Expanding KiloNeRF paradigm

Anas Shahzad Robin Brase Muhammad Khattab Aanchal Chugh

Abstract

Neural Radiance Fields (NeRF [4]) have emerged as a prominent method for high-quality novel view synthesis. However, their computational demands make them unsuitable for real-time usage. KiloNeRF [8] promises a significant speedup by splitting the scene into a voxel grid and using a tiny MLP for each voxel. However this method still requires the time consuming training of a NeRF model as a teacher. Furthermore, the improvements made by KiloNeRF are not enough for real-time execution. This project aims to improve upon this design by making both inference and training faster and adding the semantic class as an additional output.

1. Multi-resolution rendering

1.1. Method

The performance of volumetric rendering depends on two factors: the number of samples and evaluation time. This chapter introduces a method to reduce evaluation time for a subset of samples. The hypothesis is that objects in the background can be rendered with less detail without sacrificing image quality significantly. To achieve this, we use smaller MLPs to evaluate points that are further away from the camera than a predefined threshold (see Fig. 6). Because of the lower number of parameters, the inference time should be shorter, leading to faster rendering.

1.2. Network architecture

The first part of the NeRF architecture consists of a positional encoding, where the input coordinates are mapped into a higher-dimensional space using trigonometric functions. By increasing the number of frequencies, the MLP is able to learn high-frequency details [4]. Since such high-frequency details are not visible for background objects, we reduce the number of these trigonometric functions to learn fewer high-frequency features. As concrete values we choose $L = 5$ for $\gamma(\mathbf{x})$ and $L = 2$ for $\gamma(\mathbf{d})$, which is a half of the original NeRF [4] and KiloNeRF [8] architectures. Furthermore, we have also tested a second architecture which reduces the size of the hidden layers to 256. We call these two new architectures *LowFrequency* (**LF**, Fig. 7a) and *LowFrequency_Small* (**LF_Small**, Fig. 7a).

1.3. Implementation

The official source code of KiloNeRF[1] provides two implementations of the method: one using PyTorch [7], which is differentiable and allows for training, and another highly efficient CUDA implementation with a hardcoded network architecture, achieving nearly real-time performance. We have included our method only in the PyTorch version due to time constraints to facilitate faster development cycles. However, we anticipate that the benefits of smaller MLPs will also be applicable to the CUDA implementation. Furthermore, our PyTorch implementation does not take advantage of all optimization possibilities. For instance, we use the same batch size for foreground and background points, although the lower memory consumption of the weight for background points would allow a larger batch size for them.

1.4. Experiments

1.4.1 Datasets

In total, we evaluated our method on 3 different datasets. The first dataset is a synthetic Lego scene from the original SyntheticNeRF dataset [4], featuring a Lego bulldozer on a small base plate, leading to fewer background points. In contrast, the scene Barn from TanksAndTemple [3] offers real photos taken all around a building. The greater size of the building and the continuous floor result in a more balanced object distribution. To further evaluate our method, we created a third dataset based on the Lego one, positioning the bulldozer on a larger disk surrounded by other Lego objects. This arrangement creates a more balanced relationship between the background and foreground objects. The colorgrid texture on the disk helps us assess the background's detail level (see Fig. 11).

1.4.2 Baseline

As comparison we use KiloNeRF [8] as a baseline, which is still used for foreground points in our method. As described above, our method is only implemented in PyTorch, which is why we also use the PyTorch variant of the KiloNeRF

[1] https://github.com/creiser/kilonerf

	Base	Base & LF_Small	Base & LF
PSNR ↑	27.58	27.44	27.53
Total render time [s] ↓	5.05	4.39	4.83
Network inference [s] ↓	2.52	1.72	2.20
Other processing [s] ↓	2.53	2.66	2.62

Table 1. Results for scene Barn, image resolution 1920x1080

implementation for performance comparisons. The number of MLPs and the resolution of the occupancy grid remain consistent across all architectures.

