Supplementary Material

Anonymous Author(s)
Affiliation
Address
`email`

`Video Results`

1 Introduction

The supplementary material comprehensively explains our methods and presents detailed experiment results. In the methods section (Sec 2), we delve into various components of the multiview segmentation process. We discuss the COLMAP [8] data structure for finding corresponding points in 3D space from 2D pixel coordinates (Sec 2.1), the rationale behind calculating nearest points after projecting 3D points onto 2D planes (Sec 2.2), the importance of maintaining a same number of points annotations across views (Sec 2.3), and the speed loss associated with Grounded-SAM [2] (Sec 2.4), which motivates our design of converting text prompts to points prompts. Additionally, we describe the points sampling strategy for text prompts (Sec 2.5) and the patch sampling strategy for perceptual loss in the scene object removal part (Sec 2.6). The Experiments section (Sec 3) presents detailed experiment results, including an overview of all scenes and their respective editing outcomes (Sec 3.3). We also discuss the mask refinement strategy for LaMa processing (Sec 3.1) and provide implementation details (Sec 3.2).

2 Methods

2.1 2D to 3D Matchness (COLMAP data structure)

This section explains how to find the corresponding 3D position of 2D pixels through COLMAP's [8] sparse reconstruction. Specifically, given a set of 2D pixels $\mathcal{P}_{2D}$ picked by users indicating the unwanted objects on only one image, our ultimate goal is to spread $\mathcal{P}_{2D}$ to all views while ensuring that the generated point annotations always indicate the target precisely. We address the above problem by projecting the target objects' sparse point cloud in 3D space to all 2D image planes. To find the sparse point cloud indicated by $\mathcal{P}_{2D}$, we first utilize SAM (Segment-Anything [4]) to obtain an initial mask from points prompt $\mathcal{P}_{2D}$, then query COLMAP [8] sparse reconstruction with 2D pixel coordinates in the initial mask to get their corresponding 3D coordinates.

Specifically, we can obtain three files *cameras.bin*, *images.bin*, and *points3D.bin* from COLMAP [8] sparse reconstruction [1]. The *images.bin* and *points3D.bin* files provide a one-to-one mapping between the reconstructed sparse point cloud and 2D pixel coordinates as follows:

```
# images.bin: Images indexed by all images with item
Image = {POINTS2D[] as (X, Y, POINT3D_ID)}
# points3D.bin: Points3D indexed by all 3D points with item
Point3D = {TRACK[] as (IMAGE_ID, POINT2D_IDX)}
# Thus, we can find Point3D by:
Point3D=Points3D[POINT3D_ID]=Points3D[Images[IMAGE_ID][POINTS2D]]
```

[1] COLMAP output formats

2.2 Calculation of Nearest Points

COLMAP [8] only provides sparse reconstruction, making it impossible to establish a connection from 2D to 3D for arbitrary image pixels. To tackle this limitation, we propose using the nearest points to the input points belonging to COLMAP's reconstruction. This approach offers two potential solutions: 1) Querying the 3D coordinates for all points in the initial mask region and sorting them based on the distance between their corresponding 2D pixel positions and the input points prompt. Subsequently, the 3D points are projected back to the 2D image planes, and the projected 2D pixels are selected based on the sorting results. 2) Alternatively, the points in the initial mask region are sorted, and a certain number of 2D points are selected first. Then, their 3D points are queried and projected back to the 2D planes.

We adopt the first strategy as the second one carries potential risks. If the camera angle changes significantly, there may be no corresponding pixel to the selected points in the initial mask. As a result, these points would lose their annotations for such a view, as they would be "out of the image." In contrast, the first strategy selects the nearest points after projecting the points back to the 2D planes from 3D space. This approach avoids the aforementioned problem by simply filtering out invalid 2D pixel coordinates after the projection from 3D. An example of this corner case is illustrated in Fig 1.

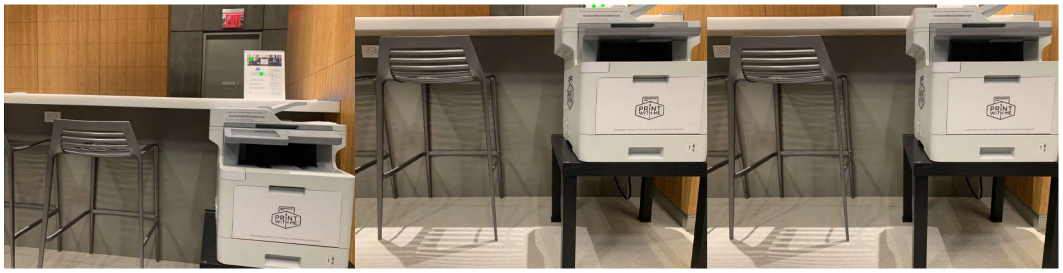


Figure 1: Comparison of two nearest points strategies. The left image displays the user prompt, while the first strategy (middle, project first) ensures annotations for extreme view angles. However, the second strategy (right, sort first) loses all annotations.

