Supplementary Materials for Multi-Space Neural Radiance Fields

Ze-Xin Yin Jiaxiong Qiu Ming-Ming Cheng Bo Ren*
VCIP, CS, Nankai University
{Zexin.Yin.cn, qiujiaxiong727}@gmail.com, {cmm, rb}@nankai.edu.cn

1. Detailed networks and experiment settings

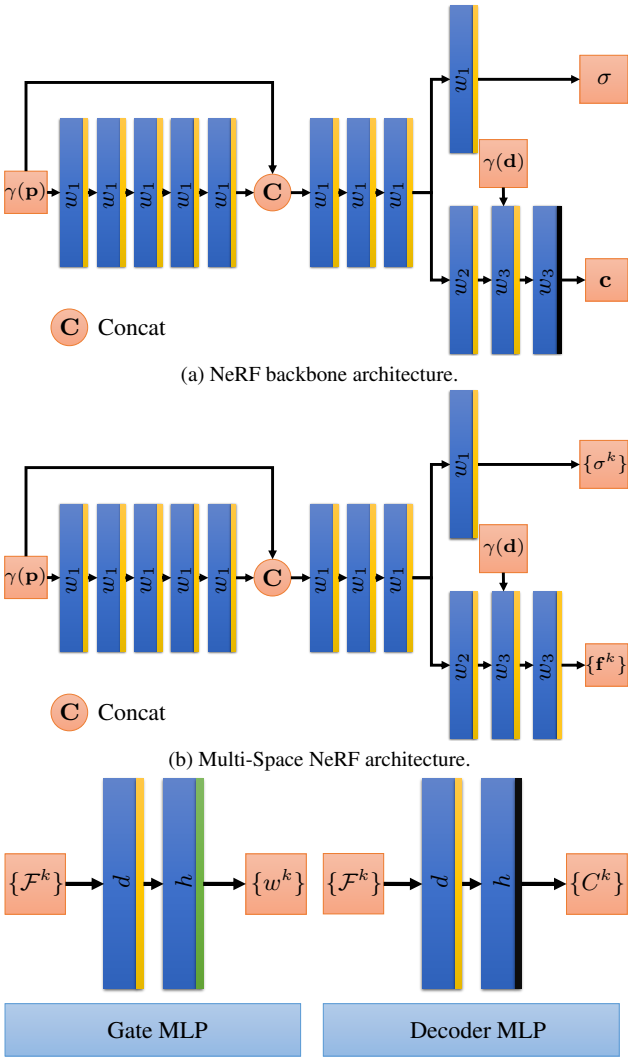
1.1. Network architecture

As illustrated in Fig. 1, we show the difference between our networks and the NeRF backbone network. NeRF [9], Mip-NeRF [1], and the 'NeRF MLP' of Mip-NeRF 360 [2] all share the same architecture except the width of fully-connected layers as shown in Fig. 1a. For NeRF and Mip-NeRF, the hyperparameters for layer width are $\{w_1 = 256, w_2 = 256, w_3 = 128\}$, and for Mip-NeRF 360 they are $\{w_1 = 1024, w_2 = 256, w_3 = 128\}$. We use $\gamma(\cdot)$ to uniformly represent the positional encoding function, as we only modify the output part of the networks and the positional encoding follows the original methods. Please refer to the original papers for more details about the positional encoding function.

As in Fig. 1b, we only change the output part of NeRF backbones. For the density branch that outputs a single density σ, we replace it with one that outputs K densities $\{\sigma^k\}$. And for the color branch that outputs a single color vector c, we replace it with one that outputs K feature vectors $\{\mathbf{f}^k\}$ of dimension d. K and d are hyperparameters for the number of sub-spaces and the dimension of the feature fields, respectively. Besides, we change the activation function of the color branch from Sigmoid to ReLU.

Most NeRF-based methods use volumetric rendering to accumulate the color c and the density σ along the ray to get the estimated color C for each pixel. Instead, we perform volumetric rendering for each pair of densities $\{\sigma^k\}$ and features $\{\mathbf{f}^k\}$ along the ray and get K accumulated features $\{\mathcal{F}^k\}$. Then we use two additional simple MLPs to decode and compose the final RGB information. As in Fig. 1c, the MLPs consist of two fully-connected layers with widths d and h. The Gate MLP uses the Softmax activation function to get the composition weights $\{w^k\}$, while the Decoder MLP uses the Sigmoid activation function to get the colors $\{C^k\}$ of each sub-space.

As illustrated above, our multi-space module consists



(a) NeRF backbone architecture.

(b) Multi-Space NeRF architecture.

(c) The architectures of Decoder MLP and Gate MLP.

Figure 1. NeRF backbone architecture and our model architectures. We denote fully-connected layers as the blue layers in the figure. We use different colors to represent different activation functions, *i.e.*, yellow for ReLU, green for Sigmoid, and black for Softmax.

*Bo Ren is the corresponding author.

of two simple MLPs and the output part of NeRF backbones. Thus we can scale our module with hyperparameters $\{K, d, h\}$. For NeRF and Mip-NeRF related experiments, we construct MS-NeRF$_S$ and MS-Mip-NeRF$_S$ with $\{K = 6, d = 24, h = 24\}$; similarly, MS-NeRF$_M$ and MS-Mip-NeRF$_M$ with $\{K = 6, d = 48, h = 48\}$, and MS-NeRF$_B$ and MS-Mip-NeRF$_B$ with $\{K = 8, d = 64, h = 64\}$. Besides, we construct MS-Mip-NeRF 360 with $\{K = 8, d = 32, h = 64\}$. To fairly compare with NeRFReN [5], we also construct MS-NeRF$_T$ with $\{K = 2, d = 128, h = 128\}$ based on NeRF.

