DiscoNeRF: Class-Agnostic Object Field for 3D Object Discovery

Corentin Dumery   Aoxiang Fan   Ren Li   Nicolas Talabot   Pascal Fua

CVLab, EPFL

Abstract

Neural Radiance Fields (NeRFs) have become a powerful tool for modeling 3D scenes from multiple images. However, NeRFs remain difficult to segment into semantically meaningful regions. Previous approaches to 3D segmentation of NeRFs either require user interaction to isolate a single object, or they rely on 2D semantic masks with a limited number of classes for supervision. As a consequence, they generalize poorly to class-agnostic masks automatically generated in real scenes. This is attributable to the ambiguity arising from zero-shot segmentation, yielding inconsistent masks across views.

In contrast, we propose a method that is robust to inconsistent segmentations and successfully decomposes the scene into a set of objects of any class. By introducing a limited number of competing object slots against which masks are matched, a meaningful object representation emerges that best explains the 2D supervision and minimizes an additional regularization term. Our experiments demonstrate the ability of our method to generate 3D panoptic segmentations on complex scenes, and extract high-quality 3D assets from NeRFs that can then be used in virtual 3D environments.

1. Introduction

Neural Radiance Fields (NeRFs) [29] have been proposed for novel view synthesis from multiple images and have now become a powerful 3D scene reconstruction tool. One limitation, however, is that they represent the scene geometry in terms of a single 3D density field in which all objects are blended. This makes downstream tasks such as editing or object identification particularly challenging. While there has been work on assigning semantic labels to parts of scenes described in this manner [26, 39, 44], they are limited to classes seen during training. Such classes usually include easily identifiable household items in interior settings and vehicles in street scenes for which labeled data is abundant. Unfortunately, this approach does not scale well to in-the-wild scenes with previously unseen objects.

Panoptic Lifting [37] and Instance-NeRF [26] consistently miss objects that are not in a predefined set of classes. When given class-agnostic masks as input, we observed that these methods still fail, as they are unable to deal with the inconsistencies that arise from zero-shot segmentation, as illustrated by Fig. 1(a). Some methods attempt to overcome this limitation by directly learning 3D features instead of labels. For example, DFF [21] and LERF [18] achieve this by distilling 2D features from Dino [4] and CLIP [36] respectively. However, the resulting segmentations tend to be fuzzy and are unable to separate different instances of the same object type. In another line of works, Garfield [19] and Contrastive Lift [3] aim to learn a continuous 3D feature field from 2D masks, which must then be clustered as a post-processing step in order to identify objects. This additional step requires additional hyperparameter tuning and tends to produce imprecise segmentations.

Finally, SA3D [5] and SANeRF-HQ [27] rely on user-interaction to segment a single object of interest in a scene. They are unable to produce panoptic segmentations over the complete 3D scene, as they simply learn a binary field around the user input. This proves to be a significantly simpler task, as real scenes can incorporate hundreds of objects and matching inconsistent masks across views is highly challenging.

In this work, we propose *DiscoNeRF*, an approach to segmenting 3D NeRFs in-the-wild without a preset number of classes or user-interaction while still producing sharp segmentations, as illustrated by Fig. 1(b). We start from the observation that foundation models such as SAM [20] or Mask-DINO [23] can assign labels to pixels in 2D images of generic scenes containing an arbitrary number of objects at the cost of these labels being potentially inconsistent across different views of the same scene. Our key insight is that we can efficiently constrain the training of an object field to produce a view-consistent 3D segmentation despite the noisiness of the supervisory signal. This enables us to produce sharp segmentations that are consistent across views, unlike those of the original segmentation algorithm, and for objects belonging to arbitrary classes. These images can then be compared to the 2D segmentation masks returned

Figure 1. **Problem statement.** (a) Given as input a set of class-agnostic 2D masks [20] with little consistency across views, we aim to learn (b) a meaningful 3D object field that segments the different instances in the scene. The discovered objects can then be extracted and rendered independently.

by the foundation model [20].

Fig. 2 depicts our DiscoNeRF architecture. In addition to the density and color that a standard NeRF predicts at each point in a volume, ours predicts a vector of probabilities of belonging to a particular class. These vectors are then used to render images whose pixels are the probabilities to belong to each class. Note that these object classes are not assigned a semantic meaning, they simply correspond to a coordinate number in the probability vector. To enforce consistency across views between probability images and 2D segmentation masks, we introduce a robust loss function that generalizes the traditional Intersection over Union (IoU) measure to floating point values and relies on the Hungarian algorithm to establish correspondences between regions in the probability images and 2D masks in the segmentations. Minimizing this loss function using the appropriate regularization constraints yields the desired result.

Our contribution is therefore an approach to reconstructing 3D scenes in such a way that objects of *any* class, as opposed to those from a restricted set of classes seen during training, can be properly segmented. This allows a wide range of additional applications of NeRF scenes, and the identified objects can be used for scene editing or provide spatial understanding to an autonomous system. To ensure reproducibility and encourage future works, we will release our implementation and model weights.

2. Related Work

In this section, we first briefly introduce the original NeRF formulation [29] and then discuss the various approaches to semantic segmentation of a NeRF style representation.

2.1. Neural Radiance Fields (NeRFs)

Given a set of *posed* images of a scene, NeRFs aim to generate new images of that scene from novel poses. This is done by learning a 3D scene representation in terms of a *radiance field*, which can be used to render the original images. More specifically, a NeRF model assigns a density $\sigma(\mathbf{x})$ and a view-dependent color $\mathbf{c}(\mathbf{x}, \mathbf{d})$ to each 3D point $\mathbf{x}$, where $\mathbf{d}$ is the direction from which $\mathbf{x}$ is seen. The density can be interpreted as the differential probability of a ray terminating at an infinitesimal particle at location $\mathbf{x}$. Given a pixel in an image and the line of sight $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ for t between the near and far bounds t_n and t_f going through it, the expected color of that pixel is

$$\hat{C}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\, \sigma(\mathbf{r}(t))\, \mathbf{c}(\mathbf{r}(t), \mathbf{d})\, dt$$
$$\text{where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right). \tag{1}$$

The function T denotes the accumulated transmittance along the ray from t_n to t, that is the probability that the ray travels from t_n to t without hitting any other particle. In practice, one uses a multi-layer perceptron with weights Θ to define a function f_Θ that assigns to each $\mathbf{x}$ and $\mathbf{d}$ a pair $f_\Theta(\mathbf{x}, \mathbf{d}) = (\sigma(\mathbf{x}), \mathbf{c}(\mathbf{x}, \mathbf{d}))$. Note that f is formulated such that the density σ does not depend on the view direction $\mathbf{d}$.

The weights Θ are adjusted so that the images generated by this model are as similar as possible to the input images of the scene. This is done by minimizing a Mean Squared Error loss of the form

$$\mathcal{L}_{RGB} = \sum\sum \|\hat{C} - C\|_2^2, \tag{2}$$

where $\hat{C}$ is the rendered color of Eq. (1), C is the actual color in the images, and the double summation is over all the pixels in all the input images. In practice, the loss is computed over batches of rays randomly sampled from all views.

