EvObj: Learning Evolving Object-centric Representations for 3D Instance Segmentation without Scene Supervision

Supplementary Material

5.1. More Details of Discerning and Evolving Object Candidates

Network Architecture: As shown in Figure 9, the discerning module adopts SparseUNet [11]. The encoder performs spatial resolution reduction through 4 stages, and decoder enhances features with upsampling. Two parallel convolution layers are concatenated at the end of UNet to produce the voxel-wise scores for foreground and background. K_{size}, S, and ch in Figure 9 denote the kernel size, stride, and feature channels for sparse convolutions, respectively. Symbol $\oplus$ represents the concatenation of voxel features.

Pretraining: The discerning module is pretrained on objects from ShapeNet dataset. To simulate realistic occlusions in real-world scenarios, we render depth images for each 3D object from 12 views and randomly select and fuse $3 \sim 6$ views to obtain a partial point cloud. Subsequently, we generate horizontal and vertical planes in a unit cube to simulate the floor and wall, which are treated as backgrounds. Additionally, the augmented object point clouds are randomly scaled from 0.8 to 1.5.

The pretraining of the discerning module is a point-wise binary segmentation task for distinguishing object and background points. Therefore, we adopt the cross-entropy loss function and Adam optimizer with a learning rate of 1e-3 with a weight decay of 1e-4. The training takes a batch size of 32 with a voxel size of 0.05 and lasts for 20 epochs.

Evolving Phase: During evolving, we gather object candidates scored by the object-centric network during reinforcement learning. Specifically, as the object-centric network is a reconstruction model, we feed points of each candidate into the network. The candidate is regarded as a valid object only when the Chamfer Distance between the input points and recovered shape is below 0.1.

The evolving is conducted every 100 epochs. We train the discerning module with the accumulated object candidates, and adopt the Adam optimizer with a learning rate of 1e-3 and a weight decay of 1e-4. Training epoch is set as 40 with a batch size of 32. Additionally, data augmentation is applied, including rotation around the z-axis from -180° to 180° and scaling within the range of 0.9 to 1.1. The accumulated object candidates will be discarded once used.

5.2. More Details of Completing Object Candidates

Object candidates are completed through a pretrained completion model, which adopts AdaPointTr [58] in our main experiments. However, the vanilla AdaPoinTr is trained with 3D objects in canonical pose, which is hard to apply to real-scanned data. Thus, we keep the model architecture and train it from scratch using our training data.

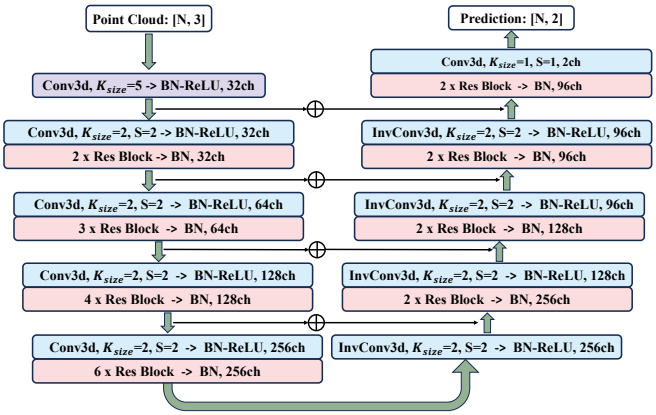

Figure 9. Network Structure of Discerning Module

Data Preparation: We randomly combine depth scans generated in Section 5.1 from 2∼4 views, and downsample to 1024 points as input data. The ground truth full 3D shape is constructed by randomly sampling 1024 points from the object mesh surface.

Training Process: The training is kept the same as AdaPoinTr. Chamfer Distance is taken as the loss function. We train the completion model by AdamW with a learning rate of 1e-4 and weight decay of 5e-4 for 300 epochs.

