Exploiting Deblurring Networks for Radiance Fields

Supplementary Material

S1. Overview

In this supplementary material, we provide additional implementation details of DeepDeblurRF-P and DeepDeblurRF-G, details on generating the BlurRF-Synth dataset, and additional experimental results. Specifically, we provide:

- Additional implementation details
- Blender rendering configurations
- Blender license information
- Ablation study on DeepDeblurRF-P
- Description for supplementary video
- Additional comparison results

S2. Additional Implementation Details

With iteration number of $N = 5$, DeepDeblurRF consists of one single-image deblurring network and four RF-guided deblurring networks for each blur type: camera motion blur and defocus blur. These networks are denoted as $\mathfrak{D}_s$, $\mathfrak{D}_g^1$, $\mathfrak{D}_g^2$, $\mathfrak{D}_g^3$, and $\mathfrak{D}_g^4$, respectively. The deblurred outputs from each network are used as inputs in each radiance field construction step to train $\mathbf{V}^{i\in\{1,\dots,5\}}$, respectively.

DeepDeblurRF-P and DeepDeblurRF-G share only the single-image deblurring network, $\mathfrak{D}_s$, which adopts the network of NAFNet-32 [1]. The RF-guided deblurring networks are trained independently for each of DeepDeblurRF-P and DeepDeblurRF-G to fully reflect the rendering results of Plenoxels [2] and 3D Gaussian splatting (3DGS) [3]. For camera motion blur, $\mathfrak{D}_s$ is trained for 223 epochs with a batch size of 32, while for defocus blur, it is trained for 53 epochs with the same batch size. The RF-guided deblurring networks use NAFNet-64 with a batch size of 32 and are trained separately for DeepDeblurRF-P and DeepDeblurRF-G.

S2.1. DeepDeblurRF-P

For RF-guided deblurring networks, $\mathfrak{D}_g^1$, $\mathfrak{D}_g^2$, $\mathfrak{D}_g^3$, and $\mathfrak{D}_g^4$ are trained for 25, 53, 43, and 20 epochs for camera motion blur, and 25, 14, 25, and 11 epochs for defocus blur. The networks $\mathfrak{D}_g^1$, $\mathfrak{D}_g^2$, and $\mathfrak{D}_g^3$ use the PSNR loss [1], while $\mathfrak{D}_g^4$ additionally incorporates the perceptual loss from Real-ESRGAN [12] to enhance deblurring quality.

For radiance field construction, we adopt the forward-facing experimental setting of Plenoxels [2]. They begin with a voxel resolution of $256 \times 256 \times 128$, with upsampling performed every 38,400 steps, progressing to resolutions of $512 \times 512 \times 128$ and finally reaching $1408 \times 1156 \times 128$. In our implementation, for $\mathbf{V}^1$ to $\mathbf{V}^4$, we adjust this scheme to start at a lower resolution of $128 \times 128 \times 128$, increasing to $256 \times 256 \times 256$, and then $512 \times 512 \times 256$. This adjustment accelerates training while progressively capturing finer details. However, for the final radiance field construction step, $\mathbf{V}^5$, we revert to the original settings to achieve optimal novel-view synthesis performance.

We utilize the default settings for the remaining hyperparameters. Specifically, learning rates are set to 3×10^1 for sigma and 1×10^{-2} for spherical harmonics, with total variation regularization values of 5×10^{-4} for density and 5×10^{-3} for spherical harmonics, along with a sparsity regularization of 1×10^{-12}. We optimize $\mathbf{V}^1$ for 51,200 iterations and $\mathbf{V}^2$ for 76,800 iterations, each with a batch size of 5k rays. For $\mathbf{V}^3$, $\mathbf{V}^4$, and $\mathbf{V}^5$, we extend optimization to 102,400 iterations with the same batch size.

S2.2. DeepDeblurRF-G

The RF-guided deblurring networks, $\mathfrak{D}_g^1$, $\mathfrak{D}_g^2$, $\mathfrak{D}_g^3$, and $\mathfrak{D}_g^4$, are trained for 53, 254, 254 and 33 epochs for camera motion blur, and 42, 42, 50 and 11 epochs for defocus blur. As with DeepDeblurRF-P, $\mathfrak{D}_g^1$, $\mathfrak{D}_g^2$, and $\mathfrak{D}_g^3$ use the PSNR loss [1], while $\mathfrak{D}_g^4$ also incorporates the perceptual loss from Real-ESRGAN [12] to enhance deblurring quality.

For the radiance field construction step of DeepDeblurRF-G, we employ the training scheme of the 3DGS [3]. We use the Adam optimizer and set the learning rate for the position of 3D Gaussians to 1.6×10^{-4}, the pruning threshold to 5×10^{-3}, and the densification threshold to 2×10^{-4}. However, since 3DGS assumes clean input images, adopting its training scheme in our framework may result in sparse and incomplete Gaussian point clouds. To address this, we incorporate the sparse point cloud compensation strategy of Deblur-GS [4] that performs depth-based pruning and adds extra Gaussian points. We use the hyperparameters proposed by Deblur-GS for the compensation strategy. We refer the readers to [4] for more details on the compensation strategy. We optimize $\mathbf{V}^1$ to $\mathbf{V}^4$ for 10,000 iterations each, and then extend the optimization for $\mathbf{V}^5$ to 20,000 iterations to capture finer scene details.

