Supplemental Material

This supplemental material includes additional details and experimental results about our LensNeRF and the baselines. In Sec. I, we illustrate the procedure of how we obtained $t^{(foc)}$ used for the in-focus loss. Sec. II details the ray undistortion procedure that we exploit via pseudo-code. Further details on how we perform the evaluations using the baseline methods are described in Sec. III. We explain dataset properties in Sec. IV through sample images. Additional qualitative and quantitative results are enumerated in Sec. V.

I. Additional Details of In-Focus Loss

In-focus point sampling factor. This part elaborates on how we obtain $t^{(foc)}$ in Eq. (11). $t^{(foc)}$ is a scalar value that makes the sampled point to be on the in-focus plane satisfying

$$\boldsymbol{x}^{(foc)} = \boldsymbol{o'} + t^{(foc)}\boldsymbol{d'}, \tag{14}$$

where $\boldsymbol{o'}$ is a ray origin in the normalized device coordinate (NDC) space and $\boldsymbol{d'}$ is a ray direction in the NDC space. To find $t^{(foc)}$, we first find the point where all the rays converge in the non-NDC space and then convert it to the NDC space, *i.e.*,

$$\pi(\boldsymbol{o} + t_{pre}^{(foc)}\boldsymbol{d}) = \boldsymbol{o'} + t^{(foc)}\boldsymbol{d'} \tag{15}$$

$$\boldsymbol{d} = \frac{\boldsymbol{r}_{out}^{(i,j)}}{||\boldsymbol{r}_{out}^{(i,j)}||}, \tag{16}$$

where π projects ray origin $\boldsymbol{o}$ and direction $\boldsymbol{d}$ in non-NDC space to ray origin $\boldsymbol{o'}$ and direction $\boldsymbol{d'}$ in NDC space. The $\boldsymbol{r}_{out}^{(i,j)}$ is a ray contributes to the pixel $\boldsymbol{p}_i$, originating from $\boldsymbol{o}_j$ as depicted in Fig. 4. Connection between $t^{(foc)}$ and $t_{pre}^{(foc)}$ is described in NeRF [4] as

$$t^{(foc)} = 1 - \frac{o_z}{o_z + t_{pre}^{(foc)} d_z}. \tag{17}$$

We adapt Eq. (17) to convert $t_{pre}^{(foc)}$ in non-NDC space to $t^{(foc)}$ in NDC space.

To derive $t_{pre}^{(foc)}$ of a ray passing through the lens center, contributing to the pixel $\boldsymbol{p}_i$, we investigate the point where all the rays in $\mathcal{R} = \{\boldsymbol{r}_{out}^{(i,j)} | j \in \mathcal{J}\}$ intersect. Here, $j \in \mathcal{J} = \{0, 1, ..., N_o - 1\}$ is the index of ray origin on the thin-lens surface and i is the pixel index of our interest. It is already known that when two rays are defined as $\boldsymbol{r}_1 = \boldsymbol{o}_1 + t_1\boldsymbol{d}_1$ and $\boldsymbol{r}_2 = \boldsymbol{o}_2 + t_2\boldsymbol{d}_2$, the value of t_1 that minimizes the distance between the two lines can be calculated as follows [1].

$$\begin{aligned} t_1^* &= \boldsymbol{f}(\boldsymbol{r}_1(\boldsymbol{o}_1, \boldsymbol{d}_1), \boldsymbol{r}_2(\boldsymbol{o}_2, \boldsymbol{d}_2)) \\ &= \frac{(\boldsymbol{o}_2 - \boldsymbol{o}_1) \cdot (\boldsymbol{d}_2 \times (\boldsymbol{d}_2 \times \boldsymbol{d}_1))}{\boldsymbol{d}_1 \cdot (\boldsymbol{d}_2 \times (\boldsymbol{d}_2 \times \boldsymbol{d}_1))}. \end{aligned} \tag{18}$$

Assume that the ray passing through the lens center is a reference ray $\boldsymbol{r}_1$. Then the point where all the other rays converge is $\boldsymbol{o}_1 + t^{(foc)}\boldsymbol{d}_1$. From this, $t^{(foc)}$ of i'th pixel can be simply defined as

$$t^{(foc)} = \frac{\sum_{j \in \mathcal{J}, \boldsymbol{o}^{(c)} \neq \boldsymbol{o}_j} \boldsymbol{f}\left(\boldsymbol{r}_1\left(\boldsymbol{o}^{(c)}, \boldsymbol{r}_{in}^{(i)}\right), \boldsymbol{r}_2\left(\boldsymbol{o}_j, \boldsymbol{r}_{out}^{(i,j)}\right)\right)}{|\mathcal{J}| - 1}, \tag{19}$$

where $\boldsymbol{o}^{(c)}$ is the lens center. Note that $r_{in}^{(i)}$ is the same as $r_{out}^{(i,c)}$, when c is the index of ray origin at lens center.

II. Ray Undistortion

To find accurate $t^{(foc)}$, rays are undistorted using distortion parameters obtained from calibration. The following shows the pseudo-code of the ray undistortion procedure. Here, k_1, k_2, p_1, p_2 are radial distortion parameters, $\mathbf{r}_d$ is an initial distorted ray, and λ_t is a hyper-parameter with the value of $1e$-5

Algorithm 1 Ray Undistortion

function UNDISTORT($\boldsymbol{r}_d, k_1, k_2, p_1, p_2$)
    function DISTORT(x, y, k_1, k_2, p_1, p_2)
        $R \leftarrow x^2 + y^2$
        $D \leftarrow 1 + k_1 R + k_2 R^2$
        $x_d \leftarrow Dx + 2p_1 xy + p_2(R + 2x^2)$
        $y_d \leftarrow Dy + p_1(R + 2y^2) + 2p_2 xy$
        return $[x_d, y_d, -1]$
    end function
    $\boldsymbol{r}_u \leftarrow \boldsymbol{r}_d$
    while True **do**:
        e $\leftarrow$ DISTORT($\boldsymbol{r}_u[0], \boldsymbol{r}_u[1], k_1, k_2, p_1, p_2$) - $\boldsymbol{r}_d$
        $\boldsymbol{r}_u \leftarrow \boldsymbol{r}_u$ - e
        if $|err| \leq \lambda_t$ **then**
            break
        end if
    end while
    return $\boldsymbol{r}_u$
end function

III. Description of Baseline Methods

DiskBlur+DVGO. The objective of novel view synthesis with defocus blur is synthesizing images with a small F-number, given in-focus images only. A simple yet naive way to achieve this goal is to directly train the NeRF architecture using synthetically blurred input images. To accomplish this, we first convert sharp images into blurry images that are as similar as possible to those captured using the target F-number. Referring to RawNeRF [3] and Srinivasan *et al.* [6], we first train DVGO using sharp images and apply

TL-NeRF Dataset				
Scene Index	**Scene Name**	**#Views (for each F-num)**	**F-nums**	**#Images (Total)**
1	AmusementPark	34	F4, F5.6, F8, F22	136
2	AppleMint	40	F4, F5.6, F8, F22	160
3	Bear	42	F4, F5.6, F8, F22	168
4	BoyAndGirl	29	F4, F5.6, F8, F22	116
5	Chrysanthemum	44	F4, F5.6, F8, F22	176
6	Gink	50	F4, F5.6, F8, F22	200
7	Sheep	47	F4, F5.6, F8, F22	188
8	Snowman	39	F4, F5.6, F8, F22	156
9	Xmas	47	F4, F5.6, F8, F22	188

Table 6. Dataset configuration.

disk kernel on Multi-Plane Image(MPI) defined in neural radiance fields (depth-wide manner). The disk kernel size d_i for each plane is defined as follows:

$$d_i = \text{round}\left(\frac{c}{\text{F-number}}\,|i - i_f|\right) \tag{20}$$

$$d_i \mathrel{+}= ((d_i + 1)\%2), \tag{21}$$

where i is the index of a plane, i_f is the index of an in-focus plane, d_i is the disk kernel size of the plane with index i, and c is the hyper-parameter. Note that as the difference between the i and i_f gets bigger, disk kernel size d_i gets larger, which makes sense since the point far from the in-focus plane is more blurry compared to the one near the in-focus plane. To get final blurry train images, a conventional volume rendering scheme is applied on blurred MPI. Although access to the images with target F-number is not allowed in usual cases, to be generous to our baseline, we peek at a single image with the target F-number that matches the viewpoint of the selected train image. And then, we perform the grid search over hyper-parameter c that minimizes the difference between the peeked image with the target F-number and the synthetic blurred image generated from the selected train image. Synthetic blurry images generated from train images are used to train DVGO. After training, novel view images are created by volume rendering scheme and utilized for the evaluation in Sec. 5.3.1, with a name of 'DiskBlur+DVGO'.

