MonoChunk3D: Monocular Online 3D Instance Segmentation with Persistent Instance States

Yibo Zhao[1] **Jingzhi Zhou**[1] **Yigong Zhang**[2] **Jian Yang**[1,2] **Jin Xie**[2*]

[1]College of Computer Science, Nankai University
[2]School of Intelligence Science and Technology, Nanjing University

yibozhao@mail.nankai.edu.cn csjxie@nju.edu.cn

Abstract

Monocular online 3D instance segmentation enables embodied agents to build object-level 3D understanding while continuously exploring their environments. Unlike RGB-D settings, monocular RGB streams lack direct depth and camera pose information, making geometry reconstruction necessary for 3D segmentation. Recent reconstruction foundation models (RFMs) have made this task more feasible by recovering 3D geometry and cross-view cues from monocular images. However, reconstructed geometry remains noisy and incomplete, and streaming observations provide only partial views of object instances, making local observations insufficient for stable instance representations and temporally consistent segmentation. We propose MonoChunk3D, an online 3D instance segmentation framework that incrementally reconstructs, aligns, and segments incoming monocular RGB chunks. Rather than treating reconstructed geometry merely as input, the framework adaptively integrates complementary RFM-derived cross-view priors with 3D geometric features to initialize current-chunk instance queries. To overcome the limitations of local observations, we maintain compact persistent instance states composed of explicit prototypes, implicit query embeddings, and spatial bounds to preserve long-term instance context. These states are propagated as history-aware queries and jointly decoded with current-chunk queries, allowing the preserved instance context to directly guide current-chunk segmentation. The resulting predictions are associated with historical instances to update the persistent states, enabling temporally consistent segmentation over streaming observations. Experiments on ScanNet200, ScanNetV2, and SceneNN show that MonoChunk3D achieves consistent improvements over existing methods in the monocular online setting. Code is available at https://github.com/zybjn/MonoChunk3D.

1 Introduction

Online 3D instance segmentation is a foundational capability for embodied agents and mobile robots, enabling object-level understanding of 3D scenes for downstream tasks such as navigation and manipulation [3, 40, 71]. Unlike offline 3D segmentation on complete scans, online segmentation must process partial observations arriving incrementally while maintaining temporally consistent instance identities. This challenge becomes even greater for monocular RGB streams, where depth and camera poses are not directly available. Thus, the 3D scene must be reconstructed incrementally from the image sequence to support instance segmentation.

Recent advances in vision foundation models (VFMs) have opened new avenues for 3D instance segmentation [22, 47, 43]. Existing VFM-assisted methods typically lift SAM-generated 2D masks

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

into 3D space and obtain 3D segmentation through multi-view association and mask merging [67, 68, 42]. Despite their effectiveness, they remain sensitive to 2D mask quality, rely on complex merging strategies, and are mainly designed for offline scene processing. Recent works have extended this paradigm to online 3D segmentation. ESAM [63] lifts 2D masks into 3D-aware queries for online refinement, while AutoSeg3D [55] formulates online 3D segmentation as instance tracking. However, these methods are primarily developed under posed RGB-D settings, where depth maps and camera poses are typically available. Recent progress in reconstruction foundation models (RFMs) has reduced this reliance by reconstructing geometry and recovering cross-view cues directly from monocular RGB sequences [58, 57, 56, 59]. MoonSeg3R [10] explores monocular online 3D instance segmentation with RFM-reconstructed geometry and VFM mask cues. However, it primarily relies on implicit query-level representations to guide current segmentation, without explicitly using historical spatial priors such as instance extents. Moreover, monocular reconstruction often suffers from noise and missing geometry, while streaming observations can only capture partial views of object instances. Consequently, current observations alone are insufficient for stable segmentation, making it crucial to leverage long-term historical instance priors for temporally consistent 3D instance segmentation.

To address these challenges, we propose MonoChunk3D, a chunk-wise online 3D instance segmentation framework for monocular RGB streams. The framework processes the input stream in chunks, reconstructing each chunk with π^3 [59] and aligning the chunk-wise reconstructions over time with an online Pi-Long adaptation [8, 1]. This online alignment provides temporally consistent geometry for segmentation while preserving online inference. Rather than treating reconstructed geometry merely as input, we further exploit RFM-derived cross-view cues for query initialization. Specifically, guided by VFM-generated 2D masks, we initialize current-chunk instance queries by adaptively aggregating complementary cross-view priors and 3D geometric features. However, current-chunk queries still rely on local observations and can be unreliable when instances are only partially visible or occluded. We therefore maintain compact persistent instance states with explicit prototypes, implicit query embeddings, and spatial bounds to preserve long-term instance context. These states are propagated as history-aware queries and jointly decoded with current-chunk queries, allowing historical context to directly guide current-chunk segmentation. The predictions are associated with historical instances to update the persistent states, enabling temporally consistent segmentation across chunks.

Our contributions are summarized as follows:

1. We propose MonoChunk3D, a chunk-wise online framework for monocular 3D instance segmentation that reconstructs and aligns RGB chunks, while initializing instance queries by adaptively aggregating RFM cross-view priors and 3D geometric features under VFM mask guidance.
2. We introduce compact persistent instance states with explicit prototypes, implicit query embeddings, and spatial bounds to preserve long-term instance context. These states are propagated as history-aware queries to guide current-chunk segmentation and support cross-chunk association.
3. Extensive experiments on ScanNet200, ScanNetV2, and SceneNN demonstrate the effectiveness of our method under the challenging setting of monocular online 3D instance segmentation.

2 Related Work

VFM-assisted 3D Segmentation. Recent vision foundation models (VFMs) pretrained on large-scale 2D data have shown strong generalization in 2D visual understanding [47, 43, 22, 4, 36, 20, 30]. Motivated by the scarcity of high-quality 3D annotations, a growing body of work leverages VFMs for 3D scene segmentation [13, 19, 21, 26, 18, 9, 44, 32, 7]. SAM3D [67] lifts 2D SAM masks into 3D space using depth and camera poses and aggregates them into 3D instances based on geometric cues. SAI3D [68] and Open3DIS [42] build superpoint relations from 2D masks and obtain 3D instances through region growing. OpenMask3D [52] first extracts 3D instance proposals with Mask3D [49] and then projects them to 2D views to obtain open-vocabulary features from VFMs for semantic labeling. Despite their strong performance, these methods generally assume accurate depth and camera poses and are typically designed for offline scene processing.

Online 3D Scene Perception. Existing 3D scene segmentation methods typically rely on offline processing of complete scenes [49, 25, 37, 24, 34, 50, 5, 15, 14, 54]. However, in robotics and embodied AI, scene understanding must be updated continuously as new observations arrive, which has motivated growing interest in online 3D scene perception [39, 41, 17, 70, 27, 69]. Representative methods process streaming inputs through incremental feature fusion, memory modules, or online

aggregation strategies [63, 53, 55, 35, 64]. ESAM [63] lifts 2D masks into 3D-aware queries and performs online refinement and cross-frame merging. OnlineAnySeg [53] performs online zero-shot 3D instance segmentation by efficiently merging lifted 2D masks with voxel hashing. AutoSeg3D [55] reformulates online 3D segmentation as instance tracking, using short-term memory for refining current instance representations and long-term memory for instance association. In contrast, our method maintains compact persistent instance states and propagates them as history-aware queries, allowing long-term instance context to directly guide current-chunk segmentation. Furthermore, these online methods mainly target RGB-D or posed inputs, whereas we study the more challenging monocular online setting, where geometry is incrementally reconstructed from unposed RGB inputs.

Monocular 3D Reconstruction and Perception. Reliable 3D geometry provides a key foundation for 3D scene understanding, yet it remains difficult to obtain from monocular or unposed RGB inputs. Recent RFMs have significantly improved monocular 3D reconstruction by recovering dense geometry from unposed images [56, 57, 59, 58, 28, 66, 2, 72]. Among them, CUT3R [57] supports streaming reconstruction but tends to accumulate drift over long sequences. VGGT [56] and π^3 [59] provide high-quality local geometry and cross-view cues, but memory limitations hinder their direct application to long RGB streams. VGGT-Long [8] extends VGGT to long RGB streams through chunk-wise reconstruction and cross-chunk alignment, while Pi-Long [1] adapts this strategy to π^3 reconstructions. Beyond reconstruction, recent works have leveraged RFMs for various downstream 3D scene understanding tasks [33, 76, 62, 51, 12, 31, 11, 46, 29]. PanoRecon [60] and EPRecon [75] focus on online panoptic 3D reconstruction from posed monocular videos, while EA3D [74] explores online open-world 3D object extraction with an online Gaussian feature map. MoonSeg3R [10] is closely related to our setting, performing monocular online 3D instance segmentation with CUT3R-based scene reconstruction [57]. In contrast, we leverage RFM-derived cross-view priors and compact persistent instance states for temporally consistent chunk-wise online 3D instance segmentation and cross-chunk association.

3 Method

Problem Setup. Given a monocular RGB stream $\mathcal{X} = \{\boldsymbol{I}_n\}_{n=1}^{N}$, we divide it into T non-overlapping chunks $\{\boldsymbol{C}_t\}_{t=1}^{T}$ and process them online. For each chunk $\boldsymbol{C}_t$, an RFM reconstructs a local 3D point cloud from the input RGB images. The observed scene $\mathcal{R}_t$ is formed by aligning and accumulating the local reconstructions from chunks $\{\boldsymbol{C}_1, \ldots, \boldsymbol{C}_t\}$. At step t, our goal is to predict instance masks for the chunk $\boldsymbol{C}_t$ and update the accumulated segmentation over $\mathcal{R}_t$ with consistent instance identities. Under the online setting, future chunks $\{\boldsymbol{C}_{t+1}, \ldots, \boldsymbol{C}_T\}$ are unavailable when processing $\boldsymbol{C}_t$. The final output is the accumulated 3D instance segmentation over $\mathcal{R}_T$.