While the original NeRF implementation suffered from drawbacks in speed and accuracy, recent works have significantly mitigated these issues. These developments make
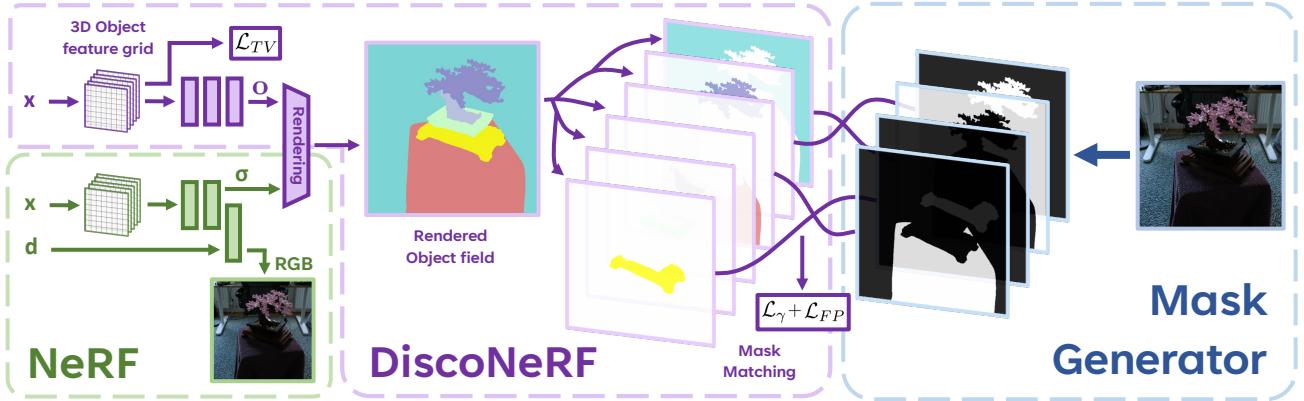
Figure 2. **DiscoNeRF pipeline.** We add to a standard NeRF (green) an object network (violet) that predicts probabilities of belonging to each object. These are used to render 2D probability images, which are compared to the segmentation masks (blue) produced by a foundation model [20] and made as consistent as possible with them. Crucially, this is done *without* assigning explicit semantic labels, which improves the generalization abilities of our approach.

NeRFs an attractive representation for 3D segmentation. A promising line of works improved the training speed of NeRFs by replacing the neural scene representation. TensoRF [6] proposes a reprensentation based on tensor-decomposition to compactly encode scene properties. More recently, InstantNGP [31] has introduced multi-resolution hash encoding to accomplish both efficient and high-fidelity scene modeling, providing a flexible and versatile tool for downstream applications. MipNeRF [1] further improves the view synthesis quality of NeRF by representing the scene at a continuously-valued scale and efficiently rendering anti-aliased conical frustums instead of rays. MipN-eRF360 [2] and NeRF++[43] propose to use different re-parametrization techniques to better model general and un-bounded scenes. These recent developments have made it possible to reconstruct complex scenes with high visual accuracy. However, the reconstructions remain fully fused in a single neural network and thus unusable for many down-stream applications, such as 3D asset generation or editing.

2.2. Segmentation of NeRF Models

After training, the weights of the network that implements f_Θ jointly capture all the scene information, including light-ing and color. However, they do not separate the different 3D objects in the scene and doing so is far from trivial. Recently, it has been proposed to interactively extract ob-jects given user-defined prompt points identifying a target in some of the training images [5] or directly in 3D [8]. The manually provided inputs eliminate much of the ambiguity arising from zero-shot segmentation, resulting in a signifi-cantly simpler task. In addition to requiring human interac-tion, this approach can only identify one object at a time. In contrast, we propose an automated method that decomposes the whole scene into a set of objects without any additional human supervision.

In a separate line of work, some methods learn a contin-uous field of 3D features that allows clustering of different objects in the scene [3, 13, 15, 19, 25]. This representa-tion does not define a specific number of objects and is in-trinsically continuous, making it a poor fit for segmenting NeRFs into discrete components. Similarly, *feature distil-lation* [21] can be used to learn a 3D field that matches the 2D features from various foundation models. This has been used by recent work to perform text-queries [18] or interac-tive segmentation [8].

Some previous work demonstrated the potential of NeRFs for 3D semantic modeling [10, 14, 26, 37, 39, 42]. Some of these algorithms [26, 39] query the NeRF model on a sparse grid, and then apply a point-cloud segmentation method based on a 3D U-Net [11]. The generality of these approaches is inherently limited by the pre-defined classes they are trained to identify and they generalize poorly to zero-shot classes.

In a different but related line of work, some meth-ods [12, 28, 32, 35, 41] have been proposed to segment 3D gaussians [17] by assigning labels to individual splats. These approaches do not define a 3D field, unlike NeRF segmentation methods, and due to their very different scene representation it is not straightforward to adapt them to seg-ment scenes reconstructed with NeRFs.

3. Approach

In NeRFs, each point in space is assigned a color and a den-sity, as discussed in Sec. 2.1. Our goal is to also assign a label, which should be the same for all points belonging to the same object. In earlier work [26, 37], this has been tried by making the function f_Θ introduced in Sec. 2.1 to output not only a density and a color but also an object label. This

3

additional output can then be supervised using 2D semantic masks extracted from the ground truth images. However, masks of in-the-wild objects from previously unseen classes are often unavailable or inconsistent across views, making the training of the extended f_Θ difficult.

In this work we expand on the idea of [26, 37] to make the training more robust to these inconsistencies. We write the output of our function f_Θ as

$$f(\mathbf{x}, \mathbf{d}) = (\sigma(\mathbf{x}), \mathbf{c}(\mathbf{x}, \mathbf{d}), \mathbf{o}(\mathbf{x})) , \quad (3)$$

where $\mathbf{o}$ is an N-dimensional vector, which we refer to as the *object probability vector*, and N is the maximum number of objects that can be detected. The i-th coordinate of $\mathbf{o}$ is interpreted as the probability of point $\mathbf{x}$ being part of the i-th object. Given $\mathbf{x}$ and $\mathbf{d}$, it computes σ and $\mathbf{c}$ in the standard NeRF manner. $\mathbf{o}$ is predicted by a separate *object network*, consisting of a small MLP and a *softmax* layer. Our approach is depicted by Fig. 2 and we describe its individual components below.

First, the input $\mathbf{x}$ is encoded by interpolating features from a learned 3D hash grid [31]. Then the encoded $\mathbf{x}$ is fed to the object decoder. This produces a 3D object field $\mathbf{o}(\mathbf{x})$, which can then be rendered to 2D using $\sigma(\mathbf{x})$ and Eq. (1). Note that, by design, $\mathbf{o}$ takes $\mathbf{x}$ as its sole input because a semantic label should be independent of the viewpoint.

3.1. Loss Function

To train our object network, we need to design a loss function that is minimized when the binary masks generated from probabilities in the vectors $\mathbf{o}$ are consistent with the 2D segmentation masks generated by the foundation model. The main difficulty being that there is *a priori* no correspondence between object labels and segmentation masks in individual images.

To this end, let us consider one specific image in which the foundation model generates K binary masks $(M_m)_{1 \le m \le K}$. In our experiments, these masks are generated by the *Segment Anything Model* [20]. Recall that each one of the N channels of $\mathbf{o}$ represents the probability of belonging to a specific object. We will refer to them as *object slots*. By using the volume rendering scheme of Eq. (1), we can generate a probability image O, in which $O_{n,i,j}$ denotes the probability of pixel (i, j) belonging to object n. Since no correspondence between the m and n indices is given, we need to establish them by comparing the M_m and O_n masks. To this end, we first compute the affinity α between M_m and O_n as

$$\alpha(O_n, M_m) = \frac{\sum\limits_{i,j} \mathbf{min}(O_{n,i,j}, M_{m,i,j})}{\sum\limits_{i,j} \mathbf{max}(O_{n,i,j}, M_{m,i,j})} . \quad (4)$$

Given that $M_{m,i,j}$ is either 0 or 1 and $0 \le O_{n,i,j} \le 1$, this definition can be understood as generalizing the Intersection over Union (IoU) score to floating point values, as illustrated by Fig. 3. Finally, we compute a matching between masks and object slots that maximizes affinity using the Hungarian algorithm [22]. We set the maximum number of object slots N to be at least as large as the number of masks K from any given view, such that all masks get assigned an object slot while some slots can be left unassigned. Note that the computation of the affinity α and the matching of masks with object slots are performed with a stop-gradient to prevent this association from interfering with the learning of the object field.

