Supplementary Material for Semantic Ray: Learning a Generalizable Semantic Field with Cross-Reprojection Attention

Fangfu Liu[1,3], Chubin Zhang[2], Yu Zheng[2], Yueqi Duan[1†]
[1]Department of Electronic Engineering, Tsinghua University
[2]Department of Automation, Tsinghua University
[3]Beijing National Research Center for Information Science and Technology

1. Additional implementation details

Training details. At the training time, we first project the query ray instead of a single point to each source view and fetch the corresponding ray-based feature, which contains rich contextual information in each intra-view. For pre-training, we train on a single NVIDIA RTX3090-Ti GPU with 24GB memory. On this hardware, we train our S-Ray for 260k iterations in 60 different scenes of ScanNet [3] (real-world data) and 100k iterations in 12 different scenes of Replica [7] (synthetic data). For finetuning, we only require 10min finetuning time corresponding to 2k iterations. This finetuning result is comparable and even better than 100k optimizations of Semantic-NeRF [10] from each independent scene.

We do not show the specific details of the semantic loss design in the paper. In code implementation, we apply two-stage (coarse and fine) ray sampling as done in NeRF [6]. Therefore, our semantic loss is actually computed as

$$L_{sem} = -\sum_{\mathbf{r}\in\mathcal{R}}\left[\sum_{l=1}^{L} p^l(\mathbf{r})\log\hat{p}_c^l(\mathbf{r}) + \sum_{l=1}^{L} p^l(\mathbf{r})\log\hat{p}_f^l(\mathbf{r})\right] \tag{1}$$

where $\mathcal{R}$ are the set of sample rays within a training batch, $1 \le l \le L$ is the class index, and $p^l, \hat{p}_c^l, \hat{p}_f^l$ are the multi-class probability at class l of the ground truth, coarse semantic logits and fine semantic logits for the query ray $\mathbf{r}$. Actually, for fair comparison in Section 4.2 of our paper, we adopt the same training loss with Semantic-NeRF [10] as:

$$\mathcal{L}_{total} = \lambda_1\mathcal{L}_{sem} + \lambda_2\mathcal{L}_{photometric}, \tag{2}$$

where the color head is from the geometry aware network with photometric loss same as [10]. Like Semantic-NeRF, we also set $\lambda_1 = \lambda_2 = 1$ in Section 4.2 and set $\lambda_1 = 0, \lambda_2 = 1$ as NeRF for ablation study in Table 2 of the paper.

Data split. Our training data consists of both synthetic data and real data. For real data training, we choose 60 different scenes from ScanNet [3] as training datasets and use the image resolution of 320×240. We then choose 10 unseen novel scenes as test datasets to evaluate the generalizability of S-Ray in real data. For synthetic data, we choose 12 different scenes (*i.e.*, 2 rooms, 2 offices, 7 apartments, 1 hotel) from Replica [7] for the training set and the remains (*i.e.*, 2 apartments, 3 offices, 1 room) as test set with the image resolution of 640×480. For each test scene, we select 20 nearby views; we then select 8 views as source input views, 8 as additional input for per-scene fine-tuning, and take the remaining 4 as testing views. Our training data includes various camera setups and scene types, which allows our method to generalize well to unseen semantic scenarios.

2. Additional experiments and analysis

More discussion of loss function. When adding color rendering, it is interesting to see the effect of the weighting factor, thus conducting the following experiments in Table 1. We observe that color rendering can benefit semantics but color rendering is not sensitive to semantics. Furthermore, Table 1 shows that the semantic loss alone can also drive our model to learn reasonable contextual geometry for semantic information as visualized in Figure 1.

λ_1/λ_2	1/0	0.75/0.25	0.5/0.5	0.25/0.75	0/1
PSNR	17.49	25.26	25.35	26.24	26.57
mIoU(%)	55.10	56.51	57.15	58.12	3.62

Table 1. Different weighting factors effect under ScanNet [3] generalization settings.

