A Additional Implementation Details

Network Architecture The detailed architecture of our simple pipeline is shown in Figure 7.

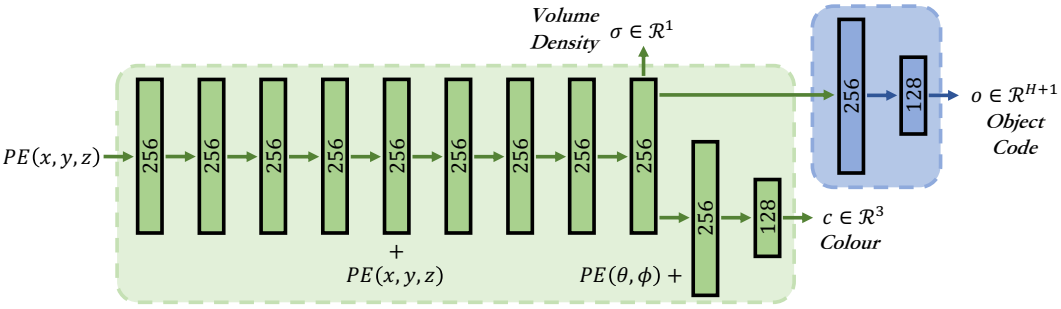

Figure 7: DM-NeRF architecture. The positional encoding $PE(\cdot)$ of the location (x, y, z) and viewing direction (θ, ϕ) are taken as the inputs of our network. The volume density σ and object code $\boldsymbol{o}$ are the functions of the location while the colour additionally depends on the viewing 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:

$$\boldsymbol{C}_{h,t}^{sIoU} = \frac{\sum_{n=1}^{N}(M_h^n * \bar{M}_t^n)}{\sum_{n=1}^{N} M_h^n + \sum_{n=1}^{N} \bar{M}_t^n - \sum_{n=1}^{N}(M_h^n * \bar{M}_t^n)} \tag{11}$$

where M_h^n and $\bar{M}_t^n$ are the n^{th} values of M_h 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:

$$\boldsymbol{C}_{h,t}^{CES} = -\frac{1}{N}\sum_{n=1}^{N}\left[\bar{M}_t^n \log M_h^n + (1 - \bar{M}_t^n)\log(1 - M_h^n)\right] \tag{12}$$

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.

$$sIoU_l = \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{C}_{t,t}^{sIoU}) \quad CES_l = \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{C}_{t,t}^{CES}) \tag{13}$$

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.

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.

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 16), 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.

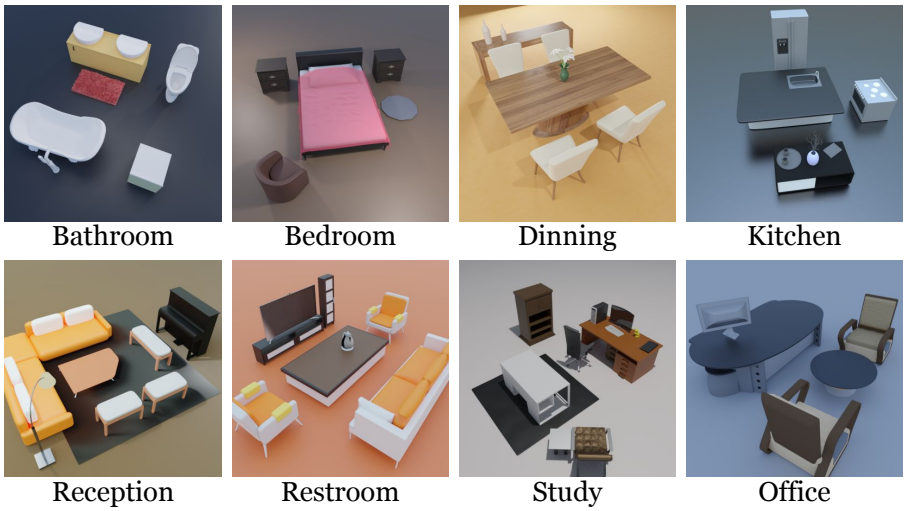

Figure 8: Eight different indoor scenes of our DM-SR dataset.

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

	$AP^{0.50}$ ↑				$AP^{0.75}$ ↑			
	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 4: 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 7/8/9. 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), denoted as w/0.025, w/0.05, w/0.10.

From Tables 5/6/7, 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	$AP^{0.75}$ ↑				$AP^{0.90}$ ↑			
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 5: 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	$AP^{0.75}$ ↑				$AP^{0.90}$ ↑			
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 6: 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 7: 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 $Detach$ (w/o): The gradients from object field component can backpropagate to the backbone of NeRF. In this case, the object field and the rendering parts will influence each other.
- With $Detach$ (w/): The object field component only depends on the output representation of NeRF and will not affect the rendering part at all.

From Tables 8/9/10, 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.