1.4.3 Results

As shown in the Tab. 1, our method achieves its goal of improving the performance without affecting the quality of the final image, except for small background objects seen only when zoomed in (Fig. 8). Additionally, the speed enhancement is not consistent; in cases with few background points, the overhead outweighs the savings, resulting in no significant speed improvement (Tab. 4). The full ablation study can be found in the appendix (Sec. 6.1).

2. Incorporating semantic information

2.1. Dataset

In this section, we aim to extend the KiloNeRF model to the ScanNet dataset [1] which is a diverse collection of indoor scenes of various office environments obtained by the Microsoft Kinect RGB-D scanner. To collect the data, the authors used Bundle fusion [2] at a resolution of 1 cm^3 to obtain accurate pose alignments. They followed up their pipeline with voxel hashing and the marching cubes algorithm to find a surface mesh. The dataset contains around 20 labels for each scene containing labels such as wall, chair and sofa etc. Segmentation of ScanNet environments is done via crowd-sourcing. ScanNet is the premier benchmark for 3D semantic segmentation and instance segmentation tasks due to the diversity of indoor environments. The raw color 3D scan and segmented 3D scans are available via .sens files from the following website.

2.2. Method

Our motivation in this task is to incorporate semantic information in KiloNeRF as well such that we have a one-by-one correspondence between the real 3D scan and 3D scan with a semantic label. We train two KiloNeRF models, one on the original 3D scan and the other on 3D scan with segmentation. To perform this task, we first had to create a dataset with uniformly sampled poses and RGB images for the KiloNeRF model to train on. To perform this task we used the open3d library in order to capture the following parameters:

- Extrinsic matrix (Poses) and Intrinsic matrix
- Bounding Box
- Near and far parameters
- Real and segmented RGB images

Points in the point cloud were enlarged to a size of two to fill in the empty spaces in the scene and we use ICP registration to align scans with partial overlap of the same scene. We took images of the segmented scene and the real scene around the x, y and z axis with angles of 0.62-radian separation. Finally, we divided the images into a 1:1:2 test, validation and train split. Due to the large number of computation constraints, we choose 10 random scenes from the ScanNet dataset. We trained our model on the Nvidia 3060 Ti for 29 hours and evaluated the results for each scene.

2.3. Results

The results for this task were not as expected with an average PSNR of 14.570 and MSE of 0.037159 over the image batch. Hence, the KiloNeRF model was not able to reconstruct the 3D scene for both real data and segmented data sufficiently. Looking at the occupancy grid we find that the inside of the scene is sparsely occupied while the outside of the scene is thickly occupied. This shows that KiloNeRF finds it very difficult to reconstruct detailed indoor environments. Another reason might be that the walls of the scenes are very thin and have some large spaces within them that reduce the partial overlap within these scenes. One strange thing our results also show is that for some scenes, the KiloNeRF model mirrors the scene instead of reproducing it. We could try increasing the number of epochs, increasing the resolution of the KiloNeRF model or sampling the ScanNet dataset with triangular mesh in order to improve our performance in the future.

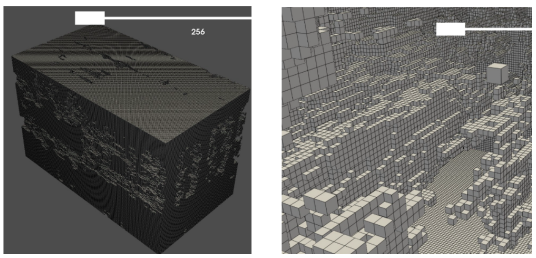


Figure 1. Occupancy grid for KiloNeRF

3. Combining KiloNeRF and DONeRF

3.1. Motivation

In order to achieve real-time rendering, we would like to address the high number of samples per ray that is being used by KiloNeRF [8] and NeRF [4] in general. The