Effectiveness of the CRA module. To further validate the computational effectiveness of our Cross-Reprojection Attention (CRA) module, we provide the comparisons with Dense Attention in FLOPs and Memory usage.

Table 2 and Table 3 show the computational performance of real data by adopting different settings of our Cross-Reprojection Attention (CRA) module. We observe that directly applying the dense attention over multi-view

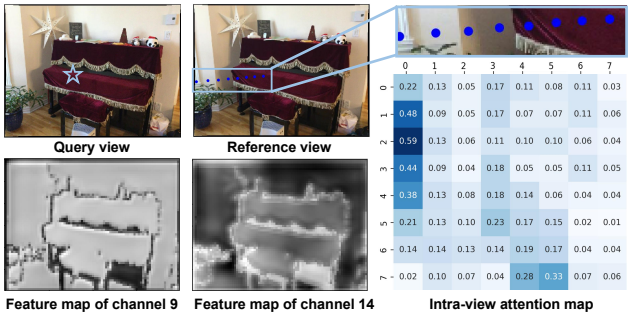

Figure 1. Visualization of 2D CNN features from ResUnet and intra-view attention map. It shows that our ResUnet can help S-Ray learn reasonable geometry for contextual semantics and the intra-view attention map is closely related to the visibility.

Description	GFLOPs	mIoU(%)	Total Acc(%)
w/o CRA	0	76.30	86.02
Dense Attention	10.25	90.46	94.52
only intra-view Att	3.05	81.24	89.58
only cross-view Att	2.35	87.01	93.34
full CRA	**5.40**	**91.08**	**98.20**

Table 2. Performance on real data [3] for different settings of Cross-Reprojection Attention module (CRA). FLOPs increments are estimated for the input of $1024 \times 64 \times 8 \times 32$.

Description	Memory(MB)	mIoU(%)	Total Acc(%)
w/o CRA	0	76.30	86.02
Dense Attention	17647	90.46	94.52
only intra-view Att	3899	81.24	89.58
only cross-view Att	1583	87.01	93.34
full CRA	**4143**	**91.08**	**98.20**

Table 3. Performance on real data [3] for different settings of Cross-Reprojection Attention module (CRA). Memory increments are estimated for an input of $1024 \times 64 \times 8 \times 32$.

reprojected rays suffers from heavy computational cost and high GPU memory. In contrast, our CRA module can achieve the comparable performance of dense attention with friendly GPU memory and high computational efficiency. Specifically, the design of CRA can improve the performance by 47.3% in FlOPs and 76.5% in GPU memory. These results prove that the proposed cross-reprojection attention can achieve high mIoU and total accuracy by capturing dense and global contextual information with computational efficiency.

Semantic ray construction. To construct the final semantic ray in Section 3.4 of our paper, we assign distinct weights to different source views and compute the semantic ray with semantic consistency. Specifically, we design the *Semantic-aware Weight Network* to rescore the significance of each view with a hyperparameter τ, as

$$\mathbf{w} \in \mathbb{C}(\tau) := \left\{ \mathbf{w} : 0 < \frac{\tau}{m} < w_i < \frac{1}{\tau m}, \sum_{i=1}^{m} w_i = 1 \right\}, \tag{3}$$

where $\mathbf{w}$ is the view reweighting vector with length m indicating the importance of source views. Instead of mean aggregation which ignores the different significance of different source views, the hyperparameter τ controls the semantic awareness of each view. The effectiveness of τ can be seen in Table 4.

hyperparameter τ	mIoU(%)	Total Accuracy(%)	Average Accuracy(%)
1	54.21	77.13	59.05
0.8	56.33	78.01	60.37
0.7	**57.15**	**78.24**	**62.55**
0.5	55.70	76.64	60.80
0.2	54.03	77.25	61.34

Table 4. Performance on real data [3] for different settings of hyperparameter τ in test set.

From Table 4, we observe that we can improve the performance of semantic segmentation by assigning different weights to each source view with hyperparameter τ. Note that $\tau = 1$ means the mean aggregation operation.