Details about the importance sampling. To aggregate colors $\{\mathbf{c}_i\}$ along the rays using densities $\{\sigma_i\}$, NeRF-based methods calculate the contribution of each point to the estimated pixel color as follows:

$$I_i = T_i(1 - exp(-\sigma_i\delta_i)) \tag{1}$$

where $T_i = exp(-\sum_{j=1}^{i-1}\sigma_j\delta_j)$, $\delta_i = t_i - t_{i-1}$, and t_i represents the distance between the camera and the i-th sample point. Most NeRF-based methods require importance sampling along the rays where there are higher color contributions I_i accumulated. However, in our implementation, there are K parallel color contributions $\{I_i^k\}$ at each sample point. To perform the importance sampling, we use the weights $\{w^k\}$ of each sub-space to aggregate color contributions at each point as the one for the importance sampling, which is:

$$I_i = \sum_{j=1}^{K} w_i^j I_i^j \tag{2}$$

1.2. Training details

NeRF-based experiments. We follow the original settings from [9] with a few changes. We re-implement NeRF using PyTorch [11] and PyTorch Lightning [4] and borrow some code from [15]. We use the Adam optimizer [6] and exponentially decay the learning rate from 5e-4 to 7e-5 with $\beta_1 = 0.9, \beta_2 = 0.999$, and $\epsilon = 1e{-}8$. For all scenes and all experiments, we use 1024 rays per batch, and train 2e5 iterations with $N_c = 64$ sampled points for the coarse network and $N_f = 128$ sampled points for the fine network.

Mip-NeRF-based and Mip-NeRF 360-based experiments. We implement our Mip-NeRF [1] based methods on top of the official implementation[1] and implement our Mip-NeRF 360 [2] based method on top of [10]. We follow most training settings, except that we train 2e5 iterations with a batch of 1024 rays.

Ref-NeRF. We also use the official code [10] to train Ref-NeRF [13], and similarly, we follow most default settings, except 2e5 iterations and a batch size of 1024 for training.

[1]https://github.com/google/mipnerf

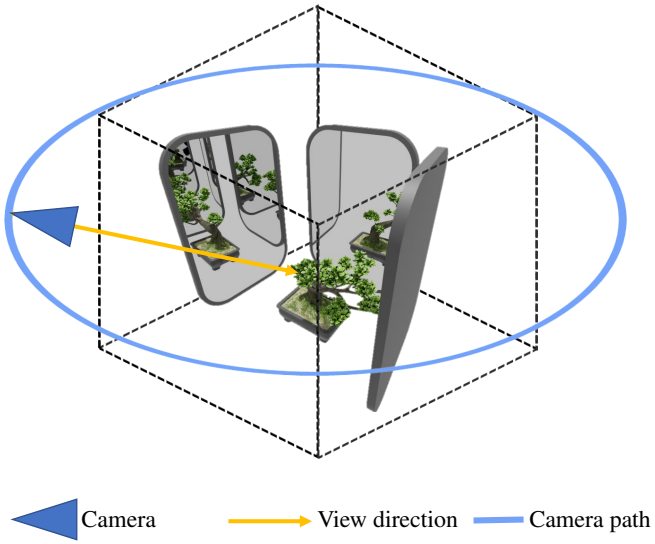


Figure 2. Illustration of the camera path in our synthetic dataset.

NeRFReN. We compare our NeRF-based variant MS-NeRF$_T$ with NeRFReN [5] on the RFFR dataset, and we re-train this method using the official code[2]. Similarly, we follow most provided settings, except that the number of used masks for reflective surfaces is zero for fair comparisons, as our methods require no masks.

1.3. Evaluation Protocols

We use PSNR, SSIM [14], and LPIPS [16] with the backbone of AlexNet [7] for quantitative comparisons. For the synthetic dataset, we evaluate the methods on the test set. For the real captured dataset, we sort all images by the names according to alphabet order and use every 1 of 8 images as the test images, as done in [8].

2. Additional details of our proposed dataset

2.1. Synthetic part

We use 3D models from BlenderKit[3], a community for sharing 3D models, textures, and others for 3D artworks, to create scenes for the synthetic dataset. We use the physically-based path tracer of Blender [3], Cycles, to render all the scenes, and we fix the height of the camera and move it around the circle in the scene. We make the camera look at the central objects and uniformly sample 120 viewpoints on the circle to render images as illustrated in Fig. 2, all at the resolution of 800×800. We randomly select 100 of the 120 images as the training images, 10 as the validation images, and 10 as the test images. As in Fig. 5, we visualize a few images for each scene. In most scenes, there are more than one mirrors that construct complex light paths, and we also introduce refractive and transparent materials.

