Supplementary material for
NeurMiPs: Neural Mixture of Planar Experts for View Synthesis

Zhi-Hao Lin[1,2] Wei-Chiu Ma[3] Hao-Yu Hsu[2] Yu-Chiang Frank Wang[2] Shenlong Wang[1]
[1]University of Illinois at Urbana-Champaign [2]National Taiwan University
[3]Massachusetts Institute of Technology

Abstract

In the supplementary material, we first visualize the training and testing views of dataset Replica in Section A. In addition, we then provide the runtime breakdown and implementation details in Section B, C. Finally, we provide more qualitative and quantitative results in Section D and E. The supplementary video "NeurMips-intro.mp4" briefly introduces our method and compares qualitative results with other works. Complete comparison videos of each scene are also provided under the folder "videos".

A. Camera visualization of Replica

We visualize the 3D scene, as well as the training and testing camera poses in Fig. A. Training camera poses are shown in blue, while testing poses are red. The RGBD images are unprojected to world coordinates with camera poses, representing the appearance and geometry of the 3D scene. Note that the testing views cover a much broader range than training views across all scenes, making this benchmark more challenging in terms of viewpoint variations. Specifically, we can utilize this dataset to evaluate different novel view synthesis models' performance regarding view interpolation, extrapolation, zoom in&out.

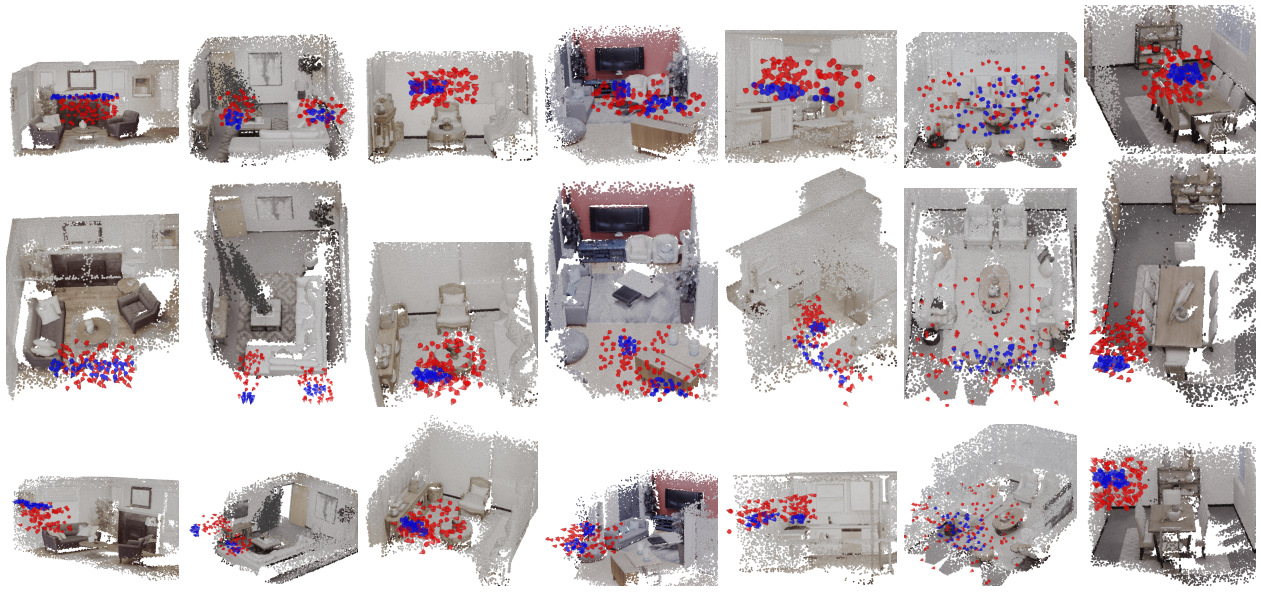

Figure A. **Camera visualization of Replica.** Blue: Training views, Red: Testing views.