S2.3. Training Details

The deblurring networks are sequentially trained: train the initial deblurring network, construct RFs, render guidance images, and train the first RF-guided deblurring network. We present the pseudocode for the entire training procedure of DeepDeblurRF to provide a clearer understanding of the overall process.

Algorithm 1 DeepDeblurRF Training

Require: A training dataset containing N scenes $H = \{H_1, H_2, \ldots, H_N\}$. Each scene H_h consists of M blurred images $B = \{B_1, B_2, \ldots, B_M\}$ and their corresponding sharp images $S = \{S_1, S_2, \ldots, S_M\}$.
Ensure: A set of trained deblurring network weights:
- W_0: Initial deblurring network weight.
- $\{W_1, W_2, \ldots, W_T\}$: RF-guided deblurring network weights.

Initialize:
Initialize initial deblurring network weight W_0
Initialize RF-guided deblurring network weights $\{W_1, W_2, \ldots, W_T\}$
Initialize deblurred images $D_{h,i} \leftarrow \emptyset$ for all h, i
Initial deblurring network training:
for each scene $h = 1$ to N **do**
  for $i = 1$ to M **do**
    $D_{h,i} \leftarrow \text{Deblur}(B_{h,i}, W_0)$ ▷ Deblur using W_0
    Compute loss L_0 using $S_{h,i}$ and $D_{h,i}$
    Update W_0 using optimizer with ∇L_0 ▷ Train initial deblurring network
  end for
end for
Radiance Field (RF) & **RF-guided Deblurring Network Training:**
for $t = 1$ to T **do** ▷ Iterate through each RF-guided deblurring network
  for each scene $h = 1$ to N **do**
    if $t = 1$ **then** ▷ Initial deblurring
      $D_{h,1}, D_{h,2}, \ldots, D_{h,M} \leftarrow \text{Deblur}(B_h, W_0)$
    else ▷ RF-guided deblurring
      $D_{h,1}, D_{h,2}, \ldots, D_{h,M} \leftarrow \text{Deblur}(B_h, R_h, W_{t-1})$
    end if
    Obtain camera poses P_h using COLMAP from D_h ▷ Estimate camera poses for RF training
    Train radiance field RF_h using D_h and P_h
    Render images $R_{h,1}, R_{h,2}, \ldots, R_{h,M}$ from RF_h at input view poses
  end for
  for each scene $h = 1$ to N **do** ▷ Train the t-th RF-guided deblurring network
    $D_h \leftarrow \text{Deblur}(B_h, R_h, W_t)$
    Compute loss L_t using S_h and D_h
    Update W_t using optimizer with ∇L_t
  end for
end for
Final deblurring network weights:
return W_0 and $\{W_1, W_2, \ldots, W_T\}$ ▷ Return trained network weights for deblurring and novel view synthesis

S3. Details on the BlurRF-Synth Dataset

In this section, we describe the details of the Blender rendering configurations and the licensing of Blender models used to generate the BlurRF-Synth dataset.

S3.1. Blender Rendering Configurations

We modify the Blender script provided by Deblur-NeRF [7] to render sharp and blurred images. To accurately model the real-world blur formation process, we configure the color management settings to use the 'RAW' color profile to render images in the linear sRGB space. Additionally, we set the `Look` setting to 'None' to avoid any contrast-related transformations, and compositing is disabled in the output settings to avoid non-linear transformations. We manually sample 29 camera poses for each model to capture different viewpoints, and for the test set, we include 5 additional poses to evaluate novel-view synthesis quality.

Examples are shown in Fig. S1. In each row, the left images show the 34 sampled camera poses, with blue lines representing the defined camera paths, white dots indicating the sampled camera positions, and orange numbers labeling some of the poses. The rectangle in each left image represents the current camera, corresponding to the rectangle framing the scene in the right image. The image within the rectangle on the right shows the view rendered from this

Figure S1. Examples of Blender models used for synthesizing the test sets of BlurRF-Synth.

camera position. The darker appearance is due to the RAW color profile used for rendering.

Camera motion blur To simulate camera shakes, we take each of the 29 original camera poses as the starting camera position ($\mathbf{P}_0$) and apply a perturbation to determine the final camera position ($\mathbf{P}_3$) during the exposure time. This perturbation is applied along a random direction vector ($\mathbf{D}$), with its magnitude constrained by predefined limits adjusted according to the scene's scale. We then interpolate between these camera positions using a Bézier curve to create smooth camera motion. The Bézier curve is defined as:

$$\mathbf{B}(t) = (1-t)^3\mathbf{P}_0 + 3(1-t)^2 t\mathbf{P}_1 + 3(1-t)t^2\mathbf{P}_2 + t^3\mathbf{P}_3,$$

where $\mathbf{P}_0$ represents the original camera pose, $\mathbf{P}_3$ is the perturbed position, and $\mathbf{P}_1$ and $\mathbf{P}_2$ are control points that shape the camera motion trajectory. The control points $\mathbf{P}_1$ and $\mathbf{P}_2$ are computed based on a normal direction to the vector $\mathbf{D}$, which represents the camera's movement direction. Specifically, we calculate a perpendicular normal vector to $\mathbf{D}$ and adjust the depth of $\mathbf{P}_1$ and $\mathbf{P}_2$ independently. These depths are randomized within predefined ranges proportional to the scene scale, introducing variation in the curve. This ensures that each control point depth varies, resulting in more realistic camera movements.