The first row of the Fig. 7 shows the sampled train images used for the DiskBlur+DVGO and our method. See how the train image domain of the DiskBlur+DVGO is similar to that of the ground-truth test image domain. Even with such a similarity, the baseline model fails to employ given blur information to novel view, while our LensNeRF can generate a novel view image similar to the ground-truth image without blur prior.

NeRFocus. Given all-in-focus images as a train set, NeRFocus [8] synthesizes novel view defocus images using NeRF framework. They define a parameter A' in relative scale and manipulate aperture size by adjusting it. All-in-focus image is obtained when A' has the value of 0.

Figure 7. Train, test, and synthesized image samples in novel view defocus task.

Similar to DiskBlur+DVGO evaluation case, we peek at a single ground-truth wide-aperture image that corresponds to one of the train images with a narrow aperture. And then, we conduct the grid search over A' to find the value that explains a referenced ground-truth image the best. Starting from 0, the parameter value is increased by the amount of 0.005. We do early stopping when the mean square error between two images increases for five steps and keep the best performing A'. Defocus images synthesized by NeRFocus [8] are used for evaluation in Sec. 5.3.1.

DoF-NeRF. Unlike other baseline methods, DoF-NeRF [9] is not only capable of training NeRF using defocus images but also able to adjust aperture size for novel-view synthesis. DoF-NeRF defines two parameters for each viewpoint, which are the aperture parameter $\mathcal{K}$ and focus distance $\mathcal{F}$. Here, $\mathcal{K}$ is defined as the multiplication of focal length and aperture diameter, $\mathcal{K} = f \times D$. To evaluate DoF-NeRF, we need to estimate parameters $\mathcal{K}$ and $\mathcal{F}$ for the target F-number. We estimate them, we first train DoF-NeRF with the target F-number images and average $\mathcal{K}$s and $\mathcal{F}$s for all views. Estimated parameters $\mathcal{K}$ and $\mathcal{F}$ are used for novel view synthesis, having the same value regardless of its viewpoint. For instance, for defocus task that is trained on F22 images and aims to render F4 images, we train DoF-NeRF using F4 images and average out the estimated parameters $\mathcal{K}$ and $\mathcal{F}$ for all views. Then use these estimated parameters for target-view F4 image rendering. Since the aperture size, image focal length, and lens focal length are fixed for each scene with the same F-number in our dataset, the average operation is a reasonable approach rather than a rough assumption. Since DoF-NeRF provide only a fraction of the code, omitting the files for the training and rendering, we complete the code based on the details from the DoF-NeRF paper and use it for our evaluation.

IV. Dataset overview

As stated in Sec. 5.1, to rigorously evaluate our proposed method, we construct a dataset of nine scenes for novel view synthesis task with varying aperture sizes. Each scene consists of images taken from different viewpoints. For each viewpoint, we collect four images with different F-numbers, which are F4, F5.6, F8, and F22. To preclude unwanted external force on the camera, we change the F-number remotely using the software supported by Canon, so that we can capture an image without pressing the button on the camera. Tab. 6 shows the overall structure of our proposed dataset, consisting of 1488 images in total. Every 8th viewpoint is selected as a test set. All baseline methods are trained for each scene from scratch. Sampled images of the given F-number and the given scene can be found in Fig. 8.

V. Additional Experimental Results

Here, we introduce additional experimental results. The brightness of the resulting images is adjusted for ease of comparison. In cases where the F-numbers in the training dataset are inhomogeneous, we employ the 'Fmix' notation. To generate the Fmix dataset, we interleave F-numbers sequentially for every viewpoint, for instance, an image with F-number 4 is selected for the first viewpoint, an image with F-number 5.6 is selected for the second viewpoint, an image with F-number 8 is selected for the third viewpoint, an image with F-number 22 is selected for the fourth viewpoint, and an image with F-number 4 is selected for the fifth viewpoint, and so on.

The Number of Outer Loops Tab. 7 shows the effect of increasing the number of outer loops N_{outer}. Raising the N_{outer} value gives better results until it reaches the value of three. We choose three as our final decision for N_{outer}.

Deblur Task		PSNR ↑	SSIM ↑	LPIPS ↓	DISTS ↓	AGG ↓
Methods	N_{outer}	F4 → F22				
LensNeRF	1	25.5282	0.8392	0.2844	0.1432	0.0822
LensNeRF	2	<u>26.1382</u>	0.8392	0.2824	0.1415	<u>0.0790</u>
LensNeRF	3	**26.1644**	**0.8447**	**0.2753**	**0.1386**	**0.0777**
LensNeRF	4	25.9976	<u>0.8413</u>	<u>0.2814</u>	<u>0.1409</u>	0.0794

Table 7. Ablation study of increasing the number of outer loops.

Qualitative results on F-number variation. Our LensNeRF not only permits varying aperture sizes for the input images but also for the output images, as illustrated in Fig. 9. Inspect our results in a column-wise manner and note the extent to which our resulting images resemble the ground-truth images depicted in the bottom row. As evident from the results, regardless of the F-numbers of the input images, the proposed LensNeRF demonstrates a high degree of reproducibility of the target image.

More qualitative results on deblur and defocus. We present additional qualitative results on the deblur and defocus tasks, supplementing those shown in Fig. 3.

Additional qualitative results for defocus task are demonstrated in Fig. 10. As we can see, our LensNeRF model is capable of achieving a depth-aware defocus effect, where the background is blurred while the foreground details are rendered sharply, resulting in more realistic images.

More qualitative results on classic task. Here, we carry additional qualitative results on classic task in Fig. 12. These results show the case when the train images and the test images have identical F-number. Check how our LensNeRF can learn decent neural radiance fields from the train images with target F-number and reproduce novel view images using the learned model.

Quantitative results for each scene. In Sec. 5, quantitative Results are averaged to show the effectiveness of the proposed method in a compact manner. Here we show the non-abbreviated, full version of the quantitative results. Tab. 8 and Tab. 9 presents the quantitative results of the novel view synthesis for defocus task. Tab. 10 and Tab. 11 depicts the quantitative results of the novel view synthesis for deblur task. Tab. 12 and Tab. 13 illustrates the quantitative results of the novel view synthesis for classic task.

Video material Our LensNeRF can synthesize novel view images corresponding to the given focal lengths, F-numbers, and camera poses. To show the model capability, we attach a video file, '2024_WACV_LensNeRF.mp4', that visualizes the three scenarios.

The first scenario concerns the defocus task. Using the networks trained on images with F-number 22 (F22), where F22 represents the camera with a narrow aperture, novel view images with varying aperture sizes are generated. Results of our LensNeRF are compared with those of NeRFocus [8] and DoF-NeRF [9]. The notable regions are highlighted with red and blue boxes in the video. The blue box highlights the region where the synthesized defocus blur of our LensNeRF successfully reflects the depth variation, showing the meaningful difference between the foreground and the background region, while the other methods cannot. The red box in the video emphasizes the region where baseline methods shows the artifacts. DiskBlur+DVGO results are not included since this requires training multiple networks for every F-number that we want to synthesize and should run the corresponding model frame by frame, which is quite intractable.

The second scenario is for the deblur task. In this task, we use the networks trained on images with the F-number 4 (F4), where F4 represents the camera with a wide aperture. Using the trained networks, novel view images with varying camera poses are exhibited. Here, synthesized images are assumed to be captured from the F22 camera. The more crispy the results are, the better the model is. We compare our LensNeRF with KPAC+DVGO, DeblurNeRF [2], and DoF- NeRF [9] in the video.

The third scenario is a novel view synthesis of our LensNeRF with varying aperture sizes, focal lengths, and camera poses. For this third scenario LensNeRF is trained on the images with F-number 22.