Overview. As shown in Fig. 1, MonoChunk3D processes monocular RGB streams chunk by chunk for online 3D instance segmentation. For each incoming chunk $\boldsymbol{C}_t$, the RFM π^3 [59] reconstructs a local point cloud and extracts cross-view prior features. The local reconstruction is then incrementally aligned with the accumulated scene. We initialize the current query set $\boldsymbol{Q}_t^{\mathrm{cur}}$ through an adaptive 2D–3D query initialization module (Sec. 3.2), which aggregates RFM-derived cross-view priors and 3D geometric features under the guidance of VFM-generated 2D masks. In parallel, we maintain a compact persistent instance memory to store historical instance states, which are propagated to form the history-aware query set $\boldsymbol{Q}_t^{\mathrm{pre}}$ for the current chunk (Sec. 3.3). The current queries $\boldsymbol{Q}_t^{\mathrm{cur}}$ and history-aware queries $\boldsymbol{Q}_t^{\mathrm{pre}}$ are then jointly decoded to produce current-chunk segmentation results. Finally, these predictions are associated with persistent instance states to update the memory and accumulated segmentation (Sec. 3.4), enabling temporally consistent instance prediction across chunks during online inference.

3.1 Online Adaptation of Pi-Long

We build on Pi-Long [8, 1] to incrementally align chunk-wise π^3 reconstructions for streaming segmentation. At each step, the alignment is optimized using only the chunks observed so far, without accessing future chunks. When a new chunk $\boldsymbol{C}_t$ arrives, we run π^3 on it, cache the local reconstruction, and estimate a sequential Sim(3) transform using a short bridging window to align $\boldsymbol{C}_t$ with $\boldsymbol{C}_{t-1}$. For loop closure, we maintain an online retrieval database over observed frames and retain high-confidence loop candidates at each optimization cycle. Following Pi-Long, the retained candidates are used to derive chunk-level loop-closing Sim(3) constraints. These constraints are

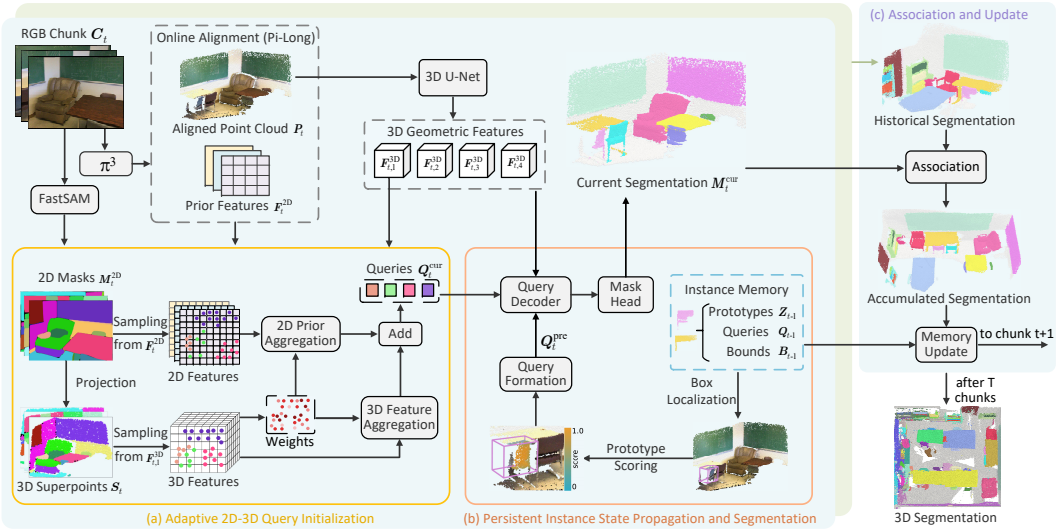


Figure 1: Framework overview of MonoChunk3D. (a) The adaptive 2D–3D query initialization module aggregates 3D geometric features with complementary RFM-derived cross-view priors under the guidance of VFM-generated 2D masks, forming current-chunk queries $\boldsymbol{Q}_t^{\mathrm{cur}}$. (b) The persistent instance state propagation module uses historical instance states stored in memory to generate history-aware queries $\boldsymbol{Q}_t^{\mathrm{pre}}$, providing long-term instance context for current-chunk segmentation. (c) The association and update module associates current instances in $\boldsymbol{M}_t^{\mathrm{cur}}$ with historical instances to update the accumulated segmentation and instance memory for subsequent chunks.

jointly optimized with sequential constraints in a lightweight incremental Sim(3) optimization over the observed chunks to correct cross-chunk drift during streaming inference. The resulting aligned point clouds provide the geometric input for subsequent query initialization and cross-chunk instance association.

3.2 Adaptive 2D–3D Query Initialization

Given the current chunk $\boldsymbol{C}_t \in \mathbb{R}^{L\times H\times W\times 3}$ with L input images, we denote by $\boldsymbol{P}_t$ the aligned point cloud reconstructed from $\boldsymbol{C}_t$. We reshape the final-layer patch tokens of π^3 [59] into spatial feature grids as the RFM-derived cross-view prior features $\boldsymbol{F}_t^{2\mathrm{D}}$. Each point in $\boldsymbol{P}_t$ stores its 3D coordinates, RGB value, geometric confidence score, frame index, and corresponding 2D pixel coordinates, where the confidence score measures the reliability of the reconstructed point. We then extract multi-scale 3D geometric features $\{\boldsymbol{F}_{t,k}^{3\mathrm{D}}\}_{k=1}^{4}$ from $\boldsymbol{P}_t$ using a 3D sparse U-Net [5]. In parallel, we use FastSAM [73] to obtain per-frame 2D masks $\boldsymbol{M}_t^{2\mathrm{D}}$ and transfer their labels to the reconstructed 3D points via the stored 2D–3D correspondences. Points assigned to the same mask within a frame form a local superpoint, and the resulting superpoints constitute S_t. Compared with point-wise modeling, superpoint-level representation reduces computation and provides more stable units for query initialization [24, 63]. We fuse $\boldsymbol{F}_t^{2\mathrm{D}}$ and $\boldsymbol{F}_{t,1}^{3\mathrm{D}}$ at the superpoint level to initialize the current query set $\boldsymbol{Q}_t^{\mathrm{cur}}$, combining multi-view consistency cues with local geometric structure.

For a superpoint $s \in S_t$ with point index set $\mathcal{I}_s$, let $\boldsymbol{f}_i^{3\mathrm{D}}$ and $\boldsymbol{f}_i^{2\mathrm{D}}$ denote the point-wise features sampled from $\boldsymbol{F}_{t,1}^{3\mathrm{D}}$ and $\boldsymbol{F}_t^{2\mathrm{D}}$ for each $i \in \mathcal{I}_s$. Let $\boldsymbol{g}_i$ denote the normalized coordinates of point i relative to the centroid of s. To mitigate the impact of noisy or misassigned points arising from 3D reconstruction and 2D segmentation errors, we perform adaptive point-wise aggregation within each superpoint. We first apply average pooling to the point-wise 3D features to obtain an initial 3D superpoint representation:

$$\bar{\boldsymbol{h}}_s^{3\mathrm{D}} = \frac{1}{|\mathcal{I}_s|}\sum_{i\in\mathcal{I}_s} \boldsymbol{f}_i^{3\mathrm{D}}. \tag{1}$$

We then compute point-wise weights by combining feature similarity to this initial representation with a relative geometric bias, and use them to aggregate the point-wise 3D features into $\boldsymbol{h}_s^{\mathrm{3D}}$:

$$a_i = \frac{\left(\boldsymbol{W}_q \bar{\boldsymbol{h}}_s^{\mathrm{3D}}\right)^\top \left(\boldsymbol{W}_k \boldsymbol{f}_i^{\mathrm{3D}}\right)}{\sqrt{d}} + \mathrm{MLP}_g(\boldsymbol{g}_i), \qquad \alpha_i = \mathrm{softmax}_{i \in \mathcal{I}_s}(a_i), \qquad \boldsymbol{h}_s^{\mathrm{3D}} = \sum_{i \in \mathcal{I}_s} \alpha_i \, \boldsymbol{W}_v \boldsymbol{f}_i^{\mathrm{3D}}, \tag{2}$$

where $\boldsymbol{W}_q$, $\boldsymbol{W}_k$, and $\boldsymbol{W}_v$ are learnable linear projections, d is the projected feature dimension, and MLP_g maps the normalized relative coordinate vector $\boldsymbol{g}_i$ to a geometric bias term.

We then combine the aggregation weights with the geometric confidence $c_i \in [0, 1]$ predicted by π^3 to aggregate the cross-view prior features:

$$\tilde{\alpha}_i = \frac{\alpha_i c_i}{\sum_{j \in \mathcal{I}_s} \alpha_j c_j + \epsilon}, \qquad \boldsymbol{h}_s^{\mathrm{2D}} = \sum_{i \in \mathcal{I}_s} \tilde{\alpha}_i \, \boldsymbol{W}_p \boldsymbol{f}_i^{\mathrm{2D}}, \tag{3}$$

where $\boldsymbol{W}_p$ is a learnable projection and ϵ is a small constant for numerical stability. We initialize the current query for superpoint s as $\boldsymbol{q}_s^{\mathrm{cur}} = \boldsymbol{h}_s^{\mathrm{3D}} + \boldsymbol{h}_s^{\mathrm{2D}}$, and collect all superpoint queries into $\boldsymbol{Q}_t^{\mathrm{cur}}$.

