MoonSeg3R: Monocular Online Zero-Shot Segment Anything in 3D with Reconstructive Foundation Priors

Supplementary Material

The supplementary materials provide 1) more implementation details (Sec. A); 2) A component analysis (Sec. B); 3) More qualitative examples for visualization and comparison (Sec. C). We will release the code and pretrained models.

A. Additional Implementation Details

Query Index Map. The Query Index Memory (QIM) introduced in Sec. 3.4 relies on query index map $\mathcal{M}$, which is a mapping between a set of spatial keys and their associated query indices.

Map Update. To ensure computational efficiency, we dynamically update the map to maintain a sparse key structure. Let $\mathcal{M}_{t-1} \in \{0,1\}^{n^k_{prev} \times n^q_{prev}}$ denote the map storing n^q_{prev} previous contextual queries at n^k_{prev} existing spatial keys. After receiving new queries and updating the global query bank $\mathcal{Q}_t \in \mathbb{R}^{(n^q_{prev}+n^q_t)\times d}$, we first extend the dimension of $\mathcal{M}_{t-1}$ to $\{0,1\}^{n^k_{prev}\times(n^q_{prev}+n^q_t)}$ by padding with zeros. Then, we compute the Euclidean distance between the incoming spatial keys and the existing keys. Inspired by [55], if an existing spatial key lies within a distance threshold $\delta_{\mathcal{M}}$ of one or more new keys, we update its position to the centroid of the matched new keys, and take the union of all corresponding query indices. Otherwise, new spatial keys that do not match any existing key are appended as new rows in $\mathcal{M}_t$.

Map Rasterization. We render the query index map as a point cloud using the PyTorch3D point rasterizer [36]. To model the occlusions, we strictly rasterize the nearest point at each pixel.

Other Implementation Details. For the query index map described above and in Sec. 3.4, we set the distance threshold $\delta_{\mathcal{M}}$ to 0.3 for merging spatial keys. As shown in Sec. 4.1, MoonSeg3R is trained on ScanNet image data. After training, we directly test the model on ScanNet200 and SceneNN.

B. Component Analysis

B.1. Speed Analysis

MoonSeg3R. The total inference time for MoonSeg3R is 321 ms per frame, including 121 ms for reconstruction and segmentation and 200 ms for 2D VFM mask generation, which corresponds to approximately 3.1 FPS. Excluding the RFM CUT3R and VFM CropFormer, our segmentation pipeline requires only 55 ms. Since the feature refinement network ψ is used to obtain reference features for training in Eq. 5 and Eq. 6, it is not involved in the inference process. The breakdown of the 55 ms pipeline is as follows:

Component	Time (ms)
Feature Extraction (DINO)	6
Query Refinement	18
Memory Retrieval	2
Memory Update	4
Intra-frame Merging	3
Cross-frame Association	22
Total	55

Table A1. **Component Analysis.** The time cost of individual component in MoonSeg3R.

Feature Extraction. We extract 2D semantic features $\mathbf{F}^{2d}$ using DINO in a single feed-forward step for 6 ms.

Query Refinement. This module enhances query features by fusing information from the current frame and the retrieved memory context. This process consumes 18 ms due to attention mechanisms and feature fusion operations.

Memory Retrieval. The retrieval step of the memory involves rasterizing the query index map $\mathcal{M}$ and indexing the global query bank $\mathcal{Q}$. Thanks to the highly efficient point rasterization algorithm and the sparsity of spatial keys, the rasterization process takes only 2 ms, which brings negligible time cost and allows for efficient contextual injection.

Memory Update. The memory update process is relatively more computational intensive compared to retrieval, as it performs extensive search over all existing keys to merge/extend each new spatial key, in order to maintain map sparsity.

Intra-frame Merging. As detailed in Sec. 3.5, to resolve the problem of oversegmentation from 2D VFM, we merge masks within the current frame based on the query feature map.

Cross-frame Association. After intra-frame merging, we associate the merged masks from the current frame with the existing masks via bipartite matching. This uses a combined metric of query similarity, bounding box IoU and our novel state distribution token similarity. As the video frame increases, the number of tracked instances grows, resulting in an average processing time of 22 ms.