Let us denote the output of the Hungarian matching as γ. We define the two losses

$$\mathcal{L}_\gamma = \frac{1}{K} \sum_{m=1}^K \|M_m - O_{\gamma(m)}\|_2^2 , \quad (5)$$

$$\mathcal{L}_{FP} = \frac{1}{K} \sum_{m=1}^K \sum_{n \ne \gamma(m)} \|M_m * O_n\|_2^2 , \quad (6)$$

where $*$ denotes the pixel-wise product. Minimizing the *Matching* loss $\mathcal{L}_\gamma$ encourages the assigned channel to match its corresponding mask, while minimizing the *False Positive* loss $\mathcal{L}_{FP}$ penalizes all other object slots that are not zero in the masked area. We compute a matching γ for each image, and average $\mathcal{L}_\gamma$ and $\mathcal{L}_{FP}$ over all images in a given training batch.

Intuitively, the matching γ identifies the most-likely object slot for each input mask. These masks sometimes suffer from *over-segmentation* arising from zero-shot segmentation. This effect is mitigated by our object field $\mathbf{o}$, which computes probabilities with a *softmax* layer such that different slots are competing for each pixel and two overlapping representations cannot coexist. Consequently, over-segmented masks will tend to be ignored as masks that are present more often in the supervision signal will spontaneously be preferred during optimization. To further improve the consistency of our segmentation, we introduce an additional regularization term in the following section.

3.2. Field Regularization

As demonstrated in our experiments, minimizing $\mathcal{L}_\gamma$ and $\mathcal{L}_{FP}$ alone produces noisy objects fields. This is expected, as the supervision signal can be highly inconsistent across views. To regularize learning, we implement the object field $\mathbf{o}$ as a 3D hash grid of learnable features [31]. To query $\mathbf{o}(\mathbf{x})$, the 3D point $\mathbf{x}$ is first encoded by interpolating the hash grid features, then these features are decoded by a fully-connected network. Unlike in earlier approaches [26, 37], this modeling allows us to regularize the object field by smoothing the learned features. Specifically,
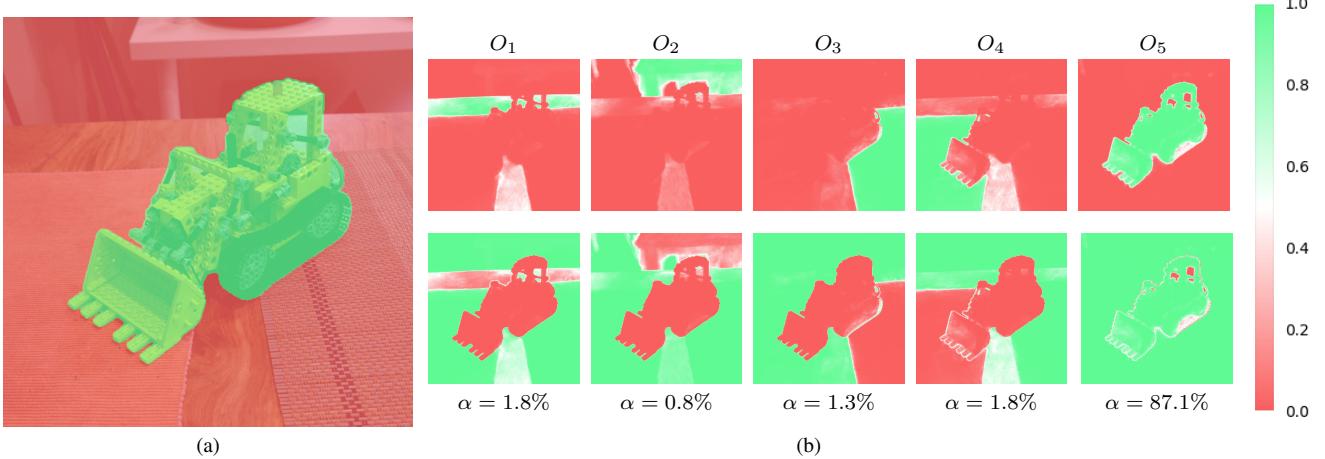
4

Figure 3. Our affinity function α considers (a) a generated mask M_m and (b) all object field channels $(O_n)_{1 \leq n \leq N}$ independently. Bottom row: visualization of $1 - |M_m - O_n|$.

we regularize $\mathbf{o}(\mathbf{x})$ by introducing an additional total variation loss $\mathcal{L}_{TV}$ taken to be

$$\mathcal{L}_{TV} = \sum_{i,j \in \mathcal{N}} \|h_i - h_j\|_2^2 \tag{7}$$

where h_i is the learned features for the i-th voxel in the grid, and $\mathcal{N}$ defines the neighborhood of adjacent voxels in the grid. We found this additional regularization to be essential to handle the ambiguities of zero-shot 2D segmentations, as we show below in an ablation study. In the presence of conflicting 2D masks such as the ones in Fig. 1, a naive implementation will attempt to fit all masks in the training set, resulting in noisy 3D segmentations. By introducing an additional TV regularizer, our method encourages the emergence of a consistent 3D segmentation. Even if the introduction of this additional loss term can in rare cases lead to overly smooth segmentations, we observed that it improved results both quantitatively and qualitatively.

3.3. Training

Given a set of posed images as input, we first train a NeRF model to reconstruct the scene by sampling random rays across different views. We use recent techniques that help make NeRF reconstruction magnitudes faster and more accurate. We train a proposal network [2] to improve the ray sampling strategy, and employ hash grid features [31] to speed up training. We exploit the fact that a coarse-to-fine activation of hash grids [24] improves the robustness of reconstruction at a minimal cost in training speed.

Then, we freeze the weights of the NeRF model and train the object network. Unlike in the previous step, this optimization is performed by sampling random views and considering all the masks associated to these views jointly. We

then minimize $\mathcal{L}_\gamma + \lambda_{FP}\mathcal{L}_{FP} + \lambda_{TV}\mathcal{L}_{TV}$, where λ_{FP} and λ_{TV} are taken to be 0.01 unless explicitly stated otherwise.

4. Experiments

Our implementation is based on *torch-ngp* [38], which incorporates the speedup of Instant-NGP [31]. To generate the binary segmentation masks of M_m of Sec. 3, we use the Segment Anything Model (SAM) [20]. For object-centric scenes, we generate an additional background mask by extracting the pixels that lie outside of the unit sphere in 3D using the reconstructed NeRF.

We use the same loss weights of $\lambda_{FP} = 0.01$ and $\lambda_{TV} = 0.01$ for all scenes and datasets. Finally, in all experiments, we automatically set $N = 2 \times K_{max}$, where K_{max} is the number of masks in the view with the most masks produced. In practice, we found this to be high enough to capture each object in all tested scenes.

4.1. Comparative Evaluation

We evaluate our method quantitatively on the Mip-NeRF 360 dataset [2], which features both indoor and outdoor scenes containing a wide variety of common and less common objects. To this end, we render our object field to an image and select the argmax per pixel to produce discrete segmentation masks. We then compare them against hand-generated ground-truth masks for each scene except the *stump* scene, which has no object and was therefore excluded from our experiments. To ensure the reproducibility of our experiments, we will publish these masks and all the results presented along with our implementation.