B. CUDA optimization details

Intersection	Pre-processing	Network inference	Integration	Total time
0.0079	0.0252	0.0177	0.0030	0.0538

Table A. **Runtime breakdown (in seconds).** The runtime estimation is evaluated on Replica.

Table A reports the runtime breakdown of our complete rendering process averaging across all testing images in the Replica dataset. Specifically, we divide the entire rendering procedure into four components: (1) "Intersection": ray-plane intersection for all viewing rays; (2) "Pre-processing": operations including depth sorting, sampling from pre-baked alpha planes, and filtering sampled alpha values; (3) "Network inference": parallel inference of multiple planar experts; (4) "Integration": alpha-composition for all rays and background color replacement for rays with low alpha weight; All components are leveraged by customized CUDA kernels to significantly accelerate the rendering pipeline.

B.1. Network inference

In a vanilla MLP, for each linear layer, a matrix-matrix multiplication is processed by multiplying a $B \times I$ input matrix with a $I \times O$ weight matrix, where B refers to batch size, I is the size of input features and O is the size of output features of the associated layer. In our method, the original NeRF model distills into multiple planar experts, with each expert representing a planar surface with a tiny MLP. Therefore, it is reasonable that batch size B_i for each network i is not identical. Directly calling (torch.matmul) in a loop for each network in PyTorch might be a straightforward solution. However, it executes CUDA kernels for each matrix multiplication sequentially, does not fully utilize GPU's SMs, and therefore achieves bad performance. Although batched matrix multiplication (torch.bmm) seems like a better choice since it allows matrix multiplications for all networks in parallel with only a single CUDA kernel launch. However, this routine supported by PyTorch requires all batch sizes B_i to be the same. Hence, we develop a routine that fused the entire network evaluation (including Fourier features calculation, linear layers, and activation functions) into a single CUDA kernel. In our CUDA kernel, each CUDA block is responsible for a particular network and each thread is responsible for handling a single network input. Due to the small size of the individual networks ($\tilde{2}$5KB), parameters for a network can be fully loaded into on-chip RAM of an SM once per frame. All intermediate results are stored in per-thread registers to avoid latency of memory transfer.

B.2. Alpha prebaking, sampling & filtering

Network evaluation of all possible ray-plane intersected points is inefficient, therefore, we aim to filter out those points with low alpha values. First, each plane is baked in PyTorch by evaluating alpha values of uniformly grid-sampled points (200×200) on the plane beforehand. Then, for each ray-plane intersected point, we get an estimated alpha value by bilinear interpolation from neighboring alpha values on the plane. Finally, points with estimated alpha weight $\prod_{i}^{j-1}(1-\alpha_i)\alpha_j$ lower than a fixed threshold $= 0.001$ are discarded, which do not contribute much to alpha composition results. With this strategy, nearly 60% of points are filtered out on the Replica dataset, 70% on the Tanks & Temples dataset.

C. Baseline implementation details

In this section, we provide more implementation details of each baseline methods for dataset Replica.

NeRF [1] We used the PyTorch3D implementation of the original paper for training and evaluation. NeRF performs hierarchical sampling on each ray. It samples 64 points uniformly in the coarse stage and then samples another 64 points according to density distribution in the fine stage. It renders each pixel by evaluating RGB-α values of total 128 points.

NeX [3] We used the official implementation in our experiments. To construct the multiplane image, the model selects one of the training views as reference, which is the closest to the 3D center of all training camera positions. 192 planes are parallel and placed in front of this reference view, rendering novel views with homography warping.

PlenOctree [5] We used the official implementation in our experiments. After training NeRF with spherical harmonics (i.e. NeRF-SH), it is then converted into octree structure for real-time rendering. However, with default settings, dense grids

cannot be retained even on a 32G GPU, and the filtering strategy leads to several blank regions and low image quality of novel views. As a result, we extract with lower grid resolution $= 256^3$ (default: 512^3), and retain all grids for rendering. This removes blank regions and generates better image quality. Please see Table. B for qualitative comparison.

