Learning Object-Compositional Neural Radiance Field for Editable Scene Rendering Supplementary Material

Bangbang Yang[1] Yinda Zhang[2] Yinghao Xu[3] Yijin Li[1]
Han Zhou[1] Hujun Bao[1] Guofeng Zhang[1] Zhaopeng Cui[1*]

[1]State Key Lab of CAD&CG, Zhejiang University [2]Google [3]The Chinese University of Hong Kong

In this supplementary material, we will describe the model architecture in Sec. A, the implementation details in Sec. B, and dataset information in Sec. C. Besides, we also provide more experiment results in Sec. D.

A. Model Architecture

The model architecture is shown in Table A. We follow Mildenhall *et al*. [5] and use separated "coarse" and "fine" models with 64 sampling locations. For the learnable voxel embeddings, we use 16-dimension for $\boldsymbol{f}_{scn}$ and 8-dimension for $\boldsymbol{f}_{obj}$, and fix the voxel size at 0.1m to maintain constant memory consumption. Besides, the design of the object branch is shallower than the scene branch since we find it sufficient to learn an accurate and intact object opacity field for editable scene rendering, as shown in Fig. I and Fig. J.

Input	Description	Output	Output Dimension
	Inputs		
/	Space Coord. Emb. $\gamma(\boldsymbol{x})$	#1	63 (3+3×10×2)
/	Ray Dir. Emb. $\gamma(\boldsymbol{d})$	#2	27 (3+3×4×2)
/	Scene Voxel Feature Emb. $\gamma(\boldsymbol{f}_{scn})$	#3	208 (16+16×6×2)
/	Object Voxel Feature Emb. $\gamma(\boldsymbol{f}_{obj})$	#4	104 (8+8×6×2)
/	Object Activation Code $\boldsymbol{l}_{obj}$	#5	128
	Scene Branch		
#1 $\oplus$ #3	MLP(256, 256, 256, 256) + LReLU	#6	256
#1 $\oplus$ #3 $\oplus$ #6	MLP(256, 256, 256, 256) + LReLU	#7	256
#7	MLP(1)	σ_{scn}	1
#7	MLP(256)	#8	256
#8 $\oplus$ #2	MLP(128) + LReLU	#9	128
#9	MLP(128, 3) + Sigmoid	$\mathbf{c}_{scn}$	3
	Object Branch		
#1 $\oplus$ #3 $\oplus$ #4 $\oplus$ #5	MLP(128, 128) + LReLU	#10	128
#1 $\oplus$ #3 $\oplus$ #4 $\oplus$ #5 $\oplus$ #6	MLP(128, 128) + LReLU	#11	128
#11	MLP(1)	σ_{obj}	1
#11	MLP(128)	#12	128
#12 $\oplus$ #2	MLP(64) + LReLU	#13	64
#13	MLP(64, 3) + Sigmoid	$\mathbf{c}_{obj}$	3

Table A. **Model architecture.** MLP(C_1, C_2) denotes the multilayer perceptron with the FC output size of C_1, C_2. $\oplus$ is the operation of concatenation.

B. Implementation Details

B.1. Training Details

We train our model with a batch size of 2048 rays on a single Nvidia RTX2080Ti-11G GPU, where the rays are sampled from all the training images. Practically, we follow NeRF's sampling strategy and jointly train two branches with the same sampled rays while assigning an object identity to each ray (Sec 3.3), which is sufficient for the object branch to converge. In this way, the training time of the object branch is not proportional to the number of object classes. We adopt the Adam optimizer with an initial learning rate of 0.001 and a polynomial decay schedule with the power of 2. For the object supervision loss (Eq.2 in the paper), we set $\lambda_1 = 1.0$ and $\lambda_2 = 10.0$, and set 0.05 for "0" signals and 1.0 for "1" signals in the balanced weight $w(\boldsymbol{r})^k$, so as to ensure a smoother convergence for object opacity learning. The training process takes about one day for each scene.

*Corresponding author

B.2. Implementation of Sparse Voxel (NSVF)

The official codebase of NSVF [4] does not include the preprocessing code or instructions to run on the ScanNet dataset, and we also fail to run it on our selected scenes due to the GPU OOM error. Thus, we use our implementation. The main difference is that we use 16-dimension for voxel embeddings to make a fair comparison, which ensures to train on the selected scenes without OOM issue. We name our implementation for NSVF as Sparse Voxel in the experiments.

