

# DoF-NeRF: Depth-of-Field Meets Neural Radiance Fields

Zijin Wu

[zjinwu@hust.edu.cn](mailto:zjinwu@hust.edu.cn)

School of AIA, Huazhong University  
of Science and Technology

Xingyi Li

[xingyi\\_li@hust.edu.cn](mailto:xingyi_li@hust.edu.cn)

School of AIA, Huazhong University  
of Science and Technology

Juewen Peng

[juewenpeng@hust.edu.cn](mailto:juewenpeng@hust.edu.cn)

School of AIA, Huazhong University  
of Science and Technology

Hao Lu

[hlu@hust.edu.cn](mailto:hlu@hust.edu.cn)

School of AIA, Huazhong University  
of Science and Technology

Zhiguo Cao\*

[zgcao@hust.edu.cn](mailto:zgcao@hust.edu.cn)

School of AIA, Huazhong University  
of Science and Technology

Weicai Zhong

[zhongweicai@huawei.com](mailto:zhongweicai@huawei.com)

CBG Search & Maps BU, Huawei

## ABSTRACT

Neural Radiance Field (NeRF) and its variants have exhibited great success on representing 3D scenes and synthesizing photo-realistic novel views. However, they are generally based on the pinhole camera model and assume all-in-focus inputs. This limits their applicability as images captured from the real world often have finite depth-of-field (DoF). To mitigate this issue, we introduce DoF-NeRF, a novel neural rendering approach that can deal with shallow DoF inputs and can simulate DoF effect. In particular, it extends NeRF to simulate the aperture of lens following the principles of geometric optics. Such a physical guarantee allows DoF-NeRF to operate views with different focus configurations. Benefiting from explicit aperture modeling, DoF-NeRF also enables direct manipulation of DoF effect by adjusting virtual aperture and focus parameters. It is plug-and-play and can be inserted into NeRF-based frameworks. Experiments on synthetic and real-world datasets show that, DoF-NeRF not only performs comparably with NeRF in the all-in-focus setting, but also can synthesize all-in-focus novel views conditioned on shallow DoF inputs. An interesting application of DoF-NeRF to DoF rendering is also demonstrated. The source code will be made available at <https://github.com/zjinwuzijin/DoF-NeRF>.

## CCS CONCEPTS

- Computing methodologies → Computer graphics; *Image manipulation; Image-based rendering;*

## KEYWORDS

neural radiance field, depth-of-field, novel view synthesis, image-based rendering

## ACM Reference Format:

Zijin Wu, Xingyi Li, Juewen Peng, Hao Lu, Zhiguo Cao, and Weicai Zhong. 2022. DoF-NeRF: Depth-of-Field Meets Neural Radiance Fields . In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22)*,

\*Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

MM '22, October 10–14, 2022, Lisboa, Portugal

© 2022 Association for Computing Machinery.

ACM ISBN 978-1-4503-9203-7/22/10...\$15.00

<https://doi.org/10.1145/3503161.3548088>

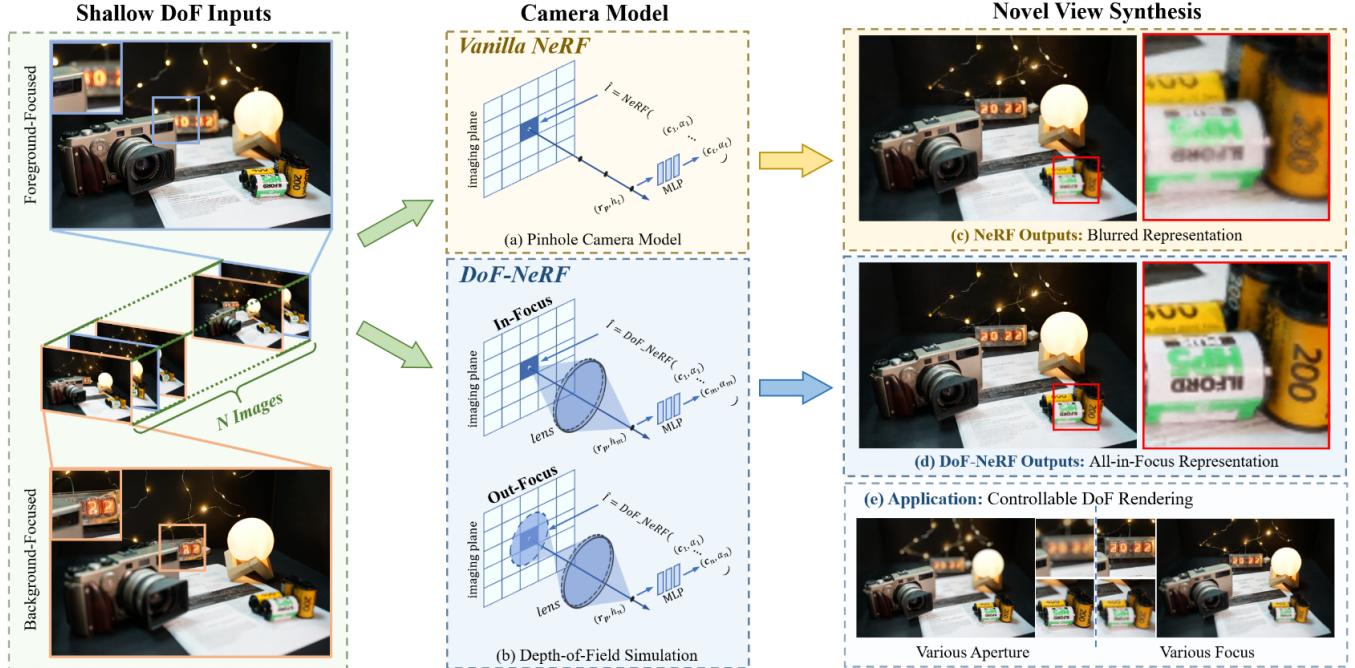
*October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 18 pages.  
<https://doi.org/10.1145/3503161.3548088>

## 1 INTRODUCTION

Novel view synthesis [3, 9, 19] is a long-standing problem in computer vision and graphics. As synthesizing novel views of a 3D scene from a sparse set of input images is a fundamental task for applications in augmented reality (AR) and virtual reality (VR), a substantial amount of work has been conducted to seek for solutions. Recently Neural Radiance Field (NeRF) [27], an implicit multi-layer perceptron (MLP) based model that regresses colors and densities from 3D coordinates and 2D viewing directions, shows an impressive level of fidelity on novel view synthesis. In particular, NeRF uses classical volume rendering techniques [17] to synthesize photo-realistic novel views by integrating the output colors and densities along emitted rays. Since the process of volume rendering is fully differentiable, NeRF can be optimized by minimizing the difference between the captured images and the rendered views.

However, NeRF and its variants [4, 24, 49] are generally based on the pinhole camera model and assume all-in-focus inputs, *i.e.*, both foreground and background are clear. In reality, images captured from the real world often have finite depth-of-field (DoF). Namely, points of light that do not lie on the focal plane are imaged to a circular region on the sensor plane, rather than to single points. The size of the circular region (dubbed circle of confusion, or CoC) is affected by the diameter of the aperture (aperture size) and the distance from the camera to the focal plane (focus distance). Hence, photos captured with a small aperture usually present wide DoF, *i.e.*, all objects are clear. In contrast, as the aperture diameter increases, objects that near the focal plane remain clear, but those far from the plane are blurred with a large CoC. This shallow DoF effect is ubiquitous in photography especially when shooting with a wide-aperture lens or taking close-up photos.

While NeRF has shown astonishing results for novel view synthesis, its performance deteriorates when processing shallow DoF inputs. This can boil down to the assumption of the pinhole camera model—when predicting pixel colors, NeRF only considers the emission of spatial points on the ray passing through the pixel and neglects the scattered radiance from neighboring rays. To resolve this issue, we present DoF-NeRF, a novel NeRF-based framework that enables NeRF to tackle shallow DoF inputs. Specifically, we introduce three key changes: (1) a differentiable representation of CoC to simulate the radiance scattered between rays; (2) learnable



**Figure 1:** Given a set of shallow depth-of-field (DoF) images, (c) performance of NeRF deteriorates due to its (a) pinhole camera model assumption. We present DoF-NeRF, a novel NeRF-based framework using (b) explicit DoF simulation to represent (d) all-in-focus scenes from shallow DoF inputs. An application of DoF-NeRF to (e) DoF rendering is also presented.

parameters to enable direct manipulation of the DoF effect; (3) a patch-based ray selecting method for efficient optimization. Our key insight is to simulate the DoF effect by an optical-conforming radiance scattering method parameterized with two learnable parameters: aperture size and focus distance. The two parameters of each training view can guide the optimization of NeRF to generate a clear 3D scene representation. Our optical modeling of DoF can not only synthesize all-in-focus novel views conditioned on shallow DoF inputs, but also provide highly controllable DoF rendering from novel viewpoints. Our main contributions include the following.

- We present DoF-NeRF, a novel neural rendering framework that can represent clear 3D scenes given shallow DoF inputs.
- We introduce the Concentrate-and-Scatter technique, a plug-and-play rendering modification for NeRF-based methods to simulate the DoF effect.
- We also contribute a new dataset for novel view synthesis of shallow DoF scenes. This dataset contains triplets of all-in-focus, foreground-focused, and background-focused from sparse viewpoints for each scene. Both real-world and synthetic data are included for further study.

## 2 RELATED WORK

### 2.1 Novel View Synthesis

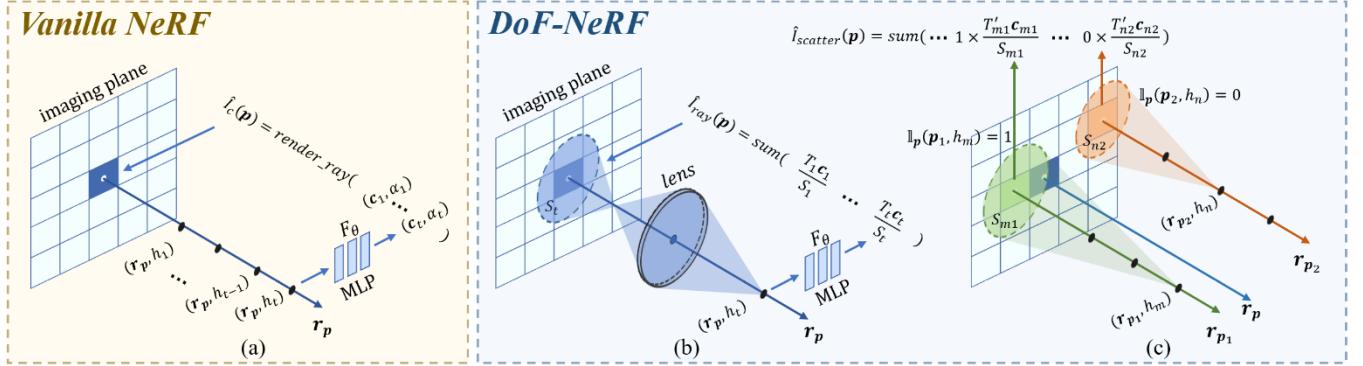
Novel view synthesis is a task of synthesizing novel camera perspectives from a set of input views and their corresponding camera poses. The research into novel view synthesis has a long history in computer vision and graphics community. Various approaches are investigated, including image-based rendering [3, 9, 19] and explicit

geometric representations such as voxel grids [16, 22, 48], point clouds [1, 6], triangle meshes [18, 33, 42], and multiplane images (MPIs) [7, 40, 52]. Recent studies [23, 26, 37, 38] have shown the superiority of implicit representations in rendering high quality novel views. For example, Mildenhall *et al.* propose Neural Radiance Field (NeRF) [27], an implicit MLP-based model that maps 3D coordinates plus 2D viewing directions to opacity and color values, which is capable of representing a complex 3D scene and rendering photo-realistic novel views.

However, drawbacks of NeRF remain, including entailing immense posed images and its high computational requirements for rendering novel views. To mitigate these issues, researchers have introduced improvements upon NeRF to extend its performance and applicability, such as faster training [4, 39], faster inference [8, 34], optimizing NeRF with low-light [25] or high dynamic range [11] images, improving generalization [28, 36], and representing dynamic scenes [21, 46]. Recently, Self-Calibrating NeRF [15] combines the pinhole camera, radial distortion, and a generic non-linear camera distortion for self-calibration by modeling distortion parameters. However, all these methods neglect the DoF effect. In this paper, we extend NeRF to simulate the aperture of lens and model the DoF effect by optimizing two learnable parameters, *i.e.*, aperture size and focus distance. This enables the synthesis of all-in-focus novel views with shallow DoF inputs as well as the rendering of 3D scenes with arbitrary aperture and focus distance settings.