KiloNeRF [2] We used the official implementation in our experiments. It trains an ordinary NeRF model (teacher) then distills knowledge of teacher into multiple tiny MLPs. To avoid model querying in empty space, the occupancy grid is extracted from the trained teacher by a threshold τ. However, blank holes appear in rendering images with threshold $\tau = 3$. Therefore, we set $\tau = 0$ to construct dense grids, which trades rendering efficiency for better image quality. Please see Table. B for qualitative comparison.

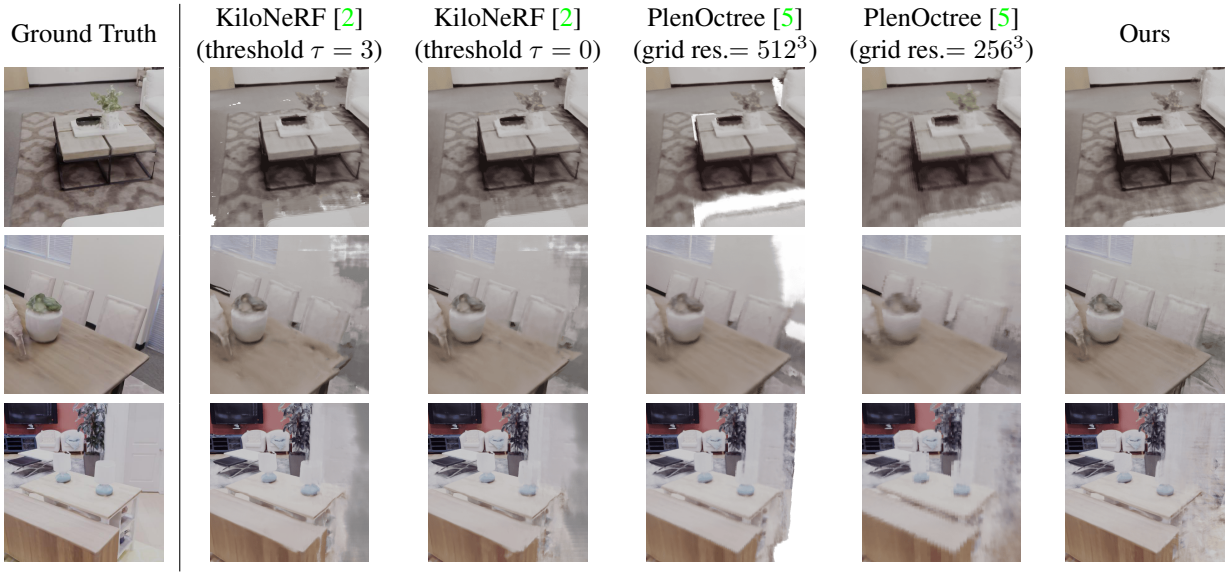

Table B. **Qualitative results of different implementation.** Dataset: Replica

D. More qualitative comparison

We show additional qualitative results in Fig. B, Fig. C, Fig. D, Fig. E, Fig. F, and Fig. G. From Fig. B, we can see that NeurMiPs can capture planar structures well, including the grid patterns on the wall (1st row), words on the Truck (3rd row), and Caterpillar (5th row). The model can also capture thin structures in the scene, as shown in the 2nd and 4th rows of the figure. Moreover, Fig. C demonstrates that NeurMiPs is able to capture complex non-planar geometry, such as the surface of human statues, especially their facial expressions (1st, 4th rows). KiloNeRF [2] produces grid-like artifacts due to its space partitioning (5th, 6th rows). In contrast, our method has a more smooth texture and fewer artifacts. Fig. D, Fig. E and Fig. F visualize the depth image of the scene, and compare the results of different methods. Our method is able to capture the boundaries between wood panels well, as seen in the last row of Fig. D. Note that dataset Tanks&Temple does not have ground truth depth images, so in Fig. D we provide RGB images as reference. Fig. E and Fig. F shows that while NeX [3] have black artifacts when viewing from extreme views, NeurMiPs learns to generate good texture and depths in novel views. Please refer to the videos for more qualitative comparisons.

