

Supplementary Material

A. Discussion

A.1. Mask Propagation

While the focus of this paper is to obtain high-quality object-level 3D reconstruction by separating it from the full scene, we have proposed a mask propagation pipeline that can be used independently to obtain consistent 3D segmentation from a 2D mask. In this section, we discuss prior works that attempt to perform 3D segmentation from 2D. These techniques can be classified into two categories: automatic and user-assisted. Automatic methods include SAM3D [1], OmniSeg3D [12], SAM-guided Graph Cut [5], NeRF-SOS [4], and Garfield [6]. User-assisted methods include SA3D [2] and ObjectCarver.

SAM3D uses inconsistent segmentation pairs, projects them into 3D, merges the labels in 3D for each pair, and then aggregates all pairs. Unlike other methods, SAM3D does not require training a model, which makes it similar to our method in that sense. OmniSeg3D, on the other hand, performs automatic segmentation by hierarchically grouping similar components. Starting from multiview inconsistent segmentation, it uses contrastive learning to refine the segmentation. SAM-guided Graph Cut also performs automatic 3D scene segmentation but it begins with a mesh or point cloud and over-segments the 3D scene then uses SAM to group and build a graph, and applies a graph-cut algorithm to improve segmentation. NeRF-SOS predicts segmentation masks directly by adding features to NeRF without using SAM. However, it is limited to segmenting only one primary focus object, unlike other methods. Garfield hierarchically groups NeRF using contrastive learning. SA3D outputs segmented NeRF by training a neural field. It requires user input from a single view and iteratively improves segmentation using self-prompting.

Methods that output segmented NeRFs typically only segment the visible regions and do not account for occluded areas, leaving the model free to produce arbitrary results in those regions such as holes. In contrast, our method explicitly addresses occlusion, making it more robust. Nevertheless, these methods can be an alternative to our mask propagation approach, and we can use these to obtain consistent masks and proceed with the object separation stage of our pipeline. However, our method is advantageous in scenarios where training a full network to produce masks

is unnecessary or when hierarchical grouping fails. Our method could also be extended to automatic segmentation using an out-of-the-box single-image segmenter. Once this segmentation is obtained, our mask propagation framework can provide consistent 3D segmentation.

A.2. Computational efficiency

With respect to computational efficiency, our choice of scene modeling NeuS [10] could be replaced by any SDF-based methods that are more recent, potentially improving the processing time. We leave this exploration to future work.

B. Runtime analysis

We provide a runtime analysis for all stages of our method, considering an image size of 512×512 with a single RTX 3090 GPU.

- Stage 1: Scene reconstruction for 200k iterations takes 5.8 hours.
- Stage 2: Segmentation takes 2 minute per image.
- Stage 3: The amount of time Object Separation takes depends on the number of objects in the scene. For two, four, and six objects, Object Separation takes 2.7, 3.5, and 7.5 hours respectively.

C. Dataset

The problem of object separation in 3D reconstruction is a fairly new topic and, as such, lacks the proper benchmark dataset. Previous methods evaluated their approach on a cropped sub-meshes from a full scene, which has in holes in occluded regions. Thus, during the evaluation, the area that needs to be properly evaluated will be ignored. Figure 6 illustrates the issue with previous datasets. To address the gap in the literature, we introduce a new benchmarking dataset composed of real-world and synthetic scenes. Below, are the details on how we created the dataset.

C.1. Real-world dataset creation

Figure 11 shows the steps we took in creating the dataset. First, we scanned the individual object to obtain the ground truth mesh using Polycam. Second, we captured the scene using a handheld camera and then obtained the camera pose using COLMAP. The fourth step involved obtaining the full

