DoF-NeRF: Depth-of-Field Meets Neural Radiance Fields Supplementary Materials

This document involves the following contents:

- Table of notations (Table 1).
- Physical model of DoF.
- Algorithm details of the concentrate-and-scatter method.
- Dataset and implementation details.
- Full experimental reports and more qualitative results.
- More synthesis of DoF effect.
- Cost of training and testing.

Table 1: 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
$\boldsymbol{x}$	Spatial location
$\boldsymbol{d}$	Viewing direction
$\boldsymbol{o}$	Camera origin
$\boldsymbol{p}$	Pixel on the imaging plane
$\boldsymbol{r_p}$	Ray specific to pixel $\boldsymbol{p}$
$\boldsymbol{c}$	Radiance predicted by MLP
α	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
Δ	Distance between adjacent sample points
h_c	Concentrated depth
δ	Diameter of CoC
δ_c	Diameter of CoC at the concentrated depth
$\mathcal{I}$	Set of input images
$\mathcal{P}$	Set of pixels
$\mathcal{K}$	Set of aperture parameters
$\mathcal{F}$	Set of focus distances
G_Θ, Θ	MLP network of NeRF and its parameters
$\mathcal{L}$	Loss
I	Input image
T	Accumulated transmittance
$\hat{I}(\boldsymbol{p})$	Predicted color of a pixel $\boldsymbol{p}$
$\hat{I}_{ray}(\boldsymbol{p})$	Diffused radiance of a pixel $\boldsymbol{p}$
$\hat{I}_{scatter}(\boldsymbol{p})$	Scattering radiance from other rays
$\hat{C}$	Scattering radiance of a spatial point
$\hat{C}_s(\boldsymbol{p})$	Diffused radiance of a pixel $\boldsymbol{p}$
K_{volume}	Volume rendering coefficient

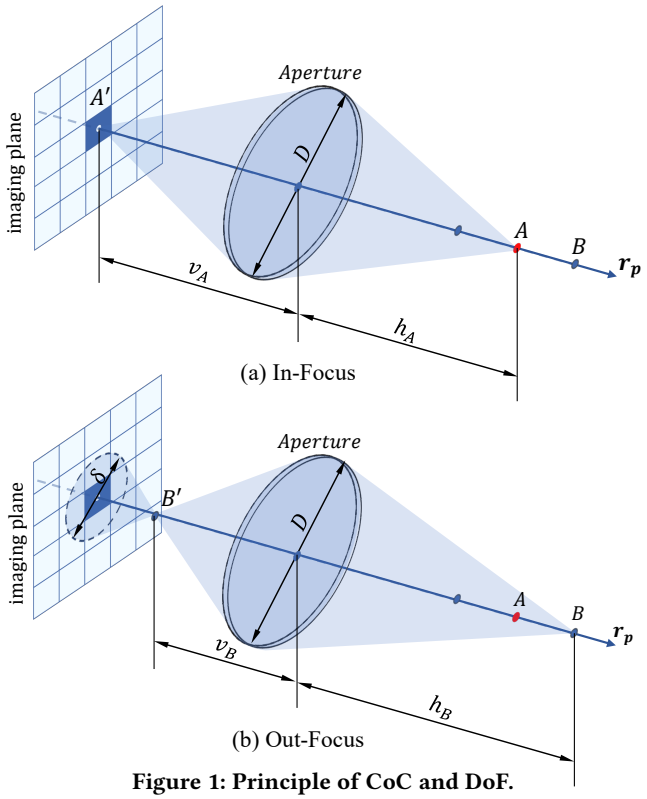


Figure 1: Principle of CoC and DoF.

1 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 diameter D, lights emitted from spatial points A and B on the ray $\boldsymbol{r_p}$ with object distance h_A and h_B converge on points A' and B' with imaging distance v_A and v_B, respectively (Fig. 1). The relation between h_A and v_A (similarly, h_B and v_B) follows the Gaussian formula

$$\frac{1}{h_A} + \frac{1}{v_A} = \frac{1}{f}, \tag{1}$$

$$\frac{1}{h_B} + \frac{1}{v_B} = \frac{1}{f}. \tag{2}$$

As shown in Fig 1, point A is imaged to the converging point A' on the imaging plane. In contrast, point B is projected to a CoC with diameter δ_B on the imaging plane. According to the properties of similar triangles, δ_B can be computed by

