ObjectCarver: Semi-automatic segmentation, reconstruction and separation of 3D objects

Gemmechu Hassena, Jonathan Moon, Ryan Fujii, Andrew Yuen,
Noah Snavely, Steve Marschner, Bharath Hariharan
Cornell University

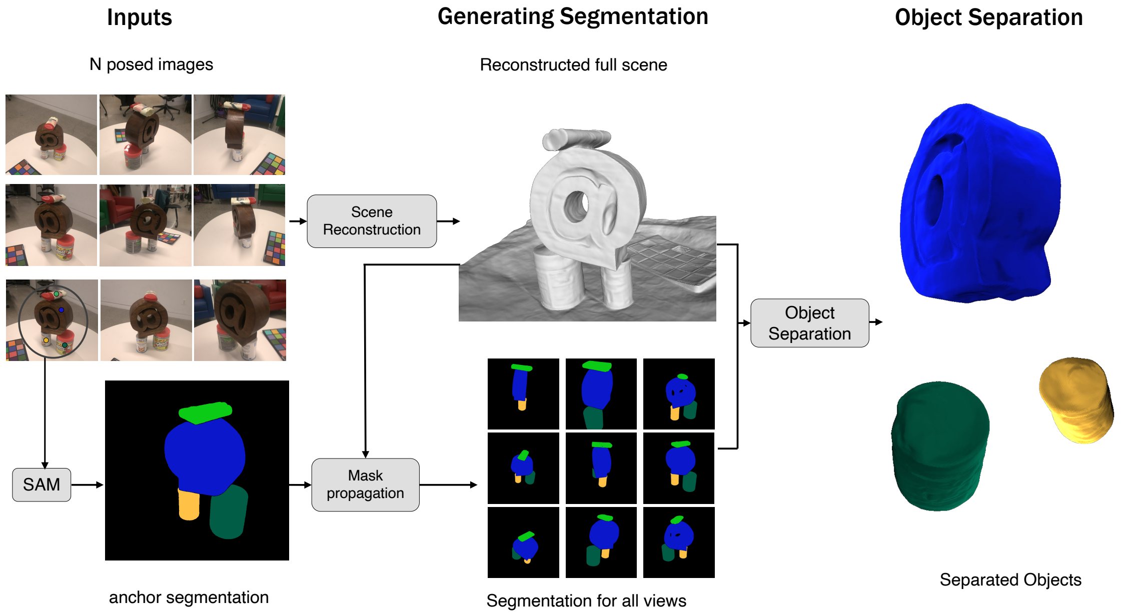


Figure 1. Our approach takes multiple views of a scene as input, along with a few point clicks on one of the views, which can be converted into segmentation masks (left). It then: (a) segments all the other images, and (b) reconstructs each segmented object, completing the occluded regions if any.

Abstract

Implicit neural fields have made remarkable progress in reconstructing 3D surfaces from multiple images; however, they encounter challenges when it comes to separating individual objects within a scene. Previous approaches to this problem require ground-truth segmentation masks and introduce floating artifacts in occluded parts of the scene. We address these challenges with ObjectCarver. ObjectCarver requires no ground-truth segmentation; all it needs is just a few user clicks in a single view. ObjectCarver also introduces a new loss function that prevents floaters and avoids inappropriate carving-out due to occlusion. Finally, ObjectCarver uses a simple initialization technique that significantly speeds up the process while preserving geometric details. We demonstrate qualitatively and quantitatively on multiple datasets (including a new dataset and benchmark with complete ground-truth) that ObjectCarver produces more accurate reconstructions of each object while minimizing artifacts.

1. Introduction

With recent advances in neural implicit scene representations, we can now reconstruct 3D scenes with complete, high-quality surfaces (represented as signed distance functions or SDFs) from a set of images taken by cameras with known poses [30, 36]. Although these techniques compute high-quality surfaces, they are limited to representing the entire scene as a single surface. This representation is fine for applications such as walkthroughs where the scene remains fixed, but for many applications it is desirable to extract and manipulate individual objects, including applications in robotics and virtual reality where simulating such scene manipulations is crucial. In this paper, we tackle this problem of 3D scene decomposition: given multiple views of a 3D scene, can we produce a reconstruction where the individual objects are separated out?

Some previous works [13, 31, 32, 34] have addressed the problem of reconstructing many separate objects. However, two key challenges remain. First, these techniques require segmentation masks of each object in each view as part of the input. Unfortunately, the cost of the manual work involved in producing such segmentations scales with the number of input views and the number of objects, making the process cumbersome.

Automated solutions like the Segment Anything Model (SAM) [10] often over-segment and result in inconsistent segmentation across multiview images (Fig. 2, left). Recent works, such as SA3D [7], that attempt this problem use volume density, but volume density cannot locate the surface exactly.

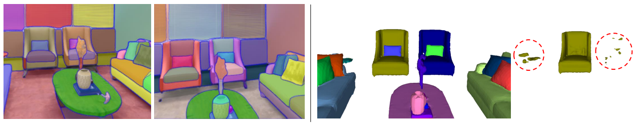


Figure 2. **Failure cases of SOTA.** Using SAM independently on each image precludes corresponding objects between views (Left). Even if one were to solve this correspondence problem, slight errors in SAM output mean that the same object may be segmented differently in different views (e.g., the top of the vase is included in the vase segment in the left image but not the right). Even with GT segmentations, prior work such as ObjectSDF++ [32] introduces floating artifacts, especially those hidden behind other objects (Right).