Training process. Given multiple views of a scene, we construct a training pair of source and query view (*i.e.*, target view) by first randomly selecting a target view, and sampling 8 nearby but sparse views as source views. We follow [4] to build our sampling strategy. The performance of our method in different training iterations can be found in Table 12. The results show that we only require 260k iterations for 20 hours to train our S-Ray over 60 different real scenes, which demonstrates the efficiency and effectiveness of our network design.

More comparisons with Semantic-NeRF. To further show our strong and fast generalizability in a novel unseen scene, we compare our performance with Semantic-NeRF [10] in per-scene optimization. The results are shown in Table 13. While Semantic-NeRF [10] needs to train one independent model for an unseen scene, we observe that our network S-Ray can effectively generalize across unseen scenes. What's more, our direct result can be improved by fine-tuning on more images for only 10 min (2k iterations), which achieves comparable quality with Semantic-NeRF for 100k iterations per-scene optimization. Moreover, Semantic-NeRF shows very limited generalizability by first generating pseudo semantic labels for an unseen scene with a pretrained model, and then training on this scene with the pseudo labels. In this way, Semantic-NeRF is able to apply to new scenes without GT labels. In contrast, our S-Ray provides stronger generalization ability by enabling directly test on unseen scenes. We provide additional experiments

in Table 5.

Comparison with GPNR. The recent work GPNR [8] also generates novel views from unseen scenes by enabling cross-view communication through the attention mechanism, which makes it a bit similar to our S-Ray. To further justify the motivation and novelty, we summarize several key differences as follows. **Tasks:** GPNR utilizes fully attention-based architecture for color rendering while our S-Ray focuses on learning a generalizable semantic field for semantic rendering. **Embeddings:** GPNR applies three forms of positional encoding to encode the information of location, camera pose, view direction, etc. In contrast, our proposed S-Ray only leverages image features with point coordinates without any handcrafted feature engineering. In this sense, our S-Ray enjoys a simpler design in a more efficient manner. **Training cost.** While GPNR requires training 24 hours on 32 TPUs, S-Ray only needs a single RTX3090-Ti GPU with similar training time.

	w/o ft	ft 5k(p)	ft 5k(gt)	ft 50k(p)	ft 50k(gt)	ft converge(p)	ft converge(gt)
S-NeRF	N/A	78.59	86.32	85.64	91.33	92.10	95.24
S-Ray	77.82	88.07	93.40	91.25	95.15	92.43	95.39

Table 5. More mIoU comparisons with SemanticNeRF(S-NeRF) in the scene0160-01 from ScanNet. Same with S-NeRF, we choose pretrained DeepLabV3+ [2] to generate pseudo semantic labels for finetuning. "p" means finetuning with pseudo labels, and "gt" means finetuning with ground truth.

More discussion for reconstruction quality. To further demonstrate the reconstruction quality and generalizability of S-Ray, we evaluate S-Ray with NeuRay [4], MVS-NeRF [1], and IBR-Net [9] on two typical benchmark datasets (*i.e.*, Real Forward-facing [5] and Realistic Synthetic 360° [6]) in Table 6. In general, Table 6 shows our Cross-Reprojection Attention module is also useful for generalizable NeRFs with out semantic supervision. While

	Realistic Synthetic 360°			Real Forward-facing		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
MVSNeRF	23.46	0.851	0.172	22.93	0.794	0.260
IBRNet	24.52	0.874	0.158	24.17	0.802	0.215
NeuRay	26.73	0.908	0.123	25.35	0.824	0.198
S-Ray(Ours)	**26.84**	**0.917**	**0.115**	**25.68**	**0.833**	**0.180**

Table 6. Quantitative comparisons of scene rendering in the generalization setting. All generalization methods including our method are pretrained on the same scenes and tested on unseen test scenes.

the three mentioned methods in Table 6 and our method are image-based rendering, the main difference lies in how to extract useful features: (a) MVS-NeRF leverages cost volume to extract geometry features, which benefits the acquisition of density; IBRNet performs feature attention on rays in 3D space and NeuRay further extracts the occlusion-aware features by explicitly modeling occlusion. Their features are sparse in 3D space but sufficient for color rendering. (b) Our method goes back to the 2D reprojection space and obtains dense attention by cascading two sparse attention modules, thus extracting rich semantic and geometric features. A key point is that we apply a ResUnet segmentation network fro context feature extraction to get semantic priors, which is not present in the previous methods.