$$\frac{\delta_B}{D} = \frac{v_A - v_B}{v_B}. \tag{3}$$

Substituting Eq.(1) and Eq.(2) into Eq.(3) takes the form

$$\delta_B = fD \times \frac{h_B - h_A}{h_B(h_A - f)}. \tag{4}$$

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

$$\delta_B = fD \times \frac{h_B - F}{h_B(F - f)}. \tag{5}$$

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

$$\delta_C = fD \times \frac{F - h_C}{h_C(F - f)}. \tag{6}$$

Thus the diameter of CoC can be formulated by

$$\delta = fD \times \frac{|F - h_t|}{h_t(F - f)}, \tag{7}$$

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. (7) such that

$$\delta(h_t) = fD \times \frac{|h_t - F|}{Fh_t} = fD \times \left|\frac{1}{F} - \frac{1}{h_t}\right|. \tag{8}$$

2 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 1). 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_i 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

$$k_{ij} = \frac{sin\left(\frac{\pi}{2} - \frac{\pi}{n}\right)}{sin\left(\frac{\pi}{2} - \frac{\pi}{n} + mod\left(\left|arctan\left(\frac{l_{ij}^y}{l_{ij}^x}\right) + \phi\right|, \frac{2\pi}{n}\right)\right)}, \tag{9}$$

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

3 DATASET DETAILS

3.1 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. 2).

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 [5]. As an example, we visualize the predicted camera poses and sparse point cloud of the scene *kendo* in Fig. 3. The original resolution of the images in the real-world dataset is 3976×2652. We set the resolution to 497×331 for training and evaluation.

Algorithm 1: Pixel-wise scattering method

Input: Concentrated radiance C, concentrated depth H_c, aperture parameter K, focus distance F, gamma value γ
Output: Scattering result B_{cr}

1 $S \leftarrow K \cdot (\frac{1}{H_c} - \frac{1}{F})$;
2 $C \leftarrow (C)^{\gamma}$;
3 $W \leftarrow [0]$;
4 $I \leftarrow [0]$;
5 **for** $p_i \leftarrow$ *TraversePatch (C)* **do**
6 $r_i \leftarrow |S_i|$;
7 **for** $p_j \leftarrow$ *TraverseNeighbor* (p_i, r_i) **do**
8 $w_{ij} \leftarrow \frac{0.5+0.5\tanh(4(r_i - l_{ij}))}{{r_i}^2+0.2}$;
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 $B_{cr} \leftarrow (B_{cr})^{\frac{1}{\gamma}}$;

3.2 Synthetic Dataset

The synthetic dataset is generated from the Real Forward-Facing dataset [2] 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 [4] to generate the disparity map and BokehMe [3] 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. 4). The original resolution of the images in the synthetic dataset is 4032×3024. We set the resolution to 504×378 for training and evaluation.

4 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

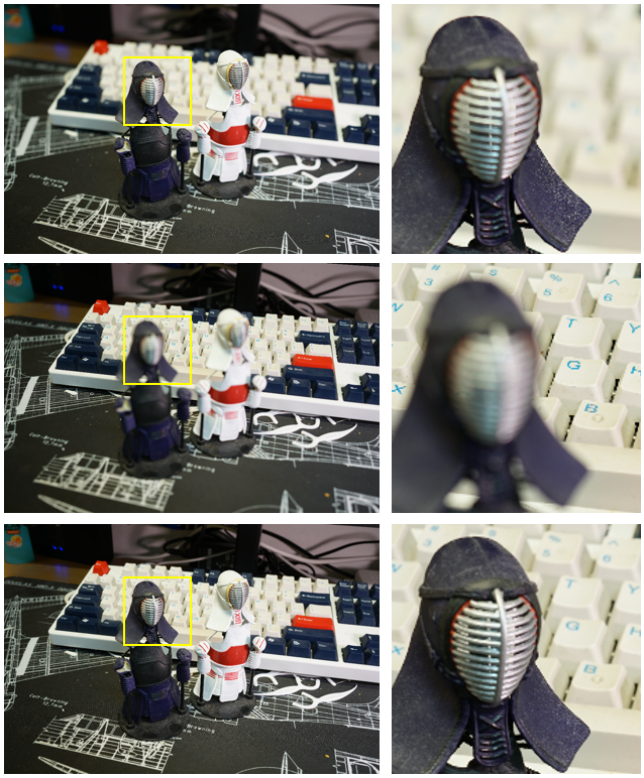


Figure 2: Visualization of the scene *kendo* in the real-world dataset. Each row respectively represents the foreground-focusing shallow DoF image, background-focusing shallow DoF image, and wide DoF image. We zoom in the yellow boxes in the images and present them on the right.