Metrics. We report the 2D IoU between ground-truth masks and masks produced by each method. We match each ground truth mask with one of the produced masks using the

5

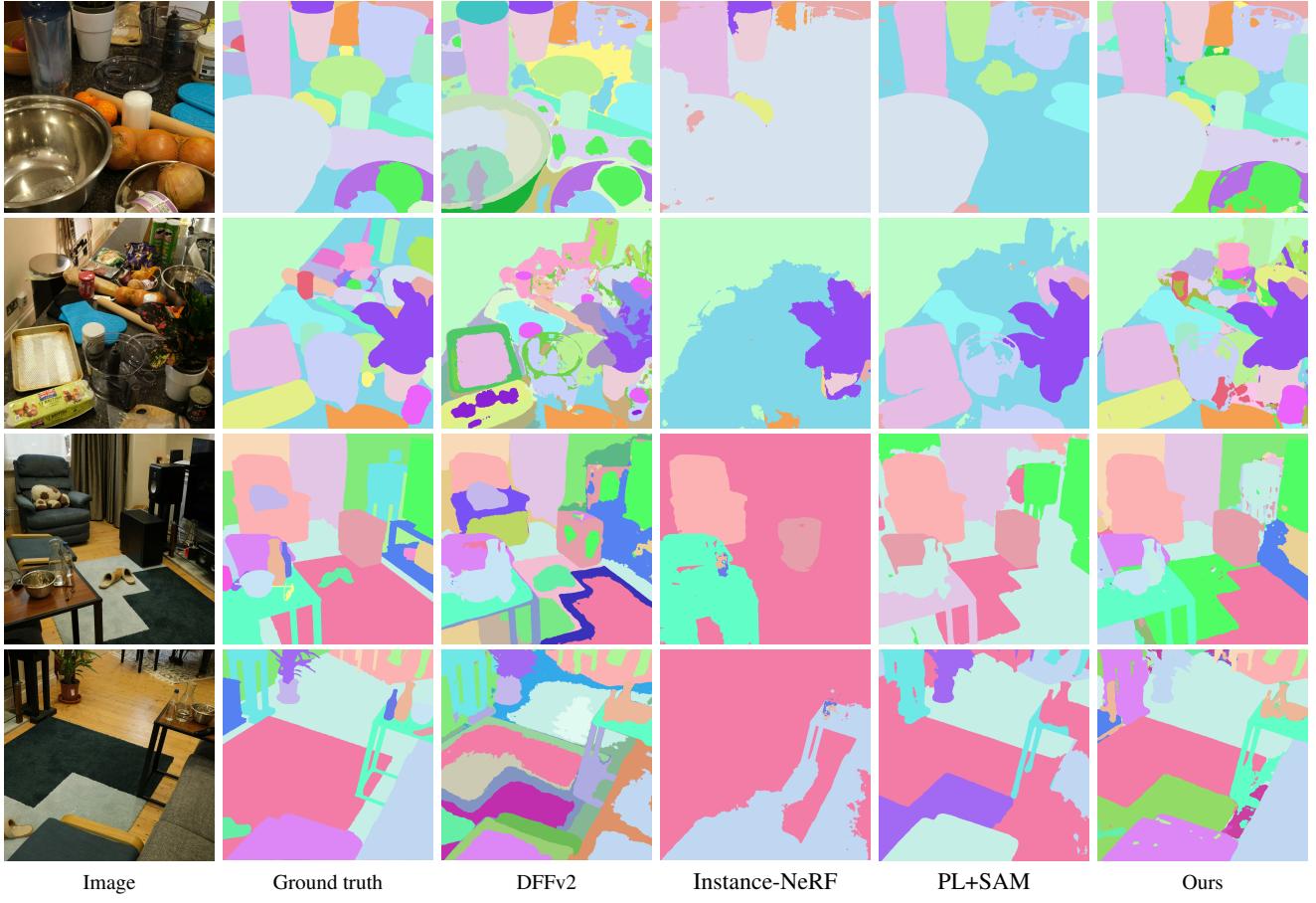| Image | Ground truth | DFFv2 | Instance-NeRF | PL+SAM | Ours |

Figure 4. Comparing 3D NeRF decomposition models for the scenes of *counter* and *room*.

IoU, and compute a weighted average based on the area in order to prevent small masks from skewing the metric. We additionally report the *Best Dice* (BD) score between the ground truth and the masks produced by our method. For two sets of masks $\mathcal{P}$ and $\mathcal{G}$, it is defined as

$$BD(\mathcal{P}, \mathcal{G}) = \frac{1}{N} \sum_{i=1}^{N} \max_{j=1:M} \frac{2|p_i \cap g_j|}{|p_i| + |g_j|} , \qquad (8)$$

along with its symmetric counterpart $SBD(\mathcal{P}, \mathcal{G}) = \min\{BD(\mathcal{P}, \mathcal{G}), BD(\mathcal{G}, \mathcal{P})\}$. An in-depth analysis of these two metrics can be found in [7].

Baselines. For comparison purposes, we implement a modernized version of Distilled Feature Fields (DFF) [21], which jointly learns a NeRF model and a 3D field replicating the features produced by a foundation model in a process called *feature distillation*. Specifically, we implement DFF with distillation of the recent DINOv2 features [33], as opposed to DINO features in the original method [4]. We will refer to this method as *DFFv2*. To make the distillation tractable, we first reduce the dimensionality of DINOv2 fea-

tures to 64 with principal component analysis. We then produce 2D segmentations by rendering the distilled features and clustering them.

We also compare our method against Panoptic Lifting [37] and Instance-NeRF [26], two recent methods for semantic segmentation of NeRF models that rely on 2D and 3D segmentation models pretrained on specific object classes. Panoptic Lifting [37] jointly trains a NeRF and a multi-layer perceptron to match 2D semantic and instance maps. Additionally, we modified Panoptic Lifting to use masks produced by SAM. We refer to this method as PL+SAM. Note that the 2D prediction confidence and the semantic losses $\mathcal{L}_{sem}$ and $\mathcal{L}_{con}$ of the original paper cannot be used in this context due to the absence of ground-truth semantic classes. Instance-NeRF [26] fuses inconsistent 2D segmentations by querying a NeRF on a 3D grid and applying a point-cloud segmentation model. Combining Instance-NeRF with SAM does not improve its performance, as it does not only rely on semantic 2D masks but also on a semantic 3D point-cloud segmentation network.
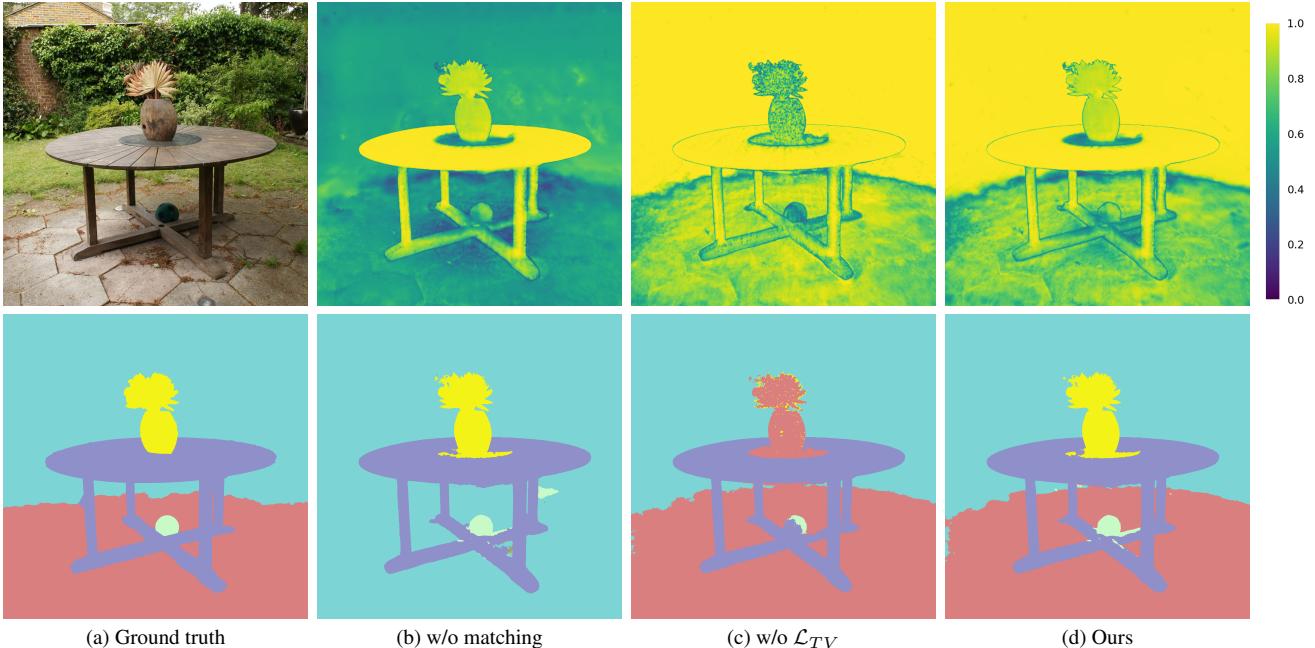
Figure 5. **Ablation.** Top row: the *prediction confidence* is defined as the maximum value across object slots. Without matching, objects are often missed or captured within the same slots. Without $\mathcal{L}_{TV}$, the segmentation is not spatially consistent and its confidence is noisy.

