Supplementary Material

The Supplementary Material is organized as follows. In Sec. A, we provide additional implementation details. In Sec. B, we present ablations on the NeRF-based pseudo-labels, showing the effect on their quality of different parameters and components of our method. In Sec. C, we report additional evaluations for the one-step adaptation experiments. In Sec. D we include in-detail results for the multi-step adaptation experiments and ablate on the replay-based strategy proposed by our method. In Sec. E we analyze in detail the memory footprint required by our method and by the different baselines that we compare against in the main paper. In Sec. F, we provide further visualizations, including examples of the pseudo-labels and network predictions produced by our method and the baselines. In Sec. G, we discuss limitations of our method and potential ways to address them. We will further release the code to reproduce our results.

Similarly to the main paper, in all the experiments we report mean intersection over union (mIoU, in percentage values) as a metric.

A. Additional implementation details

NeRF. Following Instant-NGP [8, 12], to facilitate training of the hash encoding, we re-scale and re-center the poses used to train NeRF so that they fit in a fixed-size cube. For each ray that is cast from the training viewpoints, to render the aggregated colors and semantics labels we first sample 256 points at a fixed interval and then randomly select 256 additional points according to the density values of the initial points.

The base NeRF network uses a multi-resolution hash encoding with a 16-level hash table of size 2^{19} and a feature dimension of 2. Similarly to Semantic-NeRF [14], we implement the additional semantic head as a 2-layer MLP. In all the experiments, we train all the components of the Semantic-NeRF network concurrently, setting the hyperparameters in Eq. (5) from the main paper to $w_{\mathrm{d}} = 0.1$ and $w_{\mathrm{s}} = 0.04$ as suggested in [14], sampling 4096 rays for each viewpoint, and using the Adam [4] optimizer with a fixed learning rate of $1\mathrm{e}{-2}$.

In all the experiments in which the semantic segmentation model is trained using NeRF-rendered images, we use Adaptive Batch Normalization (AdaBN) [5] when performing inference on the ground-truth images, to improve the generalization ability of the model between NeRF-rendered images and ground-truth images.

Dataset. For convenience of notation, we re-map the scene indices in the dataset from $0000 - 0706$ to $1 - 707$ (so that we refer to scene 0000 as scene 1, to scene 0001 as scene 2, etc.). For sample efficiency, we downsample each sequence from the original $30\,\mathrm{fps}$ to $3\,\mathrm{fps}$, resulting in a total of 100 to 500 frames for each video sequence.

Pre-training. To pre-train DeepLab on scenes $11 - 707$ from ScanNet, we initialize the model parameters with the weights pre-trained on the COCO semantic segmentation dataset [6]. We then run the pre-training on ScanNet using the Adam [4] optimizer with batch size of 4, and let the learning rate decay linearly from $1\mathrm{e}{-4}$ to $1\mathrm{e}{-6}$ over 150 epochs.

One-step adaptation. In all the one-step experiments with our method and with the baseline of [3], the semantic segmentation model is trained for 50 epochs with a fixed learning rate of $1\mathrm{e}{-5}$ and batch size of 4. Since CoTTA is an *online* adaptation method, in accordance with the settings introduced in the original paper, we adapt the segmentation network for a single epoch and with batch size 1, setting the learning rate to $2.5\mathrm{e}{-6}$. To prevent overfitting the semantic segmentation model to the training views of the new scene, we apply the same data augmentation procedure as in pre-training in each training step for our method and for [3]. Since CoTTA already implements a label augmentation mechanism for ensembling, we apply to the method only the augmentations used by its authors.

Multi-step adaptation. In the multi-step adaptation experiments, we use a batch size of 4 during training, where 2 samples come from the subset of the pre-training dataset used for replay (cf. main paper), and the other 2 data points are uniformly sampled from the training frames of the new scene and the replay buffer of the previous scenes.

Hardware. We train all our models using an AMD Ryzen 9 5900X with $32\,\mathrm{GB}$ RAM, and an NVIDIA RTX3090 GPU with $24\,\mathrm{GB}$ VRAM.

B. NeRF-based pseudo-labels

In the following Section, we present ablations on the NeRF-based pseudo-labels, showing how the chosen NeRF implementation and the losses used in our method influence their segmentation accuracy.

B.1. Comparison of NeRF frameworks

We compare the segmentation quality of the pseudo-labels obtained with our Instant-NGP [8, 12]-based implementation to that achieved with the original Semantic-NeRF [14] implementation, which we adapt to include the newly-introduced semantic loss (cf. Sec. 3.2 in the main paper and Sec. B.2). To this purpose, we train a semantics-aware NeRF model for scene 1 with both the methods, running the experiments 3 times for each method. In each run, we train the original implementation of Semantic-NeRF [14] for $200\mathrm{k}$ steps and the one based on Instant-NGP [12] for 10 epochs (for a total of $10 \times 447 = 4470$ steps), which allows achieving a similar color reconstruction quality (measured as PSNR) for the two methods.

Components		Scene										
$\mathcal{L}_\text{d}$	$\mathcal{L}_\text{s}$	Scene 1	Scene 2	Scene 3	Scene 4	Scene 5	Scene 6	Scene 7	Scene 8	Scene 9	Scene 10	Average
✗	Semantic-NeRF [14]	44.3±1.5	34.2±0.1	22.4±0.9	**63.5**±1.2	52.3±1.2	47.3±0.5	38.9±0.6	33.8±0.4	32.4±0.5	53.3±0.6	42.2±0.7
✗	Ours	46.4±1.1	33.0±0.2	24.2±0.3	62.6±0.7	53.4±0.7	46.8±1.1	39.3±0.8	**34.5**±0.6	33.8±0.6	55.8±0.2	43.0±0.6
✓	Semantic-NeRF [14]	44.0±0.6	34.8±0.5	22.8±0.9	63.1±0.7	55.8±2.0	49.1±1.2	39.0±0.8	33.9±0.5	33.0±1.5	55.1±0.6	43.1±0.9
✓	Ours	**48.4**±0.9	**36.0**±0.3	**26.1**±0.4	61.6±0.5	**57.0**±1.8	**50.3**±0.6	**39.8**±0.2	33.5±0.6	**35.4**±0.7	**57.4**±0.1	**44.6**±0.7

Table 1. Effect of the ℓ_1 depth loss $\mathcal{L}_\text{d}$ and of different types of semantic losses (either the original one proposed in [14] or ours) on the pseudo-label quality. The performance is evaluated on the training views of each scene and averaged over 3 runs.