Detach	PSNR↑		LPIPS↑		SSIM↓		$AP^{0.75}$ ↑	
	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 8: Quantitative results of our method on DM-SR dataset.

Detach	PSNR↑		LPIPS↑		SSIM↓		$AP^{0.75}$ ↑	
	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 9: Quantitative results of our method on Replica dataset.

Detach	PSNR↑		LPIPS↑		SSIM↓		$AP^{0.75}$ ↑	
	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 10: 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 11, 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 11: 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.

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.

D ADDITIONAL EXPERIMENTS FOR NOISY 2D LABELS

As illustrated in Figure 15, to further evaluate the robustness of our method, we use the instance masks estimated by Mask-RCNN as the supervision signals when training our DM-NeRF. Table 12 shows the quantitative results on our DM-SR dataset. We can see that even though the 2D labels are inaccurate for training, our method still achieves excellent object decomposition results.

	Bathroom	Bedroom	Dinning	Kitchen	Reception	Rest	Study	Office	Average
DM-NeRF	95.30	97.08	95.69	94.72	99.62	98.58	97.72	98.08	97.10

Table 12: Quantitative results of scene decomposition from our method trained with noisy 2D labels (estimated by Mask-RCNN) on DM-SR dataset. AP scores with IoU thresholds at 0.75 are reported.

E EXTENSION TO PANOPTIC SEGMENTATION

An extra semantic branch parallel to object code branch is added into our current DM-NeRF for panoptic segmentation. Table 13 shows the quantitative results on our DM-SR dataset where the accurate 2D semantic and instance labels are used to train our network. Figure 9 shows the qualitative results. It can be seen that both semantic categories and object codes are accurately inferred.

	Bathroom	Bedroom	Dinning	Kitchen	Reception	Rest	Study	Office	Average
DM-NeRF (Obj: $AP^{0.75}$)	100.0	100.0	99.41	100.0	100.0	97.86	96.84	100.0	99.26
DM-NeRF (Sem: mIoU)	97.58	99.08	94.64	98.72	97.42	97.13	94.29	97.85	97.09

Table 13: Quantitative results of panoptic segmentation from our method trained with accurate 2D semantic and instance labels on DM-SR dataset. The AP score of all objects and the mIoU score of all categories in each scene are reported respectively.

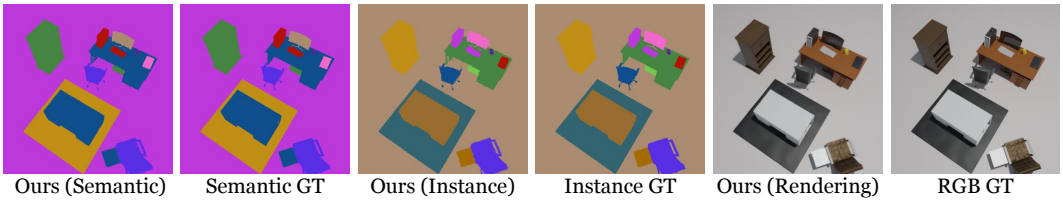

Figure 9: Qualitative results of our method for panoptic segmentation on DM-SR dataset.

F ADDITIONAL QUALITATIVE RESULTS

Scene Decomposition and Manipulation Figures 10/11/12/13 show qualitative results of 3D scene decomposition and manipulation in Sections 4.3 & 4.4.

Scene Decomposition Figures 14/15 shows qualitative results of scene decomposition from our method trained on noisy and inaccurate 2D labels.

Scene Rendering and Decomposition Figures 16/17/18 show additional qualitative results of scene rendering and decomposition.

Scene Manipulation Figures 19/20/21/12 shows additional qualitative results of scene manipulation with single/multiple objects under various transformations.

More qualitative results can be found in the video available at `https://github.com/vLAR-group/DM-NeRF`.

