ContraNeRF: Generalizable Neural Radiance Fields for Synthetic-to-real Novel View Synthesis via Contrastive Learning

Hao Yang[1]   Lanqing Hong[2]   Aoxue Li[2]   Tianyang Hu[2]
Zhenguo Li[2]   Gim Hee Lee[3*]   Liwei Wang[1,4*]

[1]Center for Data Science, Peking University   [2]Huawei, China   [3]School of Computing, National University of Singapore
[4]National Key Laboratory of General Artificial Intelligence, School of Intelligence Science and Technology, Peking University

{haoy@stu, wanglw@cis}.pku.edu.cn   gimhee.lee@comp.nus.edu.sg

{honglanqing, liaoxue2, hutianyang1, li.zhenguo}@huawei.com

Abstract

Although many recent works have investigated generalizable NeRF-based novel view synthesis for unseen scenes, they seldom consider the synthetic-to-real generalization, which is desired in many practical applications. In this work, we first investigate the effects of synthetic data in synthetic-to-real novel view synthesis and surprisingly observe that models trained with synthetic data tend to produce sharper but less accurate volume densities. For pixels where the volume densities are correct, fine-grained details will be obtained. Otherwise, severe artifacts will be produced. To maintain the advantages of using synthetic data while avoiding its negative effects, we propose to introduce geometry-aware contrastive learning to learn multi-view consistent features with geometric constraints. Meanwhile, we adopt cross-view attention to further enhance the geometry perception of features by querying features across input views. Experiments demonstrate that under the synthetic-to-real setting, our method can render images with higher quality and better fine-grained details, outperforming existing generalizable novel view synthesis methods in terms of PSNR, SSIM, and LPIPS. When trained on real data, our method also achieves state-of-the-art results. https://contranerf.github.io/

1. Introduction

Novel view synthesis is a classical problem in computer vision, which aims to produce photo-realistic images for unseen viewpoints [2,5,10,39,43]. Recently, Neural Radiance Fields (NeRF) [27] proposes to achieve novel view synthesis through continuous scene modeling through a neural network, which quickly attracts widespread attention due to its surprising results. However, the vanilla NeRF is actually designed to fit the continuous 5D radiance field of a given scene, which often fails to generalize to new scenes and datasets. How to improve the generalization ability of neural scene representation is a challenging problem.

Recent works, such as pixelNeRF [49], IBRNet [40], MVSNeRF [4] and GeoNeRF [20], investigate how to achieve generalizable novel view synthesis based on neural radiance fields. However, these works mainly focus on the generalization of NeRF to unseen scenes and seldom consider the synthetic-to-real generalization, i.e., training NeRF with synthetic data while testing it on real data. On the other hand, synthetic-to-real novel view synthesis is desired in many practical applications where the collection of dense view 3D data is expensive (e.g., autonomous driving, robotics, and unmanned aerial vehicle [37]). Although some works directly use synthetic data such as Google Scanned Objects [14] in model training, they usually overlook the domain gaps between the synthetic and real data as well as possible negative effects of using synthetic data. In 2D computer vision, it is common sense that synthetic training data usually hurts the model's generalization ability to real-world applications [1, 8, 47]. *Will synthetic data be effective in novel view synthesis?*

In this work, we first investigate the effectiveness of synthetic data in NeRF's training via extensive experiments. Specifically, we train generalizable NeRF models using a synthetic dataset of indoor scenes called 3D-FRONT [15], and test the models on a real indoor dataset called ScanNet [9]. Surprisingly, we observe that the use of synthetic data tends to result in more artifacts on one hand but better fine-grained details on the other hand (see Fig.1 and Sec.3.2 for more details). Moreover, we observe that models trained on synthetic data tend to predict sharper but less accurate volume densities (see Fig.2). In this case, better fine-grained details can be obtained once the prediction of

---

*[*]Joint last authorship.

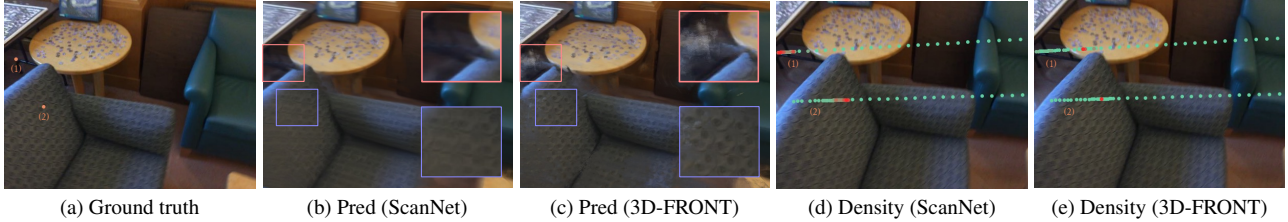| (a) Ground truth | (b) Pred (ScanNet) | (c) Pred (3D-FRONT) | (d) Density (ScanNet) | (e) Density (3D-FRONT) |

Figure 1. Fig.1a is the ground truth of the target image to be rendered. Fig.1b and Fig.1c are the rendered images when models are trained on ScanNet [9] and 3D-FRONT [15], respectively. Compare to Fig.1b, Fig.1c is more detailed (see purple box) but has more artifacts (see pink box). Fig.1d and Fig.1e further show the volume density (the redder the color, the higher the density) along the epipolar line projected from the orange points in Fig.4a to the source view. The model trained on 3D-FRONT prefers to predict the volume density with a sharper distribution, but sometimes the predicted volume density is not accurate (see line 1 in Fig.1e), resulting in severe artifacts.
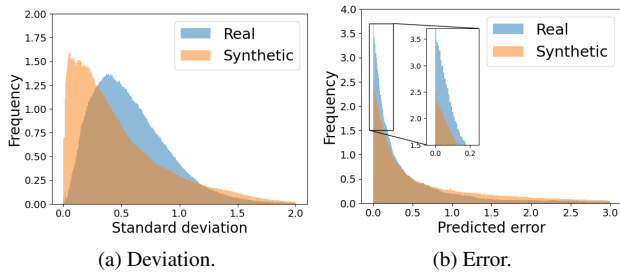

| (a) Deviation. | (b) Error. |

Figure 2. **Deviation and error of predicted depth when trained with synthetic and real data, respectively.** We count the deviation and error of the predicted depth for each pixel in the test dataset and plot them as the histogram. The depth is calculated by aggregating depth of the sampled points along the rendering ray, similar to the process of color rendering. Compared to the model trained with the real data, the model trained with the synthetic data tends to predict depths with small deviations but large errors, i.e., density distributions that are sharper but less geometrically accurate.

geometry (i.e., volume density) is correct, while severe artifacts will be produced otherwise. This motivates us to consider one effective way to generalize from synthetic data to real data in a geometry-aware manner.

To improve the synthetic-to-real generalization ability of NeRF, we propose ContraNeRF, a novel approach that generalizes well from synthetic data to real data via contrastive learning with geometry consistency. In many 2D vision tasks, contrastive learning has been shown to improve the generalization ability of models [21, 45, 46] by enhancing the consistency of positive pairs. In 3D scenes, geometry is related to multi-view appearance consistency [25, 40], and contrastive learning may help models predict accurate geometry by enhancing multi-view consistency. In this paper, we propose geometry-aware contrastive learning to learn a multi-view consistent features representation by comparing the similarities of local features for each pair of source views (see Fig.3). Specifically, for pixels of each source view, we first aggregate information along the ray projected

to other source views to get the geometry-enhanced features. Then, we sample a batch of target pixels from each source view as the training batch for contrastive learning and project them to other views to get positive and negative samples. The InfoNCE loss [38] is calculated in a weighted manner. Finally, we render the ray by learning a general view interpolation function following [40]. Experiments show that when trained on the synthetic data, our method outperforms the recent concurrent generalizable NeRF works [4, 20, 25, 40, 49] and can render high-quality novel view while preserving fine-grained details for unseen scenes. Moreover, under the real-to-real setting, our method also performs better than existing neural radiance field generalization methods. In summary, our contributions are:

1. Investigate the effects of synthetic data in NeRF-based novel view synthesis and observe that models trained on synthetic data tend to predict sharper but less accurate volume densities when tested on real data;

2. Propose geometry-aware contrastive learning to learn multi-view consistent features with geometric constraints, which significantly improves the model's synthetic-to-real generalization ability;

3. Our method achieves state-of-the-art results for generalizable novel view synthesis under both synthetic-to-real and real-to-real settings.