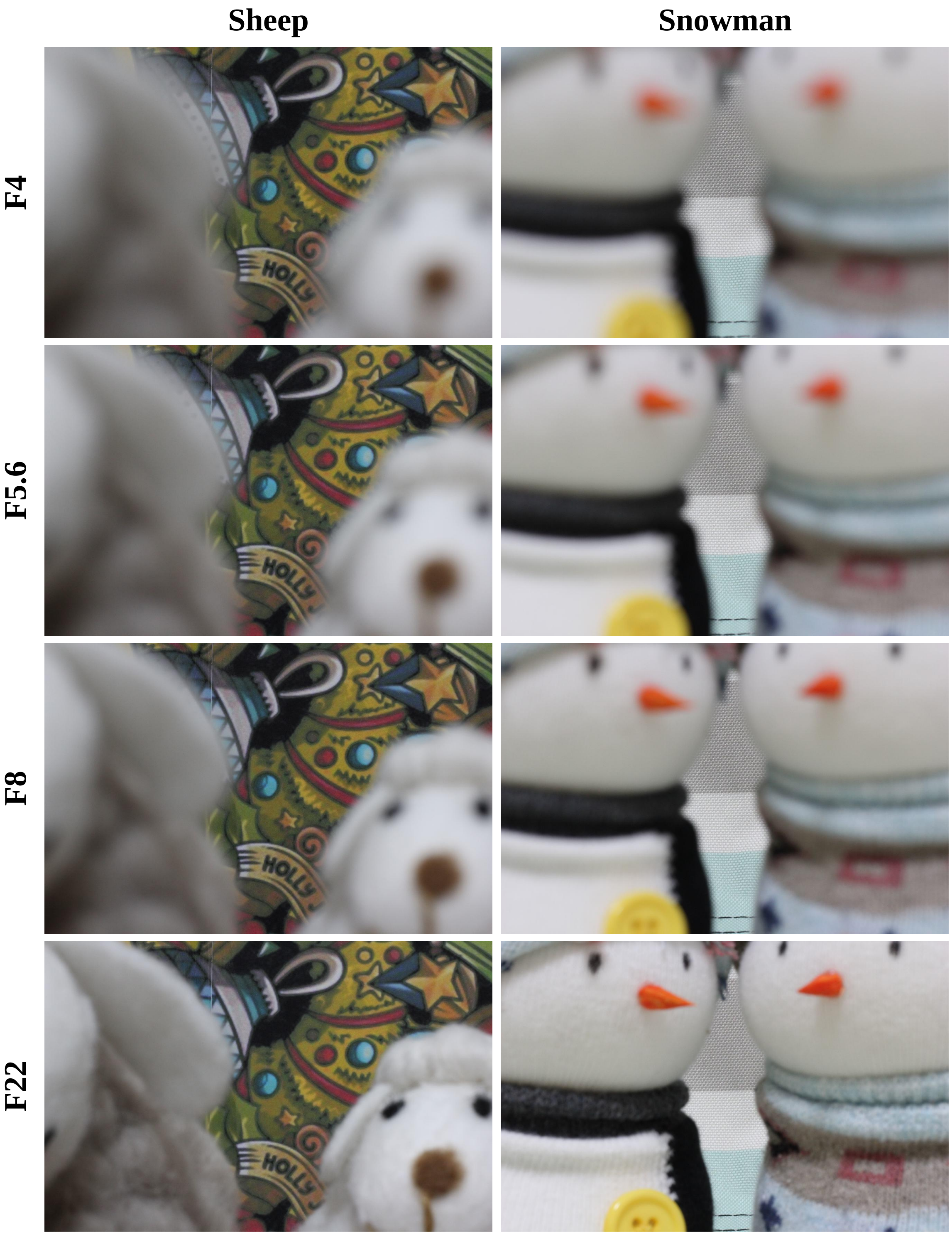

Figure 8. Sample images of our dataset. For each camera position, images with four different F-numbers are captured.

Figure 9. Additional qualitative results of our LensNeRF with a varying F-number. The F-number of the train images is denoted on the left side of the arrow, and the F-number of the target images is indicated on the right.

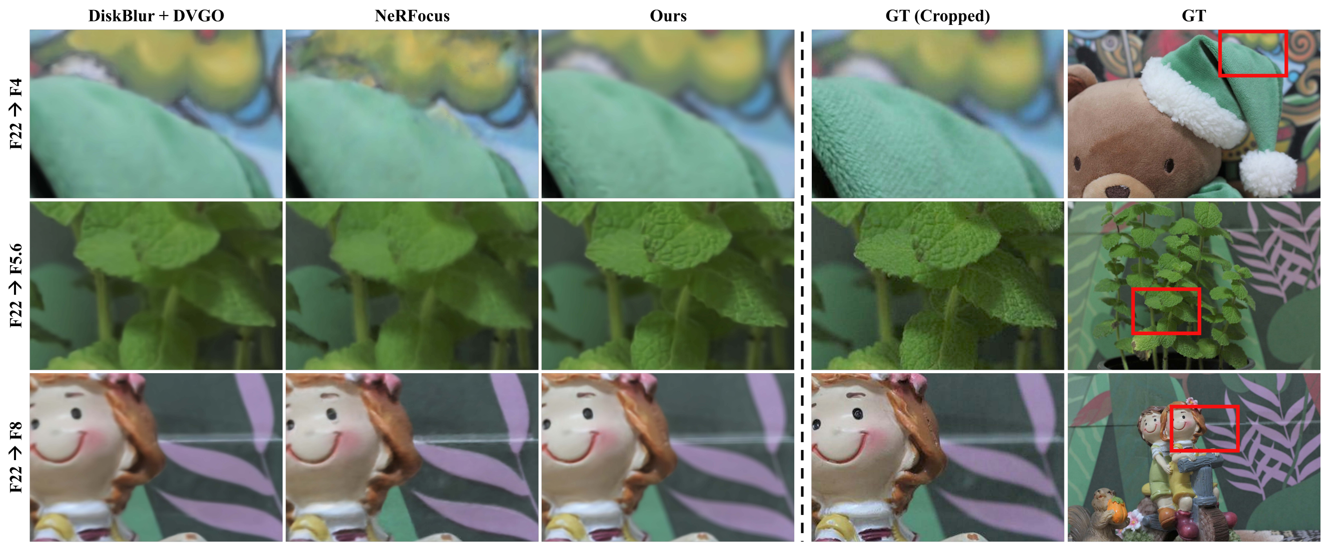

Figure 10. Additional qualitative results on novel view synthesis for the defocus task. The red box region on the ground-truth(GT) image is cropped and zoomed for ease of comparison.

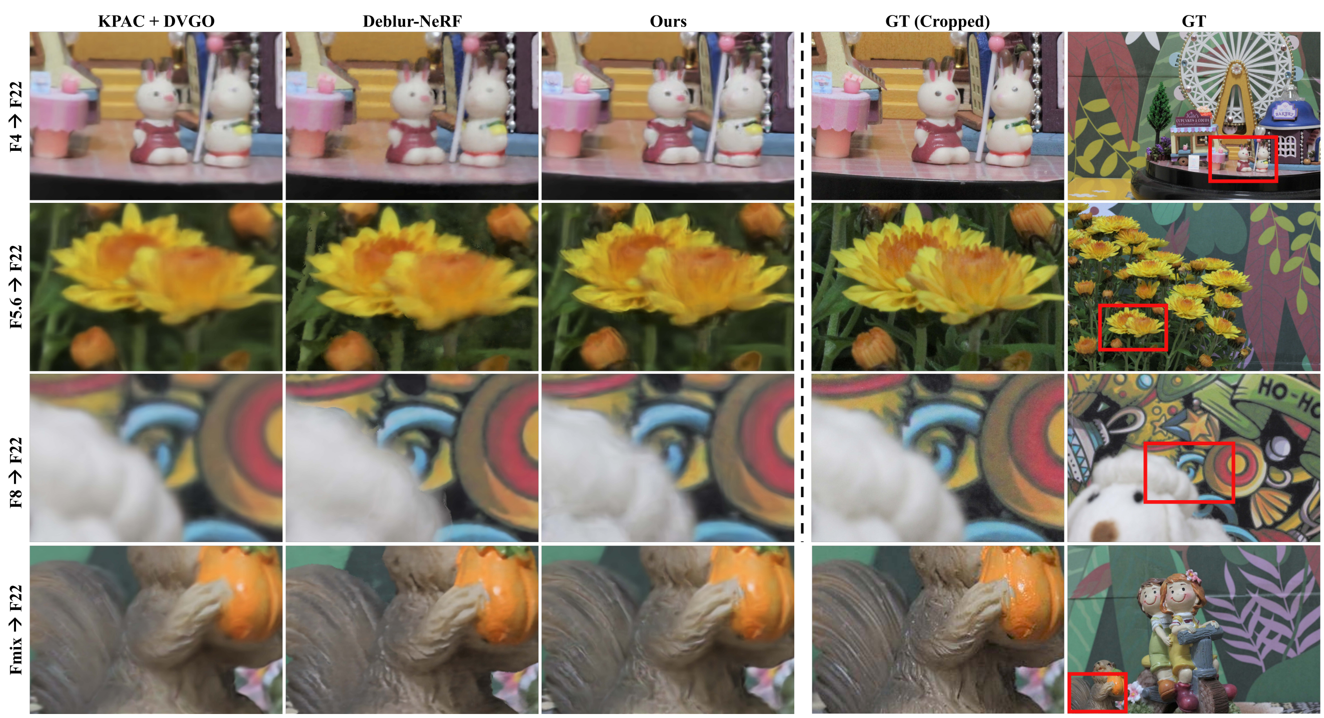

Figure 11. Additional qualitative results on novel view synthesis for the deblur task. The red box region on the ground-truth(GT) image is cropped and zoomed for ease of comparison.