[2]https://github.com/bennyguo/nerfren
[3]https://www.blenderkit.com/

2.2. Real captured part

We capture the real dataset using a Sony Alpha 6400 APS-C camera with a fixed 30mm lens. We fix the ISO, shutter speed, aperture size, and focus. We choose views carefully to avoid the appearance of the camera and the authors on the reflective surfaces. We use a few toys, books, two mirrors, a glass ball with a smooth surface, a glass ball with a diamond-like surface, and common furniture to construct our scenes, as shown in Fig. 6. Our scenes consist of 46 to 107 images, all at the resolution of 6000×4000, and the viewpoints are randomly split around the central objects. We use COLMAP [12] to estimate the camera poses and use every 1 of 8 images as the test set, and we downsample all images by a factor of 8 for training and evaluation. To demonstrate the different distribution of camera poses from our real captured dataset and the RFFR dataset, we visualize the poses of the scene 'Scan05' in our dataset and the scene 'mirror' in RFFR dataset in Fig. 3.

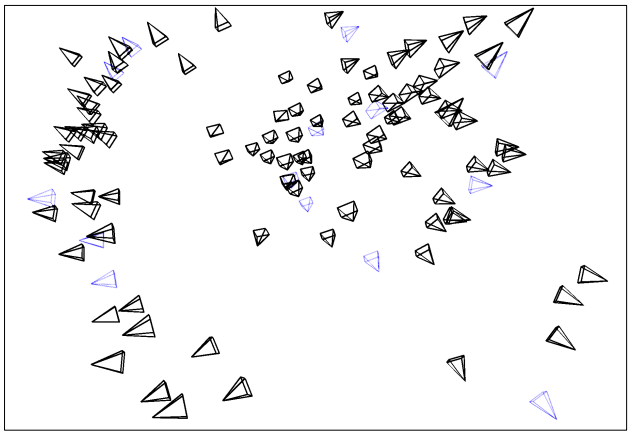
3. Additional experiment results

Results on our synthetic dataset. We report the detailed performance of our methods on each scene in our synthetic dataset using PSNR and SSIM, as in Tab. 1. And we also visualize one test view for each scene in Fig. 7a, Fig. 7b, and Fig. 7c. Our methods leverage the performance of NeRF-based methods by a large margin in reflection-related scenes, and in most scenes with merely refractions, our methods also improve the performance.

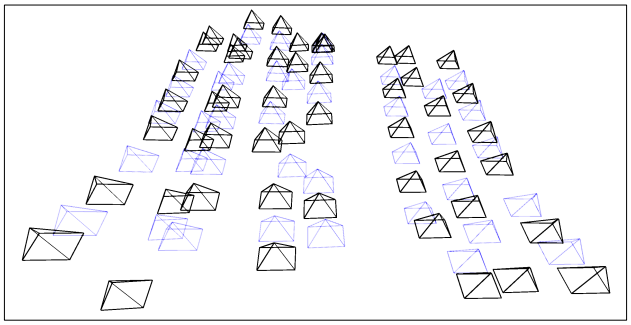
Results on our real captured dataset. We report the detailed performance of MS-Mip-NeRF 360 and Mip-NeRF 360 on each scene in our real captured dataset using PSNR and SSIM, as in Tab. 2. And we also present 1∼2 test view(s) for each scene, as in Fig. 8a and Fig. 8b. The qualitative and quantitative results both demonstrate that our method can handle mirror-like objects in the real world.

Results on RFFR dataset. We report the detailed performance of MS-NeRF$_T$ and NeRFReN on each scene in the RFFR dataset using PSNR and SSIM, as in Tab. 3. The results show that even for forward-facing scenes with mirrors, NeRFReN heavily relies on masks to guide the model, while our method performs stably in both 360-degree and forward-facing scenes.

Results on Realistic Synthetic 360° dataset and Real Forward-Facing dataset. The Realistic Synthetic 360° dataset and Real Forward-Facing dataset are first introduced in [9], which are commonly used for evaluating the ability of NeRF-based methods in novel view synthesis. We train Mip-NeRF and MS-Mip-NeRF$_B$ on these datasets because Mip-NeRF is a commonly used backbone for NeRF-based method. The results are reported in Tab. 4, which demonstrate that our multi-space module has no influence on the representation ability of NeRF-based methods on common materials. Note that we only train Mip-NeRF and MS-Mip-NeRF$_B$ with a batch size of 1024 for 2e5 iterations, while in the original paper, Mip-NeRF is trained for 1e6 iterations with a batch size of 4096. Therefore, our reported results are slower. Besides, under such settings, the models can already render satisfactory results, as in Fig. 4.



(a) Camera poses of the scene 'Scan05' in our dataset.



(b) Camera poses of the scene 'mirror' in the RFFR dataset.

Figure 3. Visualization of the camera poses in our real captured dataset and in the RFFR dataset. We draw training views in black and test views in blue.

4. Limitations and future works

Our methods leverage the performance of NeRF-based models on reflective surfaces by a large margin in an explainable way. However, to help NeRF-based methods better understand our 3D scenes, there need some constraints to help the network discriminate the virtual images from the real scenes. By now, our methods treat everything in the scenes equally.

Besides, refractive surfaces are often related to irregular shapes. Thus, the created virtual images are often distorted. Although our method is helpful in such circumstances, it fails to understand the curved light path caused by refraction. Therefore, to further improve the performance on refractive surfaces, a possible way is to introduce non-linear sub-spaces and model the distorted virtual images in them.