Second, prior work fails in the presence of occlusion (Fig. 2, right). Parts of the scene that are occluded from all views provide no supervision for existing techniques, giving the model free rein to introduce floating components in the occluded regions. These floating artifacts can be large and numerous and as such result in extremely inaccurate object reconstructions.

We introduce ObjectCarver to address these limitations. ObjectCarver takes as input a collection of posed images and point clicks of each object for segmentation in just *one* of the views. ObjectCarver then outputs object segmentations for all input views and a high-fidelity 3D surface for each object (Figure 1). This 3D surface includes not just the parts of the object that are visible but also makes reasonable completions in completely occluded regions where no image evidence is available. Crucially, ObjectCarver removes almost all floating artifacts that plague prior work. Finally, ObjectCarver achieves this reconstruction with a fairly small computational overhead beyond the computational cost of full scene reconstruction.

ObjectCarver works in three phases. First, we reconstruct the entire 3D scene as a single SDF using existing methods [30]. Then, from one segmentation mask (computed from the user's input clicks using SAM [10]), we use the reconstructed 3D surface together with SAM [10] to propagate segmentation labels to the other input images, resulting in accurate and multi-view consistent masks for each object. Finally, we jointly train per-object SDF surfaces, starting from the full-scene SDF. We introduce a novel loss function to produce a set of consistent and compact 3D surfaces.

Finally, we find that existing benchmarks for this task are limited, with incomplete ground-truth object meshes and metrics that do not correctly penalize floaters. Therefore, we introduce a new dataset of both synthetic and real-world scenes consisting of multiple objects and equipped with a ground-truth mesh for each object. We also introduce updated metrics that correctly penalize all error modes. We compare our method with prior methods both qualitatively and quantitatively in this benchmark and demonstrate that our method outperforms the previous methods for this problem. In sum, our contributions are:

1. A new **automatic segmentation** approach that leverages the 3D scene structure to generate object segmentations for all the input images from just a few points the user clicks in one view.
2. A new **object compactness loss** that removes floaters in occluded regions and produces substantially more accurate reconstruction.
3. A change of **initialization** for the object models that improves surface quality and considerably speeds up convergence.
4. New synthetic and real-world **datasets** of multi-object compositional scenes and their individual geometries.

2. Related Work

Neural field representations for geometry. Neural representations for surface geometry began with methods that trained using 3D supervision [16, 22], but soon began to focus on using more readily available multi-viewpoint images as supervision [21, 35]. Neural Radiance Fields [17] introduced a framework to use volumetric rendering to train radiance fields, leading to follow-on work improving training and rendering speed [19, 25, 29, 37, 39], handling complex, unbounded, and dynamic scenes [4, 12, 15, 23, 24, 42], and improving representation quality [3, 5].

To obtain more explicit geometric representations than NeRFs provide, some recent advances have optimized neural signed distance functions (SDFs) by using them to define smooth volume densities that are rendered in the NeRF framework, which helps guide the training process stably to accurate and detailed surfaces. VolSDF [36], NeuS [30] and Neuralangelo [14] achieve good surface reconstructions in this way; building on these methods, MonoSDF [41] incorporates monocular cues and PermutoSDF [27] achieves detailed reconstructions of small-scale features.

Decomposing 3D scenes into objects. The methods above focus on reconstructing geometry or radiance fields but don't address scene understanding as compositions of objects. Several approaches have been proposed for disentangling objects, some Some of these methods learn from observing scenes without further supervision. Niemeyer and Geiger proposed GIRAFFE [20], which utilizes latent codes to generate object-centric NeRFs and conceptualize scenes as compositional generative neural feature fields. uORF [40] learns unsupervised object composition models that can be used to factor new scenes at inference time [40]. DiscoScene [33] uses weak supervision in the form of *layout prior* for object-compositional generation but fails to generalize to unknown objects. In contrast to the high-level object decompositions of the above work, Differentiable Blocks World [18] trains a mid-level scene representation from multiple images. Rather than achieving the highest geometric quality, that method aims to decompose the scene into mid-level 3D textured primitives. In contrast ObjectCarver aims to separate the 3D objects with high geometric quality.

Other work uses joint language-visual embeddings like CLIP to identify objects in 3D scenes. Sosuke *et al.* use CLIP and DINO to learn neural feature fields, supporting editing and selection mechnisms [11]. LERF [8] learns a language field by volumetrically rendering proto-CLIP features along the ray which is supervised with multi-scale CLIP features on the training images, allowing radiance fields to be decomposed into semantically distinct areas.

In contrast to CLIP, our method relies on a pre-trained 2D image segmentation network. Other work in this vein includes [34], which separates scenes into disjoint radiance fields for each object based on rough 2D instance masks.

More recently, the emergence of SAM [10] marked a significant step towards segmenting 2D images. GARField [9] and OmniSeg3D [38] hierarchically group NeRFs using SAM. Segment Anything 3D (SA3D) [7] uses mask inverse rendering and cross-view self-prompting to construct 3D masks, demonstrating adaptability to various scenes and efficiency in achieving 3D segmentation. However, unlike our method, SA3D segments a fixed 3D representation and does not attempt to *separate* objects from one another, e.g, to modify their geometry to fill in holes at interfaces where they are in contact. Further discussion of similar prior work is in the supplementary material.

