MS-NeRF: Multi-Space Neural Radiance Fields Supplementary materials

1 Detailed networks and experiment settings

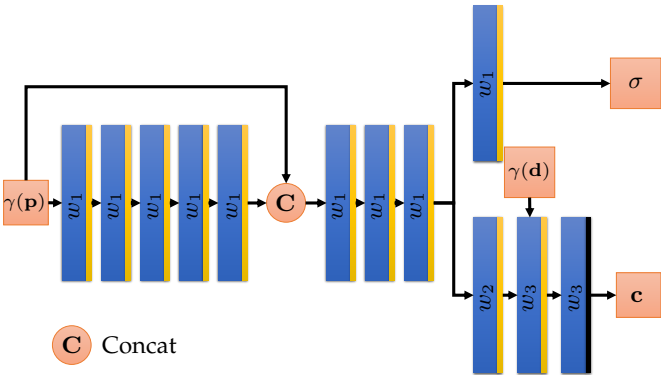
1.1 MLP-based Network architecture

As illustrated in Fig. 1, we show the difference between our networks and the NeRF backbone network. NeRF [1], Mip-NeRF [2], and the 'NeRF MLP' of Mip-NeRF 360 [3] 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.

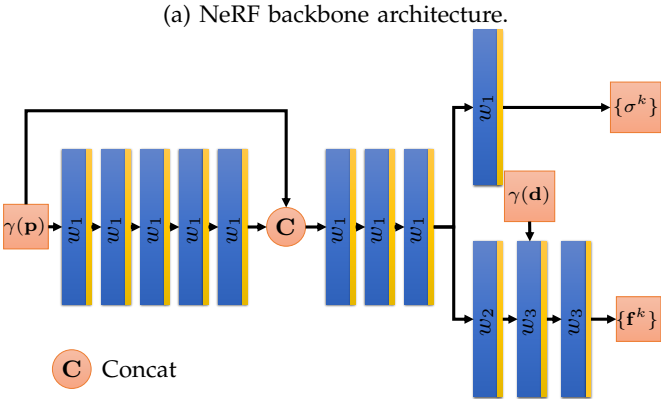
As in Fig. 1b, we only change the output part of NeRF backbones. For the density branch that outputs a single density σ, we replace it with one that outputs K densities $\{\sigma^k\}$. And for the color branch that outputs a single color vector c, we replace it with one that outputs K feature vectors $\{\mathbf{f}^k\}$ 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.

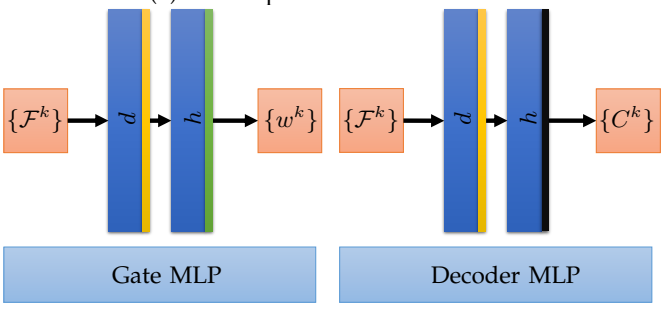
As illustrated above, our multi-space module consists of two simple MLPs and the output part of NeRF backbones. Thus we can scale our module with hyperparameters $\{K, d, h\}$. For NeRF and Mip-NeRF related experiments, we construct MS-NeRF$_S$ and MS-Mip-NeRF$_S$ with $\{K = 6, d = 24, h = 24\}$; similarly, 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 $\{K = 8, d =$



(a) NeRF backbone architecture.



(b) Multi-Space NeRF architecture.



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

Fig. 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.

$64, h = 64\}$. Besides, we construct MS-Mip-NeRF 360 with $\{K = 8, d = 32, h = 64\}$. To fairly compare with NeRFReN [4], we also construct MS-NeRF$_T$ with $\{K = 2, d = 128, h = 128\}$ based on NeRF.