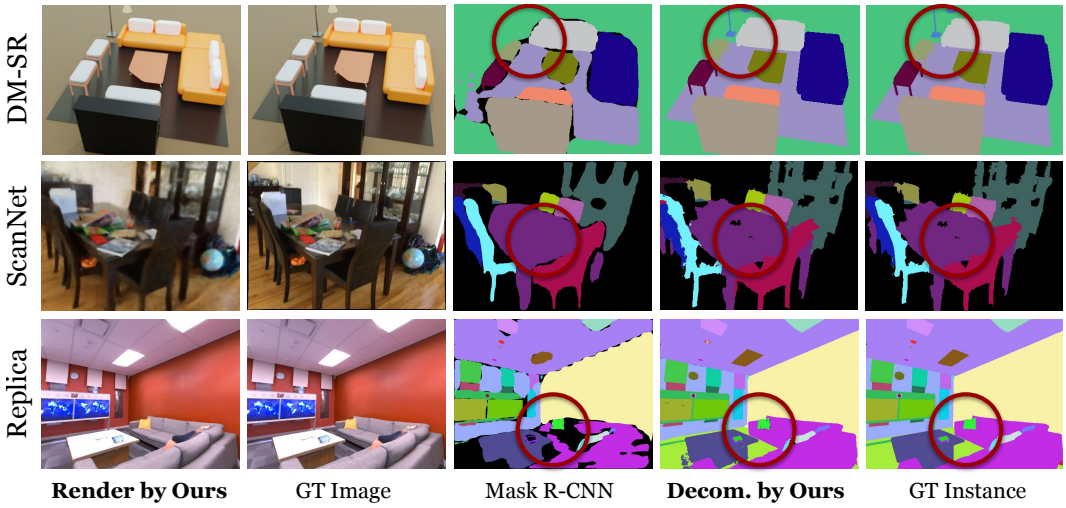

Figure 10: Qualitative results of our method and the baseline on three datasets: DM-SR, Replica and ScanNet. The dark red circles highlight the differences.

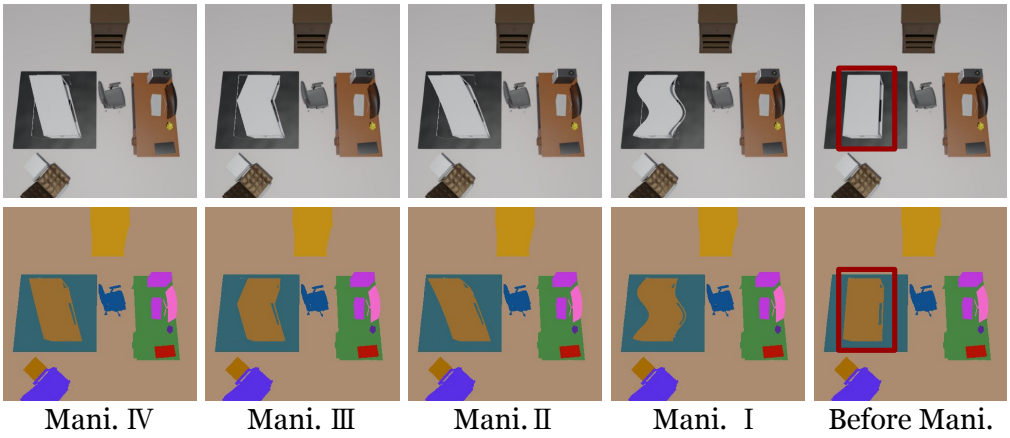

Figure 11: Qualitative results of our method for object deformation manipulation on DM-SR dataset. The dark red boxes highlight the target table to be manipulated.

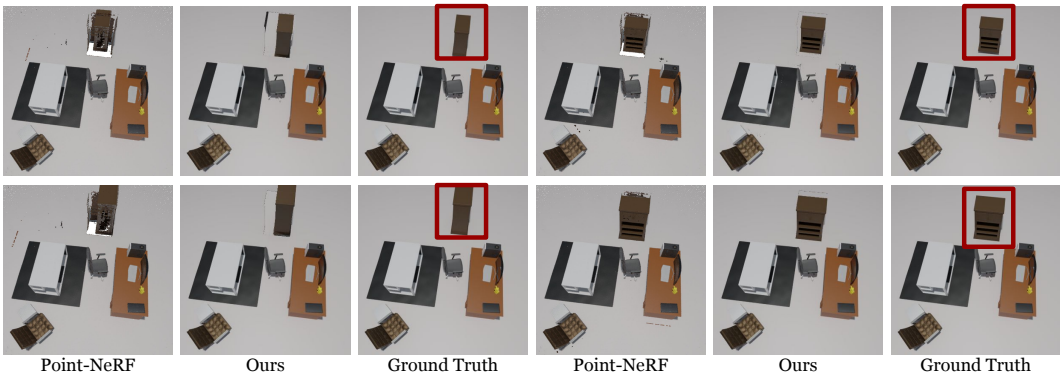

Figure 12: Qualitative results of novel view rendering after manipulating 3D objects. It can be seen that our method obtains clearly sharper object shapes after manipulation in 3D space, whereas the baseline Point-NeRF shows obvious artifacts such as holes, primarily because its manipulation is conducted 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.