2.3 Number of Points Annotation for Each View

As discussed, we project the sparse point cloud of the target objects onto the 2D image plane to obtain accurate point annotations for all scene views. However, why do we only select a certain number of projected 2D points as input for SAM [4] rather than using all projected 2D points? This decision is because using all projected 2D points as input may result in an inaccurate mask, as many points may be interpreted as multiple instances for segmentation. Fig 2 illustrates an example of using all projected 2D points as SAM input. To avoid mask ambiguity, annotating several points that indicate the unwanted objects is sufficient, and there is no need to generate many points.

2.4 Grounded-SAM speed loss

This section presents evidence for why we chose to convert the text prompt to points prompt. As shown in Figure 3, directly using the same text input to predict masks for all views using Grounded-SAM [2] runs at a speed of approximately seven seconds per frame, while using points prompt accelerates the process to two frames per second. This speed improvement is because points prompt only requires projection calculations (pure matrix) and SAM prediction, whereas Grounded-SAM [2] also requires Grounding-DINO [5] prediction. Additionally, the detection ability of Grounding-DINO [5] is unreliable for all views. Therefore, we recommend switching from text input to points prompt to obtain higher-quality masks in less time.

2.5 Points Sampling Strategy for Text Prompt

Recall that we introduced our points sampling strategy for the text prompt, which involves selecting three points from the initial mask: top left, bottom right, and centre. We slightly revised the code by switching the corner points from top left and bottom right to top right and bottom left. However, this

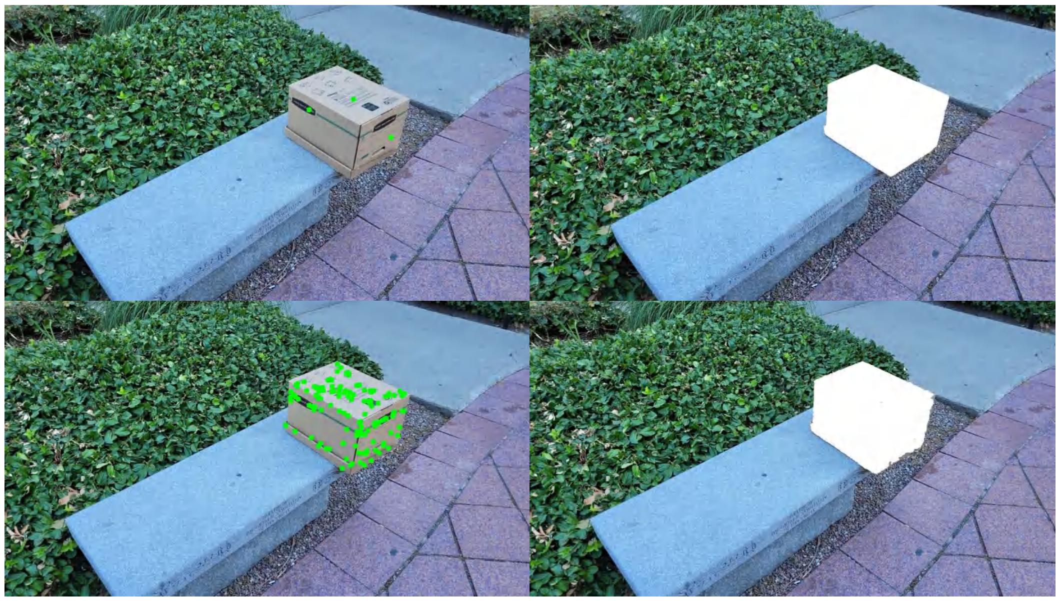
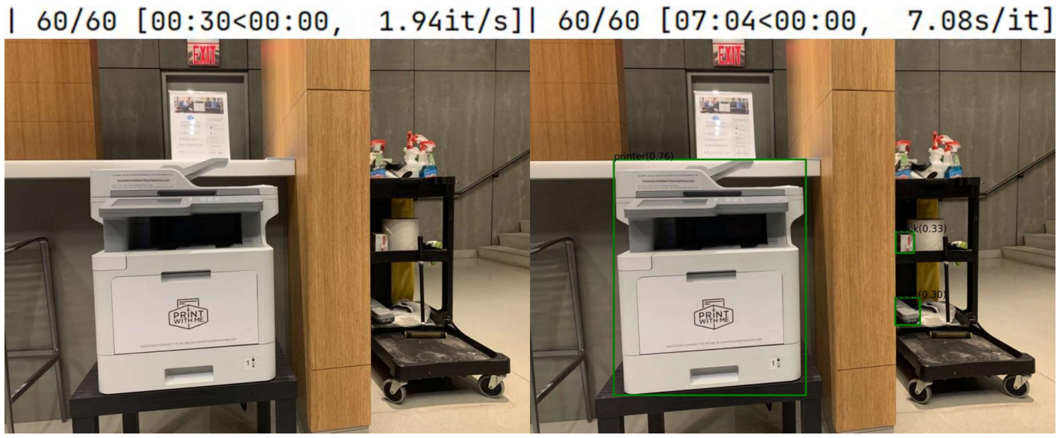


