Appendix

A. Details of Preliminary Experiments

To draw the histograms shown in Fig.2, we need to calculate the deviation and error of the predicted depth for each pixel in the test dataset. Here we choose 3D-FRONT [5] as our synthetic training set and ScanNet [3] as our real training set and test set. Refer to Sec.5.1 and App.B for more details on the dataset. We train the models on the 3D-FRONT and ScanNet respectively and use the trained models to predict depth on the test dataset. For each pixel $\mathbf{u}$ in the test dataset, we first obtain the ray $\mathbf{r}$ as a line $\mathcal{R}$ as shown in Eq.3 and sample a sequence of points $\{\mathbf{p}^s = \mathcal{R}(\delta^s)\}_{s=1}^{N_s}\}$, where δ^s is the depth of $\mathbf{p}^s$ and N_s is the number of sampling points, set by 128 default. Then we calculate the volume density σ^s at each sampling point, just like we calculate the volume density when rendering the image. Note that we use the fine stage predictions if coarse-to-fine sampling is applied. We use the following formula to assign a weight w^s to each sampling point according to the volume density σ^s

$$w^s = (1 - \exp(-\sigma^s)) \cdot \exp(-\sum_{t=1}^{s-1} \sigma^t). \tag{13}$$

Then we predict the depth $\hat{D}(\mathbf{u})$, the standard deviation of the depth $S(\mathbf{u})$, and the depth error $E(\mathbf{u})$ for each pixel $\mathbf{u}$

$$\begin{aligned} \hat{D}(\mathbf{u}) &= \sum_{s=1}^{N_s} w^s \cdot \delta^s, \\ S(\mathbf{u}) &= \left(\sum_{s=1}^{N_s} w^s \cdot (\delta^s - \hat{D}(\mathbf{u}))^2 \right)^{\frac{1}{2}}, \\ E(\mathbf{u}) &= \|\hat{D}(\mathbf{u}) - D(\mathbf{u})\|, \end{aligned} \tag{14}$$

where $D(\mathbf{u})$ is the ground truth depth for pixel $\mathbf{u}$. Finally, we can draw the histogram shown in Fig.2 based on $S(\mathbf{u})$ and $E(\mathbf{u})$.

Results of other methods We also calculate the deviation and error of the depth predicted by other generalizable NeRF models (MVSNeRF [1], GeoNeRF [8], Neuray [10], and our method) as shown in Fig.7. Note that for our method, we calculate the weight w^s following the Eq.10 as

$$w^s = \frac{\exp(\sigma^s)}{\sum_{t=1}^{N_s} \exp(\sigma^t)}. \tag{15}$$

Like IBRNet [15], previous NeRF generalization methods [1, 8, 10] tend to predict radiance fields that are sharper but less geometrically accurate under the synthetic-to-real setting. In comparison, our method predicts a more accurate radiance field while remaining sharp.

B. Preprocess of Dataset

B.1. 3D-FRONT

Here we describe how we preprocess 3D-FRONT [5] dataset. First, we randomly pick 88 rooms labeled as living room or bedroom from the dataset. For each sampled room, we iteratively select 200 camera views and we need to ensure that there is a certain overlap but also distance between the different selected camera views. The overlap and distance are calculated between the currently sampled camera view and the previously sampled camera views.

Formally, let $\mathbf{K}$, $\mathbf{E} = [\mathbf{R}, \mathbf{t}]$ denote the camera intrinsic and extrinsic parameters respectively. We use fixed intrinsic for all camera views and only need to sample the extrinsic parameters for each camera view. The overlap between camera $\mathbf{E}_1$ and camera $\mathbf{E}_2$ is obtained by calculating the Intersection Over Union (IoU) of the two camera frustums

$$\mathcal{O}(\mathbf{E}_1, \mathbf{E}_2) = \frac{A \cap B}{A \cup B}, \tag{16}$$

where A and B are the frustums of $\mathbf{E}_1$ and $\mathbf{E}_2$ respectively. The distance between camera $\mathbf{E}_1$ and camera $\mathbf{E}_2$ is calculated from the camera position and orientation

$$\mathcal{D}(\mathbf{E}_1, \mathbf{E}_2) = \|\mathbf{t}_1 - \mathbf{t}_2\|_2 + \arccos((\text{Tr}(\mathbf{R}_2^\top \mathbf{R}_1) - 1)/2). \tag{17}$$