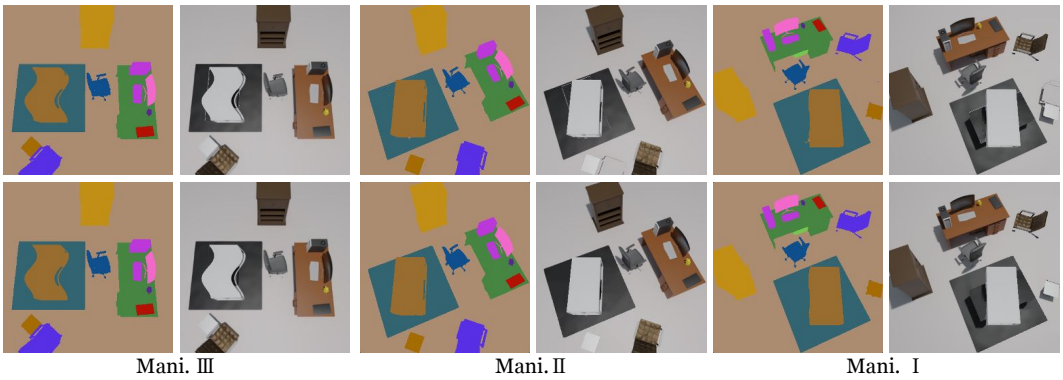

Figure 13: The artifacts can be further mitigated. By introducing a voting strategy, we improve the quality of scene decomposition for the following fine manipulation. To be specific, we also consider 8 neighbouring rays when determining the projected object code $\hat{o}$ along a light ray. If $\hat{o}$ 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.

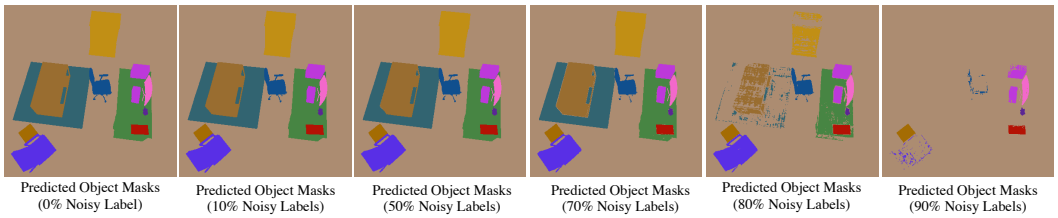

Figure 14: Estimated object masks at novel views from our models trained with different amount of noisy labels. Our method can infer satisfactory results, even though 80% of 2D labels are incorrect during training, demonstrating the robustness of our method.

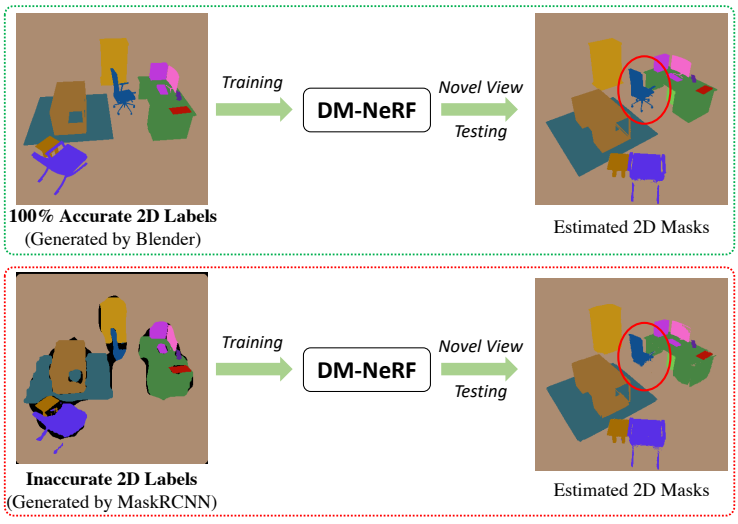

Figure 15: As shown in the red dotted block, we use the instance masks estimated by Mask-RCNN as the supervision signals when training our DM-NeRF. We can see that even though the 2D labels are inaccurate for training, our method still achieves excellent object decomposition results. The red circles show that only thin chair feet are not segmented using the inaccurate 2D labels in training. This result is consistent with that of Figure 14, showing the remarkable robustness of our method.