1.2 Grid-based Network architecture

TensoRF [5] proposes the novel VM decomposition to extract features and densities from grids and uses very small MLPs with three layers mapping the features to color vectors as in Fig. 2, where $Q(\mathbf{V}, \mathbf{p})$ are feature vectors extracted from grid parameters, and for more details, please refer to [5]. iNGP [6] also features small MLP and grid-based trainable parameters, and details are in [6]. For the main branch of our TensoRF-based iNGP-based model, we only modify the output channel; specifically, our model outputs K densities $\{\sigma^k\}$ and colors $\{\mathbf{c}^k\}$ instead of one pair. The branch controlling gate information follows the same architecture as the color MLP in the main branch, except that it takes $\gamma(\mathbf{p})$ and $\gamma(\mathbf{d})$ as input and outputs K features $\{\mathbf{f}^k\}$. We then perform volumetric rendering for each pair of densities $\{\sigma^k\}$ and colors $\{\mathbf{c}^k\}$ to get K colors $\{C^k\}$ of each sub-space. Also, we perform volumetric rendering for densities $\{\sigma^k\}$ and features $\{\mathbf{f}^k\}$ to get K features $\{\mathcal{F}^k\}$, and we decode them using the gate MLP in Fig. 1c to get the composition weights $\{w^k\}$. Finally, we can render novel views by weighted sum.

We use the reimplementation of NeuS [7] from [8], which combines NeuS with hash encoding. For NeuS-based experiments, we simply modify the output layers following the design from Sec. 1.1, except that we change the density branch to a two-layered MLP.

1.3 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:

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

1.4 Training details

NeRF-based experiments. We follow the original settings from [1] with a few changes. We re-implement NeRF using PyTorch [9] and PyTorch Lightning [10] and borrow some code from [11]. We use the Adam optimizer [12] 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.

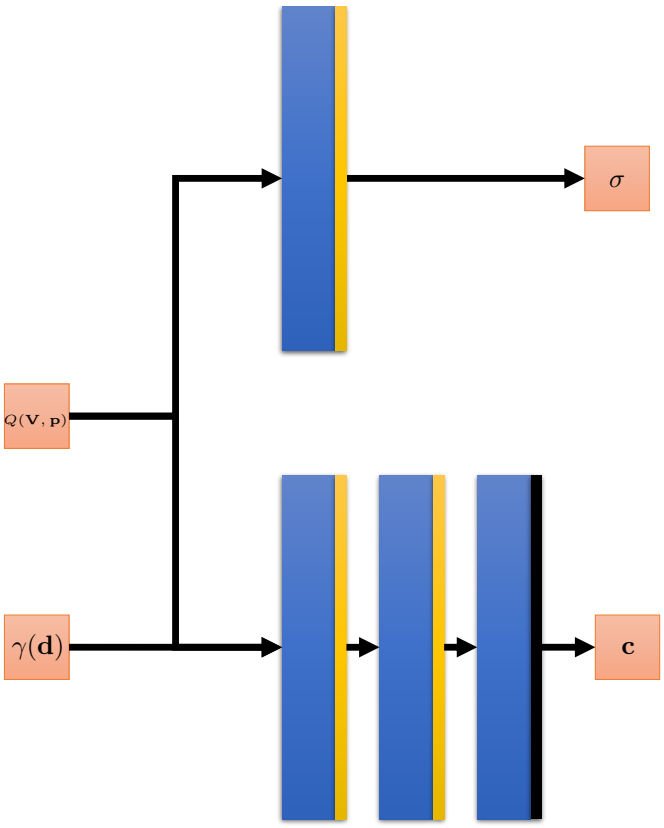


Fig. 2: TensoRF backbone architecture.

Mip-NeRF-based and Mip-NeRF 360-based experiments. We implement our Mip-NeRF [2] based methods on top of the official implementation[1] and implement our Mip-NeRF 360 [3] based method on top of [13]. We follow most training settings, except that we train 2e5 iterations with a batch of 1024 rays.

TensoRF-based and iNGP-based experiments. For TensoRF [5] based experiments, we build the models based on the official code[2] and follow all the default settings. And for iNGP-based experiments, we construct the model based on the code[3] with proposal network estimator and follow all the default settings.

NeuS-based experiments. For NeuS [7] based experiments, we build the models based on the code [8], and follow all the default settings except that we train 200k iterations as we train on scenes instead of objects.

Ref-NeRF. We also use the official code [13] to train Ref-NeRF [14], and similarly, we follow most default settings, except 2e5 iterations and a batch size of 1024 for training.