	DFFv2	Instance-NeRF	Panoptic Lifting	PL+SAM	SAM (2D)	Ours
IoU (%)	65.60	31.71	29.33	64.12	76.14	**79.24**
BD (%)	74.57	53.75	55.41	78.32	87.12	**87.51**
SBD (%)	73.82	44.19	41.73	72.90	83.39	**85.34**

Table 1. Segmentation results on the Mip-NeRF 360 [2] dataset.

Results. We provide a qualitative comparison in Fig. 4 and a quantitative one in Tab. 1. Our approach outperforms the baselines by a large margin in all the metrics. In particular, Instance-NeRF fails when evaluated on in-the-wild scenes, which are more visually complex than its training data and filled with previously unseen objects. The 3D point cloud segmentation network used by Instance-NeRF often misses or wrongly merges different objects, leading to simplistic segmentations.

Similarly, Panoptic Lifting relies on a semantic panoptic segmentation of the scene [9] and fails to properly segment novel objects for in-the-wild scenes. The results of our implementation of PL+SAM, which uses the same masks as our method as supervision, show that Panoptic Lifting is unable to derive a meaningful 3D segmentation from class-agnostic masks. In complex scenes, several objects are wrongfully clustered together or associated with the background. In contrast, our method is fully class-agnostic and can be applied to scenes with zero-shot objects.

Additionally, we evaluate the masks produced by SAM from a single 2D image in Tab. 1. As shown in Fig. 1(a),

these masks are often noisy and fail to capture all objects. We observe that our method successfully learns from multi-view masks and outperforms the single-view supervision.

4.2. Ablation study

To validate the impact of the different components of our method, we perform an ablation study. We remove the matching γ and instead match masks and objects by choosing the slot that maximizes the affinity α independently from each other, such that the same object slot can be assigned to several masks. This leads to a sharp drop in our evaluation metrics from 79.23% to 57.04% IoU, as the same object slot often captures several objects in Fig. 5(b).

We also ablate the regularization loss $\mathcal{L}_{TV}$ and compare the produced segmentations in Fig. 5(c). We observe a drop in IoU down to 75.53% that can be attributed to the noisy object field learned without our additional regularization. Interestingly, we also observed in our experiments that the addition of $\mathcal{L}_{TV}$ helped prevent different objects from using the same object slot. We view this as a consequence of our regularization, which prevents the optimization from
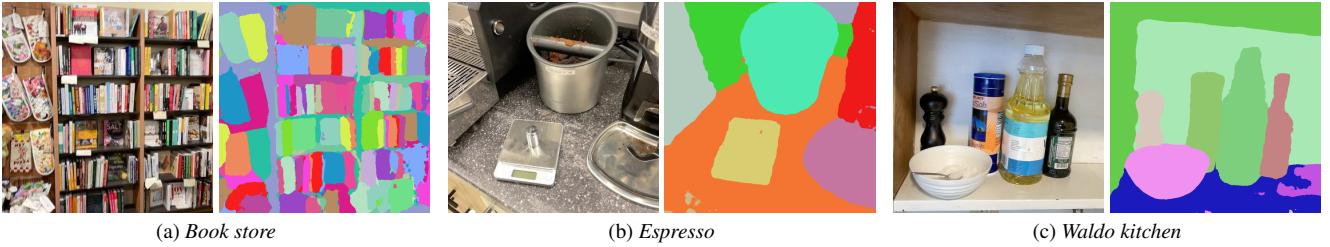
(a) *Book store*

(b) *Espresso*

(c) *Waldo kitchen*

Figure 6. Additional results on the LERF [18] dataset.

(a) Object removal

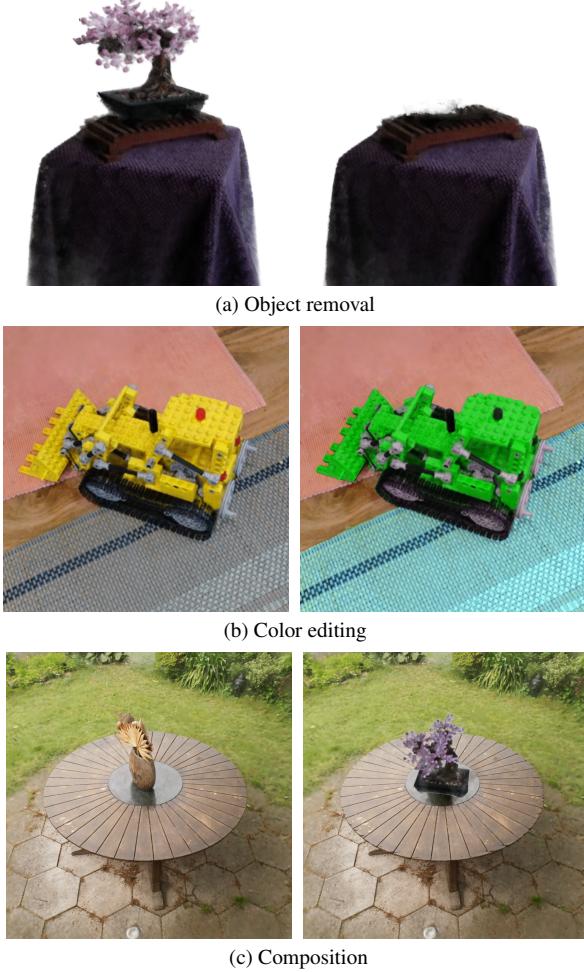(b) Color editing

(c) Composition

Figure 7. **Applications.** Given some images (*left*), we perform simple editing operations to synthesize new images (*right*) by conditioning NeRF rendering with the object field value.

getting trapped in a local minimum. We further observe that without $\mathcal{L}_{TV}$, the predicted confidence remains low and noisy, leading to fuzzy segmentations in Fig. 5(c) and making 3D object extraction less robust. In contrast, our method successfully identifies the different objects in the scene in separate object slots.

4.3. Applications

Since our method can segment the 3D scenes into individual objects, the 3D reconstructions can be used in downstream applications. An object can be extracted by conditioning the rendering of the model in Eq. (1) to only render the associated object slot. This lets a user perform editing operations conditioned on the object slots. As shown in Fig. 7(a), we can remove the bonsai from the reconstructed scene. This operation reveals uncovered geometry under the removed object, which can be fuzzy as it was not supervised during NeRF training. This can easily be enhanced in postprocessing by inpainting [30, 40]. Additionally, in Fig. 7(b), we identify objects and change their colors independently. We can also assemble objects from several scenes and render them jointly, as illustrated by Fig. 7(c).

4.4. Additional results on the LERF [18] dataset

We present 3D segmentation results for scenes from the LERF [18] dataset in Fig. 6. We further observe that our method generalizes well to the complex scenes in that dataset. In some cases, high ambiguity in the mask generation can lead to artifacts or fuzzy segmentations, as can be seen at the bottom right of the *Waldo kitchen* scene.

5. Conclusion

We have shown that we can successfully learn a meaningful 3D object field from inconsistent 2D masks. We match the channels of our object field with the supervision masks, and regularize the 3D field to produce a spatially consistent 3D segmentation. This is done without requiring semantic labels, which would limit the generality of our approach.

One current limitation is that non contiguous objects may, in rare occasions, be assigned the same label. This is mitigated by our regularization and in practice, such objects can still trivially be separated. Another limitation is that we can currently only handle static scenes. Future work will focus on dynamic NeRFs [34] and enforcing consistency of the segmentations over time. Finally, while our current approach relies on NeRF rendering, it would be interesting to extend it to other representations such as 3D gaussians [17] or physically-based differentiable rendering [16].