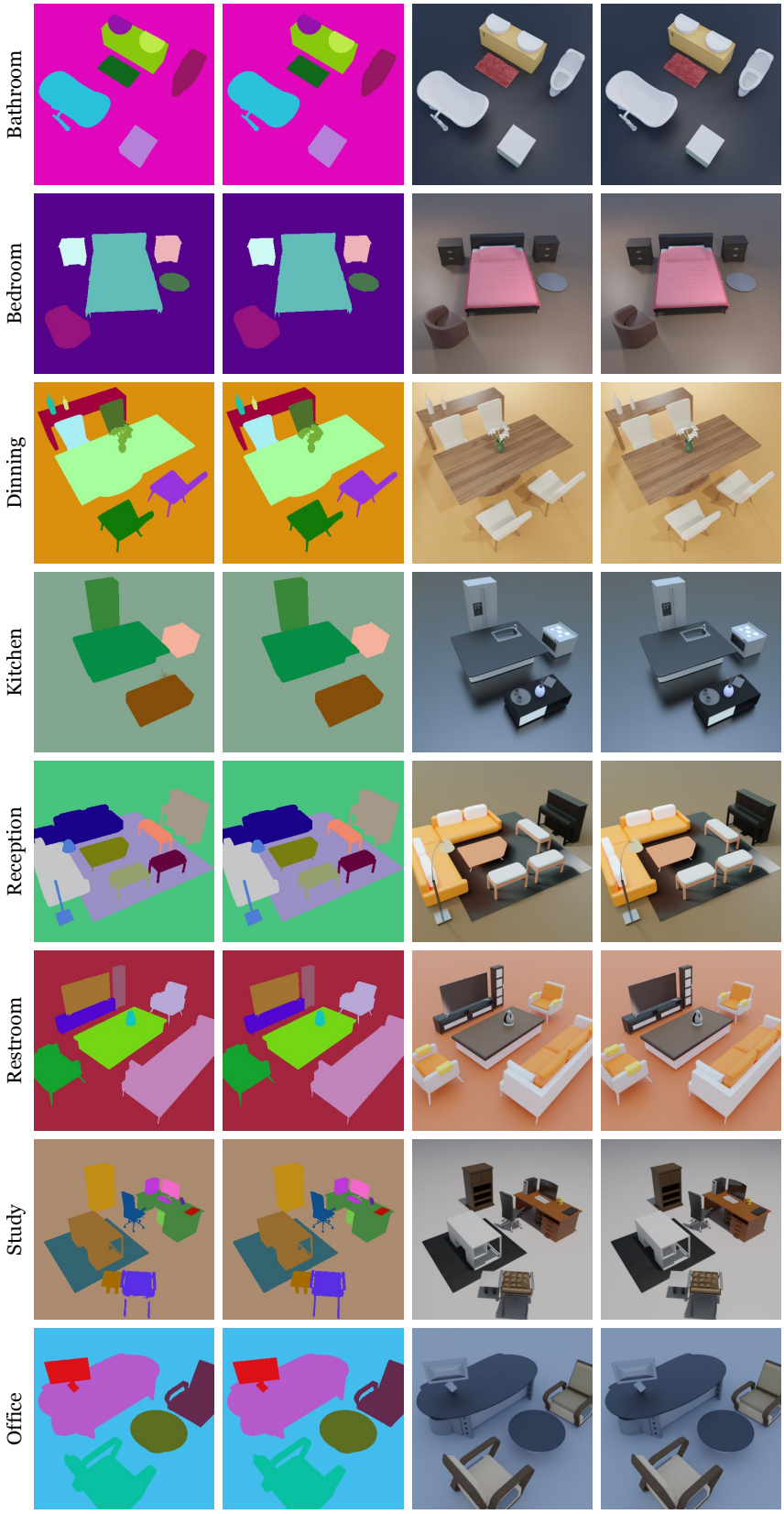

Figure 16: Qualitative results for scene rendering and decomposition on DM-SR.

