Nerflets: Local Radiance Fields for Efficient Structure-Aware 3D Scene Representation from 2D Supervision Supplementary Material

Xiaoshuai Zhang[1,3] Abhijit Kundu[1] Thomas Funkhouser[1] Leonidas Guibas[1,2]
Hao Su[3] Kyle Genova[1]

[1] Google Research [2] Stanford University [3] University of California, San Diego

A. More Details

A.1. Model Architecture

For all nerflet MLPs f_i, we follow the NeRF architecture [6] but reduce the number of hidden layers from 8 to 4, and reduce the number of hidden dimensions from 256 to 32. We also removed the shortcut connection in the original network. All other architecture details are as in [6]. The background neural field uses NeRF++ [9] style encoding, and its MLP f_{far} has 6 hidden layers and 128 hidden dimensions. We perform coarse-to-fine sampling as in [6], but both coarse and fine samples are drawn from the same MLP, not two distinct ones.

A.2. Hyper-parameters

We use $N = 512$ nerflets for all experiments in the main paper. The scaling parameter η is set to 5 for all experiments. We initialize the temperature parameter τ to 1 and multiply τ by 0.9 after each epoch. The smooth decay factor ϵ is set to 10^{-7} for all experiments. We draw 64 samples for the coarse level and 128 samples for the fine level within the bounding box. For unbounded scenes, we draw 16 coarse samples and 16 fine samples from the background MLP. We increase the weight for $\mathcal{L}_{\text{rgb}}$ from 0.0 to a maximum of 1.0, increasing the value by 0.2 after each of the first five epochs to prevent early overfitting to high-frequency information. Contrastive ray pairs are sampled within a 32×32 pixel window. The weight for the regularization loss $\mathcal{L}_{\text{reg}}$ is set to 0.1. All other losses ($\mathcal{L}_{\text{sem}}, \mathcal{L}_{\text{ins}}, \mathcal{L}_{\text{density}}, \mathcal{L}_{\text{radii}}, \mathcal{L}_{\ell_1}, \mathcal{L}_{\text{box}}$) have a weight factor 1.0.

A.3. Dataset Details

For training on each ScanNet scene, we uniformly sample 20% of the RGB frames for training and 10% of the RGB frames for evaluation– about 200 frames for training and 100 frames for evaluation. For both ScanNet and KITTI-360 scenes, we estimate the scene bounding box from the camera matrices and near-far values of the scene. Specifically, for each scene, we shoot rays from the cameras and calculate the minimum bounding box which could cover all near-far ray segments. Then we rescale the coordinate inputs within the box to $[-0.5, 0.5]$ for all experiments.

One important note about the ScanNet [2] experiments is that 2D ScanNet supervision indirectly comes from 3D. That is because the 2D ScanNet dataset was made by rendering the labeled mesh into images. We do not use this 2D ground truth directly, but PSPNet [10] is trained on it. Here, this is primarily a limitation of the evaluation rather than the method– there are many 2D models that can predict reasonable semantics and instances on ScanNet images, but we want to be able to evaluate against the exact classes present in the 3D ground truth. This does not affect the comparison to other 2D supervised methods, as all receive their supervision from the same 2D model. By comparison, KITTI-360 results are purely 2D only, but all quantitative evaluations must be done in image space.

A.4. Paper Visualization Details

For ScanNet mesh extraction, we create point samples on a grid and evaluate their density, semantic and instance information from nerflets. We then estimate point normals using the 5 nearest neighbors and create a mesh with screened Poisson surface reconstruction [4]. The mesh triangles are colored according to the semantic and instance labels of their vertices. For the teaser and KITTI-360 visualizations of our learned nerflets representation, we visualize nerflets according to their influence functions. We draw ellipsoids at influence value $e^{-\frac{1}{2}} \approx 0.607$.