Using this method, we render 51 sharp images per camera pose. The 26th image, the temporally central frame, is used as the ground truth sharp image, while all 51 frames are combined to generate realistic blurred images following the RSBlur [10] pipeline. The full Blender script will be released to ensure reproducibility and to enable further experimentation.

Defocus blur To generate realistic defocus blur, we adjust the camera's depth of field (DoF) settings for each of the 29 camera poses. By varying the camera's aperture and focal distance, we simulate different levels of defocus blur. Specifically, we enable the depth of field effect and adjust the aperture using the `f-stop` value to control the amount of blur. A lower `f-stop` value opens the aperture wider, allowing more light and creating a shallower depth of field, which increases the blur effect. Conversely, a higher `f-stop` value narrows the aperture, resulting in a deeper depth of field and reducing the blur. For each camera pose, we randomly adjust the focal distance within a predefined range to simulate varying degrees of defocus blur across frames.

Additionally, to capture diverse bokeh shapes, we randomly set the number of aperture blades between 7 and 9 for each scene, affecting the shape of the bokeh effect in the rendered images. Unlike camera motion blur synthesis, which combines multiple frames, defocus blur synthesis renders a single image per camera pose by directly adjusting the DoF settings to generate blurred images. After rendering in linear sRGB space, we apply the RSBlur [10] pipeline to introduce shot noise and sensor read noise, as done in the camera motion blur process. The full Blender script will also be released to ensure reproducibility and to enable further experimentation.

Figure S2. Example view of scenes 'Classroom 1' and 'Classroom 2'. These scenes are rendered from the same Blender model but use different sets of camera poses.

S3.2. Blender Asset Licenses

We synthetically generated our dataset using Blender models. Specifically, we collected 95 models from Blendswap[1] under Creative Commons licenses and sourced an additional 5 models from the synthetic dataset used in Deblur-NeRF [7]. Tab. S4 and Tab. S5 provide detailed information on each model, including the author, license type, and download link. Moreover, we specify whether each model was used to simulate camera motion blur or defocus blur, as well as whether it was included in the training set or the test set. To distinguish between scenes generated from the same model but with different sets of camera poses, we append numerical identifiers to the scene names. Examples are shown in Fig. S2.

[1] blendswap.com/

Model	BlurRF-Real (Camera Motion Blur) PSNR (↑)	SSIM (↑)	LPIPS (↓)	Deblur-NeRF [7] (Camera Motion Blur) PSNR (↑)	SSIM (↑)	LPIPS (↓)	Deblur-NeRF [7] (Defocus Blur) PSNR (↑)	SSIM (↑)	LPIPS (↓)	Computation Time (Hr.)
MLP										
Deblur-NeRF [7]	21.31	0.3845	0.5512	25.62	0.7645	0.1825	23.34	0.7191	0.1191	31.33
BAD-NeRF [11]	17.75	0.2279	0.7529	17.37	0.3044	0.5806	-	-	-	24.68
DP-NeRF [5]	21.17	0.3789	0.5378	25.91	0.7549	0.1638	23.60	0.7278	0.1072	30.00*
Voxel grid										
ExBluRF [6]	18.93	0.3137	0.6702	22.97	0.6787	0.2222	-	-	-	8.92
PDRF-10 [8]	21.37	0.3774	0.5759	25.87	0.7685	0.2049	23.73	0.7356	0.1084	4.26
DeepDeblurRF-P	24.20	0.4833	0.4751	25.96	0.7922	0.1521	23.49	0.7479	0.1067	1.14
3D Gaussians										
Deblur-GS [4]	20.20	0.3461	0.5026	25.53	0.7870	0.1234	23.42	0.7367	0.1146	0.33
BAGS [9]	22.87	0.4333	0.4883	26.18	0.7985	0.1185	23.50	0.7469	0.1040	1.25
DeepDeblurRF-G	24.59	0.4895	0.4495	26.82	0.8141	0.1182	23.82	0.7484	0.0951	0.28

Table S1. Quantitative comparison of novel-view synthesis on BlurRF-Real and real-world scenes from Deblur-NeRF [7]. We highlight the best metrics and the second best metrics. As discussed in Sec. 5, training times were measured using the BlurRF-Synth camera motion blur test set. Note that due to its high memory demands, DP-NeRF [5] was trained with two GPUs, while all other models were trained on a single NVIDIA TITAN RTX GPU.

# Iter.	Camera motion PSNR	SSIM	LPIPS	Defocus PSNR	SSIM	LPIPS
$N = 1$	28.04	0.8068	0.1989	30.51	0.8660	0.1711
$N = 2$	28.99	0.8400	0.1697	31.03	0.8751	0.1538
$N = 3$	29.47	0.8576	0.1405	32.03	0.8964	0.1278
$N = 4$	29.75	0.8657	0.1247	32.31	0.9025	0.1055
$N = 5$	29.81	0.8668	0.1142	32.51	0.9058	0.0961
$N = 6$	29.85	0.8690	0.1124	32.74	0.9077	0.0947

Table S2. Quantitative results of ablation study on the number of iterations N for the test sets of BlurRF-Synth.