Scene	NeRF		MS-NeRF$_B$		Mip-NeRF		Ref-NeRF		MS-Mip-NeRF$_B$		Mip-NeRF 360		MS-Mip-NeRF 360	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
Scene01	33.48	0.950	37.75	0.967	34.05	0.955	36.23	0.961	39.57	0.973	34.23	0.963	41.47	0.982
Scene02	33.21	0.956	37.06	0.967	33.54	0.960	36.47	0.966	38.84	0.974	33.72	0.966	41.69	0.983
Scene03	36.59	0.966	38.54	0.970	37.68	0.970	38.96	0.971	41.17	0.977	38.74	0.977	42.63	0.983
Scene04	30.41	0.934	35.71	0.961	30.50	0.939	32.70	0.947	36.88	0.968	29.84	0.942	38.84	0.977
Scene05	27.52	0.914	32.85	0.951	28.57	0.918	28.53	0.924	33.73	0.959	26.79	0.917	35.37	0.969
Scene06	34.37	0.937	38.90	0.950	35.31	0.940	37.50	0.950	39.60	0.953	35.38	0.958	41.89	0.973
Scene07	32.41	0.839	32.43	0.838	32.73	0.845	33.01	0.847	32.91	0.845	33.78	0.885	34.07	0.887
Scene08	26.78	0.870	28.27	0.893	27.11	0.879	29.13	0.899	30.09	0.909	28.30	0.936	29.80	0.943
Scene09	32.08	0.890	32.63	0.895	32.73	0.900	32.79	0.898	33.22	0.902	33.20	0.907	32.40	0.898
Scene10	39.04	0.968	40.87	0.976	39.34	0.972	38.78	0.965	41.66	0.979	39.77	0.974	43.06	0.983
Scene11	28.29	0.790	28.56	0.791	28.57	0.797	29.86	0.808	31.60	0.820	28.91	0.842	32.20	0.853
Scene12	31.20	0.936	35.75	0.961	31.53	0.942	32.97	0.946	36.98	0.969	30.76	0.944	37.77	0.972
Scene13	26.04	0.630	25.97	0.620	26.16	0.658	26.03	0.644	26.21	0.660	25.92	0.663	25.74	0.617
Scene14	24.61	0.766	25.54	0.785	24.79	0.776	26.47	0.804	26.06	0.802	25.99	0.852	28.72	0.884
Scene15	25.10	0.740	25.25	0.737	25.53	0.754	26.00	0.777	25.68	0.754	25.55	0.786	25.43	0.762
Scene16	31.56	0.897	34.21	0.913	32.12	0.906	33.96	0.917	35.38	0.920	32.39	0.926	37.38	0.938
Scene17	25.20	0.776	25.64	0.775	25.47	0.794	25.89	0.811	25.54	0.785	24.48	0.799	25.28	0.783
Scene18	33.36	0.927	33.53	0.927	33.68	0.935	34.58	0.938	34.11	0.936	34.38	0.950	35.37	0.953
Scene19	22.86	0.752	23.86	0.769	22.83	0.764	23.72	0.771	24.35	0.789	24.64	0.836	26.87	0.860
Scene20	37.13	0.941	38.41	0.942	38.06	0.944	38.64	0.951	40.01	0.950	38.51	0.967	42.00	0.974
Scene21	26.51	0.641	26.63	0.638	26.95	0.653	27.41	0.666	27.33	0.658	28.06	0.704	28.41	0.710
Scene22	28.80	0.830	30.09	0.866	29.49	0.850	30.40	0.872	31.33	0.895	28.79	0.833	31.22	0.891
Scene23	35.53	0.942	39.08	0.952	38.31	0.949	38.65	0.952	39.87	0.954	36.64	0.957	40.46	0.963
Scene24	32.10	0.898	34.43	0.909	32.64	0.904	34.03	0.911	35.81	0.919	32.14	0.912	36.09	0.930
Scene25	36.37	0.948	37.16	0.947	37.86	0.954	36.57	0.946	39.63	0.955	38.60	0.973	41.96	0.976

Table 1. Detailed PSNR and SSIM of each scene in our synthetic dataset.

(a) Mip-NeRF (b) MS-Mip-NeRF$_B$

Figure 4. A test view from Mip-NeRF and MS-Mip-NeRF$_B$ in scene 'lego' from the Realistic Synthetic 360° dataset.

Scene	Mip-NeRF 360		MS-Mip-NeRF 360	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑
Scan01	27.87	0.917	30.84	0.919
Scan02	27.88	0.905	28.82	0.903
Scan03	27.86	0.898	29.48	0.901
Scan04	24.41	0.846	26.27	0.861
Scan05	27.26	0.911	27.60	0.907
Scan06	24.82	0.844	26.46	0.847
Scan07	26.77	0.904	27.50	0.899

Table 2. Detailed PSNR and SSIM of each scene in our real captured dataset.