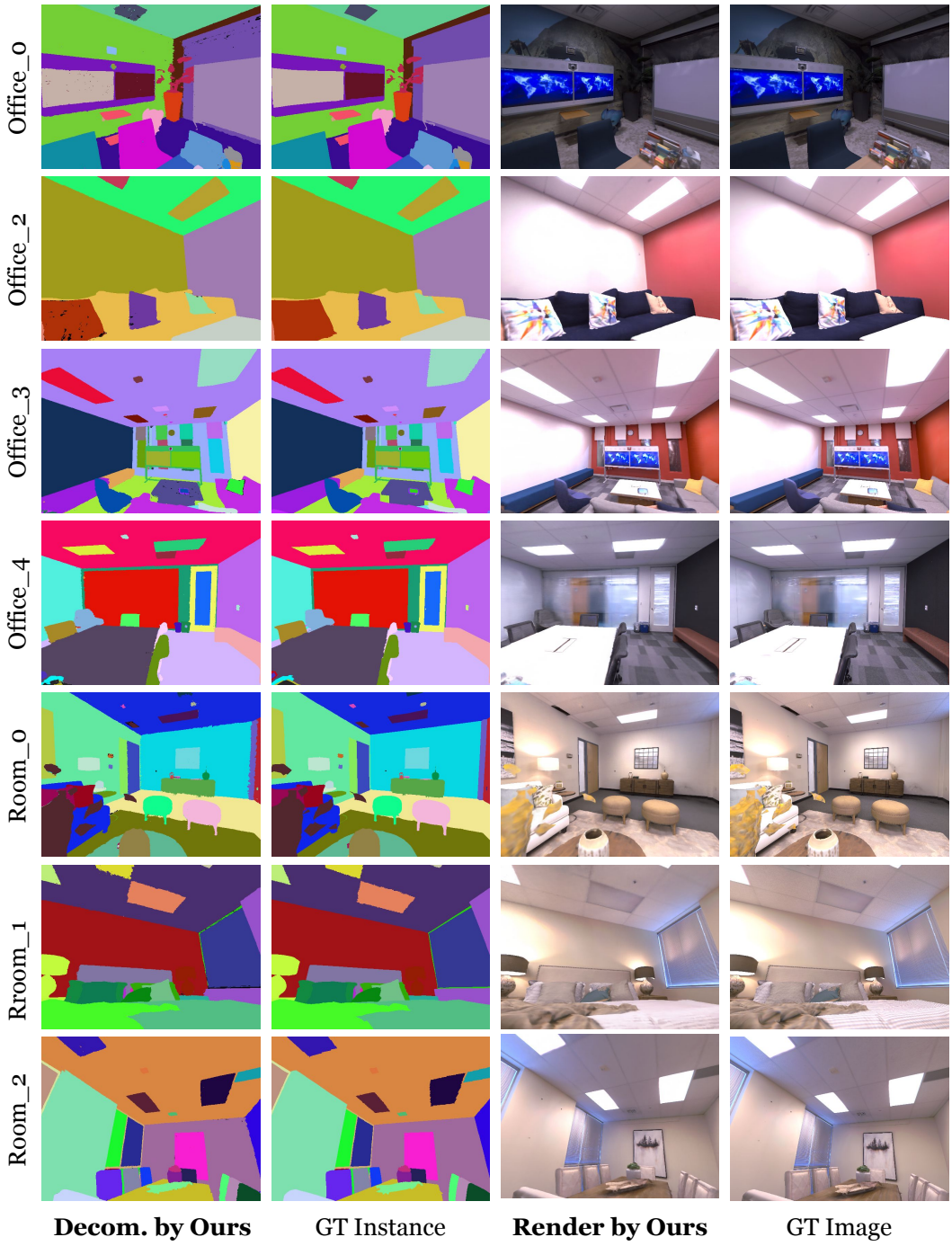

Figure 17: Qualitative results for scene rendering and decomposition on Replica.

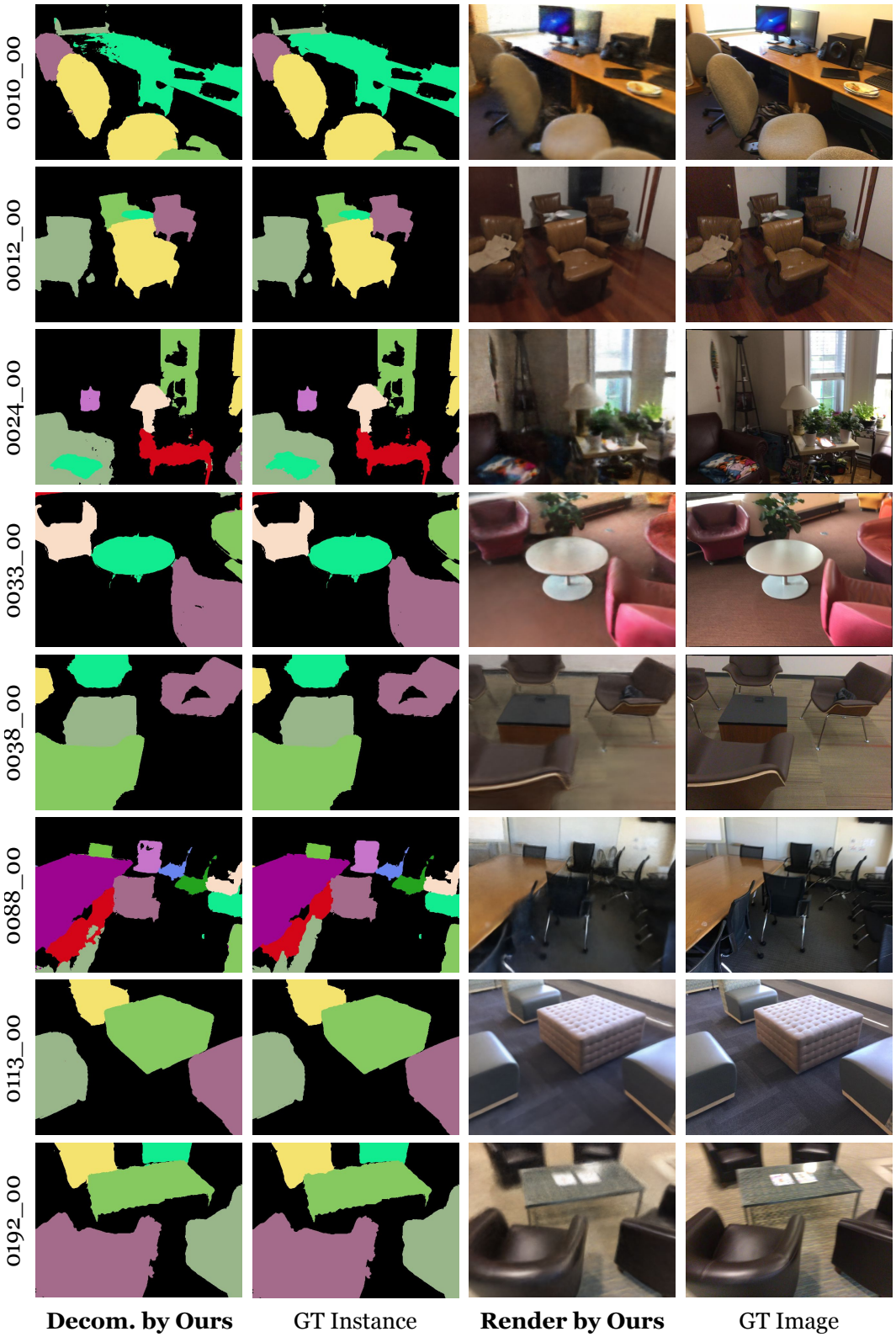

Figure 18: Qualitative results for scene rendering and decomposition on ScanNet.