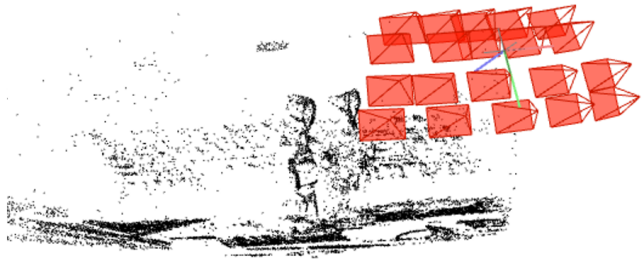


Figure 3: Visualization of the predicted camera poses and reconstructed sparse point cloud by COLMAP [5].

network. As explained in the main paper, we adopt a two-stage optimization: each stage takes $200k$ iterations, where $N_{pretrained}$ and N_{iters} are set to $200k$ and $400k$, respectively. For all-in-focus inputs, we change $N_{pretrained}$ and N_{iters} to $400k$ and $800k$, respectively. At the second optimization stage, we set N_{patch} to 48 and N_{anchor} to 16. The aperture and focus parameters are both initialized with 0.5 for DoF-NeRF. For ablation study, the aperture and focus parameters are set to 0.5 for initialization and fixed according to the experiment settings.

Figure 4: Visualization of the scene *leaves* in the synthetic dataset. The first row shows the original image from the Real Forward-Facing dataset [2] and the predicted disparity map from DPT [4]. The second and third rows are the shallow DoF images rendered by BokehMe [3]. We zoom in the yellow boxes in the images and present them on the right.

Table 2: Comparison of NeRF [2] and our framework in the real-world dataset.

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

5 EXPERIMENTAL RESULTS

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

Table 3: Comparison of NeRF [2] and our framework in the synthetic dataset.

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

Table 4: Comparison of DSNeRF [1] and our framework in the synthetic dataset.

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

We extend Table 1 and Table 2 in the main paper by conducting experiments in other scenes of the synthetic dataset and real-world dataset. Table 2 and Table 3 report the rendering quality of vanilla NeRF and our model in the real-world dataset and synthetic dataset, respectively. Fig. 7 and Fig. 8 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 4 shows the rendering quality of our method using DS-NeRF 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. 6.

Table 5: Comparison of NeRF [2] and our framework in the all-in-focus dataset.

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

Table 6: Ablation studies of components of our model. "Aperture" and "Focus" denote learnable aperture parameters and focus distances respectively.

Scene	Model	Aperture	Focus	PSNR(↑)	SSIM(↑)	LPIPS(↓)
room	NeRF	–	–	27.279	0.8863	0.1803
	ours	✓	×	27.799	0.8992	0.1640
	ours	×	✓	28.070	0.9032	0.1671
	ours	✓	✓	**28.361**	**0.9076**	**0.1556**
camera	NeRF	–	–	25.742	0.8723	0.1657
	ours	✓	×	26.639	0.8989	0.1338
	ours	×	✓	25.855	0.8786	0.1532
	ours	✓	✓	**26.962**	**0.9045**	**0.1280**

Although our framework is originally designed for shallow DoF inputs, Table 5 indicates that it shows comparable performance against vanilla NeRF using all-in-focus images as inputs.

Table 6 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.

6 DOF RENDERING

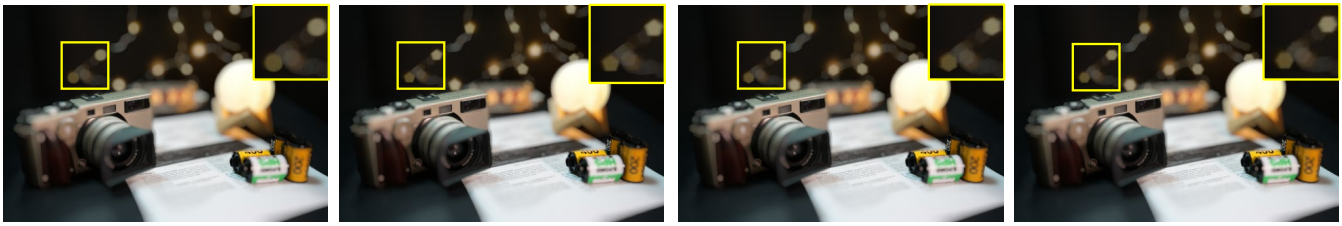
In this section, we present more synthesis of controllable DoF rendering (Fig. 5). Benefiting from the explicit modeling of DoF effect, DoF-NeRF enables direct manipulation of DoF effect by adjusting virtual aperture size and focus distance. Compared with single-image DoF rendering method, DoF-NeRF is 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.