### 2.2 DoF Rendering

Rendering DoF effects from a single all-in-focus image has been well studied in previous work. Some work [5, 12–14, 31] directly



**Figure 2: Basic principle of DoF-NeRF.** The vanilla NeRF (a) adopts a pinhole camera model which ignores the DoF effect. DoF-NeRF computes the observed color of ray  $r_p$  by adding (b) color diffusion  $\hat{l}_{ray}(p)$ : the predicted radiance of spatial points on ray  $r_p$  is diffused to a CoC on the imaging plane, and (c) radiance scattering  $\hat{l}_{scatter}(p)$ : the scattered radiance of spatial points on neighboring rays, e.g.,  $r_{p_1}, r_{p_2}$  may affect the observed color of  $r_p$ . An indicator function is used to distinguish points by the CoC diameter and the distance to ray  $r_p$ . The observed color of ray  $r_p$  is the combination of  $\hat{l}_{ray}(p)$  and  $\hat{l}_{scatter}(p)$ .

regresses a shallow DoF image using neural networks. However, these methods cannot adjust DoF effects as they are trained on EBB! dataset [12] which only provides pairs of wide and shallow DoF images. The DoF effects in [2, 30, 41, 45, 47, 51] are controllable but usually require an extra disparity map. Although the disparity map can be predicted by depth estimation, its accuracy is not guaranteed. Another challenge is the revealing of invisible background objects during rendering, because no image of other views is provided.

To address these problems, several methods focusing on synthesizing DoF effects on NeRF models have been proposed recently. RawNeRF [25] adopts a multi-plane representation to render DoF effects, while NeRFocus [43] proposes a frustum-based volume rendering to approximate the imaging of a thin lens model. However, both studies mentioned above still assume all-in-focus inputs. In this work, through appropriate optical modeling, we optimize the aperture diameters and the focus distances during training and render clear novel views with shallow DoF inputs.

### 3 PRELIMINARY

In this section, we briefly review the principle of NeRF. To represent a scene, NeRF optimizes a continuous function parameterized by an MLP network  $G_\Theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \alpha)$  which maps a spatial position  $\mathbf{x} = (x, y, z)$  and viewing direction  $\mathbf{d} = (\theta, \phi)$  to its corresponding emitted color  $\mathbf{c}$  and transparency  $\alpha$ . the observed color  $\hat{l}_n(\mathbf{p})$  of the pixel  $\mathbf{p}$  can be represented as an integral of all emitted colors weighted by the opaqueness along the camera ray  $\mathbf{r}_p(h)$ , or can be written as the weighted radiance at  $N_s$  sample points along a ray:

$$\hat{l}_n(\mathbf{p}) = \sum_{i=1}^{N_s} T_i (1 - \alpha(\mathbf{r}_p(h_i), \Delta_i)) \mathbf{c}(\mathbf{r}_p(h_i), \mathbf{d}), \quad (1)$$

where  $\Delta_i = h_{i+1} - h_i$  and

$$T_i = \prod_{j=1}^{i-1} \alpha(\mathbf{r}_p(h_j), \Delta_j) \quad (2)$$

denotes the accumulated transmittance along the ray, i.e., the probability of the emitted light travelling from  $h_1$  to  $h_i$  without hitting

other particles. Computing the color value of each pixel on the imaging plane via volume rendering above composites a complete image.

To optimize the continuous function  $G_\Theta$  from a set of input images  $\mathcal{I} = \{I_1, I_2, \dots, I_N\}$ , NeRF adopts the photometric error between the synthesized views  $\hat{\mathcal{I}}$  and corresponding observations  $\mathcal{I}$ :

$$\mathcal{L} = \sum_{n=1}^N \|I_n - \hat{I}_n\|_2^2. \quad (3)$$

Note that, vanilla NeRF assumes a linear pinhole camera model where the color value of a pixel  $\mathbf{p} = (u, v)$  on the imaging plane is only determined by a single camera ray  $\mathbf{r}(h) = \mathbf{o} + \mathbf{hd}$  that travels from the camera center  $\mathbf{o}$  and passes through the pixel  $\mathbf{p}$  along the viewing direction  $\mathbf{d}$ . Therefore, NeRF ignores the interference between rays and does not model the aperture structure, leading to its incapability to simulate the DoF effect.

### 4 APPROACH

Instead of assuming the linear pinhole camera model in volume rendering, we extend NeRF to incorporate the representation of CoC to simulate the DoF. This allows NeRF to approximate the DoF effect under arbitrary aperture size and focus distance configuration such that all-in-focus scenes can be represented with shallow DoF inputs.

The emergence of DoF consists of two steps: radiance of the spatial points scattering to a particular area and center ray accepting the scattered radiance from neighboring rays. We implement these two steps by combining optical models and classical volume rendering (Fig. 2). To optimize NeRF with shallow DoF images, we introduce two learnable parameters: the aperture parameters  $\mathcal{K} = \{K_1, K_2, \dots, K_N\}$  and the focus distances  $\mathcal{F} = \{F_1, F_2, \dots, F_N\}$  for  $N$  images in the training set  $\mathcal{I}$ . The aperture parameters and focus distances are jointly optimized in the training process.

In what follows, we first introduce our explicit aperture modeling and then explain how DoF-NeRF is optimized. The table of notations can be found in the appendix.

## 4.1 DoF in Radiance Field

The physical model of imaging and DoF has been well studied in geometric optics [10, 20]. Assume that we neglect the ray distortions caused by the lens. In an ideal optical system, a spatial point  $\mathbf{p}$  with the point-to-lens distance (object distance)  $h_t$  is projected to a circular region (CoC) on the imaging plane. The diameter of the region  $\delta(h_t)$  can be determined by the focal length  $f$ , aperture diameter  $D$ , and focus distance  $F$ , which amounts to the following equation

$$\delta(h_t) = fD \times \frac{|h_t - F|}{h_t(F - f)}. \quad (4)$$

Since the focus distance  $F$  and object distance  $h_t$  are often much larger than the focal length  $f$ , we can modify Eq. (4) such that

$$\delta(h_t) = fD \times \frac{|h_t - F|}{Fh_t} = fD \times \left| \frac{1}{F} - \frac{1}{h_t} \right| = K \times \left| \frac{1}{F} - \frac{1}{h_t} \right|, \quad (5)$$

where the product of focal length and aperture diameter can be replaced by an aperture parameter  $K = f \times D$ .

For simplicity, we assume that the emitted radiance  $\hat{C}(\mathbf{p}, h_t)$  of the spatial point evenly disperses to a CoC of diameter  $\delta(h_t)$ :

$$\hat{C}(\mathbf{p}, h_t) = \frac{(1 - \alpha(\mathbf{r}_p(h_t), \Delta_t)) c(\mathbf{r}_p(h_t), d)}{S(h_t)}, \quad (6)$$

where  $S(h_t) = \pi\delta^2(h_t)/4$  denotes the area of the CoC.

Different from the pinhole camera assumption, the observed color of the ray  $\mathbf{r}_p$  can also be affected by spatial points whose CoC radius is larger than the distance to the ray  $\mathbf{r}_p$ . Apparently, points on neighboring rays only contribute to the radiance accumulation without affecting the transmittance  $T(\mathbf{r}_p)$ . Given  $M$  rays specific to pixels  $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_M\}$  on which the scattered radiance of spatial points may affect the color prediction of center ray  $\mathbf{r}_p$ , we can compute the color  $\hat{I}$  of pixel  $\mathbf{p}$  by summing up emitted radiance, without changing the transmittance coefficient. Specifically, computing the observed color  $\hat{I}(\mathbf{p})$  can be split into two parts:  $\hat{I}_{ray}(\mathbf{p})$  and  $\hat{I}_{scatter}(\mathbf{p})$  which respectively represent the diffused radiance on ray  $\mathbf{r}_p$  and the scattered radiance from other rays. This takes the form

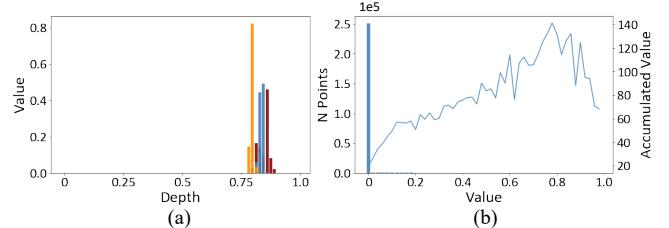
$$\hat{I}_{ray}(\mathbf{p}) = \sum_{i=1}^{N_s} T_i \hat{C}(\mathbf{p}, h_i), \quad (7)$$

$$\hat{I}_{scatter}(\mathbf{p}) = \sum_{j=1}^M \sum_{i=1}^{N_s} \mathbb{1}_{\delta(h'_i) > 2\|\mathbf{p} - \mathbf{p}_j\|_2^2} T'_{ij} \hat{C}(\mathbf{p}_j, h'_i), \quad (8)$$

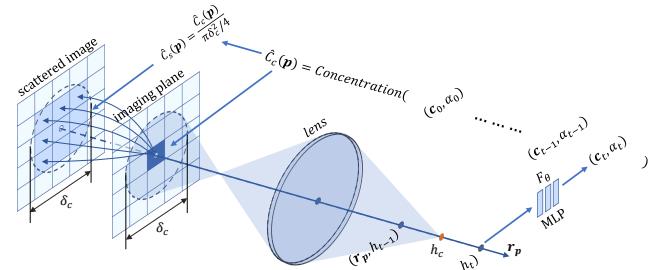
$$\hat{I}(\mathbf{p}) = \hat{I}_{ray}(\mathbf{p}) + \hat{I}_{scatter}(\mathbf{p}), \quad (9)$$

where  $h_i$  and  $h'_i$  denote the depth of the  $i$ -th sample point on the center ray and neighboring rays, respectively;  $T_i$  denotes the accumulated transmittance of the  $i$ -th sample point on the ray  $\mathbf{r}_p$ , and  $T'_{ij}$  denotes the transmittance of scattered radiance from the  $i$ -th sample point on the ray specific to pixel  $\mathbf{p}_j$ . An indicator function  $\mathbb{1}$  is used to distinguish points by the CoC diameter  $\delta(h'_i)$  and the distance to ray  $\mathbf{r}_p$ .

However, computing the color of the ray  $\mathbf{r}_p$  directly following Eq. (6)~(9) can be rather inefficient. It requires traversing all the spatial points on neighboring rays to obtain the color of the center rays.



**Figure 3: Visualization of volume coefficients  $K_{volume}$ .** In (a), we randomly choose 3 rays from the 3D scene and visualize the volume coefficient  $K_{volume}$  of sample points on each ray according to the depth. Different colors in (a) denote different rays. In (b) the histogram shows the value distribution of  $K_{volume}$  of 1024 selected rays, and the polyline shows the accumulated  $K_{volume}$  of points in each bin of the histogram.



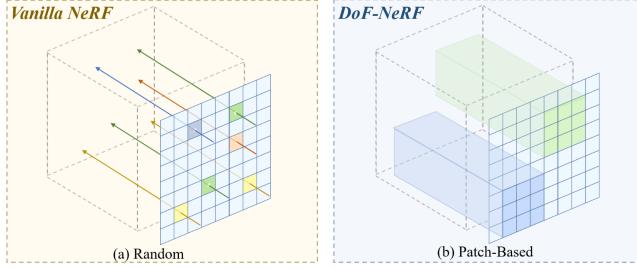
**Figure 4: Illustration of our concentrate-and-scatter rendering.** To reduce the cost of computation, we first concentrate the emitted radiance of points on the ray  $r_p$  (right side of the imaging plane) to the concentration depth  $h_c$  (orange point) and calculate the corresponding CoC diameter. Then we scatter the concentrated radiance to the neighboring areas within its CoC diameter (left side of the imaging plane).

To mitigate this issue, we analyze a fact of the volume rendering and propose our solution in what follows. For spatial points on the ray  $\mathbf{r}_p$ , we evaluate their contributions to the color prediction with a volume rendering coefficient  $K_{volume}$ :

$$K_{volume}(\mathbf{r}_p, h_t) = T_t (1 - \alpha(\mathbf{r}_p(h_t), \Delta_t)). \quad (10)$$

We randomly choose 1024 rays from input views and visualize the distribution of the volume coefficients (see Fig. 3). For most rays, points with large volume coefficients gather in a narrow depth interval. The distribution of the volume coefficient indicates that most spatial points do not affect the observed color but incur considerable computational overhead.