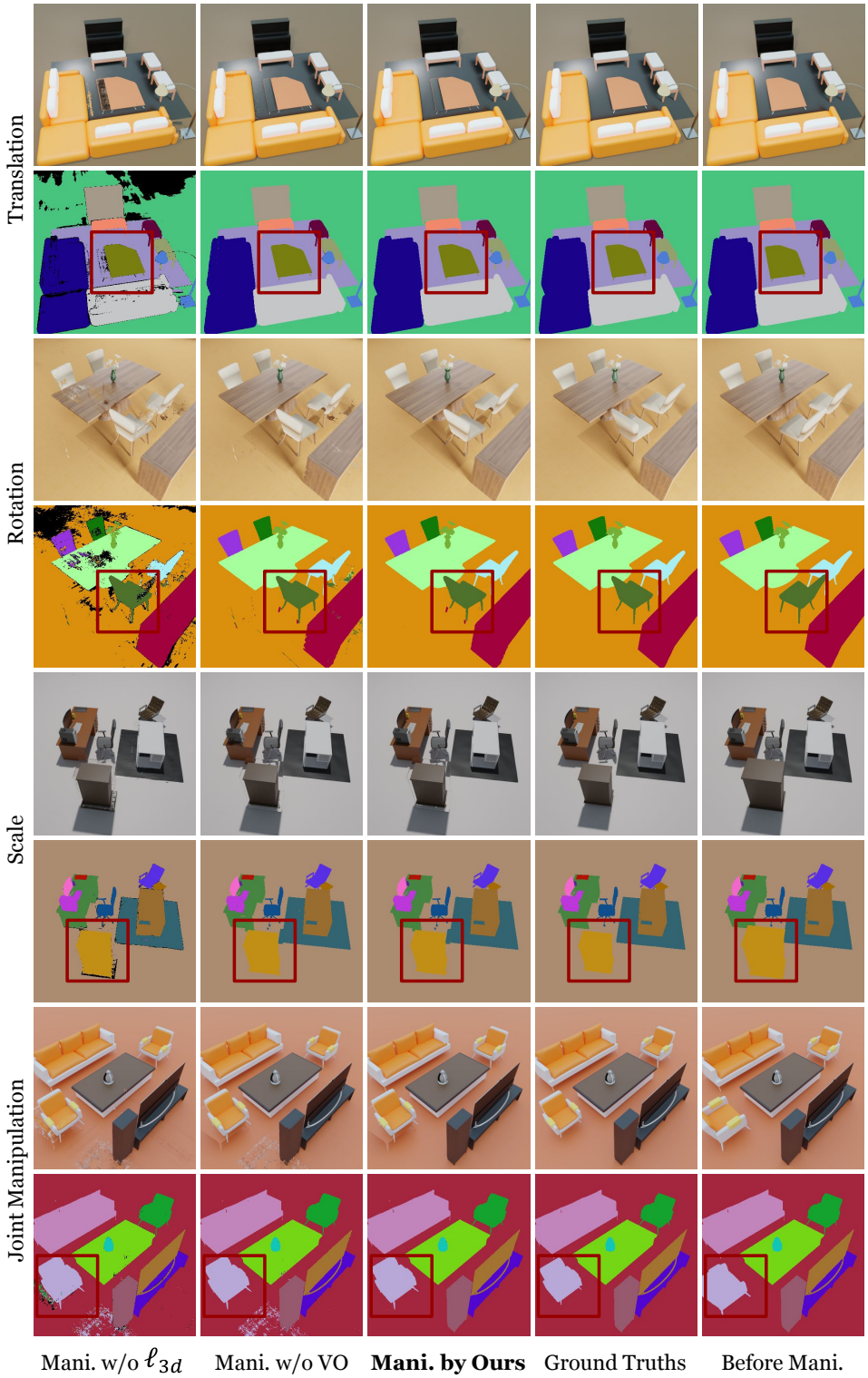

Figure 19: Qualitative results of our method for object manipulation on DM-SR dataset. The dark red boxes highlight the differences.