All-in-Focus Various Aperture Various Aperture Various Focus Distance

(a) Changing Aperture Size and Focus Distance

(b) Changing Aperture Size and Camera Poses

(c) Changing Aperture Size and Perspective



Circular CoC Pentagonal CoC Hexagonal CoC Rotated Hexagonal CoC

(d) Changing Aperture Shape

Figure 5: 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.

7 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.

7.1 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.

7.2 Testing

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

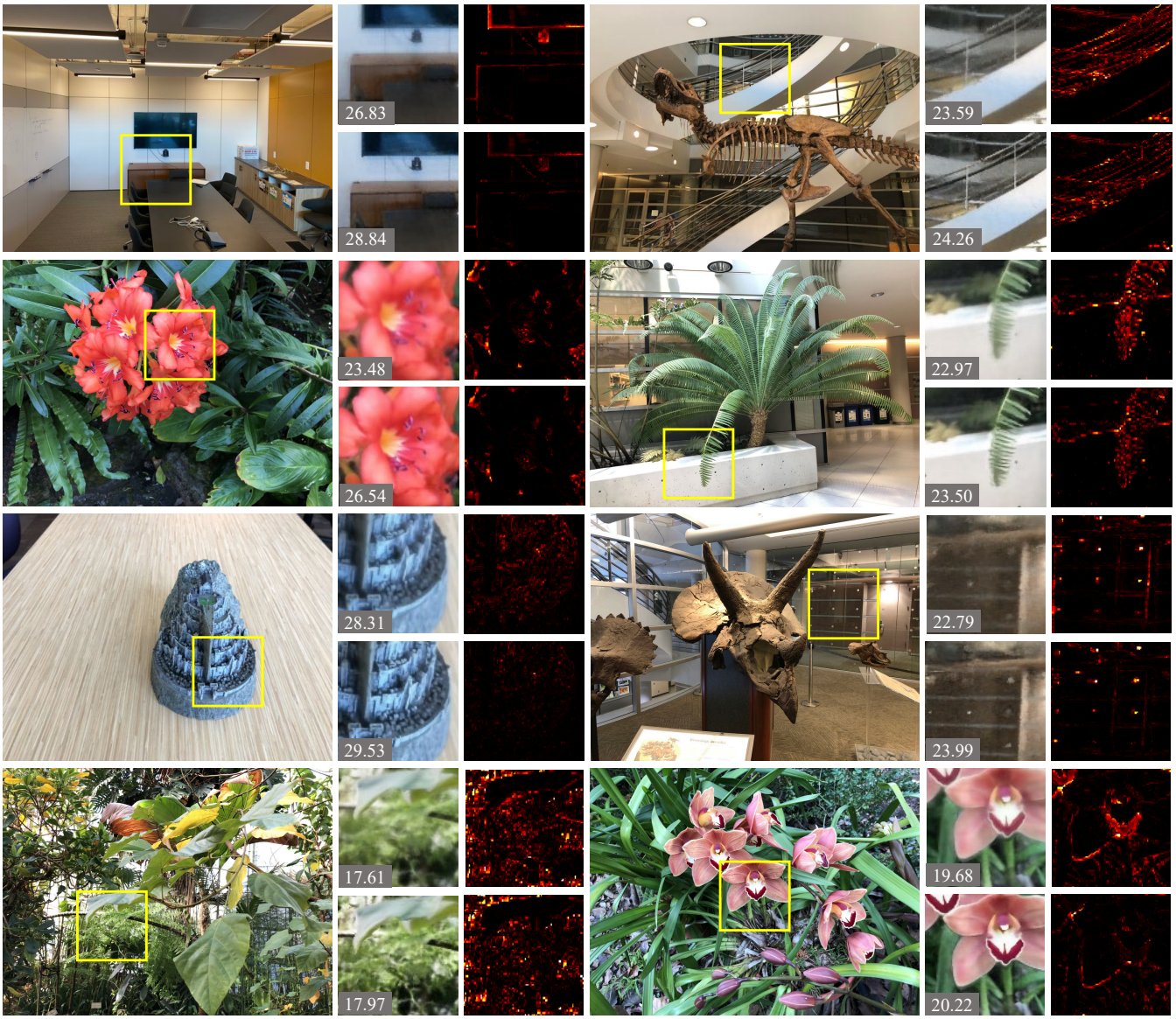


Figure 6: Additional comparison of DS-NeRF [1] and our approach in the synthetic dataset. For each scene, the zoom-in of rendered images and error maps (0 to 0.2 pixel intensity range) are presented. They are respectively obtained from DS-NeRF (first row) and our model (second row). For each subfigure, PSNR is shown on the lower left.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