Nonetheless, simply ignoring those points may lead to the brightness change of rendered images. In the view of the volume coefficients distribution, we introduce the idea of concentrate-and-scatter rendering. As shown in Fig. 4, the core idea is to concentrate the emitted radiance of all the spatial points along the ray to the concentration depth  $h_c$ , and to scatter the concentrated radiance to its corresponding CoC. Since the scattered radiance does not affect the transmittance on the ray  $\mathbf{r}_p$ , the concentration of the radiance follows the original volume rendering. For  $N_s$  sampled points along the ray  $\mathbf{r}_p$ , the concentration depth  $h_c$ , CoC diameter  $\delta_c$  of the



**Figure 5: Illustration of our patch-based optimization.** Instead of randomly choosing rays (a) as in NeRF, we optimize DoF-NeRF by selecting rays from a concentrated area in each iteration (b).

concentration point, and the scattered radiance  $\hat{C}_s(\mathbf{p})$  can be formulated by

$$h_c(\mathbf{p}) = \frac{\sum_{i=1}^{N_s} K_{volume}(\mathbf{r}_p, h_t) h_i}{\sum_{i=1}^{N_s} K_{volume}(\mathbf{r}_p, h_t)}, \quad (11)$$

$$\delta_c(\mathbf{p}) = K \left| \frac{1}{F} - \frac{1}{h_c} \right|, \quad (12)$$

$$\hat{C}_s(\mathbf{p}) = \frac{\sum_{i=1}^{N_s} K_{volume}(\mathbf{r}_p, h_t) c(\mathbf{r}_p(h_i), \mathbf{d})}{\pi \delta_c^2(\mathbf{p}) / 4}. \quad (13)$$

For  $P$  rays with respect to the pixels  $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_P\}$  whose radiance may affect the color prediction of ray  $\mathbf{r}_p$ , including the ray  $\mathbf{r}_p$  itself, the observed color of ray  $\mathbf{r}_p$  can be defined by

$$\hat{I}(\mathbf{p}) = \sum_{j=1}^P \mathbb{1}_{\delta_c(\mathbf{p}) > 2 \|\mathbf{p} - \mathbf{p}_j\|_2^2} \hat{C}_s(\mathbf{p}_j). \quad (14)$$

We implement the algorithm with the CuPy package to achieve a significant speedup. The detail of the algorithm can be found in the appendix.

## 4.2 Ray Selection

When optimizing the scene representation, vanilla NeRF randomly chooses  $N_{rand}$  rays from all input views. Since the simulation of the DoF effect requires to consider radiance scattered from neighboring rays, a straight-forward approach is to compute several rays that may affect the center ray. This method is of low efficiency in training as it only considers the observed color of the center ray in a single iteration. Yet another approach is to compute the concentrated radiance of all rays from one imaging plane and to scatter every pixels on the whole imaging plane. However, it entails to compute every ray of the whole imaging plane in each iteration, which leads to an unacceptable cost in computing.

We adopt a patch-based ray selection method (Fig. 5) which can be considered as a compromise between the two approaches above. We construct a group of anchors, where each anchor is set every  $N_{anchor}$  pixels on the imaging plane of inputs and determines the center of an  $N_{patch} \times N_{patch}$  patch. In each iteration, we randomly choose one patch and guide rays passing through pixels in the selected patch. Observed colors of the pixels are computed using the concentrate-and-scatter rendering mentioned above.

---

### Algorithm 1: Joint Optimization of DoF-NeRF

---

```

Input:  $N$  images  $\mathcal{I} = \{I\}_{i=1}^N$ 
Output: NeRF Model  $G_\Theta$ , aperture  $[\hat{K}]_{i=1}^N$ , focus distance  $[\hat{F}]_{i=1}^N$ 
1 import torch.nn as nn
2  $[\hat{K}] = nn.Parameter(shape=(N, 1), requires_grad=True)$ 
3  $[\hat{F}] = nn.Parameter(shape=(N, 1), requires_grad=True)$ 
4  $G_\Theta = \text{NeRF\_Model}(requires\_grad=True)$ 
5 for  $i$  in range( $N_{iters}$ ) do
6   if  $i < N_{pretrain}$  then
7      $[\mathbf{r}]_i, I_i = \text{random\_rays}(\mathcal{I})$ 
8      $\hat{I}_i = \text{Volume\_Rendering}(G_\Theta, [\mathbf{r}]_i)$  # Sec. 3
9      $\mathcal{L} = \text{loss}(\hat{I}_i, I_i)$  # Eq. 3
10     $\mathcal{L}.\text{backward}()$ 
11    optimizer.update( $\hat{\Theta}$ )
12  else
13     $[\mathbf{r}]_i, I_i = \text{patch\_rays}(\mathcal{I})$  # Sec. 4.2
14     $[\hat{C}]_i, [h_c]_i = \text{Concentration}(G_\Theta, [\mathbf{r}]_i)$  # Eq. 13
15     $[\delta_c]_i = \text{CoC\_Radius}([\hat{C}]_i, \hat{K}_i, \hat{F}_i)$  # Eq. 12
16     $\hat{I}_i = \text{Scatter}([\hat{C}]_i, [\delta_c]_i)$  # Eq. 14
17     $\mathcal{L} = \text{loss}(\hat{I}_i, I_i)$  # Eq. 3
18     $\mathcal{L}.\text{backward}()$ 
19    optimizer.update( $\hat{\Theta}, [\hat{K}], [\hat{F}]$ )
20  end
21 end

```

---

## 4.3 Joint Optimization

Although using randomly chosen rays in the simulation of DoF effect is unpractical, it shows high efficiency in optimizing the geometric representation of 3D scenes. The patch-based method, however, often leads to divergence or sub-optimal results due to the gathering of rays. Thus, we resort to a two-stage optimization process to reduce the complexity of learning geometry representation, aperture size, and focus distance.

At the first stage, we train the NeRF network with the aperture parameter set to 0, which degenerates the rendering model to a linear pinhole camera. In this stage, the aperture parameters and focus distances are not optimized, and the classical ray-choosing and volume rendering are adopted. This stage aims to generate a coarse 3D representation where the rendered foreground and background may be blurred due to the shallow DoF inputs. At the second stage, we further optimize the NeRF network with the concentrate-and-scatter method using patch-based ray selection. The aperture parameters, focus distances, and the NeRF network parameters are jointly optimized. We summarize our learning algorithm in Algorithm 1.

## 5 RESULTS AND DISCUSSIONS

### 5.1 Dataset and Evaluation

We evaluate our method on both a real-world dataset and a synthetic dataset. The real-world dataset consists of 7 scenes, where each contains 20 ~ 30 image triplets. Each triplet includes an all-in-focus image taken with small aperture and two images taken with large aperture focusing on the foreground and background, respectively.

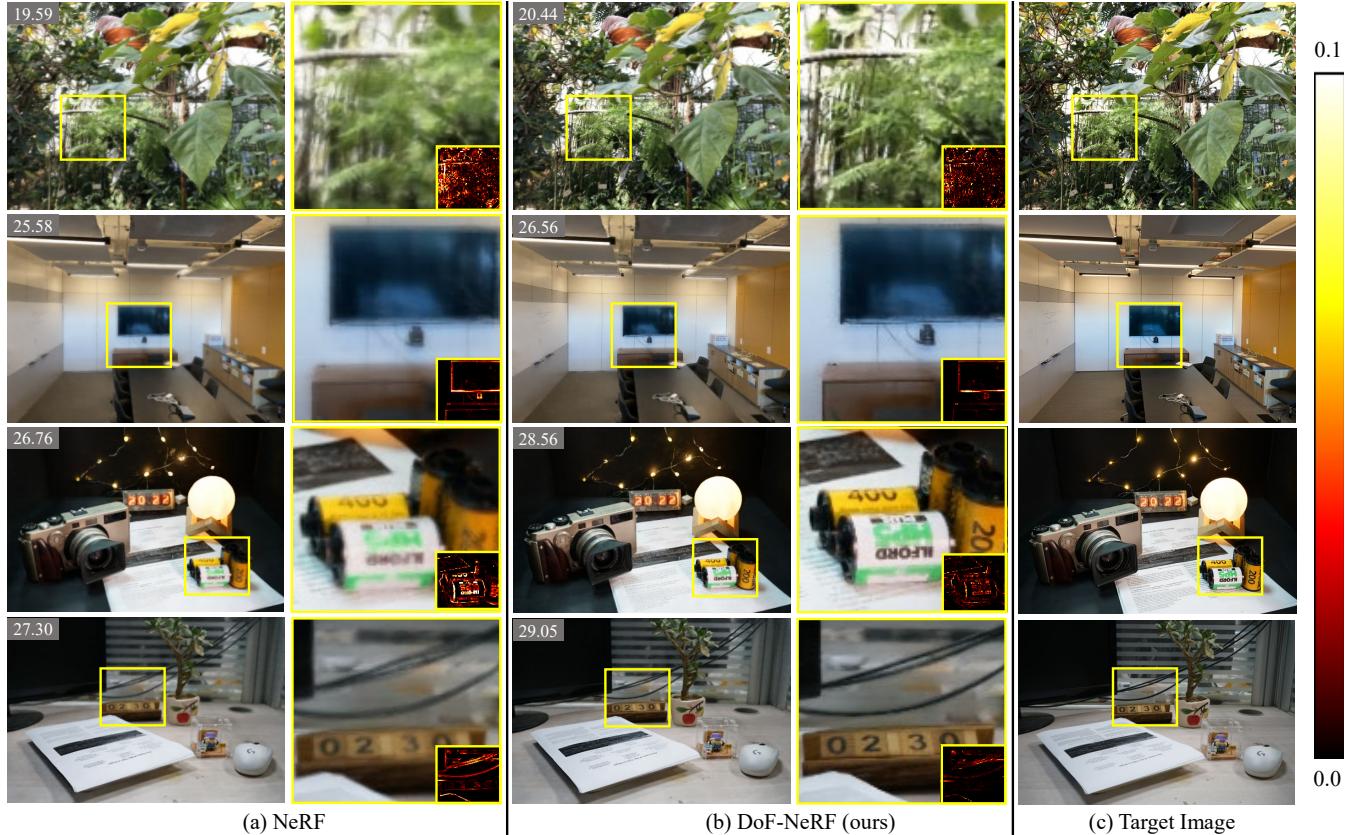


Figure 6: Comparison of NeRF and our approach in all-in-focus rendering with shallow DoF inputs. The first and third column shows the images rendered by NeRF and our approach, respectively. PSNR is shown at the upper left corner. We zoom in all the yellow boxes in the second and fourth column with error maps (0 to 0.1 pixel intensity range) shown at the lower right corner. The first two rows show the result of scenes from the synthetic dataset, and the third and fourth rows are scenes from the real-world dataset.