	Semantic-NeRF [14]	Instant-NGP [8] (impl. by [12])
PSNR	19.9 ±0.1	19.3 ±0.1
mIoU	50.0 ±0.5	48.4 ±0.9
Model size (MB)	4.9	50.0
Training time / Step (s)	0.19	0.06
Total training time (min)	633	5
Inference time / Image (s)	2.8	0.3

Table 2. Pseudo-label performance on the training views of scene 1, size of the associated model checkpoint, and the training and inference time using different NeRF frameworks. The implementation of [14] has been adapted to include the newly-introduced semantic loss (cf. Sec. 3.2 in the main paper). The results are averaged over 3 runs.

As shown in Tab. 2, the pseudo-labels produced by both implementations achieve a similar mIoU, with Semantic-NeRF slightly outperforming Instant-NGP. Furthermore, the size of the models produced by Semantic-NeRF is approximately 10 times smaller than the one required by Instant-NGP, at the cost however of longer training ($\sim 127\times$) and rendering ($\sim 9\times$) time.

Since in a real-world deployment scenario achieving fast adaptation might be of high priority, in the main paper we adopted the faster framework of Instant-NGP. However, the results above indicate that our method is agnostic to the specific NeRF framework chosen, and similar segmentation performance can be achieved by trading off between speed and model size depending on the main requirements. Further evaluations on the memory footprint in comparison also with the baselines of [3] and [13] are presented in Sec. E.

B.2. Ablation on the NeRF losses

To investigate the effect of depth supervision [1] (through the ℓ_1 depth loss $\mathcal{L}_\text{d}$) and of the proposed modifications to the semantic loss $\mathcal{L}_\text{s}$ (cf. Sec. 3.2 in the main paper), we evaluate on each scene the pseudo-labels produced by our method when ablating on these factors. For each scene, we train the NeRF model for 10 epochs without joint training, as we find training without semantic loss modifications is unstable for longer epochs. We run each experiment 3 times and report average and standard deviation across the runs. As shown in Tab. 1, both components induce a significant improvement of the pseudo-label quality. In particular, depth supervision and the use of our modified semantic loss instead of the one proposed in [14] produce an increase respectively of 0.9% mIoU and 0.8% mIoU over the baseline with no modifications. The combined use of both ablated factors further increases the pseudo-label performance, resulting in a total improvement by 2.4% mIoU.

The effect of the proposed modifications can also be observed in Fig. 1. In particular, as shown in Fig. 1a, the use of depth supervision is critical for properly reconstructing the scene geometry. The large number of artifacts in the reconstruction when the depth loss is not used are also reflected in the semantic pseudo-labels, which contain large levels of noise and often fail to assign a uniform class to each entity in the scene (Fig. 1b). Depth supervision applied together with the original semantic loss from [14] resolves some of the artifacts in the pseudo-labels, but still results in suboptimal quality. The combined use of depth supervision and of our modified semantic loss produces cleaner and smoother pseudo-labels, which also attain higher segmentation accuracy, as shown in Tab. 1.

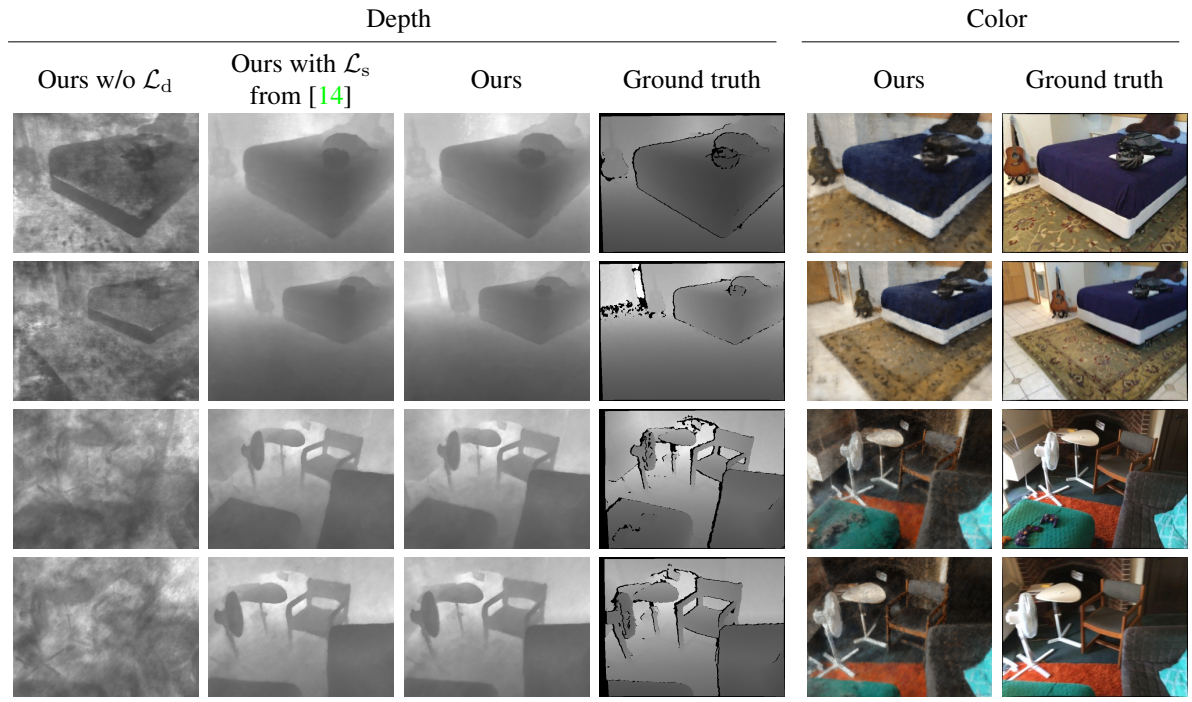
C. One-step adaptation

In this Section, we report additional results on the one-step adaptation experiments.