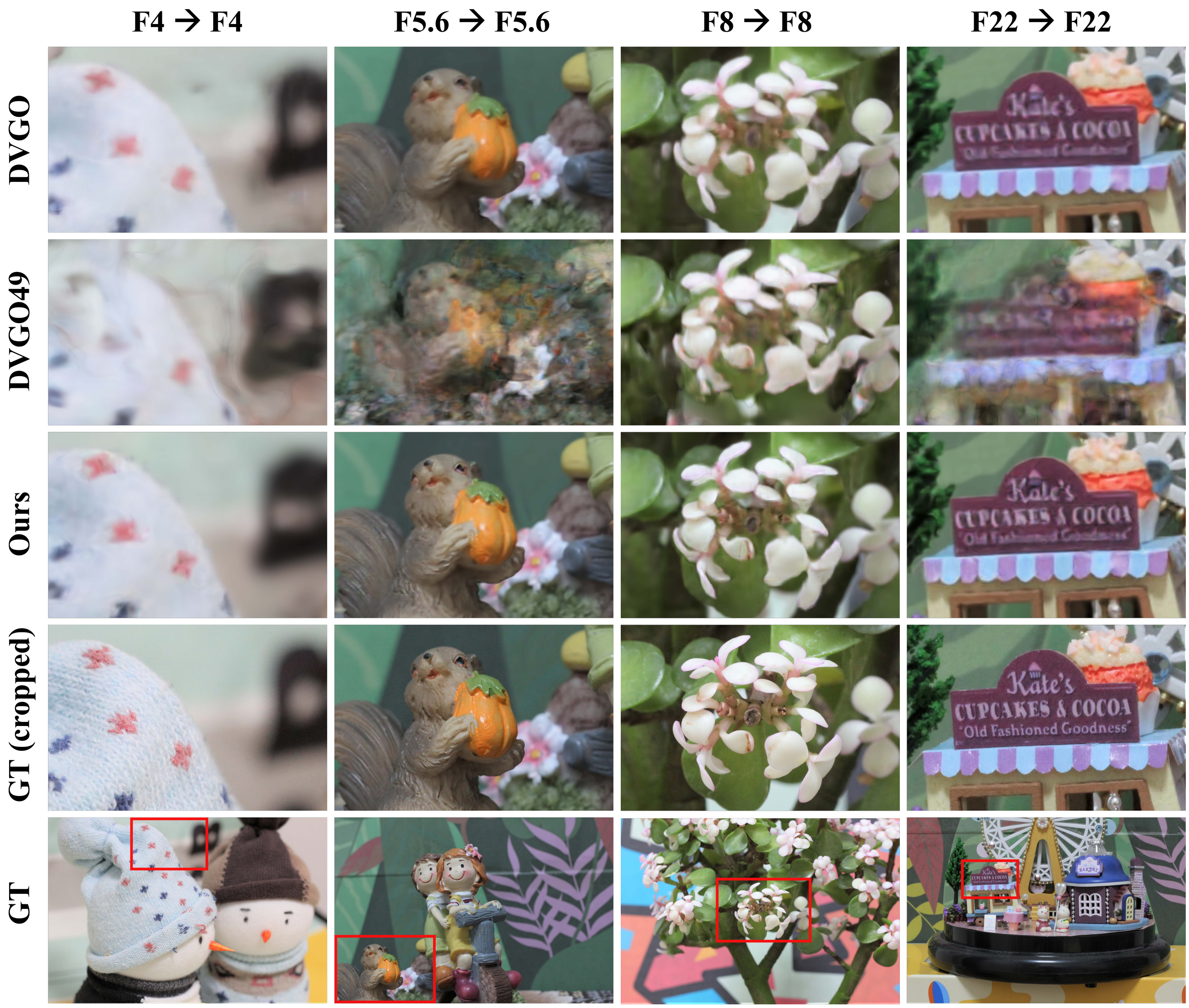

Figure 12. Additional qualitative results on novel view synthesis for the classic task. The red box region on the ground-truth(GT) image is cropped and zoomed for ease of comparison.

Scene	Method	F22 → F4			F22 → F5.6		
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
AmusementPark	DiskBlur + DVGO [7]	27.6260	0.8884	0.2101	28.0447	0.8911	0.2000
	NeRFocus [8]	25.7578	0.7638	0.2444	26.3220	0.7837	0.2118
	LensNeRF-D (ours)	**30.0711**	**0.9379**	**0.1363**	**30.4434**	**0.9361**	**0.1305**
	LensNeRF-M (ours)	<u>29.3756</u>	<u>0.9353</u>	<u>0.1413</u>	<u>29.6920</u>	<u>0.9331</u>	<u>0.1368</u>
	LensNeRF-S (ours)	29.0534	0.9257	0.1628	29.2871	0.9270	0.1480
AppleMint	DiskBlur + DVGO [7]	32.1852	0.9252	0.1938	33.1923	0.9185	0.1906
	NeRFocus [8]	29.2863	0.8281	0.2987	30.6599	0.8454	0.2361
	LensNeRF-D (ours)	<u>32.7614</u>	**0.9453**	**0.1574**	<u>33.9252</u>	**0.9396**	**0.1511**
	LensNeRF-M (ours)	**33.2451**	<u>0.9442</u>	<u>0.1587</u>	**34.1561**	<u>0.9388</u>	<u>0.1525</u>
	LensNeRF-S (ours)	32.1449	0.9371	0.1661	33.0752	0.9341	0.1531
Bear	DiskBlur + DVGO [7]	<u>29.8194</u>	0.8576	0.3578	<u>29.4636</u>	0.8334	0.3809
	NeRFocus [8]	27.1469	0.7474	0.3940	27.3402	0.7369	0.4039
	LensNeRF-D (ours)	**29.8708**	**0.8748**	**0.3163**	**30.0832**	**0.8538**	**0.3392**
	LensNeRF-M (ours)	28.5379	<u>0.8716</u>	<u>0.3209</u>	28.5975	<u>0.8500</u>	<u>0.3451</u>
	LensNeRF-S (ours)	27.3534	0.8639	0.3245	27.5374	0.8435	0.3493
BoyAndGirl	DiskBlur + DVGO [7]	**31.0509**	0.9264	0.1890	**32.0322**	0.9237	0.1763
	NeRFocus [8]	27.3666	0.7853	0.2398	27.9133	0.7986	0.2025
	LensNeRF-D (ours)	27.6177	<u>0.9390</u>	**0.1641**	28.7545	<u>0.9404</u>	**0.1463**
	LensNeRF-M (ours)	<u>29.1614</u>	**0.9441**	<u>0.1649</u>	<u>30.5447</u>	**0.9456**	<u>0.1470</u>
	LensNeRF-S (ours)	28.3230	0.9298	0.1837	29.4800	0.9360	0.1591
Chrysanthemum	DiskBlur + DVGO [7]	29.1768	0.8710	0.2408	29.3470	0.8597	0.2343
	NeRFocus [8]	27.5369	0.7920	0.2759	27.6831	0.7858	0.2677
	LensNeRF-D (ours)	**30.5801**	**0.9083**	**0.1728**	**29.9209**	**0.8938**	**0.1751**
	LensNeRF-M (ours)	<u>30.5620</u>	<u>0.9077</u>	<u>0.1742</u>	<u>29.8963</u>	<u>0.8935</u>	<u>0.1783</u>
	LensNeRF-S (ours)	29.5744	0.8969	0.1963	28.8580	0.8841	0.1906
Gink	DiskBlur + DVGO [7]	26.1505	0.8874	0.2051	26.8244	0.8917	0.1853
	NeRFocus [8]	21.7560	0.6720	0.3159	21.7547	0.6785	0.3128
	LensNeRF-D (ours)	**27.6488**	**0.9200**	**0.1480**	**28.1529**	**0.9185**	**0.1397**
	LensNeRF-M (ours)	<u>26.8747</u>	<u>0.9140</u>	<u>0.1520</u>	<u>27.1831</u>	<u>0.9126</u>	<u>0.1445</u>
	LensNeRF-S (ours)	25.9160	0.9031	0.1695	26.0182	0.9024	0.1551
Sheep	DiskBlur + DVGO [7]	27.1917	0.8655	0.4081	28.0124	0.8621	0.3989
	NeRFocus [8]	23.4382	0.6587	0.4295	24.2760	0.6718	0.4134
	LensNeRF-D (ours)	**28.9670**	**0.9027**	**0.3211**	**28.5760**	**0.8932**	**0.3252**
	LensNeRF-M (ours)	28.0896	<u>0.8999</u>	<u>0.3222</u>	27.8388	<u>0.8906</u>	<u>0.3284</u>
	LensNeRF-S (ours)	<u>28.7499</u>	0.8922	0.3483	<u>28.5063</u>	0.8867	0.3436
Snowman	DiskBlur + DVGO [7]	22.0553	0.7512	0.4681	22.6376	0.7320	0.4776
	NeRFocus [8]	21.0261	0.6123	0.4364	20.9126	0.5968	0.4554
	LensNeRF-D (ours)	**24.1898**	**0.7896**	**0.3903**	**23.9147**	<u>0.7660</u>	**0.4114**
	LensNeRF-M (ours)	<u>23.9007</u>	<u>0.7850</u>	<u>0.3978</u>	23.6968	**0.7614**	<u>0.4181</u>
	LensNeRF-S (ours)	23.8299	0.7765	0.4127	<u>23.7048</u>	0.7547	0.4249
Xmas	DiskBlur + DVGO [7]	24.1016	0.7919	0.3782	23.6838	0.7713	0.3693
	NeRFocus [8]	20.7413	0.5504	0.4794	20.5110	0.5431	0.4583
	LensNeRF-D (ours)	<u>25.1060</u>	<u>0.8415</u>	<u>0.3036</u>	<u>24.5831</u>	<u>0.8223</u>	<u>0.3008</u>
	LensNeRF-M (ours)	**25.3713**	**0.8446**	**0.2993**	**24.7528**	**0.8261**	**0.2980**
	LensNeRF-S (ours)	25.0375	0.8288	0.3412	24.4739	0.8150	0.3148