C. Datasets

C.1. ToyDesk Dataset

The ToyDesk dataset contains two image sets with 96 and 151 posed images and the corresponding 2D instance segmentation. Several toys are randomly placed on a desk, and we capture images around the desk. We use a DSLR (FoV 68.7°) and a mobile phone (FoV 67.9°) to capture

Config.	PSNR	SSIM	LPIPS
w/o SG, 3DGM	22.842	0.674	0.081
w/o 3DGM	22.853	0.673	0.077
Our Scene Renderer	**23.541**	**0.705**	**0.070**

Table B. Ablation for the effectiveness of our proposed scene guidance and 3D guard mask on learning object radiance field.

two image sets with different toy layouts and camera trajectories. Then, we utilize the SfM [6] and 3D reconstruction techniques [8, 3] to recover meshes with camera poses. Finally, we annotate objects on the meshes and project the mesh labels to generate 2D segmentation masks. Note that the quality of the 2D segmentation masks highly depends on the reconstructed mesh, and usually not accurate near object boundaries. In our evaluation, we randomly sample 80% frames for training and others for testing.

C.2. ScanNet Dataset

In our experiments, we choose 'scene0024_00', 'scene0038_00', 'scene0113_00' and 'scene0192_00' for scene rendering and editing comparison. For these sequences, we count the percentage of pixel coverage of objects (∼30%) and distance between the camera center to target objects (∼2m) for ScanNet images, which indicates that some images are captured far from objects and thus have good coverage of background. For ablation studies, we choose 'scene0033_00', 'scene0038_00', 'scene0113_00' and 'scene0192_00', as these four scenes contain occlusion observations for target objects. Besides, we also find that some of the images have a black border caused by image undistortion. Thus, we mask out the border of the images at the training process for NeRF [5], NSVF [4] and our method.

D. More Experimental Results

D.1. Evaluation of Scene Guidance and 3D Guard Mask

We also evaluate the effectiveness of the proposed scene guidance and 3D guard mask. We choose 'scene0113_00' and 'scene0192_00' and follow the same evaluation setup from the paper (4.6). The results are shown in Table B and Fig. B. It is clear that our proposed strategies not only bring a smoother rendering contour (the first row in Fig. B) but also ensure a more complete rendered object (the second row in Fig. B).

D.2. Robustness against Input Segmentation

To inspect the robustness of our method on the segmentation, we manually add noise to the input segmentation by randomly dilating or eroding input masks for several pixels, and the results are shown in Fig. F. We also visualize the

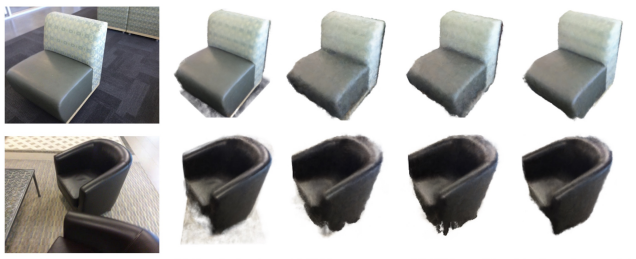


Figure A. We visualize the effectiveness of scene guidance and 3D guard mask by ablating them on the training process of the ScanNet dataset, where (b) is produced by scene branch with sampling rays clipped inside the bounding box, which can be considered as a reference view of the complete object, (c) is trained without guided biased sampling and 3D guard mask, (d) is trained without 3D guard mask, and (e) is trained with the complete model. The proposed strategies effectively prevent the overkill of the occluded region and ensure a complete and smooth object rendering.