Figure 2: Comparison of two strategies for selecting the number of points. The second row (using all projected 2D points as input for SAM [4]) shows a serrated edge, while the first row (keeping the number of points the same as the user input) exhibits a smooth mask edge.



```
| 60/60 [00:30<00:00,  1.94it/s]| 60/60 [07:04<00:00,  7.08s/it]
```

Figure 3: Comparison of two strategies for predicting masks with a text prompt. The progress bars above indicate that the switch strategy (left) is significantly faster than the non-switch strategy (right). The images below illustrate an example of the detection failure (bottom right) of the book by Grounding-DINO [5].

approach is unsuitable for complex scenes where unwanted objects result in slim, irregular masks. In such cases, the centre point calculation may be inaccurate, as shown in Fig 4. We can omit the centre point and use two corner points only to tackle this problem. Moreover, using the corner points directly can introduce problems, as points on the boundary are ambiguous for SAM [4] when predicting masks. Similarly, we use the centre point only to handle this issue.

2.6 Patch Sampling Strategy for Perceptual Loss

As perceptual loss [3] operates at the image level, we make it compatible with Neural Radiance Fields [6] training by sampling patches from the image. The patch sampling strategy used in SPIn-NeRF [7] involves downsampling the image by a factor of 16 to determine the patch shape. However, this approach can fail to adequately handle small mask regions, especially when the mask shape is smaller than the downsampled shape, as illustrated in an extreme example in Fig 1. To address this issue, we

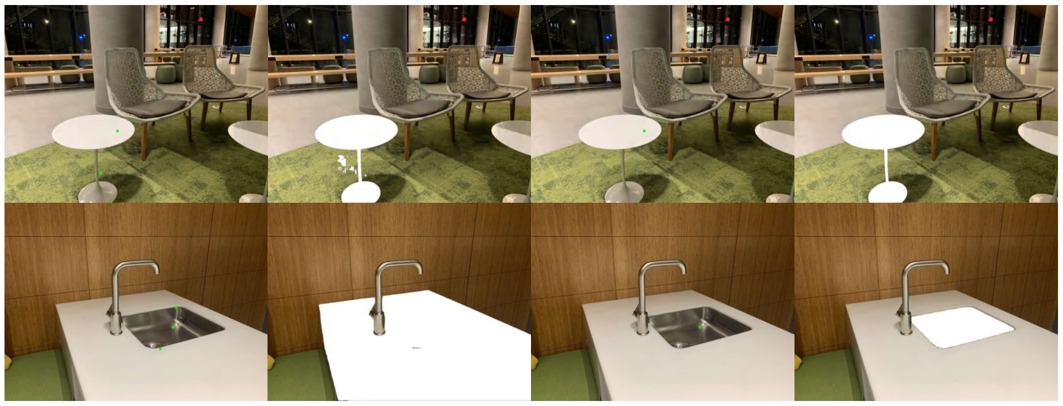


Figure 4: Comparison of two corner cases discussed in Sec 2.5. Each image pair, from left to right, displays points annotation and predicted masks. The left two columns depict the coarse sampling strategy, while the right two columns represent the improved version. The first row provides an example of an inaccurate centre point, and the second row demonstrates the ambiguous results when boundary points are applied.

propose a new sampling strategy that downsamples the mask area by a factor of 4 to determine the patch shape rather than relying on the original image shape. Additionally, we enforce a minimum patch length of 32 and a maximum length of 64. If these constraints still prove inadequate in handling extreme situations, we skip applying perceptual loss for that specific image.