2. Related work

NeRF generalization. Recently, we have witnessed a major breakthrough in novel view synthesis by NeRF [29] and the following works [24, 27, 32, 48]. However, these methods can only be applied to a single scene and cannot be generalized to unseen scenes. Therefore, generalization NeRF [4, 20, 25, 33, 40, 49, 50] has subsequently become a hot research direction which aims to construct a neural radiance field on-the-fly using only a few images as input. IBRNet [40] uses a similar network but it synthesizes novel views by
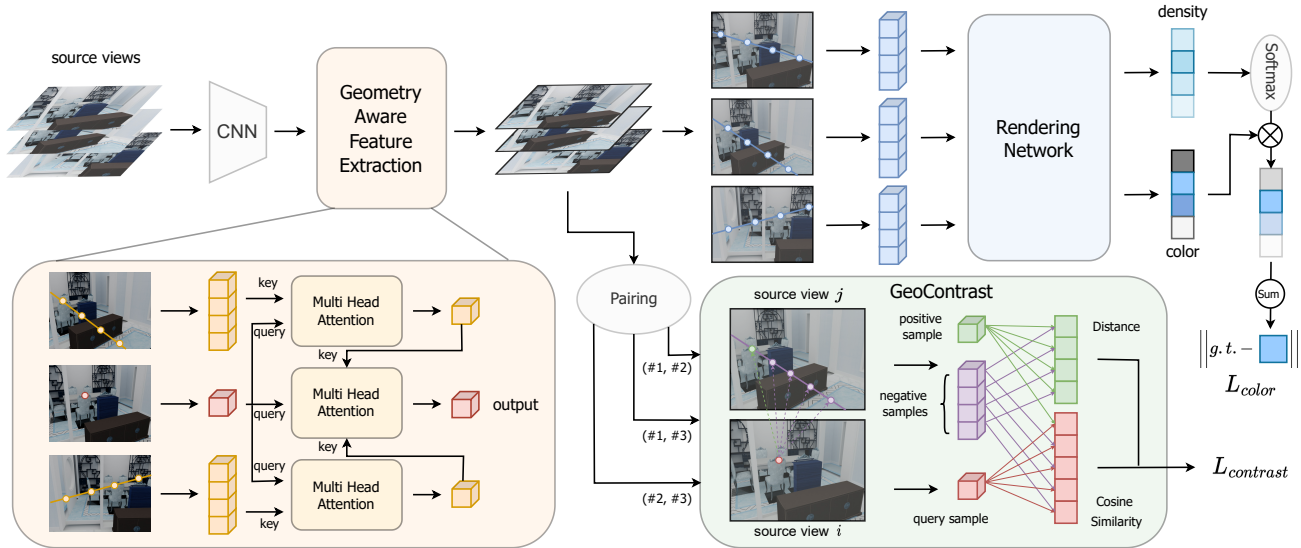
Figure 3. **Pipeline of our ContraNeRF**. 1) We first use a shared CNN to extract features for input source views. Then for each source view, we query features from other source views and aggregate them to get the geometrically enhanced feature maps (Sec.4.1). 2) For each pair of source views, we compute the contrastive loss using our GeoContrast (Sec.4.2). Specifically, for the pixel in the i-th source view, we project it to the j-th source view and sample a collection of projections to get positive and negative samples. Then the weighted contrastive loss is calculated by considering the distance between the positive sample and the negative samples. 3) Finally, for each ray in the target view, we compute colors and densities for a set of samples along the ray by aggregating local features from source views, and accumulate colors and densities to render images (Sec.4.3).

blending pixels from nearby views with weights and volume densities inferred by a network comprising an MLP and ray transformer. Neuray [25] further considers the visibility of each nearby view when constructing radiance fields and achieves good performance. MVSNeRF [4] and GeoN-eRF [20] leverage deep MVS techniques to achieve across-scene neural radiance field estimation for high-quality view synthesis. However, these works overlook the possible negative effects of using synthetic data and have difficulty in generalizing from synthetic data to real data.

NeRF with Geometry. Some recent work attempts to introduce geometry information into NeRF's training. Nerf-ingMVS [42] uses the depth priors to guide the optimization process of NeRF [29]. DS-NeRF [11] explores depth as additional supervision to guide the geometry learned by NeRF. RegNeRF [31] regularizes the geometry and appearance of patches rendered from unobserved viewpoints in sparse input scenarios. However, all these methods are designed for single-scene reconstruction without generalization ability.

Contrastive learning. Contrastive learning is a prevailing self-supervised learning technique [7, 16, 30, 36], which proposes to construct supervision information by treating each image as a class, training a model by pulling positive sample pairs closer while pushing negative sample pairs away with InfoNCE loss [38]. Compared with traditional supervised learning, contrastive learning has been shown to have better generalization ability for various 2D vision

tasks [21, 45, 46]. Previous works [6, 7, 16] usually take contrastive learning in an instance-level manner. Some recent works [18, 41, 44] try to apply contrastive learning at the pixel level for learning dense feature representations.

Synthetic-to-real generalization is a long-standing task that is desired in many applications, including autonomous driving, robotics, and unmanned aerial vehicle [37]. Although many works have considered Synthetic-to-real transfer for many tasks, such as classification [23], object detection [3], image deraining [47], and pose estimation [13], these methods cannot be directly applied to novel view synthesis. The synthetic-to-real generalization of neural radiance based novel view synthesis is seldom explored.

3. Problem Formulation

3.1. Generalizable Neural Radiance Fields

In this section, we first introduce the setting of Generalizable Neural Radiance Fields [4, 20, 25, 33, 40, 49, 50]. Let $\mathcal{D}_{train} = \{\mathcal{I}_i, \mathcal{K}_i, \mathcal{E}_i\}_{i=1}^{N_{train}}$ denote the training set, where $\mathcal{I}_i = \{\mathbf{I}_j\}_{j=1}^{N_{view}^i}$, $\mathcal{K}_i = \{\mathbf{K}_j\}_{j=1}^{N_{view}^i}$, $\mathcal{E}_i = \{\mathbf{E}_j = [\mathbf{R}_j, \mathbf{t}_j]\}_{j=1}^{N_{view}^i}$ are the images, camera intrinsic and extrinsic parameters of the i-th scenes respectively; N_{train} is the number of training scenes; and N_{view}^i is the number of camera views of the i-th scenes. The test data $\mathcal{D}_{test}$ is defined in a similar way. During training, the generalizable NeRF

3

model $\mathcal{M}(\cdot)$ renders novel views by aggregating the information of nearby source views of the same scene

$$\hat{\mathbf{I}}_r = \mathcal{M}(\{\mathbf{I}_i, \mathbf{K}_i, \mathbf{E}_i\}_{i=1}^{N_{near}}, \mathbf{K}_r, \mathbf{E}_r), \tag{1}$$

where $\mathbf{K}_r$ and $\mathbf{E}_r$ are the camera intrinsic and extrinsic of rendering view respectively; and N_{near} is the number of source views. Then we can train $\mathcal{M}(\cdot)$ by minimizing the loss between rendered images and ground truth

$$\mathcal{M} = \underset{\theta}{\arg\min} \|\hat{\mathbf{I}}_r - \mathbf{I}_r\|_2^2. \tag{2}$$

After training, we can render arbitrary views for unseen scenes by Eq.(1) without per-scene optimization.

3.2. Synthetic-to-real Generalization

In this paper, we consider the synthetic-to-real generalization of NeRF, which aims to train a NeRF model on synthetic data only and generalize it to real data. It is practical because synthetic data is usually easier to obtain. However, existing works of NeRF generalization [4,20,49] mainly use real data as the training set. A few works [40,50] tried to use a small portion of synthetic data together with real data in model training, without evaluating the possible negative effects of synthetic data. In this section, we first evaluate the effects of synthetic data in NeRF training via extensive experiments. Specifically, we choose 3D-FRONT [15] as our synthetic training set, which is a large-scale repository of synthetic indoor scenes with 18797 rooms. ScanNet [9] is used as the test set, which is also a dataset about indoor scenes. See Sec.5.1 for more experimental details. We adopt IBRNet [40] as the baseline considering its ease of use and promising performance. For comparison, we also train the model on ScanNet.