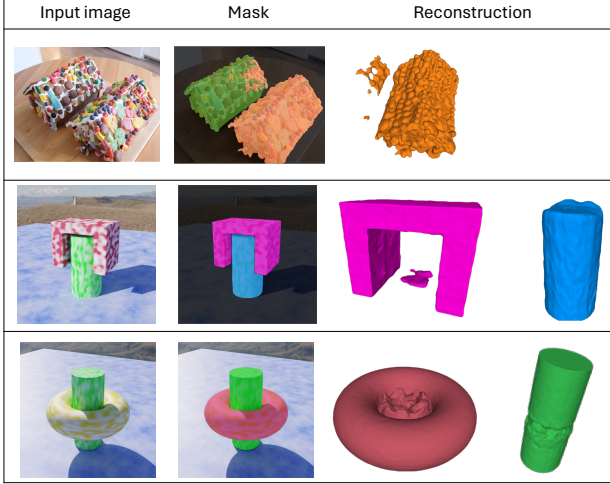


Figure 10. **Analysis of failure modes:** [first row] illustrates our method’s failure due to segmentation errors. Here, SAM itself struggles to obtain the correct mask as the two gingerbread houses are colorful, making it difficult to segment them. In the [second row], the segmentation is correct, but there is a flat surface floating in the pink box. This is due to the limitation of our amodal bounding box, which is contain this space from all views. In addition the overlap loss may not be computed at area as we are using fixed random points (of 10,000) in 3D space to compute the overlap loss and the pink floating surface is thin, and the points may not lie within this area. In the [third row], both the torus and the pipe occupy the space. Once again, this highlights the shortcomings of the amodal bounding box and overlap loss.

scene reconstruction so that we could use it to align the individual meshes. We trained vanilla NeuS to obtain the full mesh. The fifth step involved aligning the meshes, We used Blender software to transform the individual meshes to their respective positions within the scene. To further refine the alignment and avoid human error, we applied ICP to align the meshes together. Ultimately, we obtained each scene images, camera pose, and the transformation matrix for each individual object. Below, we describe in more detail how we scanned the individual meshes and the scene.

C.1.1 Individual mesh scanning.

We provide 22 object scans. We collected 80-150 images per object and used Polycam to generate the full mesh. We set the option *Isolate object from environment* to true and exported the final mesh in raw format. Since Polycam is not open-source software, we conducted an analysis of its reconstruction quality. We performed an experiment where we used the same object in different environments. We captured the object and obtained a mesh with Polycam and analyzed the robustness of Polycam. Figure 12 illustrates our analysis. We used a boot and took pictures in three different

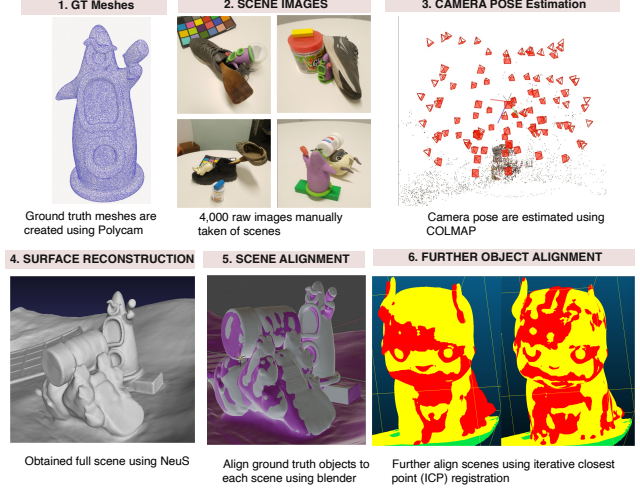


Figure 11. **Real-world dataset creation steps:** (1) Scanning individual objects with Polycam for ground truth meshes, (2) Capturing the scene with a handheld camera, (3) Estimating camera pose with COLMAP, (4) Full scene reconstruction for mesh alignment, (5) Aligning individual meshes with Blender to the full scene, and (6) Further aligning the meshes with ICP for precision. From this process, we obtain scene images, camera poses, and transformation matrices for each object.

environments and used these three sets of images to obtain the meshes. Then we compared them to each other to see if there is a significant quality drop. However, we found that they are very similar, and we concluded that Polycam is consistent in its output.

C.1.2 Scene capture.