Another key difference with the above work is that we seek not to produce segmented NeRFs, but instead segmented, separated, and high-quality *surfaces* in the form of SDFs that can be converted into convenient graphics representations like meshes. In that sense, our work is similar to ObjectSDF [31], which uses per-image input instance masks to product an SDF for each object. However, this method can encounter issues with object and scene reconstruction accuracy, slow convergence, and training speed. Its successor ObjectSDF++ [32] introduces an occlusion-aware object opacity rendering strategy and an overlap regularization term to better separate the surfaces between neighboring objects. However, it still requires per-image, per-object input masks, in contrast to our method. RICO [13] leverages geometrically motivated regularizations to smooth unobserved regions in indoor compositional scenes, whereas our method goes farther to separate and reconstruct complete objects. Our method is in the spirit of other semi-supervised methods like that of Ren *et al*. [26], but scales well to complex scenes with many objects.

3. Methodology

We assume that we are given a set of N posed images $\mathcal{I} = \{I_1, \dots, I_N\}$ of a scene. We are interested in not just reconstructing the scene, but segmenting, reconstructing and separating each of K different objects in the scene. We aim to do so as accurately, as efficiently, and with as little manual annotation as possible.

Our proposed approach operates in three stages:

1. Reconstruct the full scene as a single SDF.
2. Generate segmentation masks for each of the K objects in all images by propagating segmentation masks from one of the views.
3. Optimize K separate SDFs using a novel loss to handle occlusion for accurate reconstruction.

Next we describe each step below.

3.1. Scene Reconstruction

We first train a full scene reconstruction. Any SDF-based technique can be used; however, here we use NeuS [30] which converts the SDF into a density term to allow for optimization through volumetric rendering. Concretely, for every pixel, discrete samples are taken along the corresponding ray $\{\mathbf{p}_i = \mathbf{o} + t_i\mathbf{v} \mid i = 1, \dots n, t_i < t_{i+1}\}$ where $\mathbf{o}$ is the camera center and $\mathbf{v}$ is the viewing direction corresponding to the pixel. Then NeuS calculates densities α_i and an accumulated transmittance $T_i = \prod_{j=1}^{i-1}(1 - \alpha_i)$. The density is shown to be related to the SDF as:

$$\alpha_i = \max\left(\frac{\Phi_s\left(f(\mathbf{p}_i)\right) - \Phi_s\left(f(\mathbf{p}_{i+1})\right)}{\Phi_s\left(f(\mathbf{p}_i)\right)}, 0\right) \tag{1}$$

where Φ_s is the sigmoid function and f is the SDF. (Please refer to Wang et. al [30] for details.) Given these densities α_i and the corresponding accumulated transmittance, the rendered color at this pixel is computed as:

$$C(\mathbf{o}, \mathbf{p}) = \sum_i T_i \alpha_i c(\mathbf{p}_i, \mathbf{v}) \tag{2}$$

where $c(\mathbf{p}_i, \mathbf{v})$ is the color at the point $\mathbf{p}_i$ seen from the viewing direction $\mathbf{v}$.

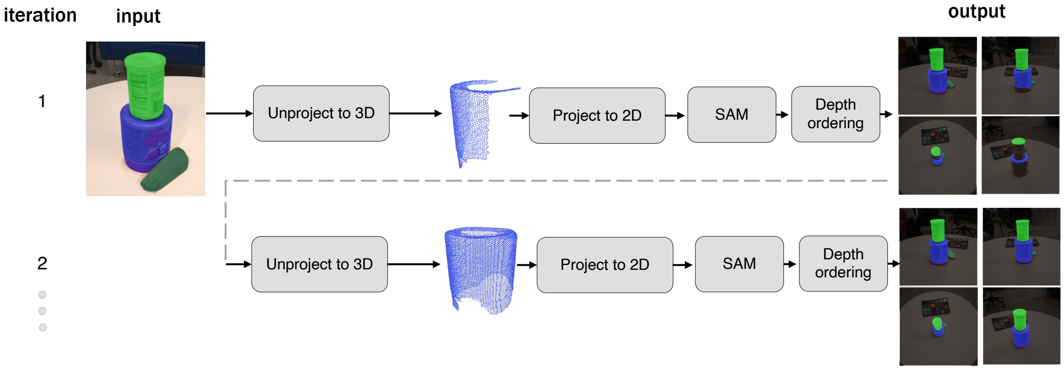


Figure 3. **Mask Propagation pipeline:** in the first iteration, a user clicks a point on each object and we use SAM [10] to generate the anchor mask, which is then unprojected into 3D (here, we only show unprojected 3D points for the bottom can). These 3D points are subsequently projected back into each image view, while checking for occlusions. The projected points serve as seeds for SAM [10] to generate masks for each object (bottom and top cans, door stop). To combine these individual segmentation masks into a single image, we use a depth ordering technique. In the next iterations, all views are used as anchor masks, allowing the pipeline to cover previously unseen regions.