1. https://github.com/google/mipnerf
2. https://github.com/apchenstu/TensoRF
3. https://github.com/nerfstudio-project/nerfacc

NeRFReN. We compare our NeRF-based variant MS-NeRF$_T$ with NeRFReN [4] on the RFFR dataset, and we re-train this method using the official code[1]. 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.

1.5 Evaluation Protocols

We use PSNR, SSIM [15], and LPIPS [16] with the backbone of AlexNet [17] 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 [18].

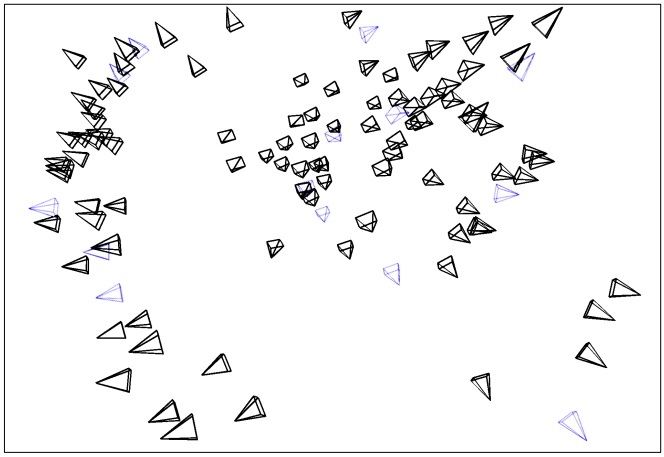
2 Additional Details of Our Proposed Dataset

2.1 Synthetic part

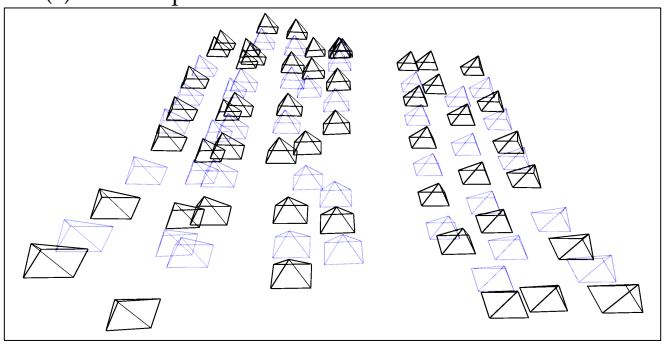
We use 3D models from BlenderKit[2], a community for sharing 3D models, textures, and others for 3D artworks, to create scenes for the synthetic dataset. As in Fig. 4, we visualize a few images for each scene. We render circle paths for all scenes, and we select 5 simple scenes and 5 challenging scenes, *i.e.*, 'Scene01', 'Scene02', 'Scene03', 'Scene04', 'Scene05', 'Scene14', 'Scene27', 'Scene30', 'Scene32', and 'Scene33', to render the spiral paths and the mirror-passing-through paths. In most scenes, there are more than one mirrors that construct complex light paths, and we also introduce refractive and transparent materials.

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. 5. Our scenes consist of 62 to 118 images, all at the resolution of 6000×4000, and the viewpoints are randomly split around the central objects. We use COLMAP [19] 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.



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



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

Fig. 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.

3 Sub-space Decomposition Results

We provide the composed RGB maps and depth maps, along with detailed sub-space weight maps, sub-space RGB maps, and sub-space depth maps in Fig. 6 to offer more transparency into the behavior of our multi-space scheme. The decomposition results indicate that our multi-scheme automatically handles virtual images in reflective surfaces by decomposing them into different sub-spaces.

Some depth visualizations exhibit that the shadow areas in the rendered images are in different values from the adjacent areas, which is not caused by our scheme but is the common issue of the NeRF-based models trained with RGB images only as shown in Fig. 7.

4 Multi-space Scheme with Varying Numbers of Input Images

To better observe the performance of the multi-space module, we demonstrate a few rendered views with different numbers of training images in Fig. 8. As indicated, our scheme robustly handles the virtual images caused by reflective surfaces, except that in the most challenging mirror-passing-through paths with 30 input images, some failure cases are observed. These results further confirm the robustness of the multi-space scheme.