5.3. Training and Evaluation on ScanNet

Following GrabS [62], we adopt a Mask3D network structure with the SparseUNet as our backbone to extract voxel features. The segmentation head is a Transformer decoder, and the policy network is a cross-attention with two separate MLPs for predicting actions and state value. The object discovery networks are trained with both segmentation and PPO loss functions, which are exactly keep same as GrabS. Training on the ScanNet training set lasts for 600 epochs with a batch size of 8, and the AdamW optimizer is configured with a fixed learning rate of 1e-4. With the successful usage of superpoints in 3D unsupervised learning [60, 61, 63], we have also incorporated them into our framework. For evaluation, all masks predicted by the segmentation head are treated as chair candidates, and the Average Precision (AP) for chairs is computed on the validation and online hidden test sets of ScanNet in Table 1 & Table 2. Figure 10 shows the differences between our EvObj and the baseline on ScanNet validation set. Experimental results in-

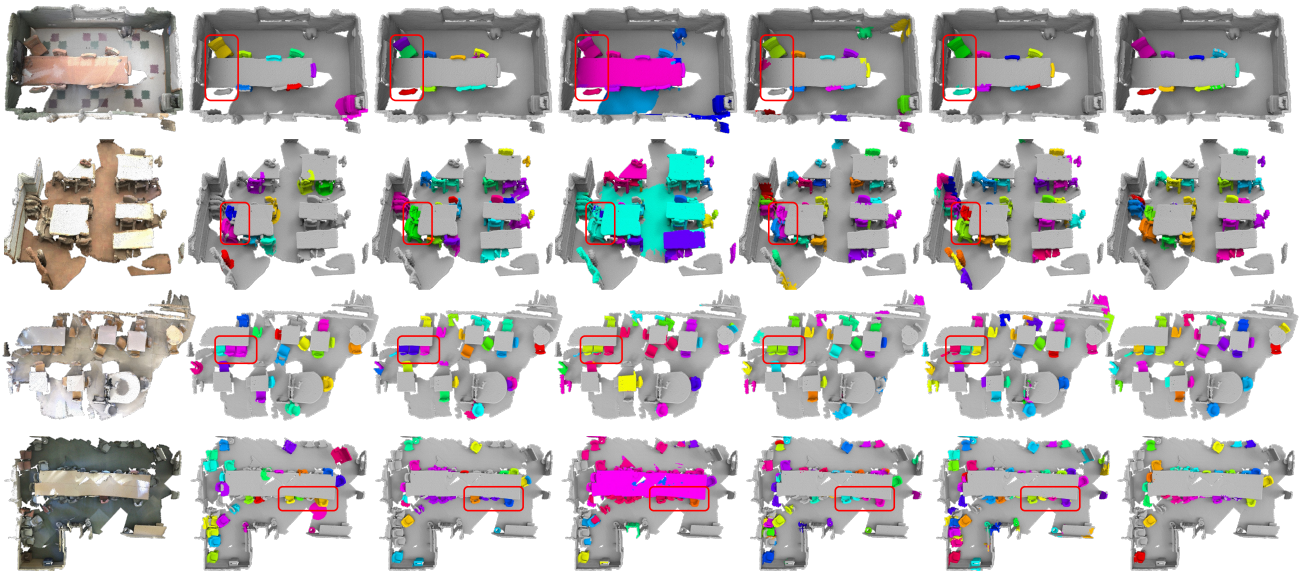

Figure 10. More qualitative results on ScanNet validation set. Red boxes highlight the differences.

dicate that our method is more effective at addressing cases where multiple chairs are closely placed.

5.4. Training and Evaluation on S3DIS

The cross-dataset evaluation is conducted on the S3DIS dataset using the well-trained models from the ScanNet training set, which is the same as Part2Object and GrabS. Tabs. 9 to 14 present the segmentation results on six areas in S3DIS, and more qualitative results are shown in Figure 11, where our EvObj significantly outperforms existing baselines.

Table 9. Quantitative results of our method and baselines on the S3DIS-Area1 [8].

		AP(%)	AP@50(%)	AP@25(%)
Unsupervised	UnScene3D [38]	33.6	63.8	85.4
	Part2Object [40]	27.1	55.4	77.5
	EFEM [26]	19.1	48.6	54.7
	GrabS-VAE [62]	45.5	68.6	73.2
	GrabS-Diffusion [62]	47.8	70.9	75.7
	EvObj (Ours-VAE)	50.7	77.6	86.7
	EvObj (Ours-Diffusion)	**52.8**	**82.2**	**90.6**

Table 10. Quantitative results of our method and baselines on the S3DIS-Area2 [8].

