A Appendix for Rebuttal

A.1 Qualitative Results of 3D Scene Decomposition and Manipulation in Sections 4.3 & 4.4

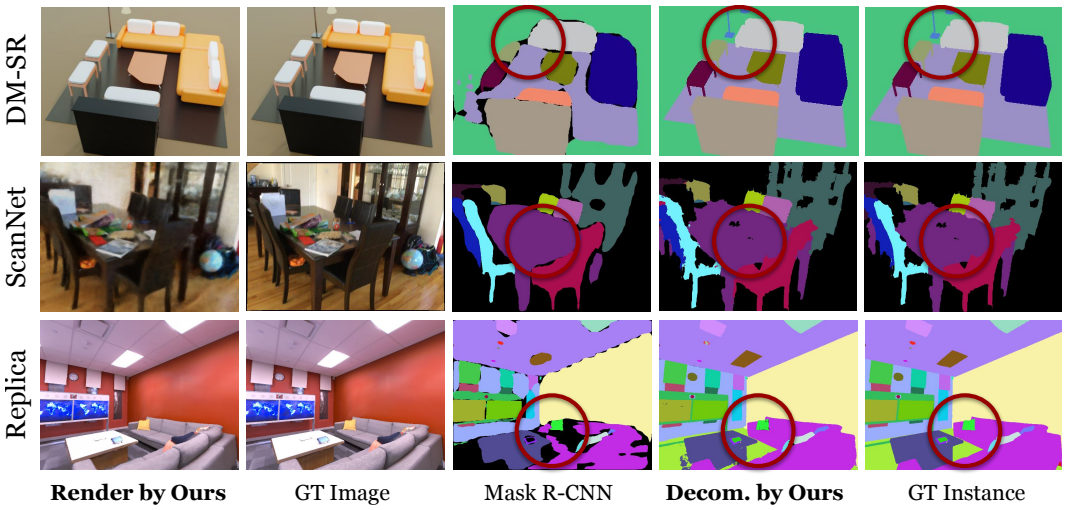


Figure 7: Qualitative results of our method and the baseline on three datasets: DM-SR, Replica and ScanNet. The dark red circles highlight the differences.

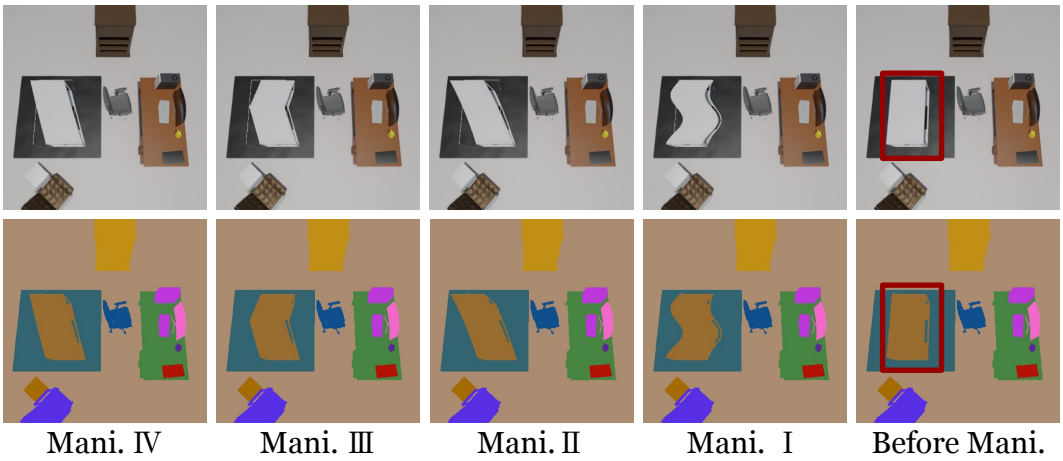


Figure 8: Qualitative results of our method for object deformation manipulation on DM-SR dataset. The dark red boxes highlight the target table to be manipulated.

A.2 Qualitative Results of Our Method Trained on Noisy 2D Labels

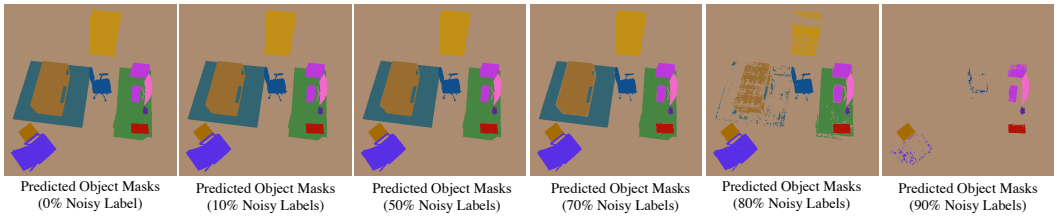


Figure 9: Estimated object masks at novel views from our models trained with different amount of noisy labels. Our method can infer satisfactory results, even though 80% of 2D labels are incorrect during training, demonstrating the robustness of our method.

A.3 Qualitative Results of Our Method Trained on Inaccurate 2D Labels (Estimated by MaskRCNN)

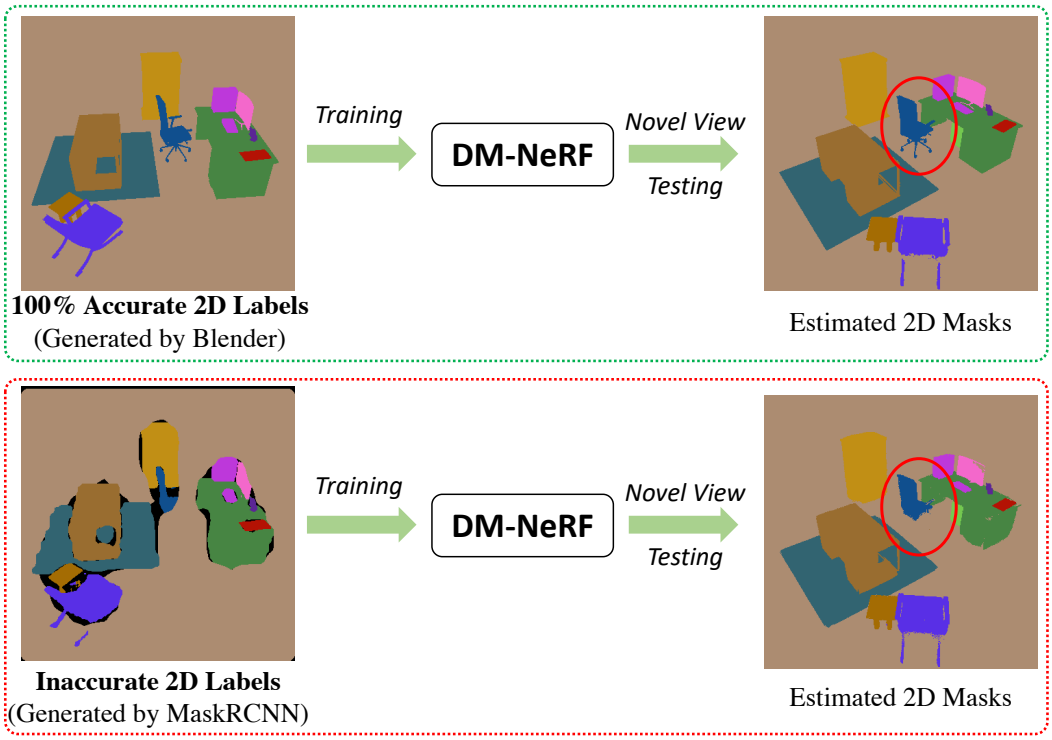
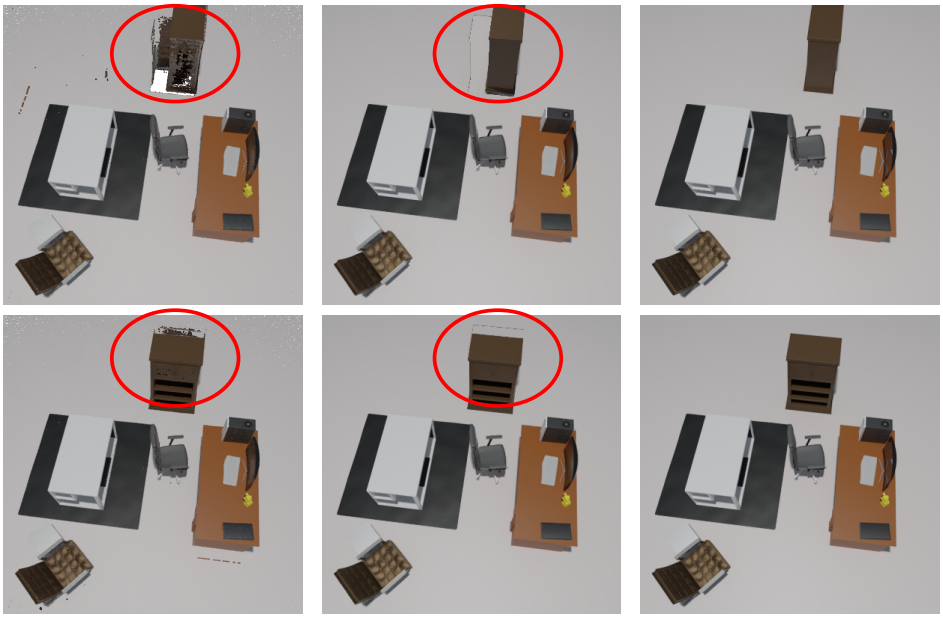


Figure 10: Qualitative results of our method for scene decomposition on DM-SR dataset. As shown in the red dotted block, we use the instance masks estimated by Mask-RCNN as the supervision signals when training our DM-NeRF. We can see that even though the 2D labels are inaccurate for training, our method still achieves excellent object decomposition results. The red circles show that only thin chair feet are not segmented using the inaccurate 2D labels in training. This result is consistent with that of Section A.2, showing the remarkable robustness of our method.

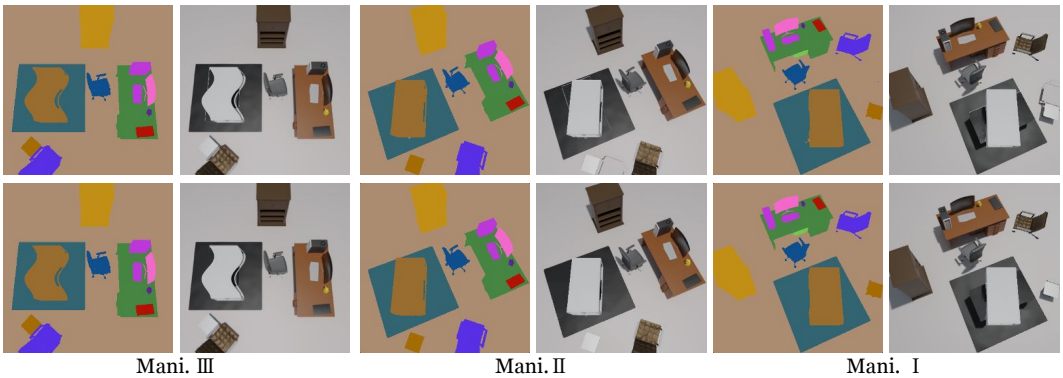
A.4 QUALITATIVE RESULTS OF OUR METHOD AND POINT-NERF FOR OBJECT MANIPULATIONS



Point-NeRF **Ours** Ground Truth

Figure 11: Qualitative results of novel view rendering after manipulating 3D objects. It can be seen that our method obtains clearly sharper object shapes after manipulation in 3D space, whereas the baseline Point-NeRF shows obvious artifacts such as holes, primarily because its manipulation is conducted on explicit 3D point clouds followed by neural rendering, but our manipulation is conducted in continuous neural radiance space and therefore has fine-grained results.

A.5 REPAIRING ARTIFACTS



Mani. Ⅲ Mani. Ⅱ Mani. Ⅰ

Figure 12: New qualitative results of object manipulations. The artifacts can be easily removed. By introducing voting strategy, we further improve the quality of scene decomposition for the following fine manipulation. To be specific, we also consider 8 neighbouring rays when determining the projected object code $\hat{o}$ along a light ray. If $\hat{o}$ is different from the codes of 8 neighbouring rays and these codes is dominated by one of them by voting (the number of dominated code is greater than 4), then $\hat{o}$ will be reset to the dominated code. Otherwise, the original code will be kept. Such an operation can generate more accurate scene decomposition results to support the Inverse Query Algorithm, especially when visual occlusion happens. More results are in the supplied video.

A.6 PANOPTIC SEGMENTATION

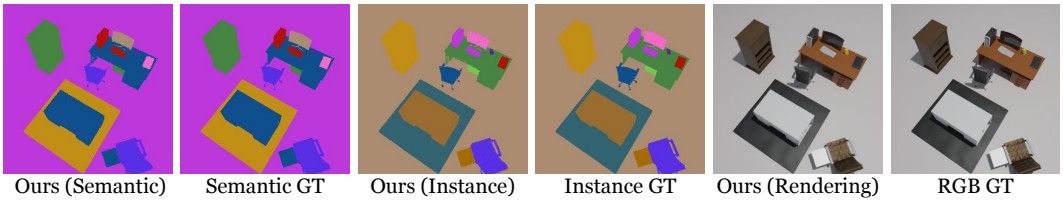


Figure 13: Qualitative results of our method for panoptic segmentation on DM-SR dataset. An extra semantic branch parallel to object code branch is added into our current DM-NeRF. It can be seen that both semantic categories and object codes are accurately inferred.

A.7 COMPUTATION

We typically train 200K iterations in ∼15 hours on each scene (∼0.27s for each iteration) with the batch size of 3072 rays, which uses ∼24GB GPU memory. In contrast, the original NeRF (rendering only) needs ∼0.22s for each iteration). During the inference of joint decomposition and rendering, our DM-NeRF costs ∼9.3s per image with the batch size of 4096 rays using ∼5GB GPU memory. For joint decomposition, manipulation and rendering, ∼23.4s are required for each image and ∼9GB GPU memory is needed when the batch size is set as 4096 rays. To render an image from a novel view, the original NeRF and Point-NeRF need ∼7.8s and ∼8.2s, respectively. All training and testing are operated on a single Nvidia GeForce RTX 3090 card.

A.8 ADAPTATION OF POINT-NERF FOR OBJECT MANIPULATION

During training, to ensure the quality of Point-NeRF on our scene-level DM-SR dataset, we directly use the ground truth dense point cloud (400 million points per scene) instead of the MVSNet generated one when generating the neural point cloud. During the inference of manipulation, we firstly feed the spatial locations of all neural points into our DM-NeRF to infer the corresponding object codes. After determining the points to be edited, we follow the pre-defined manipulation information to transform the corresponding neural points to desired locations. Then, the new neural point cloud is used to render an image with object manipulation from a given view, by point-based volume rendering in Point-NeRF.

A Additional Implementation Details

Network Architecture The detailed architecture of our simple pipeline is shown in Figure 8.

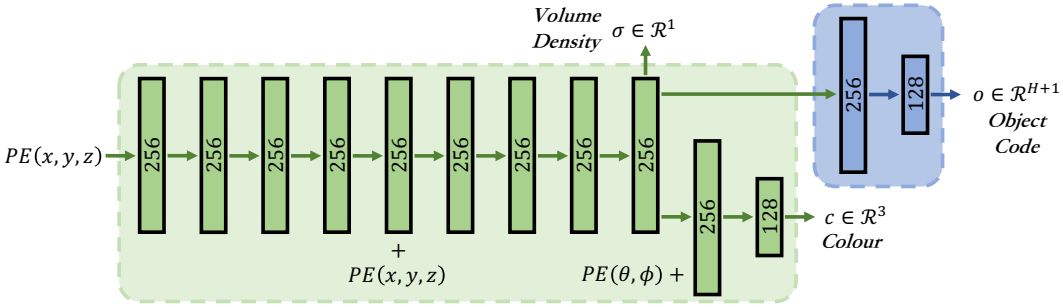


Figure 8: DM-NeRF architecture. The positional encoding $PE(\cdot)$ of the location (x, y, z) and viewing direction (θ, ϕ) are taken as the inputs of our network. The volume density σ and object code $\boldsymbol{o}$ are the functions of the location while the colour additionally depends on the viewing direction.