OnlineAnySeg. Previous work OnlineAnySeg consists of feature extraction, VoxelHashing updating and MaskGraph clustering. The workflow is more time-consuming than the reported results in their paper [45] because the time-

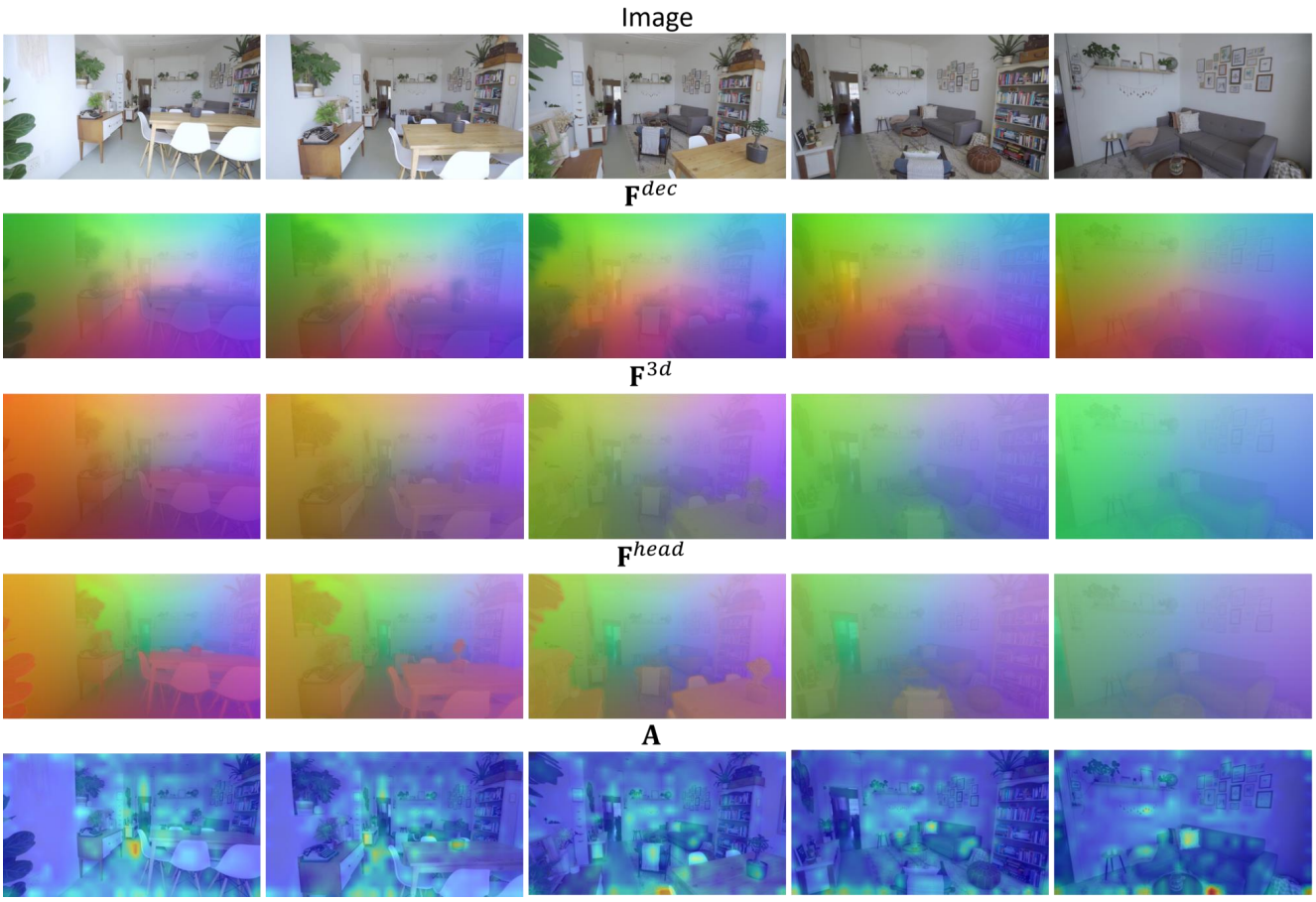

Figure A1. **Intermediate Representation Visualization**. We visualize the intermediate representations within CUT3R, specifically decoder features $\mathbf{F}^{dec}$, modulation features $\mathbf{F}^{mod}$, head features $\mathbf{F}^{head}$ and state attention maps $\mathbf{A}$ (from top to bottom). The decoder features $\mathbf{F}^{dec}$ contain local depth information. $\mathbf{F}^{mod}$ demonstrates world-coordinate consistency, as the modulation layer performs an implicit rigid transformation conditioned on the pose token. The head features $\mathbf{F}^{head}$ exhibit the sharpest geometric details and boundaries, since they are directly supervised by the pointmap regression loss. The state attention $\mathbf{A}$ consistently highlights specific semantic instances (e.g., pillow, plant), but remains spatially sparse.

consuming feature extraction and graph traversing procedure is not fully covered, which has been pointed out in a follow-up work [22] and a Github issue [9]. For instance, to extract per-instance features, OnlineAnySeg generates multi-scale crops for each instance and processes them individually with CLIP [35], which can take over 1 second for a single keyframe.