References

[1] J. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *International Conference on Computer Vision*, pages 5855–5864, 2021. 3

[2] J. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *Conference on Computer Vision and Pattern Recognition*, 2022. 3, 5, 7, 1

[3] Y. Bhalgat, I. Laina, J. Henriques, A. Zisserman, and A. Vedaldi. Contrastive lift: 3d object instance segmentation by slow-fast contrastive fusion. In *Advances in Neural Information Processing Systems*, 2023. 1, 3

[4] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging Properties in Self-Supervised Vision Transformers. In *International Conference on Computer Vision*, 2021. 1, 6

[5] J. Cen, Z. Zhou, J. Fang, C. Yang, W. Shen, L. Xie, X. Zhang, and Q. Tian. Segment anything in 3d with nerfs. In *Advances in Neural Information Processing Systems*, 2023. 1, 3

[6] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, pages 333–350. Springer, 2022. 3

[7] L. Chen, Y. Wu, J. Stegmaier, and D. Merhof. Sortedap: Rethinking evaluation metrics for instance segmentation. In *International Conference on Computer Vision*, pages 3923–3929, 2023. 6

[8] X. Chen, J. Tang, D. Wan, J. Wang, and G. Zeng. Interactive Segment Anything NeRF with Feature Imitation. *arXiv Preprint*, 2023. 3

[9] B. Cheng, I. Misra, A. Schwing, A. Kirillov, and R. Girdhar. Masked-attention mask transformer for universal image segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 1290–1299, 2022. 7

[10] X. Cheng, Y. Wu, M. Jia, Q. Wang, and J. Zhang. Panoptic compositional feature field for editable scene rendering with network-inferred labels via metric learning. In *Conference on Computer Vision and Pattern Recognition*, pages 4947–4957, 2023. 3

[11] Ö. Çiçek, A. Abdulkadir, S. Lienkamp, T. Brox, and O. Ronneberger. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In *Conference on Medical Image Computing and Computer Assisted Intervention*, pages 424–432, 2016. 3

[12] B. Dou, T. Zhang, Y. Ma, Z. Wang, and Z. Yuan. Cosseg-gaussians: Compact and swift scene segmenting 3d gaussians with dual feature fusion. *arXiv Preprint*, 2024. 3

[13] F. Engelmann, F. Manhardt, M. Niemeyer, K. Tateno, and F. Tombari. Opennerf: Open set 3d neural scene segmentation with pixel-wise features and rendered novel views. In *International Conference on Learning Representations*, 2024. 3

[14] X. Fu, S. Zhang, T. Chen, Y. Lu, L. Zhu, X. Zhou, A. Geiger, and Y. Liao. Panoptic nerf: 3d-to-2d label transfer for panoptic urban scene segmentation. In *International Conference on 3D Vision*, 2022. 3

[15] H. He, C. Stearns, A. Harley, and L. Guibas. View-consistent hierarchical 3d segmentationusing ultrametric feature fields, 2024. 3

[16] W. Jakob, S. Speierer, N. Roussel, M. Nimier-David, D. Vicini, T. Zeltner, B. Nicolet, M. Crespo, V. Leroy, and Z. Zhang. Mitsuba 3 renderer, 2022. https://mitsuba-renderer.org. 8

[17] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 3, 8

[18] J. Kerr, C. Kim, K. Goldberg, A. Kanazawa, and M. Tancik. LERF: Language Embedded Radiance Fields. In *International Conference on Computer Vision*, 2023. 1, 3, 8

[19] Chung-Min* Kim, Mingxuan* Wu, Justin* Kerr, Matthew Tancik, Ken Goldberg, and Angjoo Kanazawa. GARField: Group Anything with Radiance Fields. In *arXiv Preprint*, 2024. 1, 3

[20] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment Anything. In *arXiv Preprint*, 2023. 1, 2, 3, 4, 5

[21] S. Kobayashi, E. Matsumoto, and V. Sitzmann. Decomposing nerf for editing via feature field distillation. In *Advances in Neural Information Processing Systems*, 2022. 1, 3, 6

[22] H.W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 1955. 4

[23] F. Li, H. Zhang, H. Xu, S. Liu, L. Zhang, L. M. Ni, and H.-Y. Shum. Mask Dino: Towards a Unified Transformer-Based Framework for Object Detection and Segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2023. 1

[24] Z. Li, T. Müller, A. Evans, R. Taylor, M. Unberath, M. Liu, and C. Lin. Neuralangelo: High-Fidelity Neural Surface Reconstruction. In *Conference on Computer Vision and Pattern Recognition*, 2023. 5

[25] K. Liu, F. Zhan, J. Zhang, M. Xu, Y. Yu, A. Saddik, C. Theobalt, E. Xing, and S. Lu. Weakly supervised 3d open-vocabulary segmentation. *arXiv Preprint*, 2023. 3

[26] Y. Liu, B. Hu, J. Huang, Y.-W. Tai, and C.-K. Tang. Instance Neural Radiance Field. In *International Conference on Computer Vision*, 2023. 1, 3, 4, 6

[27] Y. Liu, B. Hu, C. Tang, and Y. Tai. Sanerf-hq: Segment anything for nerf in high quality. *arXiv Preprint*, 2023. 1

[28] W. Lyu, X. Li, A. Kundu, Y. Tsai, and M. Yang. Gaga: Group any gaussians via 3d-aware memory bank. *arXiv Preprint*, 2024. 3

[29] Ben Mildenhall, S. P. P., M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *European Conference on Computer Vision*, 2020. 1, 2

[30] A. Mirzaei, T. Aumentado-Armstrong, K. Derpanis, J. Kelly, M. Brubaker, I. Gilitschenski, and A. Levinshtein. SPIn-NeRF: Multiview segmentation and perceptual inpainting with neural radiance fields. In *Conference on Computer Vision and Pattern Recognition*, 2023. 8

[31] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv Preprint*, 2022. 3, 4, 5

[32] V. Nguyen, E. Sandidge, T. Mahendrakar, and R. White. Satsplatyolo: 3d gaussian splatting-based virtual object detection ensembles for satellite feature recognition. *arXiv Preprint*, 2024. 3

[33] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, R. Howes, P. Huang, H. Xu, V. Sharma, S. Li, W. Galuba, M. Rabbat, M. Assran, N. Ballas, G. Synnaeve, I. Misra, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. DINOv2: Learning Robust Visual Features without Supervision. *arXiv Preprint*, 2023. 6

[34] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Conference on Computer Vision and Pattern Recognition*, 2020. 8

[35] M. Qin, W. Li, J. Zhou, H. Wang, and H. Pfister. Langsplat: 3d language gaussian splatting. In *Conference on Computer Vision and Pattern Recognition*, 2024. 3

[36] A. Radford, J. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*. PMLR, 2021. 1

[37] Y. Siddiqui, L. Porzi, S. Bulò, N. Müller, M. Nießner, A. Dai, and P. Kontschieder. Panoptic lifting for 3d scene understanding with neural fields. In *Conference on Computer Vision and Pattern Recognition*, pages 9043–9052, 2023. 1, 3, 4, 6

[38] J. Tang. Torch-ngp: a pytorch implementation of instant-ngp, 2022. https://github.com/ashawkey/torch-ngp. 5

[39] S. Vora, N. Radwan, K. Greff, H. Meyer, K. Genova, M. Sajjadi, E. Pot, A. Tagliasacchi, and D. Duckworth. NeSF: Neural Semantic Fields for Generalizable Semantic Segmentation of 3D Scenes. *Transactions on Machine Learning Research*, 2022. 1, 3

[40] D. Wang, T. Zhang, A. Abboud, and S. Süsstrunk. In-NeRF360: Text-Guided 3D-Consistent Object Inpainting on 360-degree Neural Radiance Fields. *Conference on Computer Vision and Pattern Recognition*, 2024. 8

[41] M. Ye, M. Danelljan, F. Yu, and L. Ke. Gaussian grouping: Segment and edit anything in 3d scenes. *arXiv Preprint*, 2023. 3

[42] H. Yu, L. Guibas, and J. Wu. Unsupervised discovery of object radiance fields. In *International Conference on Learning Representations*, 2022. 3

[43] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and Improving Neural Radiance Fields. *arXiv Preprint*, 2020. 3

[44] S. Zhi, T. Laidlow, S. Leutenegger, and A. Davison. In-Place Scene Labelling and Understanding with Implicit Scene Representation. In *International Conference on Computer Vision*, 2021. 1

DiscoNeRF: Class-Agnostic Object Field for 3D Object Discovery

Supplementary Material

6. Additional discussion

6.1. Analysis of confidence

In Fig. 8, we visualize the confidence estimation of our object network. We define confidence as the maximum value per-pixel across all object slots. Low-confidence regions often correspond to objects that were not correctly identified, such as the bottles in the *room* scene or the isolated patch of table in the *kitchen*. In future work, an interesting direction would be to adaptively generate 2D mask supervision for regions with low object field confidence.