3.3 Persistent Instance State Propagation and Segmentation

The current query set $\boldsymbol{Q}_t^{\mathrm{cur}}$ is initialized from local observations in the current chunk. However, partial visibility and occlusion make chunk-level information insufficient for temporally consistent segmentation. We therefore maintain a persistent instance memory $\mathcal{M}_{t-1} = (\boldsymbol{Z}_{t-1}, \boldsymbol{Q}_{t-1}, \boldsymbol{B}_{t-1})$ that stores the historical states of N_{ins} instances up to chunk $t-1$, where $\boldsymbol{Z}_{t-1} = \{\boldsymbol{z}_k\}_{k=1}^{N_{\mathrm{ins}}}$, $\boldsymbol{Q}_{t-1} = \{\boldsymbol{q}_k\}_{k=1}^{N_{\mathrm{ins}}}$, and $\boldsymbol{B}_{t-1} = \{\boldsymbol{b}_k\}_{k=1}^{N_{\mathrm{ins}}}$. For the k-th historical instance, $\boldsymbol{z}_k$, $\boldsymbol{q}_k$, and $\boldsymbol{b}_k$ denote its explicit prototype, implicit query embedding, and spatial bounds, respectively. The explicit prototype is directly aggregated from historical 3D features within predicted masks, providing a discriminative instance representation for retrieving support points in the current chunk. The implicit query embedding preserves long-term instance context across chunks, while the spatial bounds store the instance extent for localizing candidate regions in the current chunk. We detail the construction and update of these memory states in Sec. 3.4. For the first chunk, history-aware propagation is skipped, and valid predictions from $\boldsymbol{Q}_1^{\mathrm{cur}}$ initialize the memory.

History-aware Query Propagation. We propagate the historical instance states in $\mathcal{M}_{t-1}$ to the current chunk to form a history-aware query set $\boldsymbol{Q}_t^{\mathrm{pre}}$. For the k-th historical instance, we use its spatial bounds $\boldsymbol{b}_k$ to localize a candidate box and retain the instance for propagation only if the box contains at least 10 current chunk points. For each retained instance, we use the historical prototype $\boldsymbol{z}_k$ to retrieve support points within the candidate box via feature similarity. Specifically, we select points whose similarity to $\boldsymbol{z}_k$ exceeds a threshold τ, and collect their indices into $\mathcal{I}_k$:

$$s_{k,i} = \frac{\boldsymbol{z}_k^\top \boldsymbol{f}_i^{\mathrm{3D}}}{\|\boldsymbol{z}_k\|_2 \, \|\boldsymbol{f}_i^{\mathrm{3D}}\|_2}, \qquad \mathcal{I}_k = \{\, i \mid \boldsymbol{x}_i \in \mathrm{Box}(\boldsymbol{b}_k),\ s_{k,i} > \tau \,\}, \tag{4}$$

where $\boldsymbol{f}_i^{\mathrm{3D}}$ is the point-wise feature sampled from $\boldsymbol{F}_{t,1}^{\mathrm{3D}}$, $\boldsymbol{x}_i$ is the 3D coordinate of point i, and $\mathrm{Box}(\boldsymbol{b}_k)$ denotes the candidate box defined by the spatial bounds $\boldsymbol{b}_k$. We then aggregate the support points indexed by $\mathcal{I}_k$ using the same adaptive point-wise aggregation described in Sec. 3.2 to derive a current-chunk query $\hat{\boldsymbol{q}}_k$. The query $\hat{\boldsymbol{q}}_k$ captures the current observation of the historical instance, while the historical query $\boldsymbol{q}_k$ preserves its long-term instance context.

We adaptively fuse $\hat{\boldsymbol{q}}_k$ with $\boldsymbol{q}_k$ according to the reliability of $\hat{\boldsymbol{q}}_k$, yielding the history-aware query $\boldsymbol{q}_k^{\mathrm{pre}}$. The reliability is estimated from semantic consistency between $\hat{\boldsymbol{q}}_k$ and $\boldsymbol{q}_k$, and geometric consistency between the bounding box of the selected support points and the historical box defined by $\boldsymbol{b}_k$:

$$r_k^{\mathrm{sem}} = \frac{\boldsymbol{q}_k^\top \hat{\boldsymbol{q}}_k}{\|\boldsymbol{q}_k\|_2 \, \|\hat{\boldsymbol{q}}_k\|_2}, \qquad r_k^{\mathrm{geo}} = \mathrm{IoU}\big(\mathrm{Box}(\{\boldsymbol{x}_i \mid i \in \mathcal{I}_k\}), \mathrm{Box}(\boldsymbol{b}_k)\big). \tag{5}$$

We feed these two consistency scores into an MLP to predict a scalar fusion gate for combining $\hat{\boldsymbol{q}}_k$ and $\boldsymbol{q}_k$:

$$\gamma_k = \mathrm{sigmoid}(\mathrm{MLP}([r_k^{\mathrm{sem}},\, r_k^{\mathrm{geo}}])), \qquad \boldsymbol{q}_k^{\mathrm{pre}} = (1 - \gamma_k)\, \boldsymbol{q}_k + \gamma_k \, \hat{\boldsymbol{q}}_k. \tag{6}$$

Collecting all $\boldsymbol{q}_k^{\mathrm{pre}}$ yields the history-aware query set $\boldsymbol{Q}_t^{\mathrm{pre}}$.

Query Decoder. We further refine the query representations with a query decoder to obtain current-chunk instance segmentation results. We first perform cross-attention between the history-aware queries $\boldsymbol{Q}_t^{\mathrm{pre}}$ and the current queries $\boldsymbol{Q}_t^{\mathrm{cur}}$, enabling long-term instance context to guide current-chunk segmentation. We then concatenate the two query sets and jointly decode them with the multi-scale 3D features $\{\boldsymbol{F}_{t,k}^{\mathrm{3D}}\}_{k=2}^{4}$ using a three-layer Transformer decoder. Following [63], the concatenated queries interact with superpoint-level 3D features through cross-attention for efficient feature aggregation. The final mask head projects point-level 3D features with an MLP and computes dot products with refined queries and applies sigmoid to obtain point-wise masks, while valid masks and their corresponding refined queries are retained as $\boldsymbol{M}_t^{\mathrm{cur}}$ and $\boldsymbol{Q}_t^{\mathrm{ref}}$, respectively.

3.4 Association and Memory Update

Given the segmentation predictions of the current chunk, we associate current instances with historical instance states and update the persistent memory accordingly. Rather than relying on pairwise mask comparisons for mask-level merging [42, 67, 68, 35, 41], we perform cross-chunk matching directly on compact instance state representations.

Instance Association. For each historical instance k and current instance j, we compute an affinity score by combining refined query similarity and spatial overlap:

$$A_{k,j} = \frac{\boldsymbol{q}_k^{\top}\boldsymbol{q}_j^{\mathrm{ref}}}{\|\boldsymbol{q}_k\|_2\,\|\boldsymbol{q}_j^{\mathrm{ref}}\|_2} + \mathrm{IoU}\big(\mathrm{Box}(\boldsymbol{b}_k), \mathrm{Box}(\boldsymbol{m}_j^{\mathrm{cur}})\big). \tag{7}$$

Here, $\boldsymbol{q}_k$ and $\boldsymbol{b}_k$ denote the query and spatial bounds of historical instance k, while $\boldsymbol{q}_j^{\mathrm{ref}} \in \boldsymbol{Q}_t^{\mathrm{ref}}$ and $\boldsymbol{m}_j^{\mathrm{cur}} \in \boldsymbol{M}_t^{\mathrm{cur}}$ denote the refined query and mask of current instance j. We then perform Hungarian matching on the affinity matrix $\boldsymbol{A}$ to establish associations between historical and current instances.

Memory Update. Based on the association results, we update the persistent instance memory. For each matched pair (k, j), we update the state of historical instance k using the current prediction of instance j. Specifically, we explicitly aggregate the 3D features $\boldsymbol{F}_{t,1}^{\mathrm{3D}}$ within the matched current mask $\boldsymbol{m}_j^{\mathrm{cur}}$ to obtain a refined prototype $\boldsymbol{z}_j^{\mathrm{ref}}$:

$$\boldsymbol{z}_j^{\mathrm{ref}} = \frac{\sum_i m_{j,i}^{\mathrm{cur}}\,\boldsymbol{f}_i^{\mathrm{3D}}}{\sum_i m_{j,i}^{\mathrm{cur}}}. \tag{8}$$

We then update the historical prototype and query by weighted averaging based on the point counts:

$$\boldsymbol{z}_k \leftarrow \frac{n_k\boldsymbol{z}_k + n_j\boldsymbol{z}_j^{\mathrm{ref}}}{n_k + n_j}, \qquad \boldsymbol{q}_k \leftarrow \frac{n_k\boldsymbol{q}_k + n_j\boldsymbol{q}_j^{\mathrm{ref}}}{n_k + n_j}, \tag{9}$$

where n_k is the accumulated point count of historical instance k, and n_j is the point count of the matched current instance j. We then update the accumulated count as $n_k \leftarrow n_k + n_j$. For the spatial bounds, we update $\boldsymbol{b}_k$ using the box corners of the matched current instance j. Instead of directly storing their global coordinates, we record these corners in the local coordinates of the corresponding chunk, together with the chunk index. After incremental Sim(3) optimization, their global coordinates can be efficiently updated using the latest chunk transformation. For unmatched current instances, we initialize new memory entries using their current prototypes, refined queries, and spatial bounds. The resulting memory $\mathcal{M}_t = (\boldsymbol{Z}_t, \boldsymbol{Q}_t, \boldsymbol{B}_t)$ is propagated to the next chunk.