Scene	NeRFReN		MS-NeRF$_T$	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑
art1	39.03	0.978	37.51	0.975
art2	41.91	0.970	41.87	0.970
art3	40.62	0.969	40.92	0.970
bookcase	30.26	0.890	29.80	0.885
tv	32.96	0.953	32.81	0.956
mirror	26.81	0.878	32.68	0.936

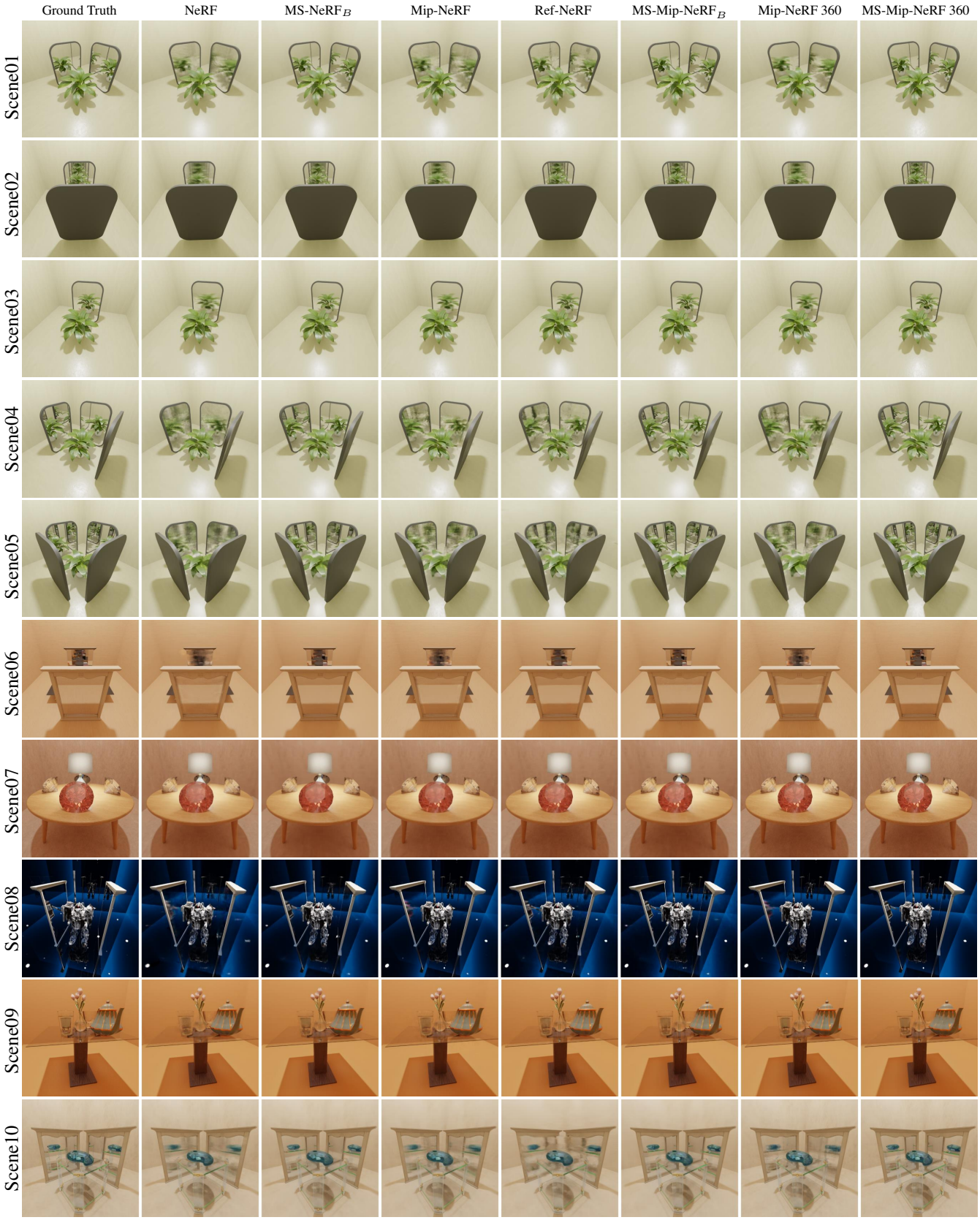
Table 3. Detailed PSNR and SSIM of each scene in the RFFR dataset [5].

dataset	Mip-NeRF		MS-Mip-NeRF$_B$	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑
Realistic Synthetic 360°	30.74	0.942	30.81	0.943
Real Forward-Facing	25.78	0.775	25.59	0.764