3 Experiments

3.1 Mask Refine for LaMa

In this section, we focus on explaining our mask refinement method. As mentioned in SPIn-NeRF [7], applying masks that precisely match the unwanted objects can lead to unexpected outcomes when using the 2D inpainting network LaMa [9]. Specifically, LaMa [9] tends to preserve the silhouettes or edge shadows of the objects when the mask shape provides some "knowledge" about the masked area (Fig 5). In practice, we employ slightly more generous masks to achieve plausible inpainting results, such as dilation or scaling. However, this refinement process must be cautiously approached, as enlarging the mask can make it more challenging for LaMa [9] to generate reasonable priors.

In SPIn-NeRF [7], they employ a dilation operation with a 5×5 kernel for 5 iterations. However, this approach is too coarse for our scenes with various characteristics. For example, the mask may become too large after dilation and include areas that should remain if the target region is relatively small. Also, masks with highly irregular shapes can lose their original form, such as crossing slim chair legs. In addition, there may exist noises for complicated scenes or ambiguous prompts. However, dilation alone will magnify noises in the mask image. To address these issues, we provide three parameterized mask refinement operations that can be customized by users. These operations include dilation with adjustable kernel size and iteration steps, contour equidistant expansion with flexible scaling steps, and a filtering mechanism to keep a specified number of masks in an image and eliminate segmentation noise.

We have carefully fine-tuned the above operations for the scenes used in our experiments to ensure that the refined mask is suitable for LaMa [7] inference while minimizing changes to the segmentation results.

3.2 Implementation Details

All our experiments are conducted on a single RTX 3090 or A5000 GPU. We evaluated our method and the SPIn-NeRF [7] approach on all 20 scenes. We tested both the vanilla NeRF [6] and TensoRF [1] architectures using our implementation. Since the use of perceptual loss significantly increases memory usage, we train our Ours-NeRF [6] model with perceptual loss using a batch size of 1,

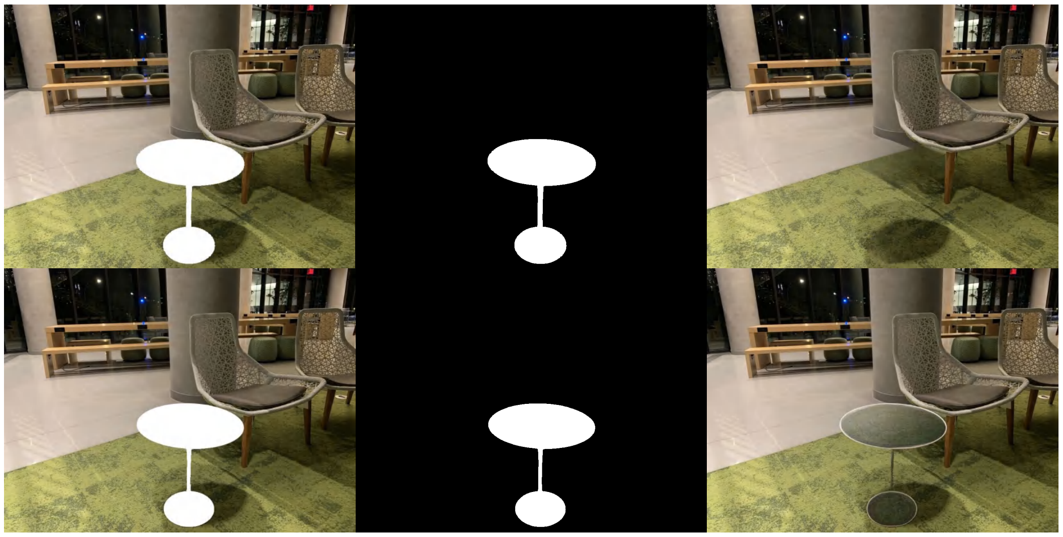


Figure 5: Example of LaMa [9] inpainting results with (top) and without (bottom) mask refinement. Images in the second row demonstrate the effect of using masks without refinement, which precisely match the objects but result in the retention of object silhouettes.

while SPIn-NeRF [7] and Ours-TensoRF [1] are trained with a batch size of 4. We follow the only configuration provided by SPIn-NeRF [7] to reproduce their results on all our scenes [2].

Regarding the testing phase, the ground truth images provided by SPIn-NeRF [7] after object deletion do not indicate which images they corresponded to before deletion. Therefore, we rendered these views using the camera parameters to generate the corresponding images for evaluation.