3.5 Training Objectives

For a sequence of T chunks, we train the model with the current-chunk segmentation loss, the propagated segmentation loss, the query contrastive loss, and the prototype ranking loss:

$$\mathcal{L} = \frac{1}{T}\sum_{t=1}^{T}\left(\mathcal{L}_{\mathrm{seg},t}^{\mathrm{cur}} + \mathcal{L}_{\mathrm{seg},t}^{\mathrm{pre}} + \lambda_{\mathrm{rank}}\mathcal{L}_{\mathrm{rank},t} + \lambda_{\mathrm{con}}\mathcal{L}_{\mathrm{con},t}\right). \tag{10}$$

For the current chunk t, we first match the mask predictions from $\boldsymbol{Q}_t^{\mathrm{cur}}$ to the ground-truth instances via Hungarian matching, and supervise the matched predictions with the standard segmentation loss $\mathcal{L}_{\mathrm{seg},t}^{\mathrm{cur}}$, including two-class query classification between object and no-object, binary cross-entropy, and Dice terms. We further apply the same supervision to the mask predictions from $\boldsymbol{Q}_t^{\mathrm{pre}}$

using their corresponding instance labels in chunk t, yielding $\mathcal{L}^{\text{pre}}_{\text{seg},t}$. For the query contrastive loss $\mathcal{L}_{\text{con},t}$, we follow [63] to pull together query features of the same instance and push apart those of different instances, thereby facilitating cross-chunk association.

In addition, we introduce a ranking loss on the support index set $\mathcal{I}_k$ in Eq. (4) to improve the discriminability of historical prototypes. For each historical prototype $\boldsymbol{z}_k$, points indexed by $\mathcal{I}_k$ that belong to the corresponding ground-truth instance are treated as positives, while the remaining points in $\mathcal{I}_k$ are treated as negatives. Based on the similarity score $s_{k,i}$ in Eq. (4), we define the average similarities over positive and negative points as follows:

$$\bar{s}_k^+ = \frac{\sum_{i\in\mathcal{I}_k} y_{k,i}\, s_{k,i}}{\sum_{i\in\mathcal{I}_k} y_{k,i}}, \qquad \bar{s}_k^- = \frac{\sum_{i\in\mathcal{I}_k} (1-y_{k,i})\, s_{k,i}}{\sum_{i\in\mathcal{I}_k} (1-y_{k,i})}, \tag{11}$$

where $y_{k,i} \in \{0,1\}$ indicates whether point i belongs to the ground-truth instance associated with $\boldsymbol{z}_k$. We then enforce the average similarity of positive points to be higher than that of negative points:

$$\mathcal{L}_{\text{rank},t} = \frac{1}{|\mathcal{K}_t|} \sum_{k\in\mathcal{K}_t} \max\big(0,\ \delta - \bar{s}_k^+ + \bar{s}_k^-\big)\,, \tag{12}$$

where $\mathcal{K}_t$ denotes the set of historical prototypes with valid positive and negative points in chunk t.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate our method for monocular online 3D instance segmentation on three indoor benchmarks, including ScanNetV2 [6], ScanNet200 [48], and SceneNN [16]. ScanNetV2 contains 1,513 RGB-D scans, and we follow the standard split with 1,201 scenes for training and 312 scenes for evaluation. ScanNet200 provides finer-grained instance annotations on the same scans and covers 200 semantic categories. SceneNN contains 50 scanned scenes with high-quality instance and semantic annotations, and we follow prior work [63, 55, 53, 10] in evaluating on the same 12 scenes.

Evaluation Protocol and Metrics. Our model takes only monocular RGB streams as input and predicts instance masks on incrementally aligned reconstructed scenes, without using ground-truth depth or camera poses during inference. For fair comparison with standard benchmarks, we evaluate predictions in the original annotated scene space. Specifically, we use a precomputed scene-level alignment transform to register each reconstructed scene to the original annotated scene space, and transfer the predicted instance labels to the original scene geometry via 3D nearest-neighbor label propagation. This alignment is used only for evaluation, and details of the procedure are provided in Appendix A.3. Following [63], we evaluate class-agnostic instance segmentation using Average Precision (AP), which measures instance mask quality independent of semantic categories. We report AP_{25} and AP_{50} at IoU thresholds of 25% and 50%, as well as mAP averaged over IoU thresholds from 50% to 95% with a step size of 5%. All reported AP values are expressed as percentages.

Implementation Details. We train our models with PyTorch on 8 NVIDIA RTX 3090 GPUs, and measure inference runtime on a single RTX 3090 GPU. The pretrained π^3 model is kept frozen during training. During training, the chunk size is randomly sampled from 3 to 12 frames, and each training sample contains 2 to 6 consecutive chunks. During inference, the chunk size is set to 12 frames by default. We use the AdamW optimizer with a learning rate of 1×10^{-4}, a weight decay of 0.05, and a batch size of 4. For hyperparameters, we set $\lambda_{\text{rank}} = 0.5$, $\lambda_{\text{con}} = 0.5$, $\delta = 0.5$, and $\tau = 0.4$.

Compared Methods. In Table 1, we compare MonoChunk3D with VFM-assisted methods, including OVIR-3D [38], MaskClustering [65], SAI3D [68], SAMPro3D [61], Open3DIS [42], and SAM3D [67], as well as online methods such as OnlineAnySeg [53], ESAM [63], AutoSeg3D [55], and MoonSeg3R [10]. In Table 2, we further compare with TD3D [23], OneFormer3D [24], INS-Conv [35], and TD3D-MA [64]. Additionally, we build chunk-wise monocular baselines by adapting ESAM and OnlineAnySeg to the same aligned reconstructed geometry used by MonoChunk3D. We denote these adapted baselines as ESAM-C and OnlineAnySeg-C, respectively. ESAM-C performs segmentation and learnable merging within each chunk, whereas OnlineAnySeg-C extracts frame-wise masks with CropFormer [45] and merges them at the chunk level.

Table 1: Class-agnostic 3D instance segmentation results on ScanNet200 and transfer results from ScanNet200 to SceneNN. ESAM and AutoSeg3D use FastSAM as the 2D segmentation model. For inference speed, we report the average per-frame segmentation time, with underlined values denoting the average per-frame RFM inference time. * For OnlineAnySeg [53], we follow [10] and report the runtime of the full pipeline. 🖼+📷 denotes posed RGB-D input, while 🖼 denotes RGB-only input.

Method	Input	Online	ScanNet200			SceneNN			Speed (ms/frame)
			mAP	AP_{50}	AP_{25}	mAP	AP_{50}	AP_{25}	
OVIR-3D [38]	🖼+📷	✗	14.4	27.5	38.8	12.3	24.4	34.6	–
MaskClustering [65]	🖼+📷	✗	19.7	36.4	51.4	16.3	31.7	46.2	–
SAI3D [68]	🖼+📷	✗	28.2	47.2	67.9	18.6	34.7	65.7	–
SAMPro3D [61]	🖼+📷	✗	18.0	32.8	56.1	12.6	25.8	53.2	–
Open3DIS [42]	🖼+📷	✗	34.6	43.1	48.5	18.2	32.2	48.9	–
SAM3D [67]	🖼+📷	✓	20.2	35.7	55.5	15.1	30.0	51.8	2887
OnlineAnySeg [53]	🖼+📷	✓	18.6	36.1	53.5	18.1	35.3	59.5	3000*
ESAM [63]	🖼+📷	✓	43.4	65.4	80.9	28.6	50.4	71.0	94
AutoSeg3D [55]	🖼+📷	✓	46.2	67.9	81.7	30.2	54.1	72.8	99
MoonSeg3R [10]	🖼	✓	16.7	33.3	50.0	14.3	31.4	48.4	55+<u>66</u>
OnlineAnySeg-C [53]	🖼	✓	12.7	25.1	41.6	10.2	25.2	49.0	3000*+<u>79</u>
ESAM-C [63]	🖼	✓	17.4	34.7	58.8	12.6	27.4	55.2	46+<u>79</u>
Ours	🖼	✓	**23.2**	**45.1**	**67.9**	**16.6**	**35.5**	**58.9**	43+<u>79</u>

Table 2: Class-agnostic 3D instance segmentation results on ScanNetV2 and transfer results from ScanNetV2 to SceneNN. 🖼+📷 denotes posed RGB-D input, while 🖼 denotes RGB-only input.

Method	Input	Online	ScanNetV2			SceneNN		
			mAP	AP_{50}	AP_{25}	mAP	AP_{50}	AP_{25}
TD3D [23]	🖼+📷	✗	46.2	71.1	81.3	–	–	–
OneFormer3D [24]	🖼+📷	✗	59.3	78.8	86.7	–	–	–
INS-Conv [35]	🖼+📷	✓	–	57.4	–	–	–	–
TD3D-MA [64]	🖼+📷	✓	39.0	60.5	71.3	26.0	42.8	59.2
ESAM [63]	🖼+📷	✓	41.6	59.6	75.2	30.3	47.6	63.4
AutoSeg3D [55]	🖼+📷	✓	43.4	62.5	77.4	33.1	52.6	63.8
OnlineAnySeg-C [53]	🖼	✓	15.5	30.3	48.6	14.2	30.4	54.5
ESAM-C [63]	🖼	✓	22.8	43.2	69.7	17.4	36.2	68.6
Ours	🖼	✓	**29.9**	**54.5**	**79.3**	**24.1**	**47.3**	**76.0**

4.2 Comparison with State-of-the-Art Methods

Quantitative Results. As shown in Table 1, we report results on ScanNet200 and transfer from ScanNet200 to SceneNN. Among RGB-only online methods, our method achieves the best performance. Specifically, it reaches 23.2 mAP on ScanNet200, outperforming ESAM-C by 5.8, and achieves 16.6 mAP on SceneNN, improving over MoonSeg3R by 2.3. We also observe clear performance drops from OnlineAnySeg to OnlineAnySeg-C and from ESAM to ESAM-C, suggesting that replacing precise posed RGB-D inputs with monocular reconstructed geometry makes segmentation and merging more challenging. Table 2 further reports results on ScanNetV2 and transfer results from ScanNetV2 to SceneNN. Our method performs best among RGB-only online methods on both datasets, achieving 29.9 mAP on ScanNetV2 and 24.1 mAP on SceneNN. It outperforms ESAM-C by 7.1 and 6.7 mAP, respectively, demonstrating strong performance and cross-dataset generalization.