The SDF is optimized to minimize rendering and eikonal losses:

$$L = L_{\text{color}} + \lambda L_{\text{eik}} \tag{3}$$

$$L_{\text{color}} = \frac{1}{m} \sum_{j} \|\hat{C}_j - C_j\| \tag{4}$$

$$L_{\text{eik}} = \frac{1}{nm} \sum_{j,i} (\|\nabla f(\mathbf{p}_{j,i})\|_2 - 1)^2 \tag{5}$$

Here j indexes over pixels in all images and i indexes over points sampled along a ray. $\hat{C}_j$ is the predicted color, C_j is the observed color, m is the number of pixels, n is the number of samples per ray, and $\mathbf{p}_{j,i}$ is the sampled point along pixel j at index i.

3.2. Generating Segmentations

Our next step is to generate segmentation for all the views. Given a few point clicks in one of the views, we use SAM [10] to generate the segmentation; we call this our anchor mask. Then we unproject the mask onto the reconstructed 3D scene, resulting in labeled 3D points for each object. Using these labeled 3D points we propagate the segmentation to all views. Finally, we iterate through this process again, using the newly obtained segmentation as the anchor mask. Below we describe each step in detail.

3D point labeling: After generating the anchor mask, we project it into 3D by tracing rays from each pixel through the object mask to determine surface intersections (Figure 3). However, segmentations can often be imprecise near object boundaries, causing the mask to leak onto other surfaces (Figure 4). To address this, we first erode the mask to remove any segmentation errors where the mask overshoots the true boundary. Second, after back-projecting the points to 3D we remove from each object mask all points whose depths are outliers, i.e., more than 2.5 standard deviations from the mean object depth. Finally, we subsample the 3D points to speed up downstream tasks, and ensure that each 3D point has a unique label by discarding points with more than one label to avoid inconsistent segmentations later on.

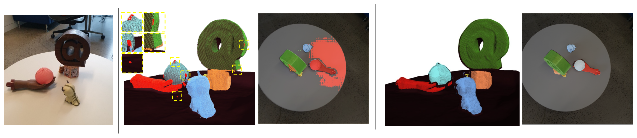


Figure 4. **Projection to 3D.** Left: Example image. Middle: points projected without mask edge erosion and outlier removal, resulting in noisy segmentation outputs. Right: by using mask erosion and outlier removal we obtain clean 3D points and subsequently obtain a correct segmentation output.

Propagating to a new view: To segment an object in a new view, we project the labeled 3D points into that view if they are unoccluded to obtain labeled 2D image points ("seeds"). While these 2D points can theoretically be used to prompt SAM, it often oversegments when presented with numerous seeds. To address this, we use a coreset selection algorithm (in the supplementary) to reduce the number of seed points while maintaining the object's shape. Finally, to resolve multiple overlapping segmentations from SAM [10], we perform a partial ordering based on depth, comparing the depths of seed points in overlapping areas and assigning the region to the object that is closer. For example, in Figure 3, this depth ordering enables correct placement of the green can pixels in front of the blue can when viewed from the top.

3.3. Object Separation

Given the images and their segmentation masks, our goal is now to produce K separated SDFs, one per object. We can train the K SDFs by updating the color loss so that each SDF is responsible only for the colors of its corresponding

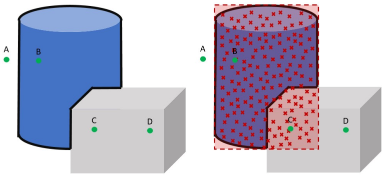


Figure 5. An occlusion event. The object of interest is the blue cylinder. On the left is the segmentation mask. On the right, the crosses (not included in the segmentation mask) represent points on the blue object that are visible in other views but occluded in this view. The red dotted box is the amodal mask, and its intersection with the occluding cuboid is the set of pixels that are "present" in the blue object, but occluded in this view.

object:

$$L_{\text{color}} = \frac{1}{m} \sum_k \sum_j M_k(j) \|\hat{C}_j - C_j\| \tag{6}$$

Here M_k is the mask for k-th object. However, this is not enough to separate out the object because the segmentation mask only covers the visible parts. When a pixel isn't in the mask, it's unclear if it's outside the object's extent or just occluded. In what follows, we first discuss the simpler case of unoccluded objects and then discuss the precise ambiguities and our proposed solution.

Special case: Unoccluded objects without contacts. Consider first the special case where each object is completely visible in each image, and does not make contact with any other part of the scene. In this case, given a candidate object SDF, we can *render* an object mask for each input viewpoint by aggregating the density along each ray. We can then add a loss term that encourages this predicted mask to match the provided segmentation mask using a simple binary cross entropy loss. Concretely, similar to Equation 2, for every object and for every pixel we calculate a *mask loss*:

$$L_{\text{mask}} = \sum_k \sum_j \text{BCE}(\hat{O}_k(j), M_k(j)), \hat{O}_k = \sum_i T_i \alpha_i^k \tag{7}$$

Here, $\hat{O}_k$ is a score representing the opacity of this pixel in the k-th SDF. This loss, as proposed in NeuS [30], causes problems in scenes with occlusions. To see this, consider Figure 5, where a blue object is occluded by a gray box. If we look at our mask loss behaviour for the blue object at various pixels, The loss would suggest that pixels A, C, and D should lie completely outside the object. Clearly, this is not the right behavior at pixel C, and thus we need a different strategy to handle occlusion.

Resolving occlusion: One option is not to impose any loss on pixels C and D at all. In other words, we could exclude all pixels where the object of interest is occluded by another object. Past work proposes an occlusion-aware loss which has a similar effect [32]. However, the effect of this is that the trained SDF may now include artifacts that are occluded from view in all images without incurring any penalty. While this kind of an object is *possible* given the input views, our intuition tells us that it is highly unlikely.