Disscusion of the number of source views. We observe that using more source views on our S-Ray model can improve semantic rendering quality. The results are shown in Table 7. The reason is that adding more reference views in training means leveraging more contextual information for semantic feature learning to build a larger 3D contextual space and reconstruct the final semantic ray, which improves the view consistency and accuracy of semantic segmentation.

N_s	mIoU(%)	Total Acc(%)	Avg Acc(%)	PSNR	SSIM
1	67.55	86.15	73.73	26.47	0.9077
4	75.41	90.51	81.06	30.90	0.9368
8	79.97	93.06	84.92	**29.52**	**0.9106**
12	83.21	93.89	88.07	28.57	0.8969
16	**84.84**	**94.33**	**89.78**	27.85	0.8859

Table 7. Performance(mIoU, Total accuracy, Average accuracy, PSNR, SSIM) on the real data scene [3] wiht different source view numbers N_s.

Disccusion of semantic-aware weight. In semantic ray construction, we learn the view reweighting vector $\mathbf{w}$ to rescore the significance of each source view. To further demonstrate the effectiveness of this rescore strategy, we show the example in Figure 2. The results show that $\mathbf{w}$ can distinct the different significance of different source views to the query semantic ray.

3. Network architecture

Semantic feature extraction. Given input views and a query ray, we project the ray to each input view and apply the semantic feature extraction module in Table 8 to learn contextual features and build an initial 3D contextual space. The details can be found in Table 8 and Section 3.2 in the paper.

Cross-Reprojection Attention. To model full semantic-aware dependencies from the 3D contextual space with computational efficiency, we design the Cross-Reprojection Attention module in Table 9 to learn dense and global contextual information, which can finally benefit the performance of semantic segmentation. The details of architecture and design can be found in Table 9 and Section 3.3 in the paper.

Type	Size/Channels	Activation	Stride	Normalization
Input 1: RGB images	-	-	-	-
Input 2: View direction differences	-	-	-	-
L1: Conv 7×7	$3, 16$	ReLU	2	Instance
L2: ResBlock 3×3	$16, 32, 32$	ReLU	$2, 1$	Instance
L3: ResBlock 3×3	$32, 64, 64$	ReLU	$2, 1$	Instance
L4: ResBlock 3×3	$64, 64, 64$	ReLU	$1, 1$	Instance
L5: ResBlock 3×3	$64, 128, 128$	ReLU	$2, 1$	Instance
L6: ResBlock 3×3	$128, 128, 128$	ReLU	$1, 1$	Instance
L7: ResBlock 3×3	$128, 128, 128$	ReLU	$1, 1$	Instance
L8: ResBlock 3×3	$128, 128, 128$	ReLU	$1, 1$	Instance
L9: ResBlock 3×3	$128, 128, 128$	ReLU	$1, 1$	Instance
L10: ResBlock 3×3	$128, 128, 128$	ReLU	$1, 1$	Instance
L11: Conv 3×3	$128, 64$	-	1	Instance
L12: Up-sample $2\times$	-	-	-	-
L13: Concat (L12, L4)	-	-	-	-
L14: Conv 3×3	$128, 64$	-	1	Instance
L15: Conv 3×3	$64, 32$	-	1	Instance
L16: Up-sample $2\times$	-	-	-	-
L17: Concat (L16, L2)	-	-	-	-
L18: Conv 3×3	$64, 32$	-	1	Instance
L19: Conv 1×1	$32, 32$	-	1	Instance
L20: Reprojection				
L21: MLP (Input 2)	$4, 16, 32$	ELU	-	-
L22: Add (L21, L20)	-	-	-	-

Table 8. Semantic feature extraction.