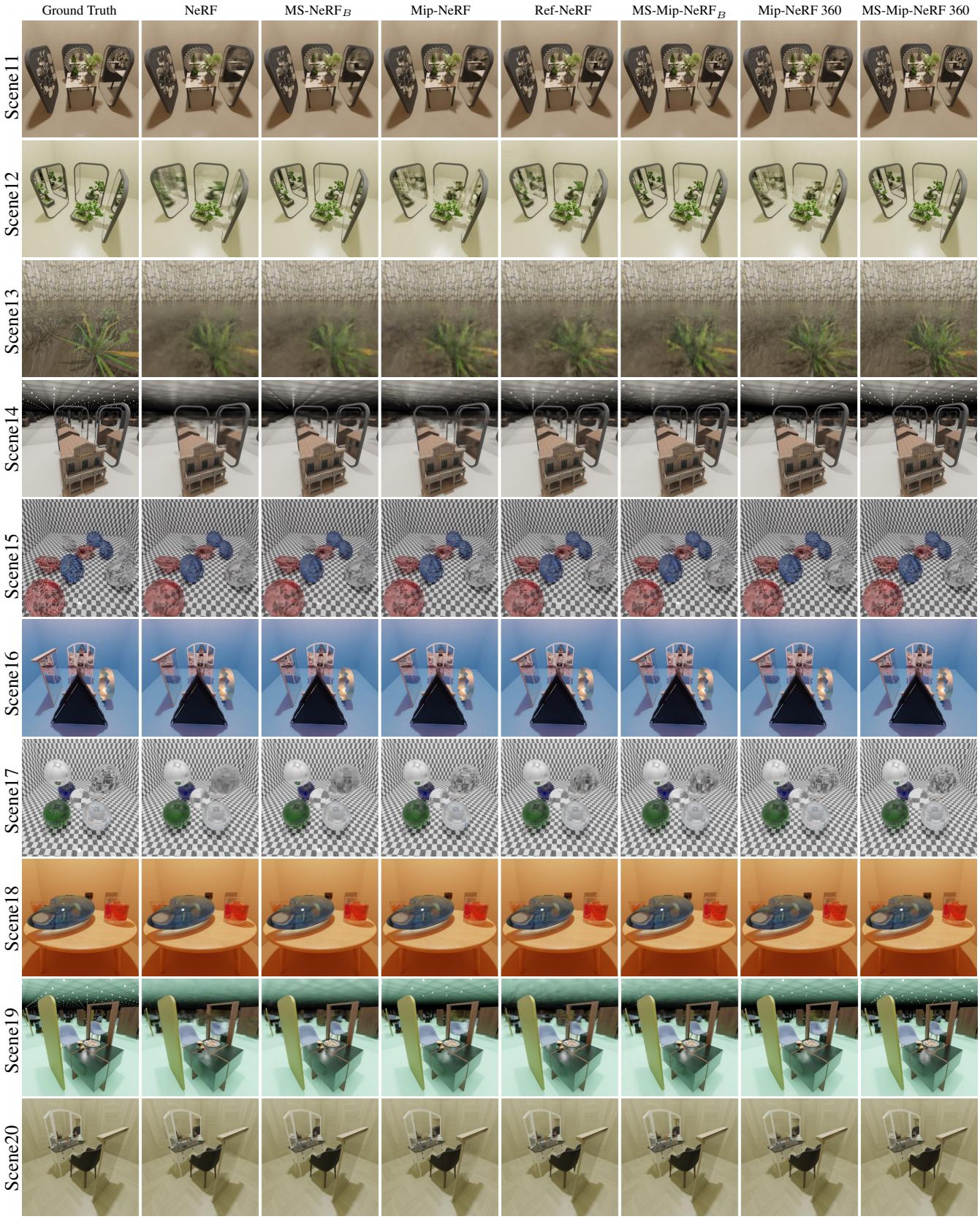
Table 4. Results on the Realistic Synthetic 360° dataset and Real Forward-Facing dataset.

Figure 5. We randomly visualize three training views and two test views for each scene in our synthetic dataset. In Scene01∼Scene05, we only change the layout and the number of the mirror(s), which can be treated as the basic part of our synthetic dataset; therefore, researchers can conduct preliminary experiments on them.

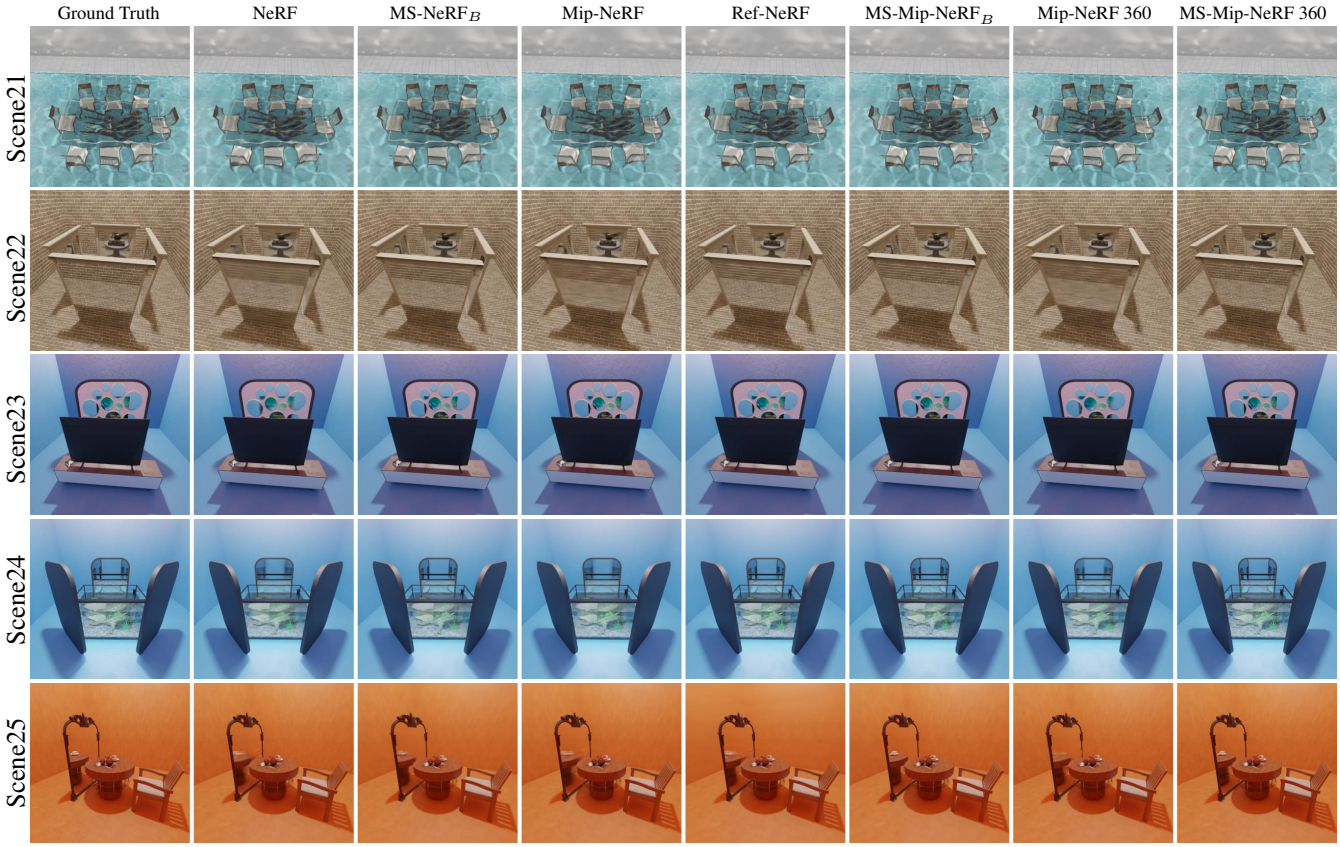
Figure 6. We randomly visualize three training views and two test views for each scene in our real captured dataset.