3.3 More Experiment Results

Quantity In this part, we give the detailed metrics tested in our experiments with all scene data available as Table 1.

Quality In this part, we give our experiments' detailed original and edited image pairs with all scene data available as Fig 6, 7 and 8.

[2]SPIn-NeRF GitHub

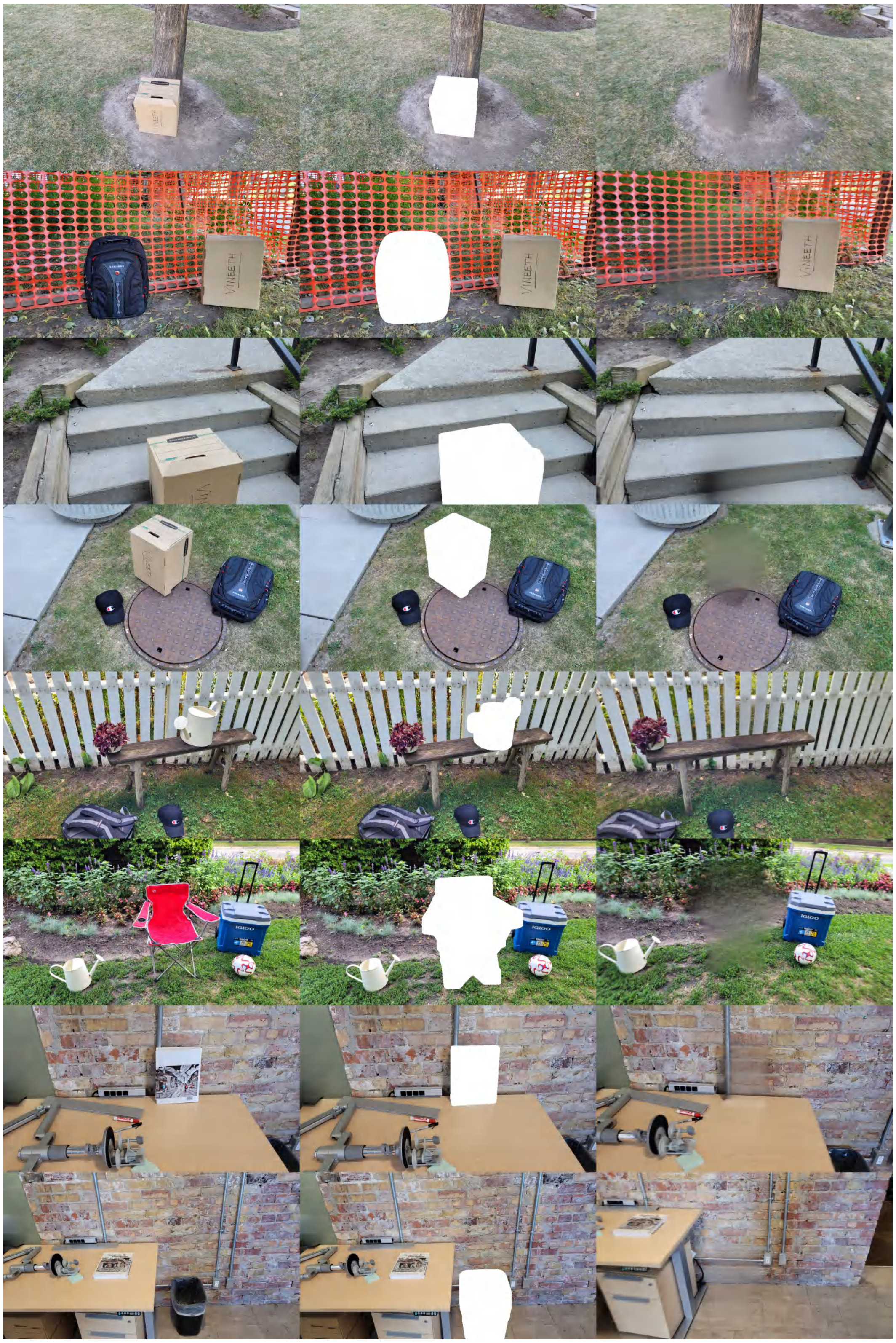


Figure 6: Original image (left), image with regions to be removed (middle), and images after removal from new views (right) for all scenes. (Begin)