Type	Feature dimension	Activation
Input: Initial 3D contextual space	-	-
L1: Transpose (Input)	-	-
L2: Position Embeddings	-	-
L3: Add (L1, L2)	-	-
L4: Multi-head Attention (nhead=4) (L3)	32	ReLU
L5: Transpose (L4)	-	-
L6: Multi-head Attention (nhead=4) (L5)	32	ReLU

Table 9. Cross-Reprojection Attention module.

Type	Feature dimension	Activation
Input 1: Initial 3D contextual space	-	-
Input 2: View direction differences	-	-
L1: Concat (Input 1, Input 2)	-	-
L2: MLP (L1)	$37, 16, 8, 1$	ELU
L3: Sigmoid (L2)	-	-

Table 10. Semantic-aware weight network.

Semantic-aware weight network. To construct the final semantic ray from refined 3D contextual space and learn the semantic consistency along the ray, we introduce the semantic-aware weight network in Table 10 to rescore the significance of each source view. More experiments about the semantic-aware weight net can be found in Table 4, and we show architecture details in Table 10 and Section 3.4 of the paper.

Geometry-aware network. To build our generalizable semantic field, we adopt a geometry-aware network to predict

density σ and render the final semantic field with semantic logits. Moreover, we also leverage this network to produce the radiance and render a radiance field to show our rendering quality. We show the details of this network in Table 11 and Section 3.4 of the paper.

References

[1] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, pages 14104–14113. IEEE, 2021. 3

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 3

[3] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 1, 2, 3, 6, 7, 8, 9

[4] Yuan Liu, Sida Peng, Lingjie Liu, Qianqian Wang, Peng Wang, Christian Theobalt, Xiaowei Zhou, and Wenping Wang. Neural rays for occlusion-aware image-based rendering. In *CVPR*, pages 7814–7823. IEEE, 2022. 2, 3, 8

[5] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019. 3

[6] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 3, 8

[7] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard A. Newcombe. The replica dataset: A digital replica of indoor spaces. *CoRR*, abs/1906.05797, 2019. 1

[8] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Generalizable patch-based neural rendering. In *ECCV*. Springer, 2022. 3

[9] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas A. Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021. 3