Runtime Analysis. Table 1 also reports the per-frame runtime of segmentation and reconstruction. Our segmentation takes 43 ms per frame, comparable to ESAM-C at 46 ms per frame. Our method, OnlineAnySeg-C, and ESAM-C share the same π^3 monocular reconstruction frontend, whose inference takes 79 ms per frame on average. In addition, chunk alignment introduces an additional

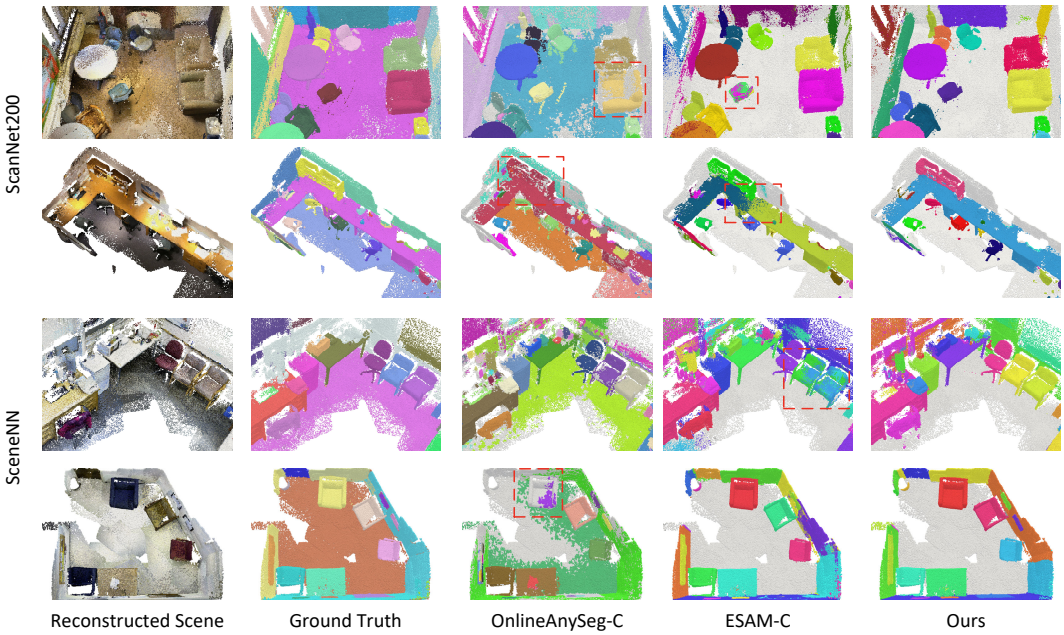


Figure 2: Visual comparisons of monocular 3D instance segmentation on ScanNet200 and SceneNN. Red dashed boxes highlight representative failure regions of compared methods.

Table 3: Ablation on query initialization.

Method	mAP	AP_{50}	AP_{25}
Without 3D geometric features	18.8	37.5	62.7
Without RFM-derived cross-view prior	21.1	42.0	65.5
Average pooling	22.2	43.6	66.4
Full model	**23.2**	**45.1**	**67.9**

overhead of about 0.2–0.3 s for Sim(3) alignment, with each loop optimization run taking about 40 ms in total [8, 1]. The appendix provides a detailed runtime breakdown of our segmentation pipeline.

Qualitative Results. We visualize representative predictions on ScanNet200 and SceneNN in Fig. 2. Compared with OnlineAnySeg-C and ESAM-C, our method produces cleaner and more complete 3D instance masks from monocular reconstructed geometry. As highlighted by the red dashed boxes, the compared methods often fragment objects, merge adjacent instances, or miss object parts. In contrast, our method better preserves object completeness and separates neighboring instances, yielding predictions closer to the ground truth.

4.3 Ablation Studies

Ablation on Query Initialization. As shown in Table 3, removing the 3D geometric features reduces mAP from 23.2 to 18.8, with AP_{50} and AP_{25} also dropping from 45.1 to 37.5 and from 67.9 to 62.7, respectively. Removing the RFM-derived cross-view prior also degrades performance, resulting in 21.1 mAP, 42.0 AP_{50}, and 65.5 AP_{25}. These results demonstrate that both feature sources contribute to query initialization, with 3D geometric features having a larger impact. Replacing adaptive weighted aggregation with average pooling further reduces mAP to 22.2, demonstrating the benefit of adaptive aggregation for fusing cross-view priors and 3D geometric features. Overall, the full model achieves the best performance across all three metrics.

Ablation on History-aware Query Propagation. Table 4 reports the ablation results of the history-aware query propagation module. Removing the propagated history-aware queries $\boldsymbol{Q}_t^{\text{pre}}$ leads to a clear performance drop, with mAP decreasing from 23.2 to 20.8 and AP_{50} from 45.1 to 41.8, showing that historical instance context is important for chunk-wise segmentation. Using only the history from the immediately preceding chunk also lowers mAP to 21.6, indicating that the model

Table 4: Ablation on history-aware query propagation.

Method	mAP	AP_{50}	AP_{25}
Without history-aware queries	20.8	41.8	65.2
Previous-chunk history only	21.6	42.4	65.7
Without prototype-derived queries	21.3	42.0	65.6
Full model	**23.2**	**45.1**	**67.9**

benefits from long-term instance context. Removing the prototype-derived queries $\hat{\boldsymbol{q}}_k$ reduces mAP to 21.3, suggesting that explicit prototypes provide instance-specific priors for history-aware query propagation. These results verify the effectiveness of the proposed history-aware query propagation design.

5 Conclusion and Limitations

Conclusion. In this work, we presented MonoChunk3D, a chunk-wise online framework for monocular 3D instance segmentation without relying on ground-truth depth or camera poses. MonoChunk3D incrementally reconstructs and aligns incoming RGB chunks, and initializes current-chunk instance queries by adaptively aggregating complementary RFM-derived cross-view priors and 3D geometric features. To address the limited context of individual chunks, MonoChunk3D maintains persistent instance states and propagates them as history-aware queries, allowing long-term instance context to guide current-chunk segmentation. By associating current predictions with historical instance states, MonoChunk3D updates persistent states online and enables temporally consistent segmentation across chunks. Extensive experiments on multiple datasets demonstrate that MonoChunk3D achieves superior performance with efficient online inference in the monocular online setting.

Limitations. MonoChunk3D relies on monocular reconstruction and cross-chunk alignment. Noisy or incomplete reconstructed geometry may degrade 3D segmentation, while alignment errors under large viewpoint changes may affect instance localization and association. Moreover, with one-to-one matching, each historical instance can be matched to at most one current prediction. When an object is over-segmented, unmatched predictions may be treated as new instances, allowing instance fragmentation to persist across chunks. Incorrect associations may also lead to inaccurate instance state updates, affecting subsequent query propagation and association.

Acknowledgments

The authors thank the anonymous reviewers for their constructive feedback. This work was supported by the National Natural Science Foundation of China under Grant Nos. 62361166670, U62276144, and 62306155. This research was also supported by the Supercomputing Center of Nankai University (NKSC). The authors declare no competing interests.

References

[1] Pi-Long: Extending π^3's capabilities on kilometer-scale with the framework of vggt-long. `https://github.com/DengKaiCQ/Pi-Long`, 2025. GitHub repository.

[2] Yohann Cabon, Lucas Stoffl, Leonid Antsfeld, Gabriela Csurka, Boris Chidlovskii, Jerome Revaud, and Vincent Leroy. Must3r: Multi-view network for stereo 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1050–1060, 2025.

[3] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *Advances in Neural Information Processing Systems*, 33:4247–4258, 2020.

[4] Tianheng Cheng, Lin Song, Yixiao Ge, Wenyu Liu, Xinggang Wang, and Ying Shan. Yolo-world: Real-time open-vocabulary object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16901–16911, 2024.

[5] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3075–3084, 2019.

[6] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[7] Angela Dai, Jean Lahoud, Hisham Cholakkal, Rao Anwer, Salman Khan, Fahad Khan, et al. Open-yolo 3d: Towards fast and accurate open-vocabulary 3d instance segmentation. In *International Conference on Learning Representations*, volume 2025, pages 65143–65156, 2025.

[8] Kai Deng, Zexin Ti, Jiawei Xu, Jian Yang, and Jin Xie. VGGT-Long: Chunk it, loop it, align it–pushing vggt's limits on kilometer-scale long rgb sequences. *arXiv preprint arXiv:2507.16443*, 2025.

[9] Runyu Ding, Jihan Yang, Chuhui Xue, Wenqing Zhang, Song Bai, and Xiaojuan Qi. Pla: Language-driven open-vocabulary 3d scene understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7010–7019, 2023.

