SlotLifter: Slot-guided Feature Lifting for Learning Object-centric Radiance Fields
Supplementary Material

Yu Liu[⋆†1,2], Baoxiong Jia[⋆1], Yixin Chen[1], and Siyuan Huang[1]

[1] State Key Laboratory of General Artificial Intelligence, BIGAI
[2] Department of Automation, Tsinghua University
[⋆]Equal contribution [†]Work done as an intern at BIGAI
https://slotlifter.github.io

A Implementation Details

A.1 SlotLifter

Architecture Design

Scene Encoding We employ a U-net-like encoder E_Φ with ResNet34 [3] to extract 2D image features, similar to IBRNet [11]. This architecture truncates after `layer3` as the encoder and adds two up-sampling layers with convolutions and skip-connections as the decoder. Instead of extracting two sets of feature maps for coarse and fine networks as IBRNet, we extract a shared feature map. In addition, we concatenate multi-view images with their corresponding ray directions and camera positions to provide more spatial information, enabling slots to learn 3D information from 2D multi-view features via Slot-Attention. Given extracted feature maps, we obtain slots via Slot-Attention and 3D point features via feature lifting described in Sec. 3.2. We add the point positional embedding $\boldsymbol{E}_p$ in Eq. (3), which considers point location $\boldsymbol{p}$ and ray direction $\boldsymbol{d}$ simultaneously by:

$$\boldsymbol{E}_p = \text{MLP}(\text{Concat}([\text{PosEmb}(\boldsymbol{p}), \text{PosEmb}(\boldsymbol{d})])),$$

where PosEmb is a Fourier transformation with a frequency of 10 while MLP is used to fuse point location and ray direction information and project positional embedding to the same dimension as point features.

Point-slot Joint Decoding Our point-slot joint decoding contains an allocation transformer and an attention-based point-slot mapping module. The allocation transformer consists of four transformer layers, and each layer includes a cross-attention layer, a 1D convolution layer, and a self-attention layer. We use 1D convolution and self-attention to model the relationship among points along a ray. The design is based on the insight that spatially adjacent points are more likely to be associated with the same slot. Additionally, We use the weighted sum of attention weights to estimate the density value in Eq. (4). As this design may restrict the scale of density by the attention weights between slots and point features, the density obtained from Eq. (4) is multiplied by a learnable parameter s_σ to rescale it.

Table A.1: Training configuration for our SlotLifter. The values in parentheses are adopted for the ScanNet and DTU datasets.

Training	Scene Batch Size	4 (2)
	Ray Batch Size	1024
	LR	5e-5
	LR Warm-up Steps	10000
	LR Decay Steps	50000
	Max Steps	250K
	Num. Source Views	1 (4)
	Grad. Clip	0.5
Scene Encoding	Feature Dimension	64 (32)
	Slot Dimension	256
	Iterations	3
	σ Annealing Steps	30000
Point-slot Decoding	Num. Layers	4
	Attention Heads	4
	Feature Dimension	64

Table A.2: Image resolution and the number of slots on different datasets.

Dataset	CLEVR567	Room-Chair	Room-Diverse	Room-Texture
Resolution	128×128	128×128	128×128	128×128
Number of slots	8	5	5	5
Dataset	Kitchen-Shiny	Kitchen-Matte	DTU MVS	ScanNet
Resolution	128×128	128×128	400×300	640×480
Number of slots	5	5	8	8

Hyperparameters and Training Details We train our SlotLifter by sampling 1024 rays for each scene with a learning rate of 5×10^{-5}, a linear learning rate warm-up of 10000 steps, and an exponentially decaying schedule. All the images are resized to 128×128 for synthetic data and 640×480 for real-world data. Image resolution and the number of slots K used on different datasets are shown in Tab. A.2. To encourage SlotLifter to segment the background properly, we use the locality constraint proposed by uORF [14]. Specifically, we set a background bound and enforce every point outside the bound being mapped to the empty slot or the first slot. The locality constraint is imposed for the first 50K iterations, preventing SlotLifter from segmenting the background as 2 separate objects. Note that our SlotLifter does not require a background-aware Slot-Attention like uORF since our slots are initialized by learnable queries, enabling Slot-Attention to learn to individually segment the background and foreground objects. On ScanNet and DTU MVS, we adopt the coarse-to-fine

Table A.3: Efficiency and performance comparisons on Room-Diverse. We evaluate all the methods on an NVIDIA RTX 3090 GPU.