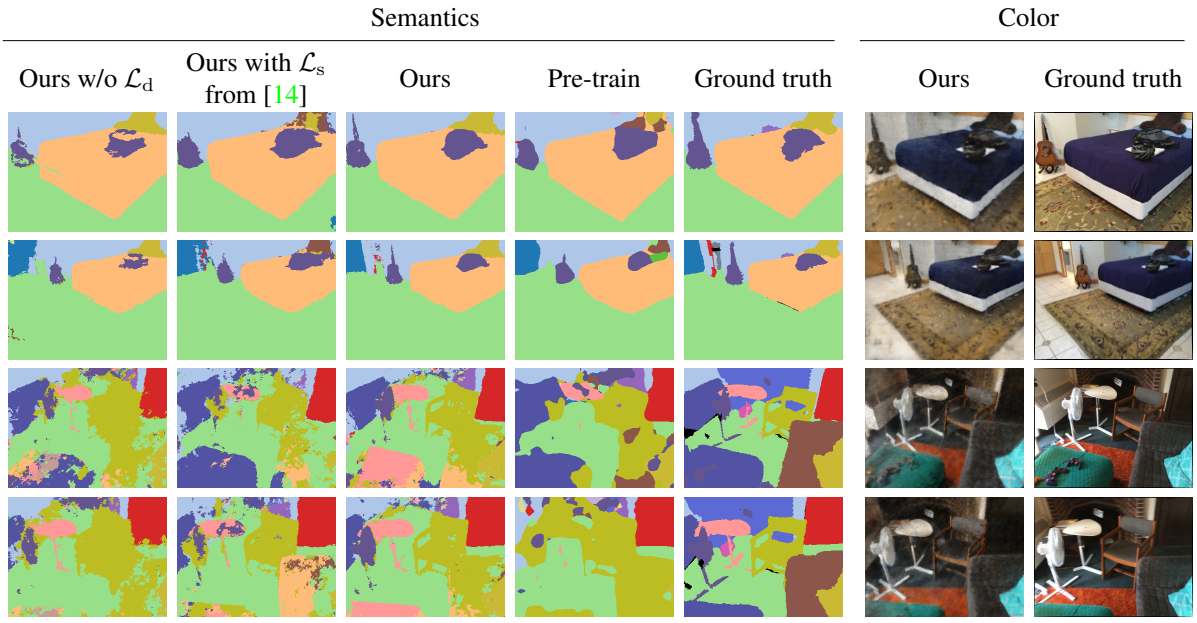
C.1. One-step adaptation performance on the training set of each scene

Since in the scenario of a deployment of the semantic segmentation network on a real-world system a scene might be revisited from viewpoints similar to those used for training, in Tab. 3 we report the one-step adaptation performance evaluated on the training views. We compare our method to the baseline of CoTTA [13] and to fine-tuning, both with the pseudo-labels of [3] and with our NeRF-based pseudo-labels. For each method, we run the experiments 10 times and report average and standard deviation across the runs.

Similarly to the results obtained on the validation views (cf. main paper), our method with joint training obtains the best average performance across all scenes. Unlike what observed on the validation views, however, on the training views joint training does not result in an average performance improvement over fine-tuning with our NeRF-based pseudo-labels (NI + NL). We note however that these results are largely influenced by the outlier of Scene 5, where joint training achieves significantly lower segmentation accuracy. In Sec. G we analyze more in detail the failure cases



(a) Rendered depth



(b) Rendered semantics

Figure 1. Effect on the rendered depth and semantics of depth supervision and of the modification to the semantic loss. Black pixels in the ground-truth depth and ground-truth semantics denote respectively missing depth measurement and missing semantic annotation.

of our method and focus specifically also on Scene 5, which we find to contain several frames with extreme illumination conditions, which makes it particularly challenging to properly reconstruct the geometry of certain parts of the scene.

D. Multi-step adaptation

In the following Section, we include in-detail results for the multi-step adaptation experiments, reporting addition-

	Pre-train	CoTTA [13]	Fine-tuning (GI + ML)	Fine-tuning (GI + NL)	Fine-tuning (NI + NL)	Joint Training
Scene 1	41.1	41.9±0.0	50.6±0.1	50.1±0.6	50.7±0.5	**55.5**±1.3
Scene 2	35.5	35.6±0.0	33.5±0.1	35.7±0.8	36.6±0.3	**39.5**±0.8
Scene 3	23.5	23.7±0.0	24.4±0.1	26.9±1.0	27.1±1.2	**27.5**±1.6
Scene 4	62.8	63.0±0.0	66.1±0.3	63.2±0.6	66.1±0.8	**67.7**±1.7
Scene 5	49.8	49.8±0.0	51.2±0.1	57.1±1.2	**59.9**±1.5	46.3±0.3
Scene 6	48.9	48.9±0.0	**53.1**±0.1	50.2±0.4	49.9±0.4	50.7±0.2
Scene 7	39.7	39.8±0.0	41.4±0.1	40.8±0.6	42.1±0.8	**43.8**±1.6
Scene 8	31.6	31.7±0.0	36.2±0.2	34.4±0.5	33.9±0.4	**38.1**±3.5
Scene 9	31.7	31.7±0.0	32.7±0.1	**35.5**±0.6	34.9±0.8	32.5±0.9
Scene 10	52.5	52.7±0.0	57.8±0.1	57.1±0.6	**58.4**±0.6	57.4±1.4
Average	41.7	41.9±0.0	44.7±0.1	45.1±0.7	**45.9**±0.7	**45.9**±1.3

Table 3. One-step adaptation performance on the training views of each scene. GI and NI denote respectively ground-truth color images and NeRF-rendered color images. ML and NL indicate adaptation using pseudo-labels formed respectively with the method of [3] and with our approach. In joint training, we use NeRF-based renderings and pseudo-labels. For each method, we run the experiments for 10 times and report average and standard deviation across the runs.

ally a set of standard metrics used in the continual learning literature. We further demonstrate the use, enabled by our method, of images and pseudo-labels rendered from *novel* viewpoints in previous scenes for multi-step adaptation. Remarkably, we find that this modification induces a further improvement in the retention of knowledge from the previous scenes.

D.1. Detailed per-step evaluation

Table 5 reports the segmentation performance on the validation views of each scene after each step of adaptation, both for our method and for the baselines of [13] and [3]. For each method, we run the experiment 3 times and report main and standard deviation across the runs. The results complement Tab. 3 in the main paper, confirming in particular that in all the adaptation steps our method is the most effective at preserving knowledge on the previous scenes.

	ACC Metric [7]	A Metric [2]	FWT [7]	BWT [7]
CoTTA [13]	44.6±0.0	40.9±0.0	**-0.2**±0.0	**-0.1**±0.0
Mapping [3]	45.8±0.6	42.1±0.5	-1.1±0.2	-1.0±0.6
Ours ($\mathbf{I}_{\text{pre}}$ replay only)	46.8±0.8	43.7±0.6	-1.4±0.7	-1.4±0.7
Ours	**47.2**±0.5	**44.3**±0.2	-1.1±0.2	-0.9±0.4

Table 4. Continual learning metrics extracted from Tab. 5.

To facilitate the analysis of the results, in Tab. 4 we further report a set of metrics commonly used in the continual learning literature. Our method achieves the best performance both according to the ACC metric [7] and to the A metric [2], meaning that it obtains the best average mIoU across all previously visited scenes both at the final step and at any arbitrary adaptation step. The baseline of CoTTA [13] attains the best forward transfer (FWT) [7] and backward transfer (BWT) [7], which indicate respectively the influence that previous scenes have on the performance on future scenes and the influence that adaptation on the current scenes has on the performance on the previous scenes (negative BWT corresponds to catastrophic forgetting). An important point to notice, however, is that the performance of CoTTA also does not vary significantly with respect to the pre-trained model, and in particular does not improve on average. Among the other methods, our method achieves the best FWT and BWT, which demonstrates the effectiveness of our NeRF-based replay buffer in alleviating forgetting and improving the generalization performance.