[10] Zhipeng Du, Duolikun Danier, Jan Eric Lenssen, and Hakan Bilen. Moonseg3r: Monocular online zero-shot segment anything in 3d with reconstructive foundation priors. *arXiv preprint arXiv:2512.15577*, 2025.

[11] Zhiwen Fan, Jian Zhang, Wenyan Cong, Peihao Wang, Renjie Li, Kairun Wen, Shijie Zhou, Achuta Kadambi, Zhangyang Wang, Danfei Xu, et al. Large spatial model: End-to-end unposed images to semantic 3d. *Advances in neural information processing systems*, 37:40212–40229, 2024.

[12] Ziren Gong, Xiaohan Li, Fabio Tosi, Jiawei Han, Stefano Mattoccia, Jianfei Cai, and Matteo Poggi. Ov3r: Open-vocabulary semantic 3d reconstruction from rgb videos. *arXiv preprint arXiv:2507.22052*, 2025.

[13] Deepti Hegde, Jeya Maria Jose Valanarasu, and Vishal Patel. Clip goes 3d: Leveraging prompt tuning for language grounded 3d recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2028–2038, 2023.

[14] Ji Hou, Angela Dai, and Matthias Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4421–4430, 2019.

[15] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11108–11117, 2020.

[16] 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 *2016 fourth international conference on 3D vision (3DV)*, pages 92–101. Ieee, 2016.

[17] Shi-Sheng Huang, Ze-Yu Ma, Tai-Jiang Mu, Hongbo Fu, and Shi-Min Hu. Supervoxel convolution for online 3d semantic segmentation. *ACM Transactions on Graphics (TOG)*, 40(3):1–15, 2021.

[18] Zhening Huang, Xiaoyang Wu, Xi Chen, Hengshuang Zhao, Lei Zhu, and Joan Lasenby. Openins3d: Snap and lookup for 3d open-vocabulary instance segmentation. In *European Conference on Computer Vision*, pages 169–185. Springer, 2024.

[19] Krishna Murthy Jatavallabhula, Alihusein Kuwajerwala, Qiao Gu, Mohd Omama, Tao Chen, Alaa Maalouf, Shuang Li, Ganesh Iyer, Soroush Saryazdi, Nikhil Keetha, et al. Conceptfusion: Open-set multimodal 3d mapping. *arXiv preprint arXiv:2302.07241*, 2023.

[20] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pages 4904–4916. PMLR, 2021.

[21] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 19729–19739, 2023.

[22] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

[23] Maksim Kolodiazhnyi, Anna Vorontsova, Anton Konushin, and Danila Rukhovich. Top-down beats bottom-up in 3d instance segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3566–3574, 2024.

[24] Maxim Kolodiazhnyi, Anna Vorontsova, Anton Konushin, and Danila Rukhovich. Oneformer3d: One transformer for unified point cloud segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20943–20953, 2024.

[25] Xin Lai, Yuhui Yuan, Ruihang Chu, Yukang Chen, Han Hu, and Jiaya Jia. Mask-attention-free transformer for 3d instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3693–3703, 2023.

[26] Seungjun Lee, Yuyang Zhao, and Gim Hee Lee. Segment any 3d object with language. *arXiv preprint arXiv:2404.02157*, 2024.

[27] Seungjun Lee, Zihan Wang, Yunsong Wang, and Gim Hee Lee. Embodiedsplat: Online feed-forward semantic 3dgs for open-vocabulary 3d scene understanding. *arXiv preprint arXiv:2603.04254*, 2026.

[28] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European conference on computer vision*, pages 71–91. Springer, 2024.

[29] Changyang Li, Xueqing Huang, Shin-Fang Chng, Huangying Zhan, Qingan Yan, and Yi Xu. Fast3dis: Feed-forward anchored scene transformer for 3d instance segmentation. *arXiv preprint arXiv:2603.25993*, 2026.

[30] Feng Li, Hao Zhang, Peize Sun, Xueyan Zou, Shilong Liu, Chunyuan Li, Jianwei Yang, Lei Zhang, and Jianfeng Gao. Segment and recognize anything at any granularity. In *European Conference on Computer Vision*, pages 467–484. Springer, 2024.

[31] Hao Li, Zhengyu Zou, Fangfu Liu, Xuanyang Zhang, Fangzhou Hong, Yukang Cao, Yushi Lan, Manyuan Zhang, Gang Yu, Dingwen Zhang, et al. Iggt: Instance-grounded geometry transformer for semantic 3d reconstruction. *arXiv preprint arXiv:2510.22706*, 2025.

[32] Hongyang Li, Jinyuan Qu, and Lei Zhang. Ovseg3r: Learn open-vocabulary instance segmentation from 2d via 3d reconstruction. *arXiv preprint arXiv:2509.23541*, 2025.

[33] Qijing Li, Jingxiang Sun, Liang An, Zhaoqi Su, Hongwen Zhang, and Yebin Liu. Semanticsplat: Feed-forward 3d scene understanding with language-aware gaussian fields. *arXiv preprint arXiv:2506.09565*, 2025.

[34] Zhihao Liang, Zhihao Li, Songcen Xu, Mingkui Tan, and Kui Jia. Instance segmentation in 3d scenes using semantic superpoint tree networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2783–2792, 2021.

[35] Leyao Liu, Tian Zheng, Yun-Jou Lin, Kai Ni, and Lu Fang. Ins-conv: Incremental sparse convolution for online 3d segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18975–18984, 2022.

[36] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pages 38–55. Springer, 2024.

[37] Jiahao Lu, Jiacheng Deng, Chuxin Wang, Jianfeng He, and Tianzhu Zhang. Query refinement transformer for 3d instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18516–18526, 2023.

[38] Shiyang Lu, Haonan Chang, Eric Pu Jing, Abdeslam Boularias, and Kostas Bekris. Ovir-3d: Open-vocabulary 3d instance retrieval without training on 3d data. In *Conference on Robot Learning*, pages 1610–1620. PMLR, 2023.

[39] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *2017 IEEE International Conference on Robotics and automation (ICRA)*, pages 4628–4635, 2017.

[40] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *International Conference on Computer Vision (ICCV)*, 2019.

[41] Gaku Narita, Takashi Seno, Tomoya Ishikawa, and Yohsuke Kaji. Panopticfusion: Online volumetric semantic mapping at the level of stuff and things. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4205–4212, 2019.

[42] Phuc Nguyen, Tuan Duc Ngo, Evangelos Kalogerakis, Chuang Gan, Anh Tran, Cuong Pham, and Khoi Nguyen. Open3dis: Open-vocabulary 3d instance segmentation with 2d mask guidance. In *CVPR*, pages 4018–4028, 2024.

[43] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[44] Songyou Peng, Kyle Genova, Chiyu Jiang, Andrea Tagliasacchi, Marc Pollefeys, Thomas Funkhouser, et al. Openscene: 3d scene understanding with open vocabularies. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 815–824, 2023.

[45] Lu Qi, Jason Kuen, Tiancheng Shen, Jiuxiang Gu, Weidong Guo, Jiaya Jia, Zhe Lin, and Ming-Hsuan Yang. High quality entity segmentation. In *ICCV*, 2023.

[46] Jinyuan Qu, Hongyang Li, and Lei Zhang. Segvggt: Joint 3d reconstruction and instance segmentation from multi-view images. *arXiv preprint arXiv:2603.19926*, 2026.

[47] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[48] David Rozenberszki, Or Litany, and Angela Dai. Language-grounded indoor 3d semantic segmentation in the wild. In *European conference on computer vision*, pages 125–141. Springer, 2022.

[49] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3d: Mask transformer for 3d semantic instance segmentation. *arXiv preprint arXiv:2210.03105*, 2022.

[50] Jiahao Sun, Chunmei Qing, Junpeng Tan, and Xiangmin Xu. Superpoint transformer for 3d scene instance segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 2393–2401, 2023.

[51] Xiangyu Sun, Haoyi Jiang, Liu Liu, Seungtae Nam, Gyeongjin Kang, Xinjie Wang, Wei Sui, Zhizhong Su, Wenyu Liu, Xinggang Wang, et al. Uni3r: Unified 3d reconstruction and semantic understanding via generalizable gaussian splatting from unposed multi-view images. *arXiv preprint arXiv:2508.03643*, 2025.

[52] Ayça Takmaz, Elisabetta Fedele, Robert W Sumner, Marc Pollefeys, Federico Tombari, and Francis Engelmann. Openmask3d: Open-vocabulary 3d instance segmentation. *arXiv preprint arXiv:2306.13631*, 2023.

[53] Yijie Tang, Jiazhao Zhang, Yuqing Lan, Yulan Guo, Dezun Dong, Chenyang Zhu, and Kai Xu. Onlineanyseg: Online zero-shot 3d segmentation by visual foundation model guided 2d mask merging. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 3676–3685, 2025.

[54] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019.

[55] Hanshi Wang, Zijian Cai, Jin Gao, Yiwei Zhang, Weiming Hu, Ke Wang, and Zhipeng Zhang. Online segment any 3d thing as instance tracking. *arXiv preprint arXiv:2512.07599*, 2025.

[56] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025.

[57] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10510–10522, 2025.

[58] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20697–20709, 2024.

[59] Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. *arXiv preprint arXiv:2507.13347*, 2025.

[60] Dong Wu, Zike Yan, and Hongbin Zha. Panorecon: Real-time panoptic 3d reconstruction from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21507–21518, 2024.

[61] Mutian Xu, Xingyilang Yin, Lingteng Qiu, Yang Liu, Xin Tong, and Xiaoguang Han. Sampro3d: Locating sam prompts in 3d for zero-shot instance segmentation. In *International Conference on 3D Vision 2025*.