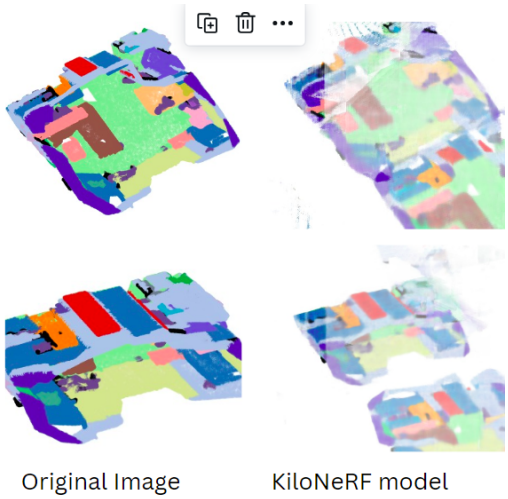


Figure 2. Results on ScanNet

	PSNR ↑	FLIP ↓
DONeRF	**35.23**	**0.048**
NeRF	35.19	0.051
Vairant	26.65	0.112

Table 2. Quality comparison on classroom scene.

(a) Ground truth (b) Estimated output

Figure 3. KiloNeRF variant estimated output vs. ground truth

original NeRF paper uses 256 samples per ray [4], while KiloNeRF uses 384 [8]. This is one of the reasons that is stopping NeRF from being used in real-time applications with the current limitations of hardware. On the other hand, DONeRF achieves real-time rendering using only 8 samples per ray and achieves high quality at the same time [6]. Therefore, we would like to combine KiloNeRF with the real-time rendering capability of DONeRF.

3.2. Introduction

DONeRF [6] uses a five stage pipeline, which takes RGBD input images for training. First they apply ray unification to disambiguate rays originating from different starting locations. Then they perform logarithmic non-linear sampling of the rays. This allows the network to focus on objects closer to the camera. Additionally, using the depth information during training can lead to reducing the number of samples even further .

The next stage of the pipeline is using sampling oracle network which classifies for each segment along the ray whether it should get samples or be skipped. The purpose of this network is to predict samples locations in order to avoid relying on surface representation and to achieve compactness. Then they apply space warping towards cell view center to avoid applying positional encoding equally for foreground and background, since the background contains high frequency that is not useful for reconstruction. The last stage is using Shading net which outputs RGBA values [6].

3.3. Method

In order to reduce the number of samples, we tried integrating the KiloNeRF into the 5 stage pipeline of DONeRF. First part of the integration included combining both environments for both projects along with custom CUDA code for the KiloNeRF. Then, we added the updated configuration classes to include the KiloNeRF additional parameters.

KiloNeRF three-stage pipeline includes which is explained in 4. With this pipeline, the abstraction and decoupling of its parts were not trivial and required more time and more understanding of the underlying parts. We proceeded with the integration for a while, however, it was not fully complete.

Instead, as a last-minute proof of concept, we provided a simple variant of KiloNeRF which uses only 2 MLPs instead of thousands of tiny MLPs. The simple variant provided was trained on Classroom dataset, for 30k epochs, using 128 samples per ray, with a linearly spaced ray march sampler with a step size of 1/128. Since the variation is still close to the original NeRF architecture, it still requires a long training time, unlike original DONeRF.

The dataset [2] used contains camera poses as well as images and depth maps. The dataset is split into 70% training, 10% validation, and 20% testing.

3.4. Results

The proposed KiloNeRF variant was integrated successfully and achieved decently given the low number of epochs compared to DONeRF, which use 300k for training each network, which is a total of 600k epochs [6]. However, since this is not the target architecture that we would like to integrate, we proceeded with 30k epochs, which suffice for elaboration of the idea. We present the results of the partial integration of the simple variant along with both NeRF and DONeRF, which use 128, 128, 8 samples per ray respectively while using log sampling and warp for all of them in table 2 as well as estimated output in 3.