Table 8. Quantitative result for each scene on defocus task. The F-number of the train images is denoted on the left side of the arrow, and the F-number of the target images is indicated on the right. The **best** results are in boldface, and the <u>second-best</u> results are underlined.

Scene	Method	F22 → F8			Fmix → F4		
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
AmusementPark	DiskBlur + DVGO [7]	28.3758	0.8919	0.1986	N/A	N/A	N/A
	NeRFocus [8]	26.5634	0.7917	0.1976	N/A	N/A	N/A
	LensNeRF-D (ours)	**30.5259**	**0.9309**	**0.1359**	**30.5696**	**0.9360**	**0.1394**
	LensNeRF-M (ours)	<u>29.7644</u>	<u>0.9274</u>	<u>0.1408</u>	<u>30.2374</u>	<u>0.9328</u>	<u>0.1505</u>
	LensNeRF-S (ours)	29.3875	0.9230	0.1490	29.0247	0.9276	0.1582
AppleMint	DiskBlur + DVGO [7]	33.7195	0.9108	0.1950	N/A	N/A	N/A
	NeRFocus [8]	31.7548	0.8590	0.2073	N/A	N/A	N/A
	LensNeRF-D (ours)	**34.3519**	**0.9324**	**0.1545**	**34.4145**	**0.9435**	<u>0.1682</u>
	LensNeRF-M (ours)	<u>34.2048</u>	<u>0.9312</u>	<u>0.1549</u>	<u>33.5465</u>	<u>0.9421</u>	0.1702
	LensNeRF-S (ours)	33.3927	0.9277	0.1573	33.3383	0.9382	**0.1675**
Bear	DiskBlur + DVGO [7]	<u>29.5772</u>	0.8162	0.4027	N/A	N/A	N/A
	NeRFocus [8]	27.4255	0.7243	0.4207	N/A	N/A	N/A
	LensNeRF-D (ours)	**30.0923**	**0.8359**	**0.3613**	**30.3099**	**0.8739**	0.3211
	LensNeRF-M (ours)	28.5800	<u>0.8319</u>	<u>0.3677</u>	<u>29.2640</u>	<u>0.8731</u>	**0.3192**
	LensNeRF-S (ours)	27.5922	0.8260	0.3740	28.2299	0.8674	<u>0.3200</u>
BoyAndGirl	DiskBlur + DVGO [7]	**31.9002**	0.9195	0.1763	N/A	N/A	N/A
	NeRFocus [8]	28.1056	0.8028	0.1842	N/A	N/A	N/A
	LensNeRF-D (ours)	28.6092	<u>0.9362</u>	<u>0.1437</u>	**31.7020**	**0.9486**	**0.1494**
	LensNeRF-M (ours)	<u>30.3865</u>	**0.9414**	**0.1421**	<u>31.0127</u>	<u>0.9482</u>	<u>0.1513</u>
	LensNeRF-S (ours)	29.3030	0.9344	0.1550	28.5908	0.9408	0.1541
Chrysanthemum	DiskBlur + DVGO [7]	28.8865	0.8461	0.2357	N/A	N/A	N/A
	NeRFocus [8]	27.2675	0.7715	0.2680	N/A	N/A	N/A
	LensNeRF-D (ours)	**30.0562**	**0.8801**	**0.1807**	**31.0876**	**0.9058**	**0.1762**
	LensNeRF-M (ours)	<u>29.9087</u>	<u>0.8791</u>	<u>0.1851</u>	30.9844	<u>0.9046</u>	<u>0.1765</u>
	LensNeRF-S (ours)	29.0372	0.8708	0.1940	<u>31.0812</u>	0.8982	0.1915
Gink	DiskBlur + DVGO [7]	27.1617	0.8880	0.1809	N/A	N/A	N/A
	NeRFocus [8]	21.6741	0.6826	0.3193	N/A	N/A	N/A
	LensNeRF-D (ours)	**28.3778**	**0.9121**	**0.1430**	**28.0766**	**0.9264**	**0.1403**
	LensNeRF-M (ours)	<u>27.3231</u>	<u>0.9050</u>	<u>0.1487</u>	<u>27.3630</u>	<u>0.9225</u>	<u>0.1419</u>
	LensNeRF-S (ours)	26.1150	0.8952	0.1582	27.2124	0.9134	0.1526
Sheep	DiskBlur + DVGO [7]	**28.3121**	0.8598	0.3905	N/A	N/A	N/A
	NeRFocus [8]	24.7532	0.6796	0.4012	N/A	N/A	N/A
	LensNeRF-D (ours)	<u>28.3012</u>	**0.8821**	**0.3313**	**29.2493**	**0.9009**	**0.3278**
	LensNeRF-M (ours)	27.5591	<u>0.8800</u>	<u>0.3376</u>	28.6573	<u>0.9005</u>	<u>0.3283</u>
	LensNeRF-S (ours)	28.2099	0.8780	0.3504	<u>29.0857</u>	0.8947	0.3375
Snowman	DiskBlur + DVGO [7]	23.1258	0.7155	0.4869	N/A	N/A	N/A
	NeRFocus [8]	20.7424	0.5766	0.4746	N/A	N/A	N/A
	LensNeRF-D (ours)	**23.8557**	**0.7445**	**0.4305**	**24.5845**	**0.7937**	**0.3708**
	LensNeRF-M (ours)	23.6008	<u>0.7390</u>	<u>0.4385</u>	24.1103	<u>0.7899</u>	<u>0.3800</u>
	LensNeRF-S (ours)	<u>23.6206</u>	0.7343	0.4387	<u>24.1708</u>	0.7855	0.3907
Xmas	DiskBlur + DVGO [7]	23.2305	0.7460	0.3722	N/A	N/A	N/A
	NeRFocus [8]	20.2799	0.5312	0.4479	N/A	N/A	N/A
	LensNeRF-D (ours)	<u>24.0493</u>	<u>0.7985</u>	**0.3084**	25.4464	**0.8476**	**0.2954**
	LensNeRF-M (ours)	**24.1652**	**0.8003**	<u>0.3109</u>	<u>25.5025</u>	<u>0.8458</u>	<u>0.2957</u>
	LensNeRF-S (ours)	23.8400	0.7898	0.3198	**25.5650**	0.8395	0.3154

Table 9. Quantitative result for each scene on defocus task. The F-number of the train images is denoted on the left side of the arrow, and the F-number of the target images is indicated on the right. The **best** results are in boldface, and the <u>second-best</u> results are underlined.