Instead, we propose a prior that the object should only include surfaces that are visible from some input view. In other words, we would like a *compact completion* of the visible surfaces that we see in the input views. Thus, in Figure 5, we would be okay with the object including point C, but not okay with any artifacts that include the point D.

We formalize this intuition as follows: Given the labeled 3D points of object k we project all these points into every view without regard to occlusion (producing e.g., the crosses in Figure 5). In each view, we then take the *bounding box* of these projected points; this is the *amodal* bounding box of the object, $\mathcal{B}_k$ (*amodal* completion is the phenomenon where humans perceive the complete shape of a background object in spite of occlusion [6]). We then intersect this amodal bounding box with the provided segmentation masks of the *other objects* to get a "present-but-occluded" mask M^{occ}. We then only apply the mask loss above to pixels outside this present-but-occluded region. Here $\mathbf{p}_k$ is the set of 3D points for object k in equation 8.

$$\mathcal{B}_k = \text{Bounding Box}\left(\pi(\mathbf{p}_k)\right) \tag{8}$$

$$M_k^{\text{occ}} = \mathcal{B}_k \cap (\cup_{i \neq k} M_i) \tag{9}$$

$$L_{\text{compactness}} = \sum_k \sum_{j \notin M_k^{\text{occ}}} \text{BCE}\left(M_k(j), \hat{O}_k(j)\right) \tag{10}$$

Resolving object interfaces. A final step is to resolve object interfaces, to ensure that each object occupies a distinct region of 3D space and does not intersect others. For this we use a loss term that we call the *overlap* loss. It adds a penalty whenever the interiors of two objects overlap. Concretely, suppose we have K SDFs $f_1, \ldots, f_K$.

For every 3D point $\mathbf{p}$ sampled randomly in space, we identify the SDF that yields the most negative value (i.e., the object for which $\mathbf{p}$ is farthest into the interior), and penalize negative values from all other SDFs using a hinge loss:

$$k^* = \arg\min_k \left(f_k(\mathbf{p})\right) \tag{11}$$

$$L_{\text{overlap}} = \sum_{\mathbf{p}} \sum_{k \neq k^*} \max\left(f_k(\mathbf{p}), 0\right) \tag{12}$$

Our final loss function is:

$$L = L_{\text{color}} + \lambda L_{\text{eik}} + \beta L_{\text{compactness}} + \gamma L_{\text{overlap}}. \tag{13}$$

We train all K SDFs in parallel using this loss with the hyper-parameters $\lambda = 0.1$, $\beta = 0.9$, $\gamma = 0.001$. We tested on 1 RTX 3090 GPU. We used batch sizes of 512 and 64 for the full scene reconstruction and per-object reconstruction respectively.

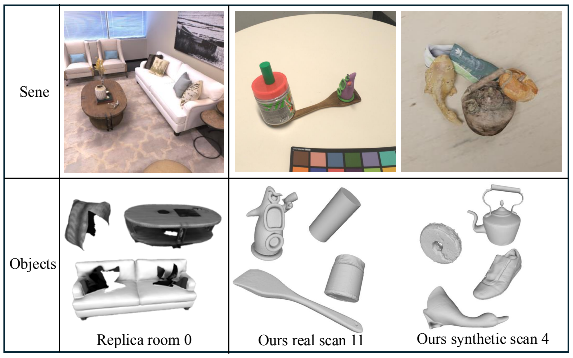


Figure 6. Left: Previous datasets, like Replica, exhibit holes in the individual objects in occluded regions. As a result, using the cropped sub-meshes as ground truth for object separation is not an adequate evaluation. Middle and right: Our proposed dataset features complete individual objects.

Initialization. One challenge with the proposed approach is that it trains K different SDFs, and so can be up to K times as expensive as training the single scene SDF. Prior work uses various strategies to reduce this training cost, such as sharing layers between the SDFs [32] or distilling from the scene SDF [34]. We propose a simpler strategy that significantly reduces running time (to a few hours instead of days) and yet preserves details: we initialize each SDF with a copy of the full scene SDF (unlike ObjectSDF++, which uses a sphere initialization). Because each SDF starts with geometry that matches the scene, it has all the details and matches the input images by default. All the network has to do is to "cut off" the scene SDF in the appropriate regions.

4. Benchmark

Previous scene decomposition techniques evaluate their methods on benchmark datasets like Replica and ScanNet. A critical limitation of these is that they do not offer complete ground truth geometries for the reconstructed objects. More specifically, per-object meshes are extracted from the full ground truth mesh of the indoor scene by cropping the ground truth mesh with the semantic masks, and therefore they lack completeness in regions occluded by other objects (Figure 6).

We introduce a new benchmark for 3D scene decomposition techniques, consisting of 32 real-world scenes and 5 synthetically generated ones. The scenes contain different combinations of objects in close contact, and we provide a high-quality complete mesh of each object.

4.1. Dataset

Real-World Scenes. We provide 22 individual 3D scanned objects and 32 scenes, each created using a combination of the individual objects. To scan the individual objects, we use Polycam [2] (for analysis of Polycam, please refer to the supplementary materials). The scenes are captured as raw images using a phone camera at a resolution of 3008 × 3008. We provide camera pose estimates from COLMAP [28], ground-truth meshes from Polycam [2], and masks generated using our mask propagation strategy. Last, we provide rotations and translations that align the ground-truth object meshes to the objects in the scenes.