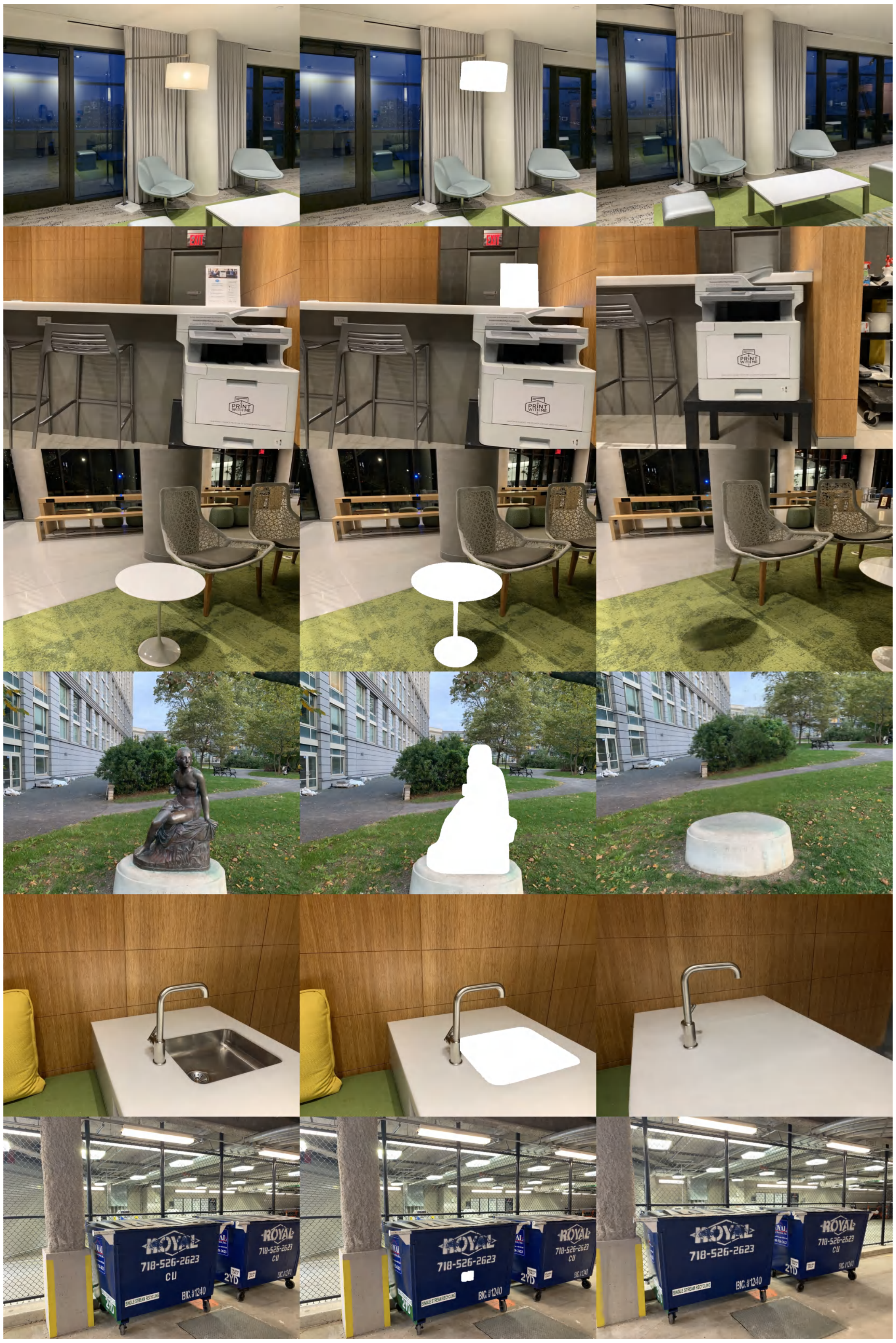


Figure 7: Original image (left), image with regions to be removed (middle), and images after removal from new views (right) for all scenes. (Cont)