6.2. Detailed quantitative results

In Tab. 2, we provide the complete results of our method and baselines on the Mip-NeRF 360 dataset. The ground truth masks are segmented by hand and cover all pixels in their respective images. The first and the one-hundredth images of each scene are selected for evaluation, and the number of object per-image ranges between 5 and 38. Similarly to the aggregated results reported, our method outperforms baselines for all tested scenes and for all metrics.

6.3. Limitations

Despite our proposed matching and regularization to reduce the effect of 2D mask inconsistency, we observed that thin structures remain a challenge. As illustrated in Fig. 9 with the *bicycle* scene, our method is unable to assign a single object slot to the bench, which is partially occluded by the bicycle. Thin structures are challenging to reconstruct accurately for NeRFs, and SAM often fails to capture them and tends to produce masks that encapsulate more than the object itself.

Furthermore, partially occluded objects may not be satisfyingly reconstructed by the initial NeRF. As a result, object extraction will sometimes reveal NeRF *floaters* in regions of space which are not supervised by the ground-truth images. This effect is visible in Fig. 10, where the table underneath the pedestal includes some reconstruction artifact.

7. Implementation details

7.1. Mask generation

We generate the 2D supervision masks with SAM [20]. Specifically, we used the pretrained ViT-H model weights and the *Automatic Mask Generator* class, which takes as input an RGB image and *no prompt*, and outputs a set of masks separating objects in the image. Since SAM pads images to be square, we first crop the input images to squares in order to produce higher quality masks. In post-processing, we remove masks which are included in another mask. We also discard masks that take up less than 3% of the total image in pixels. For each mask generated, SAM estimates a predicted IoU score and a stability score. We additionally discard masks for which any of these scores is below 70%. Note that the union of these masks may not cover each pixel. This is taken into consideration in our method, and some object slots may not be matched with a mask at every iteration.

For our experiments on the Mip-NeRF 360 dataset, we select a SAM granularity parameter $g \in [1, 3]$ such that at least 5 different masks are produced for each scene. Precisely, we use granularity levels of 1 for *bicycle*, 0 for *bonsai*, and 2 for the remaining scenes. Since the images in this dataset are very high resolution, we downscale images by a factor of 4 prior to mask generation, and observe that this tends to lead to smoother masks. For this same reason, we dilate masks by 5 pixels to reduce the effect of possible artifacts produced by SAM. For object-centric scenes, we additionally generate a background mask by masking the region outside of the unit-sphere in the 3D NeRF. The foreground is then given to SAM as input to generate masks for the remaining objects.

7.2. NeRF reconstruction

In this section, we describe the NeRF implementation used in our approach. Note that this NeRF reconstruction is not central to our method and could easily be replaced with another implementation. In fact, our work could be extended to other differentiable rendering techniques as the only requirement of our method is a differentiable method to produce 2D object images from a learned 3D field.

We first encode the position $\mathbf{x}$ using learned hash grid features. We start NeRF optimization with the 4 lowest resolution hash grids initially, and increase the number of activated hash grids every 50 iterations, up to a maximum of 16. The interpolated features are then decoded by a multilayer perceptron with 3 hidden layers and a hidden layer dimension of 64. Afterwards, features of dimension 15 are concatenated to the view direction and fed to the color decoder, which has 3 hidden layers of 32 dimensions.

We assume the region of interest is located within the $[-1, 1]^3$ cube, and contract the background region $[-128, -1] \times [1, 128]$ such that we only sample points in $[-2, 2]$, as described in [2]. Additionally, we optimize a proposal network with 2 hidden layers of 16 dimensions. In our experiments, we observed that a satisfying 3D segmentation was learned within the first minute of training. We

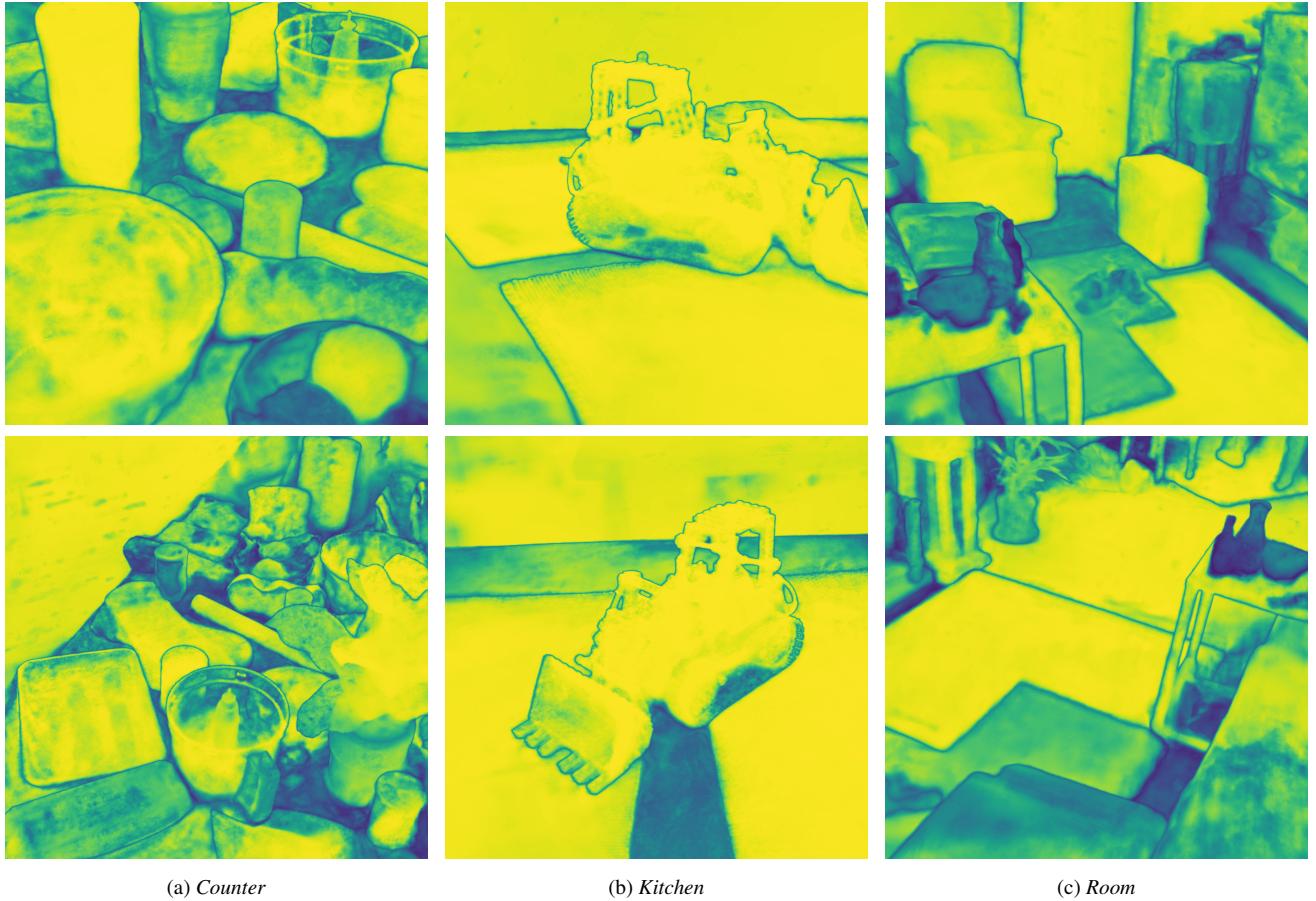(a) *Counter*   (b) *Kitchen*   (c) *Room*

Figure 8. Visualization of object prediction confidence, defined as the maximal object value per-pixel. Bright regions indicate high confidence and a well-defined object.