Model	PSNR↑	LPIPS↓	NV-ARI↑	ARI↑	GPU Memory↓	Training Time↓
uORF	25.96	0.1729	56.9	65.6	24 GB	6 days
BO-uORF	26.96	0.1515	62.5	72.6	24 GB	6 days
ours(N=256)	29.83	0.1345	76.1	88.7	3.5 GB	10 hours
ours(N=512)	**29.84**	0.1277	76.2	88.0	6 GB	19 hours
ours(N=1024)	29.80	**0.1180**	**77.5**	**90.3**	12 GB	30 hours

Table A.4: Ablations on the number of rays with different image sizes. Increasing the number of rays sightly improves rendering and segmentation quality, while reducing image size slightly decreases both rendering and segmentation quality.

Number of rays	ScanNet (640×480)		Room-Diverse (128×128)				Room-Diverse (64×64)			
	PSNR↑	NV-FG-ARI↑	PSNR↑	LPIPS↓	NV-ARI↑	ARI↑	PSNR↑	LPIPS↓	NV-ARI↑	ARI↑
256	27.27	19.8	29.83	0.1345	76.1	88.7	29.55	0.0792	69.2	82.4
512	27.92	**32.3**	**29.84**	0.1277	76.2	88.0	**29.57**	0.0731	69.9	**83.2**
1024	**28.36**	31.1	29.80	**0.1180**	**77.5**	**90.3**	**29.57**	**0.0687**	**70.2**	82.4

sampling scheme on ScanNet following previous methods, sampling 64 points along each ray for the coarse sampling and another 64 points for the fine sampling. We found that the coarse-to-fine sampling scheme aids SlotLifter in rendering novel views with higher quality. The training configuration is summarized in Tab. A.1. Additionally, we found the background occlusion regularization loss from [6, 13] is helpful on the Kitchen-Matte and Kitchen-Shiny datasets for preventing the background slot segmenting foreground objects but it has little effect on the rendering quality. We only use this loss on the Kitchen-Matte and Kitchen-Shiny datasets because we didn't find it helpful on other datasets.

A.2 Baselines

uORF and BO-uORF The experimental results of uORF [14] on CLEVR-567, Room-Chair, and Room-Diverse are taken from their paper. We trained the BO-uORF model on CLEVR-567, Room-Chair, Room-Diverse, and ScanNet using the official implementation of uORF and BO-QSA. As (BO-)uORF only accepts single source view input, we selected the closest view to the target view as the source view for it. Unfortunately, due to design limitations, such as model architecture, adversarial loss, perceptual loss, etc., we could not train the BO-uORF model at the resolution of 640×480. Therefore, we had to use a resolution of 128×128 following their original settings. We use 8 slots for uORF as same as our method.

OSRT We trained OSRT [8] on CLEVR-567, Room-Chair, Room-Diverse, and ScanNet using the implementation recommended by the authors on the project

website of OSRT. We observed that OSRT's performance of scene decomposition is highly sensitive to the batch size used during training, which is also mentioned in the implementation. Due to the computational limitations, we trained the OSRT for 250K iterations using a batch size of 64 and sampling 2048 rays for each scene with 2 Nvidia A100 GPUs. To train OSRT on the ScanNet dataset, we resized all images to 128×128.

GNT We trained GNT [10] on ScanNet using their official implementation. We trained GNT for 250K iterations with their config `gnt_full.txt` in their repository, which uses a learning rate of 5×10^{-4}, samples 2048 rays for each scene, and selects 10 nearby source views to render the target view.

B Additional Discussions

B.1 Potential Improvements

Although SlotLifter exhibits superior performance in novel-view synthesis and scene decomposition compared to state-of-the-art 3D object-centric learning methods, its scene decomposition performance still falls short under real-world settings. This is particularly noteworthy considering the recent success of 2D object-centric models on real-world images (see in Tab. A.5). We attribute this undesired effect to the unconstrained point-slot mapping process. As elaborated in Sec. 3.2, the slots are mapped to the 3D points which are later projected to the target view image. With only reconstruction loss, the information in the target image can be backpropagated to both slots and lifted point features. This adds no direct guidance or constraints on slot learning and can easily make the learned slots attend to features that best render the scene instead of decomposing it.

