]. Note that there is a challenge in

Table 3. The performance of different 3D scene perception methods on ScanNet online benchmark. We report mIoU / mAcc, mAP@25 / mAP@50 and mAP@25 / mAP@50 for semantic segmentation, object detection and instance segmentation respectively.

	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

online methods when compare to offline alternatives, as offline methods directly process the complete and clean 3D geometry of rooms while online methods deal with partial and noisy frames. According to Table 1 and Table 2, by simply inserting the memory-based adapters into \mathcal{M}_{SV} , we significantly boost their accuracy on complete scenes and achieve better performance compared with state-of-the-art online 3D scene perception models specially designed for each task. We observe the improvement upon \mathcal{M}_{SV} is especially significant on 3D object detection and instance segmentation tasks. This is because these tasks require complete predictions for each object, while it is usually very hard to infer the whole geometry of large objects with a single RGB-D frame. We also notice our method even outperforms offline methods on semantic segmentation task. Since this task requires more detailed perception of local geometry rather than the global context, our method can predict finer segmentation with only partial and noisy inputs.

Online benchmark: In this benchmark, the inputs to all methods are the posed RGB-D sequences. We concatenate the point clouds from each RGB-D frame of a sequence into a whole for offline methods. As the code of INS-Conv is not accessible, we do not compare with it on this benchmark. According to Table 3, offline methods show bad generalization ability on partial and noisy scenes, especially when the input sequence is short. Note that offline methods take in the whole observed scene S_t at each time. When processing S_{t+1} , the features extracted for S_t is wasted. On the contrary, online methods process a single frame x_t at each time and fuse the per-frame predictions, which is much more efficient and practical in real-time robotic tasks. Equipped with our memory-based adapters, \mathcal{M}_{SV} achieves the best performance compared with other offline and online methods on all tasks and experimental settings. We observe the longer the input sequence, the larger the improvement upon

Table 4. Ablation study on point cloud and image modules. We report semantic segmentation results on ScanNet. The performance of image module is based on point cloud module.

Method	mIoU	mAcc
Remove residual connection	64.6	77.9
Random initialization	66.2	78.6
Remove voxel maxpooling	64.8	76.1
Set scaling factor $s = 1$	65.3	78.4
Set scaling factor $s = 5$	66.8	79.3
Insert after neck	66.0	78.8
The final point cloud module	66.9	79.3
Remove residual connection	67.1	79.8
Random initialization	68.7	81.7
Set shift ratio $\tau = 4$	68.9	82.1
Set shift ratio $\tau = 16$	68.7	81.9
Remove 3D to 2D adapter	68.0	80.8
Insert after neck	68.4	81.6
The final image module	69.1	82.2

Table 5. Effects of our memory-based adapters when both image and point cloud backbones are fixed during finetuning.

	MkNet	FCAF3D	TD3D
Fix I	69.1 / 82.2	70.5 / 49.9	71.3 / 60.5
Fix P & I	67.3 / 79.9	66.4 / 47.1	69.1 / 58.2

\mathcal{M}_{SV} , which validates our modules can effectively aggregate long-term temporal information.

We visualize the predictions of different methods in Figure 5. It can be seen that our method is more accurate than \mathcal{M}_{SV} due to the temporal modeling ability, and more robust to number of frames than offline methods.

4.3. Ablation Study

We first ablate the design choices of two memory-based adapters on 3D semantic segmentation task on ScanNet.