Synthetic Scenes. We provide 5 synthetic scenes composed by combining objects with varying geometric complexities. We used Blender [1] 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. In addition to the multi-view images, we provide ground-truth poses, geometries, masks and transformations that align object meshes to the corresponding scene. Please refer to our supplementary material for more details on the creation of real and synthetic datasets.

4.2. Evaluation metrics

We report the **precision** and **completion ratio**. Precision is the ratio of reconstructed points that are within a distance 0.05 from the ground truth, and penalizes floaters. Completion ratio is the ratio of ground truth points that are within a distance 0.05 from the reconstruction, and penalizes incomplete reconstructions.

The **two-way Chamfer distance** is also measured between evenly-sampled vertex points on the ground truth mesh and sampled vertex points on the predicted mesh obtained by running marching cubes on the trained SDF.

To calculate these metrics between predicted and ground-truth meshes, it is crucial to maintain similar point densities to prevent imbalance. This can be difficult if the two meshes are very different in size. ObjectSDF++[32] addresses this by clipping predicted meshes using ground-truth bounds to improve density similarity and remove outliers. However, this approach can artificially inflate precision by not penalizing floaters outside the bounding box. Instead, we keep the meshes as is but propose a refinement technique that uses rejection sampling to maintain consistent point densities, adjusting for mesh saturation until the surface cannot hold more points, ensuring a fair comparison. Please see the supplement for more details.

5. Experiments

We first evaluate how our mask propagation strategy performs with increasing number of anchors and iterations. Then, we compare our full pipeline qualitatively (Fig. 8) and quantitatively (Table 2) against two baselines, ObjectSDF++ [32] and RICO [13] (Qualitative evaluations for ScanNet and

Table 1. mIOU values of the predicted masks using our mask propagation strategy, varying the number of propagation iterations and anchor images per scan. Mask quality does not improve significantly after the second round of mask propagation, and adding additional anchors offers little improvement after a few propagation iterations.

Scan		1			2			3			4			5		
No of anchor views		1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
	1	0.73	0.77	0.86	0.60	0.63	0.65	0.71	0.71	0.77	0.86	0.91	0.91	0.80	0.77	0.90
iter	2	0.90	0.90	0.91	0.64	0.64	0.64	0.78	0.78	0.78	0.91	0.92	0.91	0.94	0.94	0.94
	3	0.91	0.91	0.91	0.63	0.63	0.63	0.78	0.78	0.78	0.90	0.91	0.91	0.94	0.94	0.94

Replica dataset are in the supplementary). We benchmark these methods on the five synthetic datasets. Because ObjectSDF++ and RICO fail to produce meshes for some of the real-world scans, we evaluate on a subset of 6 real scans and 3 synthetic result for which all methods can produce valid meshes. Finally, we ablate components of our proposed method to see their impact on the quality of our solution.

5.1. Mask Propagation Evaluation

We first evaluate the performance of our segmentation propagation approach using our synthetic dataset where all scenes have corresponding ground-truth segmentation.

The first column of each scan in Table 1 shows the mIOU (Mean Intersection over Union) for each iteration, starting with one anchor image. We observe that the first iteration generally performs poorly but the mIOU improves with more iterations; however, after the second iteration, the improvement becomes minimal. Our method can also take multiple anchor images if provided; this can be useful, for instance, if all the objects are not visible in one image or the user wants to provide more information. We evaluate the effect of providing multiple anchor masks in the second and third columns of each scan. However, after the third iteration, whether we start from a single or multiple anchor masks, all converge to similar results, as shown in the third row.

A mask propagation failure is shown in scan 2 in Table 1, where the mIOU is low. This is because some parts of one object end up being labeled as another object; for example, the duck in Figure 9 (top left) is classified as the horse. Note that the same surface may be correctly labeled in a different image, since SAM is performed independently for each image.

Despite the low mIOU mask in scan 2, our object separation module still produces plausible reconstructions, because the majority of the masks are still correctly labeled. Please refer to the supplementary material for details.

5.2. Reconstruction Evaluation

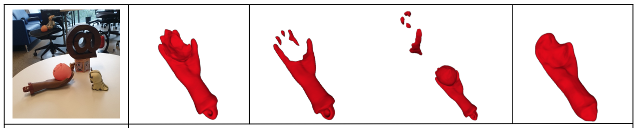
The results in Table 2 reveal the failures of the baselines. First, RICO performs poorly in the quantitative results, despite having decent qualitative results. The reason is that RICO, while often complete, produces large floaters like those seen in ‘Real scan7’ in Figure 8, which significantly hurt quantitative performance. While RICO achieves good completion metrics, it struggles to precisely generate meshes of the object of interest.

Second, ObjectSDF++, while competitive on the synthetic datasets, loses out in the real dataset benchmarks. Unlike the synthetic ground-truth masks, the masks used in the real-world benchmark were obtained using our proposed mask propagation strategy, which is still imperfect. This not only results in floaters, which are not handled due to the absence of a compactness loss, but also a loss of detail of objects at sharp edges as shown in ‘Real scan3’ and ‘Real scan16’. In contrast, we initialize the object SDF from the reconstructed scene, resulting in more robust results.