Table A.5: Quantitative segmentation results on ScanNet. FG-ARI is evaluated on the input view(s). "MV" indicates 3D multi-view inputs.

Model	Modality	FG-ARI↑	NV-FG-ARI↑	PSNR↑
Slot-Attention [5]	2D	31.1	-	-
DINOSAUR [9]	2D	**47.6**	-	-
OSRT [8]	MV	29.8	29.7	13.34
SlotLifter	MV	32.0	31.1	**28.36**
SlotLifter w/ $\mathcal{L}_{\text{feat}}$	MV	36.1	**35.7**	25.38

To account for this issue we considered guiding slots to decompose scenes with semantic priors in pre-trained models. Inspired by recent object-centric learning methods DINOSAUR [9] and VideoSAUR [15] that replace image reconstruction with feature reconstruction, we propose to improve the scene decomposition capability of SlotLifter by adding a feature reconstruction loss. Specifically, we first extract DINOv2 [7] features $\hat{\boldsymbol{H}}$ for the target view as ground truth. Similar to the color prediction in Eq. (5) we add an MLP to predict a feature grid $\boldsymbol{h}$ and render 2D features $\boldsymbol{H}$. Next, we add the feature reconstruction loss over the predicted target-view feature, *i.e.* $\mathcal{L}_{\text{feat}} = 1 - D(\boldsymbol{H}, \hat{\boldsymbol{H}})$, where D denotes the cosine similarity. As shown in Tab. A.5, this feature reconstruction loss improves the segmentation performance on ScanNet, but it harms the rendering quality of novel view images, decreasing the PSNR to 25.38. This result reveals the key conflict between the high-level semantic guidance and the low-level appearance

Table A.6: Quantitative results of OSRT in synthetic scenes. We present the best performance of our reimplemented OSRT. The performance of OSRT is hindered by its requirements for large amounts of data and computational demands.

Model	CLEVR-567			Room-Chair			Room-Diverse		
	NV-ARI↑	FG-ARI↑	PSNR↑	NV-ARI↑	FG-ARI↑	PSNR↑	NV-ARI↑	FG-ARI↑	PSNR↑
OSRT† [8]	3.1	10.3	20.73	5.4	24.0	20.99	7.4	39.3	24.58
uORF [14]	83.8±0.3	87.4±0.8	29.28	74.3±1.9	88.8±2.7	29.60	56.9±0.2	67.8±1.7	25.96
SlotLifter	**87.0±2.8**	**91.3±1.8**	**36.09**	**89.7±0.5**	**91.9±0.3**	**34.63**	**77.5±0.7**	**84.3±2.9**	**29.97**

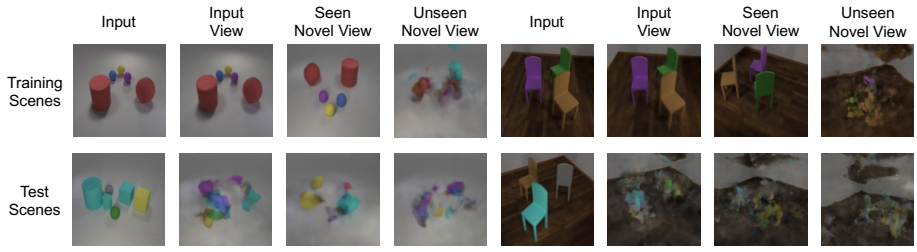

Fig. A.1: Qualitative results of OSRT. OSRT tends to overfit training scenes and training views, making it difficult to generalize to unseen scenes and unseen views.

guidance which is commonly shared in object-centric models. Adding 3D geometry or temporal constraints (*e.g.*, shape and temporal consistency) that reveal objectness can potentially solve this problem and we leave it as an important future work.

On the other hand, the superior performance on ScanNet and DTU implies better scene encoding in SlotLifter, supporting potential conjectures that these latent slots work similarly to latent feature grids with point features interpolated over them for better novel-view synthesis. This echoes the success of feature-grid-based methods (*e.g.*, Plenoxels [1]) for improving the performance of NeRF.

B.2 Further Discussions about Previous Methods

(BO-)uORF As shown in Tab. 5 and Fig. 4, BO-uORF failed to render novel views and decompose scenes in complex real-world scenes, achieving only a PSNR of 12.72 and a NV-FG-ARI of 0.0. Moreover, to demonstrate that the failure of BO-uORF is not due to lower resolution, we trained our SlotLifter with a resolution of 128×128 and achieved a PSNR of 29.31.