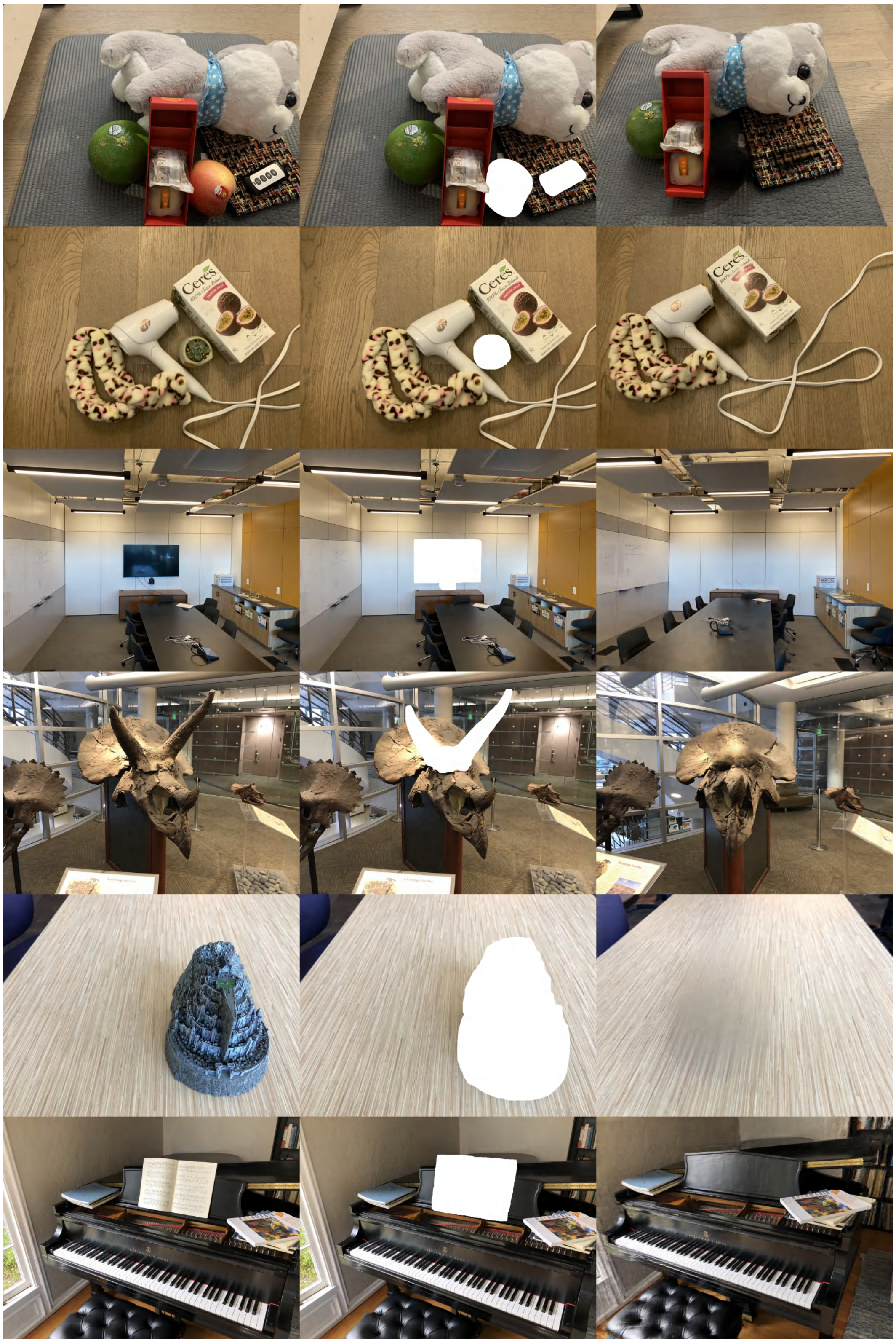


Figure 8: Original image (left), image with regions to be removed (middle), and images after removal from new views (right) for all scenes. (End)

Table 1: Experiment results for scene object removal. The first row indicates the scene name, and the first column indicates the method name. The abbreviations in the second column represent the loss modules used. 'dir' denotes training Neural Radiance Fields with LaMa [9] priors directly, 'dp' denotes partial depth, 'da' denotes all depth, and 'lpips' denotes the use of perceptual loss. It is worth noting that perceptual loss is always applied with all-depth supervision enabled.