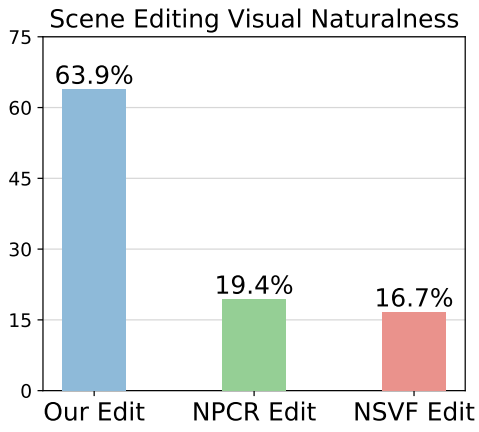
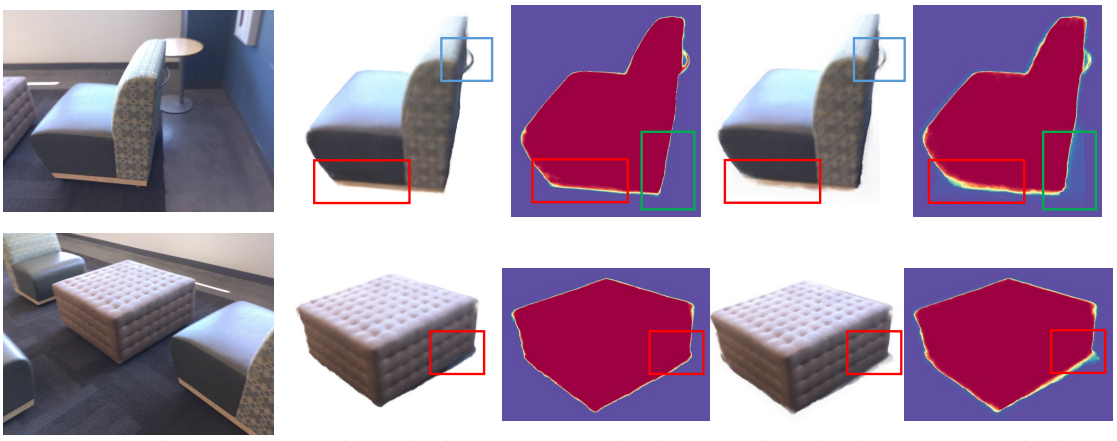


Figure B. User study of visual naturalness on scene editing.

perturbed mask at the rows of input segmentation, where the yellow border denotes the dilated area and the green border denotes the eroded area. To our surprise, even we add 30-pixel noise to the segmentation in 640×480, our model still renders complete objects with only the border being slightly affected.

D.3. Robustness against 2D Predicted Segmentation

To evaluate the robustness of our method on the auto-labeled segmentation, we train our network with segmentation predicted by BlendMask [1] that contains randomly erroneous mask boundaries (*e.g.*, occasionally over-covering or missing part as annotated with red rectangles in (d)). As shown in Fig. D, our method is not affected by inconsistent mask, and the rendered object is complete.



(a) Image view (b) Rendered object by our method (c) Rendered opacity by our method (d) Rendered object by learned mask (e) Rendered opacity by learned mask

Figure C. We show the rendered objects and opacity by our method and an alternative design which learns 3D masks.

(a) Original view (b) 3D based masks (c) Rendered Obj. with (b) (d) 2D based masks (e) Rendered Obj. with (d)

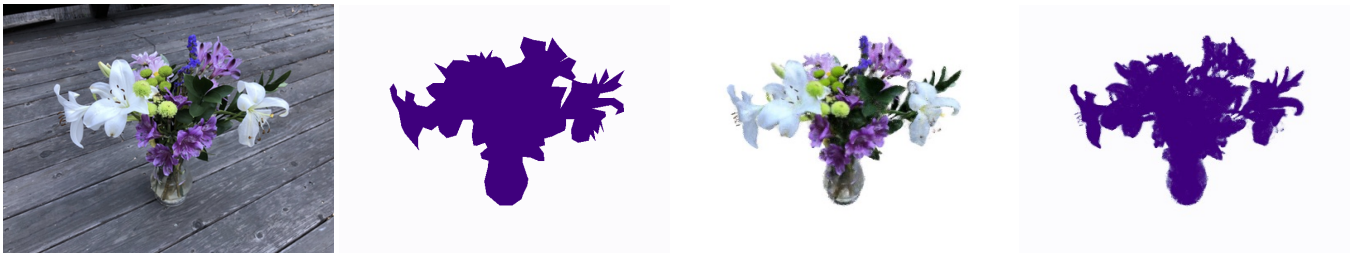
Figure D. We show the rendered object that is trained with different masks. (a) is the input image view. (b) is the 3D based masks provided by the ScanNet dataset. (c) is the rendered object trained with (b). (d) is the 2D based masks predicted by BlendMask [1] and contains randomly erroneous mask boundaries (*e.g.*, over-covering or missing part as annotated with red rectangles). (e) is the rendered object trained with (d).