OSRT We present the quantitative results of OSRT in Tab. A.6 and visualize qualitative results in Fig. A.1. The performance of OSRT is hindered by its requirements for large amounts of data and computational demands, especially on CLEVR-567 which only has 1000 training scenes. We observed that the OSRT tended to overfit training scenes (Fig. A.1), making it difficult to generalize

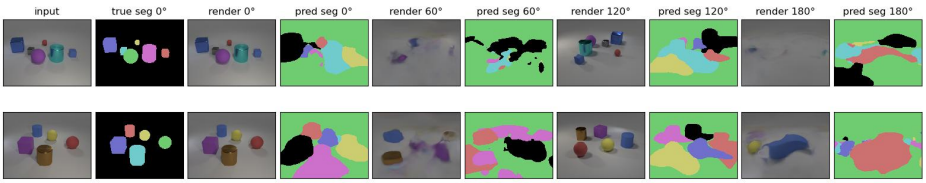

Fig. A.2: Qualitative results of OSRT from the implementation recommended by the authors of OSRT. OSRT performs well on seen views (*e.g.*, 0°, 120°), but has difficulty in unseen views (*e.g.*, 60°, 180°).

to unseen scenes. We attempted a larger batch size of 256 and trained OSRT for more iterations (750K), but the overfitting issue persisted. We also visualize the results provided by this implementation in Fig. A.2, which demonstrates a similar experimental phenomenon that OSRT performs well on seen views (*e.g.*, 0°, 120°), but has difficulty in unseen views (*e.g.*, 60°, 180°). Quantitatively, on the CLEVR-567 dataset, OSRT achieved 47+ PSNR on training scenes, but only 20.73 on test scenes. Similarly, on the ScanNet dataset, OSRT achieved 27+ PSNR on training scenes, but only 13.34 on test scenes. These results demonstrate that OSRT may memorize all the training scenes with its powerful transformer encoder-decoder, requiring a lot of data to overcome the overfitting problem. The number of training scenes used in our paper might not be sufficient to train the OSRT(1000 for CLEVR-567 and Room-Chair, 5000 for Room-Diverse, and 100 for ScanNet), leading to the failure case. We attribute this ineffectiveness to its lack of inductive bias for 3D scenes, which is a main distinction between OSRT and our SlotLifter.

C Limitations and Future Work

Inference efficiency While our SlotLifter has significantly improved training efficiency compared to other 3D object-centric models, its inference efficiency is not satisfactory compared with light field methods (*e.g.*, COLF and OSRT). The primary reason for this issue is that NeRF representations require the sampling of a large number of points with expensive computations, most of which are wasted on irrelevant vacant points. Although light field methods, such as OSRT, are very efficient for inference, they lack the use of 3D information and require a lot of data and computation commands to overcome the overfitting problem. Some recent works, such as those based on point clouds [12], surfels [2], and Gaussian Splatting [4, 16], have demonstrated high efficiency for inference and good utilization of 3D information, which could be integrated into future work to improve the inference efficiency.

Details of Complex Object As depicted in Fig. A.5 and Fig. A.6, the SlotLifter encounters challenges in accurately rendering and segmenting chair legs from different angles, particularly when dealing with real chairs in Room-Texture. A primary issue contributing to this difficulty lies in ray sampling.

NeRF-based techniques typically employ ray sampling during the training process to reduce computational load. For example, in our case, we sample 1024 rays from an image with $128 \times 128 = 16384$ pixels. Consequently, the majority of rays focus on the background and larger objects, leaving finer details like chair legs with limited attention. While increasing the number of sampled rays could address this issue, it would also escalate the computational demands. The integration of Gaussian Splatting [4] has the potential to assist in balancing computational requirements with rendering quality. Moreover, we have observed that this problem exists in other approaches as well. Nevertheless, it appears to be mitigated in uOCF [6] due to its training with the object prior, which could potentially aid our SlotLifter in addressing this particular challenge.

D Additional Visualizations

We provide more qualitative results of our SlotLifter in the following pages.

Fig. A.3: Novel view synthesis and unsupervised segmentation on CLEVR-567.

Fig. A.4: Novel view synthesis and unsupervised segmentation on Room-Chair.

Fig. A.5: Novel view synthesis and unsupervised segmentation on Room-Diverse.