2D Object Matching and Supervision As illustrated in Figure 3, assuming we generate L images of 2D object predictions $\{I_1 \dots I_l \dots I_L\}, I_l \in \mathbf{R}^{U \times V \times (H+1)}$ and have the paired L images of 2D ground truth object labels $\{\bar{I}_1 \dots \bar{I}_l \dots \bar{I}_L\}, \bar{I}_l \in \mathbf{R}^{U \times V \times T}$, in which H is the predefined number of objects and T represents the number of ground truth objects.

For each pair, we firstly take the first H solid object predictions of I and reshape it to $M \in \mathbf{R}^{N \times H}$, where $N = U \times V$. Likewise, $\bar{I}$ is reshaped to $\bar{M} \in \mathbf{R}^{N \times T}$. Then, M and $\bar{M}$ are fed into Hungarian algorithm (Kuhn, 1955) to associate every ground truth 2D object mask with a unique predicted 2D object mask according to Soft Intersection-over-Union (sIoU) and Cross-Entropy Score (CES) (Yang et al., 2019b). Formally, the Soft Intersection-over-Union (sIoU) cost between the h^{th} predicted box and the t^{th} ground truth box in the l^{th} pair is defined as follows:

$$\boldsymbol{C}_{h,t}^{sIoU} = \frac{\sum_{n=1}^{N}(M_h^n * \bar{M}_t^n)}{\sum_{n=1}^{N} M_h^n + \sum_{n=1}^{N} \bar{M}_t^n - \sum_{n=1}^{N}(M_h^n * \bar{M}_t^n)} \tag{9}$$

where M_h^n and $\bar{M}_t^n$ are the n^{th} values of M_i and $\bar{M}_t$. In addition, we also consider the cross-entropy score between M_h and $\bar{M}_t$ which is formally defined as:

$$\boldsymbol{C}_{h,t}^{CES} = -\frac{1}{N}\sum_{n=1}^{N}\left[\bar{M}_t^n \log M_h^n + (1 - \bar{M}_t^n)\log(1 - M_h^n)\right] \tag{10}$$

After association, we reorder the predicted object masks to align with the T ground truth masks, and then we directly minimize the cost values of all ground truth objects in every pair of images.

$$sIoU_l = \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{C}_{t,t}^{sIoU}) \quad CES_l = \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{C}_{t,t}^{CES}) \tag{11}$$

Training Details For all experiments, we set the batch size as 3072 rays just to fully use the memory. For each ray, we sample 64 points and 128 additional points in the coarse and fine volume, respectively. The Adam optimizer with default hyper-parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-7}$) is exploited. The learning rate is set to 5×10^{-4} and decays exponentially to 5×10^{-5} over the course of optimization. The optimization for a single scene typically take around 200–300k iterations to converge on a single NVIDIA RTX3090 GPU (about 17–25 hours).

Evaluation Details We use the MASK-RCNN code open-sourced by the Matterport at `https://github.com/matterport/Mask_RCNN` and follow their procedure to calculate the AP values. Note that, we regard IoU values as scores during the ranking procedure.

B ADDITIONAL DATASET DETAILS

To quantitatively evaluate geometry manipulation, we create a **s**ynthetic dataset with 8 types of different and complex indoor **r**ooms (shown in Figure 10), called DM-SR, containing path-traced images that exhibit complicated geometry. The room types and designs follow Hypersim dataset (Roberts & Paczan, 2021) and the rendering trajectories follow NeRF synthetic dataset (Mildenhall et al., 2020). Each scene has a physical size of $\sim 12 \times 12 \times 3$ meters. Overall, we firstly create and render 8 static scenes, and then manipulate each scene followed by second round rendering.

In our new dataset, 13 common classes of objects (chair, desk, television, fridge, bathtub, etc.) are introduced. The raw object meshes are downloaded from `https://free3d.com/`. We apply different scale/pose transformations on objects and then compose eight common indoor rooms, including bathroom, dinning room, restroom, etc.. We follow the default world coordinate system in Blender: the positive x, y and z axes pointing right, forward and up, respectively.

To increase realism, we set various types of textures and environment lights for different objects and scenes. Four commonly used types of lights (point, sun, spot, area) are included and the strength is limited within 1000W. During rendering, a camera with 50 degrees field of view is added to generate RGB, depth, semantic and instance images at the resolution of 400×400 pixels. For each scene, the training RGB images are rendered from viewpoints randomly sampled on the upper hemisphere. The viewpoints of testing images are sampled following a smooth spiral trajectory. In addition, instance images are generated by the ray cast function built in Blender.

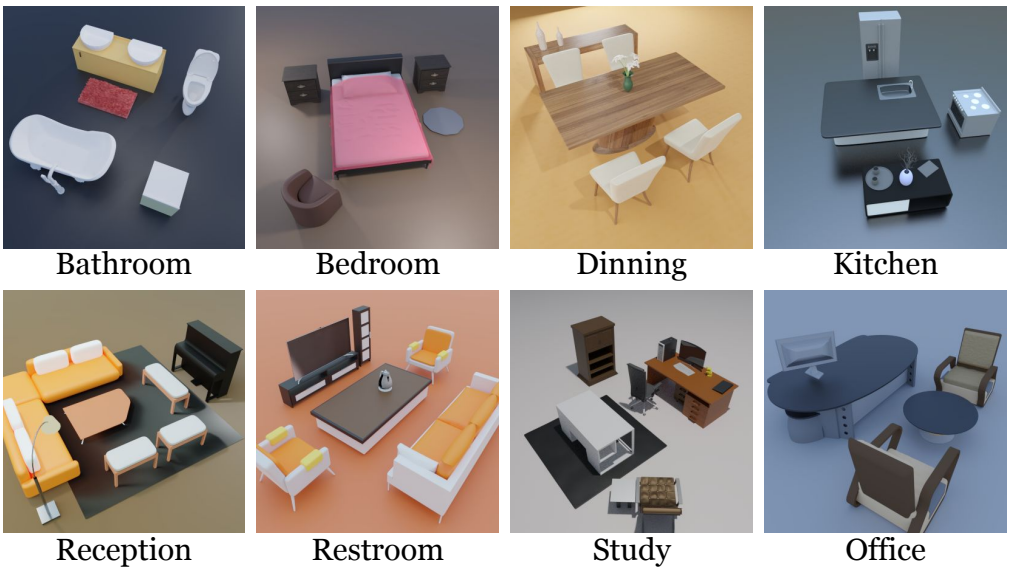


Figure 9: Eight different indoor scenes of our DM-SR dataset.

C ADDITIONAL QUANTITATIVE RESULTS

Ablations of MaskRCNN We provide additional quantitative results of AP scores with IoU thresholds at 0.5 and 0.75 in Table 5 to compare four groups of experiments for MaskRCNN (He et al., 2017) on the selected eight scenes of ScanNet. The settings are as follows:

- **G_1:** Train a single model on the 8 scenes together from scratch, and then evaluate.
- **G_2:** Load a single pretrained model, and then finetune and evaluate it on 8 scenes together.
- **G_3:** Train 8 models on the 8 scenes separately from scratch, and then evaluate respectively.
- **G_4:** Load 8 copies of the pretrained model, and then finetune and evaluate on 8 scenes separately.

Table 5 shows that Group 4 (G_4) achieves the highest scores, which also has the fairest experimental settings we can set up for comparison.

	$AP^{0.50} \uparrow$				$AP^{0.75} \uparrow$			
	G_1	G_2	G_3	G_4	G_1	G_2	G_3	G_4
0010_00	88.34	91.77	90.58	92.80	76.30	74.51	82.61	83.90
0012_00	89.89	92.68	93.63	93.49	85.76	82.77	86.35	86.90
0024_00	83.45	87.15	84.26	87.18	57.46	49.38	67.08	69.87
0033_00	92.81	93.94	93.69	93.74	88.42	87.59	88.93	88.70
0038_00	96.98	96.99	96.94	97.01	95.92	94.88	95.87	96.01
0088_00	85.01	88.07	85.95	90.04	63.29	94.88	73.34	69.06
0113_00	97.97	98.12	97.60	98.59	97.60	98.59	98.17	98.59
0192_00	97.78	97.79	97.78	97.94	96.76	95.79	95.26	96.95
Average	91.53	<u>93.31</u>	92.55	**93.85**	82.69	84.80	<u>85.95</u>	**86.25**

Table 5: Average scores of MaskRCNN on 8 scenes of ScanNet.

Ablations of the Object Field Since the backbone of our pipeline is completely the same as the original NeRF, and our proposed object field component is supervised by ℓ_{2d_obj} from 2D signals and by $(\ell_{3d\_empty} + \ell_{3d\_obj})$ from 3D signals. Note that, the losses $(\ell_{3d\_empty} + \ell_{3d\_obj})$ only involve a single hyper-parameter Δd as shown in Equations 5/6/7. To comprehensively evaluate the effectiveness of these components, we conduct additional experiments with the following ablated versions of our object field along with our main experiments.

- Without $(\ell_{3d\_empty} + \ell_{3d\_obj})$: These two losses are designed to learn correct codes for empty 3D points. In this case, we optimize the object field component only by ℓ_{2d_obj} from 2D signals.
- With $(\ell_{3d\_empty} + \ell_{3d\_obj})$: We additionally learn correct codes for empty 3D points using such losses but with different Δd (0.025/0.05/0.10 meters).

From Tables 6/7/8, we find that Δd = 0.05 achieves better scene decomposition quality. It is also clear that the pipeline trained without $(\ell_{3d\_empty} + \ell_{3d\_obj})$ has worse AP scores than the pipeline trained with $(\ell_{3d\_empty} + \ell_{3d\_obj})$. In fact, different choices of Δd produce very close results, showing that our proposed supervision for empty 3D points is rather robust.

Synthetic	$AP^{0.75} \uparrow$				$AP^{0.90} \uparrow$			
Rooms	w/o	w/0.025	w/0.05	w/0.10	w/o	w/0.025	w/0.05	w/0.10
Bathroom	100.0	100.0	100.0	100.0	98.17	97.72	98.50	98.16
Bedroom	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Dinning	99.49	99.55	99.66	99.49	81.87	81.77	81.72	81.91
Kitchen	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Reception	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Rest	99.88	99.89	99.89	99.89	98.77	98.97	99.03	99.03
Study	98.77	98.81	98.86	98.71	92.19	92.31	92.15	92.12
Office	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Average	99.77	<u>99.78</u>	**99.80**	99.76	96.38	96.35	**96.43**	<u>96.40</u>

Table 6: Quantitative results of scene decomposition from our method on DM-SR dataset. AP scores with IoU thresholds at 0.75 and 0.90 are reported.

Reconstructed	$AP^{0.75} \uparrow$				$AP^{0.90} \uparrow$			
Rooms	w/o	w/0.025	w/0.05	w/0.10	w/o	w/0.025	w/0.05	w/0.10
Office_0	79.41	75.10	82.71	76.43	57.80	56.85	55.07	53.71
Office_2	82.77	80.83	81.12	83.70	64.68	65.02	68.34	61.18
Office_3	67.38	78.08	76.30	75.07	48.53	54.77	55.90	53.79
Office_4	65.51	64.72	70.33	73.94	47.17	50.13	53.68	53.60
Room_0	77.60	79.73	79.83	76.03	51.21	49.63	49.35	46.69
Room_1	87.63	88.85	92.11	86.14	69.20	67.78	74.21	63.32
Room_2	84.01	84.69	84.78	83.29	58.01	62.69	62.83	57.70
Average	77.76	78.86	**81.03**	<u>79.23</u>	56.66	<u>58.12</u>	**59.91**	55.71

Table 7: Quantitative results of scene decomposition from our method on Replica dataset. AP scores with IoU thresholds at 0.75 and 0.90 are reported.

Real-world Rooms	$AP^{0.75}$ ↑				$AP^{0.90}$ ↑			
	w/o	w/0.025	w/0.05	w/0.10	w/o	w/0.025	w/0.05	w/0.10
0010_00	95.03	94.13	94.82	94.62	86.29	86.72	90.45	86.51
0012_00	99.25	99.25	98.86	99.75	94.40	95.71	95.35	94.54
0024_00	78.98	84.04	93.25	78.09	56.72	58.08	72.01	53.20
0033_00	94.94	92.92	97.02	95.77	89.94	87.77	93.32	88.83
0038_00	96.57	97.02	99.17	96.58	93.43	92.96	97.58	93.82
0088_00	83.22	82.59	83.59	78.59	61.42	58.23	59.23	61.34
0113_00	92.69	92.84	98.67	98.67	85.40	85.42	96.61	96.61
0192_00	97.60	97.60	99.40	99.80	96.48	96.44	98.32	98.88
Average	92.29	92.55	**95.60**	<u>92.74</u>	83.01	82.66	**87.86**	<u>84.22</u>

Table 8: Quantitative results of scene decomposition from our method on ScanNet dataset. AP scores with IoU thresholds at 0.75 and 0.90 are reported.

Ablations of Object Field Training Strategy To comprehensively evaluate the object field training strategy, we conduct additional experiments with the following ablated versions of training strategy along with our main experiments.

- Without $Detach$: The gradients from object field component can backpropagate to the backbone of NeRF. In this case, the object field and the rendering parts will mutually influence each other.
- With $Detach$: The object field component only depends on the output representation of NeRF and will not affect the rendering part at all.

From Tables 9/10/11, we find the training strategy with $Detach$ achieves better scene rendering and decomposition quality in general. It is clear that the rendering quality drops significantly if our object field component is trained without $Detach$. In contrast, when our object field component is trained with $Detach$, better scene rendering quality and comparable decomposition quality are obtained. In this paper, the training strategy with $Detach$ is adopted.

	PSNR↑		LPIPS↑		SSIM↓		$AP^{0.75}$ ↑	
Detach	w/	w/o	w/	w/o	w/	w/o	w/	w/o
Bathroom	44.05	32.05	0.994	0.944	0.009	0.150	100.0	100.0
Bedroom	48.07	32.70	0.996	0.927	0.009	0.255	100.0	100.0
Dinning	42.34	33.47	0.984	0.895	0.028	0.191	99.66	99.29
Kitchen	46.06	28.49	0.994	0.904	0.014	0.221	100.0	100.0
Reception	42.59	29.91	0.993	0.922	0.008	0.190	100.0	99.47
Rest	42.80	31.33	0.994	0.930	0.007	0.145	99.89	99.47
Study	41.08	32.08	0.987	0.935	0.026	0.161	98.86	98.88
Office	46.38	32.17	0.996	0.935	0.006	0.162	100.0	100.0
Average	**44.17**	31.53	**0.992**	0.924	**0.013**	0.185	**99.80**	99.71

Table 9: Quantitative results of our method on DM-SR dataset.

	PSNR↑		LPIPS↑		SSIM↓		$AP^{0.75}$ ↑	
Detach	w/	w/o	w/	w/o	w/	w/o	w/	w/o
Office_0	40.66	28.20	0.972	0.781	0.070	0.422	82.71	82.95
Office_2	36.98	27.98	0.964	0.837	0.115	0.361	81.12	81.69
Office_3	35.34	26.68	0.955	0.817	0.078	0.366	76.30	72.63
Office_4	32.95	27.19	0.921	0.804	0.172	0.363	70.33	77.34
Room_0	34.97	25.18	0.940	0.682	0.127	0.403	79.83	82.26
Room_1	34.72	26.54	0.931	0.717	0.134	0.425	92.11	93.71
Room_2	37.32	27.43	0.963	0.786	0.115	0.392	84.78	83.21
Average	**36.13**	27.03	**0.949**	0.775	**0.116**	0.390	81.03	**81.97**

Table 10: Quantitative results of our method on Replica dataset.