As illustrated in Fig.1b and Fig.1c, we observe that the model trained on synthetic data results in severer artifacts while better fine-grained details compared to the one trained on real data. We further visualize the volume density along the ray for pixels with severe artifacts, as shown in Fig.1d and Fig.1e. We can see that the model trained on the synthetic data tends to predict volume densities with a sharper but less accurate distribution, while the model trained on the real data tends to be more conservative. This is further demonstrated by Fig.2. The reason for these observations may be that the synthetic data is less noisy (real data usually involve inaccurate camera pose, image motion blur, and lighting changes), causing the model to be more confident in predictions and thus generate sharper densities. However, when generalizing to real data which are noisy, the model may fail to accurately predict the geometry of the scene, resulting in serious artifacts in rendered images. These observations inspire us to find a geometric-aware generalization method to solve the above-mentioned problem.

4. Method

To tackle the above problem, we propose **ContraNeRF**, a generalizable NeRF method that combines contrastive learning with geometry information, enabling generalization from synthetic data to real data. The overall framework is presented in Fig.3, and the following sections provide details of our method.

4.1. Geometry Aware Feature Extraction

Given nearby source views $\{\mathbf{I}_i, \mathbf{K}_i, \mathbf{E}_i\}_{i=1}^{N_{near}}$, we first use a shared CNN to extract features $\mathbf{F}_i$ from each image $\mathbf{I}_i$. Then we get the geometrically enhanced features $\{\mathbf{F}'_i\}_{i=1}^{N_{near}}$ by exchanging information between source views as described below.

Let $\mathbf{u}_i = [u, v]^\top$ denote the 2D coordinate of points on the i-th source view. Firstly, we obtain the ray of point $\mathbf{u}_i$ as a line $\mathcal{R}$ in world coordinates parametrized by δ as

$$\mathcal{R}(\delta) = \mathbf{t}_i + \delta\mathbf{R}_i\mathbf{K}_i^{-1}[\mathbf{u}_i^\top, 1]^\top, \tag{3}$$

where $\mathbf{K}_i, [\mathbf{R}_i, \mathbf{t}_i]$ represent the camera intrinsic and extrinsic parameters of the i-th source view respectively. Then, we sample a sequence of points $\mathbf{p}^s = \mathcal{R}(\delta^s)$ along the ray $\mathcal{R}$ and project them to the j-th source view as follows

$$d_j^s[\mathbf{v}_j^{s\top}, 1]^\top = \mathbf{K}_j\mathbf{R}_j^{-1}(\mathbf{p}^s - \mathbf{t}_j), s = 1, ..., N_s, \tag{4}$$

where $\mathbf{v}_j^s$ is the 2D coordinates of the projection in the j-th source view and d_j^s is the corresponding depth; and N_s is the number of sample points. Now we have a collection of projections $\{\mathbf{v}_j^s\}_{s=1}^{N_s}$ for the j-th source view.

Then we enhance the feature $\mathbf{F}_i$ by aggregating the features from other source views via cross-view attention, as illustrated in Fig.3. There are two stages in the aggregation. The first stage tries to aggregate the features of the projection points in the j-th source view. Formally, let $\mathbf{f}_i$ denote the local feature of $\mathbf{F}_i$ at position $\mathbf{u}_i$ and $\mathbf{g}_j^s$ denote the local feature of $\mathbf{F}_j$ at position $\mathbf{v}_j^s$. Then, we aggregate features $\{\mathbf{g}_j^s\}_{s=1}^{N_s}$ through Multi-Head Attention (MHA) layers

$$\mathbf{g}_j = \mathbf{MHA}(\mathbf{f}_i, \{\mathbf{g}_j^s + \mathcal{P}(s)\}_{s=1}^{N_s}), \tag{5}$$

where $\mathbf{f}_i$ is the query of MHA; $\{\mathbf{g}_j^s + \mathcal{P}(s)\}_{s=1}^{N_s}$ serve as the keys and values of MHA; and $\mathcal{P}(s)$ is the position embedding of s. Then, the second stage aims to aggregate the features of each source view and we achieve this through MHA too

$$\mathbf{f}'_i = \mathbf{MHA}(\mathbf{f}_i, \{\mathbf{g}_j\}_{j=1, j\neq i}^{N_{near}}), \tag{6}$$

where $\mathbf{f}_i$ is the query and $\{\mathbf{g}_j\}_{j=1, j\neq i}^{N_{near}}$ are the keys and values of MHA. We apply the above process for features of each source view and finally we get the geometrically enhanced features $\{\mathbf{F}'_i\}_{i=1}^{N_{near}}$.

Intuitively, our method tries to find the most similar features of $\mathbf{f}_i$ from other source views and aggregate them with $\mathbf{f}_i$. As a result, similar features will become more similar, which is loosely similar to clustering.

4.2. Geometry Aware Contrastive Learning

To better predict geometry, we enhance multi-view consistency through contrastive learning while taking into account the geometric constraints between views. Here we describe how our **Geometry Aware Contrastive Learning (GeoContrast)** works in detail.

As illustrated in Fig.3, we conduct contrastive learning between each pair of source views. Take the i-th source view and the j-th source view as an example. We first randomly sample a batch of pixels in the i-th source view, denoted as the $\{\mathbf{p}\}_{N_c}$, where $\mathbf{p} = (p, q)$ is the 2D coordinate of the sampled pixel and N_c is the number of samples. For each sampled pixel $\mathbf{p}$ in the i-th source view, we specify the corresponding positive sample $\mathbf{q}_+$ and negative samples $\{\mathbf{q}_-\}_{N_{neg}}$ in the j-th view according to ground truth depth, where N_{neg} is the number of the negative samples.

Positive pair. We take the pixels projected from the same 3D surface point in each source view as a positive pair. Specifically, for a sample $\mathbf{p}$ of the i-th source view, we obtain the 3D point $\mathbf{p}_3$ that $\mathbf{p}$ is projected from as $\mathbf{p}_3 = t_i + \delta_{\mathbf{p}} R_i K_i^{-1} \overline{\mathbf{p}}$ by taking Eq.(3), where $\overline{\mathbf{p}}$ is the homogeneous coordinates of $\mathbf{p}$ and $\delta_{\mathbf{p}}$ is the depth. Then we project 3D point $\mathbf{p}_3$ to the j-th source view as $\mathbf{q}_+$ by taking Eq.(4). As there may be occlusions in the scene, we only consider $(\mathbf{p}, \mathbf{q}_+)$ of unoccluded regions as positive pairs.

Negative pairs. We first project $\mathbf{p}$ to the j-th source view to get a collection of projections $\{\mathbf{q}_-\}_{N_{neg}}$ by taking Eq.(3) and Eq.(4). Then we take $\{(\mathbf{p}, \mathbf{q}_-)\}_{N_{neg}}$ as the negative pairs. Here, each negative pair is the projection of 3D point under different source views, so this sampling strategy can help the model better capture geometric information.

After determining the positive and negative samples, the contrastive loss for $\mathbf{p}$ is defined as

$$\mathcal{L}_{\mathbf{p}}^{\text{NCE}} = -\log \frac{\exp(\mathbf{p}' \cdot \mathbf{q}'_+/\tau)}{\exp(\mathbf{p}' \cdot \mathbf{q}'_+/\tau) + \sum_{\mathbf{q}_-} \lambda_{\mathbf{q}_-} \exp(\mathbf{p}' \cdot \mathbf{q}'_-/\tau)}. \quad (7)$$

where $\mathbf{p}'$ is the local feature of $\mathbf{F}'_i$ at position $\mathbf{p}$; $\mathbf{q}'_+$ and $\mathbf{q}'_-$ are the local features of $\mathbf{F}'_j$ at position $\mathbf{q}_+$ and $\mathbf{q}_-$ respectively; $\mathbf{F}'_i$ is the output of cross-view attention for the i-th source view as mentioned in Sec.4.1; τ is a learnable scalar temperature parameter; $\lambda_{\mathbf{q}_-}$ is the weight assigned to each $\mathbf{q}_-$. We calculate the weight $\lambda_{\mathbf{q}_-}$ by considering the distance between $\mathbf{q}_+$ and $\mathbf{q}_-$

$$\lambda_{\mathbf{q}_-} = N_{neg} \frac{\exp(\|\mathbf{q}_+ - \mathbf{q}_-\|_2/\tau')}{\sum_{\mathbf{q}_-} \exp(\|\mathbf{q}_+ - \mathbf{q}_-\|_2/\tau')}, \quad (8)$$

where τ' is a scalar temperature hyper-parameter, set by default to 10000. This weight measures the similarity between the positive sample and the negative sample. In this way, we can down-weight the influence of the negative samples that are similar to the positive sample. Finally, we average the loss for all samples $\mathbf{p}$ of each source view pair to form the final contrastive loss $\mathcal{L}_{\text{contrast}}$:

$$\mathcal{L}_{\text{contrast}} = \frac{1}{N_c} \sum_{\mathbf{p}} \mathcal{L}_{\mathbf{p}}^{\text{NCE}}. \quad (9)$$