D.4. Comparison to the Alternative Design

There is an alternative design that learns 3D object masks and directly applies them to the scene-level radiance field. To compare with it, we construct a learnable object opacity mask with the object branch by adding a Sigmoid function to the output of σ_{obj} in Table A. Then we apply this mask to the outputs of the scene branch to obtain the object radiance field. We present the rendered objects and 2D opacity in Fig. C, which demonstrate that this alternative design struggles to learn details of the object (*e.g.*, blurry handle of the chair at the first row), and shows less accuracy of the object boundary (*e.g.*, the bottom of the chair is much fuzziness), and even the background opacity is slightly bleeding out (green rectangle at the first row of Fig. C (e)). Besides, to render an object, we need to perform two forward passes on the scene branch and the object 'mask' branch, respectively, which increases the computational cost compared to our method. In conclusion, we think the object decomposition of neural implicit representation should be jointly learned in our way instead of learning separately.

D.5. More Results on Scene Rendering

We show more qualitative results of scene rendering on ToyDesk dataset and ScanNet dataset in Fig. G and Fig. H. Compared to other SoTA methods, our rendered images show finer details on the objects (*e.g.*, dragon teeth at the last row of Fig. G, chair leg at the second row of Fig. H) and the surrounding environment (*e.g.*, background at the second row of Fig. G).



(a) Original View (b) Supervisory Seg. (c) Rendered Object (d) Rendered Opacity

Figure E. We show the training input ((a) original view and (b) supervisory segmentation) and rendered results ((c) rendered object and (d) rendered opacity) on a complex shape. The rendered results are animatable if opened by **Adobe Reader**.

D.6. More Results on Scene Editing

We present more visual results of scene editing on the ToyDesk dataset and the ScanNet dataset in Fig. I and Fig. J. Specifically, we also show the editing examples of Sparse Voxel (Fig. J (c)) and our method without depth supervision (Fig. J (d)). For the Sparse Voxel, we transform the sampling points according to the bounding boxes of target objects, so that the input voxel embeddings are interpolated from the transformed coordinates. However, it still suffers from the similar issue of bounding box based object manipulation (*e.g.*, moved window texture at the first row and moved carpet texture at the third row). In contrast, our method consistently produces more realistic scene editing results even without the supervision of depth loss. We also perform a user study of the view naturalness on 50 edited shuffled image results with 20 users, and our method is ranked the first as shown in Fig. B. Please refer to the supplementary video for vivid animations of our scene editing.

D.7. Experiment on Complex Shapes

To test the network ability on complex shapes, we train the model on the VaseDeck sequence from Mildenhall *et al*. [5] with coarse segmentation annotated by LabelMe [7]. As shown in Fig. E (animation if opened by **Adobe Reader**), except some missing stamens, the rendered petals and leaves are clear and complete.

References

[1] Hao Chen, Kunyang Sun, Zhi Tian, Chunhua Shen, Yongming Huang, and Youliang Yan. BlendMask: Top-Down Meets Bottom-Up for Instance Segmentation. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8570–8578, 2020. 2, 3

[2] Peng Dai, Yinda Zhang, Zhuwen Li, Shuaicheng Liu, and Bing Zeng. Neural Point Cloud Rendering via Multi-Plane Projection. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7827–7836, 2020. 6

[3] Michael M. Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction. In *Proceedings of Eurographics Symposium on Geometry Processing*, pages 61–70, 2006. 2

[4] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *Proceedings of Advances in Neural Information Processing Systems*, volume 33, 2020. 1, 2, 6

[5] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Proceedings of European Conference on Computer Vision*, pages 405–421, 2020. 1, 2, 4, 6

[6] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113. IEEE Computer Society, 2016. 2

[7] Kentaro Wada. labelme: Image Polygonal Annotation with Python. https://github.com/wkentaro/labelme, 2016. 4

[8] Qingshan Xu and Wenbing Tao. Multi-Scale Geometric Consistency Guided Multi-View Stereo. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 5483–5492, 2019. 2