[10] 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, 7, 9

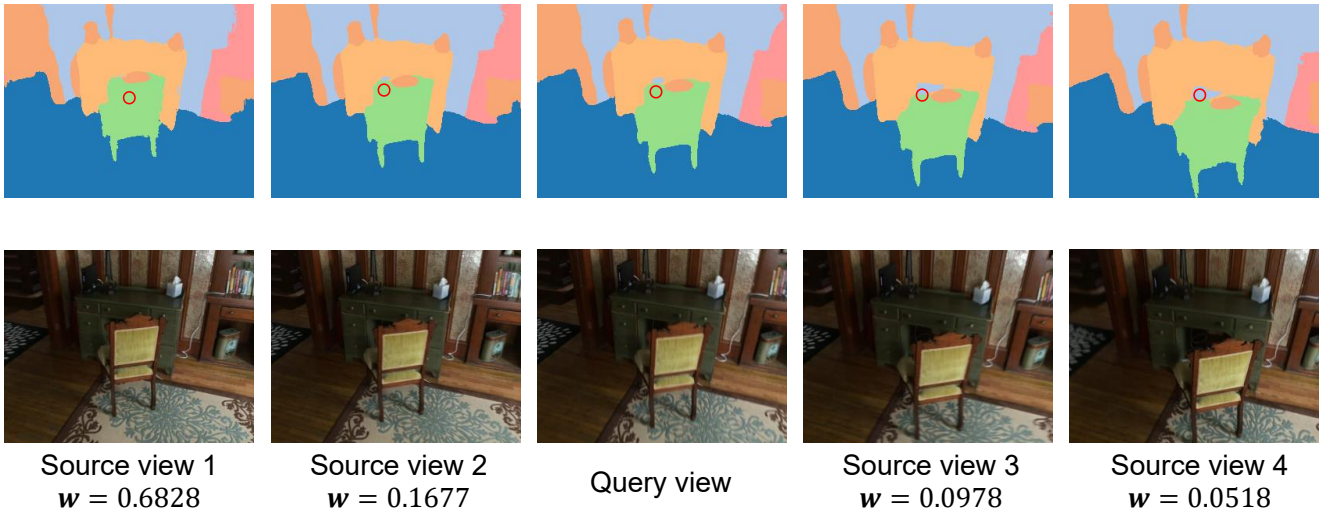

Figure 2. Different significance weight of source view. Given the query ray, we apply the semantic-aware weight network to learn the significance weight $\mathbf{w}$ to restore each source view. Note that the greater weight will be assigned to the more important source view.

Type	Feature dimension	Activation
Input: Initial 3D contextual space	-	-
L1: MLP (Input)	$32, 32$	ELU
L2: MLP (Input)	$32, 1$	ELU
L3: Sigmoid (L2)	-	-
L4: Dot-product (L1, L3)	-	-
L5: Cross-view Mean (L4)	-	-
L6: Cross-view Varience (L4)	-	-
L7: Concat (L5, L6)	-	-
L8: MLP (L7)	$64, 32, 16$	ELU
L9: Multi-head Attention (nhead=4) (L8)	16	ReLU
L10: MLP (L9)	16	ELU
L11: MLP (L10)	1	ReLU

Table 11. Geometry-aware network.

Method	Validation Set			Test Set		
	mIoU(%)	Total Acc(%)	Avg Acc(%)	mIoU(%)	Total Acc(%)	Avg Acc(%)
S-Ray (10k iters)	63.70	85.70	71.86	48.53	74.75	56.55
S-Ray (50k iters)	72.85	88.72	79.52	52.32	77.31	59.38
S-Ray (100k iters)	81.25	94.80	86.84	54.27	79.13	61.76
S-Ray (200k iters)	89.31	97.91	**91.10**	54.44	76.46	60.63
S-Ray (260k iters)	**89.57**	**98.54**	91.02	**57.15**	**78.24**	**62.55**
S-Ray (300k iters)	88.99	98.40	90.39	55.84	77.67	62.15

Table 12. Quantitative results (mIoU, total accuracy, average accuracy) of our method (S-Ray) in training process from multiple scenes in real dataset [3].