Fig. A.6: Novel view synthesis and unsupervised segmentation on Room-Texture.

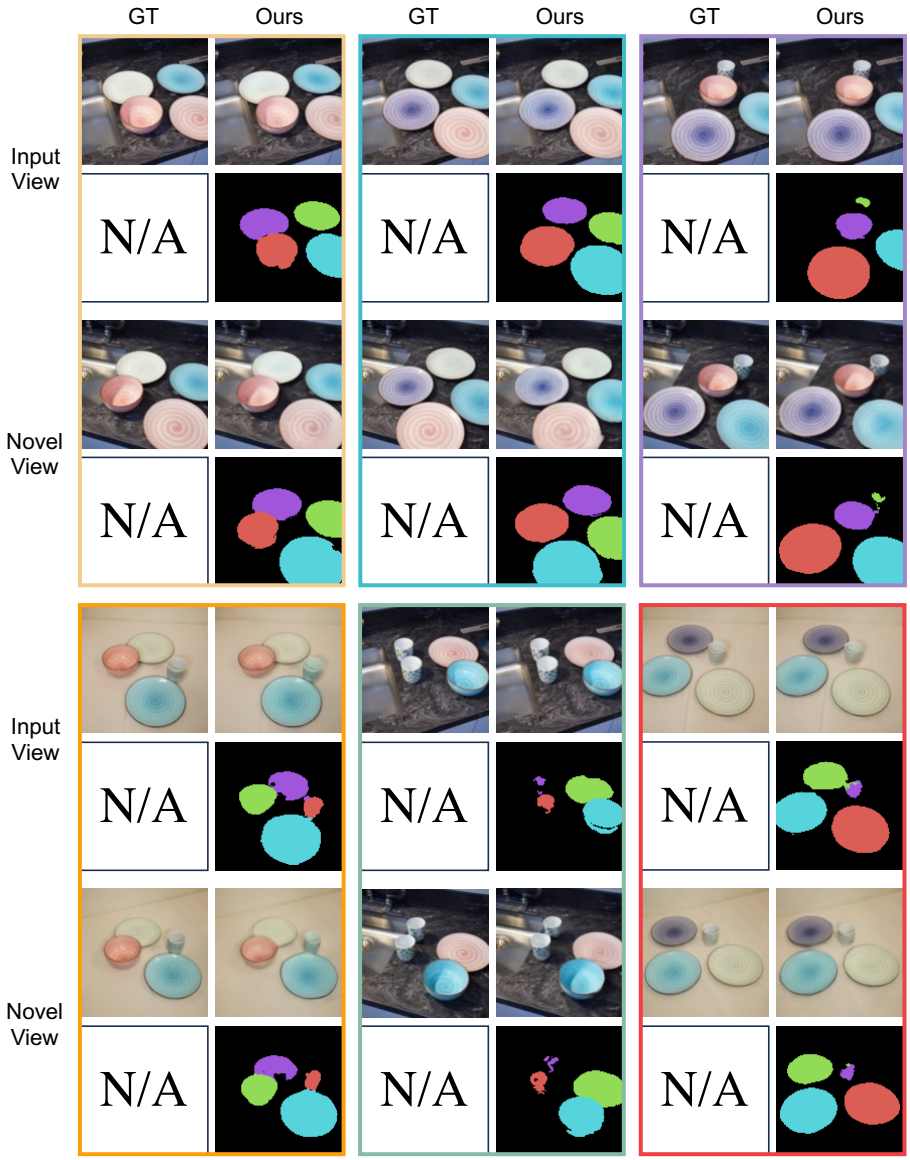

Fig. A.7: Novel view synthesis and unsupervised segmentation on Kitchen-Shiny.