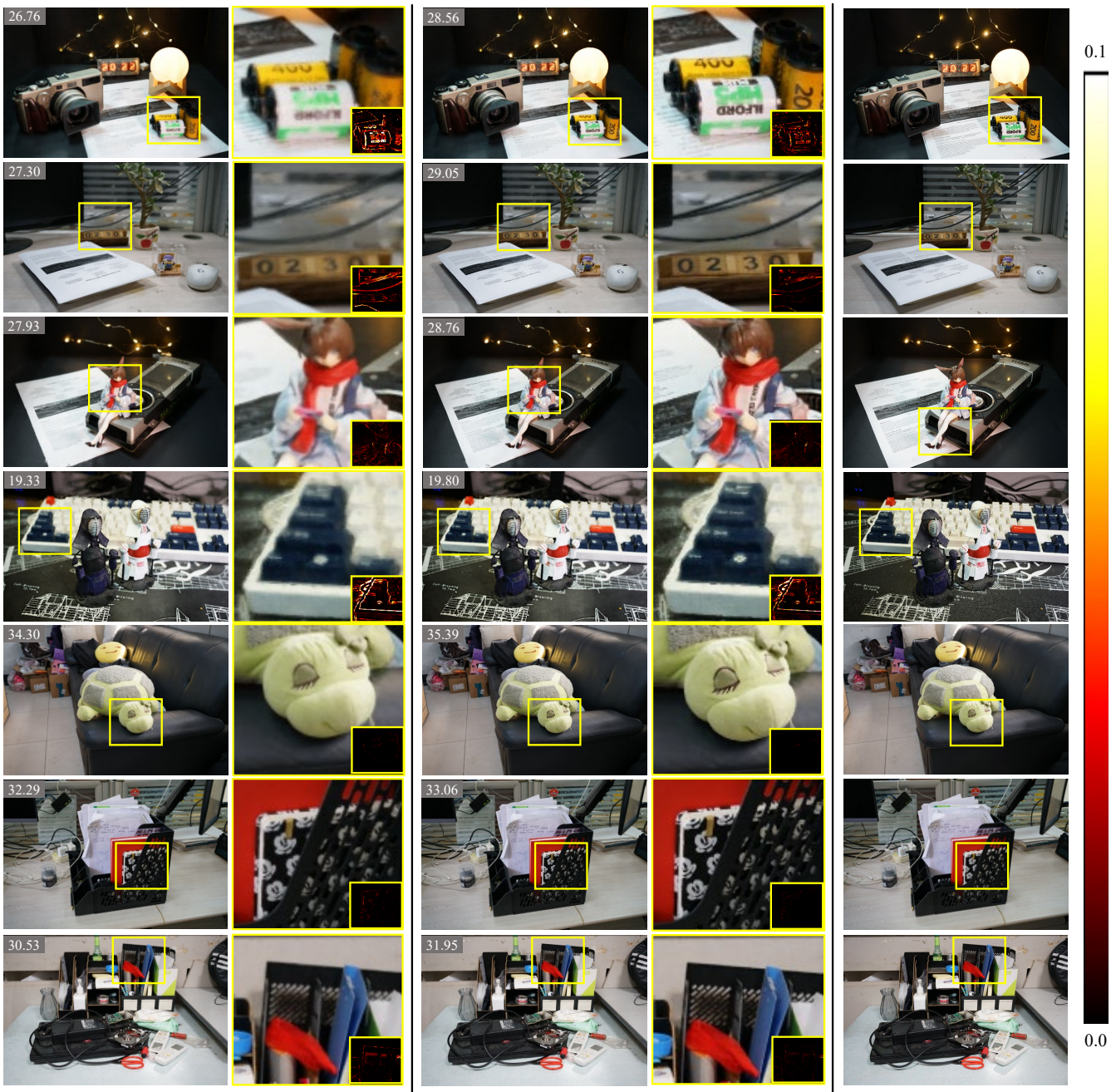


Figure 7: Additional comparison of NeRF [2] and our approach with shallow DoF inputs on the real-world dataset. The first and third column show the images rendered by NeRF and our approach, respectively. PSNR is shown at the upper left corner. We zoom in all the yellow boxes and present them in the second and fourth column with error maps (0 to 0.1 pixel intensity range) shown at the lower right corner.