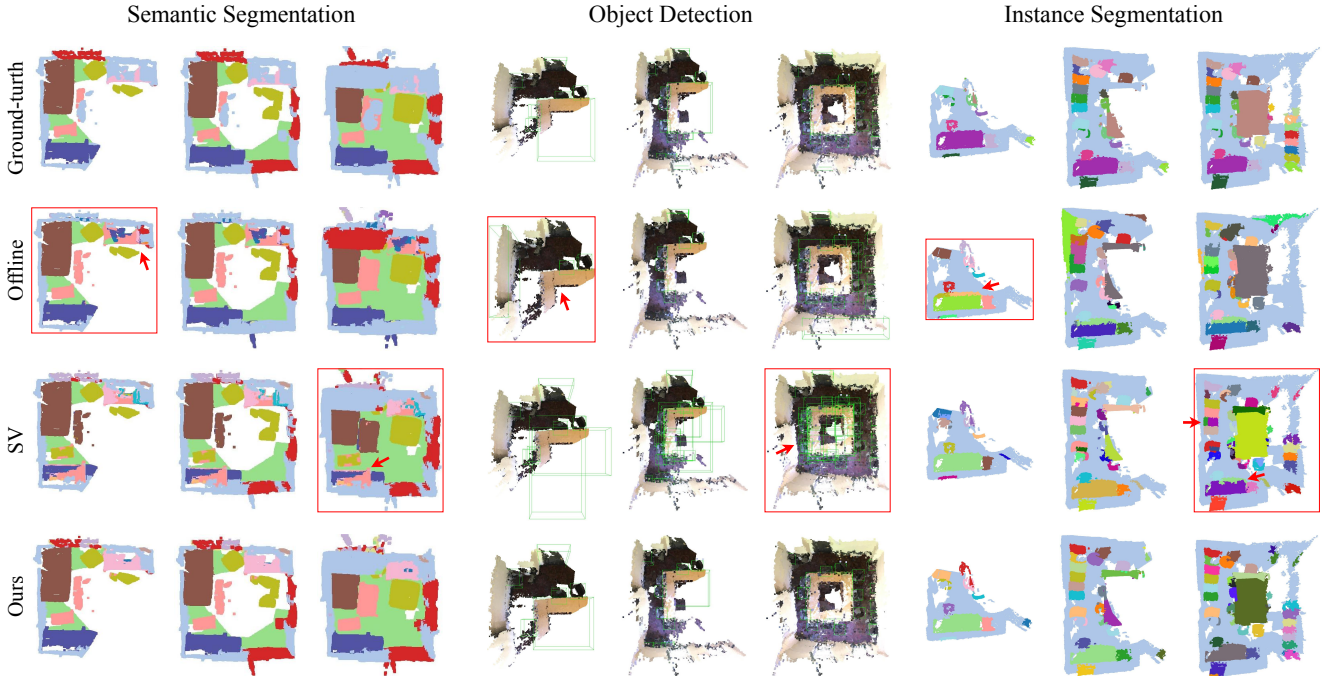


Figure 5. Visualization results on the online benchmark. Our predictions are accurate and robust to the number of frames. Note that some ground-truth masks are incomplete due to the noisy 2D annotations, in this case our predictions are more reasonable than the ground-truths.

Besides, we further show the performance of our method when both image and point cloud backbones are fixed during finetuning the adapters.

Point cloud and image modules: Table 4 validates the effectiveness of our designs. We observe removing voxel maxpooling significantly degrades the performance, which shows the importance of updating memory. With the increase of s , the performance first improves and then keeps steady or even slightly declines, which indicates the neighbor context information is important for temporal learning, but too large neighbor voxel set brings much redundant features. Large s will also increase the computation overhead, so we choose $s = 2.5$ to achieve the best accuracy-computation tradeoff. We observe the influence of τ is similar with s and thus choosing a proper value is important for both high accuracy and less memory storage. From these experiments, we also validate the effectiveness of the 'adapter paradigm', which includes residual connection, zero-initialization and inserting after backbone.

Fixed backbones: When finetuning our adapters, we fix the image backbone and finetune other parameters. We further study the effects of our method when both image and point cloud backbones are fixed. As shown in Table 5, even with both image and point cloud backbones fixed, our method still achieves state-of-the-art performance on all three online tasks. In this way, we can further reduce the memory footprint and training time, which provides the users with more efficiency-accuracy tradeoff.

5. Conclusion

In this paper, we have presented memory-based adapters for online 3D scene perception. Mainstream 3D scene perception methods are offline, which is hard to be applied in most real-time applications where only streaming RGB-D video is accessible. Existing online perception methods design model and task-specific temporal learning approaches, but most of them only focus on temporal aggregation for single modality and thus cannot fully exploit temporal relations between image and point cloud features. To this end, we propose plug-and-play temporal learning modules, which can empower offline methods with online perception ability by simply inserting memory-based adapters and finetuning on RGB-D videos. Specifically, given point cloud and image features extracted from the backbones, we first devise a queued memory mechanism to cache these information over time and maintaining a reasonable storage overhead. Then we devise aggregation modules which directly operate on the memory and pass temporal information from the cached features to current frame. As the global context of image features is limited due to the short queue, we further propose 3D-to-2D adapter to enhance image features with 3D memory. We conduct extensive experiments on ScanNet and SceneNN. By equipping offline models with our modules, we achieve leading performance on three scene perception tasks compared with state-of-the-art online methods, even without any model and task-specific designs.

Supplementary Material

This supplementary material is organized as follows:

- Section A demonstrates the detailed architecture of our baseline models in three tasks and how to insert our adapters into them.
- Section B details the training hyperparameters adopted in our experiments.
- Section C details per-class experimental results.