Scene	Method	F4 → F22				F5.6 → F22			
		PSNR ↑	SSIM ↑	LPIPS ↓	DISTS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	DISTS ↓
AmusementPark	KPAC [5] + DVGO [7]	**28.3739**	0.8679	0.2421	0.1201	29.2216	0.8834	0.2123	0.1047
	Deblur-NeRF [2]	24.0320	0.7196	0.2086	0.1100	23.8261	0.7236	0.1862	0.0944
	LensNeRF-D (ours)	28.3535	**0.8841**	**0.1866**	**0.0897**	29.0369	**0.8952**	**0.1724**	0.0845
	LensNeRF-M (ours)	27.6138	0.8772	0.1916	0.0929	**29.2290**	0.8926	0.1767	**0.0837**
	LensNeRF-S (ours)	28.2380	0.8784	0.2045	0.0977	28.9028	0.8906	0.1848	0.0873
AppleMint	KPAC [5] + DVGO [7]	**32.3640**	**0.8824**	0.2687	0.1581	**32.8811**	**0.8946**	0.2334	0.1364
	Deblur-NeRF [2]	26.4975	0.7252	0.2567	**0.1384**	26.7817	0.7277	0.2174	**0.1200**
	LensNeRF-D (ours)	29.0286	0.8785	**0.2277**	0.1450	30.2464	0.8922	**0.2093**	0.1333
	LensNeRF-M (ours)	28.0724	0.8738	0.2361	0.1481	30.0794	0.8916	0.2108	0.1328
	LensNeRF-S (ours)	27.5809	0.8648	0.2484	0.1512	29.6704	0.8844	0.2168	0.1348
Bear	KPAC [5] + DVGO [7]	28.7261	0.7888	0.4395	0.2015	28.8906	0.7945	0.4292	0.1898
	Deblur-NeRF [2]	26.0364	0.6631	0.4286	0.1904	26.3210	0.6716	0.4089	**0.1696**
	LensNeRF-D (ours)	**29.4325**	0.8017	**0.4129**	0.1845	**29.4990**	**0.8078**	**0.3965**	0.1737
	LensNeRF-M (ours)	29.3252	**0.8021**	0.4140	**0.1841**	29.4810	0.8077	0.4031	0.1770
	LensNeRF-S (ours)	28.9853	0.7970	0.4198	0.1851	29.0484	0.8025	0.4072	0.1798
BoyAndGirl	KPAC [5] + DVGO [7]	30.2865	0.8592	0.2784	0.1162	30.6799	0.8716	0.2539	0.1027
	Deblur-NeRF [2]	26.2927	0.7360	0.2537	0.0968	27.8590	0.7797	0.2131	0.0820
	LensNeRF-D (ours)	**30.3920**	**0.8775**	**0.2252**	**0.0859**	**31.5603**	0.8885	**0.2102**	0.0813
	LensNeRF-M (ours)	29.7411	0.8757	**0.2252**	0.0884	31.3324	**0.8889**	0.2117	**0.0810**
	LensNeRF-S (ours)	29.7547	0.8704	0.2387	0.0915	30.6683	0.8835	0.2203	0.0843
Chrysanthemum	KPAC [5] + DVGO [7]	**26.8967**	0.7818	0.3391	0.2079	**27.9647**	0.8020	0.2910	0.1746
	Deblur-NeRF [2]	23.6715	0.6611	0.3108	0.1927	19.6084	0.5441	0.3438	0.1754
	LensNeRF-D (ours)	26.8733	**0.7932**	**0.2756**	**0.1696**	27.2173	**0.8075**	**0.2469**	**0.1483**
	LensNeRF-M (ours)	25.2780	0.7853	0.2823	0.1772	27.5385	0.8074	0.2496	0.1539
	LensNeRF-S (ours)	26.8668	0.7820	0.3011	0.1865	27.1294	0.7969	0.2696	0.1660
Gink	KPAC [5] + DVGO [7]	**24.8969**	**0.8289**	0.2849	0.1712	**25.9055**	**0.8471**	0.2400	0.1512
	Deblur-NeRF [2]	21.1777	0.6350	0.2875	0.1621	21.6752	0.6583	0.2510	0.1487
	LensNeRF-D (ours)	24.4122	0.8225	**0.2616**	**0.1528**	25.4159	0.8410	**0.2251**	**0.1375**
	LensNeRF-M (ours)	24.0657	0.8153	0.2660	0.1588	25.6115	0.8407	0.2260	0.1377
	LensNeRF-S (ours)	23.7185	0.8060	0.2858	0.1639	25.0557	0.8288	0.2415	0.1470
Sheep	KPAC [5] + DVGO [7]	**25.9083**	0.8122	0.4259	0.2273	27.2270	0.8288	0.3945	0.1989
	Deblur-NeRF [2]	21.9172	0.5843	0.4206	0.1930	21.8571	0.5855	0.4026	0.1780
	LensNeRF-D (ours)	25.7275	**0.8276**	**0.3778**	**0.1734**	26.5171	**0.8352**	**0.3595**	**0.1647**
	LensNeRF-M (ours)	25.6815	0.8251	0.3840	0.1751	26.7543	0.8346	0.3638	0.1653
	LensNeRF-S (ours)	25.3943	0.8204	0.3904	0.1841	**27.2621**	0.8302	0.3714	0.1724
Snowman	KPAC [5] + DVGO [7]	**23.1944**	0.6828	0.5094	0.2477	**23.4589**	0.6878	0.5015	0.2445
	Deblur-NeRF [2]	22.1364	0.5918	**0.3631**	**0.1495**	21.0770	0.5548	**0.3589**	**0.1469**
	LensNeRF-D (ours)	22.6324	**0.6933**	0.4744	0.2262	22.8607	**0.6984**	0.4638	0.2243
	LensNeRF-M (ours)	22.5241	0.6917	0.4731	0.2292	22.2594	0.6927	0.4649	0.2241
	LensNeRF-S (ours)	22.2195	0.6858	0.4800	0.2335	21.9793	0.6873	0.4717	0.2309
Xmas	KPAC [5] + DVGO [7]	**21.3470**	**0.6245**	0.4978	0.2520	21.7410	0.6514	0.4419	0.2114
	Deblur-NeRF [2]	19.2838	0.4105	0.5063	0.3356	19.4209	0.4325	0.4602	0.3039
	LensNeRF-D (ours)	21.3229	0.6228	**0.4735**	**0.2343**	21.9174	**0.6612**	**0.4072**	**0.1944**
	LensNeRF-M (ours)	21.2269	0.6214	0.4810	0.2392	**21.9612**	0.6570	0.4153	0.1975
	LensNeRF-S (ours)	21.1753	0.6183	0.4892	0.2490	21.8146	0.6487	0.4280	0.2021

Table 10. Quantitative result for each scene on deblur task. The F-number of the train images is denoted on the left side of the arrow, and the F-number of the target images is indicated on the right. The **best** results are in boldface, and the second-best results are underlined.