D.2. "Replaying" from novel viewpoints

A key feature enabled by our method is the possibility of rendering both photorealistic color images and pseudo-labels from any arbitrary viewpoint inside a reconstructed scene. Crucially, this can include also *novel* viewpoints not seen during deployment and training, which can then be used for adaptation, at the fixed storage cost given by the size of the NeRF model parameters. In the following, we present an experiment demonstrating this idea in the multi-step adaptation scenario. Using the notation introduced in the paper, in each step $i \in \{1, \dots, 10\}$, the semantic segmentation network $f_{\theta_{i-1}}$ is adapted on scene $\mathcal{S}_i$, and for each previous scene $\mathcal{S}_j$, $1 \leq j < i$ images and pseudo-labels rendered from viewpoints $\hat{\mathbf{P}}_j$ are inserted in a rendering buffer and mixed to the data from the current scene. However, unlike the experiments in the main paper, we do not enforce that for each scene $\mathcal{S}_j$ the viewpoints $\hat{\mathbf{P}}_j$ used for the rendering buffer coincide with those used in training $\mathbf{P}_j := \{\boldsymbol{P}_j^k\}_{k \in \{1, \cdots, |\mathbf{I}_j|\}}$, but instead allow novel viewpoints to be used, that is, $|\hat{\mathbf{P}}_j \backslash (\hat{\mathbf{P}}_j \cap \mathbf{P}_j)| > 0$.

Specifically, in the presented experiment we apply simple average interpolation of the training poses, and for each viewpoint $\hat{\boldsymbol{P}}_j^k \in \hat{\mathbf{P}}_j$ we compute its rotation component