From Figure 8 and Table 2, we can conclude our method produces results with higher quality and fewer floating artifacts. Most of our quantitative improvement comes from the lack of undesired artifacts like floaters and carved holes. The remaining improvement, more evident qualitatively, comes from the scene initialization.

Table 2. **Quantitative evaluation:** RICO performs the lowest among all methods. ObjectSDF++ performs well on synthetic data, but its performance drops on real data, especially in terms of precision ratio. This drop is due to the imperfect masks in the real scans. On both synthetic and real datasets, our method outperforms the baseline in all metrics. We used GT masks for the synthetic evaluation and masks generated by our mask propagation for the real dataset.

Dataset	Metrics	RICO	ObjectSDF++	Ours
Synthetic	Chamfer ↓	0.124	0.010	**0.005**
	Prec. Ratio ↑	0.581	0.972	**0.990**
	Comp. Ratio ↑	0.938	**0.994**	0.985



Time	13.3 hrs	7 hrs	14 hrs	6 days
Loss variant	Compactness	Naïve mask	Occlusion aware mask	Compactness
Scene initialization	Yes	Yes	Yes	No

Figure 7. **Importance of the compactness loss and initialization.** Left: our compactness loss and initialization together avoids floating artifact and achieves high-quality results. Middle: a naive mask loss as in NeuS carves out objects whenever there is an occlusion and with occlusion aware mask, and we see floating artifacts in unobserved parts of the scene. Right: without scene initialization details are lost and the runtime grows significantly.

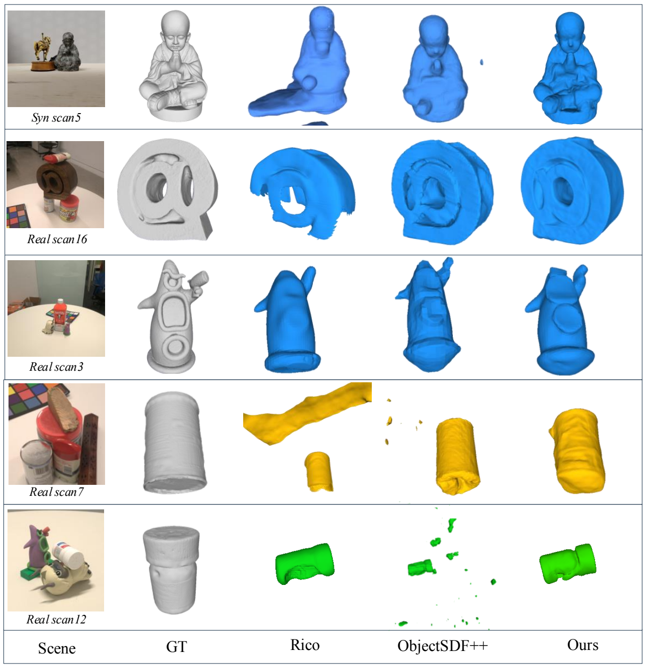


Figure 8. **Qualitative Comparison**: RICO and ObjectSDF++ produce floating artifacts, as shown in Real scans 7 and 12. RICO also sometimes carves out the object, leaving a hollow area, as shown in Real scan 16, 3 and 12. In contrast, our method produces fewer artifacts while also providing more detail.

5.3. Ablation

To understand the importance of our contributions, we ablate the proposed compactness loss and the scene initialization as shown in Figure 7.

To evaluate the compactness loss, we compare to two alternatives:

1. The first baseline is the naive mask loss used in NeuS, which does not take object occlusion into account. This loss is defined in Equation (7).
2. The second alternative without compactness is an *occlusion-aware mask loss*: we simply apply the mask loss only to the unoccluded pixels, i.e., to discount pixels that are marked as belonging to other objects.

$$\tilde{M}_k^{\text{occ}} = (\cup_{i \neq k} M_i) \tag{14}$$

$$L_{\text{occ-aware}} = \sum_k \sum_{\mathbf{j} \notin \tilde{M}_k^{\text{occ}}} \text{BCE}\left(M_k(\mathbf{j}), \hat{O}_k(\mathbf{j})\right) \tag{15}$$

While this strategy correctly avoids penalizing pixels that are part of the object but occluded, it does not encourage the object to be compact. As such, the model is free to hallucinate other floaters as long as they are completely occluded in the view.

The three loss variants are shown in the first three columns of Figure 7. When using naive mask loss, object geometries are carved out, resulting in incomplete reconstructions. This is because the model is penalized whenever it produces a surface that is occluded, evident in the hand's fingers becoming detached as a result of the object sitting on it. The occlusion-aware mask loss prevents the objects from being carved out, but introduces floaters, sometimes *inside* the other objects, which are reconstructed as hollow shells. This occurs because any floater that is completely inside the shell of another object will never be visible and therefore never be penalized. The compactness loss both removes floaters and prevents the objects from being carved out.

The last column of Figure 7 shows the reconstruction without the scene initialization. In this case, the reconstruction quality is significantly reduced and reconstruction requires a prohibitively long time. Scene initialization is critical to reducing computational time and obtaining high-quality results.