[2] https://repository.tugraz.at/records/jjs3x-4f133

4. Instant NGP as a trainer model

4.1. Motivation

"Neural graphics primitives, when parameterized by fully connected neural networks, are notorious for being expensive in terms of training and evaluation [5]. " However, the introduction of Instant NGP [5] has demonstrated a massive speedup of several orders of magnitude. This has enabled training of higher-quality neural graphics primitives in just a few seconds while rendering at a resolution of 1920×1080 takes just tens of milliseconds. To leverage this speedup, the idea was to replace the trainer model used in the original KiloNeRF paper with instant NGP. By using this, we wanted to verify how the usage of instant NGP enables faster training and knowledge distillation.

4.2. Introduction

Below are the steps to understand the KiloNeRF training process in detail:

- Training the teacher model (i.e. original NeRF)
- Extracting the occupancy grid from the teacher model
- Knowledge distillation from teacher model to KiloNeRF (child model)
- Finetuning KiloNeRF to achieve better results

4.3. Implementation

To replace the original NeRF with instant NGP as the trainer model, the first step involved training of instant NGP model. There are many implementations of ngp and we made a choice of utilizing the implementation where *Instant-ngp (only NeRF) was trained in pytorch+cuda with pytorch-lightning (high quality with high speed)*[3].

Instant NGP accepts scene scale as a parameter which has a default value of 0.5. This scale adds a constrain that the whole scene must lie in $[-0.5, 0.5]^3$ which was not the case in the original Synthetic NeRF Lego dataset. Therefore, we added an additional pre-processing step to scale the input data appropriately before the training operation. We trained on the Lego dataset for 30 epochs, which was finished within 5 minutes.

Then comes the step of extracting occupancy grids. The given ngp model's forward function takes the points and direction subsets and returns the density as the first element of tuple. For knowledge distillation, it was quite similar to the way it was carried out with the original NeRF model which required few code changes handling input and output of the instant NGP model.

4.4. Results

As per the KiloNeRF paper [5], the training time of original NeRF on Synthetic NeRF Lego dataset takes approximately 32 hours. On the other hand, the training time for instant NGP is approximately 5 mins on the same dataset reducing the training time by a factor of 385. Figure 4 shows the comparison between the occupancy grid obtained from both the pre-trained teacher models: instant NGP and original NeRF. As seen in Fig 5, although we achieved a big reduction in training time, the color of the rendered images were subdued which could be due to few training steps.

[3]https://github.com/kwea123/ngp_pl

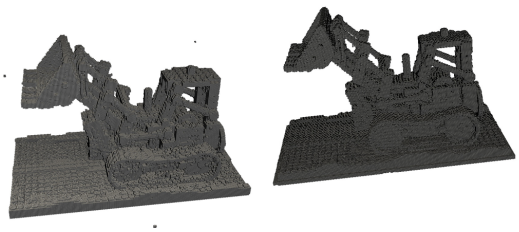


Figure 4. Occupancy grid from instant NGP (left) and NeRF (right)

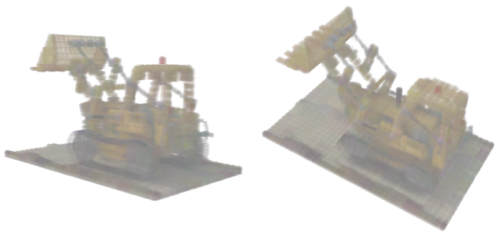


Figure 5. Rendered images of Synthetic NeRF Lego dataset

5. Conclusion

Despite our promising results, there are some issues that need to be investigated in future work. We have shown that we can both reduce the number of samples and the time for each sample individually. Future work could work on merging the two implementations to achieve an even bigger speedup. For actual real-time rendering, it would also be necessary to integrate our changes into the CUDA kernel. For the part involving ScanNet, we could try increasing the resolution of the KiloNeRF model while sampling the ScanNet dataset with a triangular mesh to improve our performance in the future. As a future step, instant NGP could be used to test the other features in less time. Although the results from training with instant NGP and NeRF models look similar (Fig. 4), future work can investigate the difference in quality in finer detail. Our additions to the KiloNeRF model show a lot of promise for future researchers and could eventually lead to a superior implementation.