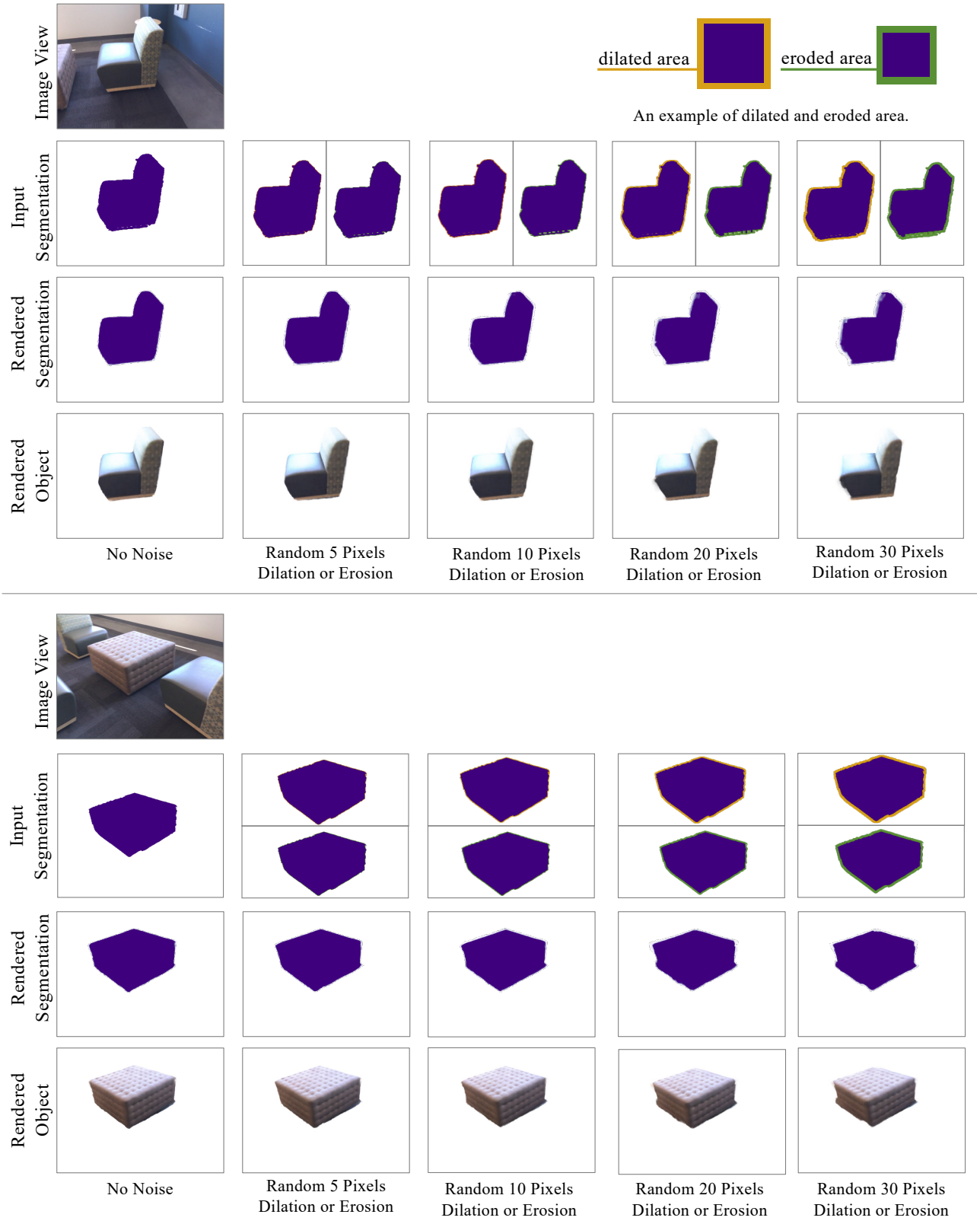
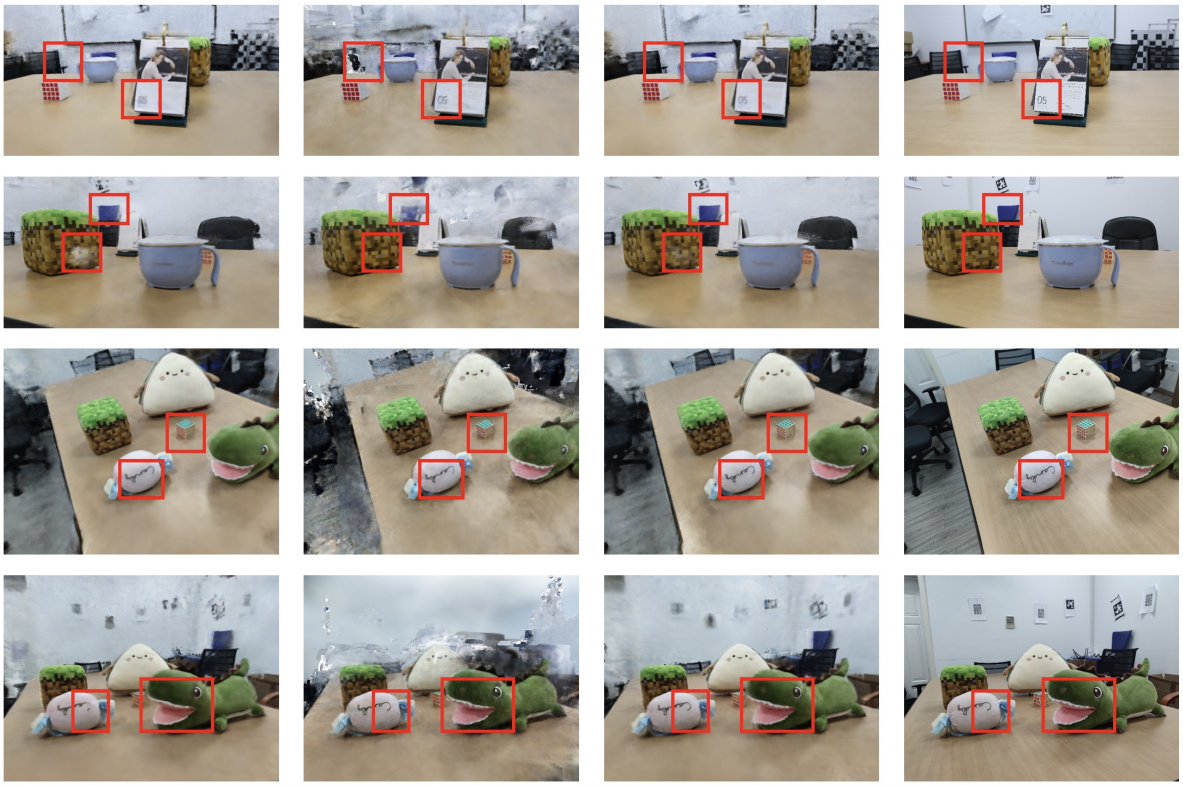