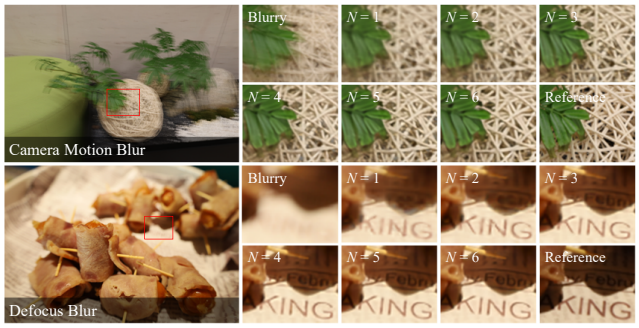

Figure S3. Qualitative results of ablation study on the number of iterations N for real-world scenes of the Deblur-NeRF [7].

S4. Additional Experimental Results

S4.1. Ablation Study on DeepDeblurRF-P

In Sec. 5, we presented both quantitative and qualitative results of the ablation study on the number of iterations N for DeepDeblurRF-G. In this section, we extend the same ablation study to DeepDeblurRF-P. Tab. S2 shows that the novel-view synthesis performance of DeepDeblurRF-P improves as N increases. While our models generally achieve optimal performance at $N = 5$, we observed that in some scenes, both DeepDeblurRF-P and DeepDeblurRF-G converge earlier, around $N = 3$ or $N = 4$. This suggests that although $N = 5$ is a robust setting for most cases, fewer iterations can achieve comparable results in specific scenes, as illustrated in Fig. S3.

S4.2. Description for Supplementary Video

Our supplementary video presents novel-view synthesis results across various datasets, including comparisons with other models. For ray-based methods [5, 8], we used the official code provided for video generation. For 3D Gaussians-based methods [4, 9], which did not provide such code, we generated novel views by randomly sampling and interpolating camera poses from the training views.

DeepDeblurRF performs high-quality novel-view synthesis on the BlurRF-Synth dataset and real-world datasets of the Deblur-NeRF [7], both of which contain scenes with camera motion blur and defocus blur.

On the BlurRF-Real dataset, which contains camera motion blur scenes under challenging low-light conditions, DeepDeblurRF also outperforms other methods, which produce noisy or blurred results.

Furthermore, other methods fail to effectively remove blur in the BlurRF-SB dataset, where all training views contain camera motion blur in the same direction but with varying intensities. This is due to the reliance of previous approaches on the assumption that input blurred images contain complementary information from different blur directions, which does not hold in the BlurRF-SB dataset. In contrast, our methods produce sharp and clear novel views, demonstrating the advantage of leveraging prior knowledge on sharp images obtained from pre-trained deblurring networks.

S4.3. Additional Comparison Results

We report experimental results on the benchmark Deblur-NeRF [7] datasets. Notably, benchmark datasets do not reflect challenging low-light conditions where blur often occurs. Instead, they contain minimal noises, thus are highly favorable to previous methods that rely on linear blur models. For a comprehensive evaluation, we retrained our model with minimal noise. The results are reported in Tab. S1 and Tab. S3. As discussed in Sec. 5, the blur-free reference images for the real-world datasets are not fully consistent with the training views due to differences in calibration and exposure, making precise quantitative evaluation challenging. Therefore, we report the quantitative results in the supplementary material for further reference.

Tab. S1 presents the results, evaluated using full-reference quality metrics. On the BlurRF-Real dataset, both DeepDeblurRF-P and DeepDeblurRF-G outperform existing models across all metrics. This is further highlighted in the qualitative results shown in Fig. S4, where other methods struggle with real-world non-linear artifacts, such as noise in low-light conditions, while our models perform high-quality novel-view synthesis.

Similarly, on the Deblur-NeRF [7] datasets, our models exhibit superior performance with significantly shorter computation times. For camera motion blur, DeepDeblurRF-G achieves superior performance in both synthetic and real-world scenes, while requiring less computation time. For defocus blur, DeepDeblurRF-G demonstrates a notable advantage in the LPIPS metric, reflecting its superior perceptual quality. DeepDeblurRF-P also surpasses other MLP and Voxel grid-based models in both SSIM and LPIPS on real-world scenes, while also maintaining the fastest training time. These trends are also visible in the qualitative results shown in Fig. S5, Fig. S6, and Fig. S7. These figures also include additional qualitative results both on the benchmark synthetic dataset [7] and BlurRF-Synth dataset, omitted from the main paper due to space limits.

Additionally, we report quantitative evaluation results on test sets of the BlurRF-Synth dataset, which include both camera motion blur and defocus blur. Tab. S6 and Tab. S7 present the results for each type of blur, respectively. The tables include PSNR, SSIM, and LPIPS metrics for all models across each scene. Notably, DeepDeblurRF-P and DeepDeblurRF-G consistently outperform other models across most quantitative metrics.