Method	Step	Scene 1	Scene 2	Scene 3	Scene 4	Scene 5	Scene 6	Scene 7	Scene 8	Scene 9	Scene 10	Average Prev.	Average
Pre-training	–	43.9	41.3	23.0	50.2	40.1	37.6	55.8	27.9	54.9	73.5	–	44.8
CoTTA [13]	1	44.0±0.0	40.9±0.0	22.9±0.0	50.3±0.0	40.1±0.1	37.5±0.0	55.9±0.0	27.6±0.0	54.7±0.0	73.6±0.0	–	44.7±0.0
	2	44.0±0.0	40.9±0.0	22.9±0.0	50.3±0.0	40.1±0.0	37.5±0.0	55.9±0.0	27.6±0.0	54.8±0.0	73.6±0.0	44.0±0.0	44.7±0.0
	3	43.6±0.1	40.7±0.1	22.7±0.0	50.1±0.1	39.9±0.0	37.5±0.0	56.1±0.0	27.3±0.0	54.6±0.0	73.7±0.0	42.2±0.0	44.6±0.0
	4	43.6±0.0	40.5±0.0	22.7±0.0	50.2±0.1	39.9±0.0	37.5±0.0	56.0±0.1	27.2±0.0	54.5±0.0	73.7±0.0	35.6±0.0	44.6±0.0
	5	43.7±0.1	40.5±0.0	22.7±0.0	50.2±0.1	40.0±0.0	37.5±0.0	55.9±0.1	27.1±0.0	54.6±0.0	73.7±0.0	39.3±0.0	44.6±0.0
	6	43.7±0.0	40.4±0.1	22.7±0.0	50.3±0.1	40.0±0.0	37.5±0.0	55.9±0.1	27.0±0.0	54.5±0.0	73.7±0.0	39.4±0.0	44.6±0.0
	7	43.7±0.1	40.4±0.1	22.7±0.1	50.3±0.1	39.9±0.1	37.6±0.1	56.0±0.1	26.9±0.0	54.5±0.0	73.7±0.0	39.1±0.0	44.6±0.0
	8	43.7±0.0	40.4±0.1	22.7±0.1	50.3±0.1	39.9±0.1	37.7±0.1	56.0±0.1	**26.9**±0.0	54.5±0.0	73.7±0.0	41.5±0.0	44.6±0.0
	9	43.7±0.0	40.3±0.1	22.7±0.1	50.2±0.1	39.9±0.1	37.7±0.1	56.0±0.1	26.8±0.0	54.5±0.0	73.8±0.0	39.7±0.0	44.6±0.0
	10	43.7±0.1	40.2±0.1	22.7±0.1	50.3±0.1	39.9±0.1	37.6±0.0	56.1±0.1	26.8±0.0	54.4±0.1	**73.8**±0.0	41.3±0.0	44.6±0.0
Mapping [3]	1	46.8±0.4	36.0±1.6	24.2±0.9	48.3±0.9	40.0±0.9	35.3±0.8	55.5±0.4	29.2±2.3	55.7±1.0	73.9±0.2	–	44.5±0.5
	2	46.5±0.1	42.1±2.0	23.6±0.9	48.4±1.3	41.3±1.0	35.5±0.7	54.8±1.1	28.3±0.8	56.5±0.9	73.7±0.2	46.5±0.1	45.1±0.2
	3	43.0±1.2	42.6±2.8	23.6±0.7	48.5±0.7	37.0±2.1	33.7±0.6	55.5±2.0	26.0±0.8	54.2±1.4	74.1±0.3	42.8±1.0	43.8±0.0
	4	45.5±0.3	42.9±2.2	23.5±0.8	**50.6**±2.6	38.5±0.8	34.1±0.9	57.7±0.3	26.7±1.3	55.8±1.9	73.9±0.2	37.3±0.9	44.9±0.5
	5	44.9±0.6	42.9±1.2	23.5±0.7	50.2±2.5	**44.0**±0.1	34.2±0.7	57.3±0.6	26.7±0.4	54.6±1.8	73.6±0.7	40.4±0.6	45.2±0.1
	6	44.8±1.1	43.5±0.6	22.8±0.9	49.6±2.4	43.9±0.4	35.8±0.5	57.9±1.3	25.7±0.0	56.1±1.7	73.3±0.6	40.9±0.7	45.3±0.3
	7	43.5±1.6	43.7±0.8	22.9±1.1	50.4±2.7	43.4±0.6	35.6±0.3	56.7±1.3	25.7±1.7	55.5±2.6	73.7±0.4	39.9±1.1	45.1±0.6
	8	42.0±0.7	43.5±1.2	23.0±0.7	50.3±2.5	43.8±0.1	35.9±1.5	57.1±0.1	26.5±1.8	56.1±2.9	73.9±0.5	42.2±0.5	45.2±0.2
	9	43.0±0.9	43.9±1.2	22.2±0.3	49.8±2.4	43.6±0.2	35.2±0.7	56.8±0.2	25.6±1.2	68.3±1.4	74.1±1.2	40.0±0.4	46.2±0.2
	10	42.5±0.7	43.5±1.3	22.5±0.3	49.7±2.5	43.6±0.2	35.6±1.1	55.6±1.0	26.2±1.4	65.8±4.0	72.7±1.0	42.8±0.7	45.8±0.6
Ours ($\mathbf{I}_{\text{pre}}$ replay only)	1	53.3±0.7	35.4±1.8	24.7±0.1	49.7±1.6	37.4±1.0	32.9±0.2	55.6±1.0	31.9±1.1	55.1±1.2	74.1±0.7	–	45.0±0.3
	2	52.3±0.3	**48.0**±2.4	22.2±0.4	50.0±0.1	43.4±0.9	34.4±1.4	50.3±0.8	29.2±1.9	63.4±3.5	73.2±1.3	52.3±0.3	46.7±0.5
	3	51.8±1.9	43.2±1.6	20.5±0.1	48.6±0.9	40.0±2.1	33.1±1.9	55.3±0.6	27.7±1.5	57.8±4.7	73.7±0.6	47.5±1.1	45.2±0.6
	4	52.9±1.3	41.9±2.3	21.1±0.8	49.0±1.5	37.9±0.9	34.3±1.7	54.5±0.6	32.3±1.1	55.4±0.7	72.9±2.0	38.6±1.1	45.2±0.5
	5	51.5±0.8	41.7±1.0	21.2±0.9	48.8±1.2	43.4±0.0	35.2±0.5	56.4±1.0	29.3±0.1	53.2±2.4	72.2±1.0	40.8±0.7	45.3±0.5
	6	53.4±1.1	44.6±1.2	20.5±0.5	49.2±1.5	44.4±0.6	39.0±1.4	51.3±5.3	30.7±2.4	57.3±2.3	71.9±1.6	42.4±0.3	46.3±0.7
	7	52.1±0.5	45.5±2.5	21.1±0.3	49.7±1.1	44.0±0.4	36.6±1.8	**62.1**±6.2	31.1±0.7	60.2±2.5	74.8±0.5	41.5±0.7	47.7±0.1
	8	50.7±2.1	47.1±2.5	21.0±0.7	49.3±1.6	44.3±1.7	38.2±1.5	59.6±7.2	26.7±3.0	57.0±0.9	74.2±0.4	44.3±1.4	46.8±1.1
	9	51.4±1.4	45.6±2.5	20.0±0.8	49.3±1.4	45.8±1.6	36.6±1.7	56.0±4.0	26.6±3.1	65.7±5.6	73.1±0.5	41.4±0.6	47.0±0.9
	10	48.7±1.5	44.5±3.9	21.1±0.3	50.1±1.5	44.2±1.0	35.5±1.9	56.8±3.5	28.3±3.2	65.8±5.4	73.0±0.5	43.9±0.9	46.8±0.8
Ours	1	**53.7**±1.3	36.6±0.5	24.5±0.9	49.7±0.8	39.7±0.9	34.0±2.4	56.5±1.5	31.7±1.3	56.4±0.5	74.8±0.5	–	45.7±0.2
	2	53.2±0.9	46.3±0.7	23.2±0.5	48.5±1.1	41.9±0.9	33.7±1.5	56.4±1.7	30.4±1.2	59.1±0.5	74.1±0.5	53.2±0.9	46.7±0.2
	3	52.3±1.1	44.0±0.6	**24.3**±2.0	49.2±0.5	38.5±2.8	32.6±0.4	53.2±0.9	28.0±0.3	59.8±5.7	73.8±0.8	48.2±0.8	45.6±0.3
	4	53.5±0.6	46.3±1.4	24.7±2.9	49.1±0.9	37.3±3.4	34.8±2.5	54.8±2.0	29.8±1.2	59.3±4.0	72.9±0.4	41.5±0.8	46.3±0.6
	5	53.0±1.1	44.4±0.8	24.8±2.9	49.1±0.7	43.7±0.3	32.7±1.7	56.0±1.8	29.3±1.3	59.0±2.3	73.2±0.5	42.8±0.8	46.5±0.2
	6	53.0±0.9	45.0±0.9	24.8±2.5	49.0±0.2	44.1±0.5	**40.4**±1.5	54.1±1.8	29.5±2.1	60.0±1.9	72.8±0.4	43.2±0.8	47.3±0.7
	7	51.6±0.4	44.7±0.5	23.8±2.6	49.6±0.5	44.1±0.3	39.2±2.0	55.8±0.8	28.6±1.8	62.1±6.4	73.7±0.3	42.2±0.8	47.3±0.8
	8	50.9±0.3	46.0±0.4	24.3±2.1	49.5±0.2	44.1±0.5	38.9±1.2	54.9±2.1	26.2±0.9	59.5±2.4	74.2±0.2	44.1±0.2	46.9±0.2
	9	51.6±0.3	46.4±1.5	23.6±2.1	49.0±0.3	44.1±0.3	37.4±1.4	55.4±2.8	25.9±0.4	**68.9**±3.2	73.2±0.1	41.7±0.2	47.6±0.2
	10	50.8±0.4	44.6±1.1	23.7±2.1	49.4±0.1	43.8±0.5	37.0±1.9	54.8±1.8	26.1±0.7	69.6±1.0	72.5±1.6	44.3±0.3	47.2±0.5

Table 5. Detail of the multi-step performance evaluated on the validation set of each scene. At Step i, the performance of the adapted network f_{θ_i} on all the scenes is reported (for scenes $\mathcal{S}_j, j > i$ the values are greyed out). Pre-training denotes the performance of the pre-trained network f_{θ_0}. For each Step i, we highlight: in **bold**, the performance of the method which achieves highest mIoU on the current scene $\mathcal{S}_i$, which is indicative of the adaptation performance; in underlined, for each scene $\mathcal{S}_j$, $1 \leq j \leq i-1$ the performance of the method which achieves highest mIoU on $\mathcal{S}_j$, which denotes the ability to preserve previous knowledge; in double-underlined, the performance of the method which achieves highest *average* mIoU on the previous scenes $\mathcal{S}_j, 1 \leq j < i$, which also provides an indication of the ability to counteract forgetting. For each method, the results are averaged over 3 runs. All Ours are with joint training.

through spherical linear interpolation [11] of the rotation components of $\boldsymbol{P}_j^k$ and $\boldsymbol{P}_j^{k+1}$, and its translation component as the average of the translation components of $\boldsymbol{P}_j^k$ and $\boldsymbol{P}_j^{k+1}$. An example visualization of the obtained poses can be found in Fig. 2. The results of the experiment are shown in Tab. 6, which extends on Tab. 3 from the main paper. All the methods are run for 3 times and mean and standard deviation across the runs are reported.