We provide 32 real-world scene. We used a Samsung Note9 with a 12-megapixel camera to capture the scenes, utilizing the raw option in the pro-mode to capture raw images. The original images are 3008×3008 pixels with 16-bit depth. We converted the raw format to .png for lossless compression and downscaled it to 1002×1002 pixels. We show the steps we took to collect the dataset on Figure 11. Both the individual scene and individual objects can be found in Figure 15.

C.2. Synthetic data creation

We generated five realistic scenes, each with its own level of difficulty. Each scene has N objects, N ranging from 3 to 10. We used Blender to create the dataset, with each scene centered at the origin. We used white indoor scene environment lighting. We rendered the scenes with 500 samples at a resolution of 512×512 using the Cycles renderer, capturing 100 images from cameras positioned on the upper hemisphere around the subject.

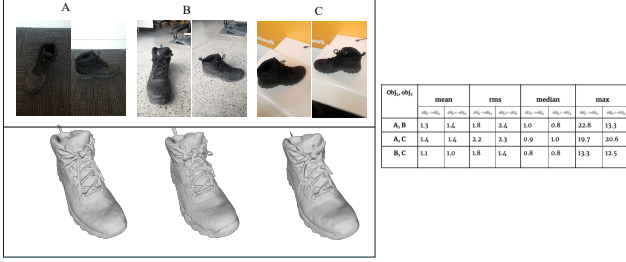


Figure 12. Analysis of the robustness of Polycam: [Left] images captured in different environments and their respective Polycam meshes. [Right] Quantitative evaluation where we compare each mesh against each other (unit in millimeters). We can observe that both qualitatively and quantitatively, Polycam is consistent.

D. Rejection sampling

When calculating the point-to-point Chamfer distance between predicted and ground-truth meshes, it is important to ensure locally similar point densities to avoid one of the directional Chamfer distances from being larger than the other. While the ideal metric here is the point-to-surface Chamfer distances (i.e. the average unsigned distances), this is often prohibitively slow; hence, it is common practice to resample the two point clouds to have the same number of points for an accurate point-to-point Chamfer distance. However, when the predicted mesh contains floaters and extraneous artifacts, this results in a diluted sampled point cloud and causes the densities to differ in the region of interest. The reverse holds true when the predicted mesh experiences carved out regions that lower its surface area for point sampling. ObjectSDF++ [11] and Rico [7] are evaluated on Replica [8] and ScanNet [3], and they clip the predicted meshes using the 3D bounds calculated from the ground-truth meshes. While this was likely done to try to ensure similar point densities, it removes any floater artifacts that exist outside the clipping bounds, yielding in a artificially lower smaller precision metric. We revise this evaluation by using a rejection sampling based approach that samples points only if it is some radius away from the growing list of samples. If the desired number of points is large enough to saturate the mesh (i.e. desired number of points is impossible due to the radius constraint), we ensure that the point density is similar between the two meshes.

E. Ablation

E.1. Effect of mask propagation on object reconstruction:

Figure 16 shows the effect of mask propagation using the mask obtained from each iteration. We see that the first iteration is not enough to capture the full object, as some parts of the segmentation are missing, resulting in carving

out. However, after the second iteration, we observe that we can obtain the full geometry.

E.2. Effect of increasing number of SDF parameters of baselines

One possible reason why ObjectCarver achieves more details in the reconstruction is that increase in model capacity with a whole SDF network being dedicated to each model. On the other hand, ObjectSDF++ and RICO use a single SDF network backbone with separate heads for each object’s SDF. In order to determine if the expressivity of the SDF networks in RICO and ObjectSDF++ is the limiting factor, we increase the learnable parameters of the SDF networks in RICO and ObjectSDF++ by the number of objects k . For ObjectSDF++, we increase the dimensionality of the feature vector learned at each level of the hash-grid by k . For RICO, we increase the width of the network by $\lceil \sqrt{k} \rceil$. These modified models are called RICO* and ObjSDF++*.

Figure 18 shows the evaluation results of this comparison. RICO* tends to over-smooth the geometry and create more floater artifacts than RICO. ObjectSDF++ achieves better reconstruction quality and even reduces the number of floaters. However, ObjectCarver still achieves the best qualitative and quantitative results among the baselines, demonstrating the effectiveness of scene initialization for learning the geometries with more parameters.