Another way to choose the negative samples of $\mathbf{p}$ is to sample pixels from the j-th source view randomly, which are commonly used in previous pixel-level contrastive learning methods [18, 41, 44]. However, this sampling strategy does not consider the geometric constraints in the 3D scenarios. Intuitively, in our method, all the negative pairs are the projections of the non-surface 3D points and the positive pairs are the projections of the surface 3D points. It will be easier to model scene geometry by the following network since our GeoContrast makes the non-surface points and surface points more distinguishable, and experiments show that our sampling strategy performs much better (see Sec.5.4).

4.3. Rendering and Training

Following IBRNet [40], we calculate colors and densities for 3D points along the rendering ray $\mathbf{r}$ by checking the consistency among the features of each source view. During rendering, different from the volume rendering equation that is commonly used in most previous NeRF-related works [4, 29, 40, 49], we accumulate colors along the ray weighted by densities after softmax.

$$\hat{C}(\mathbf{r}) = \frac{1}{\sum_i \exp(\sigma_i)} \sum_i c_i \cdot \exp(\sigma_i), \quad (10)$$

where c_i, σ_i are the color and density for the i-th 3D sample point on the ray $\mathbf{r}$ respectively. We find that Eq.(10) works well in our experiments and it can speed up the convergence of network training without affecting model performance, which is also explored in [35]. It may be because the ray transformer in IBRNet [40] already has the ability to simulate light transport and occlusion in the radiance field.

Following [25, 40], we use the coarse-to-fine sampling strategy with 64 sample points in both stages. Then we can get the color loss

$$\mathcal{L}_{\text{color}} = \sum_{\mathbf{r} \in \mathcal{R}} \left[\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right], \quad (11)$$

where $\mathcal{R}$ is the set of rays in each batch; and $\hat{C}_c(\mathbf{r})$, $\hat{C}_f(\mathbf{r})$, and $C(\mathbf{r})$ are the coarse stage RGB prediction, fine stage RGB prediction, and ground truth for ray $\mathbf{r}$ respectively. Our final loss function is the sum of the contrastive loss and color loss.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{contrast}} + \mathcal{L}_{\text{color}}. \quad (12)$$
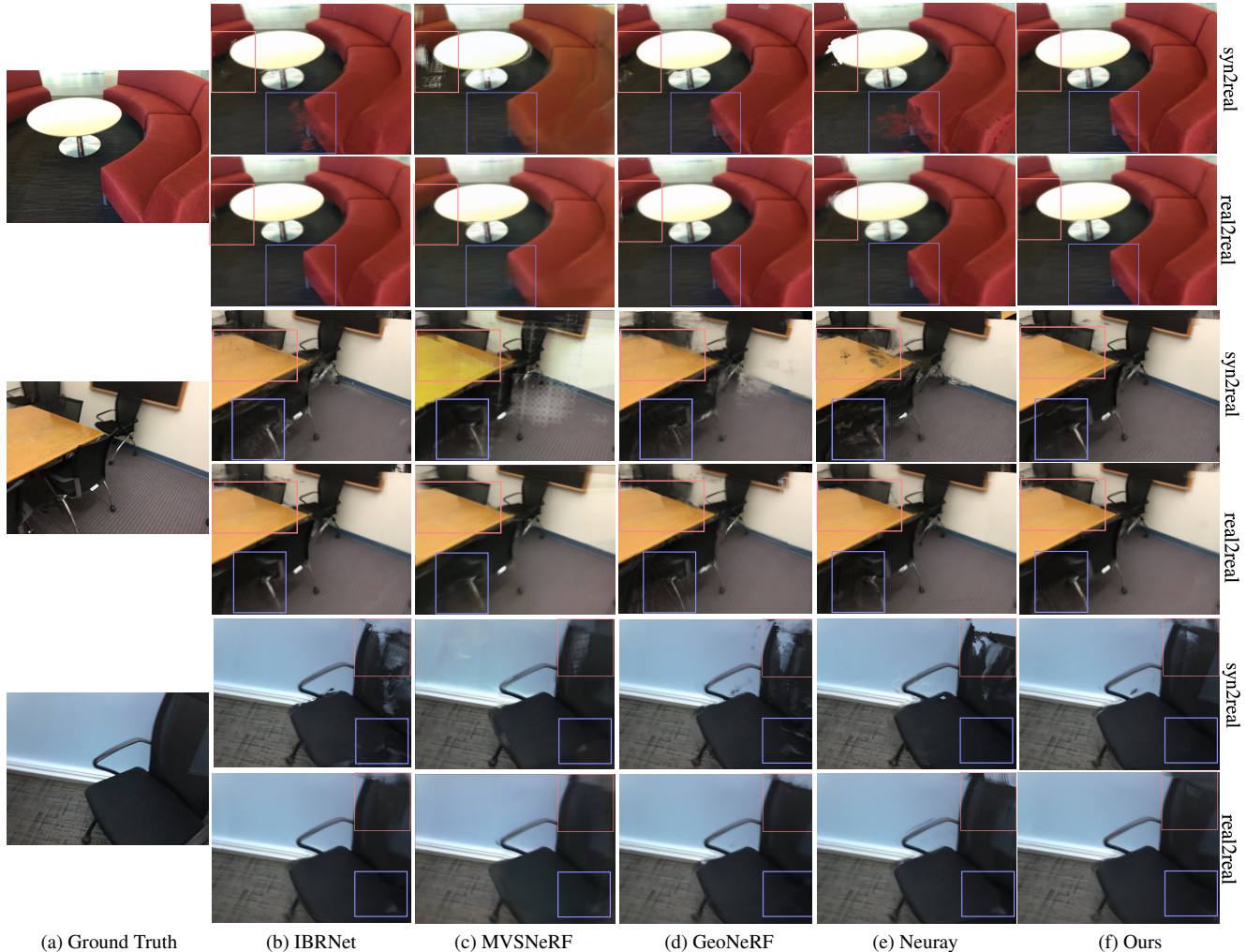
Figure 4. **Qualitative comparison on ScanNet dataset [9].** For each scene, two kinds of results for each method are shown, where the first row is the result after training on synthetic data and the second row is the result after training on real data. Our model more accurately preserves the details while it generates fewer artifacts than other generalizable NeRF methods when training on synthetic data.

5. Experiments

5.1. Experimental Settings

Datasets. In synthetic-to-real generalization, we choose 3D-FRONT [15] and ScanNet [9] as our synthetic training set and real test set respectively. (1) **3D-FRONT** is a large-scale, and comprehensive repository of synthetic indoor scenes. It contains 18,797 rooms diversely furnished by 3D objects. Following the data partition strategy in [4, 49], we randomly sample 88 scenes from 3D-FRONT as our synthetic training datasets. For each scene in 3D-FRONT, we sample 200 camera views and render each view at 640 × 480 resolution using BlenderProc [12]. (2) **ScanNet** is an RGB-D video dataset containing more than 1500 scans with 2.5 million views. Each scene contains 1K–5K views. We uniformly sample one-tenth of views and resize each image

to a resolution of 640 × 480 for use. In our experiments, we randomly select 88 scenes of ScanNet as our real training datasets and 8 scenes of ScanNet as our test datasets. On each test scene, we leave out 1/8 number of images as test views and the rest images as source views following [20, 25, 40].