		AP(%)	AP@50(%)	AP@25(%)
Unsupervised	UnScene3D [38]	3.2	6.0	10.5
	Part2Object [40]	2.8	6.2	8.8
	EFEM [26]	1.1	2.9	9.2
	GrabS-VAE [62]	**6.4**	**10.2**	17.2
	GrabS-Diffusion [62]	5.3	8.1	13.3
	EvObj (Ours-VAE)	5.7	9.5	**20.7**
	EvObj (Ours-Diffusion)	5.3	9.9	16.7

Table 11. Quantitative results of our method and baselines on the S3DIS-Area3 [8].

		AP(%)	AP@50(%)	AP@25(%)
Unsupervised	UnScene3D [38]	37.6	58.2	83.6
	Part2Object [40]	38.9	63.5	81.4
	EFEM [26]	29.0	58.3	64.2
	GrabS-VAE [62]	59.5	78.8	80.2
	GrabS-Diffusion [62]	51.4	67.0	67.0
	EvObj (Ours-VAE)	66.0	86.0	**91.8**
	EvObj (Ours-Diffusion)	**70.0**	**91.3**	91.5

Table 12. Quantitative results of our method and baselines on the S3DIS-Area4 [8].

		AP(%)	AP@50(%)	AP@25(%)
Unsupervised	UnScene3D [38]	23.9	49.1	70.8
	Part2Object [40]	26.8	57.4	75.3
	EFEM [26]	15.1	36.5	49.1
	GrabS-VAE [62]	39.2	66.4	73.4
	GrabS-Diffusion [62]	39.4	64.8	68.0
	EvObj (Ours-VAE)	41.7	70.5	85.6
	EvObj (Ours-Diffusion)	**42.0**	**72.3**	**89.4**

Table 13. Quantitative results of our method and baselines on the S3DIS-Area5 [8].

		AP(%)	AP@50(%)	AP@25(%)
Unsupervised	UnScene3D [38]	42.6	63.4	80.3
	Part2Object [40]	30.0	50.5	76.4
	EFEM [26]	14.9	35.7	45.3
	GrabS-VAE [62]	46.4	66.2	73.8
	GrabS-Diffusion [62]	44.2	58.0	62.6
	EvObj (Ours-VAE)	55.1	77.1	86.2
	EvObj (Ours-Diffusion)	**60.6**	**82.8**	**92.1**

5.5. Construction of Multiclass Synthetic Dataset

Similar to GrabS [62], we generate 5000 static scenes that contain 4 to 8 objects. We construct scenes using six object categories from ShapeNet, namely *chair, sofa, telephone,*