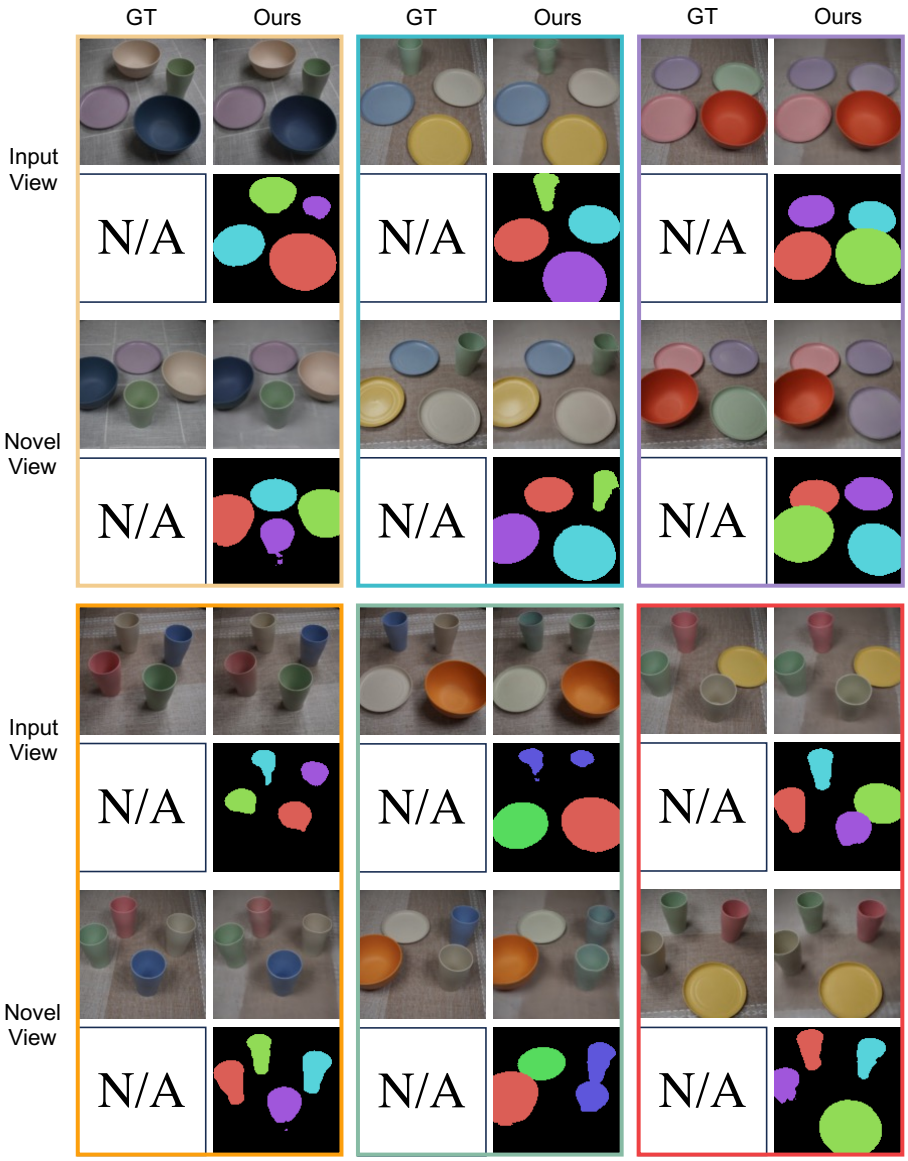

Fig. A.8: Novel view synthesis and unsupervised segmentation on Kitchen-Matte.