(a) Ground-truth   (b) Segmentation   (c) Confidence
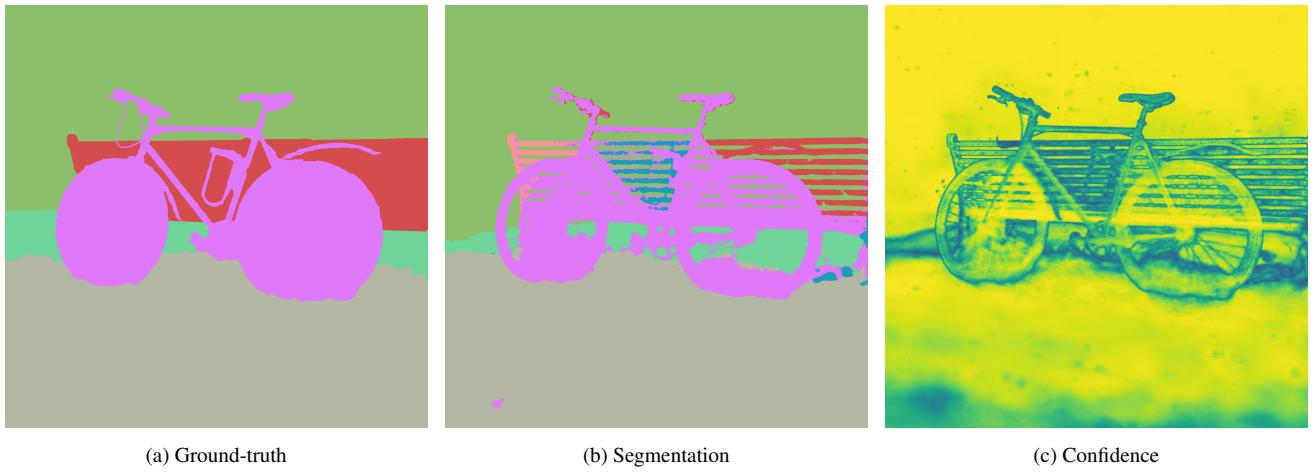
Figure 9. **Limitations.** Accurately segmenting thin structures remains a challenge.

further train for a total of 10000 iterations to attain convergence, which takes between 10 and 30 minutes depending on the complexity of the scene and the number of images.

All networks are trained on a single A100 GPU with 40Gb of VRAM.

In order to prevent NeRF *floaters* and simplify object ex-

Input	Metric	DFFv2	Instance-NeRF	Panoptic Lifting	PL+SAM	Ours
bicycle	IoU	70.29	26.04	26.04	75.06	**79.24**
	BD	82.47	51.23	51.23	84.56	**87.51**
	SBD	79.56	40.24	40.24	83.82	**85.34**
bonsai	IoU	76.57	34.68	39.15	85.26	**88.63**
	BD	83.51	53.16	70.59	**94.85**	91.12
	SBD	83.14	49.55	53.19	89.73	**90.33**
counter	IoU	49.08	25.66	19.77	46.00	**69.86**
	BD	60.84	52.76	44.83	65.45	**83.33**
	SBD	59.62	35.33	27.53	55.24	**78.52**
garden	IoU	85.64	57.57	34.45	58.85	**93.67**
	BD	88.66	72.04	62.86	72.58	**97.03**
	SBD	88.66	69.00	49.87	71.16	**96.33**
kitchen	IoU	68.28	19.98	20.81	86.05	**88.87**
	BD	77.56	40.95	41.75	93.84	**95.36**
	SBD	77.56	32.82	33.93	90.06	**92.80**
room	IoU	43.76	26.36	35.75	33.49	**59.95**
	BD	54.39	52.34	61.21	58.60	**73.64**
	SBD	54.39	38.22	45.64	47.38	**71.70**
Average	IoU	65.60	31.71	29.33	64.12	**79.24**
	BD	74.57	53.75	55.41	78.32	**87.51**
	SBD	73.82	44.19	41.73	72.90	**85.34**

Table 2. Segmentation results on the Mip-NeRF 360 [2] dataset.

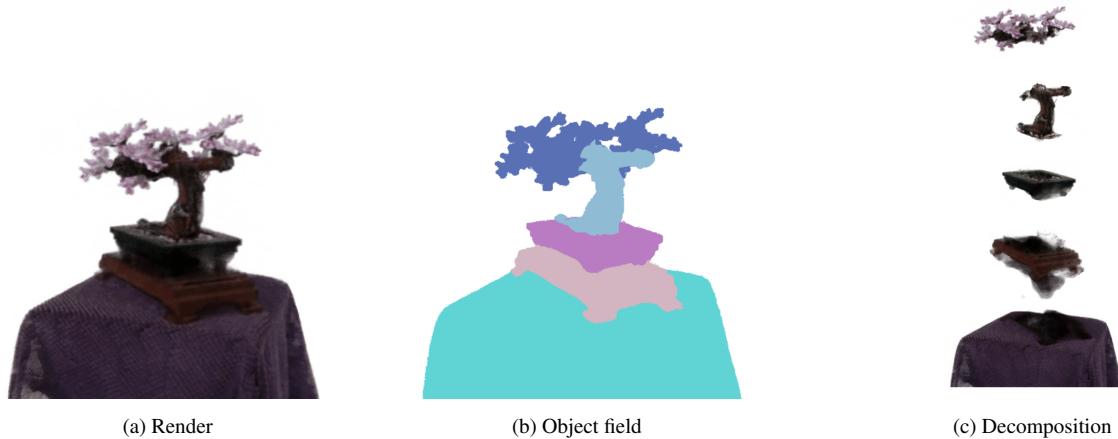(a) Render (b) Object field (c) Decomposition

Figure 10. Object decomposition of the *bonsai* model.

traction, we add an L1 loss on the densities of points randomly sampled in $[-2, 2]$, referred to as $\mathcal{L}_{empty}$. This loss is paired with a very small $\lambda_{empty} = 10^{-4}$ to regularize unseen regions of space without interfering with the actual geometry.

7.3. Object Field

We optimize the object field for 2000 iterations with a batch size of $B = 5$. This batch size defines the number of randomly sampled views each iteration, where each view is associated to up to K masks. In our experiments, K ranges from 5 to 60 based on the selected granularity level and

the complexity of the scene. Note that the matching of object slots and masks is performed per-view, such that different views don't interfere with each other and the matching method is computed B times per iteration.

For a 3D point $\mathbf{x}$, we first encode $\mathbf{x}$ using a learned feature hash grid with 8 levels. Then, these features are decoded by a fully-connected network with 2 hidden layers and a hidden dimension of 32. The final layer outputs N values, where N is the maximum number of objects. Finally, the resulting vector is fed to a softmax layer. We set the initial learning rate to 0.15, and apply a learning rate decay of 0.005 after 20 iterations of warm-up. The loss weights are set to $\lambda_{TV} = \lambda_{FP} = 0.01$.

Note that we freeze the weights of the NeRF reconstruction in order to train the object field. Previous work [37] has shown that without a stop gradient on the semantic loss, additional floaters appeared in the 3D reconstruction. Furthermore, we optimize NeRFs by sampling random rays across all views, whereas the object network is trained by sampling an image and considering all masks available for that image. This major difference in training method and the results of previous work explain our decision to not jointly train our NeRF and object field networks.