As can be observed from the Adapt results, replaying from novel viewpoints achieves similar adaptation performance on the current scene as the other baselines of Ours, but with a slightly larger variance.

The crucial observation, however, is that this strategy outperforms all the other baselines in terms of retention of previous knowledge (Previous) in almost all the steps, and improves on our method with replay of the training viewpoints on average by 0.7% mIoU. This improvement can be attributed to the novel viewpoints effectively acting as a positive augmentation mechanism and inducing an increase of knowledge on the previous scenes. In other words, rather than simply counteracting forgetting, the model de facto keeps learning on the previous scenes, through the use of newly generated data points.

We believe this opens up interesting avenues for replay-

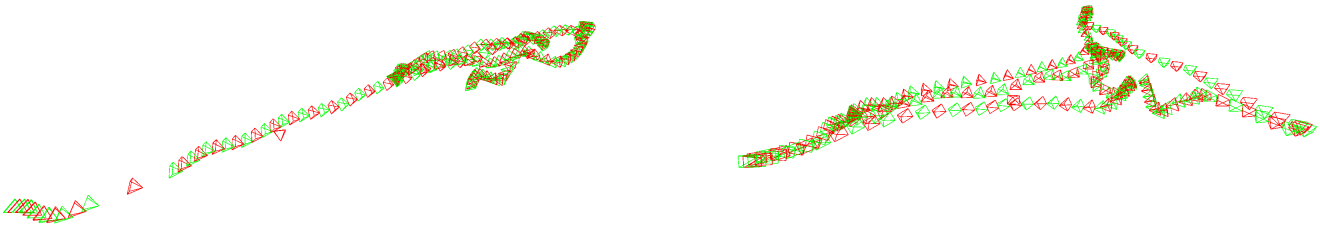


Figure 2. Visualization of the novel viewpoints used for adaptation in Sec. D.2 for two example scenes (Scene 5, left side, and Scene 6, right side). The viewpoints $\mathbf{P}_j$ used for training and the novel viewpoints $\hat{\mathbf{P}}_j$ used for "replay" are shown in green and red, respectively.

		Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8	Step 9	Step 10	Average
Pre-train		43.9	41.3	23.0	50.2	40.1	37.6	55.8	**27.9**	54.9	73.5	44.8
Adapt	CoTTA [13]	44.0±0.0	40.9±0.0	22.7±0.0	50.2±0.1	40.0±0.0	37.5±0.0	56.0±0.1	26.9±0.0	54.5±0.0	**73.8**±0.0	44.7±0.0
	Mapping [3]	46.8±0.4	42.1±2.0	23.6±0.7	**50.6**±2.6	**44.0**±0.1	35.8±0.5	56.7±1.3	26.5±1.8	68.3±1.4	72.7±1.0	46.7±1.2
	Ours ($\mathbf{I}_{\text{pre}}$ replay only)	53.3±0.7	**48.0**±2.4	20.5±0.1	49.0±1.5	43.4±0.0	39.0±1.4	**62.1**±6.2	26.7±3.0	65.7±5.6	73.0±0.5	**48.1**±2.1
	Ours	53.7±1.3	46.3±0.7	**24.3**±2.0	49.1±0.9	43.7±0.3	**40.4**±1.5	55.8±0.8	26.2±0.9	**68.9**±3.2	72.5±1.6	**48.1**±1.3
	Ours (novel viewpoints)	**53.8**±0.4	46.7±2.1	23.2±3.3	49.0±1.0	42.9±0.4	40.1±0.7	58.0±8.5	23.2±2.0	66.7±7.1	71.5±2.2	47.5±2.8
Previous	CoTTA [13]	–	44.0±0.0	42.2±0.0	35.6±0.0	39.3±0.0	39.4±0.0	39.1±0.0	41.5±0.0	39.7±0.0	41.3±0.0	40.2±0.0
	Mapping [3]	–	46.5±0.1	42.8±1.0	37.3±0.9	40.4±0.6	40.9±0.7	39.9±1.1	42.2±0.5	40.0±0.4	42.8±0.7	41.4±0.7
	Ours ($\mathbf{I}_{\text{pre}}$ replay only)	–	52.3±0.3	47.5±1.1	38.6±1.1	40.8±0.7	42.4±0.3	41.5±0.7	44.3±1.4	41.4±0.6	43.9±0.9	43.6±0.8
	Ours	–	53.2±0.9	48.2±0.8	41.5±0.8	42.8±0.8	43.2±0.8	42.2±0.8	44.1±0.2	**41.7**±0.2	**44.3**±0.3	44.6±0.6
	Ours (novel viewpoints)	–	**54.8**±0.9	**50.4**±2.1	**41.8**±0.9	**43.8**±0.8	**43.4**±0.9	**42.7**±1.0	**44.8**±0.9	41.6±0.7	**44.3**±0.2	**45.3**±0.9

Table 6. Multi-step performance evaluated on the validation set of each scene. At Step i, Pre-train and Adapt denote respectively the performance of the pre-trained network f_{θ_0} and of the adapted network f_{θ_i} on the current scene $\mathcal{S}_i$, while Previous represents the average performance of f_{θ_i} on scenes $\mathcal{S}_1$ to $\mathcal{S}_{i-1}$. All Ours are with *joint training*. Our baseline with novel viewpoints used for replay (Ours (novel viewpoints)) is able to consistently retain knowledge better than the other methods.

based adaptation. In particular, more sophisticated strategies to select the viewpoints from which to render could be designed, and further increase the knowledge retention on the previous scenes, without reducing the performance on the current scene.

E. Memory footprint

In the following, we report the memory footprint of the different methods, denoting with N the number of previous scenes at a given adaptation step.

For each previous scene, our method stores the corresponding NeRF model, which has a size of 50.0 MB with Instant-NGP [8, 12] and of 4.9 MB with Semantic-NeRF [14]. This results in either $(N \times 50.0)$MB or $(N \times 4.9)$MB of total data being stored in the *long-term* memory. Note however that during adaptation we only render data from a small subset of views to populate the replay buffer, hence the effective size of the data from the previous scenes that need to be stored in running memory during adaptation is 14.0 MB. Additionally, we save one randomly selected data point every 10 samples in the pre-training dataset, taking up additional 64.6 MB of space.

Similarly to us, the method of [3] requires 14.0 MB for the replay buffer and 64.6 MB for the replay from the pre-training dataset, but stores voxel-based maps instead of NeRF models, taking up 71.8 MB for each scene. Importantly, since the voxel-based maps only include semantic information and cannot be used to render *color* images, the method of [3] additionally needs to save color images for the training viewpoints. In the 10 scenes that we used for our experiments, their size amounted on average to approximately 30.0 MB per scene, resulting in a total storage space of around $(N \times 101.8\,\text{MB})$ required for the previous scenes.