Baselines and evaluation metrics. We compare our method with state-of-the-art generalizable NeRF methods, including PixelNeRF [49], IBRNet [40], MVSNeRF [4], GeoNeRF [20] and Neuray [25]. Following IBRNet, we evaluate all these methods using PSNR, SSIM, and LPIPS.

Implementation details. In contrastive learning, we sample a batch of 576 pixels for training and sample 512 negative pairs for each positive pair, where the parameters are tuned. Coarse and fine models share the same feature extractor. To render a novel view, we use 10 neighboring in-

Method	PSNR ↑	SSIM ↑	LPIPS ↓
PixelNeRF [49]	20.19 (22.44)	0.736 (0.774)	0.511 (0.450)
IBRNet [40]	23.67 (25.25)	0.807 (0.840)	0.355 (0.328)
MVSNeRF [4]	22.90 (24.90)	0.793 (0.824)	0.408 (0.357)
GeoNeRF [20]	23.67 (25.18)	0.797 (0.837)	0.349 (0.327)
Neuray [25]	22.75 (25.22)	0.785 (0.838)	0.369 (0.325)
Ours	**24.81** (**25.58**)	**0.831** (**0.847**)	**0.333** (**0.320**)

Table 1. **Quantitative comparisons on the ScanNet dataset [9]**. All methods are trained on the same scenes and tested on unseen real scenes. We report PSNR/SSIM (higher is better) and LPIPS (lower is better). The results of the model trained with synthetic data are shown outside the brackets, and the results of the model trained with real data are shown in brackets. Our method quantitatively outperforms prior work on all metrics.

put views as the source views, and we randomly sample 512 pixels from the novel view as a batch during training following [40]. We train the whole pipeline for 100k iterations using Adam optimizer [22] and the base learning rate is 10^{-3}. To achieve fair and accurate comparisons, we run all methods on the same experiment settings and use the official code to run the experiments. All experiments are conducted on the V100 GPU.

5.2. Results

We show the quantitative results in Tab.1 and visual comparisons in Fig.4. Tab.1 shows the superiority of our method with respect to the previous generalizable NeRF models on the synthetic-to-real setting. Note that some previous works, such as MVSNeRF [4] and GeoNeRF [20], also attempt to incorporate geometry into the model for performance improvement, but our approach differs from them in two ways. On the one hand, when performing multi-view feature fusion, our method samples 3D points randomly with an inverse depth distribution, while MVSNeRF and GeoNeRF only sample points at preset discrete positions to build the cost volume, which may introduce quantization errors. Moreover, with the help of contrastive learning, our method introduces consistency among multi-view features in a more straightforward way, while MVSNeRF and GeoNeRF only fuse the multi-view features without explicitly considering the consistency. These two points allow us to achieve better results. As shown in Fig.4, our method can produce images with fine-grained details in both geometry and appearance, and it generates fewer artifacts compared with the previous generalizable NeRF methods under both the synthetic-to-real and real-to-real settings. Note that in synthetic-to-real case, the interpolation-based generalizable NeRF method [20, 25, 40] performs better on color prediction than the method using the network to predict color [4]. This is because the color predicted by the interpolation-based methods comes from the input images, so there is no domain gap even under the synthetic-to-real setting. Meanwhile, we can see that the model trained on synthetic data

Method	PSNR ↑	SSIM ↑	LPIPS ↓
PixelNeRF [49]	19.40	0.463	0.447
IBRNet [40]	25.76	0.861	0.173
MVSNeRF [4]	23.83	0.723	0.286
GeoNeRF [20]	26.49	0.883	0.153
Neuray [25]	26.47	0.875	0.158
Ours	**27.69**	**0.904**	**0.129**

Table 2. **Quantitative comparisons on the DTU dataset [19]**. Our model is able to generate better results than previous state-of-the-art generalization NeRF models.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
PixelNeRF [49]	18.66	0.588	0.463
IBRNet [40]	25.17	0.813	0.200
MVSNeRF [4]	21.18	0.691	0.301
GeoNeRF [20]	**25.44**	0.839	0.180
Neuray [25]	25.35	0.818	0.198
Ours	**25.44**	**0.842**	**0.178**

Table 3. **Quantitative comparisons on the LLFF dataset [28]**. Our model is able to generate better results than previous state-of-the-art generalization NeRF models.

	PSNR ↑	SSIM ↑	LPIPS ↓
BaseModel	23.71 (25.27)	0.810 (0.840)	0.352 (0.327)
Random negative sampling	23.84 (25.30)	0.814 (0.840)	0.347 (0.326)
GeoContrast(w/o weight)	24.28 (25.40)	0.821 (0.843)	0.339 (0.325)
GeoContrast	24.53 (25.45)	0.825 (0.843)	0.337 (0.324)
Cross-view attention	24.25 (25.47)	0.820 (0.844)	0.342 (0.322)
Full model Ours	**24.81** (**25.58**)	**0.831** (**0.847**)	**0.333** (**0.320**)

Table 4. **Ablation study on the ScanNet dataset [9]**. The results of synthetic/real data are shown outside/in brackets, respectively. Refer to Sec.5.4 for details.

can retain more detail than the model trained on real data.

5.3. Other Benchmark Datasets

To further demonstrate the effectiveness of our method, we also conduct experiments in the settings described in the previous generalizable NeRF methods [4, 20, 25, 40, 49].
Evaluation datasets. We consider two widely adopted benchmarks, including the DTU dataset [19] and LLFF dataset [28]. Following [25], we select four objects (birds, tools, bricks, and snowman) as test objects for DTU dataset, and the test images of DTU dataset all use black backgrounds. On each test scene, we use 1/8 images as test views and the evaluation resolution is 800×600 for the DTU dataset, and 1008×756 for the LLFF dataset.
Training datasets. Following [25, 40], we use both the synthetic and real data for model training. For synthetic data, we use the Google Scanned Object dataset [14]. For real data, we use forward-facing training datasets [28, 40] and the rest training objects from the DTU dataset.
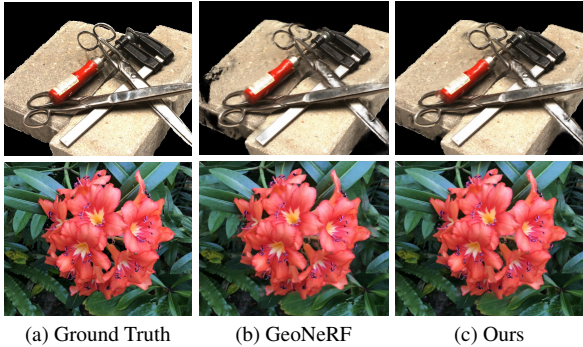
(a) Ground Truth   (b) GeoNeRF   (c) Ours

Figure 5. **Qualitative comparison on DTU dataset [19] and LLFF dataset [28].** Our method can render images with fewer artifacts.

Results. The quantitative results of DTU dataset and LLFF dataset are shown in Tab.2 and Tab.3 respectively. Our method works well on both DTU dataset and LLFF dataset. On DTU dataset, our method outperforms previous state-of-the-art generalizable NeRF methods by a large margin. This shows that our method can still work well even with a mixture of synthetic and real data. Fig.5 shows some visualization results on the DTU dataset and LLFF datasets. Similar to the results on ScanNet dataset, our model can produce images that are perceptually more similar to the ground truth than other methods.

5.4. Ablation study

Ablation on network design. We conduct ablation studies in the settings introduced in Sec.5.1 to validate the effectiveness of different design decisions. We first train a model, called 'BaseModel', that does not use cross-view attention and GeoContrast. Note that the 'BaseModel' still outperforms IBRNet [40] since we use Eq.(10) as the rendering equation. For 'Random negative sampling', we apply vanilla contrastive learning in 'BaseModel', where negative pairs are sampled from other views randomly, as mentioned in Sec.4.2. It works, but the improvement is marginal. Then we replace vanilla contrastive learning with our GeoContrast as shown at the third and fourth rows of Tab.4, where 'GeoContrast(w/o weight)' means the GeoContrast without weighted contrastive loss in Eq.(7). We can see that 'Geo-Contrast(w/o weight)' can bring improvement compared to random sampling, thanks to the introduction of geometric constraints. When equipped with weighted contrastive loss, GeoContrast can be further improved as shown at the 4-th rows of Tab.4. The results of the 5-th rows show that our cross-view attention is also helpful for the generalization. Finally, we combine cross-view attention with GeoContrast to form a complete model and it achieves the best results.