**Table 1: Comparison of NeRF [27] and our framework in the real-world dataset.**

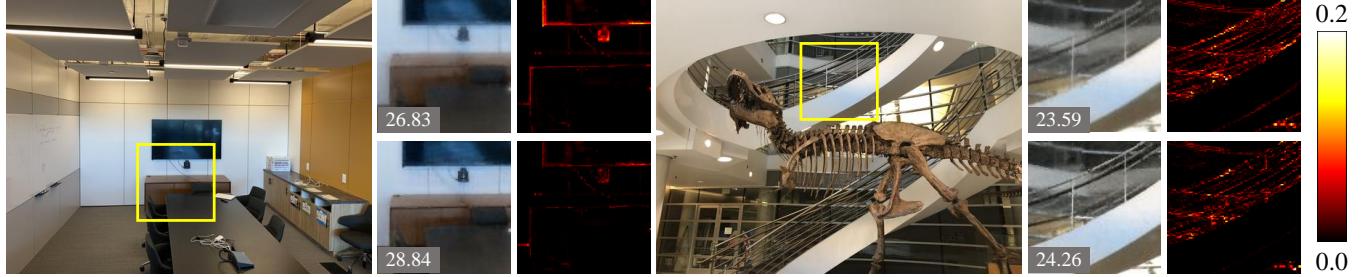
| Scene  | Model | PSNR( $\uparrow$ ) | SSIM( $\uparrow$ ) | LPIPS( $\downarrow$ ) |
|--------|-------|--------------------|--------------------|-----------------------|
| amiya  | NeRF  | 26.924             | 0.9092             | 0.1633                |
|        | ours  | <b>28.311</b>      | <b>0.9289</b>      | <b>0.1370</b>         |
| camera | NeRF  | 25.593             | 0.8862             | 0.1574                |
|        | ours  | <b>27.714</b>      | <b>0.9134</b>      | <b>0.1259</b>         |
| plant  | NeRF  | 28.272             | 0.8961             | 0.1581                |
|        | ours  | <b>30.317</b>      | <b>0.9290</b>      | <b>0.1178</b>         |
| turtle | NeRF  | 33.531             | 0.9566             | 0.0939                |
|        | ours  | <b>34.965</b>      | <b>0.9647</b>      | <b>0.0823</b>         |

**Table 2: Comparison of NeRF [27] and our framework in the synthetic dataset.**

| Scene    | Model | PSNR ( $\uparrow$ ) | SSIM( $\uparrow$ ) | LPIPS( $\downarrow$ ) |
|----------|-------|---------------------|--------------------|-----------------------|
| fortress | NeRF  | 28.142              | 0.7826             | 0.2011                |
|          | ours  | <b>29.168</b>       | <b>0.8099</b>      | <b>0.1830</b>         |
| leaves   | NeRF  | 19.450              | 0.6541             | 0.3190                |
|          | ours  | <b>20.025</b>       | <b>0.7000</b>      | <b>0.2766</b>         |
| room     | NeRF  | 26.668              | 0.8743             | 0.1961                |
|          | ours  | <b>29.443</b>       | <b>0.9135</b>      | <b>0.1502</b>         |
| trex     | NeRF  | 24.433              | 0.8379             | 0.1723                |
|          | ours  | <b>25.726</b>       | <b>0.8744</b>      | <b>0.1564</b>         |

We generate camera parameters by COLMAP [35] using all-in-focus images. The synthetic dataset is generated based on depth estimation and a recent single-image DoF rendering framework: for each image in the Real Forward-Facing dataset [27], we use DPT [32] to generate the disparity map and BokehMe [29] to render shallow DoF images. Details of the datasets can be found in the appendix.

Following [27], we adopt PSNR, SSIM [44] and LPIPS [50] as evaluation metrics. In all the experiments, all images are of  $497 \times 331$  resolution for the real-world dataset and  $504 \times 378$  for the synthetic dataset. 1/9 of the images are held out for testing. 50% foreground-focused images and 50% background-focused images make up the training images. Both our method and vanilla NeRF adopt the same mixing order of images in each scene.



**Figure 7: Comparison of DS-NeRF [4] and our approach in the synthetic dataset.** For each scene, the zoom-in of rendered images and error maps (0 to 0.2 pixel intensity range) are presented. They are respectively obtained from DS-NeRF (first row) and our model (second row). For each subfigure, PSNR is shown on the lower left.

## 5.2 Improvement over Vanilla NeRF

In this section, we compare DoF-NeRF qualitatively and quantitatively with vanilla NeRF on the real-world and synthetic datasets. We set the aperture parameters of the DoF-NeRF to zero and render all-in-focus novel views for evaluation. As shown in Table 1 and Table 2, one can observe that our method demonstrates better perceptual qualities than vanilla NeRF when rendering all-in-focus novel views on both two datasets. This implies that our explicit aperture modeling enables NeRF to tackle shallow DoF inputs and benefits all-in-focus novel view synthesis. We also provide the qualitative comparisons in Fig. 6. As can be seen, vanilla NeRF is prone to generate blurry renderings and miss some texture details. In contrast, our DoF-NeRF can synthesize photo-realistic all-in-focus novel views with fine-grained details.

## 5.3 Improvement over DS-NeRF

Since our DoF-NeRF is a modification of volume rendering, it can work on variants of NeRF as a plug-and-play module. To show this, we substitute the NeRF architecture with DS-NeRF [4] and use the same optimization technique. We then compare DS-NeRF and our model in the synthetic dataset. As shown in Table 3, the inclusion of our DoF module results in better rendering quality with shallow DoF images as inputs. The qualitative results are visualized in Fig. 7.

## 5.4 Comparison on All-in-Focus Inputs

Here we present the comparison on the all-in-focus dataset. To additionally validate the effectiveness of our method in the all-in-focus setting, we design an experiment where wide DoF images from the Real Forward-Facing dataset [27] are used as inputs and adopt the same initialization, training, and rendering settings as the experiments conducted on the shallow DoF data. Table 4 reports the quantitative comparison on the all-in-focus dataset. Although our framework is originally designed for shallow DoF inputs, experiments indicate that it shows comparable performance against vanilla NeRF using all-in-focus images as inputs.

## 5.5 DoF Rendering

Apart from representing all-in-focus 3D scenes with shallow DoF inputs, it is also possible to render the DoF effect with the optical modeling for volume rendering. By changing the aperture or focus settings, we can manipulate the DoF effect in novel view synthesis.

**Table 3: Comparison of DS-NeRF [4] and our framework in the synthetic dataset.**

| Scene    | Model   | PSNR(↑)       | SSIM(↑)       | LPIPS(↓)      |
|----------|---------|---------------|---------------|---------------|
| fortress | DS-NeRF | 28.493        | 0.7609        | 0.2373        |
|          | ours    | <b>29.704</b> | <b>0.8112</b> | <b>0.1970</b> |
| leaves   | DS-NeRF | 17.872        | 0.5096        | 0.4358        |
|          | ours    | <b>18.312</b> | <b>0.5689</b> | <b>0.3840</b> |
| room     | DS-NeRF | 26.8301       | 0.8641        | 0.2287        |
|          | ours    | <b>29.550</b> | <b>0.9045</b> | <b>0.1817</b> |
| trex     | DS-NeRF | 22.334        | 0.7063        | 0.3438        |
|          | ours    | <b>23.600</b> | <b>0.7764</b> | <b>0.2773</b> |

**Table 4: Comparison of NeRF [27] and our framework in the all-in-focus dataset.**

| Scene    | Model | PSNR(↑)       | SSIM(↑)       | LPIPS(↓)      |
|----------|-------|---------------|---------------|---------------|
| fortress | NeRF  | 33.973        | 0.9411        | 0.0469        |
|          | ours  | <b>34.007</b> | <b>0.9422</b> | <b>0.0454</b> |
| leaves   | NeRF  | <b>21.586</b> | 0.7854        | 0.1629        |
|          | ours  | 21.487        | <b>0.7861</b> | <b>0.1599</b> |
| room     | NeRF  | <b>33.852</b> | <b>0.9593</b> | <b>0.0721</b> |
|          | ours  | 33.760        | 0.9584        | 0.0735        |
| trex     | NeRF  | <b>30.287</b> | <b>0.9501</b> | <b>0.0601</b> |
|          | ours  | 30.222        | 0.9499        | 0.0605        |

In Fig. 8, we visualize the effect of aperture parameter and focusing distance by using different rendering settings.

With different lens designs and configurations various CoC styles can be created. Our method can easily simulate this phenomenon by changing the shape of blur kernel. For example, we use a deformable polygonal kernel to create various DoF effects in Fig. 9.

## 5.6 Parameter Analysis

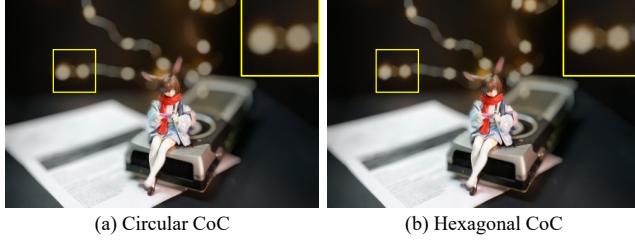
In this section, we construct image sets with mixing patterns to verify the validity of the estimated aperture parameters and focus distances, because accurate apertures and focus distances cannot be obtained in both the real-world dataset and synthetic dataset. Specifically, when validating the focus distance, we use foreground-focused images as even-indexed views, and background-focused



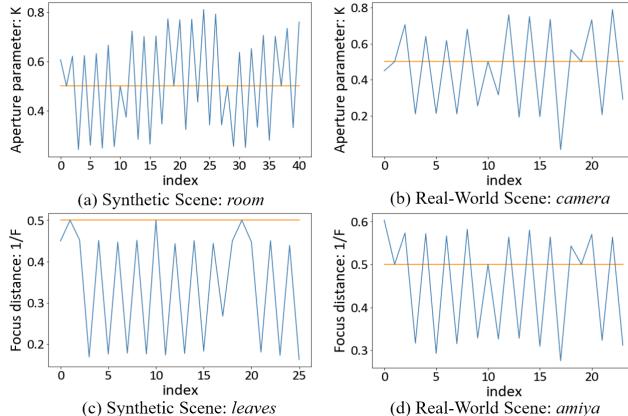
(a) All-in-Focus

(b)  $K = 8, F = 2.0$ (c)  $K = 15, F = 2.0$ (d)  $K = 15, F = 5.0$ 

**Figure 8: Visualization of adjustable DoF rendering.** Apart from (a) all-in-focus rendering, we can manipulate DoF effect by changing aperture and focus settings. We set  $F$  to 2.0 and  $K$  to (b) 8 and (c) 15 respectively to render images with various aperture parameters. By holding  $K$  at 15, we change  $F$  to (d) 5.0 to create images with different focus distance.



**Figure 9: Visualization of adjustable CoC shape rendering,** such as (a) circular and (b) hexagonal.



**Figure 10: Visualization of inferred aperture and focus parameters.** The orange lines represent the initialized values, and we scale the aperture parameters by 10. Some parameters remain unchanged because the corresponding views are held out for testing and not optimized during training.

images as odd-indexed views. When validating the aperture parameter, on odd-indexed views we use all-in-focus images, which can be seen as images taken using small aperture; on even-indexed views we use shallow DoF images, which can be seen as images taken using wide aperture. We visualize the value of apertures and focus distances of every image before and after optimization in Fig. 10.

**Table 5: Ablation studies of components of our model.** “Aperture” and “Focus” denote learnable aperture parameters and focus distances respectively.

| Scene  | Model | Aperture | Focus | PSNR( $\uparrow$ ) | SSIM( $\uparrow$ ) | LPIPS( $\downarrow$ ) |
|--------|-------|----------|-------|--------------------|--------------------|-----------------------|
| camera | NeRF  | —        | —     | 25.742             | 0.8723             | 0.1657                |
|        | ours  | ✓        | ✗     | 26.639             | 0.8989             | 0.1338                |
|        | ours  | ✗        | ✓     | 25.855             | 0.8786             | 0.1532                |
|        | ours  | ✓        | ✓     | <b>26.962</b>      | <b>0.9045</b>      | <b>0.1280</b>         |

The results show that the prediction of both aperture parameters and focus distances are optimized to the similar mixing pattern of the training set, which demonstrates that the estimated aperture parameters and focus distances are reasonable and valid.

To validate the design choice of our approach, we also conduct an ablation study on the scene *camera*. From the results of different combinations in Table 5, one can observe that, i) explicit modeling of the aperture parameters and the focus distance benefits novel view synthesis, and ii) the aperture parameter and the focus distance can promote each other when they are jointly optimized.

## 6 CONCLUSION

In this work, we present DoF-NeRF, a novel framework for recovering sharp 3D scenes from sparse shallow DoF images. To achieve this, we model the CoC to simulate the radiance scattered between rays and introduce learnable parameters to enable direct manipulation of the DoF effect. Comprehensive experiments are conducted on both synthetic and real-world datasets, where DoF-NeRF not only performs comparably with NeRF in the all-in-focus setting, but also can synthesize all-in-focus novel views conditioned on shallow DoF inputs. Moreover, by changing the aperture parameter or the focus distance, DoF-NeRF can achieve controllable DoF rendering from novel viewpoints.

## ACKNOWLEDGMENTS

This work was funded by the DigiX Joint Innovation Center of Huawei-HUST.