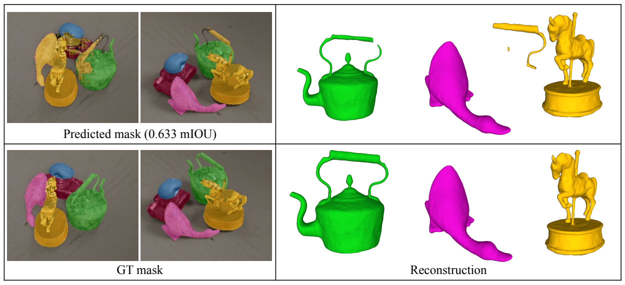


Figure 9. Top: Despite low mIOU, our object separation pipeline remains robust. This is due to our the mask labels are correct in other views. However, the reconstruction of the horse includes part of the kettle handle, as most of the mask incorrectly classifies the handle as part of the horse. Bottom: Ground truth mask and its respective reconstruction.

6. Conclusion

We proposed ObjectCarver, a method that separates objects in a scene into individual high-resolution meshes by automatically generating segmentation masks for all multi-view training images from a few clicks on just one image. We introduced compactness loss, a novel loss function that removes many of the floaters that have plagued prior methods. Finally, we show that initializing the per-object models with the scene model not only improves convergence and reduces training time but also maintains the details of the objects. While transparent and reflective objects are beyond the scope of this work, future efforts could address these by improving the scene representation.

References

[1] Blender v3.3. 6

[2] Polycam. 6

[3] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *CoRR*, abs/2103.13415, 2021. 2

[4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CoRR*, abs/2111.12077, 2021. 2

[5] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *arXiv preprint arXiv:2304.06706*, 2023. 2

[6] Toby P. Breckon and Robert B. Fisher. Amodal volume completion: 3d visual completion. *Computer Vision and Image Understanding*, 99(3):499–526, 2005. 5

[7] 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. 2, 3

[8] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields, 2023. 3

[9] 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. 3

[10] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 2, 3, 4

[11] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation, 2022. 3

[12] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. *CoRR*, abs/2011.13084, 2020. 2

[13] Zizhang Li, Xiaoyang Lyu, Yuanyuan Ding, Mengmeng Wang, Yiyi Liao, and Yong Liu. Rico: Regularizing the unobservable for indoor compositional reconstruction, 2023. 2, 3, 6

[14] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8456–8465, 2023. 2

[15] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4273–4284, 2023. 2

[16] Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *CoRR*, abs/1812.03828, 2018. 2

[17] 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. *CoRR*, abs/2003.08934, 2020. 2

[18] Tom Monnier, Jake Austin, Angjoo Kanazawa, Alexei A. Efros, and Mathieu Aubry. Differentiable blocks world: Qualitative 3d decomposition by rendering primitives, 2023. 3

[19] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *CoRR*, abs/2201.05989, 2022. 2

[20] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *CVPR*, 2021. 3

[21] Michael Niemeyer, Lars M. Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. *CoRR*, abs/1912.07372, 2019. 2

[22] Jeong Joon Park, Peter R. Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *CoRR*, abs/1901.05103, 2019. 2

[23] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Deformable neural radiance fields. *CoRR*, abs/2011.12948, 2020. 2

[24] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *CoRR*, abs/2106.13228, 2021. 2

[25] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. *CoRR*, abs/2103.13744, 2021. 2

[26] Zhongzheng Ren, Aseem Agarwala†, Bryan Russell†, Alexander G. Schwing†, and Oliver Wang†. Neural volumetric object selection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. († alphabetic ordering). 3

[27] Radu Alexandru Rosu and Sven Behnke. Permutosdf: Fast multi-view reconstruction with implicit surfaces using permutohedral lattices, 2023. 3

[28] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 6

[29] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles T. Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. *CoRR*, abs/2101.10994, 2021. 2

[30] 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, 2, 3, 5

[31] Qianyi Wu, Xian Liu, Yuedong Chen, Kejie Li, Chuanxia Zheng, Jianfei Cai, and Jianmin Zheng. Object-compositional neural implicit surfaces, 2022. 2, 3

[32] Qianyi Wu, Kaisiyuan Wang, Kejie Li, Jianmin Zheng, and Jianfei Cai. Objectsdf++: Improved object-compositional neural implicit surfaces, 2023. 2, 3, 5, 6

[33] Yinghao Xu, Menglei Chai, Zifan Shi, Sida Peng, Ivan Skorokhodov, Aliaksandr Siarohin, Ceyuan Yang, Yujun Shen, Hsin-Ying Lee, Bolei Zhou, and Sergey Tulyakov. Discoscene: Spatially disentangled generative radiance fields for controllable 3d-aware scene synthesis, 2022. 3

[34] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. *CoRR*, abs/2109.01847, 2021. 2, 3, 6

[35] Lior Yariv, Matan Atzmon, and Yaron Lipman. Universal differentiable renderer for implicit neural representations. *CoRR*, abs/2003.09852, 2020. 2

[36] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *CoRR*, abs/2106.12052, 2021. 1, 2

[37] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P Srinivasan, Richard Szeliski, Jonathan T Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–9, 2023. 2

[38] Haiyang Ying, Yixuan Yin, Jinzhi Zhang, Fan Wang, Tao Yu, Ruqi Huang, and Lu Fang. Omniseg3d: Omniversal 3d segmentation via hierarchical contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20612–20622, 2024. 3

[39] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. *CoRR*, abs/2103.14024, 2021. 2

[40] Hong-Xing Yu, Leonidas J. Guibas, and Jiajun Wu. Unsupervised discovery of object radiance fields. In *ICLR*, 2022. 3

[41] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction, 2022. 2

[42] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2