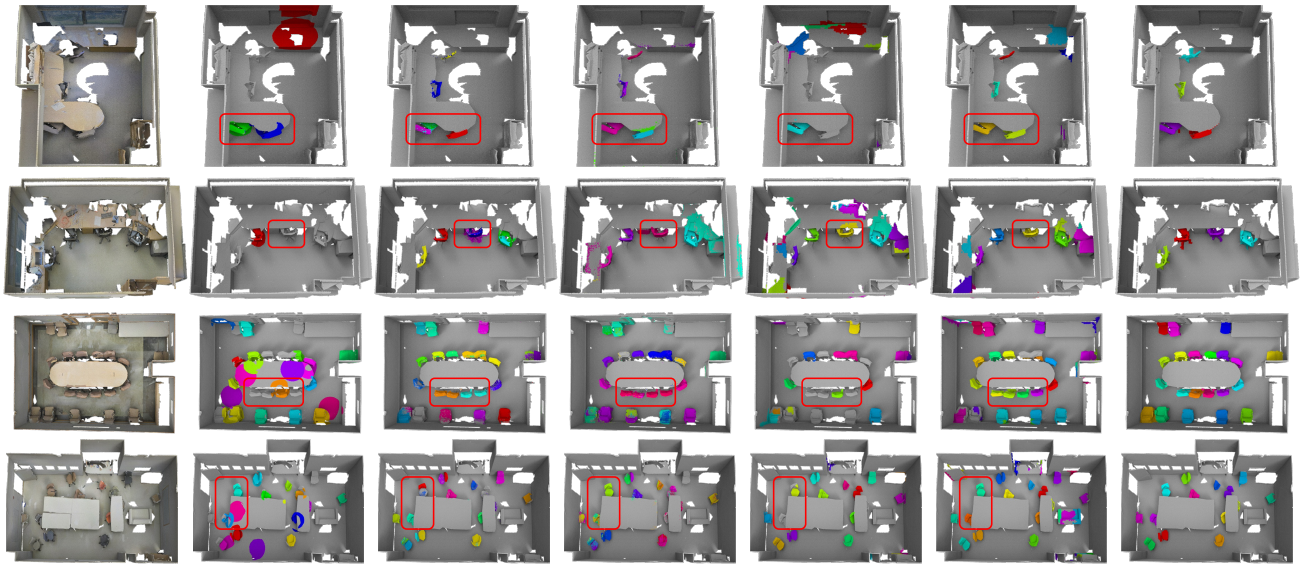

Figure 11. More qualitative results on S3DIS validation set. Red boxes highlight the differences.

Table 14. Quantitative results of our method and baselines on the S3DIS-Area6 [8].

		AP(%)	AP@50(%)	AP@25(%)
Unsupervised	UnScene3D [38]	41.3	70.9	92.3
	Part2Object [40]	26.5	57.4	83.0
	EFEM [26]	18.3	45.2	53.1
	GrabS-VAE [62]	52.1	80.4	84.3
	GrabS-Diffusion [62]	47.1	74.5	76.2
	EvObj (Ours-VAE)	**58.8**	**84.4**	**93.6**
	EvObj (Ours-Diffusion)	53.9	77.2	91.1

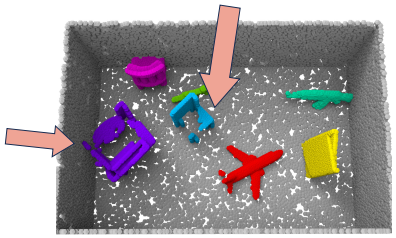

Figure 12. Generated Synthetic Dataset

airplane, rifle, cabinet. Specifically, we generate 4000 and 1000 3D indoor rooms for training and testing, respectively. To prevent data leakage, 3D objects in training scenes are sampled from the ShapeNet validation set, while those in test scenes are selected solely from the ShapeNet test set.

Occlusion Simulation: In real-world scenarios, point clouds suffer from incompleteness due to self-occlusion or limitations of scanning angles. For example, when a sofa is placed against a wall on the floor, the scanning device fails to capture the back and bottom surfaces of the sofa. To better simulate real-world scenarios, we generated incomplete point clouds through multi-view projection. Specifically, the camera is positioned at a distance of 2 units from the object, with a randomly sampled pitch angle ranging from 0° to 30° and an azimuth angle from -180° to 180°. The camera FOV is set to 20°, and we randomly generate 12 view configurations. Depth maps are rendered under these configurations and then projected to 3D, obtaining partial object point clouds. The incomplete object is made by randomly combining 2∼4 partial point clouds. Figure 12 demonstrates the incomplete objects in the generated synthetic dataset.

The aspect ratio of the ground plane in each scene is uniformly sampled between 0.6 and 1.0. All objects share the same size, which is set as 1. To simulate a real scene, we rotate each object around the vertical z-axis from -180° to 180°, then put them on the generated ground. The four sides of the scene are enclosed by generated walls. The resulting point clouds contain only the coordinates without color information. Each scene has 20000 points in total.

Object Placement: To avoid object overlap, we sequentially place objects into the scene and perform collision detection using their bounding boxes against those of the already placed objects. If a collision occurs, the position of objects will be adjusted with a maximum of 1000 attempts. In case no suitable position for the object is found after exceeding 1000 attempts, the scene will be discarded.

5.6. Training and Evaluation on Synthetic Dataset

Training Process: Training hyperparameters for synthetic dataset are aligned with those of ScanNet. The query number in Mask3D is set as 10 due to the maximum number of objects in this synthetic dataset being 8. We train the model for 200 epochs with a batch size of 10, using the AdamW optimizer configured with a fixed learning rate of 1e-4.