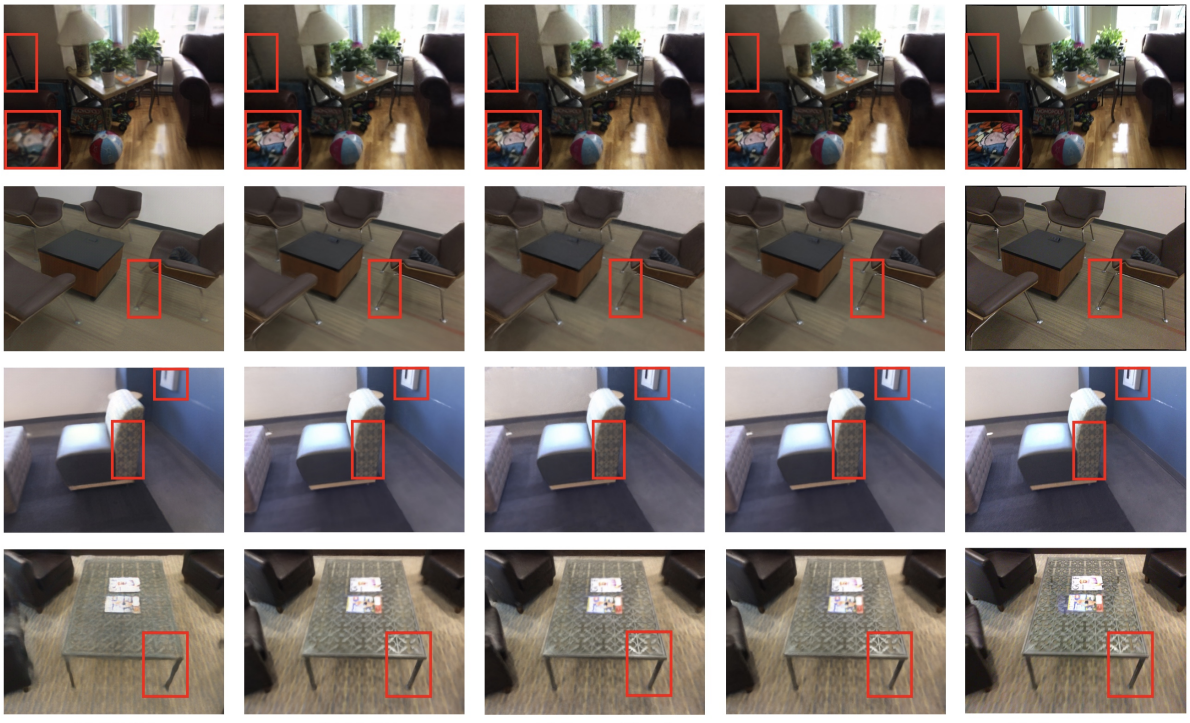