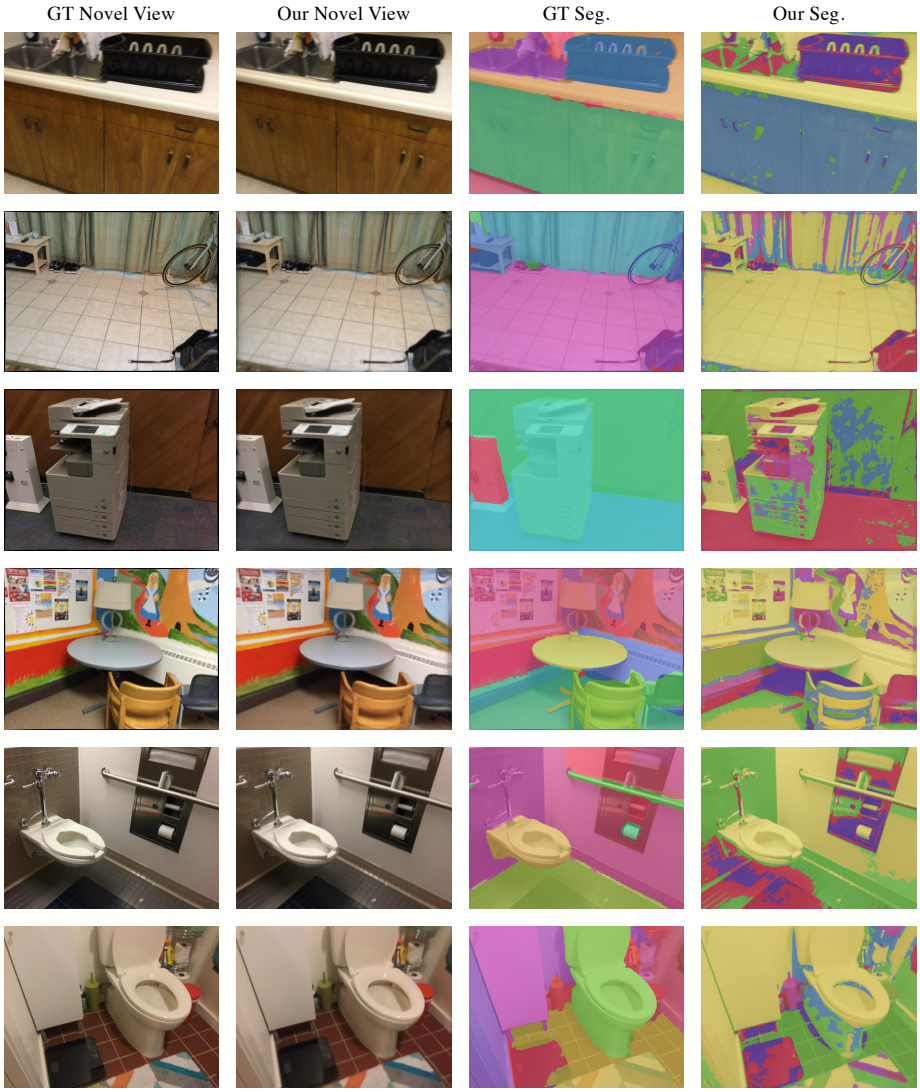

Fig. A.9: Novel view synthesis and unsupervised segmentation on ScanNet.