A.5. Interactive Visualizer Details

Our interactive visualizer allows real-time previewing of nerflet editing results while adjusting the bounding boxes of objects in the scene. The visualizer draws the following components. First, a volume-rendered RGB or depth image at an interactive resolution of up to 320x240. This enables viewing the changes being made to the scene in real-time.

Second, the nerflets directly, by rendering an ellipsoid per nerflet at a configurable influence threshold. This enables seeing the scene decomposition produced by the nerflets. Third, a dynamic isosurface mesh extracted via marching cubes that updates as the scene is edited, giving some sense of where the nerflets are in relation to the content of the scene. Fourth, a set of bounding box manipulators, one per object instance, with draggable translation and rotation handles. These boxes are instantiated by taking the bounding box of the ellipsoid outline meshes for all nerflets associated with a single instance ID. A transformation matrix that varies per instance is stored and pushed to the nerflets on each edit.

Most of the editor is implemented in OpenGL, with the volume rendering implemented as a sequence of CUDA kernels that execute asynchronously and are transferred to the preview window when ready. In Table 3, we report performance numbers for top-1 evaluation, which is often the right compromise for maximizing perceived quality in a given budget (e.g., pixel count can be more important), though interactive framerates with top-16 or top-3 evaluation are possible at somewhat lower resolutions.

A.6. Instance Label Assignment

To assign instance labels for each nerflet, we render the nerflet influence map W_i for each view and compare with corresponding 2D semantic and instance segmentation maps to match each l_i^j (2D object instance or stuff with local id j in view i) to a set of nerflets $M(l_i^j)$. Here $M(\cdot)$ maps an instance ID to its set of associated nerflets. We then create a set of 3D instances $G = \{g_k\}$ according to the segmentation result of the first view – we create a 3D global instance for each detected 2D object instance. For each new view i, we match l_i^j to the 3D instance g_k if $|M(l_i^j) \cap M(g_k)|/|M(l_i^j)| \geq \delta$, and then update $M(g_k)$ to $M(g_k) \cup M(l_i^j)$. If no match is found in $\{g_k\}$, we create a new 3D instance and insert it into G. An illustration of this step could be seen in Fig. 1. Before inserting any new global instance g, we remove all nerflets that are already covered by the global set G from g. By using this first-come-first-serve greedy strategy we always guarantee no nerflet is associated with 2 different global instances. After this step, each nerflet is associated with a global instance ID, and our representation can be used to reason at an instance level effectively.

B. More Results

The following experiments are done on the subset of ScanNet from [1].

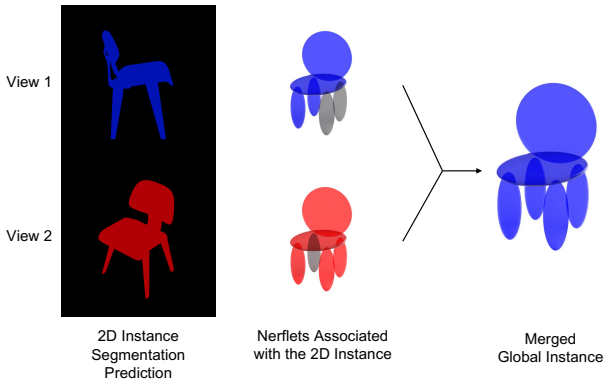

Figure 1. Illustration of instance label assignment. Instance predictions from all views are matched and merged iteratively into global instances.

	PSNR	mIOU
$n = 64$	26.34	53.23
$n = 128$	28.34	62.41
$n = 256$	28.81	69.97
$n = 512$	29.12	73.63
$n = 1024$	29.19	74.09

Table 1. Ablation experiment on ScanNet for different numbers of nerflets.

B.1. Number of Nerflets

We perform an ablation study on the number of nerflets on ScanNet. The results are in Tab. 1. We find that increasing the number of nerflets monotonically improves both the photometric and semantic metrics. However, the improvement above 512 nerflets is marginal, at least on this dataset. To balance performance and efficiency, we use 512 nerflets.

