Enforcing View-Consistency in Class-Agnostic 3D Segmentation Fields

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 [21]. 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 [38] 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.