	PSNR↑		LPIPS↑		SSIM↓		$AP^{0.75}$ ↑	
Detach	w/	w/o	w/	w/o	w/	w/o	w/	w/o
0010_00	26.82	22.30	0.809	0.697	0.381	0.513	94.82	97.44
0012_00	29.28	22.98	0.753	0.601	0.389	0.546	98.86	97.67
0024_00	23.68	19.41	0.705	0.552	0.452	0.573	93.25	90.45
0033_00	27.76	22.39	0.856	0.743	0.342	0.470	97.02	97.48
0038_00	29.36	24.79	0.716	0.614	0.415	0.573	99.17	98.42
0088_00	29.37	23.87	0.825	0.692	0.386	0.513	83.59	85.45
0113_00	31.19	22.93	0.878	0.727	0.320	0.498	98.67	99.00
0192_00	28.19	21.97	0.732	0.576	0.376	0.549	99.40	98.75
Average	**28.21**	22.58	**0.784**	0.650	**0.383**	0.529	**95.60**	95.58

Table 11: Quantitative results of our method on ScanNet dataset.

Scene Manipulation and Decomposition To better demonstrate the superiority of our DM-NeRF that simultaneously reconstructs, decomposes, manipulates and renders complex 3D scenes in a single pipeline, we conduct additional experiments with the following comparison of scene decomposition along with our main experiments.

- Decomposition after Manipulation: We manipulate objects within a scene and generate corresponding object masks for decomposition using Mask-RCNN with the weights trained on the same scene before manipulation.
- Simultaneous Manipulation and Decomposition: We appeal to our DM-NeRF to simultaneously manipulate and decompose a scene. The weights we used are directly from the same scene but without manipulation.

From Table 12, we can see that, for the same scene, the AP scores reported by Mask-RCNN (He et al., 2017) has an obvious decrease after manipulation. However, our method presents very close AP scores for all scenes before and after manipulation. Fundamentally, this is because Mask-RCNN only considers every 2D image independently for object segmentation, while our DM-NeRF explicitly leverages the consistency between 3D and 2D across multiple views.

Metric	$AP^{0.5}$ ↑				$AP^{0.75}$ ↑			
Method	Mask-RCNN		**Ours**		Mask-RCNN		**Ours**	
Manipulation	Before	After	Before	After	Before	After	Before	After
Bathroom	97.90	96.36	100.0	99.38	93.81	88.89	100.0	97.57
Bedroom	98.91	97.14	100.0	100.0	97.92	94.84	100.0	99.38
Dinning	98.85	98.20	100.0	99.15	98.85	96.33	99.66	97.14
Kitchen	92.06	93.56	100.0	100.0	92.04	91.39	100.0	98.75
Reception	98.81	97.03	100.0	100.0	98.81	94.63	100.0	99.40
Rest	98.89	97.18	100.0	100.0	98.89	95.86	99.89	99.86
Study	96.87	97.64	99.69	99.41	96.86	95.75	98.86	98.38
Office	98.93	89.97	100.0	100.0	97.83	74.24	100.0	75.94
Average	97.65	95.88	**99.96**	<u>99.74</u>	96.87	91.49	**99.80**	<u>95.80</u>

Table 12: Quantitative results of scene decomposition from our method and Mask-RCNN on DM-SR dataset. AP scores with IoU thresholds at 0.5 and 0.75 are reported.

D Additional Qualitative Results

Scene Rendering and Decomposition Figures 10/11/12 show additional qualitative results of scene rendering and decomposition.

Scene Manipulation Figures 13/14/15 shows additional qualitative results of scene manipulation with single/multiple objects under various transformations.

More qualitative results can be found in the supplied video.

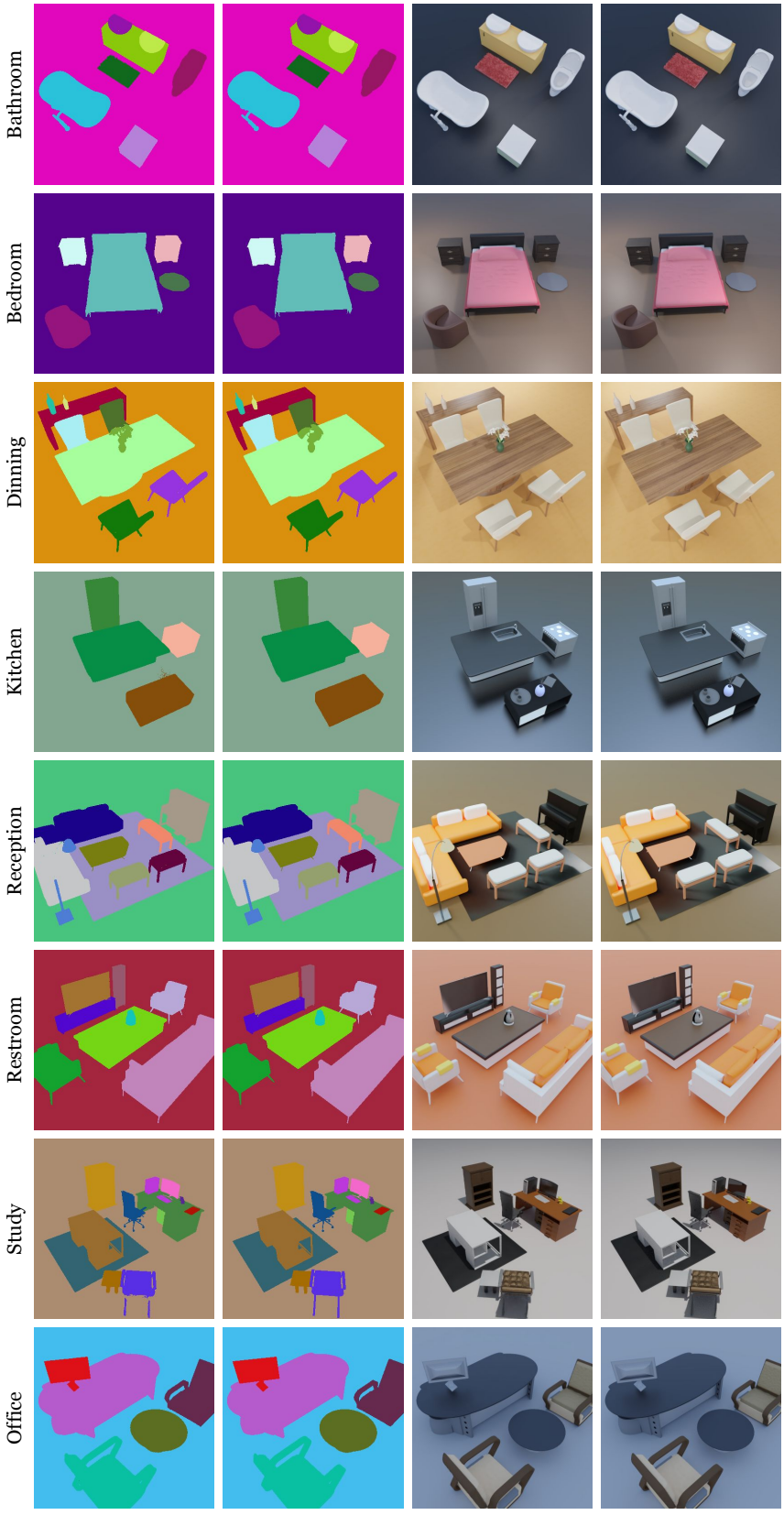


Figure 10: Qualitative results for scene rendering and decomposition on DM-SR.

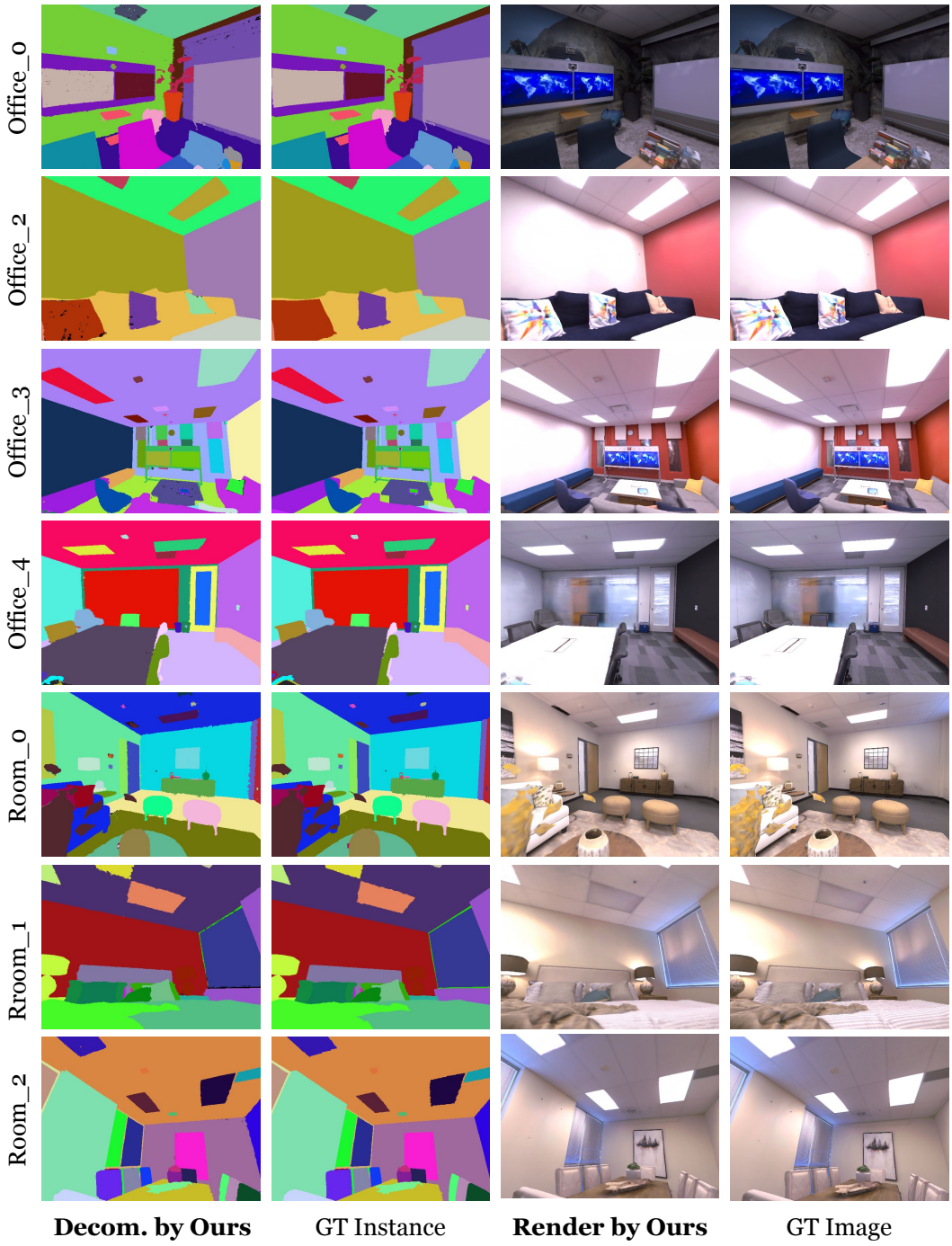


Figure 11: Qualitative results for scene rendering and decomposition on Replica.

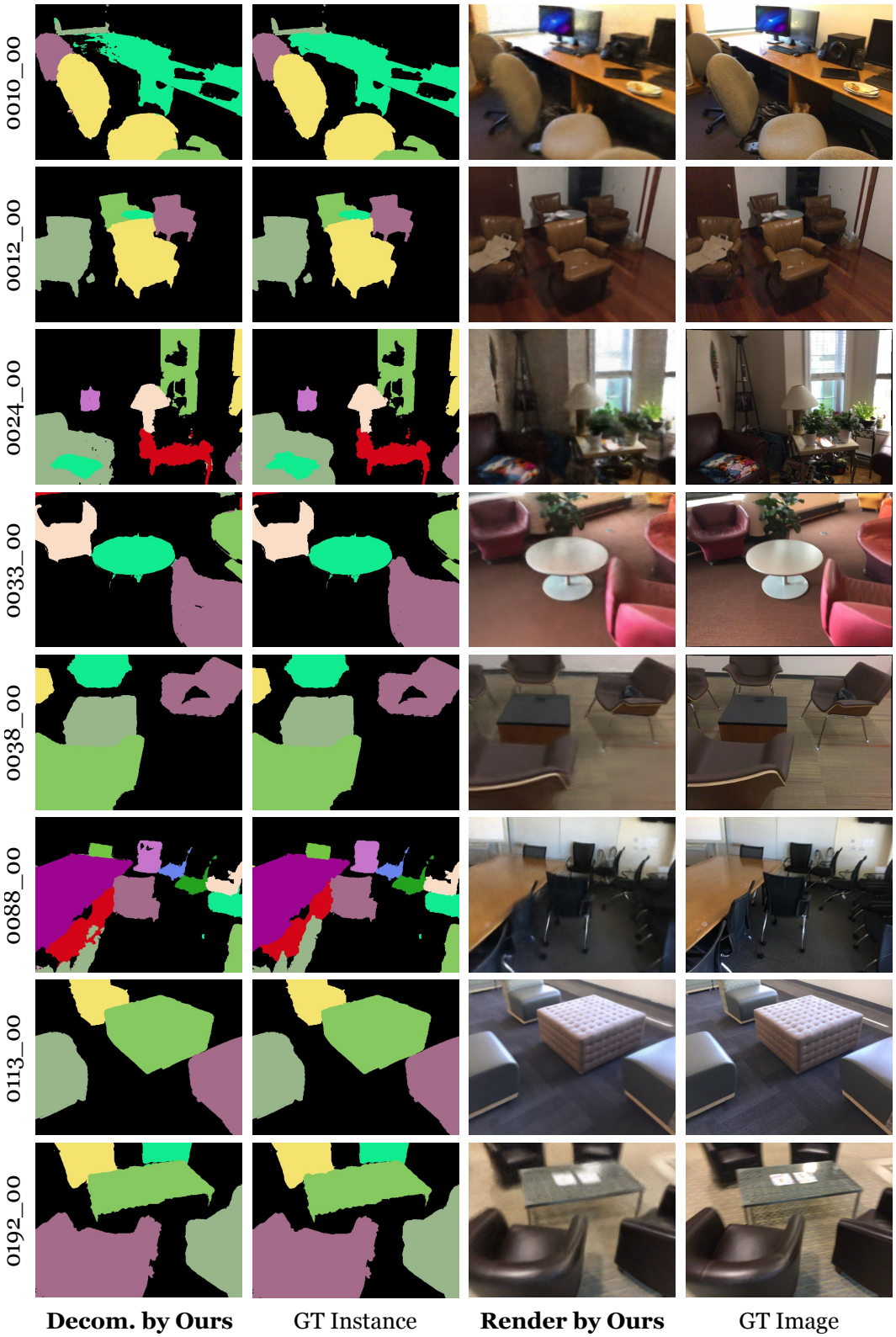


Figure 12: Qualitative results for scene rendering and decomposition on ScanNet.