[62] Qi Xu, Dongxu Wei, Lingzhe Zhao, Wenpu Li, Zhangchi Huang, Shunping Ji, and Peidong Liu. Siu3r: Simultaneous scene understanding and 3d reconstruction beyond feature alignment. *arXiv preprint arXiv:2507.02705*, 2025.

[63] Xiuwei Xu, Huangxing Chen, Linqing Zhao, Ziwei Wang, Jie Zhou, and Jiwen Lu. Embodiedsam: Online segment any 3d thing in real time. *arXiv preprint arXiv:2408.11811*, 2024.

[64] Xiuwei Xu, Chong Xia, Ziwei Wang, Linqing Zhao, Yueqi Duan, Jie Zhou, and Jiwen Lu. Memory-based adapters for online 3d scene perception. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21604–21613, 2024.

[65] Mi Yan, Jiazhao Zhang, Yan Zhu, and He Wang. Maskclustering: View consensus based mask graph clustering for open-vocabulary 3d instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28274–28284, 2024.

[66] Jianing Yang, Alexander Sax, Kevin J Liang, Mikael Henaff, Hao Tang, Ang Cao, Joyce Chai, Franziska Meier, and Matt Feiszli. Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21924–21935, 2025.

[67] Yunhan Yang, Xiaoyang Wu, Tong He, Hengshuang Zhao, and Xihui Liu. Sam3d: Segment anything in 3d scenes. *arXiv preprint arXiv:2306.03908*, 2023.

[68] Yingda Yin, Yuzheng Liu, Yang Xiao, Daniel Cohen-Or, Jingwei Huang, and Baoquan Chen. Sai3d: Segment any instance in 3d scenes. In *CVPR*, pages 3292–3302, June 2024.

[69] Hongjia Zhai, Qi Zhang, Xiaokun Pan, Xiyu Zhang, Yitong Dong, Huaqi Zhang, Dan Xu, and Guofeng Zhang. Onlinepg: Online open-vocabulary panoptic mapping with 3d gaussian splatting. *arXiv preprint arXiv:2603.18510*, 2026.

[70] Jiazhao Zhang, Chenyang Zhu, Lintao Zheng, and Kai Xu. Fusion-aware point convolution for online semantic 3d scene segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4534–4543, 2020.

[71] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6672–6682, 2023.

[72] Shangzhan Zhang, Jianyuan Wang, Yinghao Xu, Nan Xue, Christian Rupprecht, Xiaowei Zhou, Yujun Shen, and Gordon Wetzstein. Flare: Feed-forward geometry, appearance and camera estimation from uncalibrated sparse views. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21936–21947, 2025.

[73] Xu Zhao, Wenchao Ding, Yongqi An, Yinglong Du, Tao Yu, Min Li, Ming Tang, and Jinqiao Wang. Fast segment anything. *arXiv preprint arXiv:2306.12156*, 2023.

[74] Xiaoyu Zhou, Jingqi Wang, Yuang Jia, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Ea3d: Online open-world 3d object extraction from streaming videos. *arXiv preprint arXiv:2510.25146*, 2025.

[75] Zhen Zhou, Yunkai Ma, Junfeng Fan, Shaolin Zhang, Fengshui Jing, and Min Tan. Eprecon: An efficient framework for real-time panoptic 3d reconstruction from monocular video. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2026–2033. IEEE, 2025.

[76] Lojze Zust, Yohann Cabon, Juliette Marrie, Leonid Antsfeld, Boris Chidlovskii, Jerome Revaud, and Gabriela Csurka. Panst3r: Multi-view consistent panoptic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5856–5866, 2025.

A More Implementation Details

A.1 Superpoint and Memory Details

For each frame, overlapping FastSAM masks are processed in descending order of area, with smaller masks overwriting larger ones to form a non-overlapping label map. The resulting 2D mask labels are projected onto the reconstructed 3D points through the established 2D–3D correspondences. Projected groups containing fewer than 15 points are discarded, and the remaining superpoints are independently used for query initialization without explicit cross-view merging. For query decoding, superpoint-level features at each scale are obtained by pooling the corresponding point features within each superpoint. After query decoding, highly overlapping predictions are suppressed using Matrix NMS. We retain at most 30 current instances for cross-chunk association. Unmatched historical instances remain in memory with their states unchanged. We cap the persistent memory at 300 historical instances, removing the instance unmatched for the longest time when the cap is exceeded.

A.2 Training Label Construction

To supervise mask prediction on RFM-reconstructed geometry, we construct point-level instance labels using the filtered per-frame 2D instance masks released with ScanNet. For ScanNetV2 and ScanNet200, these masks are obtained by projecting the 3D instance annotations of each scan onto its RGB-D frames and filtering unreliable projected regions. Since the masks are derived from the same 3D instance annotations, the same object is assigned a consistent instance ID across frames within a scan. Each RFM-reconstructed point records the frame index and pixel coordinates of its corresponding image observation, allowing us to assign it the instance ID from the filtered 2D instance mask at that pixel. Points mapped to invalid or unlabeled pixels are ignored. The resulting point-level instance labels are used only for training supervision.

A.3 Scene-level Alignment for Evaluation

MonoChunk3D predicts instance masks in the π^3 reconstructed coordinate system, while ground-truth instance labels are defined in the original annotated scene coordinate system. We therefore precompute a scene-level Sim(3) transformation to align each reconstructed scene to the annotated scene space for evaluation. Ground-truth depth and camera poses are used only to construct this evaluation transform and are not used by the model during inference.

Specifically, let $\boldsymbol{x}_i^{\pi^3}$ and $\boldsymbol{x}_i^{\mathrm{ann}}$ denote the coordinates of the i-th corresponding point in the π^3 reconstructed coordinate system and the annotated scene coordinate system, respectively. We estimate the Sim(3) transformation:

$$\boldsymbol{x}_i^{\mathrm{ann}} = s\boldsymbol{R}\boldsymbol{x}_i^{\pi^3} + \boldsymbol{t}, \tag{13}$$

where s, $\boldsymbol{R}$, and $\boldsymbol{t}$ denote the scale, rotation, and translation, respectively.

To construct 3D correspondences, we store each reconstructed point with its 3D coordinates, frame index, and pixel coordinates. The frame index and pixel coordinates identify the corresponding RGB-D pixel, which is unprojected using its depth value and camera intrinsics, and then transformed to the annotated scene coordinate system using the camera pose. We estimate the similarity transformation from these correspondences using RANSAC and the Umeyama algorithm. This transformation is fixed for each scene and used only to transfer predictions to the annotated scene space for evaluation.

B Additional Experimental Results

Sensitivity to Chunk Size. We further study the effect of chunk size during inference in Table 5. Our method consistently outperforms chunk-based ESAM-C across all chunk sizes, showing that the improvement is not tied to a specific chunk-size choice. When the chunk size is reduced from 12 to 3, our method shows smaller performance drops than ESAM-C across all metrics. Specifically, the mAP, AP_{50}, and AP_{25} of our method decrease by 1.9, 3.4, and 0.8, respectively, while ESAM-C drops by 3.1, 4.1, and 4.5. This indicates that MonoChunk3D is more robust to chunk-size variation. We use 12 frames by default to balance the range of local observations and online update frequency, which also achieves the highest mAP among the tested chunk sizes.

Table 5: Sensitivity to chunk size.

	Ours			ESAM-C [63]		
Chunk Size (frames)	mAP	AP_{50}	AP_{25}	mAP	AP_{50}	AP_{25}
3	21.3	41.7	67.1	14.3	30.6	54.3
6	22.9	44.5	67.7	16.8	33.8	58.4
12	23.2	45.1	67.9	17.4	34.7	58.8

Ablation on Association Strategy. Table 6 studies different strategies for cross-chunk association. Removing box overlap causes a large mAP drop from 23.2 to 14.5, indicating that spatial relations are important for cross-chunk association. Removing query similarity also reduces mAP to 20.8, showing that query similarity provides effective instance-level cues for distinguishing and merging cross-chunk instances. The full model achieves the best performance by combining query similarity and box overlap.

Table 6: Ablation on association strategy.

Method	mAP	AP_{50}	AP_{25}
Remove box overlap	14.5	31.1	57.5
Remove query similarity	20.8	42.0	66.0
Full model	**23.2**	**45.1**	**67.9**

Ablations Across Reconstruction Settings. We further evaluate MonoChunk3D under different reconstruction settings to investigate the contributions of historical context and spatial association. As shown in Table 7, removing history-aware queries reduces mAP from 19.8 to 17.2 with CUT3R and from 23.2 to 20.8 with our default π^3-based setting, showing that the historical context provided by persistent instance states remains effective across reconstruction frontends. Removing box overlap reduces mAP to 12.3 and 14.5, respectively, indicating that spatial overlap is an important association cue under both settings. Meanwhile, the π^3-based setting consistently outperforms the CUT3R-based setting, indicating that the reconstruction setting also affects segmentation performance. Since spatial overlap is computed from reconstructed geometry, its reliability remains sensitive to reconstruction and cross-chunk alignment quality.

Table 7: Ablations under different reconstruction settings on ScanNet200.

Reconstruction	Setting	mAP	AP_{50}	AP_{25}
CUT3R	w/o history-aware queries	17.2	35.8	57.8
	w/o box overlap	12.3	28.4	51.5
	Full model	19.8	39.5	59.8
π^3	w/o history-aware queries	20.8	41.8	65.2
	w/o box overlap	14.5	31.1	57.5
	Full model	**23.2**	**45.1**	**67.9**

Hyperparameter Sensitivity. We study the sensitivity to the loss weights λ_{con} and λ_{rank}, the support point threshold τ, and the ranking margin δ. As shown in Table 8, removing either auxiliary loss degrades performance, while the model remains stable around the default settings.