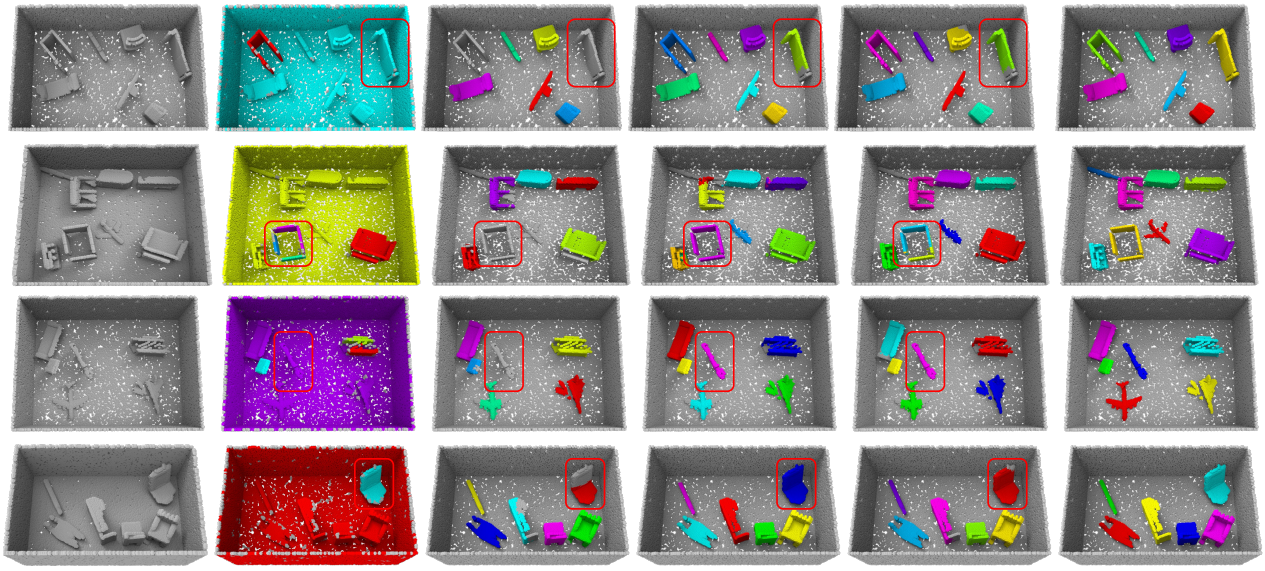

Figure 13. More qualitative results on the test set of our Synthetic dataset. Red boxes highlight the differences.

All object categories *chair, sofa, telephone, airplane, rifle, cabinet* are evaluated on the test set of our synthetic dataset. More visualizations of the results are provided in Figure 13. Benefiting from the object candidate completion module, our model significantly outperforms baseline methods when handling incomplete objects.

5.7. Training and Evaluation on ScanNet++

ScanNet++: It is a high-quality 3D indoor scene dataset. Compared with ScanNet, ScanNet++ [54] exhibits significantly higher intra-class diversity across object categories.

To better validate the robustness and generalizability of our method, we conduct experiments under a cross-dataset validation setup. Specifically, we reuse the well-trained segmentation models trained on ScanNet and perform a class-agnostic evaluation on all chairs in the validation set of ScanNet++, which contains 50 3D scenes.

Analysis: Quantitative results are shown in Table 15. Our method significantly surpasses all unsupervised baselines on the validation set. The reason is that our method can mitigate the domain gap between synthetic scenes and real scenes and thus better handle intra-class diversity.

Table 15. Quantitative results of cross dataset validation on validation set of ScanNet++.

		AP(%)	AP@50(%)	AP@25(%)
Unsupervised	Part2Object [40]	13.7	31.8	57.2
	UnScene3D [38]	14.4	35.6	65.1
	GrabS [62]	19.8	35.6	48.1
	EvObj	**42.0**	**69.3**	**82.7**

5.8. Evaluation on Object Candidate Discerning Module

Table 7 reports object candidate coverage for our EvObj and GrabS, demonstrating that EvObj discovers more objects in 3D scenes during RL training, thus yielding superior segmentation performance.

Discovered object candidates are counted as true objects if their IoU with ground truth masks exceeds a threshold. Table 16 presents additional comparative results across different IoU thresholds. After training more than 100 epochs, our EvObj consistently outperforms GrabS [62] in all cases, while EvObj without the discerning module exhibits a significant performance drop, validating the effectiveness of our object candidate discerning module in object discovery.