Suppose we have sampled a series of camera extrinsics $\mathcal{E} = \{\mathbf{E}_i = [\mathbf{R}_i, \mathbf{t}_i]\}_{i=1}^{N}$, where N is the number of sampled extrinsics. For the camera extrinsic $\mathbf{E}$, We calculate the overlap and distance between $\mathbf{E}$ and $\mathcal{E}$ by the following formulas

$$\textbf{Overlap}(\mathbf{E}, \mathcal{E}) = \max(\mathcal{O}(\mathbf{E}, \mathbf{E}_1), ..., \mathcal{O}(\mathbf{E}, \mathbf{E}_N)), \tag{18}$$

and

$$\textbf{Distance}(\mathbf{E}, \mathcal{E}) = \min(\mathcal{D}(\mathbf{E}, \mathbf{E}_1), ..., \mathcal{D}(\mathbf{E}, \mathbf{E}_N)). \tag{19}$$

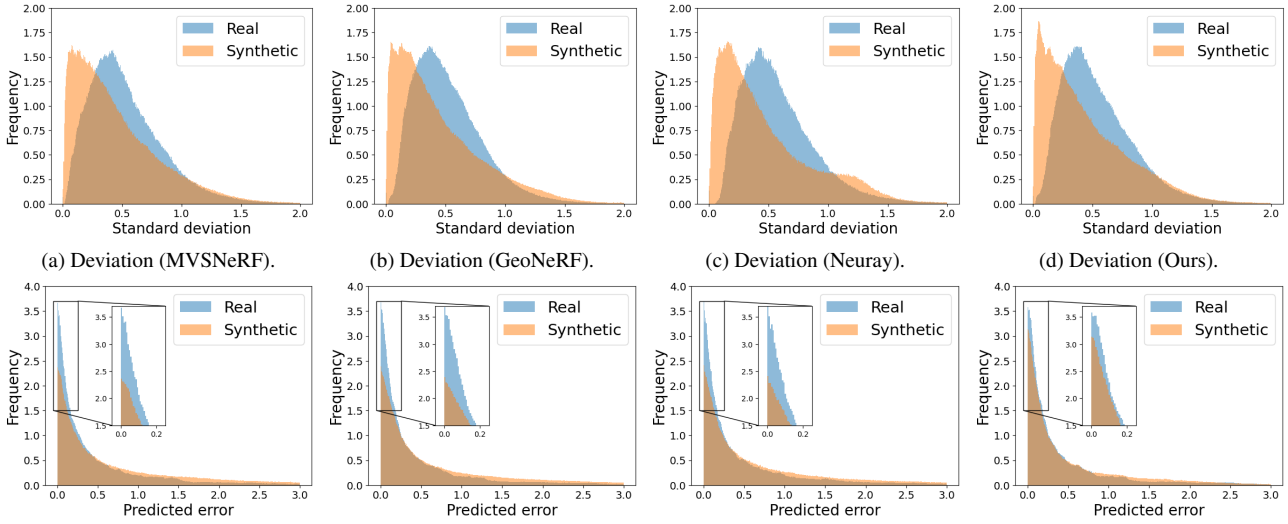
Once the overlap and distance reach a certain threshold, we add $\mathbf{E}$ to $\mathcal{E}$. Finally, the camera view selecting algorithm is shown in Alg.1. We show some images of 3D-FRONT [5] as shown in Fig.8.

B.2. ScanNet

In this paper, we randomly select 8 scenes of ScanNet [3] as our test datasets. The test scene numbers are 'scene0204', 'scene0205', 'scene0269', 'scene0289', 'scene0456', 'scene0549', 'scene0587', and 'scene0611', respectively. We also show some images of ScanNet [3] as shown in Fig.8.

B.3. Other Benchmark Datasets

During training, depth information is needed to determine the selection of positive pairs in our GeoContrast



(a) Deviation (MVSNeRF). (b) Deviation (GeoNeRF). (c) Deviation (Neuray). (d) Deviation (Ours).