B.2. Trade-off between the Number of Nerflets and the MLP Size

Here we present an experiment investigating the trade-off between the number of nerflets vs. the size of the MLPs while keeping the number of the parameters comparable. As shown in Table 2, our setting (512x15k) achieves a balance between network capacity and decomposition granularity. The performance gain is marginal if we further increase the number of nerflets used or MLP size, and when using 2048 nerflets, we see a performance drop in PSNR possibly due to the limited representation power of each nerflet.

B.3. Effect of Top-k Evaluation

We perform an ablation study on the quality impact of k when we only evaluate the nerflets within the highest k influence weights for each point sample. The results are in Tab. 3. We find that evaluating 32 versus 16 nerflets has little influence on the model performance since each nerflet

	64	128	256	512	1024	2048
3k	23.89	26.23	27.91	28.08	28.49	28.99
7k	26.86	27.93	28.02	28.59	29.15	-
15k	27.98	28.09	28.43	29.12	29.18	-
32k	28.12	28.34	28.89	29.15	-	-
65k	28.39	28.56	29.01	-	-	-
132k	28.49	28.71	-	-	-	-

Table 2. Performance comparison of different design choices (number of nerflets and MLP sizes). "-" indicates failed runs due to OOM or divergent optimization. Our setting (512×15k, marked in red) achieves a balance between model performance and size.

	PSNR	mIOU
$k = 32$	29.13	73.72
$k = 16$	29.12	73.63
$k = 3$	29.05	72.95
$k = 1$	28.35	70.73

Table 3. Ablation experiment on ScanNet for evaluating only nerflets with top-k influence weights during training and testing.

only contributes locally. From 16 to 3, there is little change in PSNR but some change in mIoU. We see moderate performance drops in both metrics when only evaluating one nerflet with the highest influence weight. We use $k = 16$ for all experiments in the paper, though it is possible some intermediate value provides an even better balance.

B.4. Inactive Nerflets

One known problem [3] with training using RBFs that have learned extent is that when an RBF gets too small or too far from the scene, it does not contribute to the construction results. The radii loss $\mathcal{L}_{\text{radii}}$ and box loss $\mathcal{L}_{\text{box}}$ are proposed to alleviate this issue. To estimate the actual number of inactive nerflets, we utilize the nerflet influence map W and count nerflets that do not appear on any of these maps in any view. In KITTI-360 experiments, we estimate to have 10.6 inactive nerflets on average per scene, making up $\sim$ 2.07% of all available nerflets. In ScanNet experiments in the main paper, we estimate to have 30.6 inactive nerflets on average per scene, making up $\sim$ 5.98% of all available nerflets.

B.5. ScanNet 2D Segmentations

In Figure 2, we visualize more examples on ScanNet comparing our panoptic predictions with reference annotations from the dataset. It can be seen that our representation learned from 2D supervision contains rich information and can produce more accurate segmentation results than reference maps in some cases. Our method produces clearer boundaries, fewer holes, and discovers missing objects in the reference results thanks to its ability to fuse segmentation maps from multiple views with a 3D sparsity prior to the structure of the representation.

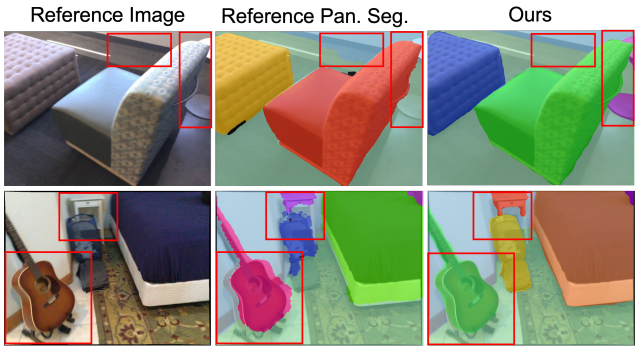

Figure 2. Comparison of ScanNet reference panoptic segmentation maps and our panoptic segmentation predictions overlaid on reference images.