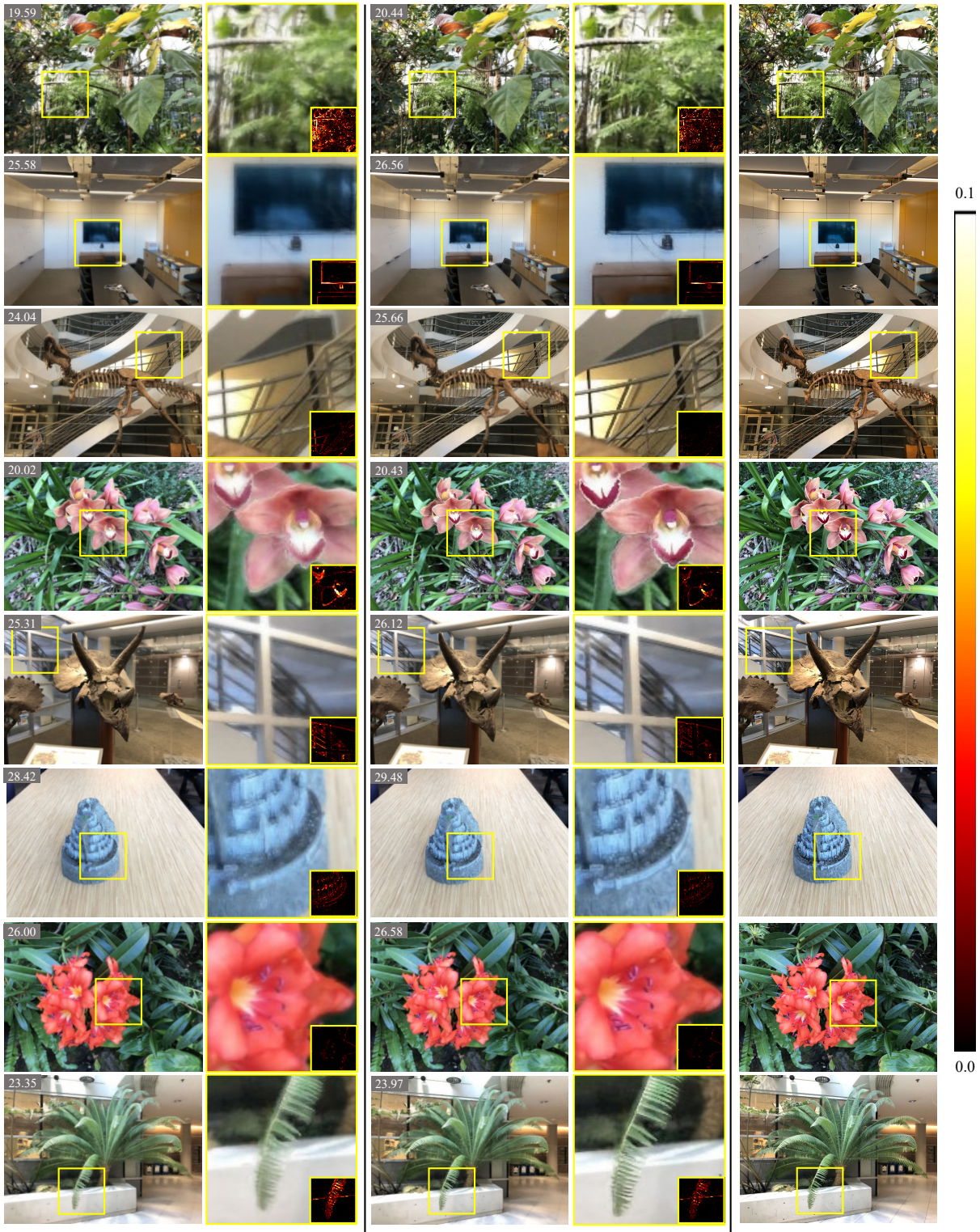


Figure 8: Additional comparison of NeRF [2] and our approach with shallow DoF inputs on the synthetic dataset. The first and third column show the images rendered by NeRF and our approach, respectively. PSNR is shown at the upper left corner. We zoom in all the yellow boxes and present them in the second and fourth column with error maps (0 to 0.1 pixel intensity range) shown at the lower right corner.