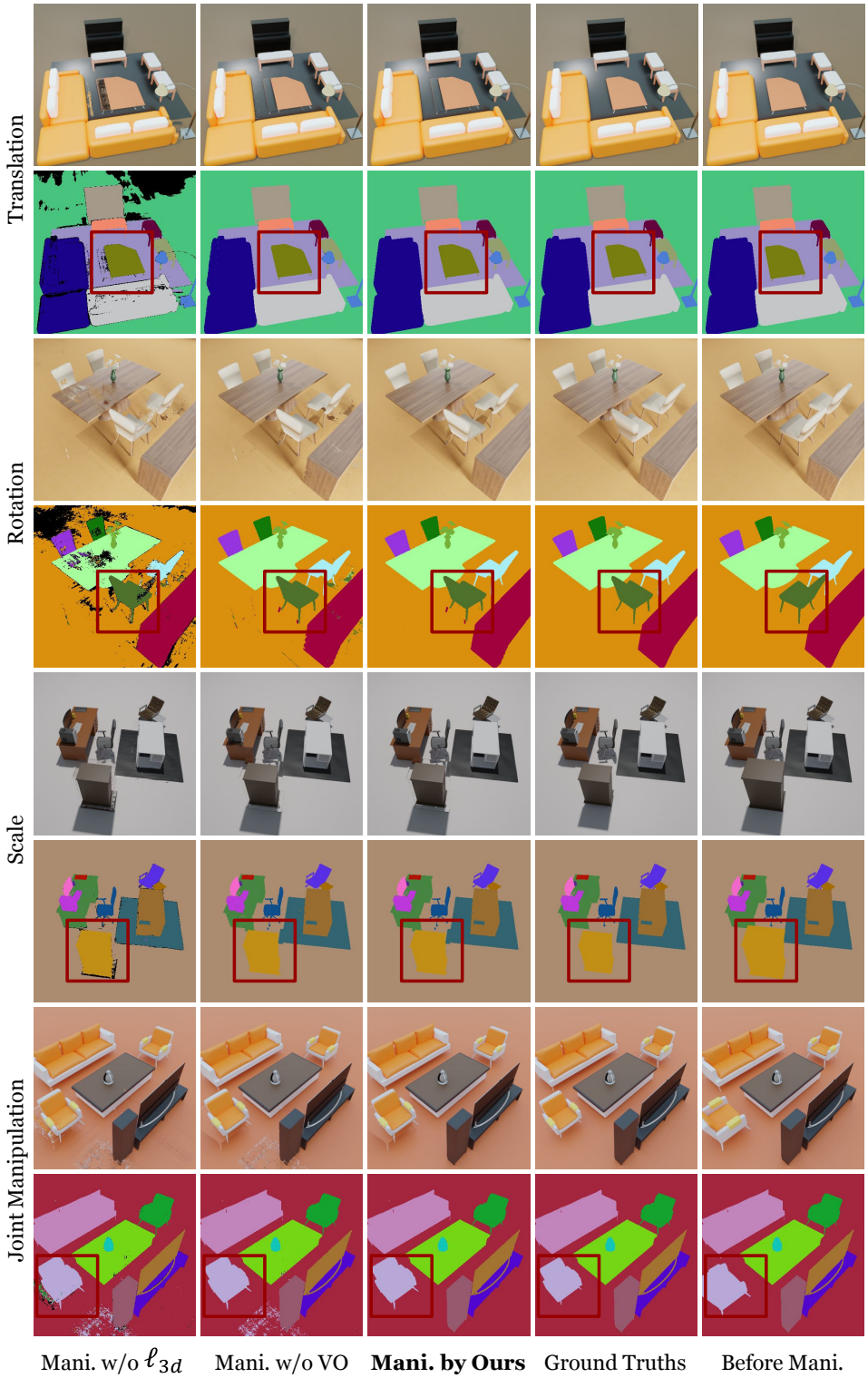


Figure 13: Qualitative results of our method for object manipulation on DM-SR dataset. The dark red boxes highlight the differences.

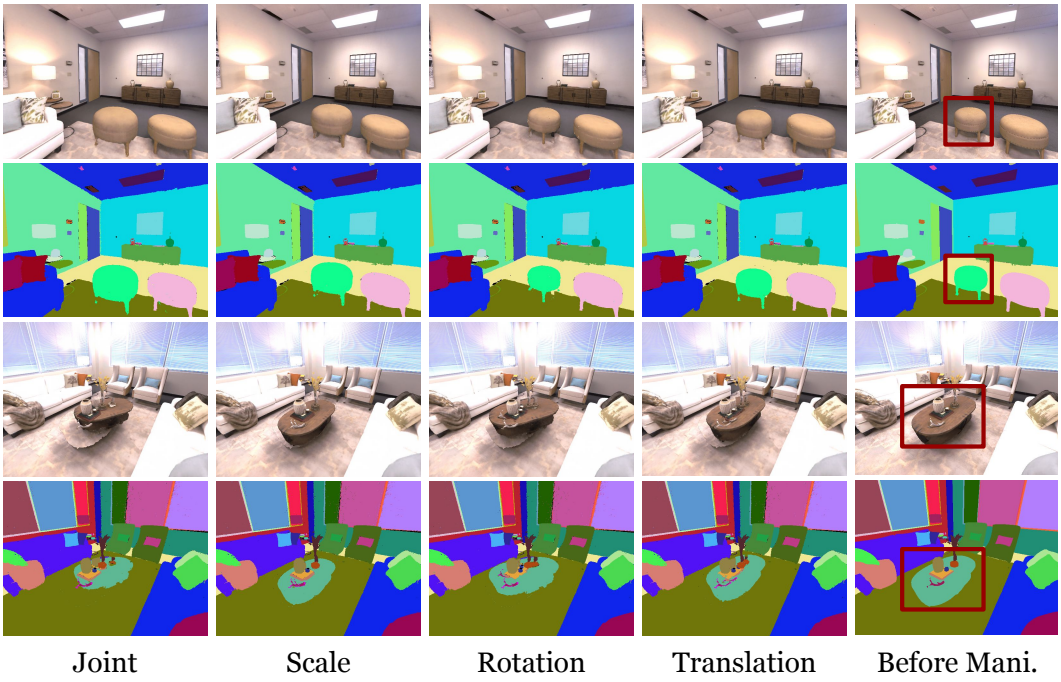


Figure 14: Qualitative results for single object manipulation on Replica dataset. The dark red box in the rightmost column highlights the manipulated object.

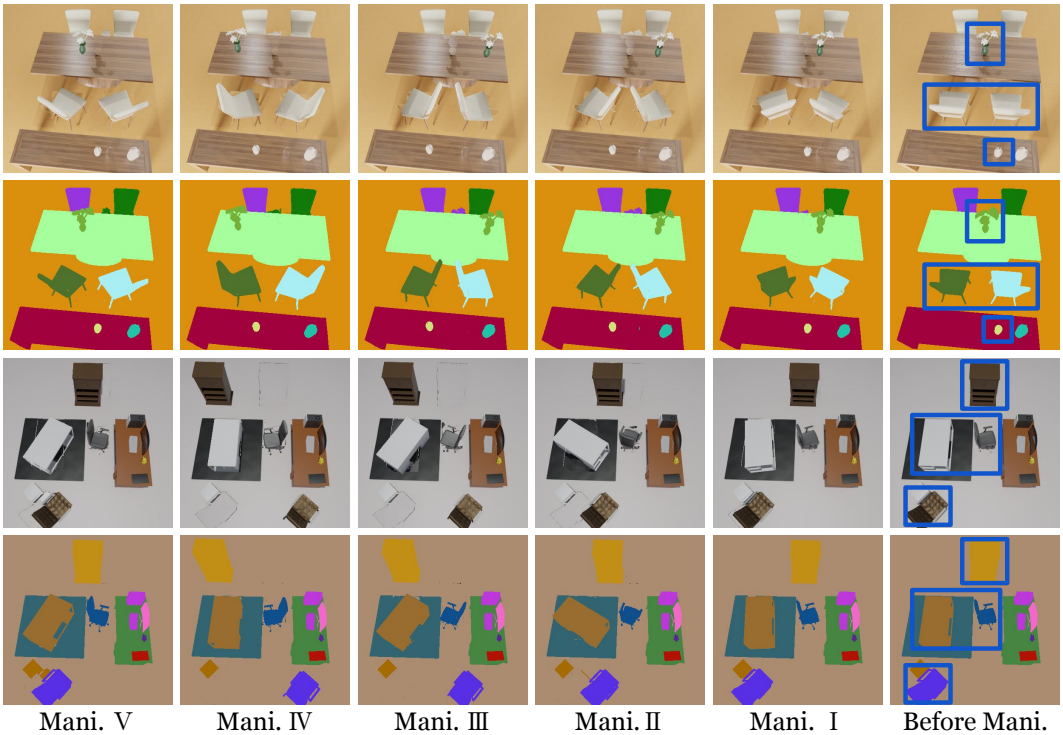


Figure 15: Qualitative results for multiple objects manipulation on DM-SR dataset. The dark blue boxes in the rightmost column highlight the manipulated objects.

E SUMMARY OF CHANGES

This manuscript was initially submitted to ECCV'22 with mixed feedback (two acceptances, one rejection). The review comments and rebuttal letter are provided in the supplementary materials. Thanks to the valuable suggestions from the peer reviewers, the current ICLR submission is clearly improved including clarified presentation, more experiments, and consistent results. The key changes are summarized as follows:

- The title is changed from "OMG: 3D Scene Geometry Decomposition and Manipulation from 2D Images" to "DM-NeRF: 3D Scene Geometry Decomposition and Manipulation from 2D Images".
- Regarding the relevant works ObjectNeRF (Yang et al., 2021) and Semantic-NeRF (Zhi et al., 2021a), we have added more clarifications in Section 4.2 "The most relevant work to us is Object-NeRF, its design is vastly different ...". Considering that our pipeline is only trained and tested with 2D images and masks, the powerful Mask-RCNN (He et al., 2017) is used as a solid baseline.
- Regarding our manipulation algorithm, we have added more explanations in Section 3.2 "Intuitively, there could be two strategies ...". To sum up, our method is NOT equivalent to a 2D image manipulation. Instead, it is 3D manipulation, which fully exploits the accurate 3D object code, volume density, and color $\{\boldsymbol{o}, \sigma, \boldsymbol{c}\}$ empowered by our object field component and NeRF.
- Regarding the decomposition results of Mask-RCNN (He et al., 2017), we have further conducted four groups of experiments by setting four different training modes with the Mask-RCNN, and the experimental results can be found in Table 5. Clearly, the fourth Group (G_4) achieves the highest scores, which also has the fairest experimental settings we can set up for comparison.
- In addition, we have added more qualitative results of our proposed DM-NeRF. **First**, Figure 7 and the video DM-NeRF_2D.mp4 (23$\sim$34s) present the performance on deformation manipulation with different deformable transformations, further indicating that our method allows any interested object to be freely manipulated. **Second**, the video DM-NeRF_3D.mp4 demonstrates that the proposed DM-NeRF can reconstruct and decompose 3D scenes only from 2D images.
- Lastly, details of dataset generation, experimental setup, and discussion of more recent works are also provided in this manuscript accordingly.

Overall, we take into account all feedback from previous reviewers and area chairs, and carefully address them in this ICLR submission.

View Meta-Reviews

Paper ID
6096

Paper Title
OMG: 3D Scene Geometry Decomposition and Manipulation from 2D Images

META-REVIEWER #1

META-REVIEW QUESTIONS

2. Meta-review: Consolidation report explaining the decision for the submission, based on the reviews, the rebuttal, and the discussion with reviewers and AC-buddies.
Two reviewers are positive in their assessment of this paper while one reviewer has expressed major concerns. Most of these were addressed clearly in the rebuttal, but a key one remains regarding the image manipulation. This causes one of the reviewers to remain negative about accepting the paper. The AC somewhat agrees with this reviewer, in that it is unclear what exactly the contribution of the proposed method is. If it is the fact that encoding object instance IDs into a NeRF representation enable image manipulation, the experimental results are weak. They have artifacts as pointed out by R4 and the overall quality is only marginally better than some basic baselines. If the main contribution of the paper is the inclusion of object codes in NeRF, then why is it not compared with other NeRF methods that incorporate semantic segmentation as an output, most representative of which would be Semantic-NeRF? The scene decomposition should be compared with those works, not just Mask-RCNN. On balance, although the idea appears to be interesting, its advantages both in representing a scene and in what it enables are not fully validated with experimental results. The paper would benefit from a more careful and thorough revision including additional comparative and ablation studies.

META-REVIEWER #2

META-REVIEW QUESTIONS

2. Meta-review: Consolidation report explaining the decision for the submission, based on the reviews, the rebuttal, and the discussion with reviewers and AC-buddies.
We agree with the primary AC on the decision to reject this
submission. The decision has been agreed upon during the AC meeting,
where the paper, the reviews, [the rebuttal,], and the discussion
board were all taken into account.

View Reviews

Paper ID
6096

Paper Title
OMG: 3D Scene Geometry Decomposition and Manipulation from 2D Images

Reviewer #1

Questions

2. Summarize the paper's claimed primary contributions: In 5-7 sentences, describe the key ideas, results, findings, and significance as claimed by the paper's authors.
The authors introduce an object field component to learn unique codes for all individual objects in 3D space only from 2D supervision. To best of my knowledge, it's the first work to handle 3D space decomposition, manipulation and 3D reconstruction together.

3. What do you see as the main strengths of this work? Consider, among others, the significance of critical ideas, validation, writing quality, and data contribution. Explain clearly why these aspects of the paper are valuable. ACs are instructed to ignore unsupported responses.
The overall idea in novel. The structure and written is very clear. The authors firstly introduce the difficulty of the problems and provide solid solution to the problem. More specifically, the authors firstly mentioned that 2D singal supervision from 3D is non-trival and several losses were proposed to tackle the problem. I'm sorry I'm not familiar with manipulation.

4. What do you see as the main weaknesses of this work? Clearly explain why these are weak aspects of the paper, e.g., why a specific prior work has already demonstrated the key contributions, why the experiments are insufficient to validate the claims, etc. ACs are instructed to ignore unsupported responses.
In my opinions, place the overall algorithm before detail might better.

5. Reproducibility: Could the work be reproduced by a talented graduate student from the information in the paper?
Agreement accepted

6. [Rate the paper as it stands now (pre-rebuttal). Borderline will not be an option for your final post-rebuttal recommendation, and so it should only be used rarely now.]
Accept

7. Justify your rating. Be specific: What are the most critical factors in your rating? What points should the authors cover in their rebuttal? Your reply should clearly explain to the authors what you need to see in order to increase your rating.
I'm satisfied with the paper.

11. Justify your post-rebuttal assessment. Acknowledge any rebuttal and be specific about the final factors for and against acceptance that matter to you. (Will be visible to authors after author notification)
Eventhough there are artifacts in the video and it's only 2D manipulation, I think it's a good attempt.

12. Give your final rating for this paper. Don't worry about poster vs oral. Consider the input from all reviewers, the authors' feedback, and any discussion. (Will be visible to authors after author notification)
Weak Accept. I tend to vote for accepting this submission, but rejecting it would not be that bad.

Reviewer #2

Questions

2. Summarize the paper's claimed primary contributions: In 5-7 sentences, describe the key ideas, results, findings, and significance as claimed by the paper's authors.
In this paper, the authors propose a framework to simultaneously recover 3D scene geometry, segment individual objects in 3D space, as well as support object manipulation, only from 2D images. To achieve this, they present an object field learning module to learn object-level representation and an inverse query algorithm to manipulate 3D objects. They conduct experiments on several datasets, which demonstrate the superiority of their framework.

3. What do you see as the main strengths of this work? Consider, among others, the significance of critical ideas, validation, writing quality, and data contribution. Explain clearly why these aspects of the paper are valuable. ACs are instructed to ignore unsupported responses.
The paper is well-written and easy to follow. 3D reconstruction from 2D images is a cutting-edge research topic. Instead of incrementally improving the 3D scene reconstruction performance, e.g., NeRF, Semantic-NeRF, the authors try to solve another important problem, i.e., object-level decomposition and manipulation in 3D scenes, which may potentially be applied to more application scenarios.

4. What do you see as the main weaknesses of this work? Clearly explain why these are weak aspects of the paper, e.g., why a specific prior work has already demonstrated the key contributions, why the experiments are insufficient to validate the claims, etc. ACs are instructed to ignore unsupported responses.
1. The authors should discuss more about the limitations of their framework.
2. The experiment results should be further explained.