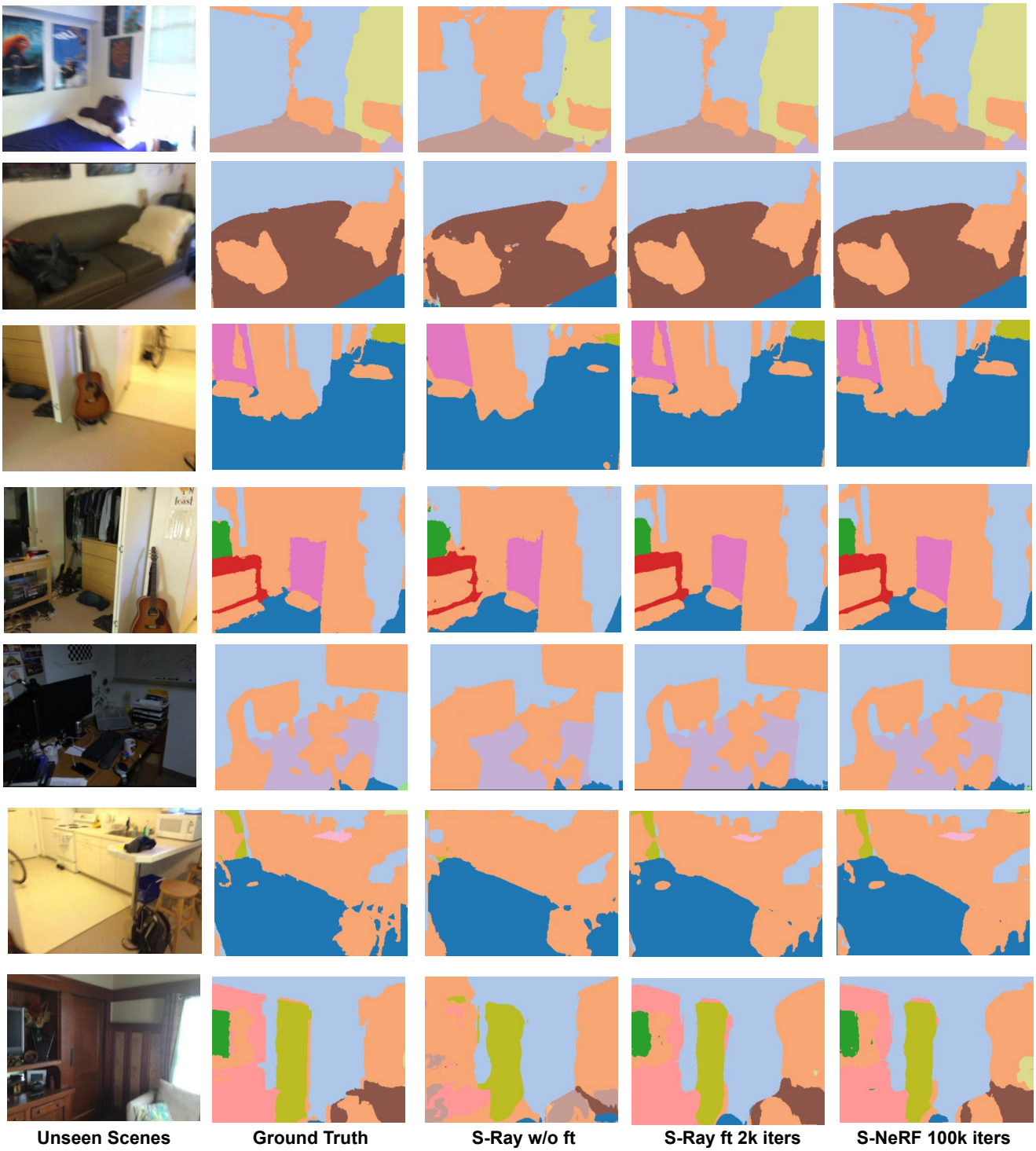

Figure 3. Additional semantic rendering quality comparison. More qualitative comparisons between our method S-Ray and non-generalizable method Semantic-NeRF [10] (S-NeRF for short) for semantic rendering in real data [3].