Table 16. The sufficiency (%) of qualified object candidates discovered over epochs on the ScanNet training .

	Epoch	100	200	300	400	500
IoU 50%	GrabS [62]	55.2	59.4	61.4	62.5	62.6
	EvObj without f_{dis_evo}	65.3	68.6	69.6	69.2	69.6
	EvObj with f_{dis_evo}	66.3	70.5	71.4	71.7	71.6
IoU 60%	GrabS [62]	47.9	51.3	52.6	53.5	53.7
	EvObj without f_{dis_evo}	52.5	54.8	55.3	55.5	56.2
	EvObj with f_{dis_evo}	55.1	59.2	60.9	61.2	61.3
IoU 70%	GrabS [62]	37.5	40.0	40.8	41.6	41.7
	EvObj without f_{dis_evo}	38.0	38.4	38.6	38.7	39.1
	EvObj with f_{dis_evo}	41.3	45.0	46.7	47.5	46.9
IoU 80%	GrabS [62]	23.4	24.5	24.8	25.8	25.8
	EvObj without f_{dis_evo}	21.4	22.1	22.0	21.6	21.6
	EvObj with f_{dis_evo}	24.8	29.0	30.2	31.4	30.8

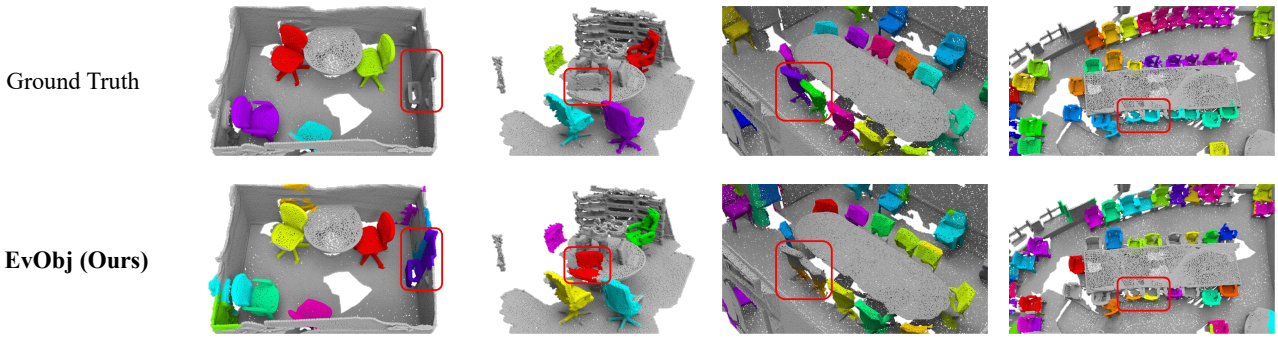

Figure 14. Failure cases of our method.

Table 17. Ablation results on the validation set of ScanNet.

	AP(%)	AP@50(%)	AP@25(%)
Ablations on Discerning Module			
EvObj$_{SparseUNet}$[11]	**55.0**	**76.9**	**88.2**
EvObj$_{PointNet++}$[35]	54.5	76.5	87.4
EvObj$_{PointTransformer}$[64]	54.7	76.5	87.8
Ablations on Chamfer Distance δ_c			
$\delta_c = 0.08$	48.0	67.0	78.9
$\delta_c = 0.09$	53.2	74.9	86.0
$\delta_c = 0.10$	**55.0**	**76.9**	**88.2**
$\delta_c = 0.11$	52.5	75.6	88.1
$\delta_c = 0.12$	51.0	73.8	87.7

5.9. Ablation Study on Object Candidate Discerning Module

For training the discerning module, we adopt SparseUNet [11] as the backbone network and further conduct an ablation study to investigate the impact of backbone selection for this module. Specifically, we evaluate two alternative backbones: PointNet++ [35], and Point Transformer [64], which yield similar performance as shown in Table 17.

5.10. Ablation Study on Chamfer Distance

During reinforcement learning, the reward is defined by the reconstruction error of the object-centric network, which is measured by the Chamfer Distance δ_c between the input and output point clouds of the object-centric network. Given the critical role of rewards in guiding the agent, we perform five ablation studies on the threshold δ_c. As shown in Table 17, the results are robust to the choice of this hyperparameter, and we set δ_c as 0.10 in our main experiments.