(a) Visual comparisons on Scene01∼Scene10. Please zoom in to see the details.

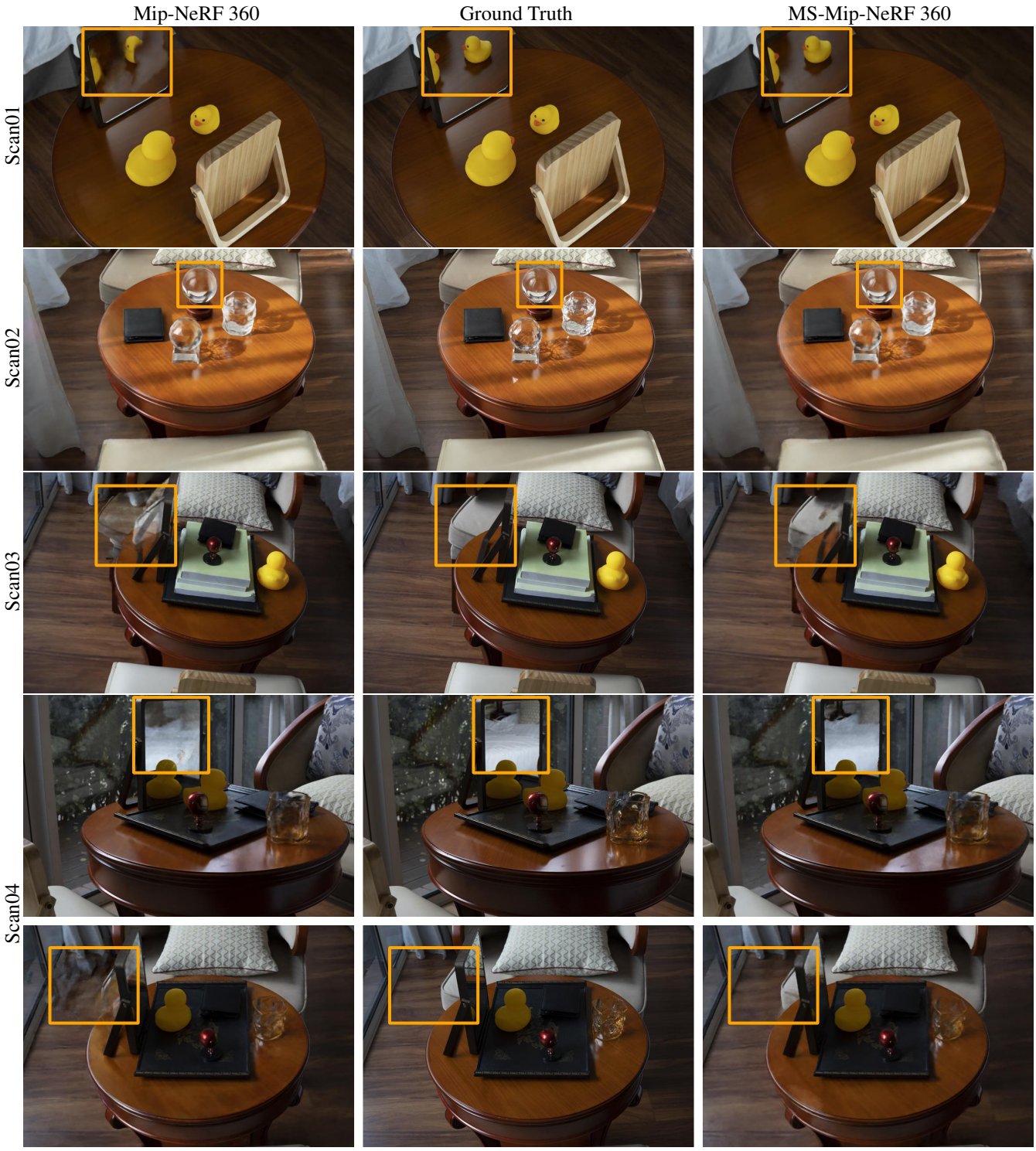


(b) Visual comparisons on Scene11∼Scene20. Please zoom in to see the details.