We also evaluate and compare our method on BlendedMVS dataset [4], and provide qualitative results in Fig. G.

E. More detailed quantitative results

The detailed quantitative results of per-scene breakdown are reported in Tab. C, Tab. D and Tab. E. In Tab. C, we want to highlight that while NeurMiPs achieve comparable performance on human statues (*e.g.* Family, Ignatius), it outperforms others in the scenes which are rich in planar structures (*e.g.* Barn, Caterpillar, Truck), and this demonstrates the strength of the effectiveness of our planar experts. Tab. D shows that our method can produce high-quality images in challenging novel views and outperform the current state-of-the-art MPI-based method [3] by a large margin. Tab. E shows that our model produces reasonable results in dataset BlendedMVS [4]. However, it is challenging for our planes to fit the complex interior

Figure B. Qualitative results for the scenes of Barn (Tanks & Temples), Truck (Tanks & Temples), and Caterpillar (Tanks & Temples)

Figure C. Qualitative results for the scenes of Family (Tanks & Temples) and Ignatius (Tanks & Temples)

Figure D. Qualitative results of depth images for the scenes of Barn (Tanks & Temples), Caterpillar (Tanks & Temples), Family (Tanks & Temples), Ignatius (Tanks & Temples), and Truck (Tanks & Temples)

surface structures (*e.g.* "Jade", "Fountain"). We conjecture the reasons are two-fold: 1) limited expressiveness of our mixture of planar representations for such complicated structure 2) poor geometry initialization from COLMAP in these two scenes. We leave these challenging cases for future work.