A. Detailed Architecture

We illustrate the architectures of both image and point cloud backbones and show how to insert the memory-based adapters into them in Figure 6. For online 3D semantic segmentation, we use U-Net [37] as the image backbone and Minkowski-UNet [6] as the point cloud backbone, which is shown in Figure 6 (D) and (C) respectively. For online 3D object detection, we adopt ResNet [12] with FPN [21] as the image backbone and FCAF3D [38] as the point cloud backbone, which is shown in Figure 6 (E) and (B) respectively. For online 3D instance segmentation, we use the same image backbone as the object detection task and adopt TD3D [18] as the point cloud backbone, which is shown in Figure 6 (E) and (A) respectively. Note that for TD3D, the backbone maintains a high-resolution scene representation for ROI-wise instance prediction. We construct a point cloud memory to cache this scene representation, which ensures the point clouds within each ROI are the most complete up to current time. This design helps us acquire complete instance mask by simply performing 3D NMS, which avoids complicated mask fusion strategy [22] to merge instance masks of different frames.

B. Training Hyperparameters

We train the online perception models in two stage. Firstly we train single-view perception model \mathcal{M}_{SV} on ScanNet-25k [7]. Secondly we insert the memory-based adapters into \mathcal{M}_{SV} and finetune the network on ScanNet RGB-D videos.

For online semantic segmentation, we set max epoch as 250, weight decay as 0.01, initial learning rate as 0.0008 and adopt AdamW optimizer with OneCycleLR scheduler for the first stage. Then we set max epoch as 36, weight decay as 0.01, initial learning rate as 0.008 and adopt AdamW optimizer with a stepwise scheduler which steps at 24 and 32 epoch for the second stage.

For online object detection, we set max epoch as 12, weight decay as 0.0001, initial learning rate as 0.001 and adopt AdamW optimizer with a stepwise scheduler which steps at 8 and 11 epoch for the first stage. Then we adopt the same hyperparameters for finetuning in the second stage.

For online instance segmentation, we set max epoch as 33, weight decay as 0.0001, initial learning rate as 0.001 and adopt AdamW optimizer with a stepwise scheduler which

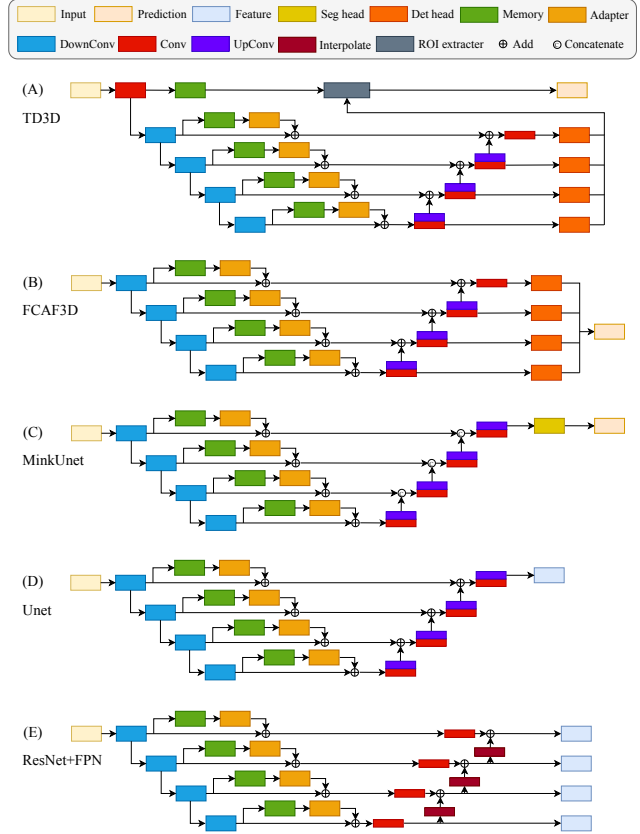


Figure 6. Details about the architectures of image and point cloud backbones and how to insert the adapters into them.

steps at 28 and 32 epoch for the first stage. Then we adopt the same hyperparameters for finetuning.

C. Class-specific Results

We provide class-specific experimental results of our method on three 3D scene perception tasks. Table 6 and 7 show the 3D semantic segmentation results on ScanNet and SceneNN dataset with per-class IoU. Table 8 and 9 show the 3D object detection results on ScanNet dataset with per-class AP₂₅ and AP₅₀. Table 10 and 11 show the 3D object detection results on ScanNet dataset with per-class AP₂₅ and AP₅₀.