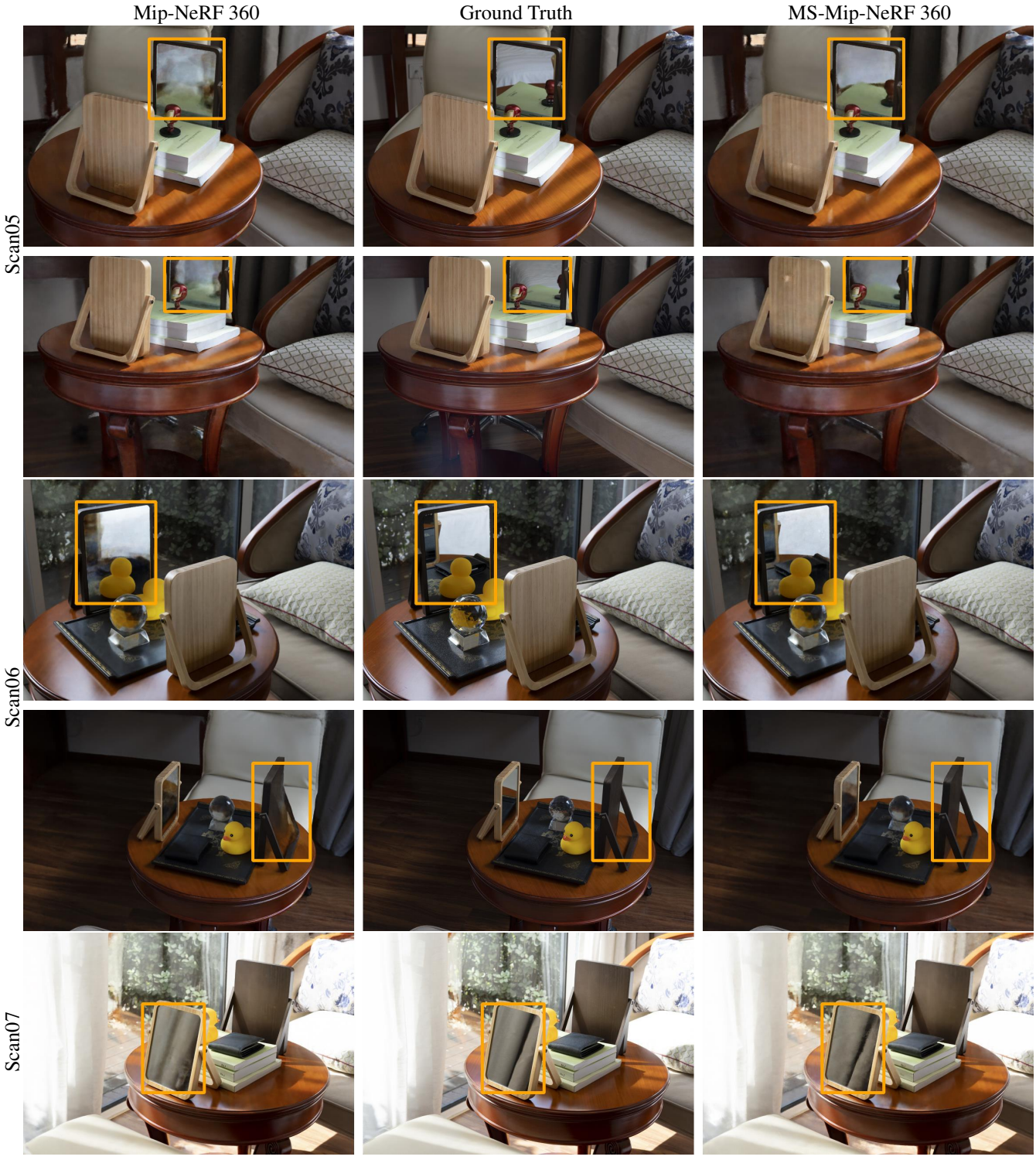


(c) Visual comparisons on Scene21∼Scene25. Please zoom in to see the details.

Figure 7. We randomly visualize one test view for each scene in our synthetic dataset.



(a) Visual comparisons on Scan01∼Scan04.



(b) Visual comparisons on Scan05∼Scan07.

Figure 8. We visualize 1∼2 test view(s) for each scene in our real captured dataset.

References

[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, pages 5855–5864, 2021. 1, 2

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fiecvprlds. In *IEEE CVPR*, pages 5470–5479, 2022. 1, 2

[3] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2022. 2

[4] William Falcon and The PyTorch Lightning team. Pytorch lightning, 3 2019. 2

[5] Yuan-Chen Guo, Di Kang, Linchao Bao, Yu He, and Song-Hai Zhang. Nerfren: Neural radiance fields with reflections. In *IEEE CVPR*, pages 18409–18418, 2022. 2, 4

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 2

[8] 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 TOG*, 38(4):1–14, 2019. 2

[9] 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, 2, 3

[10] Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, Peter Hedman, Ricardo Martin-Brualla, and Jonathan T. Barron. MultiNeRF: A Code Release for Mip-NeRF 360, Ref-NeRF, and RawNeRF, 2022. 2

[11] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 2

[12] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *IEEE CVPR*, pages 4104–4113, 2016. 3

[13] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *CVPR*, 2022. 2

[14] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 2

[15] Lin Yen-Chen. Nerf-pytorch. https://github.com/yenchenlin/nerf-pytorch/, 2020. 2

[16] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE CVPR*, pages 586–595, 2018. 2