## REFERENCES

- [1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. 2018. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*. PMLR, 40–49.
- [2] Benjamin Busam, Matthieu Hog, Steven McDonagh, and Gregory Slabaugh. 2019. StereoF: Efficient image refocusing with stereo vision. In *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*. 0–0.
- [3] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. 1996. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 11–20.
- [4] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. 2022. Depth-Supervised NeRF: Fewer Views and Faster Training for Free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 12882–12891.
- [5] Saikat Dutta, Sourya Dipta Das, Nisarg A Shah, and Anil Kumar Tiwari. 2021. Stacked deep multi-scale hierarchical network for fast bokeh effect rendering from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2398–2407.
- [6] Haoqiang Fan, Hao Su, and Leonidas J Guibas. 2017. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 605–613.
- [7] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. 2019. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2367–2376.
- [8] Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. 2021. FastNeRF: High-Fidelity Neural Rendering at 200FPS. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 14346–14355.
- [9] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. 1996. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 43–54.
- [10] E. Hecht. 2017. *Optics*. Pearson Education, Incorporated.
- [11] Xin Huang, Qi Zhang, Ying Feng, Hongdong Li, Xuan Wang, and Qing Wang. 2022. HDR-NeRF: High Dynamic Range Neural Radiance Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 18398–18408.
- [12] Andrey Ignatov, Jagruti Patel, and Radu Timofte. 2020. Rendering natural camera bokeh effect with deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 418–419.
- [13] Andrey Ignatov, Jagruti Patel, Radu Timofte, Bolun Zheng, Xin Ye, Li Huang, Xiang Tian, Saikat Dutta, Kuldeep Purohit, Praveen Kandula, et al. 2019. Aim 2019 challenge on bokeh effect synthesis: Methods and results. In *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*. IEEE, 3591–3598.
- [14] Andrey Ignatov, Radu Timofte, Ming Qian, Congyu Qiao, Jiamin Lin, Zhenyu Guo, Chenghua Li, Cong Leng, Jian Cheng, Juewen Peng, et al. 2020. Aim 2020 challenge on rendering realistic bokeh. In *Proceedings of the European Conference on Computer Vision Workshops (ECCVW)*. Springer, 213–228.
- [15] Yoonwoo Jeong, Seokjun Ahn, Christopher Choy, Anima Anandkumar, Minsu Cho, and Jaesik Park. 2021. Self-Calibrating Neural Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 5846–5854.
- [16] Danilo Jimenez Rezende, SM Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. 2016. Unsupervised learning of 3d structure from images. *Advances in neural information processing systems* 29 (2016), 4996–5004.
- [17] James T Kajiya and Brian P Von Herzen. 1984. Ray tracing volume densities. *ACM SIGGRAPH computer graphics* 18, 3 (1984), 165–174.
- [18] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. 2018. Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 371–386.
- [19] Marc Levoy and Pat Hanrahan. 1996. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 31–42.
- [20] L. Li, Y. Huang, and Y. Wang. 2005. *Applied Optics*. Beijing Institute of Technology Press.
- [21] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. 2021. Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 6498–6508.
- [22] Yiyi Liao, Simon Donne, and Andreas Geiger. 2018. Deep marching cubes: Learning explicit surface representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2916–2925.
- [23] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh. 2019. Neural Volumes: Learning Dynamic Renderable Volumes from Images. *ACM Transactions on Graphics* 38, 4CD (2019), 65.1–65.14.
- [24] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. 2021. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 7210–7219.
- [25] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul P. Srinivasan, and Jonathan T. Barron. 2022. NeRF in the Dark: High Dynamic Range View Synthesis From Noisy Raw Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 16190–16199.
- [26] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. 2019. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–14.
- [27] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2020. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 405–421.
- [28] Michael Niemeyer and Andreas Geiger. 2021. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 11453–11464.
- [29] Juewen Peng, Zhiguo Cao, Xianrui Luo, Hao Lu, Ke Xian, and Jianming Zhang. 2022. BokehMe: When neural rendering meets classical rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 16283–16292.
- [30] Juewen Peng, Xianrui Luo, Ke Xian, and Zhiguo Cao. 2021. Interactive portrait bokeh rendering system. In *Proc. IEEE International Conference on Image Processing (ICIP)*. IEEE, 2923–2927.
- [31] Ming Qian, Congyu Qiao, Jiamin Lin, Zhenyu Guo, Chenghua Li, Cong Leng, and Jian Cheng. 2020. Bggan: Bokeh-glass generative adversarial network for rendering realistic bokeh. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 229–244.
- [32] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. 2021. Vision Transformers for Dense Prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 12179–12188.
- [33] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. 2018. Generating 3D faces using convolutional mesh autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 704–720.
- [34] Christian Reiser, Songyu Peng, Yiyi Liao, and Andreas Geiger. 2021. KiloNeRF: Speeding Up Neural Radiance Fields With Thousands of Tiny MLPs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 14335–14345.
- [35] Johannes L Schonberger and Jan-Michael Frahm. 2016. Structure-from-motion revisited. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4104–4113.
- [36] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. 2020. Graf: Generative radiance fields for 3D-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 33.
- [37] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. 2019. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2437–2446.
- [38] Vincent Sitzmann, Michael Zollhofer, and Gordon Wetzstein. 2019. Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. <https://proceedings.neurips.cc/paper/2019/file/b5dc4e5d9b495d0196f61d45b26ef33e-Paper.pdf>
- [39] Cheng Sun, Min Sun, and Hwann-Tzong Chen. 2022. Direct Voxel Grid Optimization: Super-Fast Convergence for Radiance Fields Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5459–5469.
- [40] Richard Tucker and Noah Snavely. 2020. Single-view view synthesis with multi-plane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 551–560.
- [41] Neal Wadhwa, Rahul Garg, David E Jacobs, Bryan E Feldman, Nori Kanazawa, Robert Carroll, Yair Movshovitz-Attias, Jonathan T Barron, Yael Pritch, and Marc Levoy. 2018. Synthetic depth-of-field with a single-camera mobile phone. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–13.
- [42] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. 2018. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 52–67.
- [43] Yinhua Wang, Shuzhou Yang, Yujie Hu, and Jian Zhang. 2022. NeRFocus: Neural Radiance Field for 3D Synthetic Defocus. *arXiv preprint arXiv:2203.05189* (2022).
- [44] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.
- [45] Ke Xian, Juewen Peng, Chao Zhang, Hao Lu, and Zhiguo Cao. 2021. Ranking-based salient object detection and depth prediction for shallow depth-of-field. *Sensors* 21, 5 (2021), 1815.

- [46] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. 2021. Space-time Neural Irradiance Fields for Free-Viewpoint Video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 9421–9431.
- [47] Lei Xiao, Anton Kaplanyan, Alexander Fix, Matthew Chapman, and Douglas Lanman. 2018. Deepfocus: Learned image synthesis for computational displays. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–13.
- [48] Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, and Shengping Zhang. 2019. Pix2vox: Context-aware 3d reconstruction from single and multi-view images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2690–2698.
- [49] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. 2020. NeRF++: Analyzing and Improving Neural Radiance Fields. *arXiv:2010.07492* (2020).
- [50] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 586–595.
- [51] Xuaner Zhang, Kevin Matzen, Vivien Nguyen, Dillon Yao, You Zhang, and Ren Ng. 2019. Synthetic defocus and look-ahead autofocus for casual videography. *ACM Transactions on Graphics (TOG)* 38 (2019), 1–16.
- [52] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. 2018. Stereo magnification: learning view synthesis using multiplane images. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

## A APPENDIX SUMMARY

The appendix involves the following contents:

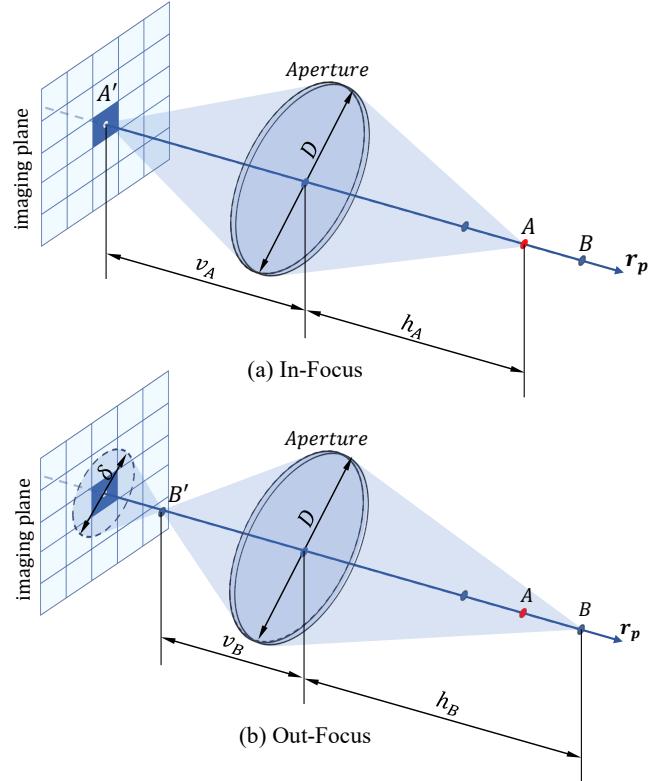
- Table of notations (Table 6).
- Physical model of DoF.
- Algorithm details of the concentrate-and-scatter method.
- Implementation details.
- Datasets details.
- Full experimental reports and more qualitative results.
- More synthesis of DoF effect.

**Table 6: Table of notations**

| Notation               | Description   |
|------------------------|---|
| $N$                    | Number of the images in the training set              |
| $N_s$                  | Number of the sample points on a single ray           |
| $N_{patch}$            | Size of the ray patches                               |
| $N_{anchor}$           | Distance between adjacent anchors                     |
| $N_{iters}$            | Number of iterations for joint optimization           |
| $N_{pretrain}$         | Number of iterations for the first-stage optimization |
| $x$                    | Spatial location                                      |
| $d$                    | Viewing direction                                     |
| $o$                    | Camera origin   |
| $p$                    | Pixel on the imaging plane                            |
| $r_p$                  | Ray specific to pixel $p$                             |
| $c$                    | Radiance predicted by MLP                             |
| $\alpha$               | Transparency predicted by MLP                         |
| $f$                    | Focal length  |
| $D$                    | Aperture diameter                                     |
| $K$                    | Aperture parameter                                    |
| $F$                    | Focus distance  |
| $h$                    | Depth of a spatial point along a ray                  |
| $\Delta$               | Distance between adjacent sample points               |
| $h_c$                  | Concentrated depth                                    |
| $\delta$               | Diameter of CoC                                       |
| $\delta_c$             | Diameter of CoC at the concentrated depth             |
| $I$                    | Set of input images                                   |
| $\mathcal{P}$          | Set of pixels   |
| $\mathcal{K}$          | Set of aperture parameters                            |
| $\mathcal{F}$          | Set of focus distances                                |
| $G_{\Theta}, \Theta$   | MLP network of NeRF and its parameters                |
| $\mathcal{L}$          | Loss  |
| $I$                    | Input image   |
| $T$                    | Accumulated transmittance                             |
| $\hat{I}(p)$           | Predicted color of a pixel $p$                        |
| $\hat{I}_{ray}(p)$     | Diffused radiance of a pixel $p$                      |
| $\hat{I}_{scatter}(p)$ | Scattering radiance from other rays                   |
| $\hat{C}$              | Scattering radiance of a spatial point                |
| $\hat{C}_s(p)$         | Diffused radiance of a pixel $p$                      |
| $K_{volume}$           | Volume rendering coefficient                          |

## B OPTICS PRINCIPLE OF DOF

In this section, we provide a detailed derivation of CoC diameters. Given an ideal optical lens system with focal length  $f$  and aperture



**Figure 11: Principle of CoC and DoF.**

diameter  $D$ , lights emitted from spatial points  $A$  and  $B$  on the ray  $r_p$  with object distance  $h_A$  and  $h_B$  converge on points  $A'$  and  $B'$  with imaging distance  $v_A$  and  $v_B$ , respectively (Fig. 11). The relation between  $h_A$  and  $v_A$  (similarly,  $h_B$  and  $v_B$ ) follows the Gaussian formula

$$\frac{1}{h_A} + \frac{1}{v_A} = \frac{1}{f}, \quad (15)$$

$$\frac{1}{h_B} + \frac{1}{v_B} = \frac{1}{f}. \quad (16)$$

As shown in Fig 11, point  $A$  is imaged to the converging point  $A'$  on the imaging plane. In contrast, point  $B$  is projected to a CoC with diameter  $\delta_B$  on the imaging plane. According to the properties of similar triangles,  $\delta_B$  can be computed by

$$\frac{\delta_B}{D} = \frac{v_A - v_B}{v_B}. \quad (17)$$

Substituting Eq.(15) and Eq.(16) into Eq.(17) takes the form

$$\delta_B = fD \times \frac{h_B - h_A}{h_B(h_A - f)}. \quad (18)$$

As the lens system focuses on point  $A$ , we can replace  $h_A$  with focus distance  $F$ :

$$\delta_B = fD \times \frac{h_B - F}{h_B(F - f)}. \quad (19)$$

For an out-focus point  $C$  with object distance  $h_C$  smaller than focus distance  $F$ , the diameter of CoC  $\delta_C$  is formed as

$$\delta_C = fD \times \frac{F - h_C}{h_C(F - f)}. \quad (20)$$

Thus the diameter of CoC can be formulated by

$$\delta = fD \times \frac{|F - h_t|}{h_t(F - f)}, \quad (21)$$

where  $h_t$  denotes the object distance of a spacial point. Since the focus distance  $F$  and object distance  $h_t$  are often much larger than the focal length  $f$ , we can modify Eq. (21) such that

$$\delta(h_t) = fD \times \frac{|h_t - F|}{Fh_t} = fD \times \left| \frac{1}{F} - \frac{1}{h_t} \right|. \quad (22)$$

## C ALGORITHM DETAILS

Here we present the implementation details of the Concentrate-and-Scatter algorithm.