Sensitivity to Filtering and Memory Limits. We further study the minimum point threshold, current instance cap, and persistent memory cap on the ScanNet200 validation set. As shown in Table 9, both AP and average recall (AR) remain stable across the tested settings, with only minor degradation under more aggressive filtering or tighter caps. In terms of efficiency, query decoding, association, and memory update times range from 3.7 to 4.1, 0.9 to 1.0, and 2.2 to 2.4 ms/frame, respectively, across all tested settings. These results demonstrate that MonoChunk3D maintains stable accuracy and efficiency across these implementation settings. Additionally, under the default settings, the 15-point threshold filters 58.5% of projected groups, while the current-instance cap is exceeded in only 4.9% of chunks and truncates 2.6% of predictions. The memory contains 31.5 instances on average and at most 91.

Table 8: Sensitivity to hyperparameters on ScanNet200. Default values are shown in bold.

λ_{con}

Value	mAP	AP_{50}	AP_{25}
0	21.6	42.7	65.5
0.5	23.2	45.1	67.9
1.0	23.0	45.0	67.8

λ_{rank}

Value	mAP	AP_{50}	AP_{25}
0	22.0	43.0	66.1
0.5	23.2	45.1	67.9
1.0	23.3	45.3	67.9

τ

Value	mAP	AP_{50}	AP_{25}
0.2	22.7	44.4	67.3
0.4	23.2	45.1	67.9
0.6	23.0	44.8	67.6

δ

Value	mAP	AP_{50}	AP_{25}
0.3	22.9	44.6	67.7
0.5	23.2	45.1	67.9
0.7	23.2	45.0	67.9

Table 9: Sensitivity to filtering and memory limits on ScanNet200. All metrics are reported in %. AR denotes average recall over the same IoU thresholds as mAP. Default values are shown in bold.

Min. points

Value	mAP	AP_{50}	AP_{25}	AR
None	22.4	43.8	67.4	31.7
5	22.8	44.4	67.5	31.9
15	23.2	45.1	67.9	32.2
20	23.0	45.0	68.0	32.0
30	22.5	44.0	66.6	31.3

Current cap

Value	mAP	AP_{50}	AP_{25}	AR
20	23.3	45.2	67.2	31.9
30	23.2	45.1	67.9	32.2
40	23.2	44.9	68.2	32.2
60	23.1	44.8	68.1	32.1
80	23.0	44.8	68.2	32.1

Memory cap

Value	mAP	AP_{50}	AP_{25}	AR
60	22.4	43.2	65.8	31.3
70	22.8	44.4	67.5	32.0
80	23.1	44.9	68.1	32.4
90	23.2	45.1	67.9	32.2
300	23.2	45.1	67.9	32.2

Effect of 3D Feature Scale. We study the effect of using different 3D U-Net feature scales in current query initialization, historical support point retrieval, prototype construction and update, and prototype ranking supervision. For the all-scale setting, features from the four 3D U-Net levels are independently projected to a common dimension and averaged across scales. As shown in Table 10, the first-scale and fourth-scale features achieve comparable performance, while using all four scales provides no additional benefit. Since the query decoder already exploits multi-scale 3D features, we retain $\boldsymbol{F}_{t,1}^{3\text{D}}$ as a simple and effective choice.

Table 10: Effect of different 3D feature scales on ScanNet200.

3D Feature	mAP	AP_{50}	AP_{25}
$\boldsymbol{F}_{t,1}^{3\text{D}}$	**23.2**	**45.1**	**67.9**
$\boldsymbol{F}_{t,4}^{3\text{D}}$	23.1	44.8	67.8
All scales	22.7	44.2	67.3

Inference Time Decomposition. Table 11 reports the per-frame inference time of each component in our segmentation pipeline. The pipeline takes 43.0 ms per frame, with 22.0 ms for FastSAM [73] and 21.0 ms for online 3D segmentation. The 3D backbone is the main cost in the 3D segmentation stage, taking 10.6 ms per frame. In contrast, query initialization, history-aware propagation, and association with memory update take only 0.2 ms, 2.9 ms, and 3.3 ms, respectively. This shows that the proposed modules and cross-chunk association account for only a small portion of the inference time.

Visualization of Online Segmentation Stream. In Fig. 3, we present the chunk-wise reconstruction and segmentation stream of MonoChunk3D. The visualization shows that our method incrementally

Table 11: Inference time decomposition of our segmentation pipeline in ms/frame. FastSAM [73] is used for 2D mask generation, and the inference time of π^3 is reported separately in Table 1.

2D Mask	Online 3D Segmentation					Total
FastSAM	3D Backbone	Query Init.	Hist. Prop.	Decoder	Assoc./Update	
22.0	10.6	0.2	2.9	4.0	3.3	43.0

reconstructs and segments incoming RGB chunks, producing temporally consistent 3D instance masks over the accumulated scene.

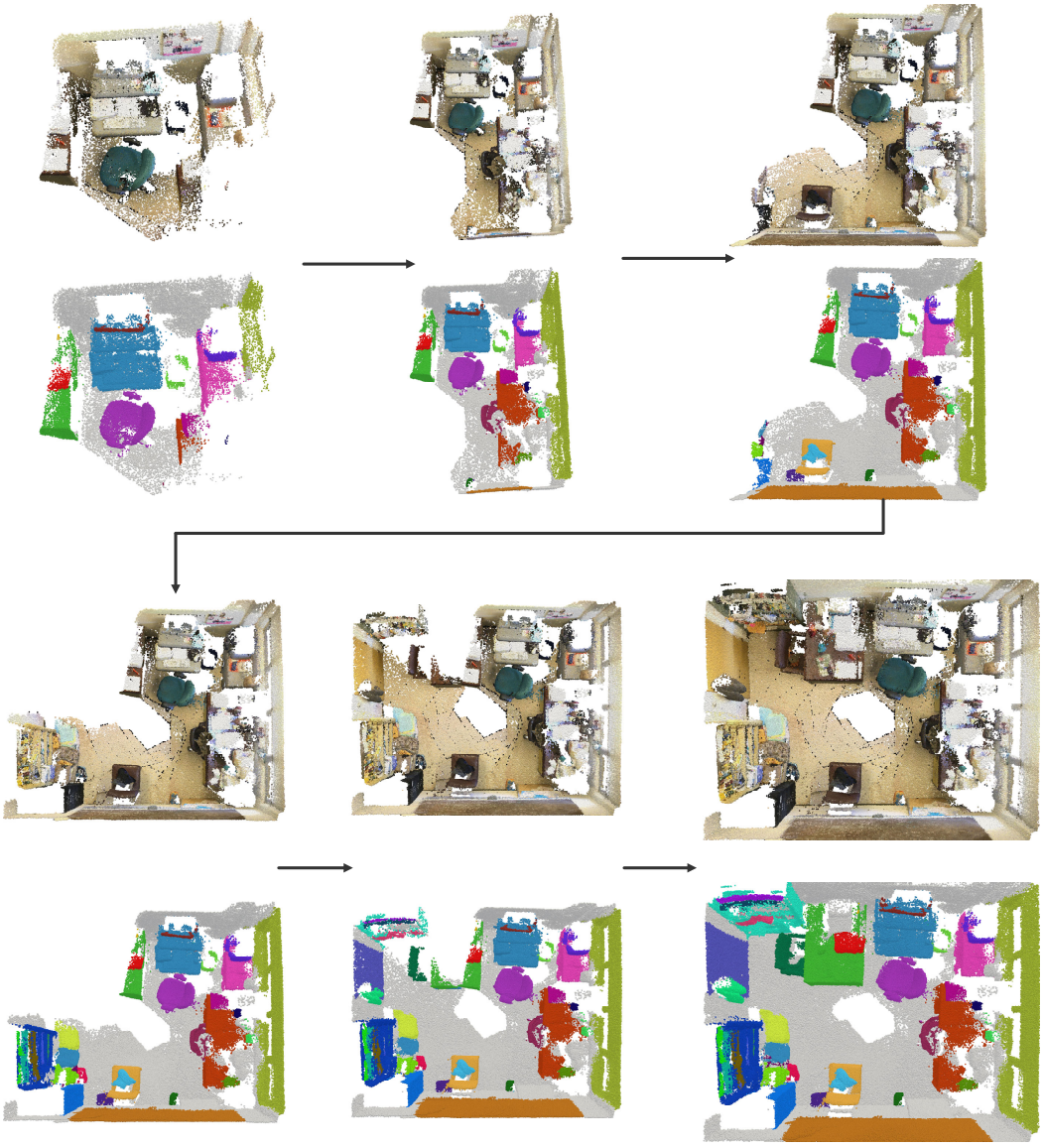


Figure 3: Visualization of chunk-wise reconstruction and online 3D instance segmentation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction are consistent with our contributions and are supported by the experimental results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our method in Section 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: Our paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The detailed experimental settings are provided in Section 4 and the supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We use publicly available datasets and release the code with default configuration.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Experimental details are provided in Section 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Reporting error bars would require substantial computational cost. In repeated runs with the same hyperparameters, we observed no significant variation in validation metrics.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please refer to Section 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We adhere to the NeurIPS Code of Ethics in all our experiments.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper highlights the potential positive societal impact of 3D instance segmentation for robotics in the introduction. We do not anticipate specific negative societal consequences from this work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work does not release high-risk models or datasets that require special safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original datasets, models, and codebases used in our experiments and respect their licenses and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: This research does not use LLMs as part of the core methodology.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.