Model	Camera motion			Defocus		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
Deblur-NeRF [7]	28.77	0.8593	0.1400	28.37	0.8527	0.1188
DP-NeRF [5]	29.23	0.8674	0.1184	29.33	0.8713	0.0987
PDRF-10 [8]	29.29	0.8798	0.1051	30.08	0.8931	0.1101
Deblurring-3DGS [4]	28.24	0.8580	0.1048	29.51	0.8977	0.1015
BAGS [9]	28.19	0.8516	0.1201	29.26	0.8983	0.1073
DeepDeblurRF-G	29.44	0.8811	0.0906	29.78	0.8992	0.0744

Table S3. Quantitative results of novel-view synthesis on the synthetic dataset of Deblur-NeRF [7]. We highlight the best metrics and the second best metrics.

References

[1] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. In *European conference on computer vision*, pages 17–33. Springer, 2022. 1

[2] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 1

[3] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023. 1

[4] Byeonghyeon Lee, Howoong Lee, Xiangyu Sun, Usman Ali, and Eunbyung Park. Deblurring 3d gaussian splatting, 2024. 1, 4, 5, 8

[5] Dogyoon Lee, Minhyeok Lee, Chajin Shin, and Sangyoun Lee. Dp-nerf: Deblurred neural radiance field with physical scene priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12386–12396, 2023. 4, 5, 8

[6] Dongwoo Lee, Jeongtaek Oh, Jaesung Rim, Sunghyun Cho, and Kyoung Mu Lee. Exblurf: Efficient radiance fields for extreme motion blurred images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17639–17648, 2023. 4, 8

[7] Li Ma, Xiaoyu Li, Jing Liao, Qi Zhang, Xuan Wang, Jue Wang, and Pedro V Sander. Deblur-nerf: Neural radiance fields from blurry images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12861–12870, 2022. 2, 3, 4, 5, 7, 8, 10, 11, 12

[8] Cheng Peng and Rama Chellappa. Pdrf: progressively deblurring radiance field for fast scene reconstruction from blurry images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2029–2037, 2023. 4, 5, 8

[9] Cheng Peng, Yutao Tang, Yifan Zhou, Nengyu Wang, Xijun Liu, Deming Li, and Rama Chellappa. Bags: Blur agnostic gaussian splatting through multi-scale kernel modeling, 2024. 4, 5, 8

[10] Jaesung Rim, Geonung Kim, Jungeon Kim, Junyong Lee, Seungyong Lee, and Sunghyun Cho. Realistic blur synthesis for learning image deblurring. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 3

[11] Peng Wang, Lingzhe Zhao, Ruijie Ma, and Peidong Liu. Bad-nerf: Bundle adjusted deblur neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4170–4179, 2023. 4, 8

[12] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *International Conference on Computer Vision Workshops (ICCVW)*, 2021. 1