As mentioned in the main paper, the concentration of the radiance follows the original volume rendering. We implement the radiance scattering method by a pixel-wise rendering algorithm (see Algorithm 2). Assuming that the aperture shape is circular, we first compute the signed defocus map  $S$  with concentrated depth  $H_c$ , aperture parameter  $K$  and focus distance  $F$ . We apply a gamma transformation to transform radiance  $C$  to linear space. Two accumulation buffers  $W$  and  $I$  are initialized with zero. Function *TraversePatch()* is adopted to traverse all pixels of  $C$ . The scattering radius  $r_i$  can be calculated by the absolute value of  $S_i$ . We then traverse the neighboring pixels of  $p_i$  by the function *TraverseNeighbor()*. We calculate the weight  $w_{ij}$  from  $p_i$  to its neighboring pixel  $p_j$ . For pixel  $p_i$ , its radiance can only scatter to  $p_j$  if scattering radius  $r_i$  is larger than the distance  $l_{ij}$  between  $p_i$  and  $p_j$ .

To produce smooth and natural DoF effect, a soft CoC kernel in the calculation of weight  $w_{ij}$  is adopted. We additionally divide  $w_{ij}$  by the square of  $r_i$  due to the uniform distribution of radiance. The calculated  $w_{ij}$  and radiance  $C_i$  weighted by  $w_{ij}$  are then accumulated in  $W_j$  and  $I_j$ , respectively. After traversing all pixels, the rendering result  $B_{cr}$  can be obtained by the element-wise division of  $I$  and  $W$ . A inverse gamma transformation is applied subsequently.

To create polygonal CoC, we can modify row 8 by multiplying a factor  $k_{ij}$  to  $r_i$ . The factor  $k_{ij}$  is defined by

$$k_{ij} = \frac{\sin\left(\frac{\pi}{2} - \frac{\pi}{n}\right)}{\sin\left(\frac{\pi}{2} - \frac{\pi}{n} + \text{mod}\left(\left|\arctan\left(\frac{l_{ij}^y}{l_{ij}^x}\right) + \phi\right|, \frac{2\pi}{n}\right)\right)}, \quad (23)$$

where  $n$  denotes the number of aperture blades and  $\phi$  represents the rotation angle of polygonal CoC.  $l_{ij}^x$  and  $l_{ij}^y$  denote the horizontal and vertical component of distance  $l_{ij}$ .

## D IMPLEMENTATION DETAILS

We use a batch size of 1024 rays for NeRF and DS-NeRF. The learning rate of NeRF and DS-NeRF is set to 0.0005 and decays exponentially to one-tenth for every 250000 steps. For NeRF and DS-NeRF, we use 64 samples for the coarse network and 64 samples for the fine network. As explained in the main paper, we adopt a two-stage optimization: each stage takes 200k iterations, where  $N_{\text{pretrained}}$  and  $N_{\text{iters}}$  are set to 200k and 400k, respectively. For all-in-focus inputs, we change  $N_{\text{pretrained}}$  and  $N_{\text{iters}}$  to 400k and 800k, respectively. At the second optimization stage, we set  $N_{\text{patch}}$  to 48 and  $N_{\text{anchor}}$  to 16. The aperture and focus parameters are both initialized with 0.5 for DoF-NeRF. For ablation study, the aperture and

---

### Algorithm 2: Pixel-wise scattering method

---

```

Input: Concentrated radiance  $C$ , concentrated depth  $H_c$ , aperture
parameter  $K$ , focus distance  $F$ , gamma value  $\gamma$ 
Output: Scattering result  $B_{cr}$ 
1  $S \leftarrow K \cdot \left( \frac{1}{H_c} - \frac{1}{F} \right);$ 
2  $C \leftarrow (C)^\gamma;$ 
3  $W \leftarrow [0];$ 
4  $I \leftarrow [0];$ 
5 for  $p_i \leftarrow \text{TraversePatch}(C)$  do
6    $r_i \leftarrow |S_i|;$ 
7   for  $p_j \leftarrow \text{TraverseNeighbor}(p_i, r_i)$  do
8      $w_{ij} \leftarrow \frac{0.5+0.5 \tanh\left(4(r_i-l_{ij})\right)}{r_i^2+0.2};$ 
9      $W_j \leftarrow W_j + w_{ij};$ 
10     $I_j \leftarrow I_j + w_{ij} \cdot C_i;$ 
11   end
12 end
13  $B_{cr} \leftarrow \frac{I}{W};$ 
14  $B_{cr} \leftarrow (B_{cr})^{\frac{1}{\gamma}};$ 

```

---

focus parameters are set to 0.5 for initialization and fixed according to the experiment settings.

## E DATASET DETAILS

### E.1 Real-World Dataset

The real-world dataset consists of 7 scenes: *amiya*, *camera*, *plant*, *kendo*, *desk*, *shelf*, and *turtle*. Each scene contains 20 ~ 30 image triplets. Each triplet includes a wide DoF image taken with small aperture and two images taken with large aperture focusing on the foreground and background respectively (Fig. 12).

All images are taken by a Sony ILCE-7RM2 camera with an FE 35mm f/1.8 lens. We secure the camera to a tripod and use remote control when taking images, in order to preserve identical camera parameters. Instead of using the maximum aperture to create shallow DoF, we use the aperture of  $f/4$  to avoid distinct lens distortion and vignette. The camera parameters are generated by COLMAP [35]. As an example, we visualize the predicted camera poses and sparse point cloud of the scene *kendo* in Fig. 13. The original resolution of the images in the real-world dataset is  $3976 \times 2652$ . We set the resolution to  $497 \times 331$  for training and evaluation.

### E.2 Synthetic Dataset

The synthetic dataset is generated from the Real Forward-Facing dataset [27] which consists of 8 scenes: *fern*, *flower*, *fortress*, *horns*, *leaves*, *orchids*, *room*, and *trex*. Since the images are captured with a handheld cellphone, the original images from the Real Forward-Facing dataset are used as wide DoF images in the triplets. The shallow DoF images are generated based on depth estimation and a recent single-image bokeh rendering framework. For each wide DoF image, we use DPT [32] to generate the disparity map and BokehMe [29] to synthesize the shallow DoF images. The blur parameter of BokehMe is set to 20 and the focus distances are set to 0.1 and 0.9 to simulate background focused and foreground focused shallow DoF images (Fig. 14). The original resolution of the images

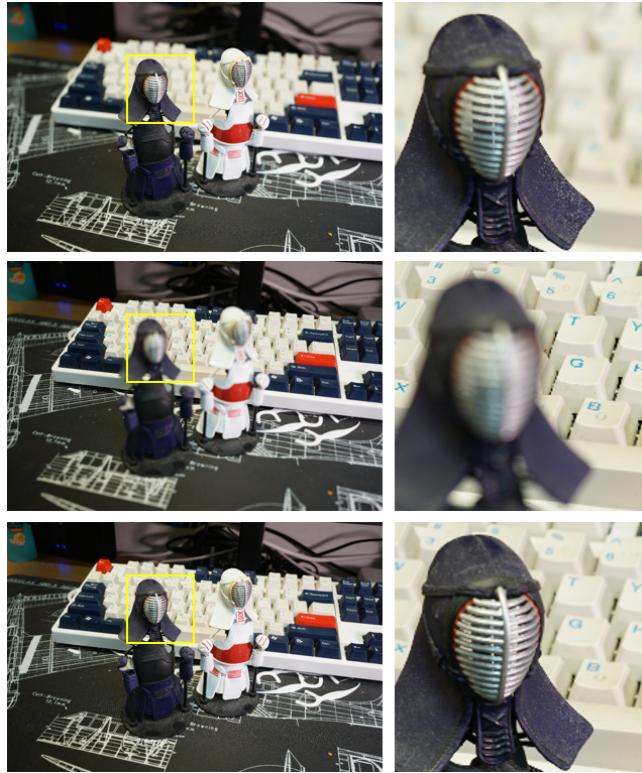


Figure 12: Visualization of the scene *kendo* in the real-world dataset. Each row respectively represents the foreground-focusing shallow DoF image, background-focusing shallow DoF image, and wide DoF image. We zoom in the yellow boxes in the images and present them on the right.

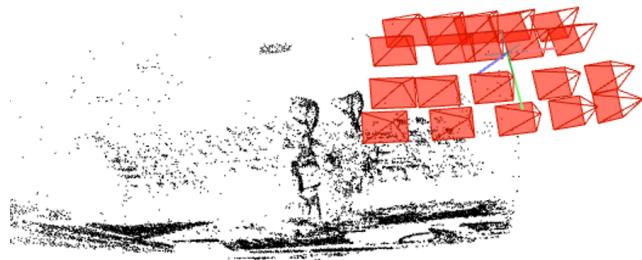


Figure 13: Visualization of the predicted camera poses and reconstructed sparse point cloud by COLMAP [35].

in the synthetic dataset is  $4032 \times 3024$ . We set the resolution to  $504 \times 378$  for training and evaluation.

## F EXPERIMENTAL RESULTS

In this section, we present full reports of experiments conducted in main paper and more qualitative comparisons.

We extend Table 2 and Table 1 in the main paper by conducting experiments in other scenes of the synthetic dataset and real-world dataset. Table 7 and Table 8 report the rendering quality of vanilla



Figure 14: Visualization of the scene *leaves* in the synthetic dataset. The first row shows the original image from the Real Forward-Facing dataset [27] and the predicted disparity map from DPT [32]. The second and third rows are the shallow DoF images rendered by BokehMe [29]. We zoom in the yellow boxes in the images and present them on the right.

Table 7: Comparison of NeRF [27] and our framework in the real-world dataset.

| Scene  | Model | PSNR( $\uparrow$ ) | SSIM( $\uparrow$ ) | LPIPS( $\downarrow$ ) |
|--------|-------|--------------------|--------------------|-----------------------|
| amiya  | NeRF  | 26.924             | 0.9092             | 0.1633                |
|        | ours  | <b>28.311</b>      | <b>0.9289</b>      | <b>0.1370</b>         |
| camera | NeRF  | 25.593             | 0.8862             | 0.1574                |
|        | ours  | <b>27.714</b>      | <b>0.9134</b>      | <b>0.1259</b>         |
| plant  | NeRF  | 28.272             | 0.8961             | 0.1581                |
|        | ours  | <b>30.317</b>      | <b>0.9290</b>      | <b>0.1178</b>         |
| turtle | NeRF  | 33.531             | 0.9566             | 0.0939                |
|        | ours  | <b>34.965</b>      | <b>0.9647</b>      | <b>0.0823</b>         |
| kendo  | NeRF  | 18.457             | 0.6643             | 0.2906                |
|        | ours  | <b>19.311</b>      | <b>0.6970</b>      | <b>0.2773</b>         |
| desk   | NeRF  | 29.618             | 0.9374             | 0.1099                |
|        | ours  | <b>30.548</b>      | <b>0.9426</b>      | <b>0.1047</b>         |
| shelf  | NeRF  | 31.552             | 0.9515             | 0.0811                |
|        | ours  | <b>32.002</b>      | <b>0.9565</b>      | <b>0.0723</b>         |