Scene	Method	F8 → F22 PSNR ↑	SSIM ↑	LPIPS ↓	DISTS ↓	Fmix → F22 PSNR ↑	SSIM ↑	LPIPS ↓	DISTS ↓
AmusementPark	KPAC [5] + DVGO [7]	**29.5907**	0.8916	0.1920	0.0953	29.6560	0.8899	0.1984	0.0982
	Deblur-NeRF [2]	23.4794	0.7105	0.1651	0.0857	23.1667	0.7084	0.1651	0.0884
	LensNeRF-D (ours)	27.8131	**0.8962**	**0.1619**	**0.0804**	**29.6984**	**0.9032**	**0.1632**	**0.0812**
	LensNeRF-M (ours)	26.3508	0.8843	0.1710	0.0826	28.5575	0.8974	0.1804	0.0862
	LensNeRF-S (ours)	28.5217	0.8935	0.1728	0.0812	29.3999	0.8980	0.1728	0.0855
AppleMint	KPAC [5] + DVGO [7]	**33.0230**	0.9006	0.2136	0.1281	**33.6286**	0.8991	0.2191	0.1324
	Deblur-NeRF [2]	27.8957	0.7541	**0.1864**	**0.1088**	30.0880	0.8099	**0.1717**	**0.1064**
	LensNeRF-D (ours)	30.3849	0.8997	0.1961	0.1210	30.8337	0.9019	0.1958	0.1226
	LensNeRF-M (ours)	31.2816	**0.9029**	0.1941	0.1217	31.6540	**0.9028**	0.1971	0.1222
	LensNeRF-S (ours)	29.4309	0.8942	0.2015	0.1252	30.1835	0.8966	0.2021	0.1236
Bear	KPAC [5] + DVGO [7]	28.7286	0.7960	0.4212	0.1845	28.9707	0.7954	0.4245	0.1873
	Deblur-NeRF [2]	25.2712	0.6536	0.4143	**0.1647**	25.6789	0.6559	0.4155	**0.1673**
	LensNeRF-D (ours)	**30.4717**	**0.8119**	**0.3914**	0.1684	**30.7691**	**0.8104**	**0.3983**	0.1709
	LensNeRF-M (ours)	30.2048	0.8104	0.3946	0.1692	29.5646	0.8080	0.4002	0.1731
	LensNeRF-S (ours)	29.2393	0.8051	0.4020	0.1749	29.0679	0.8035	0.4050	0.1758
BoyAndGirl	KPAC [5] + DVGO [7]	30.9342	0.8801	0.2328	0.0921	31.1469	0.8787	0.2373	0.0965
	Deblur-NeRF [2]	24.4676	0.6797	0.2090	0.0787	23.4344	0.6377	0.2139	0.0832
	LensNeRF-D (ours)	**31.4174**	**0.8963**	**0.1956**	0.0770	**31.8167**	**0.8982**	**0.1938**	**0.0768**
	LensNeRF-M (ours)	30.8653	0.8955	0.1977	0.0760	31.7581	0.8969	0.2010	0.0780
	LensNeRF-S (ours)	31.0685	0.8920	0.1994	**0.0756**	30.4726	0.8914	0.2037	0.0803
Chrysanthemum	KPAC [5] + DVGO [7]	**28.2782**	0.8136	0.2559	0.1532	**28.5539**	0.8117	0.2656	0.1620
	Deblur-NeRF [2]	23.4855	0.6463	0.2458	0.1550	24.6327	0.6855	**0.2195**	0.1480
	LensNeRF-D (ours)	27.7245	0.8186	**0.2276**	0.1378	28.1865	**0.8191**	0.2276	**0.1379**
	LensNeRF-M (ours)	28.1649	**0.8195**	0.2294	**0.1376**	27.9069	0.8170	0.2302	0.1400
	LensNeRF-S (ours)	27.5404	0.8096	0.2435	0.1509	28.1871	0.8125	0.2371	0.1464
Gink	KPAC [5] + DVGO [7]	**26.6345**	**0.8579**	0.2126	0.1362	**26.6667**	0.8579	0.2199	0.1428
	Deblur-NeRF [2]	21.6700	0.6716	0.2185	0.1355	21.8077	0.6757	0.2021	0.1378
	LensNeRF-D (ours)	25.9566	0.8548	0.1999	0.1260	27.0457	**0.8642**	**0.1939**	**0.1266**
	LensNeRF-M (ours)	25.9322	0.8544	**0.1988**	**0.1237**	26.5467	0.8595	0.1967	0.1292
	LensNeRF-S (ours)	25.7988	0.8430	0.2127	0.1326	25.9147	0.8471	0.2044	0.1310
Sheep	KPAC [5] + DVGO [7]	**27.8938**	0.8386	0.3738	0.1774	**27.7670**	0.8387	0.3765	0.1799
	Deblur-NeRF [2]	21.6657	0.5796	0.3878	0.1554	21.2183	0.5726	0.3931	**0.1467**
	LensNeRF-D (ours)	27.5818	**0.8428**	**0.3444**	**0.1530**	27.2895	**0.8466**	**0.3450**	0.1486
	LensNeRF-M (ours)	27.7009	0.8420	0.3504	0.1564	26.2731	0.8438	0.3520	0.1534
	LensNeRF-S (ours)	27.6946	0.8384	0.3561	0.1610	26.4936	0.8400	0.3539	0.1573
Snowman	KPAC [5] + DVGO [7]	**23.3518**	0.6905	0.4958	0.2409	**23.3714**	0.6898	0.4966	0.2379
	Deblur-NeRF [2]	22.5869	0.6127	**0.3406**	**0.1390**	19.0811	0.4731	**0.3948**	**0.1520**
	LensNeRF-D (ours)	23.0384	**0.6998**	0.4591	0.2230	22.6467	**0.6999**	0.4572	0.2210
	LensNeRF-M (ours)	22.4972	0.6956	0.4682	0.2255	22.3046	0.6958	0.4647	0.2216
	LensNeRF-S (ours)	22.4932	0.6922	0.4681	0.2290	22.3106	0.6942	0.4641	0.2252
Xmas	KPAC [5] + DVGO [7]	22.0415	0.6715	0.4016	0.1786	22.0375	0.6708	0.4107	0.1829
	Deblur-NeRF [2]	19.4655	0.4512	0.4144	0.2658	19.5910	0.4559	0.4142	0.2632
	LensNeRF-D (ours)	22.3944	**0.6924**	**0.3589**	0.1620	**22.6566**	**0.7036**	**0.3476**	**0.1441**
	LensNeRF-M (ours)	**22.4682**	0.6903	0.3656	**0.1607**	22.5173	0.7010	0.3537	0.1488
	LensNeRF-S (ours)	22.3810	0.6847	0.3745	0.1682	22.4521	0.6949	0.3650	0.1541

Table 11. Quantitative result for each scene on deblur task. The F-number of the train images is denoted on the left side of the arrow, and the F-number of the target images is indicated on the right. The **best** results are in boldface, and the second-best results are underlined.

Scene	Method	PSNR ↑	SSIM ↑ F4 → F4	LPIPS ↓	PSNR ↑	SSIM ↑ F5.6 → F5.6	LPIPS ↓
AmusementPark	DVGO [7]	28.9877	0.8995	0.1943	29.3536	0.9044	0.1780
	LensNeRF-D (ours)	<u>30.2320</u>	<u>0.9316</u>	**0.1420**	<u>28.7988</u>	<u>0.9285</u>	**0.1377**
	LensNeRF-M (ours)	**30.5307**	**0.9329**	<u>0.1454</u>	**30.3660**	**0.9318**	<u>0.1413</u>
	LensNeRF-S (ours)	29.7200	0.9283	0.1544	28.5335	0.9246	0.1509
AppleMint	DVGO [7]	**36.1986**	0.9334	0.1761	**35.9373**	0.9270	0.1699
	LensNeRF-D (ours)	<u>34.7835</u>	**0.9401**	**0.1693**	34.7055	<u>0.9342</u>	**0.1632**
	LensNeRF-M (ours)	33.6269	<u>0.9380</u>	0.1728	<u>34.7507</u>	**0.9343**	0.1640
	LensNeRF-S (ours)	32.6642	0.9321	<u>0.1711</u>	33.7557	0.9296	<u>0.1636</u>
Bear	DVGO [7]	30.9147	0.8608	0.3384	30.6581	0.8409	0.3579
	LensNeRF-D (ours)	**32.4768**	**0.8767**	**0.3214**	**32.0588**	**0.8564**	**0.3373**
	LensNeRF-M (ours)	<u>31.6488</u>	<u>0.8763</u>	<u>0.3215</u>	<u>31.3990</u>	<u>0.8554</u>	<u>0.3416</u>
	LensNeRF-S (ours)	29.6380	0.8700	0.3232	30.1461	0.8508	0.3439
BoyAndGirl	DVGO [7]	33.0087	0.9291	0.1754	**33.0253**	0.9271	0.1614
	LensNeRF-D (ours)	33.9178	<u>0.9506</u>	**0.1447**	31.4763	<u>0.9459</u>	**0.1328**
	LensNeRF-M (ours)	**34.6582**	**0.9514**	<u>0.1460</u>	<u>32.0842</u>	**0.9481**	<u>0.1339</u>
	LensNeRF-S (ours)	<u>34.4109</u>	0.9478	0.1490	31.7902	0.9433	0.1414
Chrysanthemum	DVGO [7]	30.5700	0.8821	0.2181	<u>30.2862</u>	0.8712	0.2112
	LensNeRF-D (ours)	**31.5985**	**0.9022**	**0.1863**	30.2057	<u>0.8881</u>	**0.1865**
	LensNeRF-M (ours)	29.0257	<u>0.8969</u>	<u>0.1887</u>	**30.2883**	**0.8868**	<u>0.1893</u>
	LensNeRF-S (ours)	<u>30.3991</u>	0.8899	0.1988	29.9367	0.8785	0.2013
Gink	DVGO [7]	27.0991	0.8974	0.1845	27.6730	0.8997	0.1691
	LensNeRF-D (ours)	**28.4802**	**0.9248**	**0.1421**	**28.4980**	**0.9188**	**0.1413**
	LensNeRF-M (ours)	<u>28.0098</u>	<u>0.9214</u>	<u>0.1446</u>	<u>28.4343</u>	<u>0.9172</u>	<u>0.1436</u>
	LensNeRF-S (ours)	27.3524	0.9147	0.1547	27.1320	0.9074	0.1556
Sheep	DVGO [7]	28.2883	0.8668	0.4004	28.8608	0.8649	0.3827
	LensNeRF-D (ours)	**29.3843**	**0.8969**	**0.3337**	29.1469	**0.8883**	**0.3346**
	LensNeRF-M (ours)	<u>29.3404</u>	<u>0.8948</u>	<u>0.3358</u>	<u>29.2367</u>	<u>0.8872</u>	<u>0.3370</u>
	LensNeRF-S (ours)	28.8323	0.8887	0.3480	**29.4681**	0.8815	0.3461
Snowman	DVGO [7]	23.1324	0.7656	0.4320	23.2311	0.7443	0.4507
	LensNeRF-D (ours)	**25.0261**	**0.7931**	**0.3813**	**24.3356**	**0.7686**	**0.4036**
	LensNeRF-M (ours)	<u>24.7963</u>	<u>0.7904</u>	<u>0.3817</u>	<u>23.7217</u>	<u>0.7630</u>	<u>0.4047</u>
	LensNeRF-S (ours)	24.4192	0.7846	0.3915	23.4501	0.7581	0.4116
Xmas	DVGO [7]	24.7823	0.8033	0.3511	24.1902	0.7834	0.3384
	LensNeRF-D (ours)	<u>25.1673</u>	**0.8401**	**0.3037**	**24.7233**	**0.8218**	**0.2979**
	LensNeRF-M (ours)	24.8679	0.8373	<u>0.3101</u>	<u>24.6237</u>	<u>0.8164</u>	<u>0.3054</u>
	LensNeRF-S (ours)	**25.3955**	<u>0.8374</u>	0.3167	24.3595	0.8072	0.3194