Ablation on proportion of real data. Although our method makes better use of synthetic data, there is still a performance gap between the model trained on synthetic data and the model trained on real data. A natural ques-
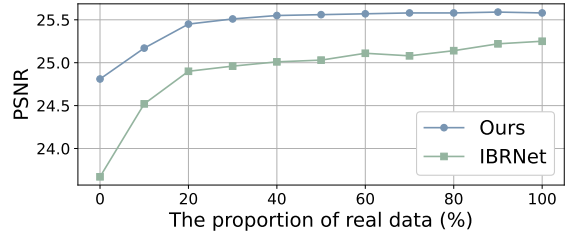

Figure 6. **Curves of PSNR of our method and IBRNet [40] with different proportions of real and synthetic data.**

tion is can we use a mix of real and synthetic data to boost model performance and how much real data do we need? Here, we conduct experiments with different proportions of real and synthetic data on our model and IBRNet [40]. Fig.6 shows the PSNR under varying proportions of real and synthetic data. We find that as the proportion of real data gradually increases, the performance of the model is improved. However, when the proportion of real data reaches a certain value, the performance will not continue to improve. For example, in our method, when the proportion of real data reaches 40%, the performance of the model saturates. This means that we only need to use a small amount of real data and a certain amount of synthetic data to achieve the same effect as using real data completely, while IBRNet needs to use more real data to achieve better results.

6. Conclusion

We present a generalizable neural radiance field method for synthetic-to-real novel view synthesis. Unlike the real-to-real novel view synthesis, models trained on synthetic data tend to predict sharper but less accurate volume densities on real data, which may result in severe artifacts in rendered images. To address this problem, we introduce geometry-aware contrastive learning to enable better modeling of scene geometry, thereby improving the model's ability to generalize from synthetic data to real data. Experiments demonstrate that our method can render high-quality images while preserving fine details in the synthetic-to-real setting.

References

[1] Rogerio Bonatti, Ratnesh Madaan, Vibhav Vineet, Sebastian Scherer, and Ashish Kapoor. Learning visuomotor policies for aerial navigation using cross-modal representations. In *IROS*, 2020. 1

[2] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *SIGGRAPH*, 2001. 1

[3] Qi Cai, Yingwei Pan, Chong-Wah Ngo, Xinmei Tian, Lingyu Duan, and Ting Yao. Exploring object relation in mean teacher for cross-domain detection. In *CVPR*, 2019. 3

[4] 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, 3, 4, 5, 6, 7

[5] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *SIGGRAPH*, 1993. 1

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

[7] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 3

[8] Zeyuan Chen, Yangchao Wang, Yang Yang, and Dong Liu. Psd: Principled synthetic-to-real dehazing guided by physical priors. In *CVPR*, 2021. 1

[9] 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. 1, 2, 4, 6, 7, 3, 5

[10] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *SIGGRAPH*, 1996. 1

[11] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *CVPR*, 2022. 3

[12] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, and Harinandan Katam. Blenderproc. *arXiv preprint arXiv:1911.01911*, 2019. 6

[13] Carl Doersch and Andrew Zisserman. Sim2real transfer learning for 3d human pose estimation: motion to the rescue. *NeurIPS*, 2019. 3

[14] 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. 1, 7, 4

[15] 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. 1, 2, 4, 6

[16] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 3

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

[18] Hanzhe Hu, Jinshi Cui, and Liwei Wang. Region-aware contrastive learning for semantic segmentation. In *ICCV*, 2021. 3, 5

[19] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *CVPR*, 2014. 7, 8, 2, 4, 6

[20] Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. Geonerf: Generalizing nerf with geometry priors. In *CVPR*, 2022. 1, 2, 3, 4, 6, 7, 5

[21] Daehee Kim, Youngjun Yoo, Seunghyun Park, Jinkyu Kim, and Jaekoo Lee. Selfreg: Self-supervised contrastive regularization for domain generalization. In *ICCV*, 2021. 2, 3

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[23] Akash Kumar, Arnav Bhavsar, and Rajesh Verma. Syn2real: Forgery classification via unsupervised domain adaptation. In *WACV Workshop*, 2020. 3

[24] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021. 2, 5, 6, 7

[25] 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. 2, 3, 5, 6, 7, 1, 4

[26] 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

[27] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *CVPR*, 2021. 1, 2

[28] 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. 7, 8, 2, 4, 6

[29] 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. 2, 3, 5, 4, 6

[30] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020. 3

[31] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, 2022. 3

[32] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021. 2

[33] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *ICCV*, 2021. 2, 3

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

[35] Yujiao Shi, Hongdong Li, and Xin Yu. Self-supervised visibility learning for novel view synthesis. In *CVPR*, 2021. 5

[36] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning. *arXiv:2005.10243*, 2020. 3

[37] Joanne Truong, Sonia Chernova, and Dhruv Batra. Bi-directional domain adaptation for sim2real transfer of embodied navigation agents. *IEEE Robot. Autom. Lett*, 2021. 1, 3

[38] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018. 2, 3

[39] Michael Waechter, Nils Moehrle, and Michael Goesele. Let there be color! large-scale texturing of 3d reconstructions. In *ECCV*, 2014. 1

[40] 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. 1, 2, 3, 4, 5, 6, 7, 8

[41] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. In *CVPR*, 2021. 3, 5

[42] Yi Wei, Shaohui Liu, Yongming Rao, Wang Zhao, Jiwen Lu, and Jie Zhou. Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In *ICCV*, 2021. 3

[43] Daniel N Wood, Daniel I Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H Salesin, and Werner Stuetzle. Surface light fields for 3d photography. In *SIGGRAPH*, 2000. 1

[44] Zhenda Xie, Yutong Lin, Zheng Zhang, Yue Cao, Stephen Lin, and Han Hu. Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning. In *CVPR*, 2021. 3, 5

[45] Jiawei Yang, Hanbo Chen, Jiangpeng Yan, Xiaoyu Chen, and Jianhua Yao. Towards better understanding and better generalization of few-shot classification in histology images with contrastive learning. *arXiv preprint arXiv:2202.09059*, 2022. 2, 3

[46] Xufeng Yao, Yang Bai, Xinyun Zhang, Yuechen Zhang, Qi Sun, Ran Chen, Ruiyu Li, and Bei Yu. Pcl: Proxy-based contrastive learning for domain generalization. In *CVPR*, 2022. 2, 3

[47] Rajeev Yasarla, Vishwanath A Sindagi, and Vishal M Patel. Syn2real transfer learning for image deraining using gaussian processes. In *CVPR*, 2020. 1, 3

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

[49] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021. 1, 2, 3, 4, 5, 6, 7

[50] Xiaoshuai Zhang, Sai Bi, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Nerfusion: Fusing radiance fields for large-scale scene reconstruction. In *CVPR*, 2022. 2, 3, 4

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

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 [15] as our synthetic training set and ScanNet [9] 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

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

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 $\mathbf{u}$

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

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 [4], GeoNeRF [20], Neuray [25], and our method) as shown in Fig.7. Note that for our method, we calculate the weight w^s following the Eq.10 as

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

Like IBRNet [40], previous NeRF generalization methods [4, 20, 25] 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 [15] 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

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

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

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

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

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

and

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

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 [15] as shown in Fig.8.

B.2. ScanNet

In this paper, we randomly select 8 scenes of ScanNet [9] 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 [9] as shown in Fig.8.