1. https://github.com/bennyguo/nerfren
2. https://www.blenderkit.com/

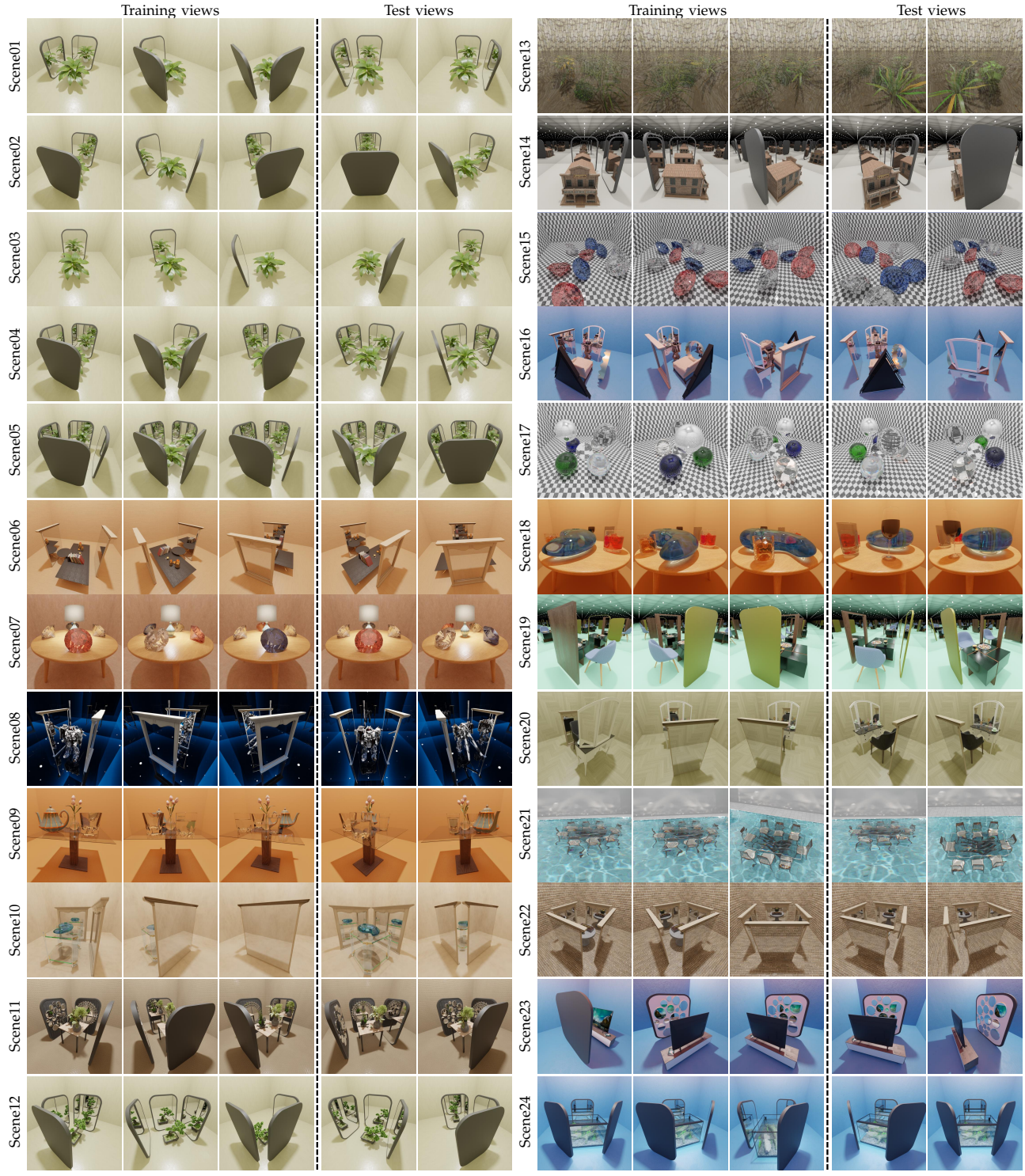
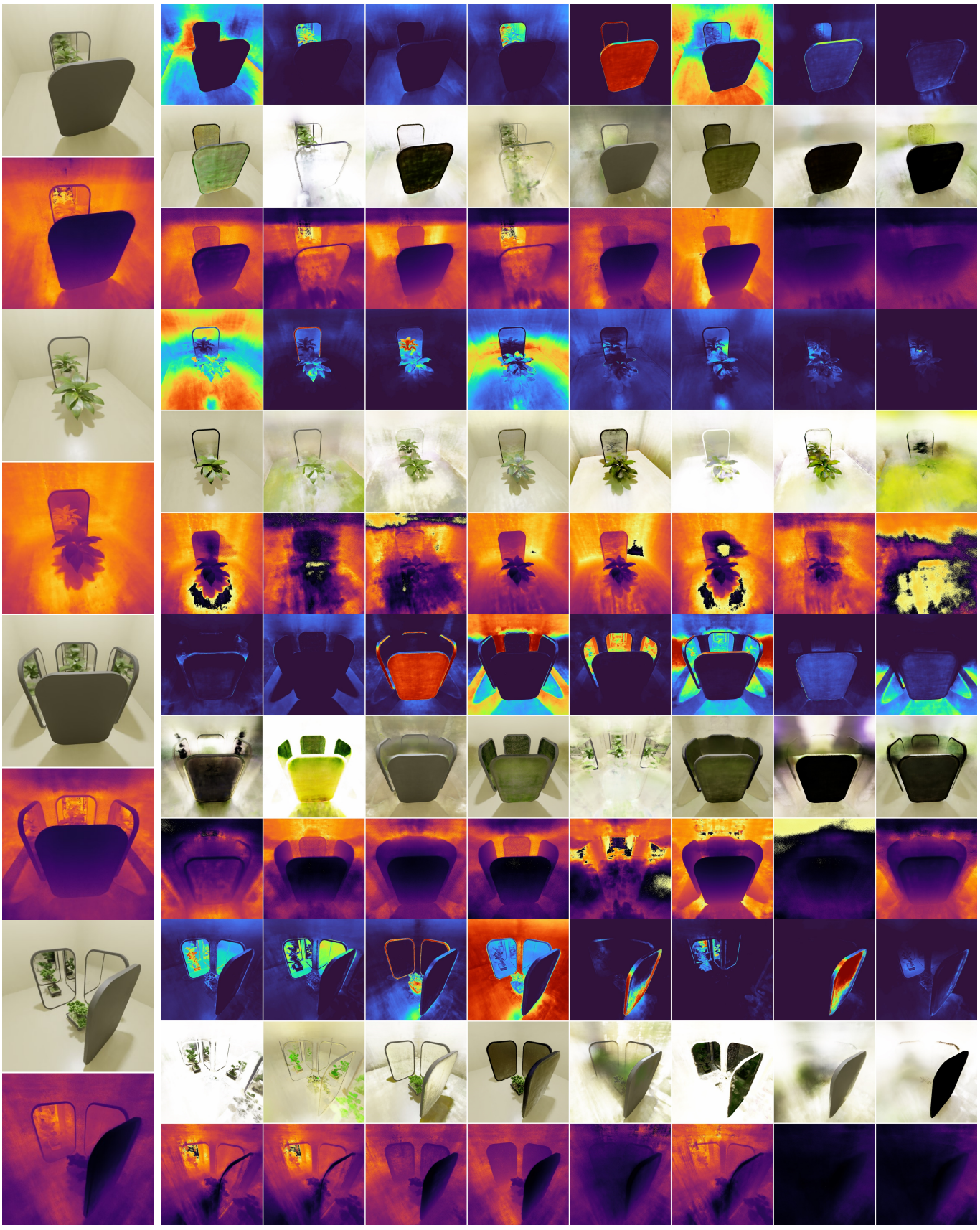