Figure F. Qualitative evaluation w.r.t. noise on the input 2D segmentation mask. We randomly erode or dilate input segmentation masks by some pixels and visualize them in the second row and the sixth row, where the yellow border denotes the dilated area and the green border denotes the eroded area.



(a) NeRF (b) Sparse Voxel (c) Our Method (d) Ground truth

Figure G. We compare scene rendering quality with NeRF [5] and Sparse Voxel [4] on the ToyDesk dataset. Please zoom in for more details.



(a) NPCR (b) NeRF (c) Sparse Voxel (d) Our Method (e) Ground truth

Figure H. We show scene rendering examples of NPCR [2], NeRF [5], Sparse Voxel [4] and our method on the ScanNet dataset. Please zoom in for more details.

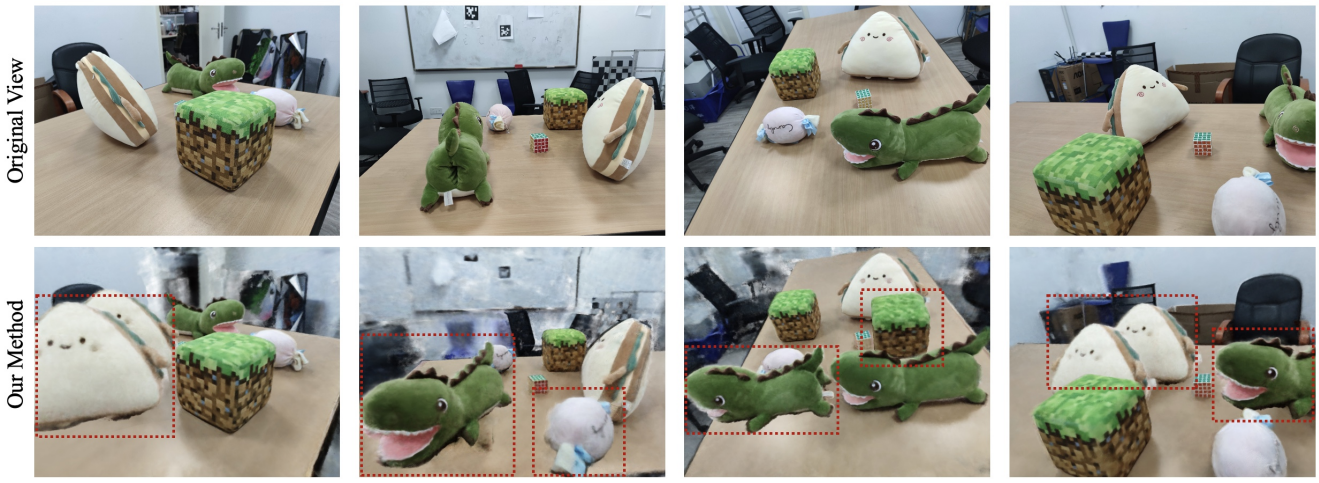


Figure I. Scene editing comparison on the ToyDesk dataset. Please zoom in for more details.