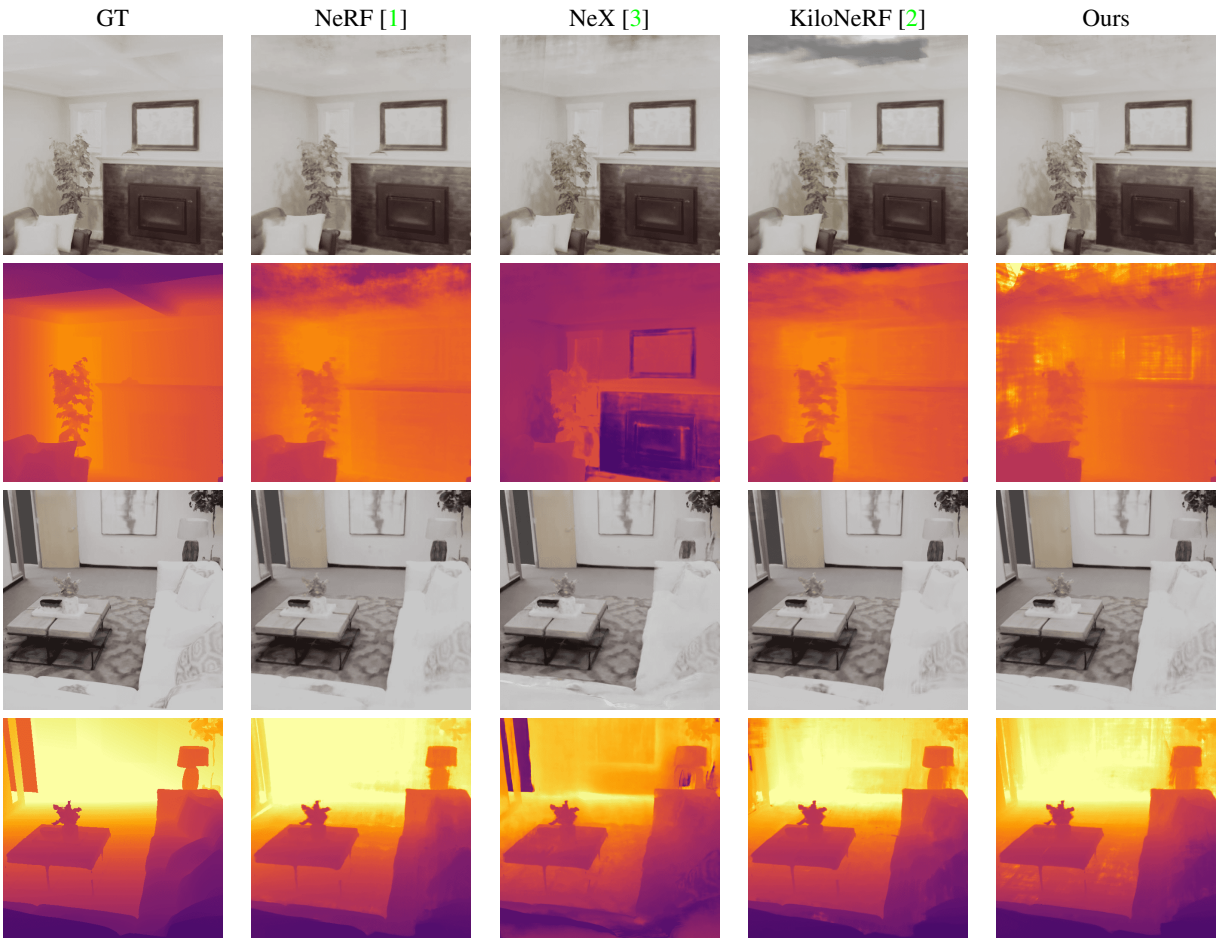

Figure E. **Qualitative comparisons of rendered RGB and depth images of Replica.**