With the 200 ms cost of the 2D VFM CropFormer [34] included, OnlineAnySeg requires a total of 3200 ms per frame. In contrast, our method, MoonSeg3R, takes only 321 ms, making it approximately 10 times faster even while performing simultaneous reconstruction and segmentation.

B.2. Design Choice

Geometric Features. In Sec. 3.1, we extract geometric features $\mathbf{F}^{3d}$ from CUT3R. To differentiate from the 2D semantic information $\mathbf{F}^{2d}$, $\mathbf{F}^{3d}$ must encode 3D spatial structure in world coordinates. We analyze different feature sources from CUT3R and visualize them in Fig. A1.

Decoder Features $\mathbf{F}^{dec}$. Features from the image decoder $\mathcal{D}_I$ show similar color pattern in the image plane even as the camera moves, as it primarily contains depth information in the camera's self coordinate system.

Modulation Features $\mathbf{F}^{mod}$. These features are extracted from the modulation layer, which is between the decoder and the world-coordinate prediction head. This layer uses self-attention conditioned on the pose token to perform an implicit rigid transformation of $\mathbf{F}^{dec}$ into world coordinates. Consequently, $\mathbf{F}^{mod}$ show consistent patterns that vary correctly with the changing 3D position.

Head Features $\mathbf{F}^{head}$. CUT3R uses a DPT head to predict world pointmaps. The features $\mathbf{F}^{head}$ taken immediately before the final prediction layer contain the most explicit

Figure A2. State distribution similarity in a long loop.

3D information. Compared to $\mathbf{F}^{mod}$, $\mathbf{F}^{head}$ exhibits more distinct geometric boundaries and fine-grained spatial details, as it is directly supervised by the pointmap regression loss.

Despite $\mathbf{F}^{mod}$ and $\mathbf{F}^{head}$ both showing spatially varied geometric structure in world coordinates, we empirically find that $\mathbf{F}^{mod}$ provides better stability during training. Our hypothesis is that $\mathbf{F}^{head}$ is highly sensitive to high-frequency geometric noise due to the regression supervision, whereas $\mathbf{F}^{mod}$ offers a smoother and more robust representation. Therefore, we select $\mathbf{F}^{mod}$ as our geometric feature $\mathbf{F}^{3d}$.

State Distribution Token. In the last row of Fig. A1, we visualize the attention map that state representation allocates to the image. The visualizations convey that, though the state consistently attends to specific objects (*e.g.*, the plant pot on the table, the pillow on the sofa), the attention distribution is generally sparse, making it challenging to directly obtain pixel-wise association across frames. In Sec. 3.5, our proposed state distribution token aggregates this sparse signal into a robust instance-level representation. Our intuition is that the subset of state tokens responsible for encoding a specific instance region can be more stable and consistent. By performing a spatial masked summation over state attention map, we effectively identify the state tokens with high attention scores as the ones responsible for tracking the object within the mask, which produces a consistent state distribution token that successfully mitigates the correspondence ambiguity for inference-time association.

State Distribution Consistency. In Fig. A2, we show the same sofa in a long loop of 166 keyframes. Although it leaves the view after frame 18 and reappears at frame 150, its masks are still correctly matched via the highest state distribution similarity among all instances.

C. Visualization

Mask Prediction Visualization. In Fig. A3, we show examples of the predicted masks from MoonSeg3R with the input VFM masks. When VFM oversegments an object, our method consistently recovers the complete instance mask. This illustrates that the updated queries successfully capture comprehensive instance-level information, and the reference features have been effectively refined to be instance-discriminative, demonstrating the efficacy of our proposed self-supervised learning framework and spatial-semantic distillation strategy (see Sec. 3.3). Additionally, this property is critical for effective intra-frame merging in Sec. 3.5, as it ensures that queries corresponding to different parts of the same instance are mutually similar.

Qualitative examples. More qualitative examples of our method on ScanNet200 and SceneNN are provided in Fig. A4. Consistent with Fig. 3, our method merges masks more robustly on predicted geometry from CUT3R than OnlineAnySeg-M, which highly depends on precise geometry information to construct its VoxelHashing representation.