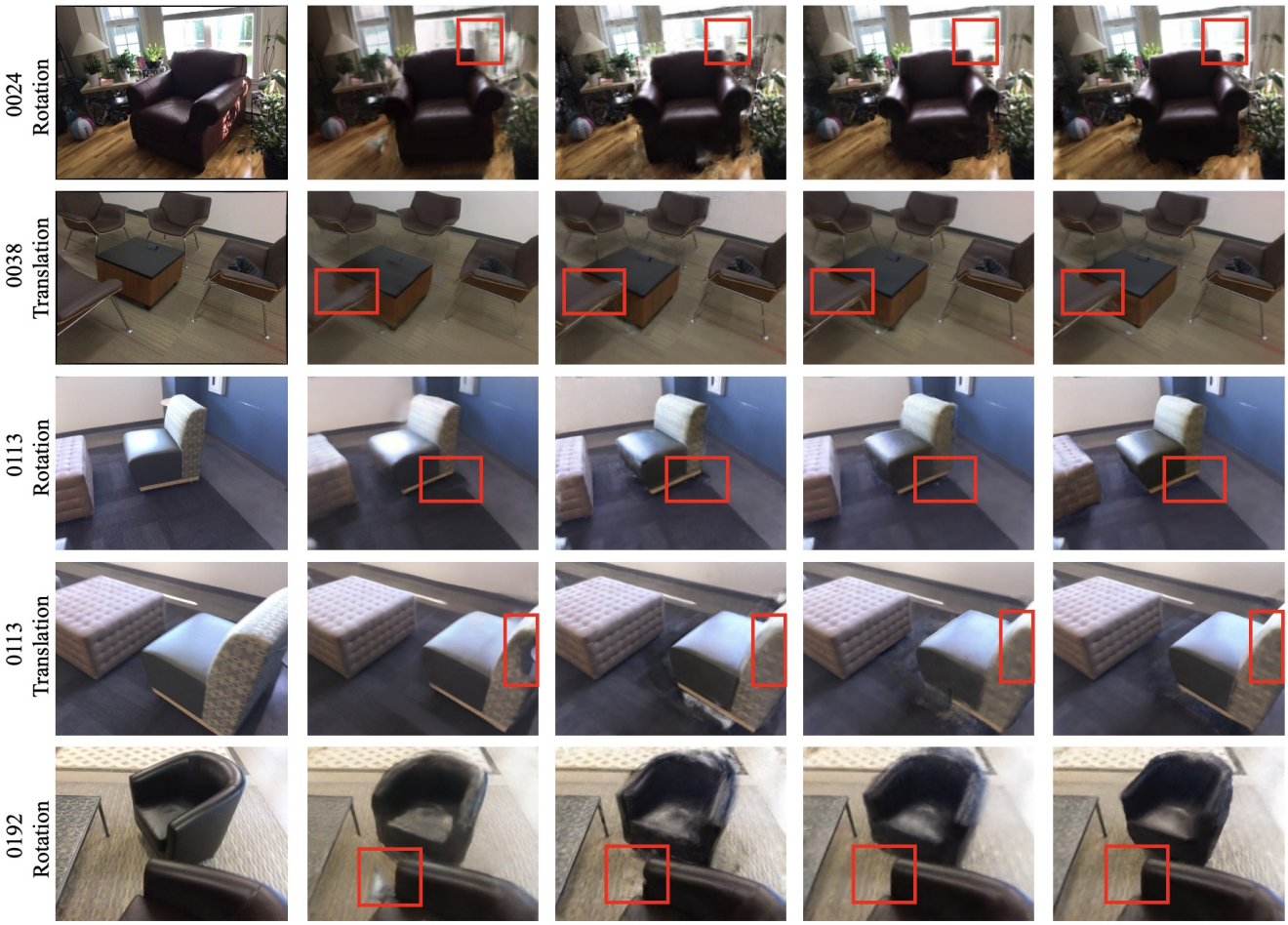


Figure J. Scene editing comparison on the ScanNet dataset. Please zoom in for more details.