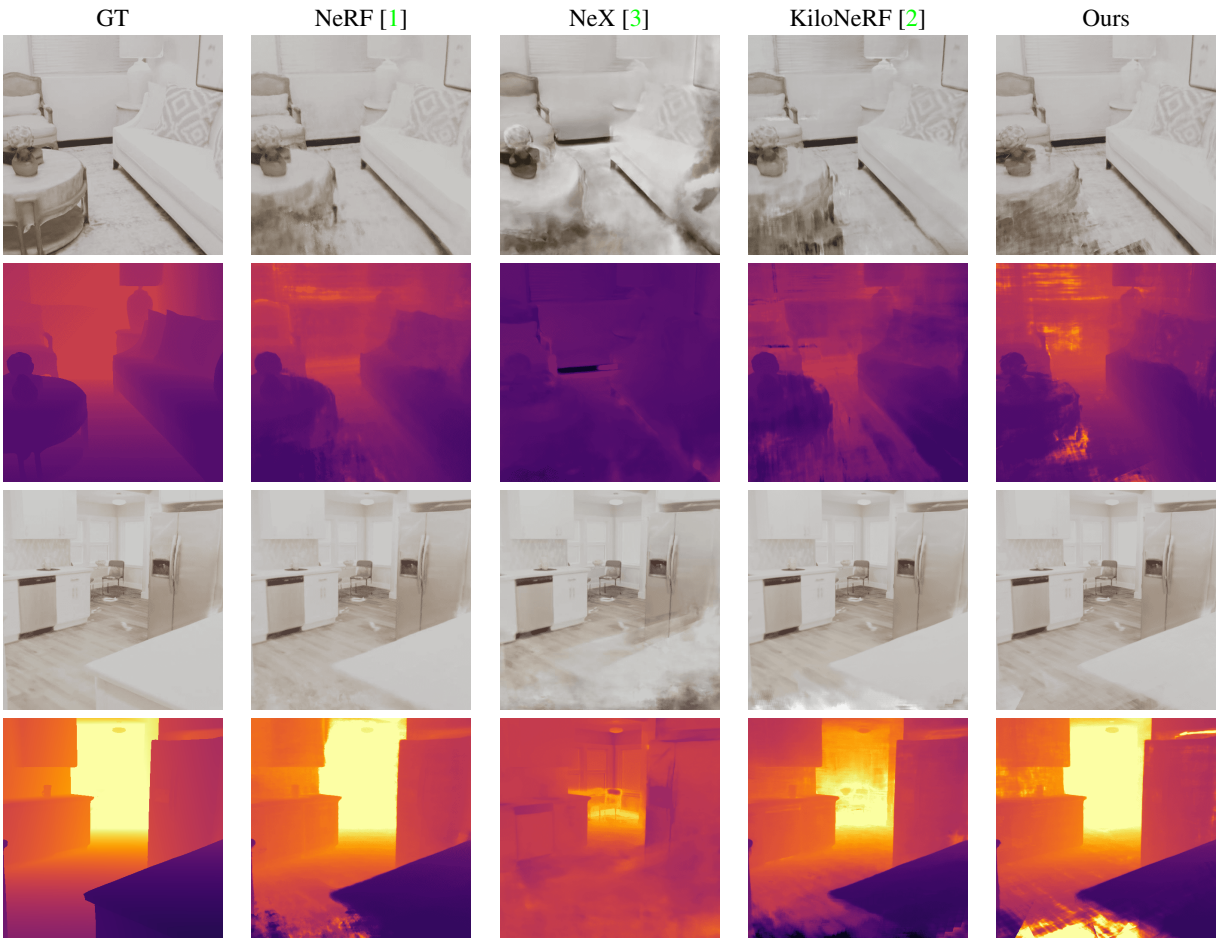

Figure F. **Qualitative comparisons of rendered RGB and depth images of Replica.**