Table S4. Blender license information with dataset usage

No.	Scene	Author	License	Link	Camera Motion		Defocus	
					Train	Test	Train	Test
1	airplane	fabien	CC-BY-3.0	https://blendswap.com/blend/15016	O		O	
2	antique	b2przemo	CC-BY-3.0	https://blendswap.com/blend/18909	O		O	
3	applepear	strapazie	CC-BY-3.0	https://blendswap.com/blend/19210	O		O	
4	arts	singroff	CC-BY-0	https://blendswap.com/blend/25860	O			
5	basketball	CGMasters	CC-BY-3.0	https://blendswap.com/blend/14758	O		O	
6	blenderroom 1	Warcos	CC-BY-SA-3.0	https://blendswap.com/blend/20975	O		O	
7	blenderroom 2	Warcos	CC-BY-SA-3.0	https://blendswap.com/blend/20975	O		O	
8	cafeteria 1	Dennis Mg	CC-BY-0	https://blendswap.com/blend/31447	O			
9	cafeteria 2	Dennis Mg	CC-BY-0	https://blendswap.com/blend/31447	O			
10	castle 1	3dfiles	CC-BY-0	https://blendswap.com/blend/31236	O			
11	castle 2	3dfiles	CC-BY-0	https://blendswap.com/blend/31236	O			
12	castle 3	3dfiles	CC-BY-0	https://blendswap.com/blend/31236	O			
13	classroom 1	SwastikDas	CC-BY-3.0	https://blendswap.com/blend/21410	O		O	
14	classroom 2	SwastikDas	CC-BY-3.0	https://blendswap.com/blend/21410	O		O	
15	deer	Spine69	CC-BY-0	https://blendswap.com/blend/26863	O			
16	desert	nacimus	CC-BY-0	https://blendswap.com/blend/26925	O		O	
17	designtable	gandre82	CC-BY-NC	https://blendswap.com/blend/26159	O		O	
18	diningroom 1	MaTTeSr	CC-BY-3.0	https://blendswap.com/blend/18762	O		O	
19	diningroom 2	MaTTeSr	CC-BY-3.0	https://blendswap.com/blend/18762	O			
20	drum	bryanajones	CC-BY-3.0	https://blendswap.com/blend/13383	O		O	
21	entrance	oldtimer	CC-BY-SA-3.0	https://blendswap.com/blend/13545		O	O	
22	fireplace	tkobyl2	CC-BY-SA	https://blendswap.com/blend/23941	O			
23	grasstool 1	kevinjohnwimberly	CC-BY-0	https://blendswap.com/blend/31101	O			
24	grasstool 2	kevinjohnwimberly	CC-BY-0	https://blendswap.com/blend/31101	O			
25	homelibrary 1	ThePefDispenser	CC-BY-3.0	https://blendswap.com/blend/19984		O	O	
26	homelibrary 2	ThePefDispenser	CC-BY-3.0	https://blendswap.com/blend/19984		O	O	
27	hotdog	erickfree	CC-BY-0	https://blendswap.com/blend/23962	O		O	
28	hotliving 1	oldtimer	CC-BY-SA-3.0	https://blendswap.com/blend/13707	O		O	
29	hotliving 2	oldtimer	CC-BY-SA-3.0	https://blendswap.com/blend/13707	O		O	
30	industry 1	levigibson	CC-BY-0	https://blendswap.com/blend/26527	O		O	
31	industry 2	levigibson	CC-BY-0	https://blendswap.com/blend/26527	O			
32	industry 3	levigibson	CC-BY-0	https://blendswap.com/blend/26527	O			
33	interior	oldtimer	CC-BY-3.0	https://blendswap.com/blend/11624		O		
34	japan	red0004	CC-BY-0	https://blendswap.com/blend/23230	O		O	
35	kitchen	Gorion	CC-BY-SA-3.0	https://blendswap.com/blend/10286	O		O	
36	kitchenrail	Sambor	CC-BY-0	https://blendswap.com/blend/31127	O			
37	LG	AzuritHD	CC-BY-0	https://blendswap.com/blend/24405	O			
38	livingroom	Rendars	CC-BY-0	https://blendswap.com/blend/17574	O		O	
39	lotus	Britdawgmasterfunk	CC-BY-0	https://blendswap.com/blend/31343		O	O	
40	market 1	tokabilitor	CC-BY-0	https://blendswap.com/blend/8498	O			
41	market 2	tokabilitor	CC-BY-0	https://blendswap.com/blend/8498	O			
42	markethouse	UncleAmi	CC-BY-0	https://blendswap.com/blend/30651	O		O	
43	military	holmen	CC-BY-SA-3.0	https://blendswap.com/blend/20349	O		O	
44	office	exedesign	CC-BY-3.0	https://blendswap.com/blend/5402	O		O	
45	oilvinegar	oldtimer	CC-BY-3.0	https://blendswap.com/blend/9549	O		O	
46	oldcar	thecali	CC-BY-0	https://blendswap.com/blend/13575	O		O	
47	oldphone	oldtimer	CC-BY-SA-3.0	https://blendswap.com/blend/12507	O		O	
48	oldroom	oldtimer	CC-BY-SA-3.0	https://blendswap.com/blend/12562	O		O	
49	oldtruck	oldtimer	CC-BY-3.0	https://blendswap.com/blend/10372	O		O	
50	payphone	Ndakasha	CC-BY-0	https://blendswap.com/blend/16649	O		O	