NeRF and our model in the real-world dataset and synthetic dataset,

**Table 8: Comparison of NeRF [27] and our framework in the synthetic dataset.**

| Scene    | Model | PSNR ( $\uparrow$ ) | SSIM( $\uparrow$ ) | LPIPS( $\downarrow$ ) |
|----------|-------|---------------------|--------------------|-----------------------|
| fortress | NeRF  | 28.142              | 0.7826             | 0.2011                |
|          | ours  | <b>29.168</b>       | <b>0.8099</b>      | <b>0.1830</b>         |
| leaves   | NeRF  | 19.450              | 0.6541             | 0.3190                |
|          | ours  | <b>20.025</b>       | <b>0.7000</b>      | <b>0.2766</b>         |
| room     | NeRF  | 26.668              | 0.8743             | 0.1961                |
|          | ours  | <b>29.443</b>       | <b>0.9135</b>      | <b>0.1502</b>         |
| trex     | NeRF  | 24.433              | 0.8379             | 0.1723                |
|          | ours  | <b>25.726</b>       | <b>0.8744</b>      | <b>0.1564</b>         |
| fern     | NeRF  | 23.202              | 0.7232             | 0.3125                |
|          | ours  | <b>23.407</b>       | <b>0.7311</b>      | <b>0.3055</b>         |
| flower   | NeRF  | 26.339              | 0.8209             | 0.1999                |
|          | ours  | <b>27.075</b>       | <b>0.8474</b>      | <b>0.1678</b>         |
| horns    | NeRF  | 24.260              | 0.7554             | 0.2983                |
|          | ours  | <b>24.829</b>       | <b>0.7766</b>      | <b>0.2759</b>         |
| orchids  | NeRF  | 19.755              | 0.6401             | 0.2932                |
|          | ours  | <b>20.048</b>       | <b>0.6678</b>      | <b>0.2718</b>         |

**Table 9: Comparison of DSNeRF [4] and our framework in the synthetic dataset.**

| Scene    | Model  | PSNR( $\uparrow$ ) | SSIM( $\uparrow$ ) | LPIPS( $\downarrow$ ) |
|----------|--------|--------------------|--------------------|-----------------------|
| fortress | DSNeRF | 28.493             | 0.7609             | 0.2373                |
|          | ours   | <b>29.704</b>      | <b>0.8112</b>      | <b>0.1970</b>         |
| leaves   | DSNeRF | 17.872             | 0.5096             | 0.4358                |
|          | ours   | <b>18.312</b>      | <b>0.5689</b>      | <b>0.3840</b>         |
| room     | DSNeRF | 26.8301            | 0.8641             | 0.2287                |
|          | ours   | <b>29.550</b>      | <b>0.9045</b>      | <b>0.1817</b>         |
| trex     | DSNeRF | 22.334             | 0.7063             | 0.3438                |
|          | ours   | <b>23.600</b>      | <b>0.7764</b>      | <b>0.2773</b>         |
| fern     | NeRF   | 23.100             | 0.7050             | 0.3355                |
|          | ours   | <b>23.640</b>      | <b>0.7307</b>      | <b>0.3116</b>         |
| flower   | NeRF   | 25.388             | 0.7683             | 0.2517                |
|          | ours   | <b>26.392</b>      | <b>0.8131</b>      | <b>0.2024</b>         |
| horns    | NeRF   | 21.562             | 0.5938             | 0.4625                |
|          | ours   | <b>22.487</b>      | <b>0.6491</b>      | <b>0.4111</b>         |
| orchids  | NeRF   | 19.435             | 0.6017             | 0.3247                |
|          | ours   | <b>19.872</b>      | <b>0.6500</b>      | <b>0.2876</b>         |

respectively. Fig. 17 and Fig. 18 respectively show additional qualitative results using the real-world and synthetic shallow DoF inputs. Given shallow DoF inputs, our method demonstrates better perceptual qualities than vanilla NeRF when rendering all-in-focus novel views on both the real-world dataset and synthetic dataset.

Table 9 shows the rendering quality of our method using DSNeRF as baseline on all scenes from the synthetic dataset. Our method can work as a plug-and-play module to improve the performance of NeRF-based method with shallow DoF inputs. Qualitative results are visualized in Fig. 16.

**Table 10: Comparison of NeRF [27] and our framework in the all-in-focus dataset.**

| Scene    | Model | PSNR( $\uparrow$ ) | SSIM( $\uparrow$ ) | LPIPS( $\downarrow$ ) |
|----------|-------|--------------------|--------------------|-----------------------|
| fortress | NeRF  | 33.973             | 0.9411             | 0.0469                |
|          | ours  | <b>34.007</b>      | <b>0.9422</b>      | <b>0.0454</b>         |
| leaves   | NeRF  | <b>21.5860</b>     | 0.7854             | 0.1629                |
|          | ours  | 21.4869            | <b>0.7861</b>      | <b>0.1599</b>         |
| room     | NeRF  | <b>33.852</b>      | <b>0.9593</b>      | <b>0.0721</b>         |
|          | ours  | 33.760             | 0.9584             | 0.0735                |
| trex     | NeRF  | <b>30.287</b>      | <b>0.9501</b>      | <b>0.0601</b>         |
|          | ours  | 30.222             | 0.9499             | 0.0605                |
| fern     | NeRF  | <b>26.582</b>      | 0.8490             | 0.1480                |
|          | ours  | 26.565             | <b>0.8497</b>      | <b>0.1470</b>         |
| flower   | NeRF  | <b>30.135</b>      | 0.9221             | 0.0643                |
|          | ours  | 30.093             | <b>0.9224</b>      | <b>0.0626</b>         |
| horns    | NeRF  | <b>28.205</b>      | 0.8879             | 0.1332                |
|          | ours  | 28.165             | <b>0.8888</b>      | <b>0.1313</b>         |
| orchids  | NeRF  | <b>20.831</b>      | 0.7362             | 0.1896                |
|          | ours  | 20.795             | <b>0.7374</b>      | <b>0.1876</b>         |

**Table 11: Ablation studies of components of our model. “Aperture” and “Focus” denote learnable aperture parameters and focus distances respectively.**

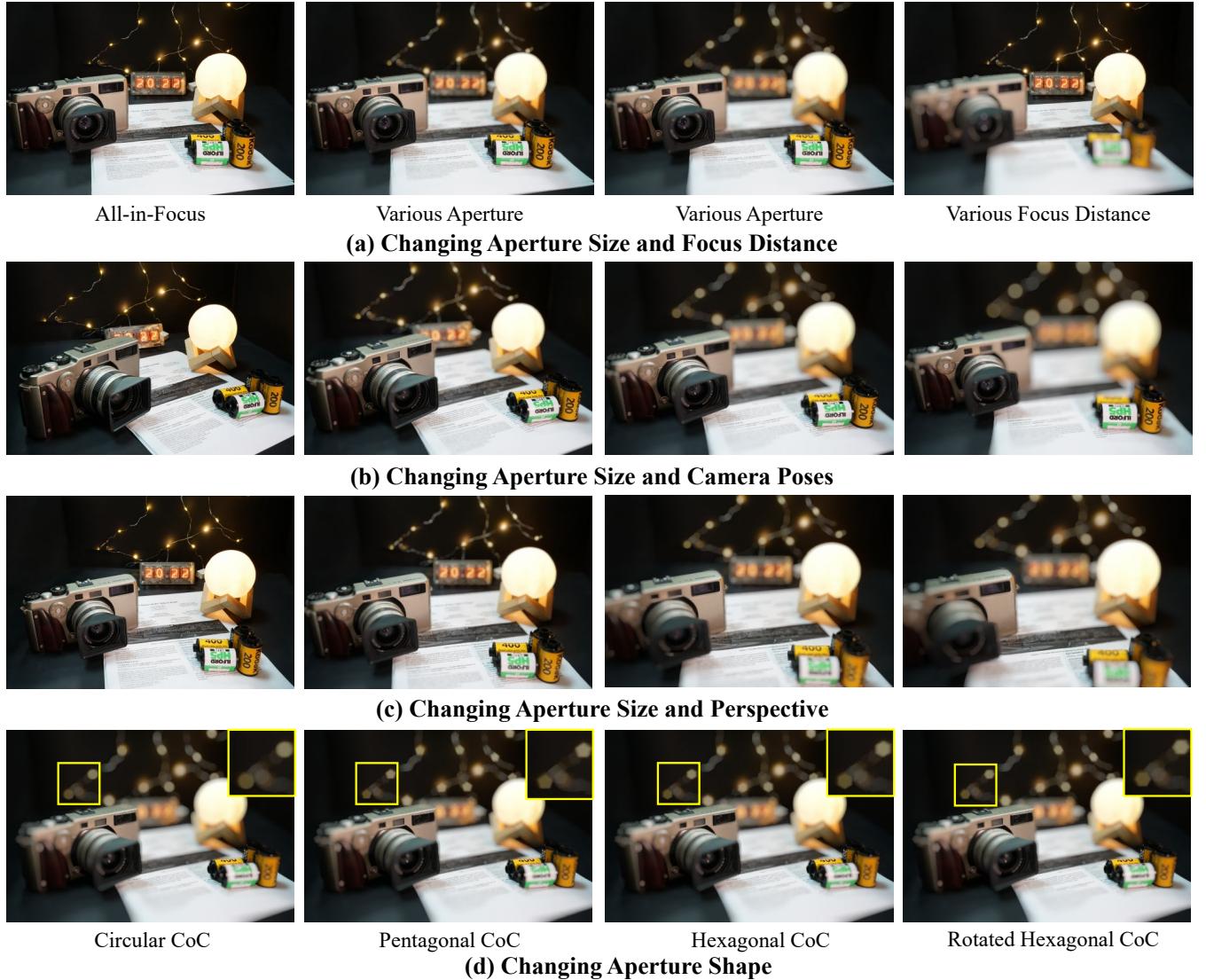
| Scene  | Model | Aperture | Focus | PSNR( $\uparrow$ ) | SSIM( $\uparrow$ ) | LPIPS( $\downarrow$ ) |
|--------|-------|----------|-------|--------------------|--------------------|-----------------------|
| room   | NeRF  | —        | —     | 27.279             | 0.8863             | 0.1803                |
|        | ours  | ✓        | ✗     | 27.799             | 0.8992             | 0.1640                |
|        | ours  | ✗        | ✓     | 28.070             | 0.9032             | 0.1671                |
|        | ours  | ✓        | ✓     | <b>28.361</b>      | <b>0.9076</b>      | <b>0.1556</b>         |
| camera | NeRF  | —        | —     | 25.742             | 0.8723             | 0.1657                |
|        | ours  | ✓        | ✗     | 26.639             | 0.8989             | 0.1338                |
|        | ours  | ✗        | ✓     | 25.855             | 0.8786             | 0.1532                |
|        | ours  | ✓        | ✓     | <b>26.962</b>      | <b>0.9045</b>      | <b>0.1280</b>         |

Although our framework is originally designed for shallow DoF inputs, Table 10 indicates that it shows comparable performance against vanilla NeRF using all-in-focus images as inputs.

Table 11 shows the results of the ablation study conducted on scene *room* from the synthetic dataset and on the scene *camera* from the real-world dataset.

## G DOF RENDERING

In this section, we present more synthesis of controllable DoF rendering (Fig. 15). Benefiting from the explicit modeling of DoF effect, DoF-NeRF enables direct manipulation of DoF effect by adjusting virtual aperture size and focus distance. Compared with single-image DoF rendering method, DoF-NeRF is capable of rendering DoF effect from novel viewpoints by adjusting the corresponding camera parameters. With different lens designs and configurations such as the shape of aperture, various CoC styles can be created.



**Figure 15: More synthesis of controllable DoF effect.** DoF-NeRF can manipulate the DoF rendering by changing (a) aperture size and focus distance, (b) camera poses, (c) perspective, and (d) aperture shape.

Our concentrate-and-scatter method can easily simulate this phenomenon by changing the shape of blur kernel. For example, we use a deformable polygonal kernel to create various DoF effects.

## H COST OF TRAINING AND TESTING

We compare the training and testing time of DoF-NeRF and vanilla NeRF on the same machine using a single NVIDIA RTX 3090 GPU. The following report are measured on the Scene Camera from the real-world dataset.