In each step, in addition to the model that gets adapted on the current scene, CoTTA [13] requires storing the teacher model from which pseudo-labels for online adaptation are generated, and an additional version of the original, pre-trained model, to preserve source knowledge. The parameters of the DeepLab network used in our experiments have a size of 224.3 MB, resulting in a total of $(2 \times 224.3)\text{MB} = 448.6\,\text{MB}$ of data that need to be stored.

A comparison of the memory footprint of the different methods as a function of the number of previous scenes can be found in tabular form in Tab. 7 and in graphical form in Fig. 3. For our method and for [3], we include in the total

		Previous scenes		Source knowledge	Total
		Offline	Online		
Ours	Instant-NGP [8, 12]	$(N \times 49.9)$MB*	14.0 MB	64.6 MB	$(78.6 + N \times 49.9)$MB
	Semantic-NeRF [14]	$(N \times 4.9)$MB*			$(78.6 + N \times 4.9)$MB
CoTTA [13]		–	224.3 MB†	224.3 MB†	448.6 MB
Mapping [3]		$\sim (N \times 101.8)$MB**	14.0 MB	64.6 MB	$\sim (78.6 + N \times 101.8)$MB

Table 7. Comparison of the memory footprint of different methods. N denotes the number of previous scenes. * The numbers refer to the storage cost required by the NeRF models. For actual adaptation (Online), only renderings from a subset of views are used, and inserted in a memory buffer of size 14.0 MB. ** The numbers refer to the storage cost required by each voxel-based map (71.8 MB), plus the explicit training views that need to be stored for each scene, which amount to an average of $\sim$ 30.0 MB per scene. Similarly to Ours, for actual adaptation, a memory buffer of size 14.0 MB is used. † CoTTA requires storing a teacher model for online adaptation, and an additional version of the original, pre-trained model, to preserve source knowledge.

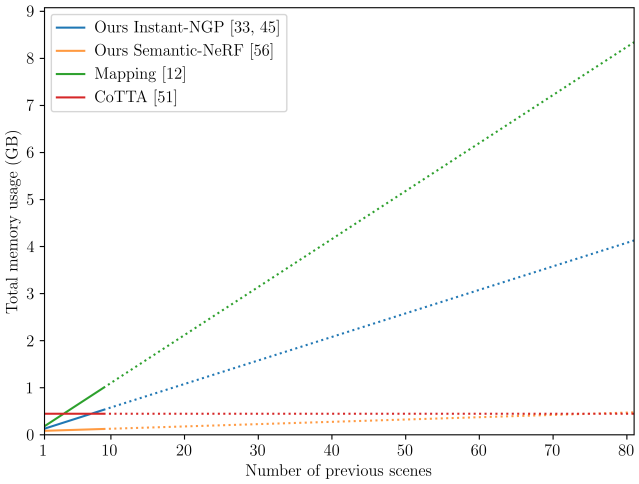


Figure 3. Memory footprint of the different methods as a function of the number of the previous scenes. Please refer to the text and to Tab. 7 for a detailed explanation. We use solid lines for the number of scenes used in our experiments.

size both the data stored offline and the one inserted in the replay buffer.

Note that using the lighter implementation of Semantic-NeRF [14], the comparison is in our favour up to 75 scenes, and up to 91 scenes when only considering the size of the NeRF models.

F. Further visualizations

In Fig. 4 we provide examples of the pseudo-labels produced on the training views by our method and by the different baselines. As previously observed by the authors of [3], the mapping-based pseudo-labels suffer from artifacts induced by the discrete voxel-based representation. Thanks to the continuous representation enabled by the coordinate-based multi-layer perceptrons, our NeRF-based pseudo-labels produce instead smoother and sharper segmentations. However, they occasionally fail to assign a uniform class label to each object in the scene (cf. last row in Fig. 4). This phenomenon, which we also observe in the mapping-based pseudo-labels, can be attributed to the inconsistent per-frame predictions of DeepLab, that cannot be fully filtered-out by the 3D fusion mechanism. By jointly training the per-frame segmentation network and the 3D-aware Semantic-NeRF, we are however able to effectively reduce the extent of this phenomenon, producing more uniform pseudo-labels.

Figure 5 further shows examples of the predictions returned by the segmentation network on the validation views after being adapted using the different methods. In accordance with what observed in the quantitative evaluations, while being able to preserve knowledge, CoTTA achieves limited improvements with respect to the initial performance. As a consequence, the predicted labels match very closely those of the pre-trained network. Fusing the predictions from multiple viewpoints into a 3D representation allows both the baseline of [3] and our method to reduce the amount of artifacts due to misclassifications in the per-frame predictions. The positive effect of this 3D fusion can be successfully transferred to the segmentation network through adaptation, as visible by comparing the predictions in the three rightmost columns of Fig. 5 to those of the pre-trained network (third column from the left in Fig. 5). We observe that fine-tuning with NeRF-based pseudo-labels instead of voxel-based pseudo-labels often results in a more consistent class assignment to different pixels of the same instance. This effect is amplified when using joint training, which often produces more accurate pseudo-labels compared to fine-tuning.

G. Limitations

Since our approach relies on the assumption that a good reconstruction of the scene can be obtained, we find that our method achieves suboptimal performance when this assumption is not fulfilled. This is the case for instance for Scene 5 (cf. first and second row in Fig. 6), in which a large number of frames are overexposed and the ground-truth depth measurements are missing for a large part of the frame. Specular effects (cf. third row in Fig. 6) can further break the assumptions required by the volume rendering formulation of NeRF. Related to these problems is

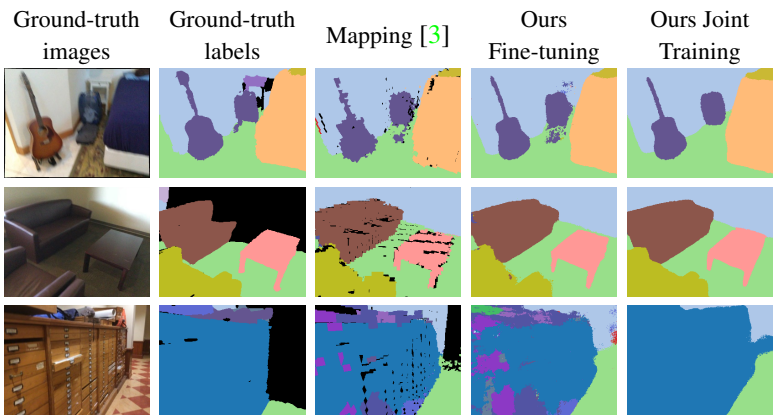


Figure 4. Comparison of example pseudo-labels obtained on the training views by the different methods.