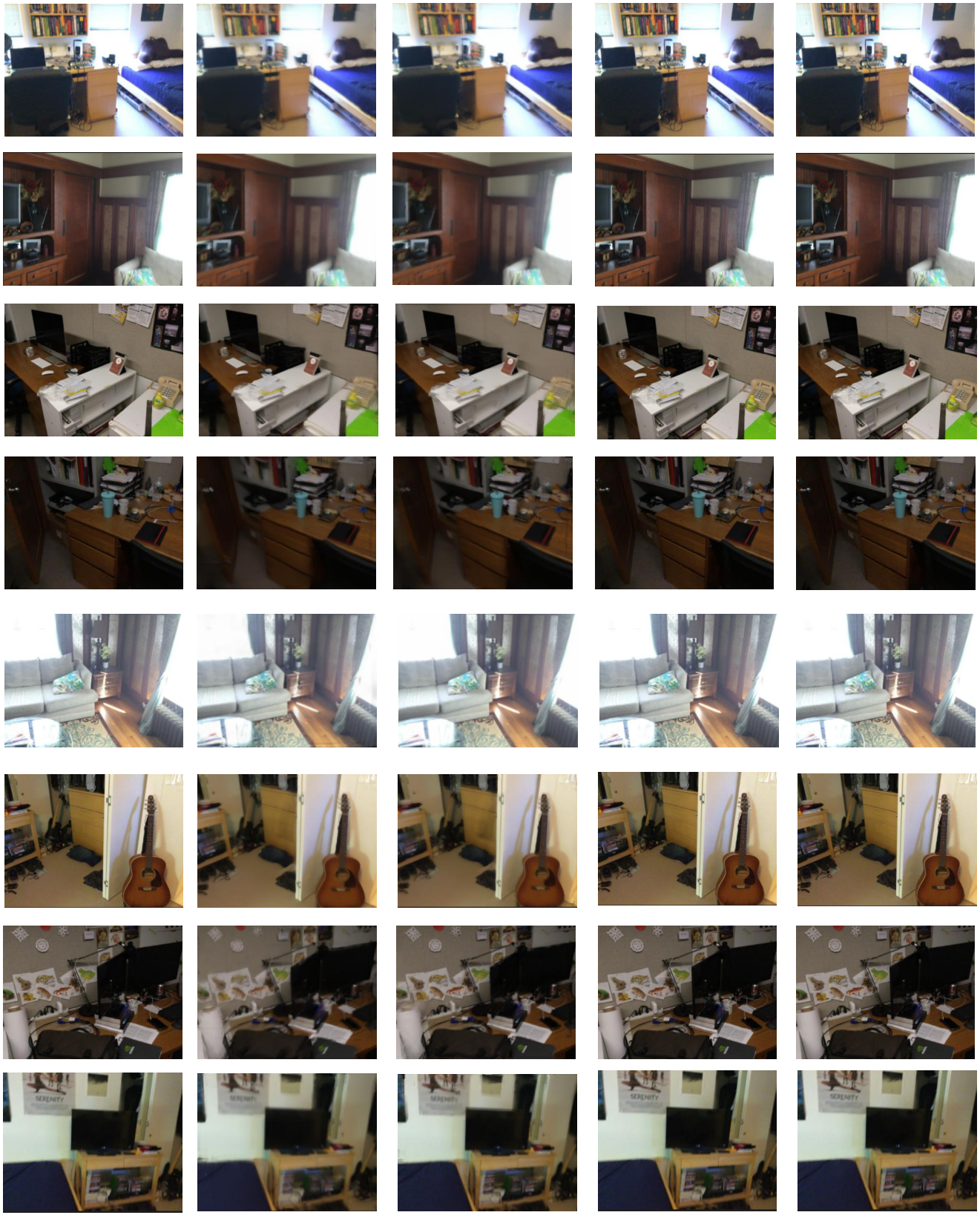

Figure 4. Qualitative results of scene rendering for generalization (w/o ft) and fine-tuning settings (ft) in real data [3]. Adding a color head from the geometry-aware network, We compare our method S-Ray with the generalizable rendering method NeuRay [4] and Valina NeRF [6] with 200k iterations.

Steps	Method	mIoU(%)	Average Accuracy(%)	Total Accuracy(%)	PSNR
0	Ours	77.22	81.68	88.53	29.47
	Semantic NeRF	-	-	-	-
2k	Ours	92.66	98.73	98.73	29.80
	Semantic NeRF	78.32	82.69	85.81	20.62
4k	Ours	93.40	98.97	98.97	29.86
	Semantic NeRF	86.97	86.61	87.48	21.85
6k	Ours	94.17	99.06	99.06	29.92
	Semantic NeRF	87.08	87.85	88.01	22.62
8k	Ours	94.59	99.15	99.15	29.95
	Semantic NeRF	88.78	88.57	89.67	22.94
30k	Ours	95.10	99.43	99.42	30.05
	Semantic NeRF	91.78	94.86	95.78	24.78
100k	Ours	-	-	-	-
	Semantic NeRF	95.05	98.73	99.02	29.97

Table 13. Performance of per-scene optimization in ScanNet [3]. We compare our method S-Ray with Semantic-NeRF [10] in per-scene optimization to show our fast generalizability in real data. Specifically, we choose the unseen scene0160_02 for comparison.