Fig. 4: We randomly visualize three training views and two test views for each scene in our synthetic dataset. In Scene01∼Scene05, we only change the layout and the number of the mirror(s), which can be treated as the basic part of our synthetic dataset; therefore, researchers can conduct preliminary experiments on them.

Fig. 4: 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.

Fig. 5: We randomly visualize three training views and two test views for each scene in our real captured dataset.

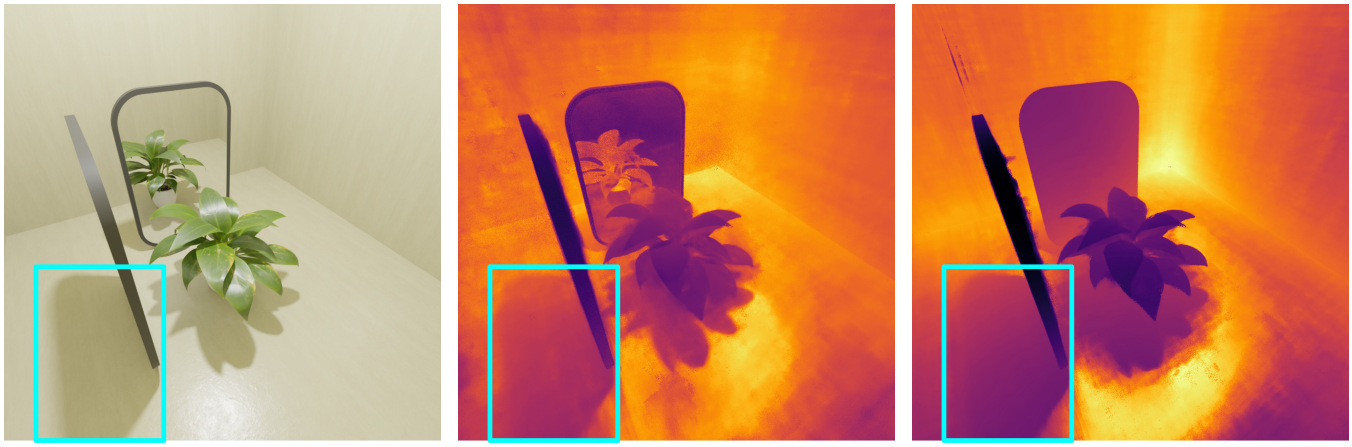


(a) (b)

Fig. 6: Visualizations of sub-space decomposition from MS-NeRF$_B$. (a) composed RGB and depth map. (b) from top to bottom are sub-space weight maps, sub-space RGB maps, and sub-space depth maps.

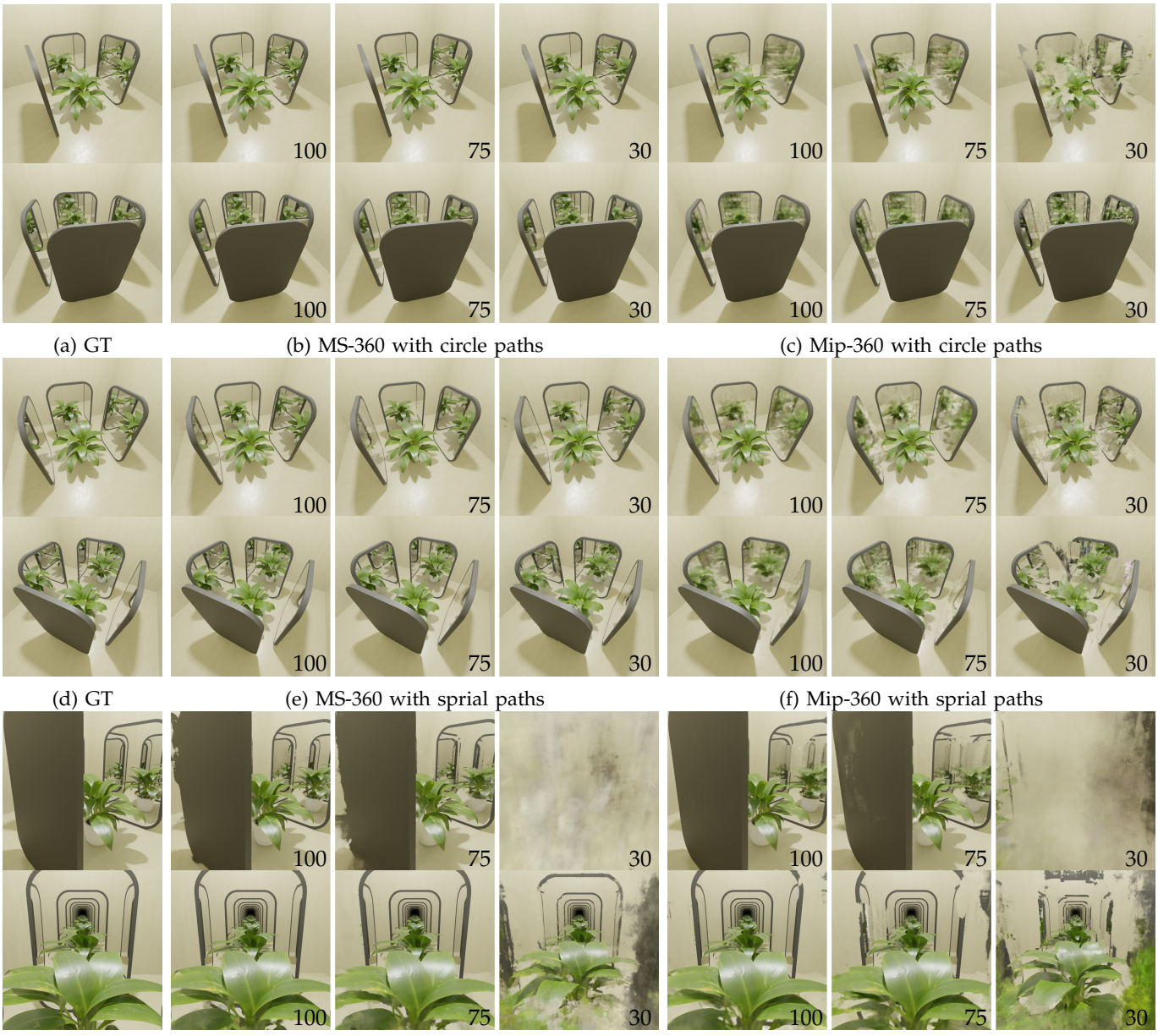
(c) (d)