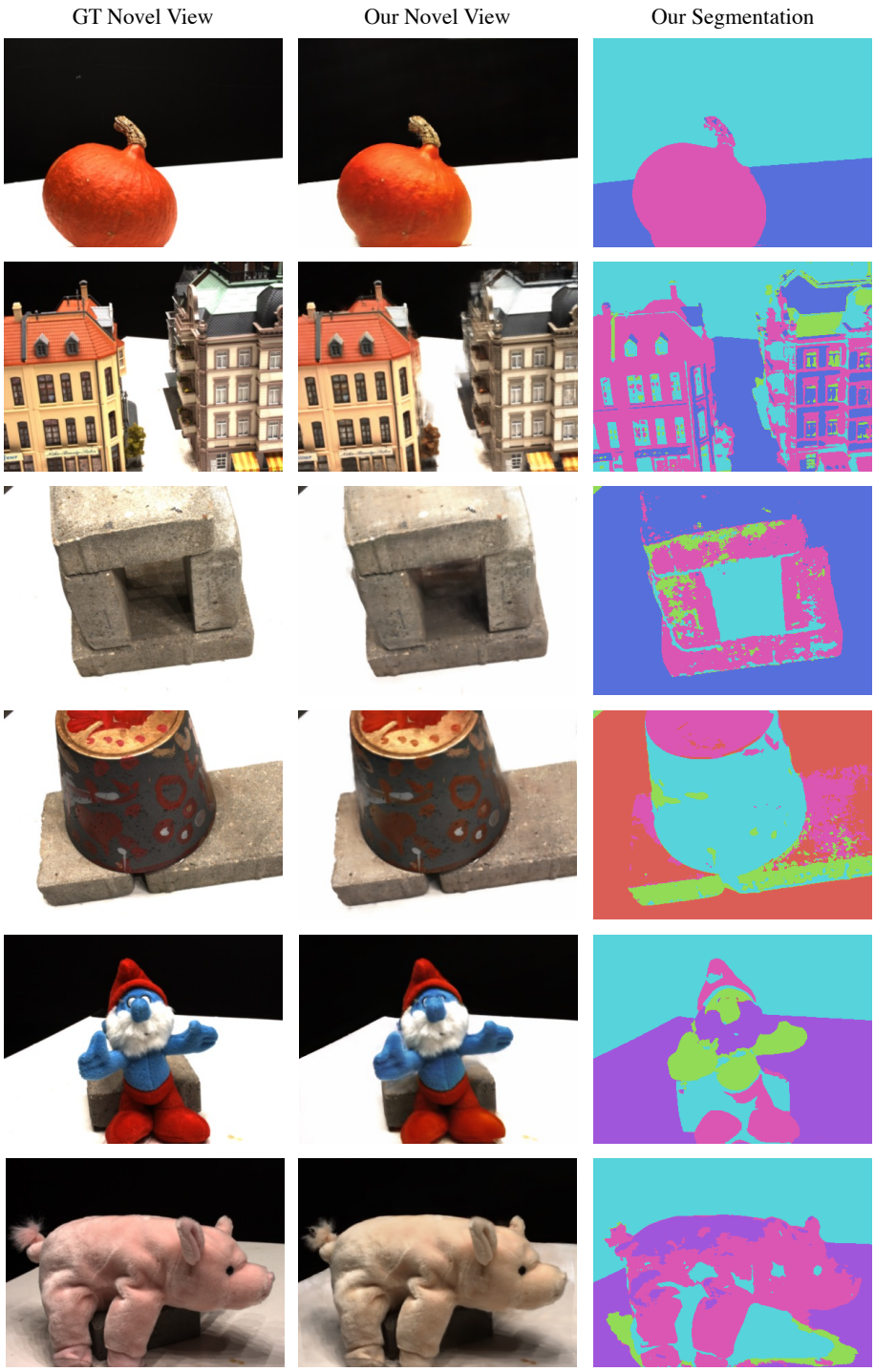

Fig. A.10: Novel view synthesis and unsupervised segmentation on DTU MVS.

References

1. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR) (2022)
2. Gao, Y., Cao, Y.P., Shan, Y.: Surfelnerf: Neural surfel radiance fields for online photorealistic reconstruction of indoor scenes. In: Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR) (2023)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
4. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics (TOG) **42**(4) (2023)
5. Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., Kipf, T.: Object-centric learning with slot attention. In: Proceedings of Advances in Neural Information Processing Systems (NeurIPS) (2020)
6. Luo, R., Yu, H.X., Wu, J.: Unsupervised discovery of object-centric neural fields. arXiv preprint arXiv:2402.07376 (2024)
7. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193 (2023)
8. Sajjadi, M.S., Duckworth, D., Mahendran, A., van Steenkiste, S., Pavetić, F., Lučić, M., Guibas, L.J., Greff, K., Kipf, T.: Object scene representation transformer. In: Proceedings of Advances in Neural Information Processing Systems (NeurIPS) (2022)
9. Seitzer, M., Horn, M., Zadaianchuk, A., Zietlow, D., Xiao, T., Simon-Gabriel, C., He, T., Zhang, Z., Schölkopf, B., Brox, T., et al.: Bridging the gap to real-world object-centric learning. In: Proceedings of International Conference on Learning Representations (ICLR) (2023)
10. Varma, M., Wang, P., Chen, X., Chen, T., Venugopalan, S., Wang, Z.: Is attention all that nerf needs? In: Proceedings of International Conference on Learning Representations (ICLR) (2022)
11. Wang, Q., Wang, Z., Genova, K., Srinivasan, P.P., Zhou, H., Barron, J.T., Martin-Brualla, R., Snavely, N., Funkhouser, T.: Ibrnet: Learning multi-view image-based rendering. In: Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
12. Xu, Q., Xu, Z., Philip, J., Bi, S., Shu, Z., Sunkavalli, K., Neumann, U.: Point-nerf: Point-based neural radiance fields. In: Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR) (2022)
13. Yang, J., Pavone, M., Wang, Y.: Freenerf: Improving few-shot neural rendering with free frequency regularization. In: Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR) (2023)
14. Yu, H.X., Guibas, L.J., Wu, J.: Unsupervised discovery of object radiance fields. In: Proceedings of International Conference on Learning Representations (ICLR) (2022)
15. Zadaianchuk, A., Seitzer, M., Martius, G.: Object-centric learning for real-world videos by predicting temporal feature similarities. In: Proceedings of Advances in Neural Information Processing Systems (NeurIPS) (2023)

16. Zheng, S., Zhou, B., Shao, R., Liu, B., Zhang, S., Nie, L., Liu, Y.: Gps-gaussian: Generalizable pixel-wise 3d gaussian splatting for real-time human novel view synthesis. arXiv preprint arXiv:2312.02155 (2023)