5.11. Time and Memory Costs

Training the entire pipeline of EvObj takes 104 hours. Specifically, training the discerning module takes 0.5 hours with 4 GB GPU memory, the completion module takes 6 hours with 19 GB, the object-centric net takes 29 hours with 8 GB, and the policy network takes 69 hours with 20 GB.

While the total training time of our method is slightly longer than the baseline GrabS [62], which takes 91 hours, our approach maintains the same inference speed. On average, it processes one scene in 0.063 seconds with 5 GB of GPU memory. All experiments are conducted on a single NVIDIA RTX 3090 GPU with an AMD R9 5900X CPU.

Table 18. Quantitative results of Plug and Play on the validation set of ScanNet.

	AP(%)	AP@50(%)	AP@25(%)
UnScene3D [38]	37.2	62.4	79.2
UnScene3D$_{f_{dis_evo}}$	**41.1**	**65.7**	**79.4**
Part2Object [40]	34.4	56.8	73.9
Part2Object$_{f_{dis_evo}}$	**39.2**	**64.5**	**81.8**

5.12. Plug-and-Play Mask Refinement

To validate the generalizability of our method, we apply our discerning module as a post-processing (plug-and-play) technique to UnScene3D [38] and Part2Object [40], refining their predicted masks. We conduct experiments on the chair category of the validation set of ScanNet, treating all predicted masks as chairs. Table 18 shows that both baselines can be significantly improved, highlighting the versatility of our method.

5.13. Visualizations of Failure Cases

Our method EvObj presents two types of failure cases as shown in Figure 14. For the first type, it yields erroneous segmentation for objects with similar shape to the target objects. For instance, it incorrectly segments parts of walls and a trash bin as a single chair. The second type involves segmenting only object parts rather than whole instances. For example, it only segments the legs of the chair instead of the entire chair. We attribute this issue to the completion module.

5.14. Pipeline Complexity and Error Accumulation

As we introduce two additional modules, namely the discerning module and completion module, the complexity of the overall pipeline is increased. We analyze whether this

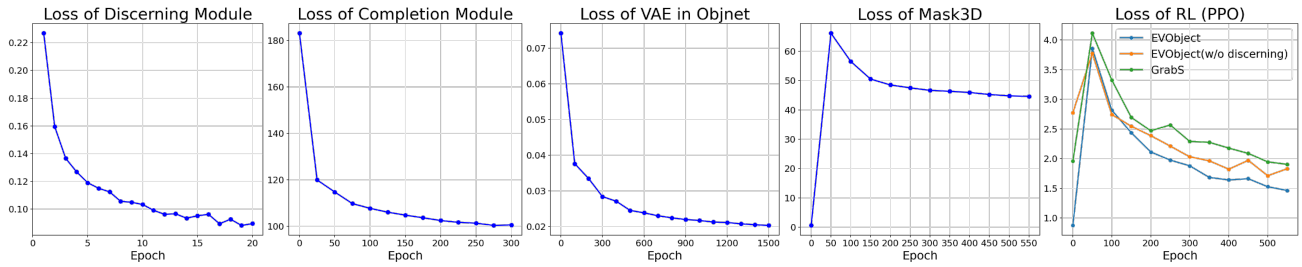

Figure 15. Training losses of all modules in our pipeline.

introduces difficulties in training and leads to error accumulation between modules.

Figure 15 shows the training losses for the discerning, completion, object-centric, segmentation, and policy modules, highlighting the stable convergence and ease of training achieved by our modular pipeline.

We analyze error accumulation by applying the trained agent on training scenes in ScanNet and then obtaining object masks from the discerning, completion, and object-centric networks. Specifically, for the discerning module, we compute the IoU between the predicted masks and the Ground Truth masks. For the completion module, we compute the point distances from scene point clouds to the completed object point clouds, and filter them by a threshold of 0.1, and finally get masks from the completion module to compute IoU with Ground Truth. For object-centric networks, we also compute the point distance from scene points to the generated mesh, then get masks by filtering with a 0.1 threshold. This follows the same procedure as the completion module for matching and IoU calculation.

The average IoUs against ground truth are 42.0, 50.7, and 46.9, respectively, indicating that error accumulation is not significant.