5. Reproducibility: Could the work be reproduced by a talented graduate student from the information in the paper?
Agreement accepted

6. [Rate the paper as it stands now (pre-rebuttal). Borderline will not be an option for your final post-rebuttal recommendation, and so it should only be used rarely now.]
Weak Accept

7. Justify your rating. Be specific: What are the most critical factors in your rating? What points should the authors cover in their rebuttal? Your reply should clearly explain to the authors what you need to see in order to increase your rating.
1. For color image synthesis evaluation, the authors only show PSNR/SSIM/LPIPS scores of their own framework, but fail to compare them with other methods/baselines. It's hard to judge the performance only from the standalone scores, e.g., how good of a PSNR score of 44.17? The authors should provide more information.
2. Although the authors claim that their framework shows good qualitative results, we can still observe many blurred boundaries around the manipulated objects. Moreover, from the Appendix-Manipulation.mp4, in 00:00:02 and 00:00:10, there is an unexpected moving object that appears in the

hole area of the bottom table. The authors should provide more intuitively/theoretically analysis about the limitations of their framework.

11. Justify your post-rebuttal assessment. Acknowledge any rebuttal and be specific about the final factors for and against acceptance that matter to you. (Will be visible to authors after author notification)
Thanks for the efforts to address my concerns.

12. Give your final rating for this paper. Don't worry about poster vs oral. Consider the input from all reviewers, the authors' feedback, and any discussion. (Will be visible to authors after author notification)
Weak Accept. I tend to vote for accepting this submission, but rejecting it would not be that bad.

Reviewer #4

Questions

2. Summarize the paper's claimed primary contributions: In 5-7 sentences, describe the key ideas, results, findings, and significance as claimed by the paper's authors.
This paper presents a method leveraging 2D instant segmentation for 3D scene geometry Decomposition and manipulation. This method builds upon NERF and adds an additional object code to encode the volume's object information. This volumetric object code is supervised by 2D ground truth with Hungarian matching after being rendered using a similar pipeline as the RGB pixels. Once the volumetric object code is fitted, the author proposes the "Inverse Query Algorithm" to manipulate the generate edited 2D images.

3. What do you see as the main strengths of this work? Consider, among others, the significance of critical ideas, validation, writing quality, and data contribution. Explain clearly why these aspects of the paper are valuable. ACs are instructed to ignore unsupported responses.
The paper is well structured and easy to follow. The method is explained in an intuitive way. This paper attempts to address a valid problem and proposed "OMG-SR" for quantitative evaluation of geometry manipulation tasks which could be beneficial to the community.

4. What do you see as the main weaknesses of this work? Clearly explain why these are weak aspects of the paper, e.g., why a specific prior work has already demonstrated the key contributions, why the experiments are insufficient to validate the claims, etc. ACs are instructed to ignore unsupported responses.
1. The main concern of this paper is the manipulation method. Instead of manipulating individual objects, the proposed method moves the scene as a whole (what L373 does is equivalent to moving the rendering camera), crops out the 2D area guided by the target object code, and pastes it into the original image. This can be justified by the artifacts that appear in the supplementary video eg. the first scene in "Manipulation.mp4", where the bigger vase on the table on the right appears in the hole where the left bottle is originally was during object translation.

Overall, the proposed method is equivalent to a 2D image manipulation method and does not exploit the rich 3D information empowered by NERF.

2. Since [41] is mainly solving the same problem(editing), I still think comparing with [41] can be beneficial.

3. Is finetune the MaskRCNN on every single scene necessary? Why not simply use the GT 2D segmentation mask? How does a finetune-free MaskRCNN perform?

4. I found it hard to associate the abbreviation “OMG” with the proposed pipeline. But this is not a major issue and can be my personal preference.
5. Adding comments to L379-L383 can be beneficial for better understanding the algorithm.

5. Reproducibility: Could the work be reproduced by a talented graduate student from the information in the paper?
Agreement accepted

6. [Rate the paper as it stands now (pre-rebuttal). Borderline will not be an option for your final post-rebuttal recommendation, and so it should only be used rarely now.]
Weak Reject

7. Justify your rating. Be specific: What are the most critical factors in your rating? What points should the authors cover in their rebuttal? Your reply should clearly explain to the authors what you need to see in order to increase your rating.
The most critical factor that leads to my rating is the proposed manipulation algorithm, as to my understanding, it does not give advantages compare to pure 2D image manipulation. However, I’m open to increasing the rating. If the author can convince me otherwise.

11. Justify your post-rebuttal assessment. Acknowledge any rebuttal and be specific about the final factors for and against acceptance that matter to you. (Will be visible to authors after author notification)
I do appreciate the authors' effort in addressing my concerns. I'm satisfied with the responses on concerns 2-5, but the main concern remains. I agree that the proposed method works on Strategy 1 (first edit the object in 3D space, and then project it into 2D images). But after revising the algorithm a few times, I still believe this is equivalent to 2D manipulation. We can go through the core of the algorithm line by line: L380 and L381 are equivalent to "put the rendered target object after manipulation on top". L382 is equivalent to "set the original rendered target object in the image to transparent". L383: "don't edit the image where no manipulation occurs". A pure 2D way to achieve the same thing can be: "1. select the target object in the original rendered image, and set it to transparent. 2. move the camera and render a different view. 3. crop out the target object in the new render image and paste it on top of the original image. 4. paste the rest of the new render image as a background layer to fill the transparent hole in the original image."

I understand that getting the 2D segmentation mask requires 3D object code and the proposed pipeline exploits NeRF's capability, but overall, the proposed manipulation method generates way too similar results compare to a 2D image editing method.

Thus, I am still not confident enough to accept this submission.

12. Give your final rating for this paper. Don't worry about poster vs oral. Consider the input from all reviewers, the authors' feedback, and any discussion. (Will be visible to authors after author notification)
Weak Reject. I tend to vote for rejecting this submission, but accepting it would not be that bad.

OMG: 3D Scene Geometry Decomposition and Manipulation from 2D Images

We sincerely thank all reviewers for the valuable comments.

(1) Clarification of our Manipulation Algorithm (R#4)
Intuitively, there could be two strategies for manipulation:

- **Strategy 1:** Firstly edit objects in 3D space, and then project into 2D images;
- **Strategy 2:** Firstly project 3D scene into 2D images, and then edit objects in 2D space.

Essentially, our proposed algorithm follows **Strategy 1** and aims to edit every 3D target point in 3D space, as clearly described by the *for loop* in lines 372-383. This is easily achieved by comparing whether the estimated object code of a 3D query point is exactly matched with the target object code. Given a novel viewing angle with a bunch of light rays, the millions of 3D points sampled on the rays are firstly edited using our Algorithm 1, and then rendered into a 2D image, as described by the last line 386.

To sum up, our method is NOT equivalent to a 2D image manipulation. Instead, it fully exploits the accurate object code, volume density, and color $\{\boldsymbol{o}, \sigma, \boldsymbol{c}\}$ empowered by our Object Field Component and NeRF.

We appreciate the feedback of R#4 and will add the clarification in the next version.

(2) Comparison with Yang *et al*. [1] (R#4)
At first glance, the existing method Yang *et al*. [1] addresses a similar problem with us. However, after carefully checking its recently released code[2], we confirm that its design principle and the amount of input information required for training and editing are vastly different from our method.

- First, [1] requires a point cloud as input for voxel initialization during training, but we do not.
- Second, [1] needs GT bounding boxes of target objects to manually prune point samples during editing. Put another way, it needs manual annotations on novel views to finish editing. However, we do not need any annotations during editing, just relying on the estimated codes.
- Third, [1] only learns a binary classifier to segment the foreground object and background, by pre-defining an Object Activation Code Library during training and editing. However, the object code of our method is completely learned from scratch.

Above all, it is inappropriate to compare with [1] due to the fundamental differences in design and evaluation. To respect prior art, we will add clarification in next version.

(3) Comparison with MaskRCNN (R#4)
As requested, we compare four groups of experiments for MaskRCNN on the selected eight scenes of ScanNet.

	Group 1	Group 2	Group 3	Group 4
Average $AP^{0.5}$	91.53	93.31	92.55	**93.85**

Table 1. Average scores of MaskRCNN on 8 scenes of ScanNet.

- **Group 1:** Train a single MaskRCNN model on the 8 scenes together from scratch, and then evaluate.
- **Group 2:** Load a single pretrained MaskRCNN model, and then finetune and evaluate it on the 8 scenes together.
- **Group 3:** Train 8 MaskRCNN models on the 8 scenes separately from scratch, and then evaluate respectively.
- **Group 4:** Load 8 copies of the pretrained model, and then finetune and evaluate on 8 scenes separately, see Sec 4.3.

Table 1 shows that Group 4 achieves the highest score, which also has the fairest experimental settings we can set up in paper. We are open to add these results in next version.

(4) Comparison with Image Synthesis Methods (R#2)
Basically, our paper does not aim at improving the quality of novel view synthesis, but simply uses the success of NeRF as it is. In particular, as supplied in Section C of appendix, we adopt the **detach** strategy to disjointly train the NeRF backbone and our object code component, avoiding the adversarial effects of mutual optimization.

Naturally, as expected by R#2, given more advanced NeRF variants as the backbone, the image synthesis quality is likely to be improved. We will discuss it in next version.

(5) Explanation and Analysis of Video Demo (R#2, #4)
We agree that there are artifacts, *e.g*., the hole after the vase in the video. Yet, they cannot be addressed in our current NeRF based pipeline. Fundamentally, this is because the 3D geometry and appearance in the never-seen volume space, *e.g*., the button of a vase, cannot be accurately interpolated by the NeRF backbone. Nevertheless, we believe that these issues may be alleviated by introducing more geometric constraints. For example, the neighbouring 3D points are likely to be the same object and also share similar colors. These priors can be designed as additional losses.

We will provide these analyses in the next version as suggested by R#2. We hope that our work can serve as the first baseline and inspire more sophisticated methods to thoroughly tackle the remaining issues in the future.

(6) Writing and Presentation (R#1, #2, #4)
We appreciate all comments about the writing and presentation. More details and a better structure will be provided to Algorithm 1. More discussion and analysis will be added for the experiments and framework limitations. The abbreviation will be reconsidered as well.

[1] Yang, B. *et al*. 'Learning Object-Compositional Neural Radiance Field for Editable Scene Rendering', ICCV, 2021.
[2] https://github.com/zju3dv/object_nerf

OMG: 3D Scene Geometry Decomposition and Manipulation from 2D Images

Anonymous ECCV submission

Paper ID 6096

Abstract. In this paper, we study the problem of 3D scene geometry decomposition and manipulation from 2D views. By leveraging the recent implicit neural representation techniques, particularly the appealing neural radiance fields, we introduce an object field component to learn unique codes for all individual objects in 3D space only from 2D supervision. The key to this component is a series of carefully designed loss functions to enable every 3D point, especially in non-occupied space, to be effectively optimized even without 3D labels. In addition, we introduce an inverse query algorithm to freely manipulate any specified 3D object shape in the learned scene representation. Notably, our manipulation algorithm can explicitly tackle key issues such as object collisions and visual occlusions. Our method, called OMG, is among the first to simultaneously reconstruct, decompose, manipulate and render complex 3D scenes in a single pipeline. Extensive experiments on three datasets clearly show that our method can accurately decompose all 3D objects from 2D views, allowing any interested object to be freely manipulated in 3D space such as translation, rotation and size adjustment.

Keywords: 3D Scene Decomposition, 3D Scene Manipulation

1 Introduction

In many cutting-edge applications such as mixed reality on mobile devices, users may desire to virtually manipulate objects in 3D scenes, such as moving a chair or making a flying broomstick in a 3D room. This would allow users to easily edit real scenes at fingertips and view objects from new perspectives. However, this is particularly challenging as it involves 3D scene reconstruction, decomposition, manipulation, and photorealistic rendering in a single framework [30].

A traditional pipeline could firstly reconstruct explicit 3D structures such as point clouds or polygonal meshes using SfM/SLAM techniques [27, 4], and then identify 3D objects followed by manual editing. However, these explicit 3D representations inherently discretize continuous surfaces, and changing the shapes often requires additional repair procedures such as remeshing [2]. In addition, such discretized and manipulated 3D structures can hardly retain geometric details with appearance, resulting in the generated novel views to be unappealing. Given this, it is worthwhile to design a new pipeline which can recover continuous 3D scene geometry only from 2D views and enable object decomposition and manipulation.

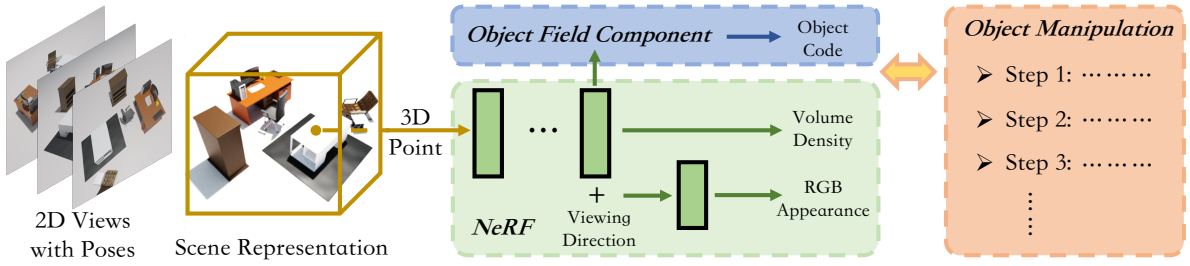


Fig. 1. The general workflow and components of our framework. The existing NeRF is used as our backbone shown by the green block, and we propose the object field and manipulation components shown by the blue and orange blocks.

Recently, implicit representations, especially NeRF [21], emerge as a promising tool to represent continuous 3D geometries from images. A series of succeeding methods [3, 5, 46] are rapidly developed to decouple lighting factors from structures, allowing free edits of illumination and materials. However, these methods fail to decompose the 3D scene geometry into individual objects. Therefore, it is not possible to manipulate individual object shapes in complex scenes. A handful of recent works [33, 45] have started to learn disentangled shape representations for potential geometry manipulation. However, they either focus on single objects or simple synthetic scenes, and can hardly extend to real-world 3D scenes with dozens of objects and severe visual occlusions. Another two recent works [41, 26] address the similar task as ours. However, the work [41] only decomposes a foreground object instead of segmenting every object, while the work [26] focuses on decomposing dynamic objects.

In this paper, we aim to design a new method to simultaneously recover continuous 3D scene geometry, segment all individual objects in 3D space, and support flexible object shape manipulation such as translation, rotation and size adjustment. In addition, the edited 3D scenes can be also rendered from novel views. However, this task is extremely challenging as it requires: 1) an object decomposition approach amenable to continuous and implicit 3D fields, without relying on any 3D labels for supervision due to the infeasibility of collecting labels in continuous 3D space; 2) an object manipulation method agreeable to the learned implicit and decomposed fields, with a ability to clearly address visual occlusions inevitably caused by manipulation.

To tackle these challenges, we introduce a simple pipeline, called **OMG**, which is built on the successful NeRF, but able to decompose the entire 3D space into **o**bject fields and freely **m**anipulate their **g**eometries for realistic novel view rendering. As shown in Figure 1, our method consists of three major components: 1) the existing radiance fields as backbone to learn volume density and appearance for every 3D point in space; 2) the object field which learns a unique object code for every 3D point; 3) the object manipulator which directly edits the shape of any specified object and automatically tackles visual occlusions.

The **object field** is the core of our framework. This component aims to predict a unique one-hot vector, *i.e.,* object code, for every 3D point in the entire scene space. However, to learn such code involves critical issues: 1) there

are no ground truth 3D object codes available for full supervision; 2) the number of total objects is variable and there is no fixed order for objects; 3) the non-occupied (empty) 3D space must be taken into account, but there are no labels for supervision as well. As detailed in Section 3.2, we show that our object field together with a series of careful designed loss functions can address them properly, under the supervision of color images with object 2D masks only.

Once the object field is well learned, our **object manipulator** aims to directly edit the geometry and render novel views when specifying the target objects, viewing angels, and manipulation settings. A naïve method is to obtain explicit 3D structures followed by manual editing and rendering, so that any shape occlusion and collision can be explicitly addressed. However, it is extremely inefficient to evaluate dense 3D points from implicit fields. To this end, as detailed in Section 3.3, we introduce a lightweight inverse query algorithm to automatically edit the scene geometry and appearance.