			2	3	4	7	10	12	book	trash	Mean
Ours-NeRF [6]	dir	PSNR↑	16.05	11.60	13.05	14.44	13.02	11.50	15.88	16.77	14.04
		FID↓	102.41	46.29	80.40	38.64	49.14	42.67	90.33	39.02	61.11
		LPIPS↓	0.8556	0.5730	0.7560	0.7246	0.6541	0.8084	0.5446	0.5510	0.6834
	da	PSNR↑	15.99	11.61	13.06	14.45	13.01	11.48	15.92	16.77	14.04
		FID↓	106.36	49.04	81.42	39.28	55.07	45.87	95.98	48.62	65.21
		LPIPS↓	0.8542	0.5759	0.7624	0.7315	0.6605	0.8132	0.5484	0.5682	0.6893
	dp	PSNR↑	16.07	11.59	13.10	14.60	13.26	11.53	16.16	16.97	14.16
		FID↓	111.16	47.94	77.93	40.40	49.37	46.18	91.72	52.97	64.71
		LPIPS↓	0.8643	0.5757	0.7843	0.7418	0.6775	0.8177	0.5560	0.6000	0.7022
	lpips	PSNR↑	16.03	11.62	13.10	14.62	13.28	11.49	16.15	16.99	14.16
		FID↓	70.23	39.58	74.77	44.94	55.96	41.99	87.27	50.48	58.15
		LPIPS↓	0.8522	0.5532	0.7157	0.7157	0.6750	0.7593	0.5474	0.5917	0.6763
Ours-TensoRF [1]	dir	PSNR↑	15.91	11.40	13.04	14.41	12.88	11.37	15.84	16.62	13.93
		FID↓	90.05	37.04	80.09	37.55	43.84	38.38	64.63	34.66	**53.28**
		LPIPS↓	0.8039	0.5068	0.7200	0.6810	0.6179	0.7454	0.5167	0.5042	0.6370
	da	PSNR↑	15.92	11.41	13.05	14.51	13.15	11.37	16.03	16.85	14.04
		FID↓	91.64	38.70	82.86	37.37	49.12	39.72	94.61	80.29	64.29
		LPIPS↓	0.8071	0.5069	0.7298	0.6995	0.6496	0.7482	0.5234	0.5306	0.6494
	lpips	PSNR↑	15.94	11.42	13.01	14.46	13.12	11.41	16.05	16.84	14.03
		FID↓	72.10	34.83	76.56	36.53	47.27	38.65	95.62	76.39	59.74
		LPIPS↓	0.7909	0.4941	0.6790	0.6606	0.6443	0.7192	0.5159	0.5143	**0.6273**
SPIn-NeRF [7]	dir	PSNR↑	16.78	12.05	14.92	15.35	12.72	12.47	17.77	16.73	**14.85**
		FID↓	94.80	93.80	73.19	26.41	62.10	55.37	105.74	48.76	70.02
		LPIPS↓	0.8636	0.5839	0.7285	0.7028	0.6971	0.8110	0.4388	0.6222	0.6810
	da	PSNR↑	16.54	12.00	14.93	15.31	12.71	12.45	17.84	16.74	14.82
		FID↓	93.94	93.11	76.07	29.10	54.14	53.19	113.91	47.08	70.07
		LPIPS↓	0.8687	0.5653	0.7299	0.6941	0.6880	0.8071	0.4394	0.6090	0.6752
	lpips	PSNR↑	16.69	12.08	14.90	15.34	12.73	12.39	17.84	16.70	14.83
		FID↓	71.75	68.35	61.10	43.95	91.73	50.52	102.71	47.98	67.26
		LPIPS↓	0.8489	0.5472	0.6815	0.6552	0.7003	0.7518	0.4226	0.5972	0.6506

References

[1] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial Radiance Fields. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 333–350, 2022. ISBN 978-3-031-19823-6. doi: 10.1007/978-3-031-19824-3_20. URL `https://doi.org/10.1007/978-3-031-19824-3_20`.

[2] IDEA-Research. Grounded-sam. `https://github.com/IDEA-Research/Grounded-Segment-Anything`, 2023.

[3] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 694–711, 2016. ISBN 978-3-319-46475-6.

[4] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment Anything, 2023. URL `http://arxiv.org/abs/2304.02643`.

[5] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[6] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, pages 405–421, 2020. ISBN 978-3-030-58451-1. doi: 10.1007/978-3-030-58452-8_24. URL `https://doi.org/10.1007/978-3-030-58452-8_24`.

[7] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Konstantinos G. Derpanis, Jonathan Kelly, Marcus A. Brubaker, Igor Gilitschenski, and Alex Levinshtein. SPIn-NeRF: Multiview Segmentation and Perceptual Inpainting with Neural Radiance Fields, 2023. URL `http://arxiv.org/abs/2211.12254`.

[8] Johannes Lutz Schönberger, True Price, Torsten Sattler, Jan-Michael Frahm, and Marc Pollefeys. A vote-and-verify strategy for fast spatial verification in image retrieval. In *Asian Conference on Computer Vision (ACCV)*, 2016.

[9] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust Large Mask Inpainting with Fourier Convolutions. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3172–3182, 2022. doi: 10.1109/WACV51458.2022.00323.