Fig. 6: Visualizations of sub-space decomposition from MS-NeRF$_B$. (c) composed RGB and depth map. (d) from top to bottom are sub-space weight maps, sub-space RGB maps, and sub-space depth maps.



(a) Ground-Truth view. (b) Depth map from MS-NeRF$_B$. (c) Depth map from NeRF.

Fig. 7: Comparisons between the depth maps on shadow areas.



(a) GT (b) MS-360 with circle paths (c) Mip-360 with circle paths

(d) GT (e) MS-360 with sprial paths (f) Mip-360 with sprial paths

(g) GT (h) MS-360 with mirror-passing-through paths (i) Mip-360 with mirror-passing-through paths

Fig. 8: Visual comparisons between our model and Mip-NeRF 360 model with varying numbers of input images. MS-360 stands for MS-Mip-NeRF 360 and Mip-360 represents Mip-NeRF 360. We mark the number of training images in the lower right corner of each rendered image.

REFERENCES

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021. 1, 2

[2] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," in *Int. Conf. Comput. Vis.*, 2021, pp. 5855–5864. 1, 2

[3] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-nerf 360: Unbounded anti-aliased neural radiance fiecvprlds," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022, pp. 5470–5479. 1, 2

[4] Y.-C. Guo, D. Kang, L. Bao, Y. He, and S.-H. Zhang, "Nerfren: Neural radiance fields with reflections," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022, pp. 18 409–18 418. 2, 3

[5] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, "Tensorf: Tensorial radiance fields," in *European Conference on Computer Vision (ECCV)*, 2022. 2

[6] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, jul 2022. 2

[7] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction," *Advances in Neural Information Processing Systems*, vol. 34, pp. 27 171–27 183, 2021. 2

[8] Y.-C. Guo, "Instant neural surface reconstruction," 2022, https://github.com/bennyguo/instant-nsr-pl. 2

[9] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019. 2

[10] W. Falcon and T. P. L. team, "Pytorch lightning," 3 2019. [Online]. Available: https://www.pytorchlightning.ai 2

[11] L. Yen-Chen, "Nerf-pytorch," https://github.com/yenchenlin/nerf-pytorch/, 2020. 2

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. 2

[13] B. Mildenhall, D. Verbin, P. P. Srinivasan, P. Hedman, R. Martin-Brualla, and J. T. Barron, "MultiNeRF: A Code Release for Mip-NeRF 360, Ref-NeRF, and RawNeRF," 2022. [Online]. Available: https://github.com/google-research/multinerf 2

[14] D. Verbin, P. Hedman, B. Mildenhall, T. Zickler, J. T. Barron, and P. P. Srinivasan, "Ref-NeRF: Structured view-dependent appearance for neural radiance fields," *CVPR*, 2022. 2

[15] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004. 3

[16] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 586–595. 3

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017. 3

[18] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–14, 2019. 3

[19] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 4104–4113. 3