F. Additional Results

We provide more qualitative results on our dataset in Figure 19 and large indoor scene on Replica and Scannet dataset Figure 17. We can observe that our method produces much higher quality and fewer floating artifacts compared to previous methods.

For the indoor scene, we build on NeuRIS [9] instead of NeuS [10] due to the latter’s difficulty with indoor and large scenes. We omit a quantitative table because neither Replica nor Scannet includes complete ground-truth geometries. For the indoor scene, one could also represent the background as a separate SDF, though our background SDF would not benefit from the compactness loss, making our method similar to ObjectSDF++ [11].

G. Implementation details

G.1. Coreset Algorithm

This algorithm takes as input a set of projected 2D points and selects n points that will later be used as *seed points* for SAM to segment a specific object (*seed points* being the set of (x, y) coordinates used to prompt SAM to segment the object). The intuition behind using this algorithm is to simulate how a human would select the seed point to segment an object using SAM. It starts by clicking the centroid of the object; the next point will be far away from the centroid,



Figure 13. Full list of captured real-world scenes.

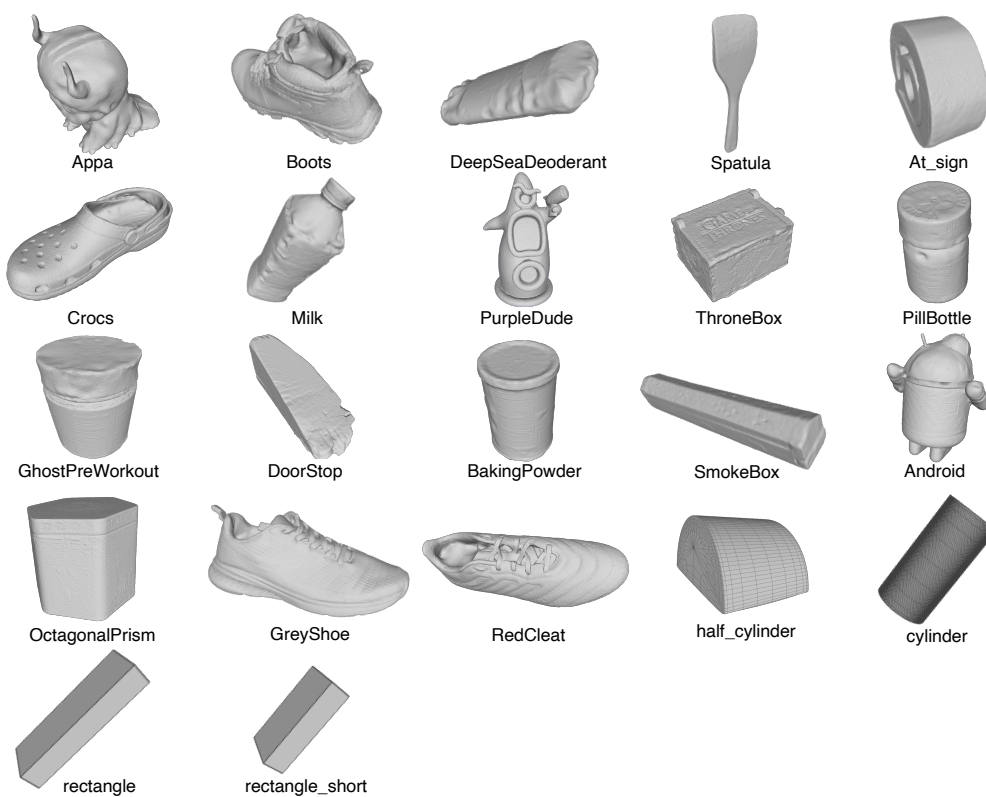


Figure 14. Full list of scanned individual meshes using Polycam.