Figure A3. **Mask Prediction Visualization**. From top to bottom, we respectively visualize the original images, the input VFM masks, the predicted masks and the ground truth instance masks. While the VFM mask represents part of an oversegmented instance, its query correctly segments the whole instance in the predicted mask, validating that the self-supervised query learning and spatial-semantic distillation strategy mitigates the oversegmentation problem in 2D VFMs.

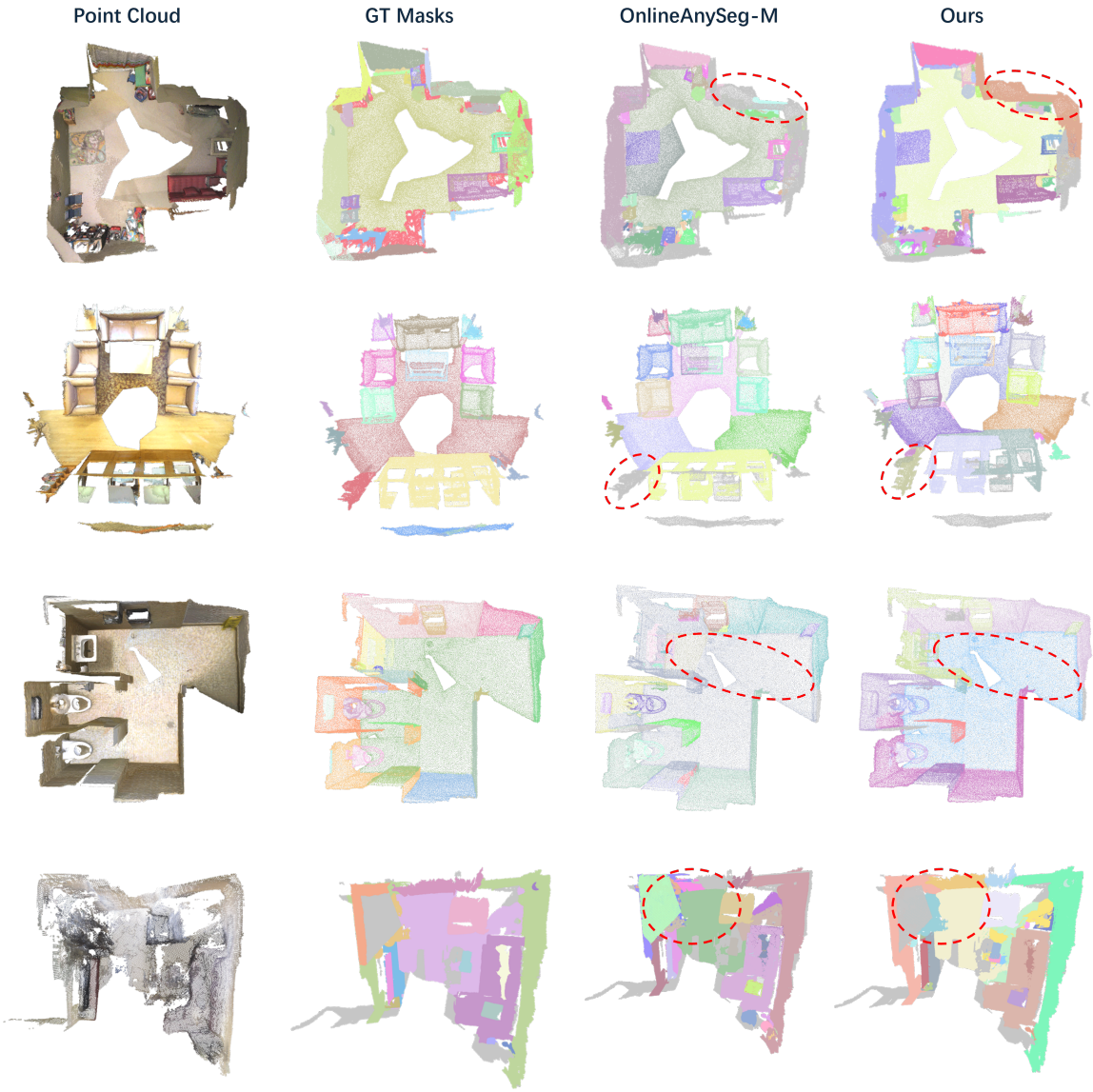

Figure A4. **Qualitative Comparison.** Qualitative examples of OnlineAnySeg-M and our method on ScanNet200 and SceneNN sequences. These results visually demonstrate that MoonSeg3R achieves superior instance segmentation. In contrast, OnlineAnySeg-M tends to fail in associating masks, which leaves significant unsegmented areas, as shown in the red dashed circles. The segmentation results are unprojected to ground truth point cloud for visualization.