(e) Error (MVSNeRF). (f) Error (GeoNeRF). (g) Error (Neuray). (h) Error (Ours).

Figure 7. **Deviation and error of predicted depth when trained with synthetic and real data, respectively**. We plot the deviation and error of the predicted depth as the histogram for MVSNeRF [1] (column 1), GeoNeRF [8] (column 2), Neuray [10], and our method (column 4). Our method is able to predict more accurate depth while maintaining its sharpness under synthetic-to-real setting.

Figure 8. **Images in 3D-FRONT dataset [5] and ScanNet dataset [3]**. The first row shows the images in 3D-FRONT dataset. The second row shows the images in ScanNet dataset.

(Sec.4.2). Since DTU dataset [7] and LLFF dataset [12, 15] only contain RGB images and not depth, we use COLMAP [14] to estimate the depth for each RGB image following [10]. Note that depth information is only used during training, not during testing.

C. Network Architecture

In the feature extraction, we use a U-Net like network, where ResNet34 [6] truncated after layer3 as the encoder, and two additional up-sampling layers with convolutions and skip-connections as the decoder, to extract features from input images following [10,15]. All convolution layers use ReLU as activation function and all batch normalization layers are replaced by instance normalization layers. The output dimensions of each layer of the encoder are 32, 64, 128 respectively, and the output dimensions of each layer of the decoder are 64 and 32 respectively. As for cross-view attention, we use the subtraction attention [17] as our attention module for its effectiveness in geometric relationship reasoning, and the attention layer of the different stages (see Sec.4.1) does not share parameters. We apply cross-view attention between the encoder and the decoder of U-Net and sample 16 projections as the key values for each query in the first stage, due to GPU memory limitation. Before entering the cross-view attention, we use a linear layer to reduce the feature dimension from 128 to 32. After cross-view attention, we also use a linear layer to increase the feature dimension from 32 to 128. The architecture of cross-view attention is illustrated in Fig.9. We implement the rendering net-

Algorithm 1 Camera view selecting

Initialization: $\mathcal{E} \leftarrow \{\}, n \leftarrow 0, N_v \leftarrow 200, T_o \leftarrow 0.3, T_d \leftarrow 0.2$
repeat
  Sample camera view as $\mathbf{E}$
  if $n = 0$ **then**
    $\mathcal{E} \leftarrow \mathcal{E} + \{\mathbf{E}\}$
    $n \leftarrow n + 1$
  else
    overlap $\leftarrow$ **Overlap**$(\mathbf{E}, \mathcal{E})$
    distance $\leftarrow$ **Distance**$(\mathbf{E}, \mathcal{E})$
    if overlap $\geq T_o$ **and** distance $\geq T_d$ **then**
      $\mathcal{E} \leftarrow \mathcal{E} + \{\mathbf{E}\}$
      $n \leftarrow n + 1$
    end if
  end if
until $n \geq N_v$
Output: A list of camera views $\mathcal{V}$.

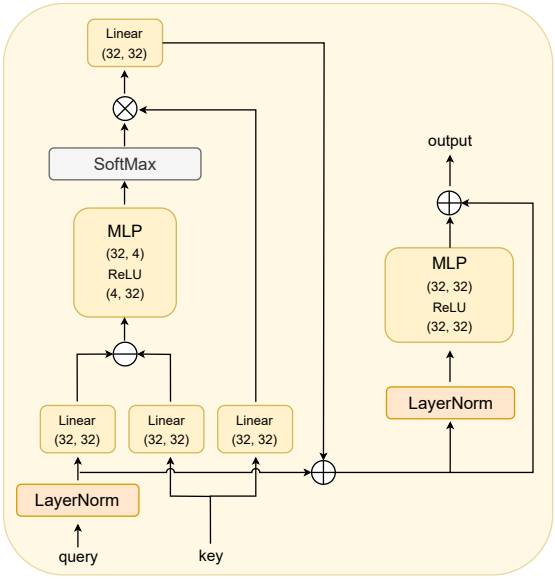


Figure 9. **Architecture of cross-view attention.**

work mainly following IBRNet [15], where the multi-view feature aggregation module aggregates the density information of all samples on the ray to enable visibility reasoning, and the ray transformer is then applied to calculate the volume density.