References

- [1] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *ICCV*, pages 1534–1543, 2016. 3
- [2] Joao Carreira, Viorica Patraucean, Laurent Mazare, Andrew Zisserman, and Simon Osindero. Massively parallel video networks. In *ECCV*, pages 649–666, 2018. 2
- [3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2
- [4] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *NeurIPS*, 33: 4247–4258, 2020. 1, 2, 3
- [5] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*, 2022. 2
- [6] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, pages 3075–3084, 2019. 1, 2, 4, 6, 9
- [7] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017. 2, 3, 5, 6, 9
- [8] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019. 2
- [9] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *ICRA*, pages 1355–1361, 2017. 2
- [10] Benjamin Graham. Spatially-sparse convolutional neural networks. *arXiv preprint arXiv:1409.6070*, 2014. 2
- [11] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, pages 9224–9232, 2018. 2, 4
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 6, 9
- [13] Ji Hou, Angela Dai, and Matthias Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *CVPR*, pages 4421–4430, 2019. 2, 6
- [14] Ji Hou, Saining Xie, Benjamin Graham, Angela Dai, and Matthias Nießner. Pri3d: Can 3d priors help 2d representation learning? In *ICCV*, pages 5693–5702, 2021. 6
- [15] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *3DV*, pages 92–101, 2016. 2, 5
- [16] Shi-Sheng Huang, Ze-Yu Ma, Tai-Jiang Mu, Hongbo Fu, and Shi-Min Hu. Supervoxel convolution for online 3d semantic segmentation. *TOG*, 40(3):1–15, 2021. 2
- [17] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *CVPR*, pages 4867–4876, 2020. 2, 6
- [18] Maksim Kolodiaznyi, Danila Rukhovich, Anna Vorontsova, and Anton Konushin. Top-down beats bottom-up in 3d instance segmentation. *arXiv preprint arXiv:2302.02871*, 2023. 2, 6, 9
- [19] Dan Kondratyuk, Liangzhe Yuan, Yandong Li, Li Zhang, Mingxing Tan, Matthew Brown, and Boqing Gong. Movinets: Mobile video networks for efficient video recognition. In *CVPR*, pages 16020–16030, 2021. 2, 5
- [20] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, pages 7083–7093, 2019. 2
- [21] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017. 6, 9
- [22] Leyao Liu, Tian Zheng, Yun-Jou Lin, Kai Ni, and Lu Fang. Ins-conv: Incremental sparse convolution for online 3d segmentation. In *CVPR*, pages 18975–18984, 2022. 2, 3, 6, 9
- [23] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *NeurIPS*, pages 963–973, 2019. 2
- [24] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *ICRA*, pages 4628–4635. IEEE, 2017. 2
- [25] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *ICCV*, pages 2901–2910, 2019. 1, 2
- [26] Gaku Narita, Takashi Seno, Tomoya Ishikawa, and Yohsuke Kaji. Panopticfusion: Online volumetric semantic mapping at the level of stuff and things. In *IROS*, pages 4205–4212. IEEE, 2019. 2
- [27] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgb-d images. In *ECCV*, 2012. 3
- [28] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016. 2
- [29] Junting Pan, Ziyi Lin, Xiatian Zhu, Jing Shao, and Hongsheng Li. St-adapter: Parameter-efficient image-to-video transfer learning. *NeurIPS*, 35:26462–26477, 2022. 2
- [30] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Colored point cloud registration revisited. In *ICCV*, pages 143–152, 2017. 3
- [31] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *CVPR*, pages 5648–5656, 2016. 2

Table 6. Per-class 3D semantic segmentation results (IoU) of our method on the ScanNet validation set.

	wall	floor	cabinet	bed	chair	sofa	table	door	window	bookshelf	picture	counter	desk	curtain	fridge	curtain	toilet	sink	bathub	others	mean
Ours	85.7	97.1	63.1	80.7	89.0	76.1	73.7	66.1	63.6	77.1	41.8	65.5	61.2	58.9	61.0	72.7	95.2	77.9	94.2	53.6	72.7

Table 7. Per-class 3D semantic segmentation results (IoU) of our method on the SceneNN validation set.

	wall	floor	cabinet	bed	chair	sofa	table	door	window	bookshelf	picture	counter	desk	curtain	fridge	sink	mean
Ours	75.3	82.6	59.8	82.8	62.0	57.8	18.4	52.4	20.5	55.9	29.4	52.6	44.9	50.2	80.3	81.8	56.7