Overall, our pipeline can simultaneously recover 3D scene geometry, decompose and manipulate object instances only from 2D images. Extensive experiments on multiple datasets demonstrate that our method can precisely segment all 3D objects and effectively edit 3D scene geometry, without sacrificing high fidelity of novel view rendering. Our key contributions are:

- We propose an object field to learn a unique code for all 3D objects and non-occupied space only from 2D images.
- We propose an inverse query algorithm to effectively edit specified object shapes, while generating realistic scene images from novel views.
- We demonstrate superior performance for 3D decomposition and manipulation, and also contribute the first synthetic dataset for quantitative evaluation of scene editing.

2 Related Work

Explicit 3D Representations: To represent 3D geometry of objects and scenes from images or point clouds, voxel grids [7], octree [35], triangle meshes [16, 12], point clouds [10] and shape primitives [49] are widely used. Although impressive progress has been achieved in shape reconstruction [42, 40], completion [32], shape generation [18], and scene understanding [38, 11], the quality of these discrete shape representations are inherently limited by the spatial resolution and memory footprint. As a consequence, they are hard to represent complex 3D scenes.

Implicit 3D Representations: To overcome the discretization issue of explicit representations, coordinate based MLPs have been recently proposed to learn implicit functions to represent continuous 3D shapes. These implicit representations can be generally categorized as: 1) signed distance fields [28], 2) occupancy fields [20], 3) unsigned distance fields [6], 4) radiance fields [21], and 5) hybrid fields [39]. Among them, both occupancy fields and signed distance fields can only recover closed 3D shapes, and is hard to represent open scene

geometries. In the past two years, these representations have been extensively studied for novel view synthesis [25, 37] and 3D scene understanding [44, 47]. Thanks to the powerful representation capability, impressive results have been achieved, especially the neural radiance fields and its succeeding methods. In this paper, we also leverage the success of implicit representations, particularly NeRF, to recover the geometry and appearance of 3D scenes from 2D images.

3D Object Segmentation: To identify 3D objects from complex scenes, existing methods generally include 1) image based 3D object detection [22], 2) 3D voxel based detection methods [48] and 3) 3D point cloud based object segmentation methods [43]. Given large-scale datasets with full 3D object annotations, these approaches have achieved excellent object segmentation accuracy. However, they are designed to process explicit and discrete 3D geometries. Therefore, they are unable to segment continuous and fine-grained shapes, and fail to support geometry manipulation and realistic rendering. With the fast development of implicit representation of 3D scenes, it is desirable to learn object segmentation for implicit surfaces. To the best of our knowledge, this paper is among the first to segment all 3D objects of implicit representations for complex scenes, only with color images and 2D object labels for supervision.

3D Scene Editing: To edit 3D scenes from images, existing methods can be categorized as 1) appearance editing and 2) shape editing. A majority of works [31, 3, 46, 5] focus on lighting factor decomposition for appearance editing. Although achieving appealing results, these approaches cannot separately manipulate individual objects. A number of recent works [23, 19, 15, 33, 13] start to learn disentangled shape representations for potential geometry manipulation. However, they can only deal with single objects or simple scenes, without being able to learn unique object codes for precise shape manipulation and novel view rendering. In addition, there are also a plethora of works [36, 24, 9, 1] on generation based scene editing. Although they can manipulate the synthesized objects and scenes, they cannot discover and edit objects from real-world images.

3 OMG

3.1 Overview

Given a set of L images for a static scene with known camera poses and intrinsics $\{(\mathcal{I}_1, \boldsymbol{\xi}_1, \boldsymbol{K}_1) \cdots (\mathcal{I}_L, \boldsymbol{\xi}_L, \boldsymbol{K}_L)\}$, NeRF [21] uses simple MLPs to learn the continuous 3D scene geometry and appearance. In particular, it takes 5D vectors of query point coordinates $\boldsymbol{p} = (x, y, z)$ and viewing directions $\boldsymbol{v} = (\theta, \phi)$ as input, and predicts the volume density σ and color $\boldsymbol{c} = (r, g, b)$ for point $\boldsymbol{p}$. In our pipeline, we leverage this villina NeRF as the backbone to learn the continous scene representations, although other NeRF variants can also be used. Our method aims to decompose all individual 3D objects, and freely manipulate any object in the 3D scene space. To achieve this, we design an object field component to parallelly learn a unique object code for every query point $\boldsymbol{p}$, together with a object manipulator to edit the learned radiance fields and object fields.

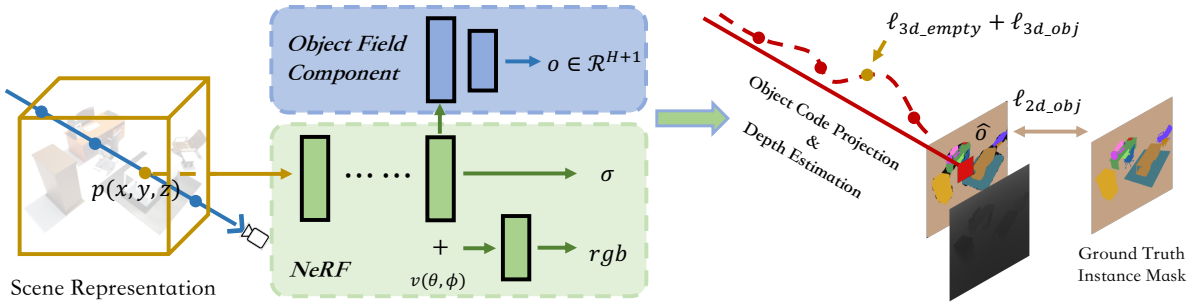


Fig. 2. The architecture of our pipeline. Given a 3D point $\boldsymbol{p}$, we learn an object code through a series of loss functions using both 2D and 3D supervision signals.

3.2 Object Fields

Object Field Representation: As shown in Figure 2, given the input point $\boldsymbol{p}$, we model the object field as a function of its coordinates, because the object signature of a 3D point is irrelevant to the viewing angles. The object field is represented by a one-hot vector $\boldsymbol{o}$. Basically, this one-hot object code aims to accurately describe the object ownership of any point in 3D space.

However, there are two issues involved here: 1) the total number of objects in 3D scenes are variable and it can be 1 or many; 2) the entire 3D space has a large non-occupied volume in addition to solid objects. To tackle these issues, we define the object code $\boldsymbol{o}$ as $H+1$ dimensional, where H is a predefined number of solid objects that the network are expected to predict in maximum. We can safely choose a relative large value for H in practice. The last dimension of $\boldsymbol{o}$ is particularly reserved to represent the non-occupied space. Notably, this careful design is crucial for tackling occlusion and collision during object manipulation discussed in Section 3.3. Formally, the object field is defined as:

$$\boldsymbol{o} = f(\boldsymbol{p}), \quad \text{where } \boldsymbol{o} \in \mathcal{R}^{H+1} \tag{1}$$

The function f is parameterized by a series of MLPs. If the last dimension of code $\boldsymbol{o}$ is 1, it represents the input point $\boldsymbol{p}$ is non-occupied or the point is empty.

Object Code Projection: Considering that it is infeasible to collect object code labels in continuous 3D space for full supervision while it is fairly easy and low-cost to collect object labels on 2D images, we aim to project the object codes along the query light ray back to a 2D pixel. Since the volume density σ learned by the backbone NeRF represents the geometry distribution, we simply approximate the projected object code of a pixel $\hat{\boldsymbol{o}}$ using the sampling strategy and volume rendering formulation of NeRF. Formally, it is defined as:

$$\hat{\boldsymbol{o}} = \sum_{k=1}^{K} T_k \alpha_k \boldsymbol{o}_k, \quad \text{where} \quad T_k = exp(-\sum_{i=1}^{k-1} \sigma_i \delta_i), \quad \alpha_k = 1 - exp(-\sigma_k \delta_k) \tag{2}$$

with K representing the total sample points along the light ray shooting from a pixel, σ_i representing the learned density of the i^{th} sample point, δ_k representing the distance between the $(k+1)^{th}$ and k^{th} sample points. From this projection formulation, we can easily obtain 2D masks of 3D object codes given the pose and camera parameters of any query viewing angles.

Object Code Supervision: Having the projected 2D object predictions at hand, we naturally choose 2D images with object annotations for supervision. However, there are two issues: 1) The number and order of ground truth objects can be very different across different views due to visual occlusions. For example, as to the same 3D object in space, its object annotation in image #1 can be quite different from its annotation in image #2. Therefore, it is non-trivial to correctly and consistently utilize 2D object annotations to supervise the network. 2) The 2D object annotations can only provide labels for 3D solid objects, because the non-occupied 3D space will never be recorded in 2D images. Therefore, it is impossible to directly supervise the non-occupied space, *i.e.*, the last dimension of $\hat{\boldsymbol{o}}$, from 2D annotations. Due to these issues, to simply borrow existing 2D object segmentation methods such as MaskRCNN [14] is ineffective, because they fundamentally do not consider the consistency between 3D and 2D.

To tackle the first issue, we use the Optimal Association and Supervision strategy proposed by 3D-BoNet [43]. As illustrated in Figure 3, assuming we generate L images of 2D object predictions and have the paired L images of 2D ground truth object labels, in every single image, we use Hungarian algorithm [17] to associate every ground truth 2D object mask with a unique predicted 2D object mask according to two criteria, Soft Intersection-over-Union (sIoU) and Cross-Entropy Score (CES) [43]. Once the predicted and ground truth 2D objects are matched, we directly maximize the average sIoU score and minimize the CES across all L images. Formally, the loss is defined as follows. Details of the calculation and matching are in appendix.

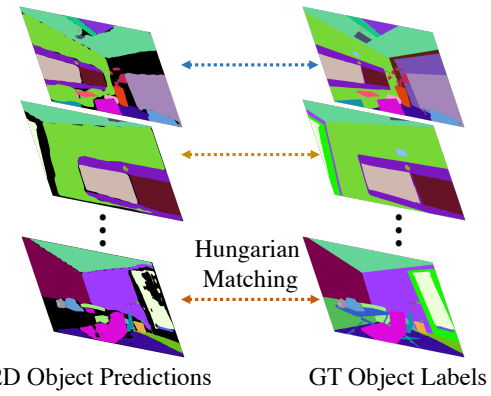


Fig. 3. Illustration of 2D object matching and supervision.

$$\ell_{2d\_obj} = \sum_{l=0}^{L}(-sIoU_l + CES_l) \tag{3}$$

Noted that, we only use the first H object predictions in every image to match with ground truth 2D objects because these are the solid objects, while the last dimension of $\hat{\boldsymbol{o}}$ is never used at this stage. After all, our optimal association strategy explicitly takes into account the consistency across all images from different viewing angles, driving the network to automatically learn a unique code for all 3D solid objects from only 2D labels.

To tackle the second issue, we turn to supervise the non-occupied object code in 3D space with the aid of estimated surface distances. In particular, given a specific query light ray on which we sample K 3D points to compute the projected 2D object code $\hat{\boldsymbol{o}}$, we simultaneously compute an approximate distance d between the surface and camera center along that query light:

$$d = \sum_{k=1}^{K} T_k \alpha_k \delta_k, \quad \text{where} \quad T_k = exp(-\sum_{i=1}^{k-1} \sigma_i \delta_i), \quad \alpha_k = 1 - exp(-\sigma_k \delta_k) \tag{4}$$

As illustrated in Figure 4, once we have the surface distance d at hand, we can easily know the relative position between every sample point k and the surface point s along the light ray. Naturally, we can then identify the subset of sample points surely belonging to empty space as indicated by green points, the subset of sample points near the surface as indicated by red points, and the remaining subset of sample points behind the surface as indicated by black points. Such geometric information provides critical signals to supervise empty space, *i.e.*, the last dimension of object code $\boldsymbol{o}$. Note that, the sample points behind the surface may not surely belong to empty space, and therefore we should not use them as supervision signals. Mathematically, we use the following kernel functions to obtain a surfaceness score s_k and an emptiness score e_k for the k^{th} sample point with the indicator function represented by $\mathbb{1}()$.

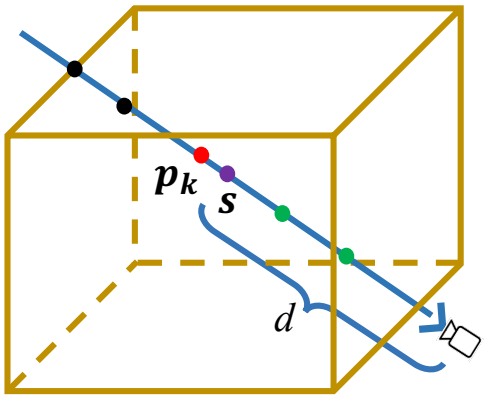


Fig. 4. Empty points identification from estimated surface distance.

$$s_k = exp\big(-(d_k - d)^2\big) \quad e_k = (1 - s_k) * \mathbb{1}(d - \Delta d - d_k > 0) \tag{5}$$

where d_k represents the distance between camera center and the k^{th} sample point, and Δd is a hyperparameter to compensate the inaccuracy of estimated surface distance d. Note that, the indicator function is used to mask out the sample points behind surface point during loss calculation.

Given the emptiness and surfaceness scores for the total K sample points along the ray, we use the simple log loss to supervise the last dimension of object code denoted as o_k^{H+1}.

$$\ell_{3d\_empty} = -\frac{1}{K}\sum_{k=1}^{K}\Big(e_k * log(o_k^{H+1}) + s_k * log(1 - o_k^{H+1})\Big) \tag{6}$$

Since there should be no solid object at all in the empty space, we apply the following loss on the first H dimensions of the object code to push them to be zeros. The h^{th} dimension of the k^{th} sample point's object code is denoted by o_k^h.

$$\ell_{3d\_obj} = -\frac{1}{K}\sum_{k=1}^{K}\Big(e_k * \sum_{h=1}^{H} log(1 - o_k^h)\Big) \tag{7}$$

To sum up, we firstly project object codes in 3D space back to 2D pixels along light rays using volume rendering equation, and then use optimal association strategy to compute the loss value with 2D object labels only. In addition, we introduce the key emptiness and surfaceness scores for points in 3D space with the aid of estimated surface distances. These unique scores are used to supervise the non-occupied 3D space. The whole object field is jointly supervised by the following loss:

$$\ell = \ell_{2d\_obj} + \ell_{3d\_empty} + \ell_{3d\_obj} \tag{8}$$

3.3 Object Manipulator

If we want to manipulate a specific 3D object shape such as translation, rotation and size adjustment, how does the whole scene look like in a new perspective after our manipulation? assuming that the object code and manipulation matrix can be precomputed from the users' interaction such as click, drag or zoom.

Fundamentally, this object manipulation task on implicit fields is to address the core question: how do we edit the codes $\sigma/\boldsymbol{c}/\boldsymbol{o}$ of every sample point along the query light ray, such that the generated novel view exactly shows the new appearance? This is nontrivial because:

- First, we need to address potential collisions between objects during manipulation. In fact, this is quite intuitive, thanks to our special design of the emptiness score in the last dimension of object code $\boldsymbol{o}$.
- Second, due to visual occlusions, the object codes behind surface points may not be accurate because they are not sufficiently optimized. By comparison, the projected object code $\hat{\boldsymbol{o}}$ along a light ray tends to be more accurate primarily because we have ground truth 2D labels for strong supervision.
- At last, we need a systematic procedure to update the codes with the known manipulation information. To this end, we design an inverse query approach.

Inverse Query: As illustrated in Figure 5, for any sample point $\boldsymbol{p}_k$ along a specific query light ray, given the *target* (*i.e.*, to-be-edited) object code $\boldsymbol{o}_t$ and its manipulation settings: relative translation $\Delta\boldsymbol{p} = (\Delta x, \Delta y, \Delta z)$, rotation matrix $\mathbf{R}^{3\times3}$, and scaling factor $t > 0$, we firstly compute an inverse point $\boldsymbol{p}_{k'}$, and then evaluate whether $\boldsymbol{p}_k$ and $\boldsymbol{p}_{k'}$ belong to the target object, and lastly decide to whether edit the codes or not. Formally, we introduce Inverse Query Algorithm 1 to conduct a single light ray rendering for object shape manipulation. Naturally, we can shoot a bunch of rays from any novel viewing angles to generate images of manipulated 3D scenes.