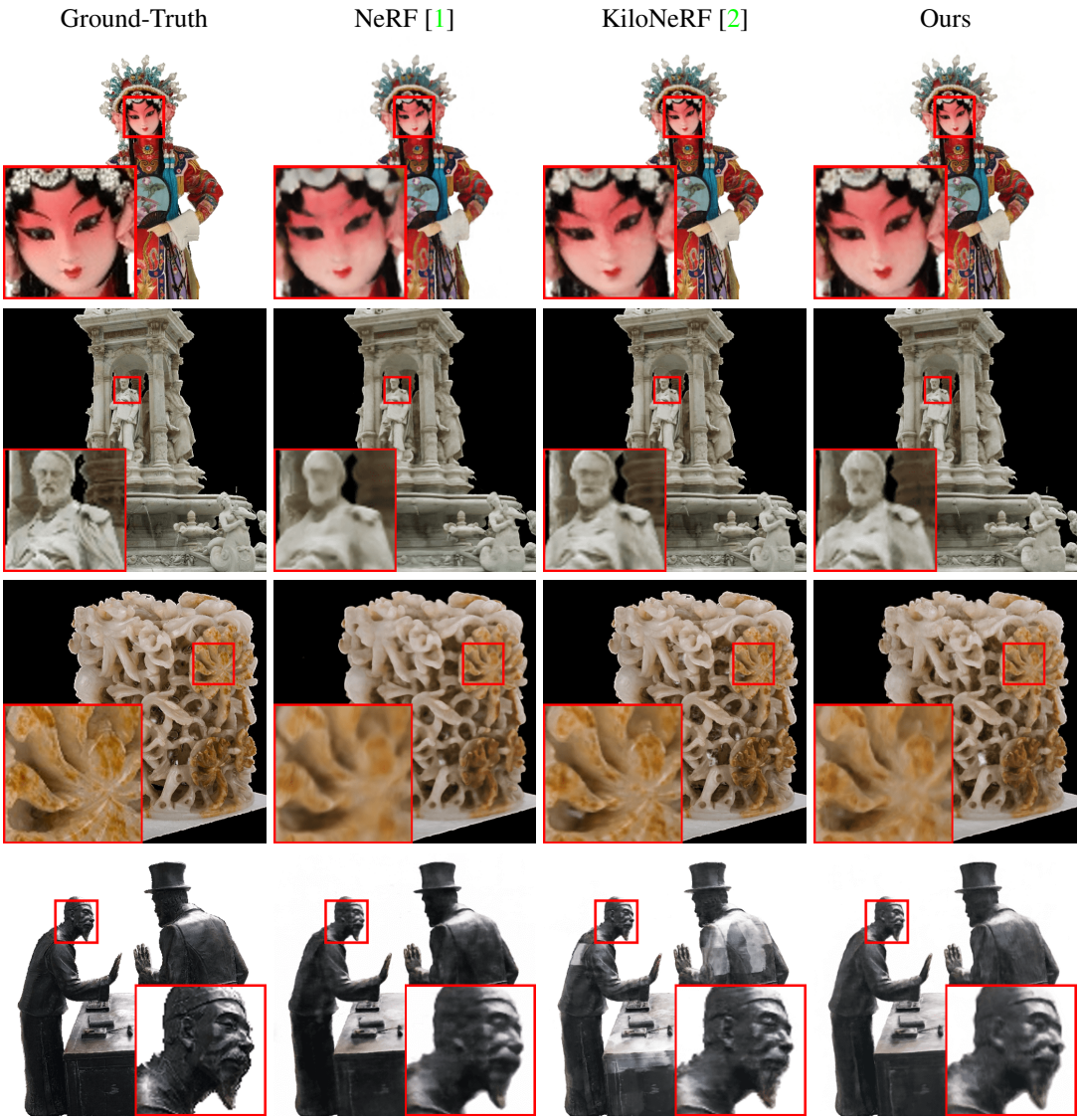

Figure G. Qualitative results for the scenes of Character (BlendedMVS), Fountain (BlendedMVS), Jade (BlendedMVS), Statues (BlendedMVS).

PSNR↑					
	Barn	Caterpillar	Family	Ignatius	Truck
NeRF (original)	24.05	23.75	30.29	25.43	25.36
NeRF	27.39	25.24	32.47	27.95	26.66
SRN	22.44	21.14	27.57	26.70	22.62
Neural Volumes	20.82	20.71	28.72	26.54	21.71
NSVF	27.16	**26.44**	33.58	27.91	26.92
KiloNeRF	27.81	25.61	**33.65**	27.92	27.04
PlenOctree	26.80	25.29	32.85	**28.19**	26.83
Ours	**28.06**	26.41	32.79	27.96	**27.16**
SSIM↑					
	Barn	Caterpillar	Family	Ignatius	Truck
NeRF (original)	0.750	0.860	0.932	0.920	0.860
NeRF	0.842	0.892	0.951	0.940	0.896
SRN	0.741	0.834	0.908	0.920	0.832
Neural Volumes	0.721	0.819	0.916	0.922	0.793
NSVF	0.823	0.900	0.954	0.930	0.895
KiloNeRF	0.850	0.900	0.960	0.940	0.900
PlenOctree	0.856	**0.907**	**0.962**	**0.948**	**0.914**
Ours	**0.858**	0.901	0.948	0.936	0.896
LPIPS↓					
	Barn	Caterpillar	Family	Ignatius	Truck
NeRF (original)	0.395	0.196	0.098	0.111	0.192
NeRF	0.286	0.189	0.092	0.102	0.173
SRN	0.448	0.278	0.134	0.128	0.266
Neural Volumes	0.479	0.280	0.111	0.117	0.312
NSVF	0.307	0.141	0.063	0.106	0.148
KiloNeRF	0.160	0.100	**0.040**	**0.060**	**0.100**
PlenOctree	0.226	0.148	0.069	0.080	0.130
Ours	**0.140**	**0.091**	0.050	0.063	0.103