- [32] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, 2017. 1, 2
- [33] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, pages 5099–5108, 2017. 2
- [34] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, pages 9277–9286, 2019. 2
- [35] Santhosh Kumar Ramakrishnan, Devendra Singh Chaplot, Ziad Al-Halah, Jitendra Malik, and Kristen Grauman. Pon: Potential functions for objectgoal navigation with interaction-free learning. In *CVPR*, pages 18890–18900, 2022. 3
- [36] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *TPAMI*, 39(6):1137–1149, 2016. 6
- [37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015. 6, 9
- [38] Danila Rukhovich, Anna Vorontsova, and Anton Konushin. Fcaf3d: fully convolutional anchor-free 3d object detection. In *ECCV*, pages 477–493. Springer, 2022. 1, 2, 6, 9
- [39] Danila Rukhovich, Anna Vorontsova, and Anton Konushin. Tr3d: Towards real-time indoor 3d object detection. *arXiv preprint arXiv:2302.02858*, 2023. 3
- [40] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3d for 3d semantic instance segmentation. *arXiv preprint arXiv:2210.03105*, 2022. 1, 2, 6
- [41] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, pages 567–576, 2015. 3
- [42] Thang Vu, Kookhoi Kim, Tung M Luu, Thanh Nguyen, and Chang D Yoo. Softgroup for 3d instance segmentation on point clouds. In *CVPR*, pages 2708–2717, 2022. 1, 2, 6
- [43] Haiyang Wang, Lihe Ding, Shaocong Dong, Shaoshuai Shi, Aoxue Li, Jianan Li, Zhenguo Li, and Liwei Wang. Ca-group3d: Class-aware grouping for 3d object detection on point clouds. *arXiv preprint arXiv:2210.04264*, 2022. 1, 2, 6
- [44] Zhenyu Wu, Xiuwei Xu, Ziwei Wang, Chong Xia, Linqing Zhao, Jiwen Lu, and Haibin Yan. Anyview: generalizable indoor 3d object detection with variable frames. *arXiv preprint arXiv:2310.05346*, 2022. 6
- [45] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas J Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. In *CVPR*, pages 3947–3956, 2019. 2, 6
- [46] Jiazhao Zhang, Chenyang Zhu, Lintao Zheng, and Kai Xu. Fusion-aware point convolution for online semantic 3d scene segmentation. In *CVPR*, pages 4534–4543, 2020. 2, 6
- [47] Jiazhao Zhang, Liu Dai, Fanpeng Meng, Qingnan Fan, Xuelin Chen, Kai Xu, and He Wang. 3d-aware object goal navigation via simultaneous exploration and identification. In *CVPR*, pages 6672–6682, 2023. 1, 2, 3
- [48] Xiaoming Zhao, Harsh Agrawal, Dhruv Batra, and Alexander G Schwing. The surprising effectiveness of visual odometry techniques for embodied pointgoal navigation. In *ICCV*, pages 16127–16136, 2021. 3

Table 8. Per-class 3D object detection results (AP_{25}) of our method on the ScanNet validation set.

	cabinet	bed	chair	sofa	table	door	window	bookshelf	picture	counter	desk	curtain	fridge	curtain	toilet	sink	bathtub	others	mean
Ours	55.2	85.4	88.7	87.2	63.3	62.5	47.3	66.2	36.0	65.2	80.1	65.0	58.1	76.3	99.7	76.7	93.3	62.0	70.5

Table 9. Per-class 3D object detection results (AP_{50}) of our method on the ScanNet validation set.

	cabinet	bed	chair	sofa	table	door	window	bookshelf	picture	counter	desk	curtain	fridge	curtain	toilet	sink	bathtub	others	mean
Ours	36.7	75.6	73.9	77.9	57.0	33.8	19.8	43.7	19.4	26.3	62.8	32.4	41.1	24.6	89.2	46.7	84.8	52.2	49.9

Table 10. Per-class 3D instance segmentation results (AP_{25}) of our method on the ScanNet validation set.

	cabinet	bed	chair	sofa	table	door	window	bookshelf	picture	counter	desk	curtain	fridge	curtain	toilet	sink	bathtub	others	mean
Ours	60.3	86.8	91.5	80.3	72.8	56.0	55.3	67.5	45.1	48.9	72.9	68.4	56.5	86.3	99.7	81.3	87.8	65.3	71.3

Table 11. Per-class 3D instance segmentation results (AP_{50}) of our method on the ScanNet validation set.

	cabinet	bed	chair	sofa	table	door	window	bookshelf	picture	counter	desk	curtain	fridge	curtain	toilet	sink	bathtub	others	mean
Ours	50.9	79.1	82.5	71.3	63.6	44.0	36.0	45.5	38.5	30.3	57.3	49.8	52.9	78.9	99.7	66.6	84.9	56.9	60.5