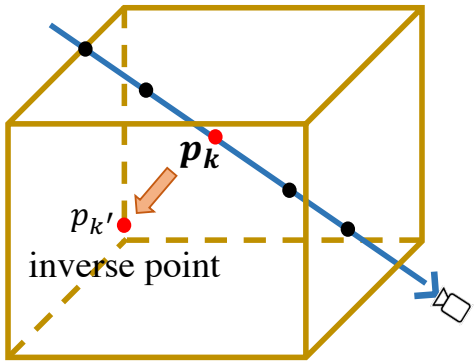


Fig. 5. An illustration of computing inverse points.

3.4 Implementation

In order to preserve the high-quality of image rendering, our loss in Equation 8 is only used to optimize the MLPs of object field branch. The backbone is only optimized by the original NeRF photo-metric loss [21]. The whole network is end-to-end trained together from scratch. The original position encoding and both coarse and fine networks are used. Our model is implemented in PyTorch and is trained on a single RTX3090 GPU. The single hyper-parameter for losses of our object field Δd is set as 0.05 meters in all experiments. Other hyperparameters are the same as NeRF [21] except for the batch size of rays which is set to 3072 to fully use the memory. All other implementation details are in appendix.

Algorithm 1 Our Inverse Query Algorithm to manipulate the learned implicit fields. (1) $\boldsymbol{o}_t$ is the target object code, one-hot with $H+1$ dimensions. $\{\Delta\boldsymbol{p}, \mathbf{R}^{3\times3}, t>0\}$ represent the manipulation information for the target object. (2) $\{\boldsymbol{p}_1 \cdots \boldsymbol{p}_k \cdots \boldsymbol{p}_K\}$ represent the K sample points along a specific query light ray $\boldsymbol{r}$. Note that, we convert all object codes into hard one-hot vectors for easy implementation.

Input:
- The target object code $\boldsymbol{o}_t$, manipulation information $\{\Delta\boldsymbol{p}, \mathbf{R}^{3\times3}, t\geq 0\}$;
- The sample points $\{\boldsymbol{p}_1 \cdots \boldsymbol{p}_k \cdots \boldsymbol{p}_K\}$ along a specific query light ray $\boldsymbol{r}$;

Output:
- The final pixel color $\bar{\boldsymbol{c}}$ rendered from the query ray $\boldsymbol{r}$ after manipulation;
- The final pixel object code $\bar{\boldsymbol{o}}$ rendered from the query ray $\boldsymbol{r}$ after manipulation;

Preliminary step:
- Obtain the projected pixel object code $\hat{\boldsymbol{o}}$ of light ray $\boldsymbol{r}$ before manipulation;

for $\boldsymbol{p}_k$ in $\{\boldsymbol{p}_1 \cdots \boldsymbol{p}_k \cdots \boldsymbol{p}_K\}$ **do**
- Compute the inverse point $\boldsymbol{p}_{k'}$ for $\boldsymbol{p}_k$: $\boldsymbol{p}_{k'} = (1/t)\mathbf{R}^{-1}(\boldsymbol{p}_k - \Delta\boldsymbol{p})$;
- Obtain the codes $\{\sigma_k, \boldsymbol{c}_k, \boldsymbol{o}_k\}$ for the point $\boldsymbol{p}_k$;
- Obtain the codes $\{\sigma_{k'}, \boldsymbol{c}_{k'}, \boldsymbol{o}_{k'}\}$ for the inverse point $\boldsymbol{p}_{k'}$;
- *Tackle visual occlusions:*
  if $\boldsymbol{o}_k = \boldsymbol{o}_t$ and $\boldsymbol{o}_k \neq \hat{\boldsymbol{o}}$ do: $\boldsymbol{o}_k \leftarrow \hat{\boldsymbol{o}}$
  Note: the target object is behind the surface but will be manipulated;
- Obtain new implicit codes $\{\bar{\sigma}_k, \bar{\boldsymbol{c}}_k, \bar{\boldsymbol{o}}_k\}$ for $\boldsymbol{p}_k$ after manipulation:
  if $\boldsymbol{o}_k \neq \boldsymbol{o}_t$ and $o_k^{H+1} \neq 1$ and $\boldsymbol{o}_{k'} = \boldsymbol{o}_t$ do: collision detected, EXIT.
  if $\boldsymbol{o}_k \neq \boldsymbol{o}_t$ and $\boldsymbol{o}_{k'} = \boldsymbol{o}_t$ do: $\{\bar{\sigma}_k, \bar{\boldsymbol{c}}_k, \bar{\boldsymbol{o}}_k\} \leftarrow \{\sigma_{k'}, \boldsymbol{c}_{k'}, \boldsymbol{o}_{k'}\}$;
  if $\boldsymbol{o}_k = \boldsymbol{o}_t$ and $\boldsymbol{o}_{k'} = \boldsymbol{o}_t$ do: $\{\bar{\sigma}_k, \bar{\boldsymbol{c}}_k, \bar{\boldsymbol{o}}_k\} \leftarrow \{\sigma_{k'}, \boldsymbol{c}_{k'}, \boldsymbol{o}_{k'}\}$;
  if $\boldsymbol{o}_k = \boldsymbol{o}_t$ and $\boldsymbol{o}_{k'} \neq \boldsymbol{o}_t$ do: $\{\bar{\sigma}_k, \bar{\boldsymbol{c}}_k, \bar{\boldsymbol{o}}_k\} \leftarrow \{0, \mathbf{0}, \mathbf{0}\}$;
  if $\boldsymbol{o}_k \neq \boldsymbol{o}_t$ and $\boldsymbol{o}_{k'} \neq \boldsymbol{o}_t$ do: $\{\bar{\sigma}_k, \bar{\boldsymbol{c}}_k, \bar{\boldsymbol{o}}_k\} \leftarrow \{\sigma_k, \boldsymbol{c}_k, \boldsymbol{o}_k\}$;

After the above *for loop*, every point $\boldsymbol{p}_k$ will get new implicit codes $\{\bar{\sigma}_k, \bar{\boldsymbol{c}}_k, \bar{\boldsymbol{o}}_k\}$. According to volume rendering equation, the final pixel color and object code are:
- $\bar{\boldsymbol{c}} = \sum_{k=1}^{K} \bar{T}_k \bar{\alpha}_k \bar{\boldsymbol{c}}_k, \quad \bar{\boldsymbol{o}} = \sum_{k=1}^{K} \bar{T}_k \bar{\alpha}_k \bar{\boldsymbol{o}}_k$
  where $\bar{T}_k = exp(-\sum_{i=1}^{k-1} \bar{\sigma}_i \delta_i), \quad \bar{\alpha}_k = 1 - exp(-\bar{\sigma}_k \delta_k)$.

4 Experiments

We quantitatively and qualitatively show superior performance for 3D scene decomposition and manipulation. The reader is urged to view our supplied video.

4.1 Datasets

OMG-SR: To the best of our knowledge, there is no existing 3D scene dataset suitable for quantitatively evaluation of geometry manipulation. Therefore, we create a **s**ynthetic dataset with 8 types of different and complex indoor **r**ooms, called OMG-SR. The room types and designs follow Hypersim dataset [29]. Each scene has a physical size of about $12 \times 12 \times 3$ meters with around 8 objects. We generate the following 5 groups of images.

– Group 1 (Before Manipulation): Each scene is rendered with color images and 2D object masks at 400×400 pixels from viewpoints sampled on the

upper hemisphere. Each scene generates 300 views for training and 100 for testing. The rendering trajectories follow NeRF synthetic dataset [21].
- Group 2 (Translation Only): One object in each scene is picked up to be translated along X or Y axis with about 0.3 meter. Then 100 views are generated for testing at the same testing viewpoints in Group 1.
- Group 3 (Rotation Only): One object in each scene is picked up to be rotated around z axis with about 90 degrees. Then 100 views are generated for testing at the same testing viewpoints in Group 1.
- Group 4 (Scaling Only): One object in each scene is picked up to be scaled down about 0.8× smaller. Then 100 views are generated for testing at the same testing viewpoints in Group 1.
- Group 5 (Joint Translation/Rotation/Scaling): One object in each scene is picked up to be simultaneously translated about 0.3 meter, rotated about 90 degrees, scaled down about 0.8× smaller. Then 100 views are generated for testing at the same testing viewpoints in Group 1.

Overall, we create 8 indoor scenes, firstly render the static scenes, and then manipulate each scene followed by second round rendering. We will release this dataset and keep updating it for future research in the community.

Replica: Replica [34] is a reconstruction-based 3D dataset of high fidelity scenes. We request the authors of Semantic-NeRF [47] to generate (180 training + 180 testing) color images and 2D object masks with camera poses at 640×480 pixels for each of 7 scenes. Each scene has $59 \sim 93$ objects with very diverse sizes. Details of camera settings and trajectories can be found in [47].

ScanNet: ScanNet [8] is a large-scale challenging real-world dataset. We select 8 scenes for evaluation. Each scene has about 3000 raw images with 2D object masks and camera poses, among which we evenly select 300 images/masks for training and 100 for testing. There are around 10 objects in each scene.

4.2 Baseline and Metrics

The most similar work to us at the time of our submission is [41]. However, it does not learn a unique object code for every 3D object, which is confirmed by communications with its authors. This means that it is not directly comparable with our method. In addition, there is no actual code released in its GitHub page and we cannot adapt its method as a baseline neither. Considering that our pipeline is trained and tested with 2D images and masks, we turn to use the classic Mask-RCNN [14] as a solid baseline. We hope that our OMG could serve as the first baseline for 3D scene decomposition and manipulation in the future.

Mask-RCNN: Since we train our OMG network in scene-specific fashion, we also fine-tune a COCO-pretrained R50-FPN Mask R-CNN model released by Detectron2 Library on every single scene as well. In particular, we carefully fine-tune the model using up to 480 epochs with learning rate $5e^{-4}$ and then pick up the best model on the testing split of every scene for comparison.

Metrics: We use the standard PSNR/SSIM/LPIPS scores to evaluate color image synthesis [21], and use AP (IoU=0.5) of all 2D testing images to evaluate 3D scene decomposition. Note that object classification is irrelevant in this paper.

4.3 3D Scene Decomposition

We evaluate the performance of scene decomposition on three datasets. Note that, for OMG-SG dataset, we evaluate our method and the baseline on images of Group 1 only, while the images of Groups 2/3/4/5 are used for object manipulation. For every single scene in the three datasets, we train a separate model for our method, and fine-tune a separate model for Mask-RCNN for fairness.

Tables 1/2/3 show the quantitative results on three datasets. It can be seen that our method, not surprisingly, achieves excellent results for novel view rendering thanks to the original NeRF backbone. Notably, our method obtains nearly perfect object segmentation results across multiple viewing points of complex 3D scenes in all three datasets, clearly outperforming the baseline. We also provide additional quantitative results of AP scores with IoU threshold at 0.9 in appendix to show the clear gaps between our method and Mask-RCNN. Figure 6 shows the qualitative results and we can see that our results have much sharper object boundaries thanks to the explicit 3D geometry applied in our object field.

	Novel View Synthesis			Decomposition	
Synthetic Rooms	PSNR↑	SSIM↑	LPIPS↓	MR[14] $AP^{0.5}$ ↑	**OMG** $AP^{0.5}$ ↑
Bathroom	44.05	0.994	0.009	97.90	100.0
Bedroom	48.07	0.996	0.009	98.91	100.0
Dinning	42.34	0.984	0.028	98.85	100.0
Kitchen	46.06	0.994	0.014	92.06	100.0
Reception	42.59	0.993	0.008	98.81	100.0
Rest	42.80	0.994	0.007	98.89	100.0
Study	41.08	0.987	0.026	98.93	99.76
Office	46.38	0.996	0.006	96.87	100.0
Average	44.17	0.992	0.013	97.65	**99.97**

Table 1. Quantitative results of our method on 8 scenes of OMG-SR dataset.

	Novel View Synthesis			Decomposition	
Reconstructed Rooms	PSNR↑	SSIM↑	LPIPS↓	MR[14] $AP^{0.5}$ ↑	**OMG** $AP^{0.5}$ ↑
Office_0	40.66	0.972	0.070	90.88	95.31
Office_2	36.98	0.964	0.115	90.07	95.17
Office_3	35.34	0.955	0.078	88.97	91.21
Office_4	32.95	0.921	0.172	86.84	85.39
Room_0	34.97	0.940	0.127	82.09	94.75
Room_1	34.72	0.931	0.134	87.04	97.79
Room_2	37.32	0.963	0.115	90.99	95.01
Average	36.13	0.949	0.116	88.12	**93.52**

Table 2. Quantitative results of our method on 7 scenes of Replica dataset.

	Novel View Synthesis			Decomposition	
Real-world Rooms	PSNR↑	SSIM↑	LPIPS↓	MR[14] $AP^{0.5}$ ↑	**OMG** $AP^{0.5}$ ↑
0010_00	26.82	0.809	0.381	92.80	97.92
0012_00	29.28	0.753	0.389	93.49	100.0
0024_00	23.68	0.705	0.452	87.18	93.26
0033_00	27.76	0.856	0.342	93.74	98.69
0038_00	29.36	0.716	0.415	97.01	98.44
0088_00	29.37	0.825	0.386	90.04	90.32
0113_00	31.19	0.878	0.320	98.59	99.67
0192_00	28.19	0.732	0.376	97.94	99.80
Average	28.21	0.784	0.383	93.85	**97.26**

Table 3. Quantitative results of our method on 8 scenes of ScanNet dataset.

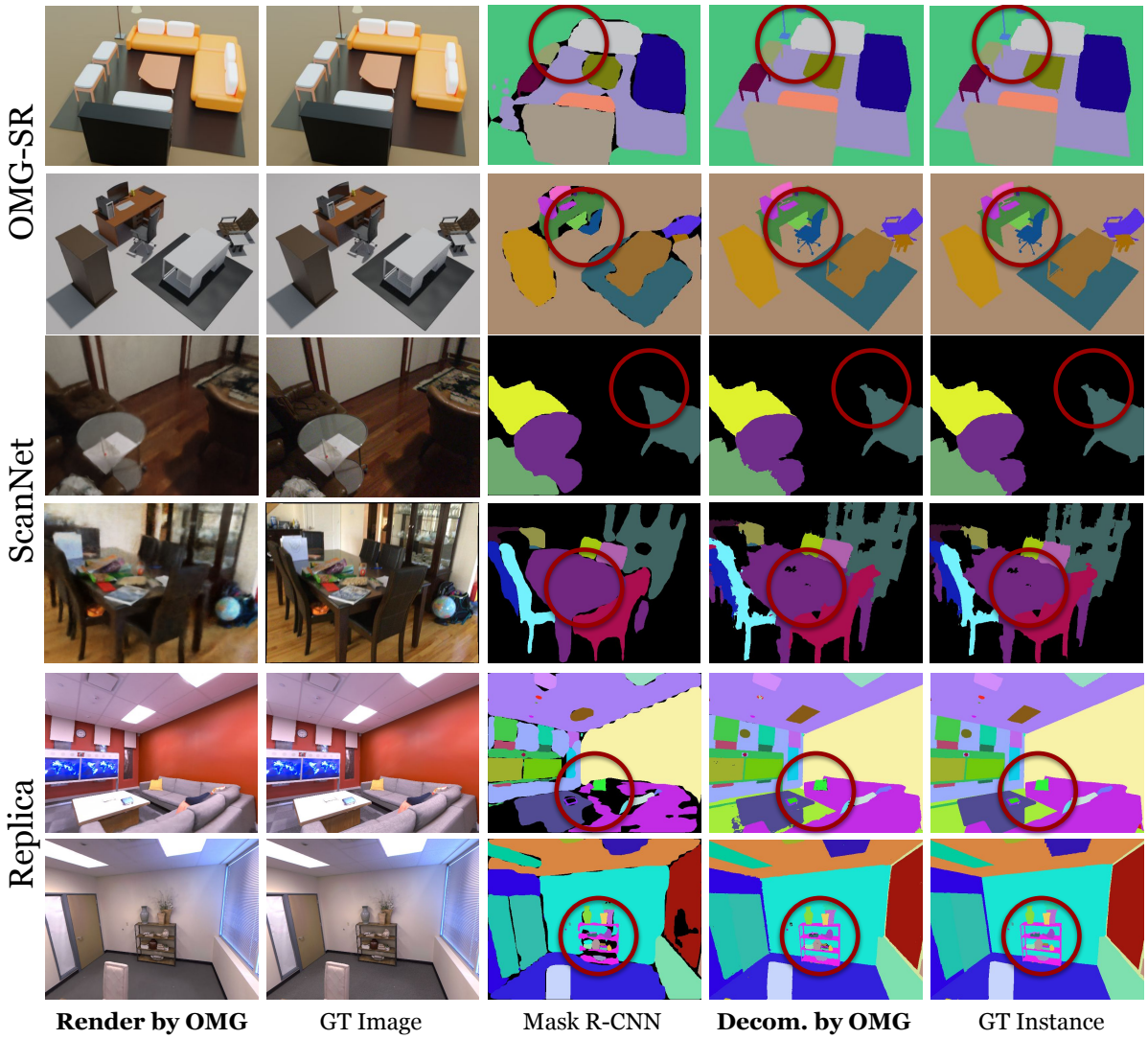


Fig. 6. Qualitative results of our method and baselines on three datasets: OMG-SR, Replica and ScanNet. The dark red circles highlight the differences.