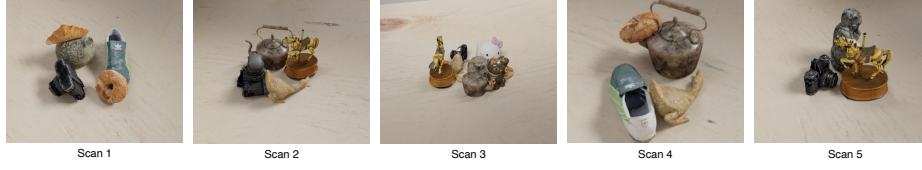


Figure 15. Full list of synthetic scenes.

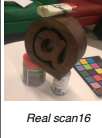
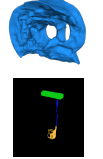
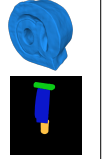
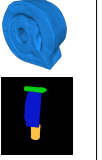
			
Real scan16			
chamfer	0.16	0.143	0.143
Iteration	1	2	3

Figure 16. Effect of mask propagation on the reconstruction, [Top] reconstruction of using the mask, [bottom] mask used for.

and the following point will be far away from both of the previous points. As a result, these points capture the overall shape of the object. We chose $n = 15$ as it works for most cases.

Algorithm 1 Modified Coreset Algorithm

- 1: **Input:** projected points $S \subset \mathbb{R}^2$, coreset size n
 - 2: **Output:** C
 - 3: $C \leftarrow \{\}$
 - 4: $x_0 \leftarrow \arg \min_{s \in S} \|s - \text{mean}(S)\|_2$
 - 5: $C.add(x_0)$
 - 6: $S.remove(x_0)$
 - 7: **while** $|C| < n$ **do**
 - 8: $y \leftarrow \arg \max_{s \in S} \min\{\|s - c\|_2 : c \in C\}$
 - 9: $C.add(y)$
 - 10: $S.remove(y)$
-

G.2. Partial depth ordering

When there is an overlap between two segmentation outputs from SAM, the partial depth ordering is used to break the tie. Below we describe the steps:

Step 0: Initialization

- Initialize the depth as zero for each of the K objects.

Step 1: Overlap Checking

- For each pair of objects (k_h, k_i) where $h \neq i$: Check if the segmentation masks of object k_h and object k_i overlap.

Step 2: Overlap Resolution

- If there is an overlap between the segmentation masks of objects k_h and k_i :
 1. Count the number of seed points in the overlapping region for both segmentation masks.
 2. Identify the object with more seed points in the overlapping region as the "top" object and the one with fewer seed points as the "bottom" object.
 3. Increase the depth of the top object by one relative to the depth of the bottom object.






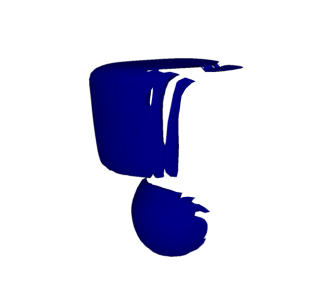
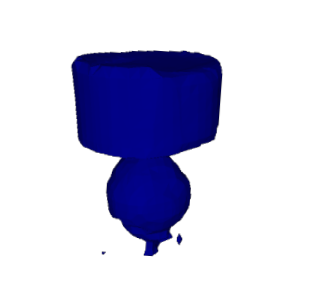
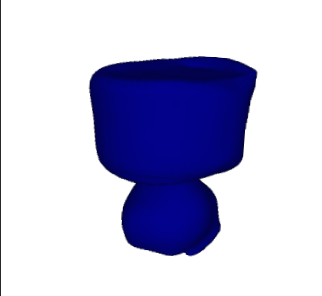
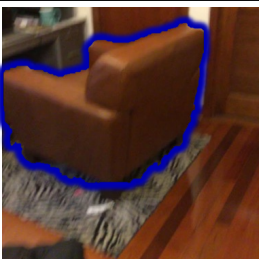
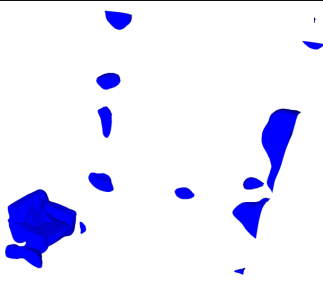
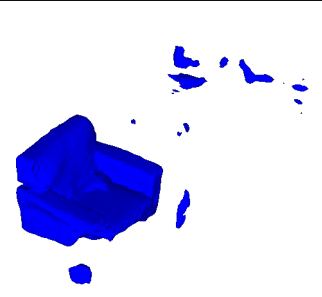
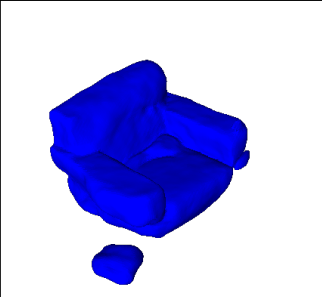