References

[1] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. Scannet:

Richly-annotated 3d reconstructions of indoor scenes. *CoRR*, abs/1702.04405, 2017. 2

[2] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *CoRR*, abs/1604.01093, 2016. 2

[3] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. 1

[4] 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, Aug. 2020. arXiv:2003.08934 [cs]. 1, 2, 3

[5] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Transactions on Graphics*, 41(4):1–15, July 2022. arXiv:2201.05989 [cs]. 4

[6] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 40(4):45–59, July 2021. arXiv:2103.03231 [cs]. 3

[7] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. 1

[8] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs, Aug. 2021. arXiv:2103.13744 [cs]. 1, 2, 3

6. Appendix

6.1. Multi-resolution rendering

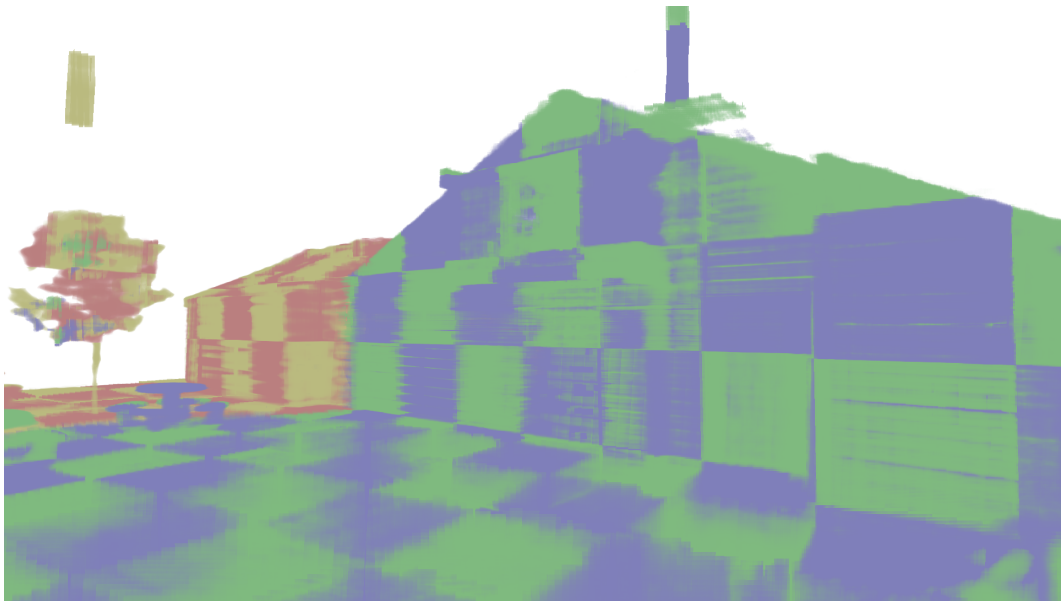
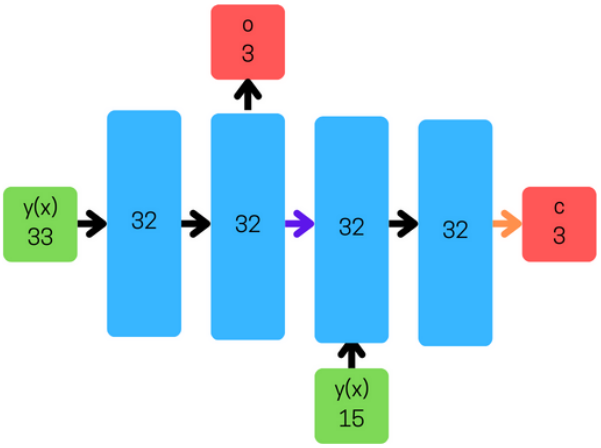
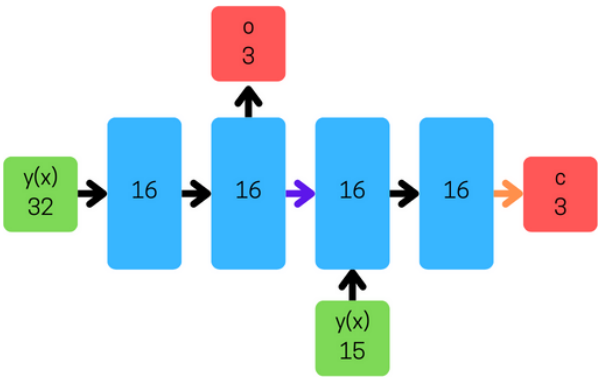