D. More Experimental Results

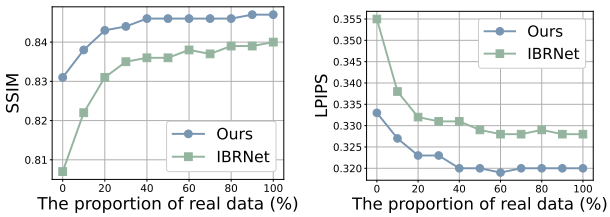
D.1. Additional Ablation Study

Ablation on the number of negative pairs. The number of negative pairs is shown to have a larger effect on the performance of contrastive learning as described in [2], and a larger number of negative pairs has a significant advantage over the smaller ones. Here in our method, we conduct ablation studies with different numbers of negative pairs to see the effect of the number of negative pairs on the model performance. As shown in Tab.5, the performance of our method increases as the number of negative pairs increases, with the best performance when the number reaches 512, which is consistent with the phenomenon in [2]. We also tried another way of multi-view consistency optimization, that is, we directly optimize the similarity between positive pairs $\|\mathbf{p}' - \mathbf{q}'_+\|_2$, where $\mathbf{p}'$ and $\mathbf{q}'_+$ are defined in Eq.7. The result is shown in row 1 of Tab.5. We can see that the above optimization method performs worse than our GeoContrast, which further reflects the importance of negative pairs.

Description	PSNR ↑	SSIM ↑	LPIPS ↓
No negative pairs	23.81	0.812	0.350
negative pairs $N_{neg} = 32$	23.88	0.814	0.348
negative pairs $N_{neg} = 64$	24.19	0.819	0.342
negative pairs $N_{neg} = 128$	24.56	0.825	0.337
negative pairs $N_{neg} = 256$	24.73	0.828	0.335
negative pairs $N_{neg} = 512$	**24.81**	**0.831**	**0.333**
negative pairs $N_{neg} = 1024$	24.80	**0.831**	0.334
negative pairs $N_{neg} = 2048$	24.79	0.830	**0.333**

Table 5. **Ablation study on the ScanNet dataset [3] with respect to the number of negative pairs**.

Ablation on proportion of real data. Here we present the SSIM and LPIPS under varying proportions of real and synthetic data as shown in Fig.10. Similar to the curve of PSNR in Fig.6, the performance of our method continues to improve as the proportion of real data increases, but the performance saturates when the proportion of real data reaches 40%.



(a) SSIM. (b) LPIPS.

Figure 10. **Curves of SSIM and LPIPS of our method and IBRNet [15] with different proportions of real and synthetic data.**

D.2. NeRF Synthetic

We also conduct experiments on the NeRF synthetic dataset [13]. The NeRF synthetic dataset contains 8 objects, each of which has 100 training views and 200 test views at 800 × 800 resolution. The experimental settings are the

Method	PSNR ↑	SSIM ↑	LPIPS ↓
PixelNeRF [16]	22.65	0.808	0.202
IBRNet [15]	26.73	0.908	0.101
MVSNeRF [1]	25.15	0.853	0.159
GeoNeRF [8]	**28.33**	**0.938**	**0.060**
Neuray [10]	28.29	0.927	0.080
Ours	27.92	0.930	0.078

Table 6. **Quantitative comparisons on the NeRF Synthetic dataset [13].**

same as those on the DTU dataset [7] and LLFF dataset [12], where we use the Google Scanned Object datqset [4], forward-facing datasets [12, 15], and DTU dataset [7] as the training dataset. The quantitative and qualitative results of NeRF synthetic dataset are shown in Tab.6 and Fig.12 respectively. Compared with the baseline IBRNet [15], our method has a large performance improvement on NeRF synthetic dataset. Compared with the state-of-the-art methods [8, 10], our method also achieves comparable results. The reason why our method underperforms the state-of-the-art methods on the NeRF synthetic dataset may be that the multi-view consistency of synthetic data is relatively better than that of real data, resulting in limited improvement of the methods of learning multi-view consistent representations on synthetic data. On the other hand, the input to these state-of-the-art methods contains additional geometric information (such as depth) during testing, which will facilitate the modeling of the geometry, while our method's input only contains RGB images.