Method	Appearance PSNR	Semantics mIOU	Mem Footprint MB
SemanticNeRF [11]	21.09	72.89	5.8
Instant-NGP [7]	21.34	74.22	25.2
Ours	**21.68**	**74.87**	**1.2**

Table 4. Results on novel view color and semantic synthesis tasks on KITTI-360 [5] (our train-validation split). Nerflets achieve better quality while having much smaller memory footprint than both SemanticNeRF and Instant-NGP.

B.6. More Baselines on KITTI-360

In order to evaluate more baselines, we create our own training and validation splits on the provided training data of KITTI-360. We evaluate Nerflets, Instant-NGP, and SemanticNeRF on these splits. The results are reported in Tab. 4.

The results show that Nerflets outperform SemanticNeRF and Instant-NGP in both RGB and semantic quality. Also, both methods consume more memory and do not produce panoptic segmentation maps.

B.7. Initialize from 3D Geometry

Our method can benefit from additional 3D inputs. For example, given a point cloud of the scene, we can initialize the nerflets around object surfaces. To validate the effect of using additional 3D inputs we perform experiments on 8 ToyBox-5 scenes [8], where 3D ground truth is available. For each scene, we sample 512 points on the ground-truth geometry with Poisson disk sampling and initialize the centers of the nerflets on these points.

According to the results on ToyBox-5 scenes (Fig. 3, we can achieve the same performance on a scene (PSNR $\approx$ 27.23) in $\sim$10k steps when initializing with a point cloud versus $\sim$25k steps when initializing randomly. However, the converged final performance is similar.

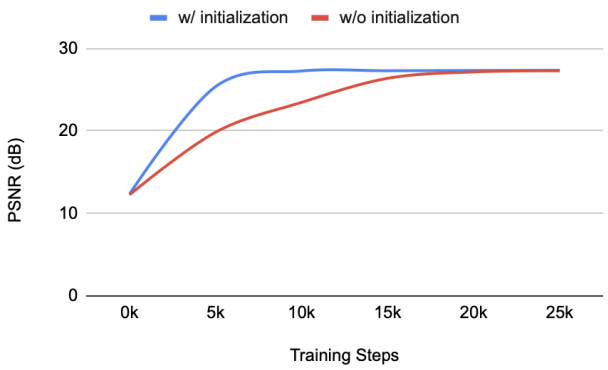

Figure 3. Validation PSNR of models trained with vs. without 3D initialization. The final converged performance is similar, but the model learns much faster with proper geometry initialization.

B.8. Performance Variance

To evaluate the performance variance of our model, we fit each scene in our train split of KITTT-360 five times, and evaluate the performance on our validation split, and report the variability. On our KITTI-360 test split, the mean PSNR is 21.67 and the mean stddev on PSNR is 0.062. The mean mIOU is 75.03 and the mean stddev is 0.221. The results show small variance of our performance and indicate our optimization is stable across different runs.

References

[1] Wang Bing, Lu Chen, and Bo Yang. Dm-nerf: 3d scene geometry decomposition and manipulation from 2d images. *arXiv preprint arXiv:2208.07227*, 2022. 2

[2] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 1

[3] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *ICCV*, pages 7154–7164, 2019. 3

[4] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013. 1

[5] Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *arXiv preprint arXiv:2109.13410*, 2021. 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. *Communications of the ACM*, 65(1):99–106, 2021. 1

[7] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 3

[8] Suhani Vora, Noha Radwan, Klaus Greff, Henning Meyer, Kyle Genova, Mehdi SM Sajjadi, Etienne Pot, Andrea Tagliasacchi, and Daniel Duckworth. NeSF: Neural semantic fields for generalizable semantic segmentation of 3d scenes. *arXiv preprint arXiv:2111.13260*, 2021. 3

[9] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 1

[10] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 1

[11] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew Davison. In-place scene labelling and understanding with implicit scene representation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 3