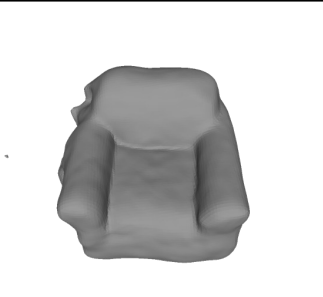
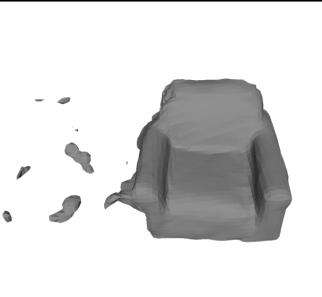
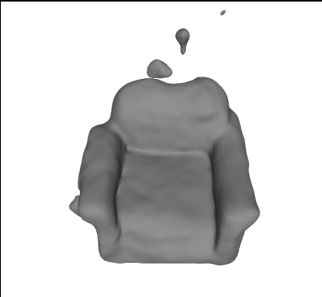
 <p><i>Replica scan1</i></p>			
 <p><i>Replica scan2</i></p>			
 <p><i>Scannet scan1</i></p>			
 <p><i>Scannet scan4</i></p>			
Scene	Rico	ObjSDF++	Ours

Figure 17. Qualitative comparison on large indoor scenes: Due to our compactness loss, our method results in fewer artifacts compared to the baseline, which is plagued by floating artifacts, most apparent in row 3, and carving of the objects shown in row 2 of the RICO output.

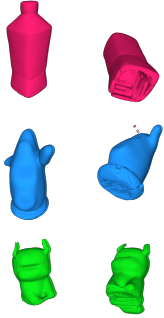
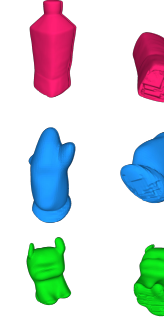
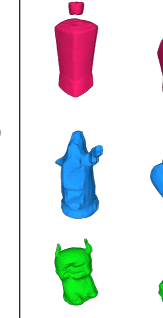
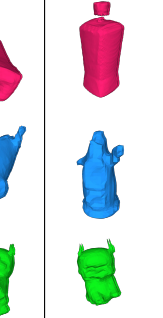
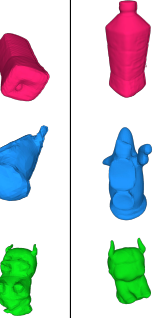
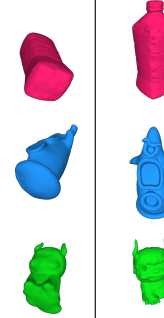
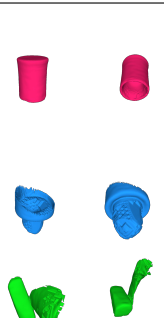