Table S5. Blender license information with dataset usage

No.	Scene	Author	License	Link	Camera Motion		Defocus	
					Train	Test	Train	Test
51	pinetree	Usch18	CC-BY-0	https://blendswap.com/blend/22602	O			
52	plantpot	MZiemys	CC-BY-1.0	https://blendswap.com/blend/17472	O		O	
53	portion	aXel	CC-BY-3.0	https://blendswap.com/blend/8909	O		O	
54	prison 1	shadows555	CC-BY-0	https://blendswap.com/blend/13423	O		O	
55	prison 2	shadows555	CC-BY-0	https://blendswap.com/blend/13423	O		O	
56	readingroom	oldtimer	CC-BY-3.0	https://blendswap.com/blend/11431	O		O	
57	redroom 1	Wig42	CC-BY-3.0	https://blendswap.com/blend/13491	O		O	
58	redroom 2	Wig42	CC-BY-3.0	https://blendswap.com/blend/13491	O		O	
59	roots	AceTop	CC-BY-0	https://blendswap.com/blend/24472	O			
60	snow	Kaluura	CC-BY-3.0	https://blendswap.com/blend/7892	O		O	
61	snowroom	akshdlfps	CC-BY-0	https://blendswap.com/blend/25057	O			
62	Stone	rajatg8008	CC-BY-0	https://blendswap.com/blend/22490	O		O	
63	sunnyroom	ermmus	CC-BY-3.0	https://blendswap.com/blend/6468	O		O	
64	suv	FA1RYDUST	CC-BY-0	https://blendswap.com/blend/29248	O		O	
65	threejugs	BigBadCat	CC-BY-0	https://blendswap.com/blend/23234	O			
66	towelrail	gianmariaveronese	CC-BY-3.0	https://blendswap.com/blend/16186	O		O	
67	toyfruit	darkst0ne	CC-BY-SA-3.0	https://blendswap.com/blend/17890	O		O	
68	Valley 1	ShadowCrystol	CC-BY-0	https://blendswap.com/blend/27325	O			
69	Valley 2	ShadowCrystol	CC-BY-0	https://blendswap.com/blend/27325	O			
70	woodfence	gianmariaveronese	CC-BY-3.0	https://blendswap.com/blend/16218	O			
71	aloe	dbar	CC-BY-0	https://blendswap.com/blend/29333			O	
72	axe	Mukhammad	CC-BY-0	https://blendswap.com/blend/28884			O	
73	brownbathroom	imperfection.png	CC-BY-0	https://blendswap.com/blend/29184			O	
74	cactus	Prokster	CC-BY-0	https://blendswap.com/blend/27983			O	
75	champagne	Teleport3d	CC-BY-3.0	https://blendswap.com/blend/28262			O	
76	chocolate	Bagoule	CC-BY-0	https://blendswap.com/blend/30805			O	
77	computer	Supper800	CC-BY-3.0	https://blendswap.com/blend/28505			O	
78	hobbyroom	muhuk	CC-BY-0	https://blendswap.com/blend/28537			O	
79	junglegym	DoiMoi	CC-BY-0	https://blendswap.com/blend/30371			O	
80	kitcheninterior	Mukhammad	CC-BY-3.0	https://blendswap.com/blend/28822			O	
81	oldoak	manoh	CC-BY-0	https://blendswap.com/blend/19222			O	
82	range	Ginibird	CC-BY-3.0	https://blendswap.com/blend/29247			O	
83	recyclebin	jamesstar	CC-BY-0	https://blendswap.com/blend/30905			O	
84	salon	carmule	CC-BY-0	https://blendswap.com/blend/30615			O	
85	samosas	Hope	CC-BY-0	https://blendswap.com/blend/28022			O	
86	scissors	salimrached	CC-BY-0	https://blendswap.com/blend/27712			O	
87	shaving	narmoo	CC-BY-3.0	https://blendswap.com/blend/30020			O	
88	teaset	SavageIndie	CC-BY-3.0	https://blendswap.com/blend/28729			O	
89	trophies	ScreamingOrange	CC-BY-0	https://blendswap.com/blend/27622			O	
90	victorianoffice	Teleport3d	CC-BY-3.0	https://blendswap.com/blend/28382			O	
91	garage	Bagoule	CC-BY-0	https://blendswap.com/blend/31126				O
92	key	raven0246	CC-BY-NC-3.0	https://blendswap.com/blend/20439				O
93	paintingroom	celoaz	CC-BY-1.0	https://blendswap.com/blend/17967				O
94	shotgun	Britdawgmasterfunk	CC-BY-0	https://blendswap.com/blend/30712				O
95	swimmingpool	BlenderByte	CC-BY-3.0	https://blendswap.com/blend/15015				O
96	cozyroom	Deblur-NeRF [7]				O		O
97	factory	Deblur-NeRF [7]				O		O
98	pool	Deblur-NeRF [7]				O		O
99	tanabata	Deblur-NeRF [7]				O		O
100	wine	Deblur-NeRF [7]				O		O

Total Blender Models: 100

Camera Motion Blur: Train (65), Test (10)

Defocus Blur: Train (65), Test (10)

Table S6. Quantitative results of novel-view synthesis on the camera motion blur test set of BlurRF-Synth. We highlight the best metrics and the second best metrics.

Model	Cozyroom			Factory			Pool			Tanabata			Wine		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Deblur-NeRF [7]	27.39	0.8577	0.1155	22.84	0.7648	0.2083	30.22	0.8206	0.1896	22.14	0.7764	0.1747	23.58	0.7967	0.1336
BAD-NeRF [11]	19.46	0.4487	0.5098	15.85	0.3379	0.5447	22.89	0.4483	0.5675	13.29	0.2532	0.6184	21.19	0.7062	0.1685
DP-NeRF [5]	27.59	0.8611	0.1021	23.14	0.7764	0.1897	30.51	0.8246	0.1683	22.72	0.7889	0.1517	24.01	0.8045	0.1187
ExBluRF [6]	25.97	0.8118	0.1398	20.01	0.6824	0.2577	29.05	0.7589	0.3370	20.88	0.7072	0.1991	20.78	0.7014	0.2101
PDRF-10 [8]	27.91	0.8607	0.1008	26.13	0.8298	0.1839	29.16	0.7960	0.2254	23.94	0.8191	0.1491	24.79	0.8271	0.1170
Deblur-GS [4]	27.60	0.8321	0.1059	21.54	0.6683	0.2261	28.26	0.7530	0.2339	21.83	0.7112	0.1859	22.67	0.7664	0.1251
BAGS [9]	27.69	0.8429	0.0896	22.89	0.7386	0.2045	28.33	0.7514	0.2016	23.82	0.7771	0.1587	24.34	0.8160	0.1021
DeepDeblurRF-P (Ours)	29.74	0.8732	0.0665	24.65	0.8128	0.1706	31.25	0.8389	0.1609	25.30	0.8450	0.1270	25.63	0.8532	0.1073
DeepDeblurRF-G (Ours)	30.02	0.8887	0.0728	24.87	0.8030	0.1599	31.04	0.8367	0.1468	25.81	0.8529	0.1043	25.66	0.8532	0.0887