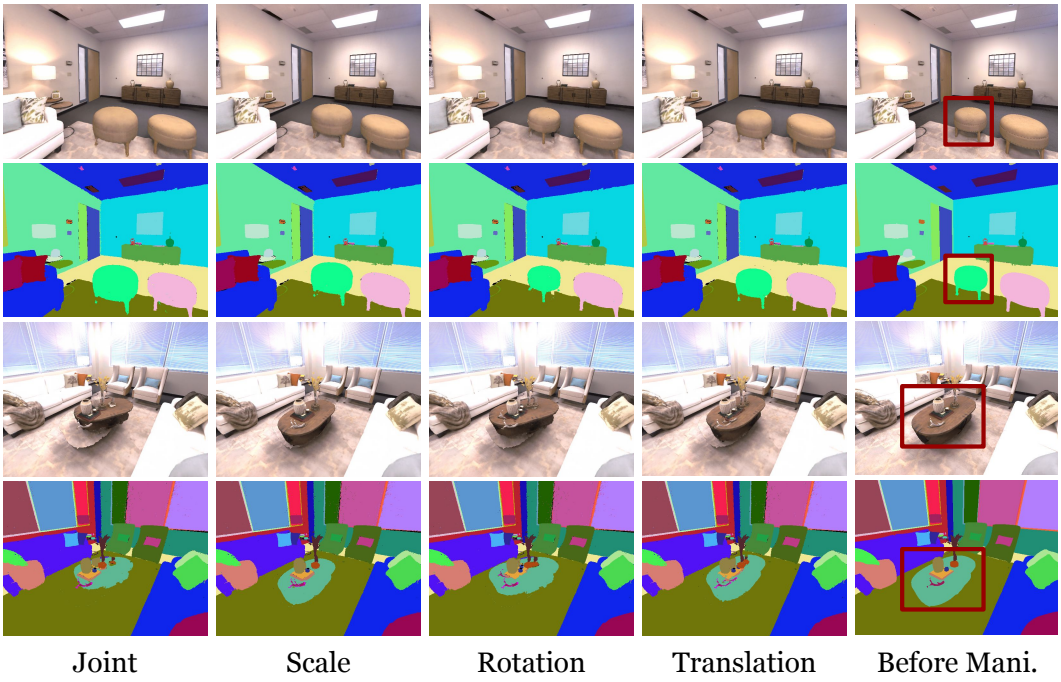

Figure 20: Qualitative results for single object manipulation on Replica dataset. The dark red box in the rightmost column highlights the manipulated object.

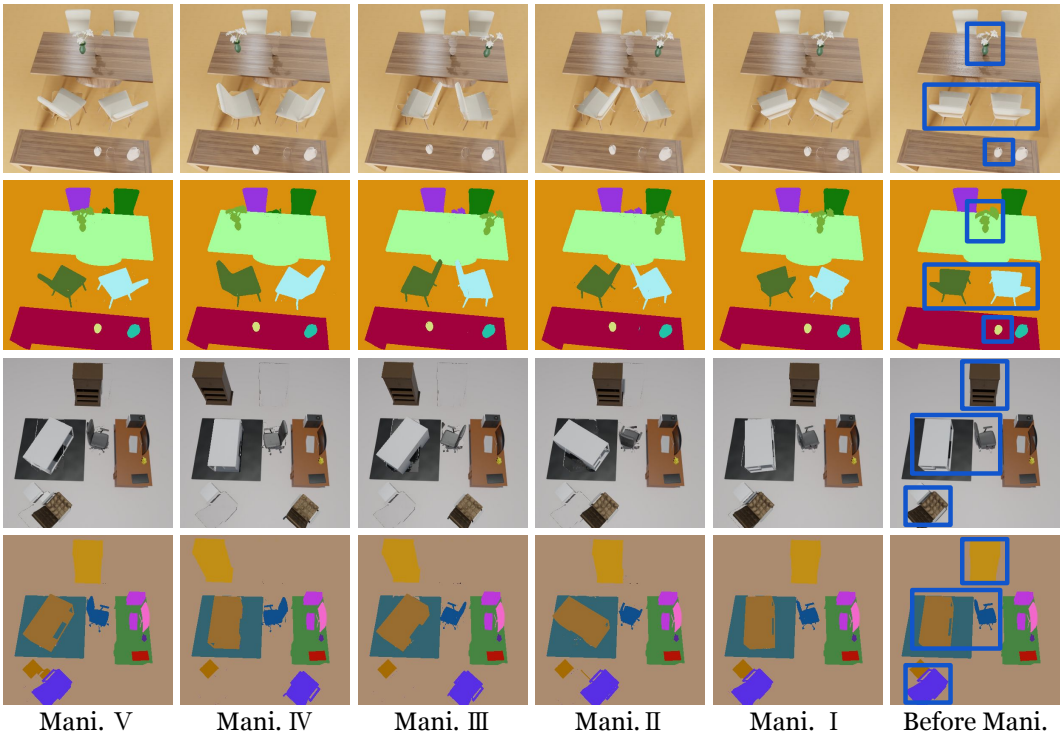

Figure 21: Qualitative results for multiple objects manipulation on DM-SR dataset. The dark blue boxes in the rightmost column highlight the manipulated objects.