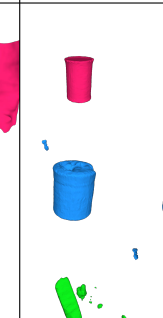
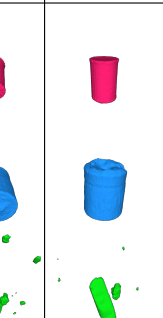
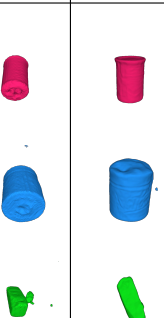
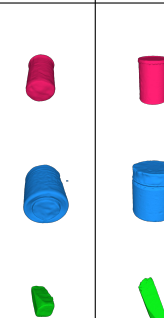
real_scan3	 Chamfer*: 0.00665	 Chamfer*: 0.00605	 Chamfer*: 0.00720	 Chamfer*: 0.0246	 Chamfer*: 0.00443	
	 Chamfer*: 0.0654	 Chamfer*: 0.608	 Chamfer*: 0.0401	 Chamfer*: 0.0228	 Chamfer*: 0.00489	
	RICO	RICO*	ObjSDF++	ObjSDF++*	Ours	GT

Figure 18. Comparison of increasing the expressivity of the model backbone of RICO and ObjectSDF++











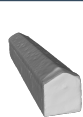

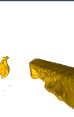
























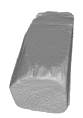





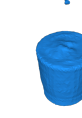
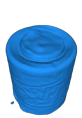




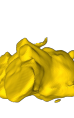




 <i>Real scan3</i>								
 <i>Real scan4</i>								
 <i>Real scan6</i>								
 <i>Real scan7</i>								
 <i>Real scan9</i>								
 <i>Real scan12</i>								
Scene	GT	Rico	ObjSDF++	Ours	GT	Rico	ObjSDF++	Ours

Figure 19. Comparison between RICO, ObjectSDF++ and our approach. ObjectSDF++ produces fewer details and more floating artifacts

References

- [1] Nhat-Tan Bui, Dinh-Hieu Hoang, Minh-Triet Tran, Gianfranco Doretto, Donald Adjeroh, Brijesh Patel, Arabinda Choudhary, and Ngan Le. Sam3d: Segment anything model in volumetric medical images. *arXiv:2309.03493*, 2023. 1
- [2] Jiazhong Cen, Zanwei Zhou, Jiemin Fang, Wei Shen, Lingxi Xie, Xiaopeng Zhang, and Qi Tian. Segment anything in 3d with nerfs. *arXiv preprint arXiv:2304.12308*, 2023. 1
- [3] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 3
- [4] Zhiwen Fan, Peihao Wang, Yifan Jiang, Xinyu Gong, De-jia Xu, and Zhangyang Wang. Nerf-sos: Any-view self-supervised object segmentation on complex scenes, 2022. 1
- [5] Haoyu Guo, He Zhu, Sida Peng, Yuang Wang, Yujun Shen, Ruizhen Hu, and Xiaowei Zhou. Sam-guided graph cut for 3d instance segmentation. In *ECCV*, 2024. 1
- [6] Chung Min Kim, Mingxuan Wu, Justin Kerr, Ken Goldberg, Matthew Tancik, and Angjoo Kanazawa. Garfield: Group anything with radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21530–21539, 2024. 1
- [7] Zizhang Li, Xiaoyang Lyu, Yuanyuan Ding, Mengmeng Wang, Yiyi Liao, and Yong Liu. Rico: Regularizing the unobservable for indoor compositional reconstruction, 2023. 3
- [8] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijnmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 3
- [9] Jiepeng Wang, Peng Wang, Xiaoxiao Long, Christian Theobalt, Taku Komura, Lingjie Liu, and Wenping Wang. Neuris: Neural reconstruction of indoor scenes using normal priors. *arXiv preprint*, 2022. 3
- [10] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *CoRR*, abs/2106.10689, 2021. 1, 3
- [11] Qianyi Wu, Kaisiyuan Wang, Kejie Li, Jianmin Zheng, and Jianfei Cai. Objectsdf++: Improved object-compositional neural implicit surfaces, 2023. 3
- [12] Haiyang Ying, Yixuan Yin, Jinzhi Zhang, Fan Wang, Tao Yu, Ruqi Huang, and Lu Fang. Omnise3d: Omniversal 3d segmentation via hierarchical contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20612–20622, 2024. 1