Figure 6. Network assignment for each sample. Green and blue samples are evaluated using the Baseline architecture, red and yellow ones using a smaller MLP (**LF** or **LF_Small**.)



(a) Architecture for the LowFrequency (**LF**) model.



(b) Architecture for the LowFrequency_Small (**LF_Small**) model.

Figure 8. Qualitative results for one image from the Barn scene.

	Base	Base & LF_Small	Base & LF	LF_Small	LF
PSNR	27.58	27.44	27.53	26.46	22.13
Total render time [s]	5.05	4.39	4.83	3.72	4.26
Network inference [s]	2.52	1.72	2.20	1.44	2.07
Other processing [s]	2.53	2.66	2.62	2.28	2.20

Table 3. Full results for scene Barn, image resolution 1920x1080

	Base	Base & LF_Small	Base & LF	LF_Small	LF
PSNR	30.95	30.55	30.91	30.71	27.57
Total render time [s]	0.75	0.84	0.86	0.74	0.72
Network inference [s]	0.088	0.099	0.107	0.082	0.058
Other processing [s]	0.66	0.74	0.75	0.66	0.66

Table 4. Full results for scene Lego, image resolution 600x600

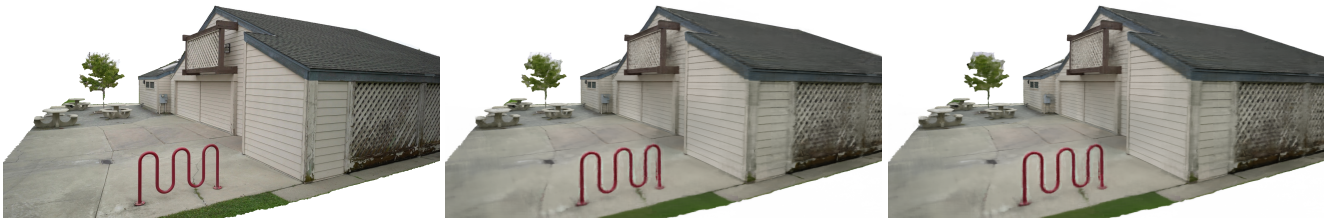
	Base	Base & LF_Small	Base & LF	LF_Small	LF
PSNR	20.782	20.778	20.774	19.760	20.716
Total render time [s]	0.81	0.769	0.787	0.686	0.697
Network inference [s]	0.109	0.064	0.077	0.040	0.059
Other processing [s]	0.701	0.705	0.710	0.646	0.638