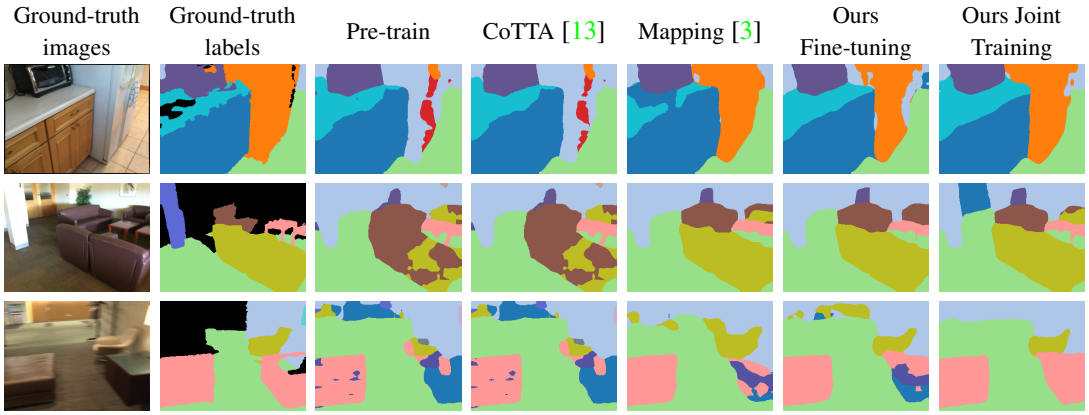


Figure 5. Comparison of the predictions of the semantic segmentation network on the validation views, when adapted using the different methods.

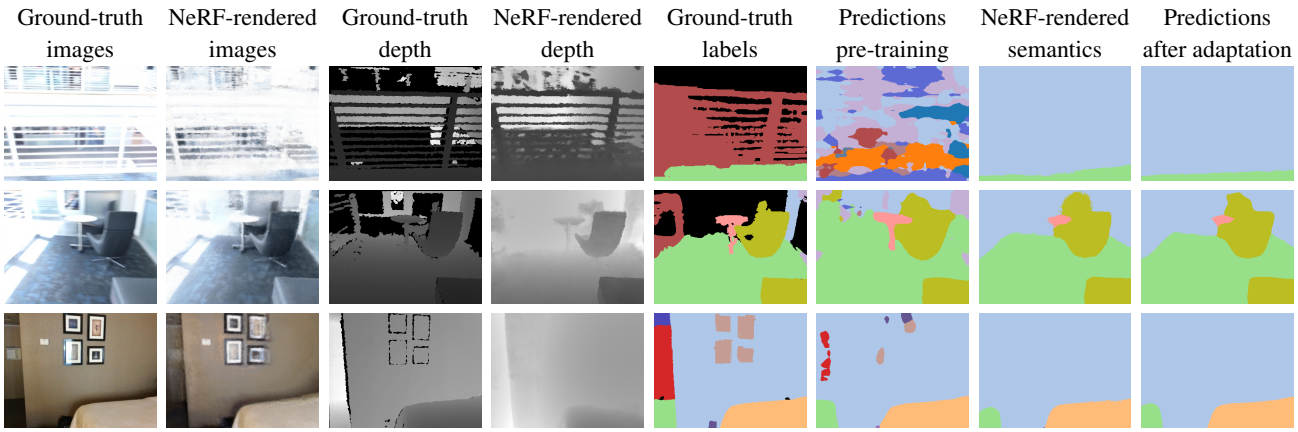


Figure 6. Examples of failure cases. First row: Poor lighting, incomplete ground-truth depth measurements, and noisy initial predictions result in both the rendered pseudo-labels and the predictions from the adapted network assigning uniform labels to large parts of the scene and failing to correctly segment fine details in the scene. Second row: Motion blur, diffusion lighting, shadows, and insufficient number of observations can also degrade the reconstruction quality and make the label propagate to the wrong objects. Third row: Specular effects can break the assumptions of the volume rendering formulation of NeRF; large flat areas with small variations in depth can be hard to reconstruct, resulting in smoothed-out, uniform labels for the background.

also the quality of the initial predictions of the segmentation network: Particularly when lighting conditions are poor, we observe that the predictions of the pre-trained segmentation network are very noisy (see, *e.g.*, first row in Fig. 6). The combination of these factors results in the pseudo-labels produced by our method assigning a uniform label to a large part of the scene and failing to correctly segment smaller details.

We observe that these degenerate cases can have a particularly large influence on the quality of the pseudo-labels and of the network predictions when jointly training the segmentation network and NeRF. We hypothesize that this might be due to the 2D-3D knowledge transfer enabled by our method inducing a negative feedback loop when poor segmentation predictions are combined with suboptimal reconstructed geometry. A possible way to tackle this problem in future work is by making use of regularization techniques, for instance by limiting large deviations of the predictions across adaptation steps, to avoid collapse, or by minimizing the entropy of the semantic predictions of both NeRF and the segmentation network.

A general limitation of our method is that it assumes scenes to be static. Extending the pipeline to handle dynamic scenes through the use of temporally-aware NeRFs [9, 10] is an interesting direction for future work.

References

[1] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer Views and Faster Training for Free. In *CVPR*, 2022. 2

[2] Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: new metrics for Continual Learning. In *NeurIPS Workshop*, 2018. 4

[3] Jonas Frey, Hermann Blum, Francesco Milano, Roland Siegwart, and Cesar Cadena. Continual Adaptation of Semantic Segmentation using Complementary 2D-3D Data Representations. *IEEE Robot. Autom. Lett.*, 2022. 1, 2, 4, 5, 6, 7, 8

[4] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. 1

[5] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting Batch Normalization For Practical Domain Adaptation. *CoRR:1603.04779*, 2016. 1

[6] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *ECCV*, 2014. 1

[7] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient Episodic Memory for Continual Learning. In *NeurIPS*, 2017. 4

[8] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM TOG*, 41(4):102:1–102:15, 2022. 1, 2, 6, 7

[9] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable Neural Radiance Fields . In *ICCV*, 2021. 9

[10] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *CVPR*, 2021. 9

[11] Ken Shoemake. Animating Rotation with Quaternion Curves. In *SIGGRAPH*, 1985. 5

[12] Jiaxiang Tang. Torch-ngp: a PyTorch implementation of instant-ngp. `https://github.com/ashawkey/torch-ngp`, 2022. 1, 2, 6, 7

[13] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual Test-Time Domain Adaptation. In *CVPR*, 2022. 2, 4, 5, 6, 7, 8

[14] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J. Davison. In-Place Scene Labelling and Understanding with Implicit Scene Representation. In *ICCV*, 2021. 1, 2, 3, 6, 7