4.4 3D Object Manipulation

In this section, we directly use our model trained on the images of Group 1 to test on the remaining images of Groups 2/3/4/5 in OMG-SR dataset. In particular, with the trained model, we feed the known manipulation information of Groups 2/3/4/5 into Algorithm 1, generating images and 2D object masks. These (edited) images and masks are compared with the ground truth 2D views.

Table 4 shows the quantitative results. More strict AP scores with IoU=0.9 are used to better show the difference of object segmentation. It can be seen that the quality of novel view rendering decreases after manipulation compared with non-manipulation in Table 1, primarily because the lighting factors are not decomposed and the illumination of edited objects shows discrepancies. However, the object decomposition is still nearly perfect, as also shown in Figure 4. More quantitative and qualitative results on other datasets are in appendix.

4.5 Ablation Study

To evaluate the effectiveness of our key designs of object field and manipulator, we conduct two ablation studies. 1) We only remove the loss functions ($\ell_{3d\_empty} + \ell_{3d\_obj}$) which are jointly designed to learn correct codes for empty 3D points, denoted as w/o ℓ_{3d}. 2) During manipulation, we remove the step "Tackle visual occlusions" in Algorithm 1, denoted as w/o VO. As shown in Table 4, both our loss functions for empty space regularization and visual occlusion handling step are crucial for accurate 3D scene decomposition and manipulation. More ablation results are in appendix.

	Translation				Rotation			
	PSNR↑	SSIM↑	LPIPS↓	$AP^{0.9}$ ↑	PSNR↑	SSIM↑	LPIPS↓	$AP^{0.9}$ ↑
w/o ℓ_{3d}	32.84	0.967	0.048	91.49	30.38	0.945	0.090	86.33
w/o VO	33.54	0.970	0.045	92.21	30.57	0.953	0.076	85.85
OMG (Full)	**33.94**	**0.975**	**0.033**	**93.10**	**31.94**	**0.969**	**0.038**	**90.36**
	Scale				Joint			
	PSNR↑	SSIM↑	LPIPS↓	$AP^{0.9}$ ↑	PSNR↑	SSIM↑	LPIPS↓	$AP^{0.9}$ ↑
w/o ℓ_{3d}	31.84	0.959	0.062	83.60	29.95	0.947	0.088	81.80
w/o VO	32.43	0.964	0.054	84.38	29.85	0.951	0.075	81.63
OMG (Full)	**33.40**	**0.971**	**0.037**	**90.39**	**30.65**	**0.965**	**0.045**	**87.11**

Table 4. Quantitative results of our object manipulation results on OMG-SG dataset.

5 Discussion and Conclusion

We have shown that it is feasible to simultaneously reconstruct, decompose, manipulate and render complex 3D scenes in a single pipeline only from 2D views. By adding an object field component into the MLP based implicit representation, we can successfully decompose all individual objects in 3D space. The decomposed object shapes can be further freely edited using our visual occlusion aware manipulator. The limitation of our work is the lack of decomposing lighting factors, which is left for our future work.

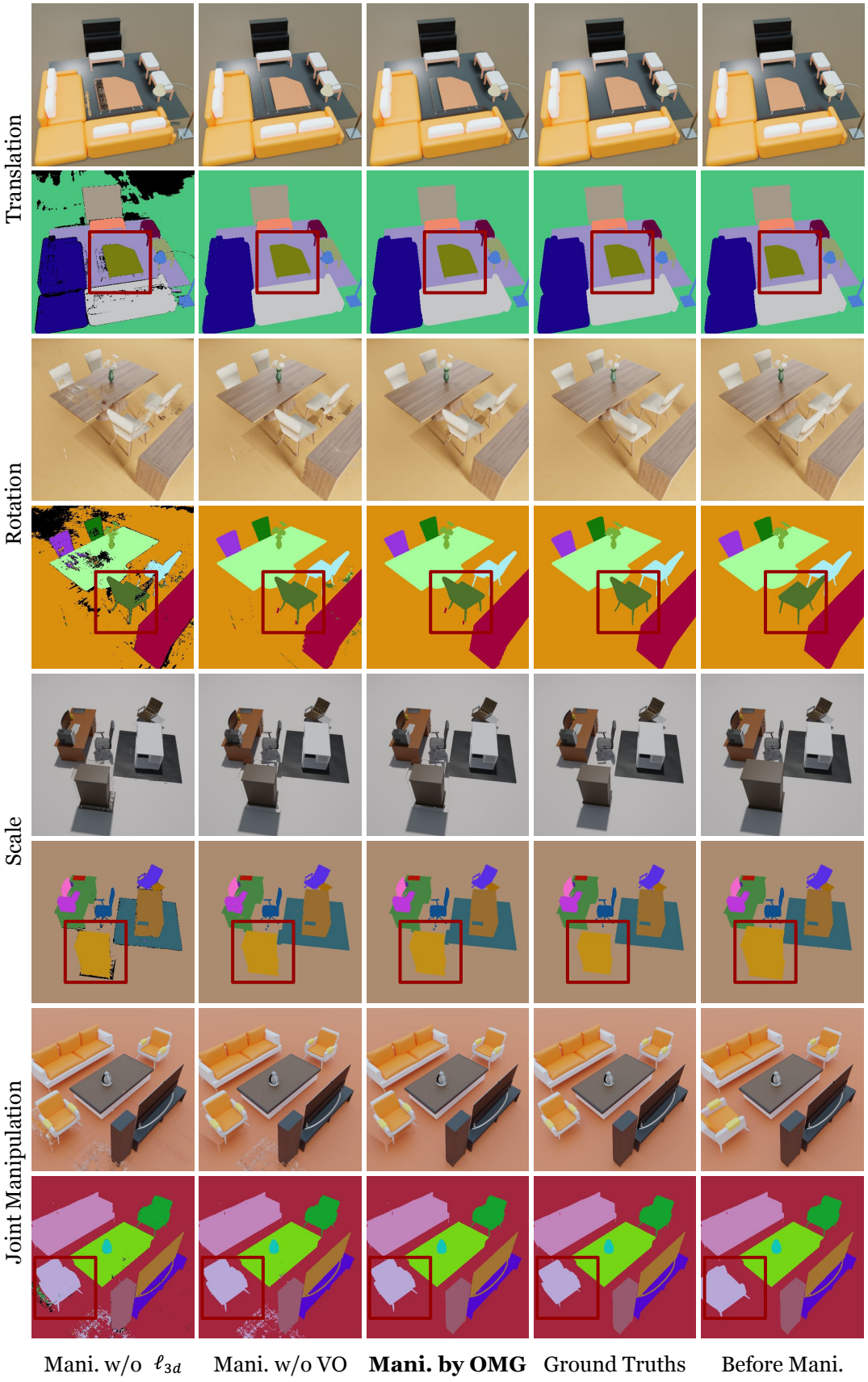


Fig. 7. Qualitative results of our method for object manipulation on OMG-SR dataset. The dark red boxes highlight the differences.

References

1. Alaluf, Y., Mokady, R., Bermano, A.H.: HyperStyle: StyleGAN Inversion with HyperNetworks for Real Image Editing. CVPR (2022)
2. Alliez, P., Meyer, M., Desbrun, M.: Interactive geometry remeshing. ACM TOG (2002)
3. Boss, M., Jampani, V., Braun, R., Liu, C., Barron, J.T., Hendrik: Neural-PIL: Neural Pre-Integrated Lighting for Reflectance Decomposition. NeurIPS (2021)
4. Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I.D., Leonard, J.J.: Past, Present, and Future of Simultaneous Localization and Mapping: Towards the Robust-Perception Age. IEEE Transactions on Robotics (2016)
5. Chen, W., Litalien, J., Gao, J., Wang, Z., Tsang, C.F., Khamis, S., Litany, O., Fidler, S.: DIB-R++: Learning to Predict Lighting and Material with a Hybrid Differentiable Renderer. NeurIPS (2021)
6. Chibane, J., Mir, A., Pons-Moll, G.: Neural Unsigned Distance Fields for Implicit Function Learning. NeurIPS (2020)
7. Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. ECCV (2016)
8. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. CVPR (2017)
9. Dhamo, H., Manhardt, F., Navab, N., Tombari, F.: Graph-to-3D: End-to-End Generation and Manipulation of 3D Scenes Using Scene Graphs. ICCV (2021)
10. Fan, H., Su, H., Guibas, L.: A Point Set Generation Network for 3D Object Reconstruction from a Single Image. CVPR (2017)
11. Gkioxari, G., Malik, J., Johnson, J.: Mesh R-CNN. ICCV (2019)
12. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: A Papier-Mache Approach to Learning 3D Surface Generation. CVPR (2018)
13. Guandao Yang, Belongie, S., Hariharan, B., Koltun, V.: Geometry Processing with Neural Fields. NeurIPS (2021)
14. He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask R-CNN. IEEE International Conference on Computer Vision pp. 2961–2969 (2017)
15. Jang, W., Agapito, L.: CodeNeRF: Disentangled Neural Radiance Fields for Object Categories. ICCV (2021)
16. Kato, H., Ushiku, Y., Harada, T.: Neural 3D Mesh Renderer. CVPR (2018)
17. Kuhn, H.W.: The Hungarian Method for the assignment problem. Naval Research Logistics Quarterly **2**(1-2), 83–97 (1955)
18. Lin, C.H., Kong, C., Lucey, S.: Learning Efficient Point Cloud Generation for Dense 3D Object Reconstruction. AAAI (2018)
19. Liu, S., Zhang, X., Zhang, Z., Zhang, R., Zhu, J.Y., Russell, B.: Editing Conditional Radiance Fields. ICCV (2021)
20. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy Networks: Learning 3D Reconstruction in Function Space. CVPR (2019)
21. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. ECCV (2020)
22. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3D Bounding Box Estimation Using Deep Learning and Geometry. CVPR (2017)
23. Munkberg, J., Hasselgren, J., Shen, T., Gao, J., Chen, W., Evans, A., Müller, T., Fidler, S.: Extracting Triangular 3D Models, Materials, and Lighting From Images. arXiv:2111.12503 (2021)

24. Niemeyer, M., Geiger, A.: GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields. CVPR pp. 11453–11464 (2021)
25. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision. CVPR (2020)
26. Ost, J., Mannan, F., Thuerey, N., Knodt, J., Heide, F.: Neural Scene Graphs for Dynamic Scenes. CVPR (2021)
27. Ozyesil, O., Voroninski, V., Basri, R., Singer, A.: A Survey of Structure from Motion. Acta Numerica **26**, 305–364 (2017)
28. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. CVPR (2019)
29. Roberts, M., Paczan, N.: Hypersim: A Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding. ICCV (2021)
30. Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., Parikh, D., Batra, D.: Habitat: A Platform for Embodied AI Research. ICCV (2019)
31. Sengupta, S., Gu, J., Kim, K., Liu, G., Jacobs, D.W., Kautz, J.: Neural Inverse Rendering of an Indoor Scene from a Single Image. ICCV (2019)
32. Song, S., Yu, F., Zeng, A., Chang, A.X., Savva, M., Funkhouser, T.: Semantic Scene Completion from a Single Depth Image. CVPR (2017)
33. Stelzner, K., Kersting, K., Kosiorek, A.R.: Decomposing 3D Scenes into Objects via Unsupervised Volume Segmentation. arXiv:2104.01148 (2021)
34. Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J.J., Mur-artal, R., Ren, C., Verma, S., Clarkson, A., Yan, M., Budge, B., Yan, Y., Pan, X., Yon, J., Zou, Y., Leon, K., Carter, N., Briales, J., Gillingham, T., Mueggler, E., Pesqueira, L., Savva, M., Batra, D., Strasdat, H.M.: The Replica Dataset : A Digital Replica of Indoor Spaces. arXiv:1906.05797 (2019)
35. Tatarchenko, M., Dosovitskiy, A., Brox, T.: Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs. ICCV (2017)
36. Tewari, A., Elgharib, M., Bharaj, G., Bernard, F., Seidel, H.P., Perez, P., Zollhofer, M., Theobalt, C.: StyleRig: Rigging StyleGAN for 3D Control over Portrait Images. CVPR (2020)
37. Trevithick, A., Yang, B.: GRF: Learning a General Radiance Field for 3D Representation and Rendering. ICCV (2021)
38. Tulsiani, S., Gupta, S., Fouhey, D., Efros, A.A., Malik, J.: Factoring Shape, Pose, and Layout from the 2D Image of a 3D Scene. CVPR (2018)
39. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. NeurIPS (2021)
40. Xie, H., Yao, H., Sun, X., Zhou, S., Zhang, S., Tong, X.: Pix2Vox: Context-aware 3D Reconstruction from Single and Multi-view Images. ICCV (2019)
41. Yang, B., Zhang, Y., Xu, Y., Li, Y., Zhou, H., Bao, H., Zhang, G., Cui, Z.: Learning Object-Compositional Neural Radiance Field for Editable Scene Rendering. ICCV (2021)
42. Yang, B., Rosa, S., Markham, A., Trigoni, N., Wen, H.: Dense 3D Object Reconstruction from a Single Depth View. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019)
43. Yang, B., Wang, J., Clark, R., Hu, Q., Wang, S., Markham, A., Trigoni, N.: Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds. NeurIPS (2019)

44. Zhang, C., Cui, Z., Zhang, Y., Zeng, B., Pollefeys, M., Liu, S.: Holistic 3D Scene Understanding from a Single Image with Implicit Representation. CVPR (2021)
45. Zhang, J., Liu, X., Ye, X., Zhao, F., Zhang, Y., Wu, M., Zhang, Y., Xu, L., Yu, J.: Editable free-viewpoint video using a layered neural representation. ACM Transactions on Graphics **40**(4) (2021)
46. Zhang, X., Srinivasan, P.P., Deng, B., Debevec, P., Freeman, W.T., Barron, J.T.: NeRFactor: Neural Factorization of Shape and Reflectance Under an Unknown Illumination. SIGGRAPH Asia (2021)
47. Zhi, S., Laidlow, T., Leutenegger, S., Davison, A.J.: In-Place Scene Labelling and Understanding with Implicit Scene Representation. ICCV (2021)
48. Zhou, Y., Tuzel, O.: VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. CVPR (2018)
49. Zou, C., Yumer, E., Yang, J., Ceylan, D., Hoiem, D.: 3D-PRNN: Generating Shape Primitives with Recurrent Neural Networks. ICCV (2017)