Table 5. Full results for scene Lego UV, image resolution 600x600

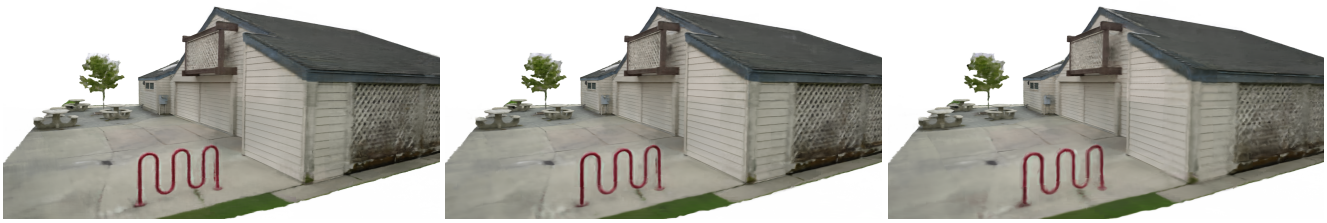
(a) Ground truth (b) Baseline (c) Baseline + LF_Small

(d) Baseline + LF (e) LF (f) LF_Small

Figure 9. Qualitative results for the Lego scene



(a) Ground truth (b) Baseline (c) Baseline + LF_Small



(d) Baseline + LF (e) LF (f) LF_Small

Figure 10. Qualitative results for the Barn scene

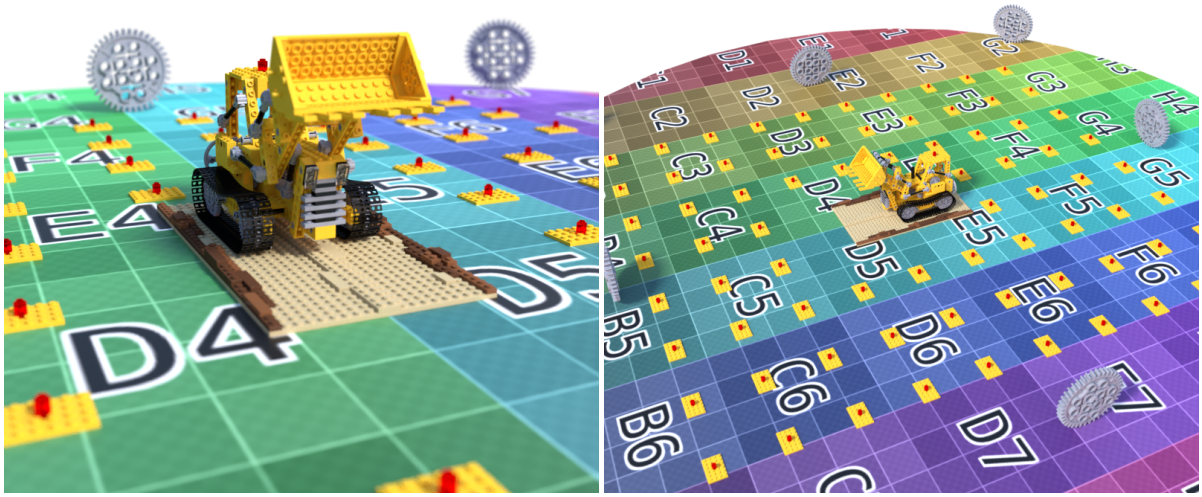


Figure 11. Example images from our own dataset.

(a) Ground truth (b) Baseline (c) Baseline + LF_Small

(d) Baseline + LF (e) LF (f) LF_Small

Figure 12. Qualitative results for the Lego UV scene

Table 6. Scannet statistics

Statistic	ScanNet
# of scans	1513
# of RGB-D frames	2,492,518
floor area (avg / sum m^2)	22.6 / 34,453
surface area (avg / sum m^2)	51.6 / 78,595
labeled objects (avg / sum)	24.1 / 36,213

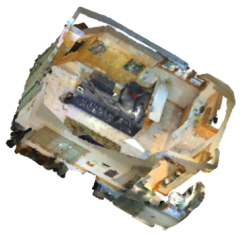


Figure 13. Example indoor environment for scannet

6.2. Combining KiloNeRF and DONeRF

(a) Ground truth

(b) Estimated output

Figure 14. KiloNeRF variant estimated output vs. ground truth

(a) Ground Truth

(b) Estimated Depth

Figure 15. KiloNeRF Variant estimated depth vs. ground truth depth