B.3. Other Benchmark Datasets

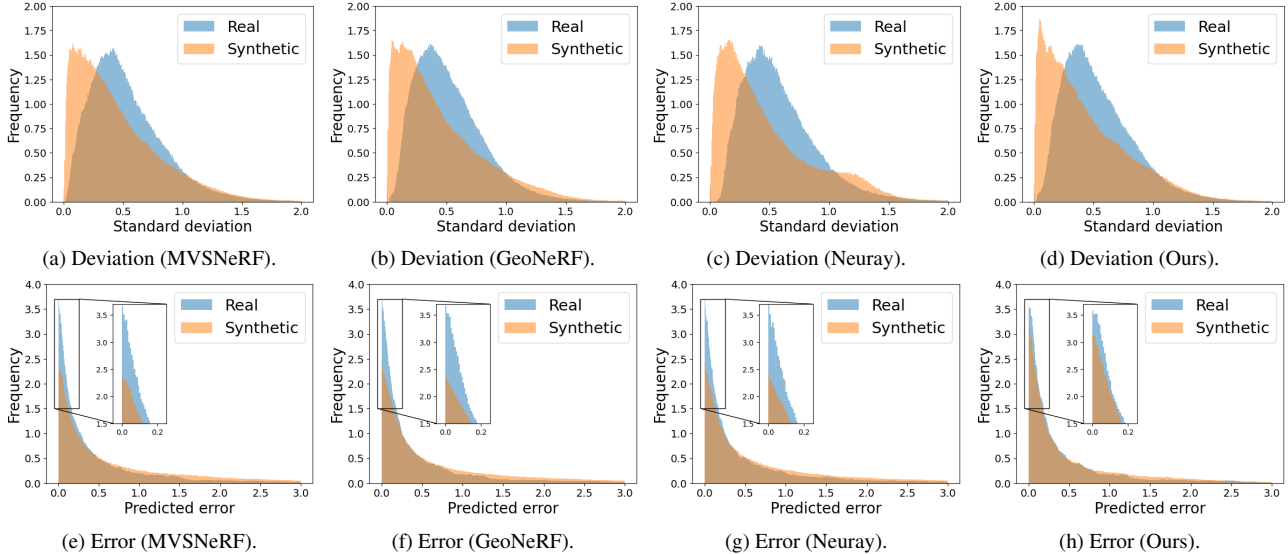During training, depth information is needed to determine the selection of positive pairs in our GeoContrast

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 [4] (column 1), GeoNeRF [20] (column 2), Neuray [25], and our method (column 4). Our method is able to predict more accurate depth while maintaining its sharpness under synthetic-to-real setting.
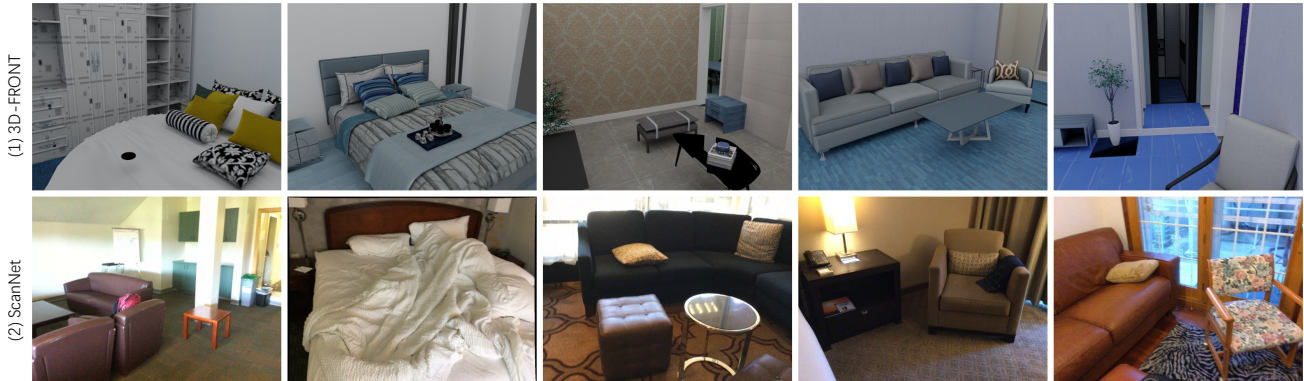

Figure 8. **Images in 3D-FRONT dataset [15] and ScanNet dataset [9]**. The first row shows the images in 3D-FRONT dataset. The second row shows the images in ScanNet dataset.

(Sec.4.2). Since DTU dataset [19] and LLFF dataset [28,40] only contain RGB images and not depth, we use COLMAP [34] to estimate the depth for each RGB image following [25]. 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 [17] 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 [25,40]. 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 [51] 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

1: Initialization: $\mathcal{E} \leftarrow \{\ \}, n \leftarrow 0, N_v \leftarrow 200, T_o \leftarrow 0.3, T_d \leftarrow 0.2$
2: **repeat**
3:     Sample camera view as **E**
4:     **if** $n = 0$ **then**
5:         $\mathcal{E} \leftarrow \mathcal{E} + \{\mathbf{E}\}$
6:         $n \leftarrow n + 1$
7:     **else**
8:         overlap $\leftarrow$ **Overlap**$(\mathbf{E}, \mathcal{E})$
9:         distance $\leftarrow$ **Distance**$(\mathbf{E}, \mathcal{E})$
10:        **if** overlap $\geq T_o$ **and** distance $\geq T_d$ **then**
11:            $\mathcal{E} \leftarrow \mathcal{E} + \{\mathbf{E}\}$
12:            $n \leftarrow n + 1$
13:        **end if**
14:     **end if**
15: **until** $n \geq N_v$
Output: A list of camera views $\mathcal{V}$.

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 [9] with respect to the number of negative pairs.**

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 [6]. 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.

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%.
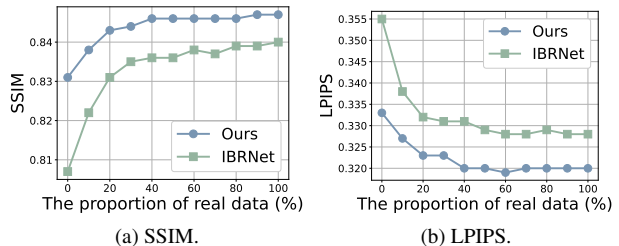

Figure 9. **Architecture of cross-view attention.**

work mainly following IBRNet [40], 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 [6], and a

(a) SSIM.   (b) LPIPS.

Figure 10. **Curves of SSIM and LPIPS of our method and IBRNet [40] with different proportions of real and synthetic data.**

D.2. NeRF Synthetic

We also conduct experiments on the NeRF synthetic dataset [29]. 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 [49]	22.65	0.808	0.202
IBRNet [40]	26.73	0.908	0.101
MVSNeRF [4]	25.15	0.853	0.159
GeoNeRF [20]	**28.33**	**0.938**	**0.060**
Neuray [25]	28.29	0.927	0.080
Ours	27.92	0.930	0.078

Table 6. **Quantitative comparisons on the NeRF Synthetic dataset [29]**.