Table 12. Quantitative results for each scene when the train images and the test images have identical F-number. The F-number of the train images is denoted on the left side of the arrow, and the F-number of the target images is indicated on the right. The **best** results are in boldface, and the <u>second-best</u> results are underlined.

Scene	Method	PSNR ↑	SSIM ↑ F8 → F8	LPIPS ↓	PSNR ↑	SSIM ↑ F22 → F22	LPIPS ↓
AmusementPark	DVGO [7]	29.6689	0.9066	0.1682	30.0489	0.8994	0.1746
	LensNeRF-D (ours)	<u>29.9317</u>	**0.9262**	**0.1386**	**30.5479**	**0.9123**	**0.1480**
	LensNeRF-M (ours)	28.6188	0.9178	<u>0.1467</u>	29.7249	0.9073	<u>0.1524</u>
	LensNeRF-S (ours)	**30.3037**	<u>0.9225</u>	0.1523	<u>30.4876</u>	<u>0.9085</u>	0.1554
AppleMint	DVGO [7]	**35.1123**	0.9201	0.1692	**34.4317**	0.9060	0.1983
	LensNeRF-D (ours)	33.2631	<u>0.9265</u>	<u>0.1655</u>	<u>33.7055</u>	**0.9145**	**0.1788**
	LensNeRF-M (ours)	<u>33.8426</u>	**0.9280**	**0.1632**	32.9038	<u>0.9120</u>	<u>0.1809</u>
	LensNeRF-S (ours)	32.1758	0.9225	0.1682	32.6651	0.9087	0.1839
Bear	DVGO [7]	<u>30.3196</u>	0.8234	0.3748	<u>29.5623</u>	0.8031	0.4090
	LensNeRF-D (ours)	**30.8108**	**0.8364**	**0.3602**	**30.1386**	**0.8149**	**0.3894**
	LensNeRF-M (ours)	29.5448	<u>0.8345</u>	<u>0.3615</u>	28.5868	<u>0.8092</u>	<u>0.3977</u>
	LensNeRF-S (ours)	29.2065	0.8300	0.3680	28.3721	0.8049	0.4041
BoyAndGirl	DVGO [7]	<u>33.0075</u>	0.9246	0.1584	<u>31.7103</u>	0.8888	0.2114
	LensNeRF-D (ours)	**33.3078**	**0.9441**	<u>0.1336</u>	31.2961	<u>0.9054</u>	**0.1773**
	LensNeRF-M (ours)	31.8228	<u>0.9434</u>	**0.1335**	**32.4859**	**0.9076**	<u>0.1787</u>
	LensNeRF-S (ours)	31.6523	0.9394	0.1418	31.3252	0.9033	0.1801
Chrysanthemum	DVGO [7]	29.9292	0.8594	0.2080	**29.2169**	0.8250	0.2328
	LensNeRF-D (ours)	**30.3027**	**0.8744**	**0.1921**	27.5880	**0.8316**	**0.2032**
	LensNeRF-M (ours)	<u>30.2810</u>	<u>0.8735</u>	<u>0.1940</u>	<u>27.5973</u>	<u>0.8315</u>	<u>0.2086</u>
	LensNeRF-S (ours)	29.7214	0.8644	0.2073	26.7848	0.8226	0.2147
Gink	DVGO [7]	<u>27.8355</u>	0.8955	0.1632	<u>27.4917</u>	0.8691	0.1947
	LensNeRF-D (ours)	**27.9629**	**0.9081**	**0.1492**	**27.8150**	**0.8762**	**0.1744**
	LensNeRF-M (ours)	27.7914	<u>0.9073</u>	<u>0.1495</u>	27.1873	<u>0.8698</u>	<u>0.1791</u>
	LensNeRF-S (ours)	27.2479	0.8979	0.1639	26.2656	0.8587	0.1864
Sheep	DVGO [7]	**29.2461**	0.8640	0.3705	**28.8904**	**0.8524**	0.3446
	LensNeRF-D (ours)	28.4339	**0.8778**	**0.3390**	<u>26.4151</u>	<u>0.8501</u>	**0.3252**
	LensNeRF-M (ours)	<u>28.9944</u>	<u>0.8775</u>	<u>0.3450</u>	26.1083	0.8479	<u>0.3323</u>
	LensNeRF-S (ours)	28.7054	0.8736	0.3533	26.6508	0.8470	0.3356
Snowman	DVGO [7]	23.4141	0.7246	0.4618	**23.8170**	0.6955	0.4816
	LensNeRF-D (ours)	**24.1798**	**0.7449**	**0.4262**	<u>23.0085</u>	**0.7030**	**0.4558**
	LensNeRF-M (ours)	<u>23.5818</u>	<u>0.7409</u>	0.4327	22.8233	<u>0.6972</u>	<u>0.4624</u>
	LensNeRF-S (ours)	23.5794	0.7376	<u>0.4297</u>	22.9582	0.6924	0.4625
Xmas	DVGO [7]	23.6573	0.7619	0.3342	22.4655	0.7004	0.3683
	LensNeRF-D (ours)	**24.2313**	**0.7971**	**0.3079**	**22.7910**	<u>0.7209</u>	**0.3276**
	LensNeRF-M (ours)	<u>24.1292</u>	<u>0.7934</u>	<u>0.3119</u>	<u>22.7370</u>	**0.7215**	<u>0.3300</u>
	LensNeRF-S (ours)	24.0314	0.7869	0.3244	22.6395	0.7128	0.3419

Table 13. Quantitative results for each scene when the train images and the test images have identical F-number. The F-number of the train images is denoted on the left side of the arrow, and the F-number of the target images is indicated on the right. The **best** results are in boldface, and the <u>second-best</u> results are underlined.

References

[1] The shortest distance between two rays in 3d. https://math.stackexchange.com/questions/1036959/midpoint-of-the-shortest-distance-between-2-rays-in-3d. 1

[2] Li Ma, Xiaoyu Li, Jing Liao, Qi Zhang, Xuan Wang, Jue Wang, and Pedro V. Sander. Deblur-nerf: Neural radiance fields from blurry images. *arXiv preprint arXiv:2111.14292*, 2021. 4, 11, 12

[3] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul P. Srinivasan, and Jonathan T. Barron. NeRF in the dark: High dynamic range view synthesis from noisy raw images. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2022. 1

[4] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020. 1

[5] Hyeongseok Son, Junyong Lee, Sunghyun Cho, and Seungyong Lee. Single image defocus deblurring using kernel-sharing parallel atrous convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2642–2650, 2021. 11, 12

[6] Pratul P Srinivasan, Rahul Garg, Neal Wadhwa, Ren Ng, and Jonathan T Barron. Aperture supervision for monocular depth estimation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 1

[7] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2022. 9, 10, 11, 12, 13, 14

[8] Yinhuai Wang, Shuzhou Yang, Yujie Hu, and Jian Zhang. Nerfocus: Neural radiance field for 3d synthetic defocus. *arXiv preprint arXiv:2203.05189*, 2022. 2, 4, 9, 10

[9] Zijin Wu, Xingyi Li, Juewen Peng, Hao Lu, Zhiguo Cao, and Weicai Zhong. Dof-nerf: Depth-of-field meets neural radiance fields. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 1718–1729, 2022. 2, 4