Table C. **Quantitative results on Tanks & Temples.**

PSNR↑

	Apartment 0	Apartment 1	Apartment 2	FRL 0	Kitchen	Room 0	Room 2
NeRF	29.75	32.08	31.78	30.98	33.17	26.92	26.17
NeX	25.00	28.12	23.39	26.28	24.65	21.12	24.74
PlenOctree	26.90	28.09	28.59	27.65	30.78	25.56	26.45
KiloNeRF ($\tau = 3$)	29.98	27.98	30.83	29.40	29.98	24.54	27.10
KiloNeRF ($\tau = 0$)	29.84	31.56	29.52	30.90	30.49	24.95	**28.37**
Ours	**30.96**	**33.28**	**33.26**	**32.16**	**33.68**	**27.4**	28.16

SSIM↑

	Apartment 0	Apartment 1	Apartment 2	FRL 0	Kitchen	Room 0	Room 2
NeRF	0.899	0.928	0.917	0.913	0.937	0.870	0.840
NeX	0.826	0.899	0.760	0.880	0.834	0.794	0.834
PlenOctree*	0.856	0.889	0.890	0.866	0.916	0.864	0.822
KiloNeRF*	**0.905**	0.931	0.913	**0.924**	0.924	0.865	**0.863**
Ours	0.901	**0.938**	**0.928**	0.918	**0.941**	**0.879**	0.861

LPIPS↓

	Apartment 0	Apartment 1	Apartment 2	FRL 0	Kitchen	Room 0	Room 2
NeRF	0.093	0.069	0.073	0.083	0.064	0.134	0.165
NeX	0.152	0.088	0.212	0.100	0.160	0.200	0.154
PlenOctree	0.181	0.148	0.155	0.180	0.124	0.168	0.265
KiloNeRF ($\tau = 3$)	0.095	0.093	0.080	0.080	0.091	0.142	0.149
KiloNeRF ($\tau = 0$)	0.096	0.063	0.085	0.072	0.085	0.141	0.134
Ours	**0.084**	**0.052**	**0.056**	**0.071**	**0.054**	**0.120**	**0.117**

Table D. **Quantitative results on Replica.**

PSNR↑

	Character	Fountain	Jade	Statues
SRN	21.98	21.04	18.57	20.46
Neural Volumes	24.10	22.71	22.08	23.22
NeRF	29.43	28.04	26.52	25.17
NSVF	27.95	27.73	26.96	24.97
KiloNeRF	**29.44**	**28.50**	**27.14**	24.49
Ours	28.54	27.23	24.64	**25.76**

SSIM↑

	Character	Fountain	Jade	Statues
SRN	0.853	0.717	0.715	0.794
Neural Volumes	0.876	0.762	0.750	0.785
NeRF	**0.950**	0.910	0.890	0.870
NSVF	0.921	0.913	0.901	0.858
KiloNeRF	**0.950**	**0.930**	**0.910**	**0.880**
Ours	0.945	0.901	0.846	0.866

LPIPS↓

	Character	Fountain	Jade	Statues
SRN	0.208	0.291	0.323	0.354
Neural Volumes	0.140	0.263	0.292	0.277
NeRF	**0.030**	0.070	0.080	0.090
NSVF	0.074	0.113	0.094	0.171
KiloNeRF	0.040	**0.060**	**0.060**	**0.080**
Ours	0.042	0.075	0.109	0.115

Table E. **Quantitative results on BlendedMVS.**

References

[1] 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. 2, 4, 5, 6, 7, 8, 9

[2] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. *ICCV*, 2021. 3, 4, 5, 6, 7, 8, 9

[3] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *CVPR*, 2021. 2, 3, 7, 8

[4] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In *CVPR*, 2020. 3

[5] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. *ICCV*, 2021. 2, 3