D.3. Additional Qualitative Results

In this section, we provide additional qualitative results. Fig.11 shows the qualitative results of IBRNet [15], MVSNeRF [1], GeoNeRF [8], Neuray [10] and our method on ScanNet dataset [3]. All models are trained on 3D-FRONT [5] and the experimental settings are the same as in Sec.5.1. Fig.12 shows the qualitative results on DUT dataset [7], LLFF dataset [12], and NeRF synthetic dataset [13], and the experimental settings are the same as in Sec.5.3.

E. Limitation and Failure Case

Our method generally achieves high-quality image rendering under the synthetic-to-real setting. However, previous generalizable NeRF methods [1, 8, 10, 15], as well as ours, struggle to generate high-quality images for highly blurred scenes, which are frequently found in real dataset. We show an example in Fig.13, where motion blur occurs in the pink boxed region and all methods fail to predict a sharp image. Deblur-NeRF [11] tries to recover a sharp NeRF from blurry input with the Deformable Sparse Kernel module. However, Deblur-NeRF only considers the per-scene optimization case. Rendering images with high bulr under-ing the synthetic-to-real generalization setting is a challenging problem and it can be an interesting and practical future direction.

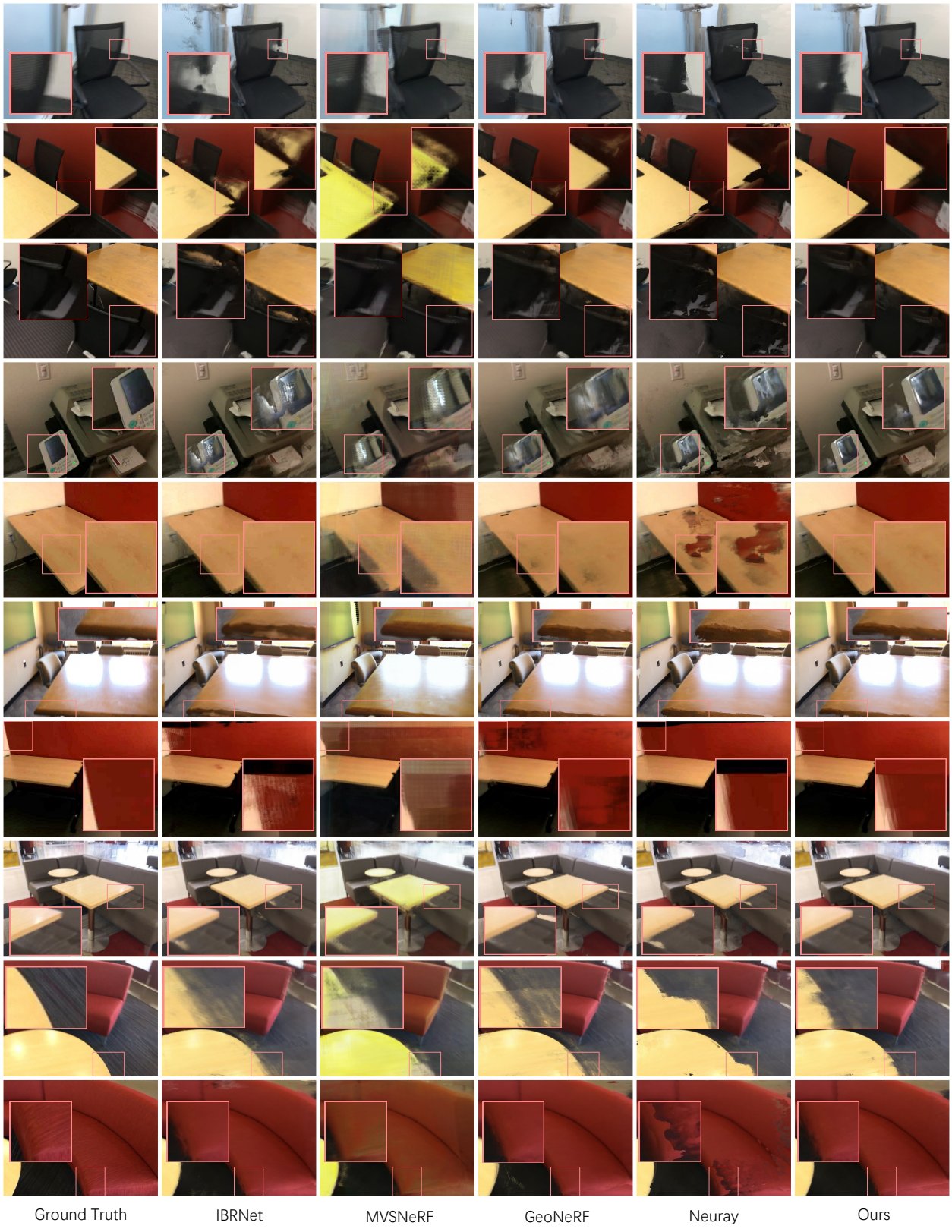


Figure 11. **Qualitative comparison on ScanNet dataset [3].** The first column shows the ground truth images. The last column shows the rendered images of our method. The remaining columns represent the images rendered by IBRNet [15], MVSNeRF [1], GeoNeRF [8], Neuray [9], respectively. Each model is trained on the synthetic dataset.