same as those on the DTU dataset [19] and LLFF dataset [28], where we use the Google Scanned Object datqset [14], forward-facing datasets [28, 40], and DTU dataset [19] 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 [40], our method has a large performance improvement on NeRF synthetic dataset. Compared with the state-of-the-art methods [20, 25], 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 [40], MVS-NeRF [4], GeoNeRF [20], Neuray [25] and our method on ScanNet dataset [9]. All models are trained on 3D-FRONT [15] and the experimental settings are the same as in Sec.5.1. Fig.12 shows the qualitative results on DUT dataset [19], LLFF dataset [28], and NeRF synthetic dataset [29], 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 [4, 20, 25, 40], 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 [26] 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.
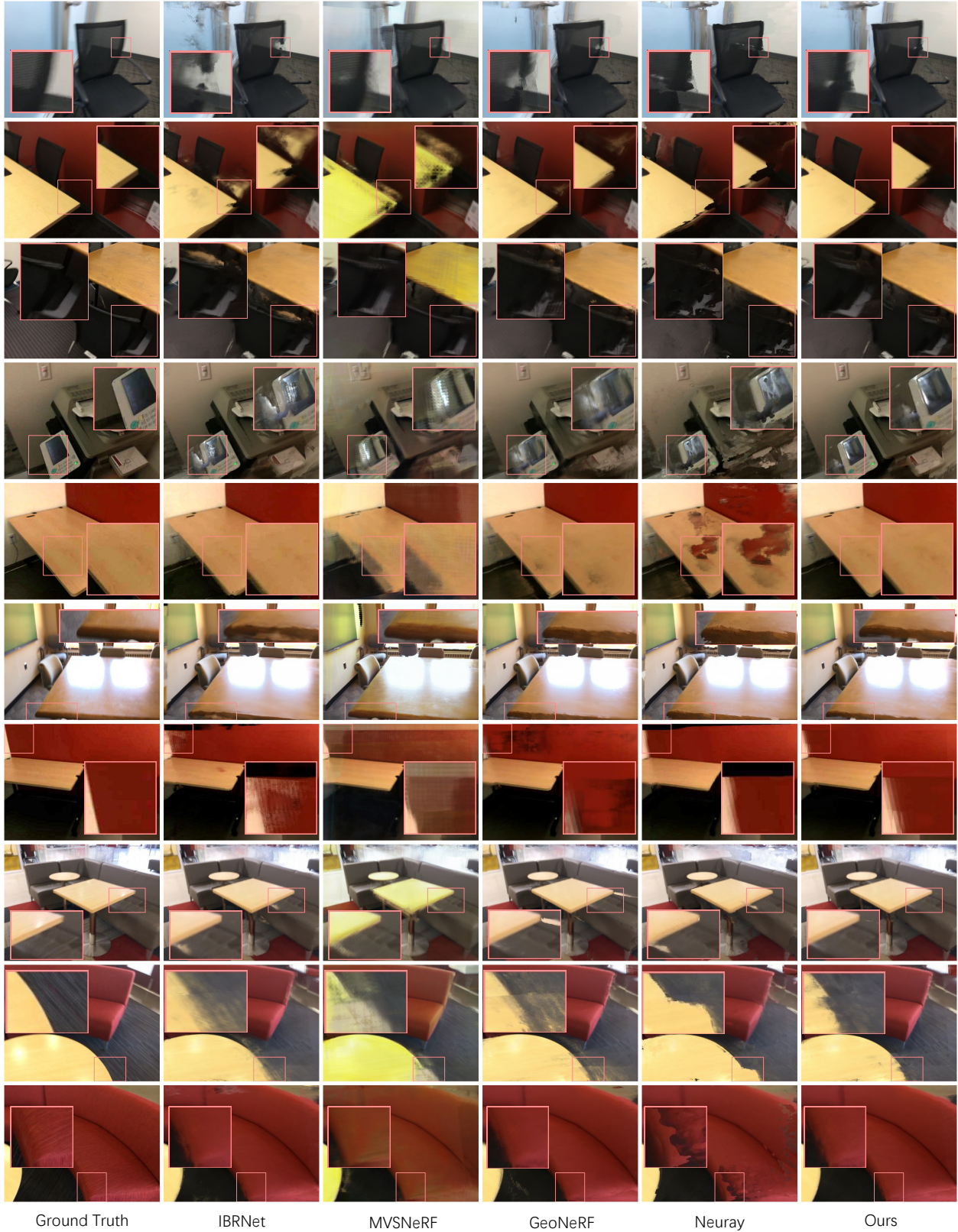
Figure 11. **Qualitative comparison on ScanNet dataset [9]**. The first column shows the ground truth images. The last column shows the rendered images of our method. The remaining columns represent the images rendered by IBRNet [40], MVSNeRF [4], GeoNeRF [20], Neuray [24], respectively. Each model is trained on the synthetic dataset.
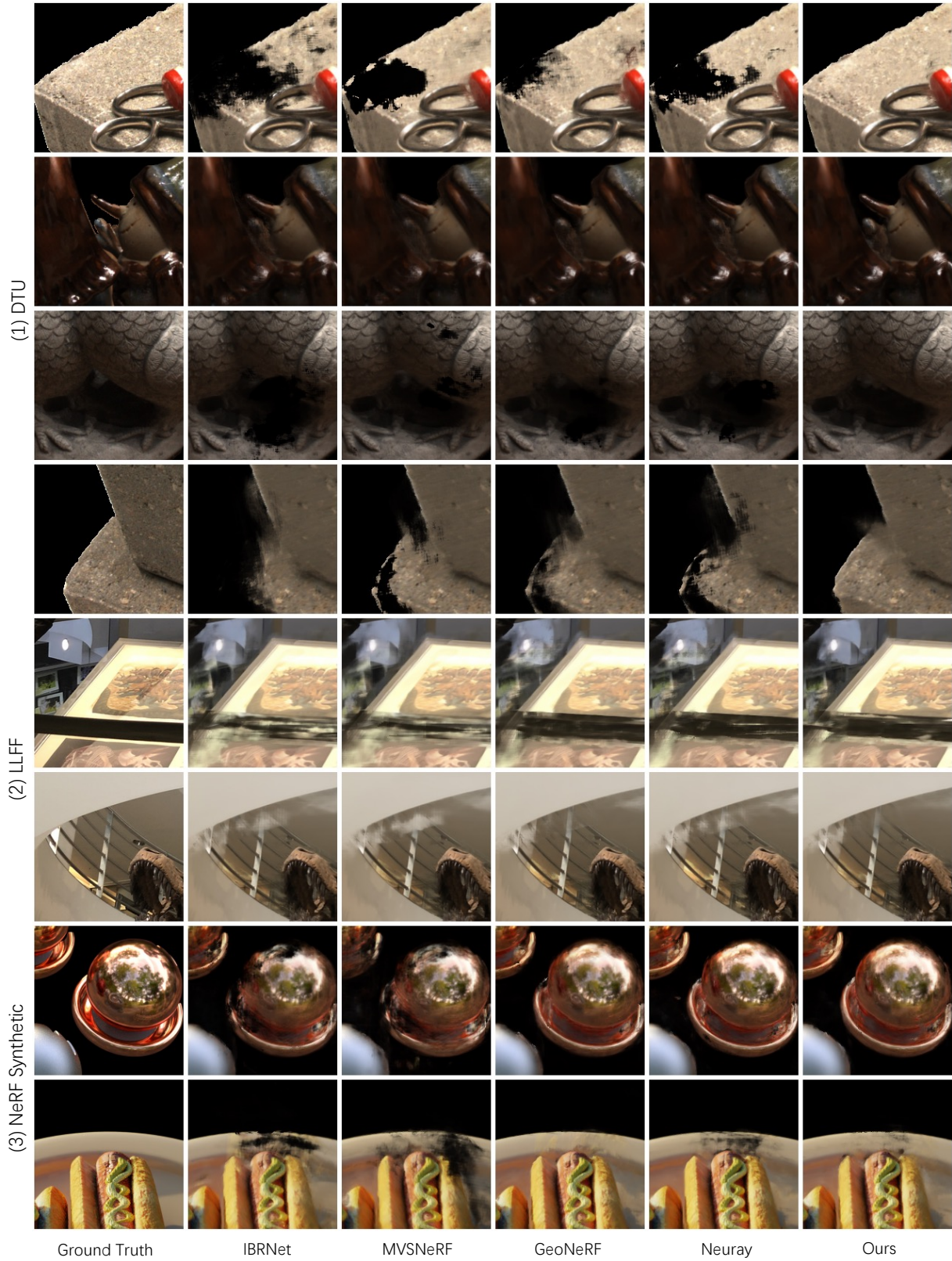
Figure 12. **Qualitative comparison on DTU dataset [19] (rows 1 to 4), LLFF dataset [28] (rows 5 to 6), and NeRF synthetic dataset [29] (rows 7 to 8)**. The first column shows the ground truth images. The last column shows the rendered images of our method. The remaining columns represent the images rendered by IBRNet [40], MVSNeRF [4], GeoNeRF [20], Neuray [24], respectively.

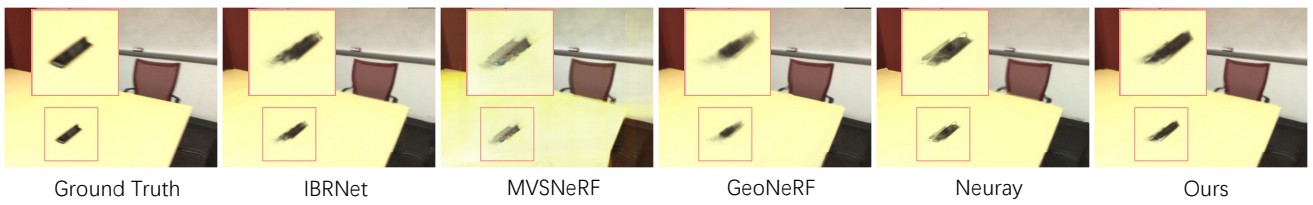| Ground Truth | IBRNet | MVSNeRF | GeoNeRF | Neuray | Ours |

Figure 13. **Failure case on ScanNet dataset [9]**. 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 [40], MVSNeRF [4], GeoNeRF [20], Neuray [24], respectively. Each model is trained on the synthetic dataset.