### H.1 Training

We train each model for a total of 400k iterations, *i.e.*, 400k iterations for vanilla NeRF, 200k iterations each for the first and second stage of DoF-NeRF. The batch size is set to 1024 rays for NeRF

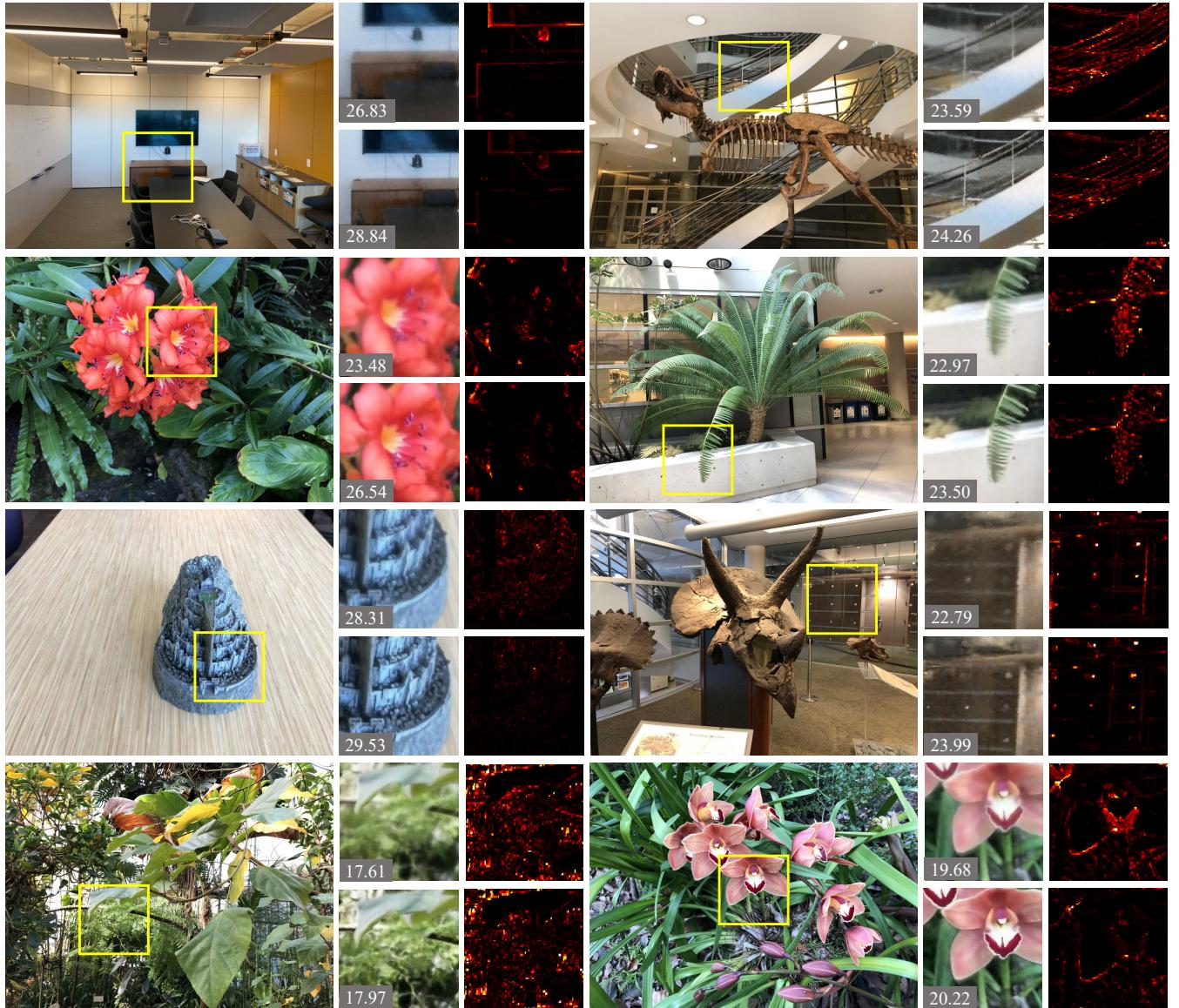
and the first stage of DoF-NeRF. We set  $N_{patch}$  to 32 in order to calculate the same amount of rays in every iteration.

- Vanilla NeRF takes 6h26min.
  - DoF-NeRF takes 6h41min, 3.8% longer than vanilla NeRF.
- Both models take about 6GB GPU memory during training.

### H.2 Testing

We render 120 images from different viewpoints using NeRF and DoF-NeRF model. All the synthesized images are of  $497 \times 331$  resolution.

- Vanilla NeRF takes 403s, *i.e.*, 3.36s per image.
- DoF-NeRF takes 405s, *i.e.*, 3.38s per image, 0.60% longer than vanilla NeRF.



**Figure 16: Additional comparison of DS-NeRF [4] and our approach in the synthetic dataset. For each scene, the zoom-in of rendered images and error maps (0 to 0.2 pixel intensity range) are presented. They are respectively obtained from DS-NeRF (first row) and our model (second row). For each subfigure, PSNR is shown on the lower left.**

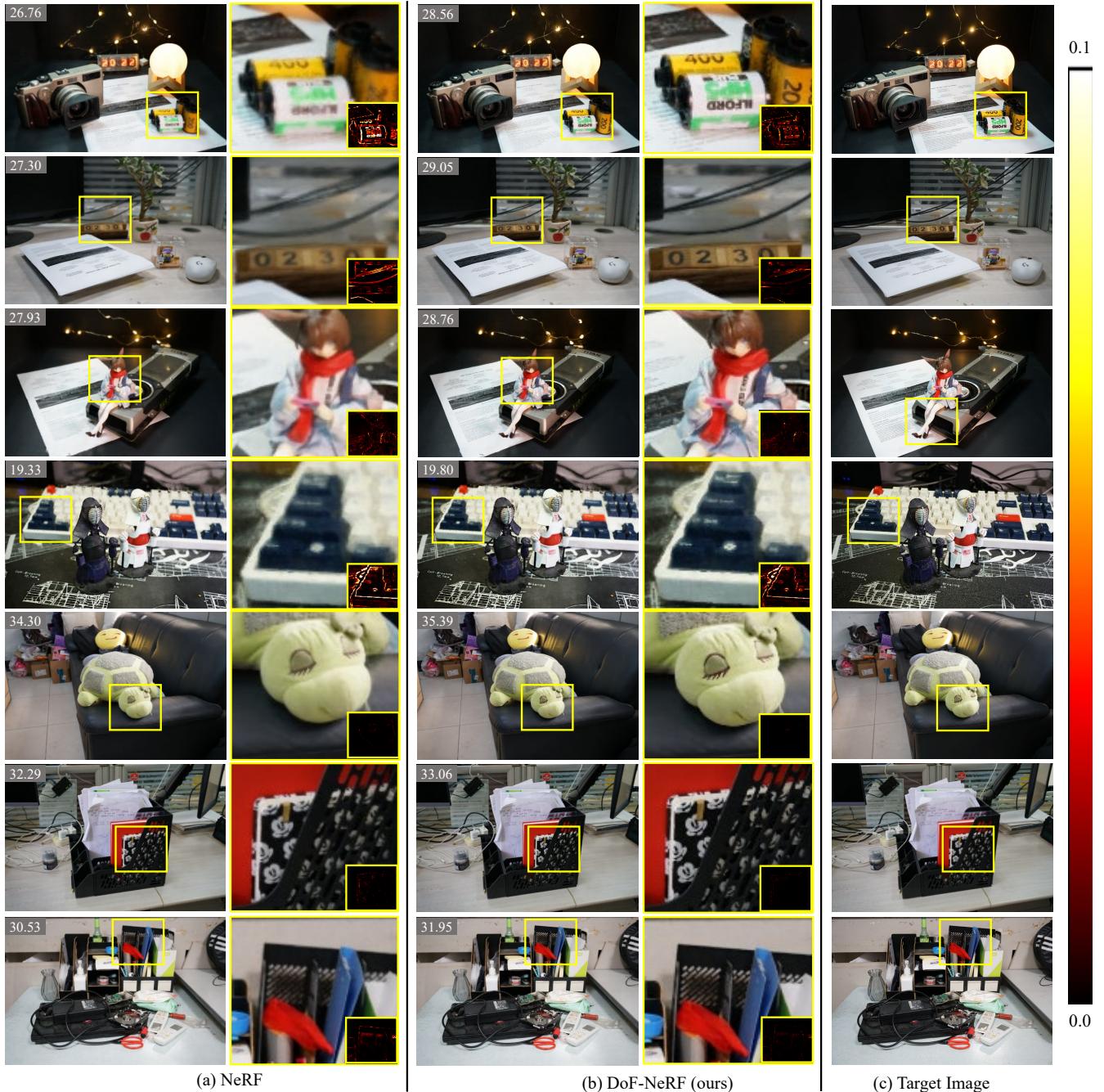
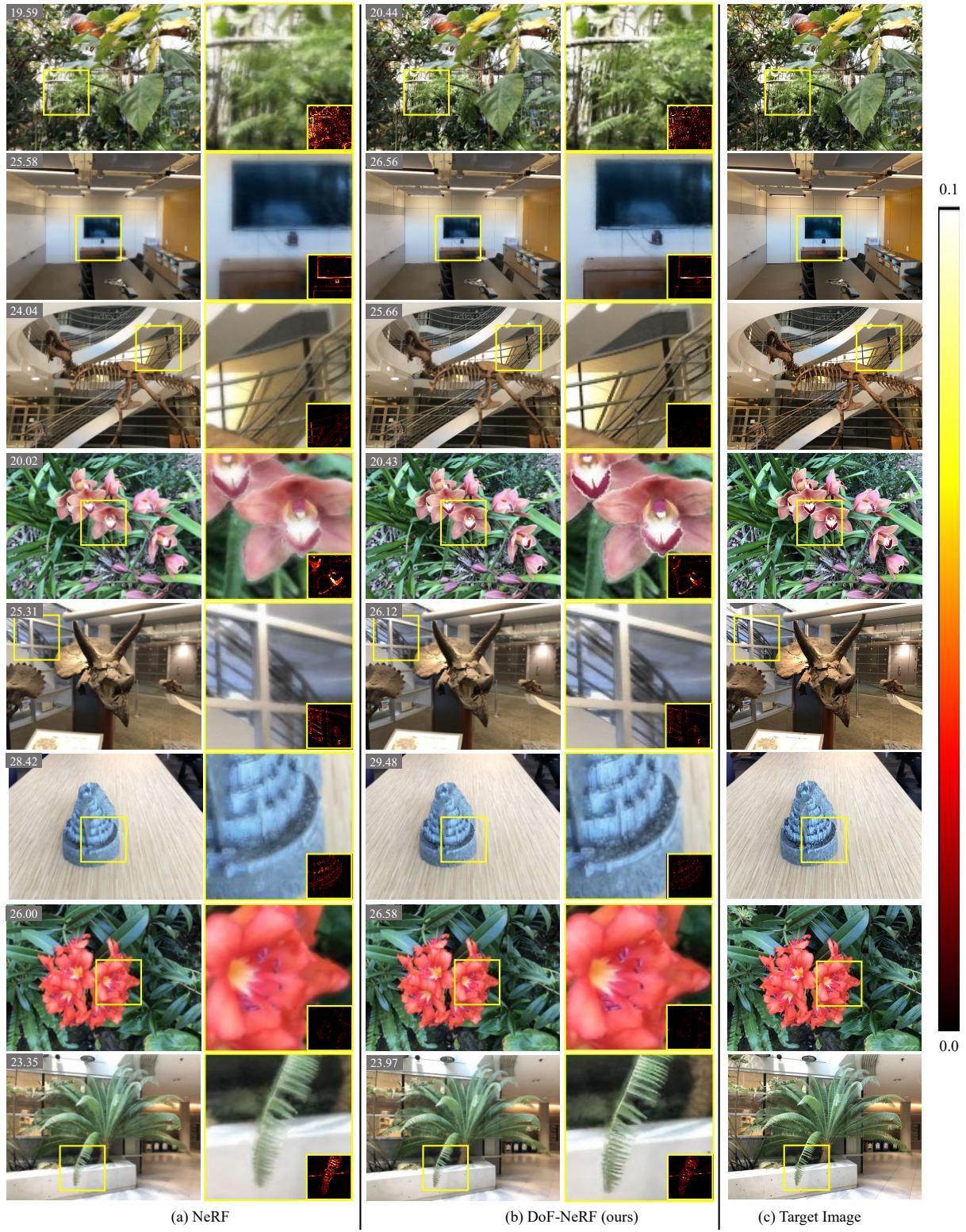


Figure 17: Additional comparison of NeRF [27] and our approach with shallow DoF inputs on the real-world dataset. The first and third column show the images rendered by NeRF and our approach, respectively. PSNR is shown at the upper left corner. We zoom in all the yellow boxes and present them in the second and fourth column with error maps (0 to 0.1 pixel intensity range) shown at the lower right corner.



**Figure 18: Additional comparison of NeRF [27] and our approach with shallow DoF inputs on the synthetic dataset. The first and third column show the images rendered by NeRF and our approach, respectively. PSNR is shown at the upper left corner. We zoom in all the yellow boxes and present them in the second and fourth column with error maps (0 to 0.1 pixel intensity range) shown at the lower right corner.**