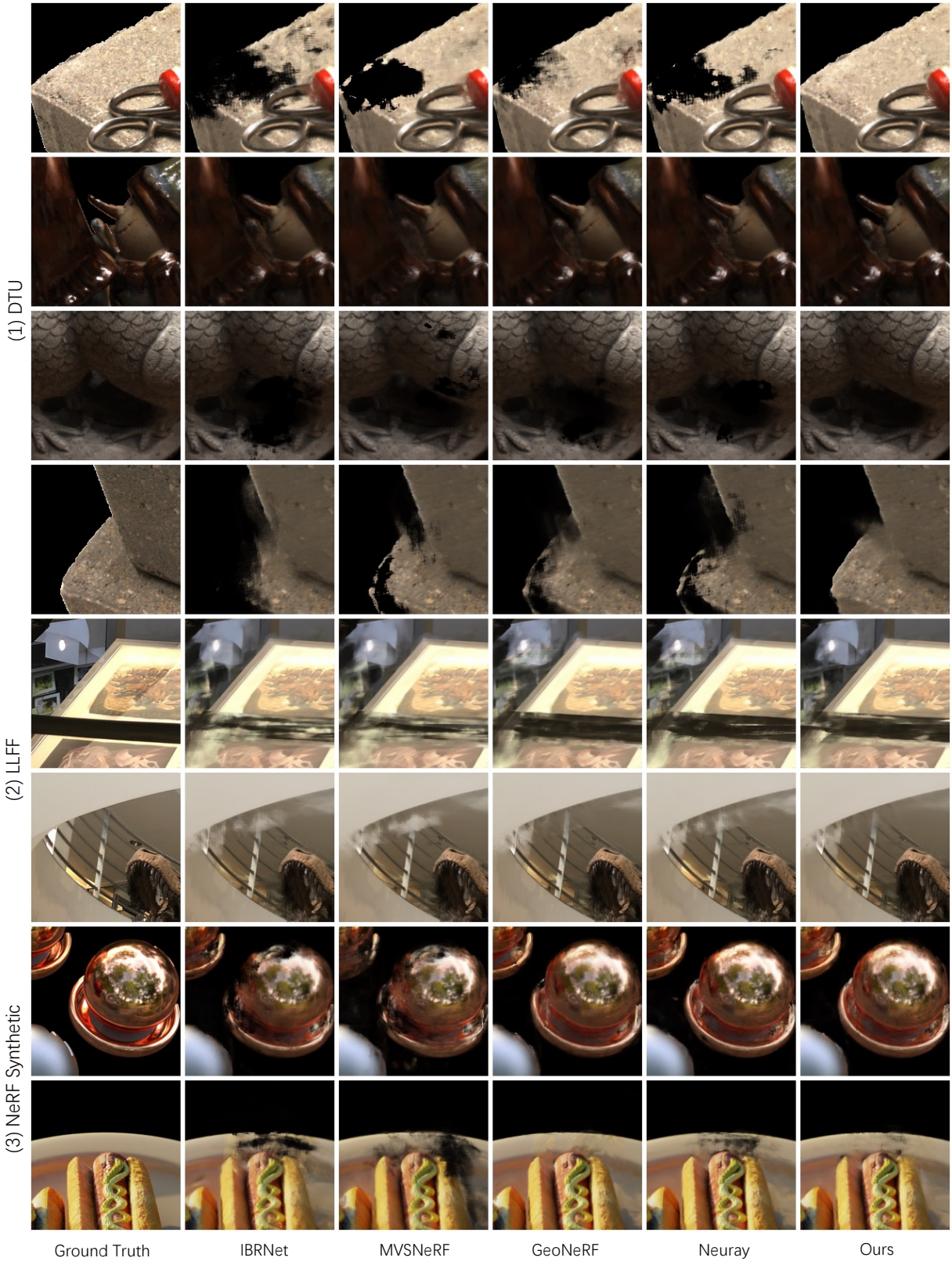


Figure 12. **Qualitative comparison on DTU dataset [7] (rows 1 to 4), LLFF dataset [12] (rows 5 to 6), and NeRF synthetic dataset [13] (rows 7 to 8)**. The first column shows the ground truth images. The last column shows the rendered images of our method. The remaining columns represent the images rendered by IBRNet [15], MVSNeRF [1], GeoNeRF [8], Neuray [9], respectively.

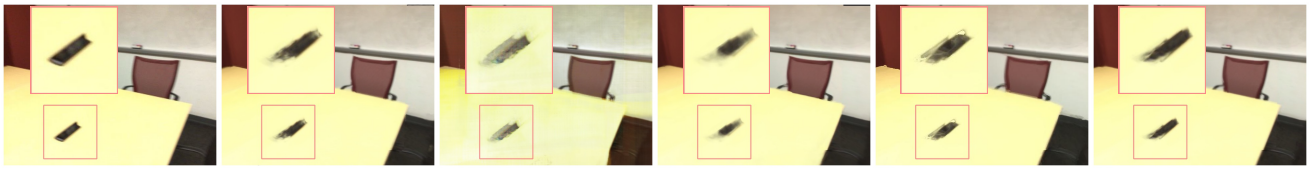


Figure 13. **Failure case on ScanNet dataset [3].** The first column shows the ground truth images. The last column shows the rendered images of our method. The remaining columns represent the images rendered by IBRNet [15], MVSNeRF [1], GeoNeRF [8], Neuray [9], respectively. Each model is trained on the synthetic dataset.

References

[1] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, 2021. 1, 2, 4, 5, 6, 7

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 3

[3] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 2, 5, 6, 1, 3, 4, 7

[4] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. *arXiv preprint arXiv:2204.11918*, 2022. 4

[5] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *ICCV*, 2021. 2, 1, 4

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[7] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *CVPR*, 2014. 6, 2, 4

[8] Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. Geonerf: Generalizing nerf with geometry priors. In *CVPR*, 2022. 1, 2, 4, 5, 6, 7

[9] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021. 5, 6, 7

[10] Yuan Liu, Sida Peng, Lingjie Liu, Qianqian Wang, Peng Wang, Christian Theobalt, Xiaowei Zhou, and Wenping Wang. Neural rays for occlusion-aware image-based rendering. In *CVPR*, 2022. 1, 2, 4

[11] Li Ma, Xiaoyu Li, Jing Liao, Qi Zhang, Xuan Wang, Jue Wang, and Pedro V Sander. Deblur-nerf: Neural radiance fields from blurry images. In *CVPR*, 2022. 4

[12] 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. *TOG*, 2019. 6, 2, 4

[13] 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. *Commun. ACM*, 2021. 3, 4, 6

[14] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016. 2

[15] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021. 7, 1, 2, 3, 4, 5, 6

[16] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021. 4

[17] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021. 2