Model	Entrance			Homelibrary 1			Interior			Lotus			Homelibrary 2		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Deblur-NeRF [7]	31.45	0.9167	0.0693	29.65	0.8241	0.1335	32.03	0.9086	0.1088	29.77	0.8356	0.1884	27.62	0.8383	0.1283
BAD-NeRF [11]	21.77	0.6295	0.2261	28.51	0.7672	0.1916	30.46	0.8641	0.1402	24.20	0.4690	0.4773	19.77	0.3739	0.5249
DP-NeRF [5]	31.90	0.9217	0.0585	30.04	0.8266	0.1153	32.18	0.9108	0.0960	30.33	0.8532	0.1613	27.83	0.8437	0.1057
ExBluRF [6]	31.24	0.8960	0.0846	29.41	0.7970	0.1553	31.84	0.8917	0.1165	29.29	0.8077	0.2559	27.09	0.7688	0.1986
PDRF-10 [8]	31.75	0.9143	0.0803	30.34	0.8318	0.1510	31.58	0.9043	0.1098	30.08	0.8325	0.2330	27.61	0.8193	0.1447
Deblur-GS [4]	28.93	0.8669	0.1005	27.17	0.7539	0.1761	29.70	0.8360	0.1660	28.51	0.7895	0.2561	26.82	0.7520	0.1527
BAGS [9]	30.81	0.9012	0.0676	29.66	0.8224	0.1097	30.52	0.8763	0.1220	29.03	0.8068	0.2081	27.04	0.7754	0.1182
DeepDeblurRF-P (Ours)	34.13	0.9428	0.0431	31.77	0.8495	0.1227	34.21	0.9441	0.0578	31.58	0.8626	0.1764	29.82	0.8462	0.1098
DeepDeblurRF-G (Ours)	32.52	0.9321	0.0542	32.20	0.8373	0.1378	35.70	0.9554	0.0516	31.40	0.8518	0.1631	30.19	0.8696	0.0795

Table S7. Quantitative results of novel-view synthesis on the defocus blur test set of BlurRF-Synth. We highlight the best metrics and the second best metrics.

Model	Cozyroom			Factory			Garage			Key			Paintingroom		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Deblur-NeRF [7]	29.77	0.8941	0.0744	25.69	0.8503	0.1289	31.84	0.8818	0.1183	33.08	0.9136	0.0653	34.98	0.9438	0.0403
BAD-NeRF [11]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DP-NeRF [5]	29.62	0.8911	0.0650	25.71	0.8538	0.1127	31.80	0.8819	0.1124	32.44	0.9024	0.0619	34.64	0.9410	0.0369
ExBluRF [6]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PDRF-10 [8]	29.06	0.8804	0.0783	27.92	0.8731	0.1218	31.77	0.8833	0.1208	31.99	0.8814	0.0895	33.05	0.9215	0.0785
Deblur-GS [4]	27.54	0.8542	0.1158	26.50	0.8584	0.1568	31.55	0.8532	0.1555	31.27	0.8555	0.1391	33.50	0.9262	0.0679
BAGS [9]	30.45	0.8852	0.0613	27.75	0.8881	0.1065	32.16	0.8839	0.1323	30.07	0.8081	0.1682	33.70	0.9312	0.0437
DeepDeblurRF-P (Ours)	31.47	0.9149	0.0552	29.33	0.9008	0.1019	33.90	0.9099	0.1037	36.85	0.9544	0.0505	38.19	0.9660	0.0246
DeepDeblurRF-G (Ours)	31.33	0.9117	0.0505	28.06	0.9088	0.0674	34.63	0.9162	0.0695	38.06	0.9600	0.0478	39.41	0.9720	0.0201

Model	Pool			Shotgun			Swimmingpool			Tanabata			Wine		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Deblur-NeRF [7]	31.10	0.8513	0.1582	34.80	0.9534	0.0661	32.53	0.8613	0.1583	23.19	0.7924	0.1614	23.30	0.7848	0.1661
BAD-NeRF [11]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DP-NeRF [5]	31.61	0.8627	0.1185	34.62	0.9521	0.0560	33.08	0.8672	0.1392	23.90	0.8098	0.1458	24.11	0.8014	0.1421
ExBluRF [6]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PDRF-10 [8]	30.14	0.8414	0.1711	33.69	0.9406	0.1099	32.71	0.8633	0.1710	25.05	0.8373	0.1414	24.93	0.8281	0.1428
Deblur-GS [4]	29.74	0.8219	0.1814	33.98	0.9435	0.0757	32.49	0.8510	0.2114	22.80	0.7776	0.2056	24.29	0.8036	0.1611
BAGS [9]	29.36	0.8217	0.1336	33.58	0.9407	0.0687	32.06	0.8559	0.1815	24.85	0.7998	0.1359	24.98	0.8231	0.1203
DeepDeblurRF-P (Ours)	33.01	0.8865	0.1317	37.26	0.9673	0.0418	34.79	0.8749	0.1759	25.22	0.8457	0.1353	25.03	0.8376	0.1406
DeepDeblurRF-G (Ours)	31.55	0.8641	0.1044	38.89	0.9727	0.0361	34.69	0.8762	0.1637	24.78	0.8487	0.0995	24.42	0.8294	0.1147

Figure S4. Qualitative comparison of novel-view synthesis on the BlurRF-Real dataset.

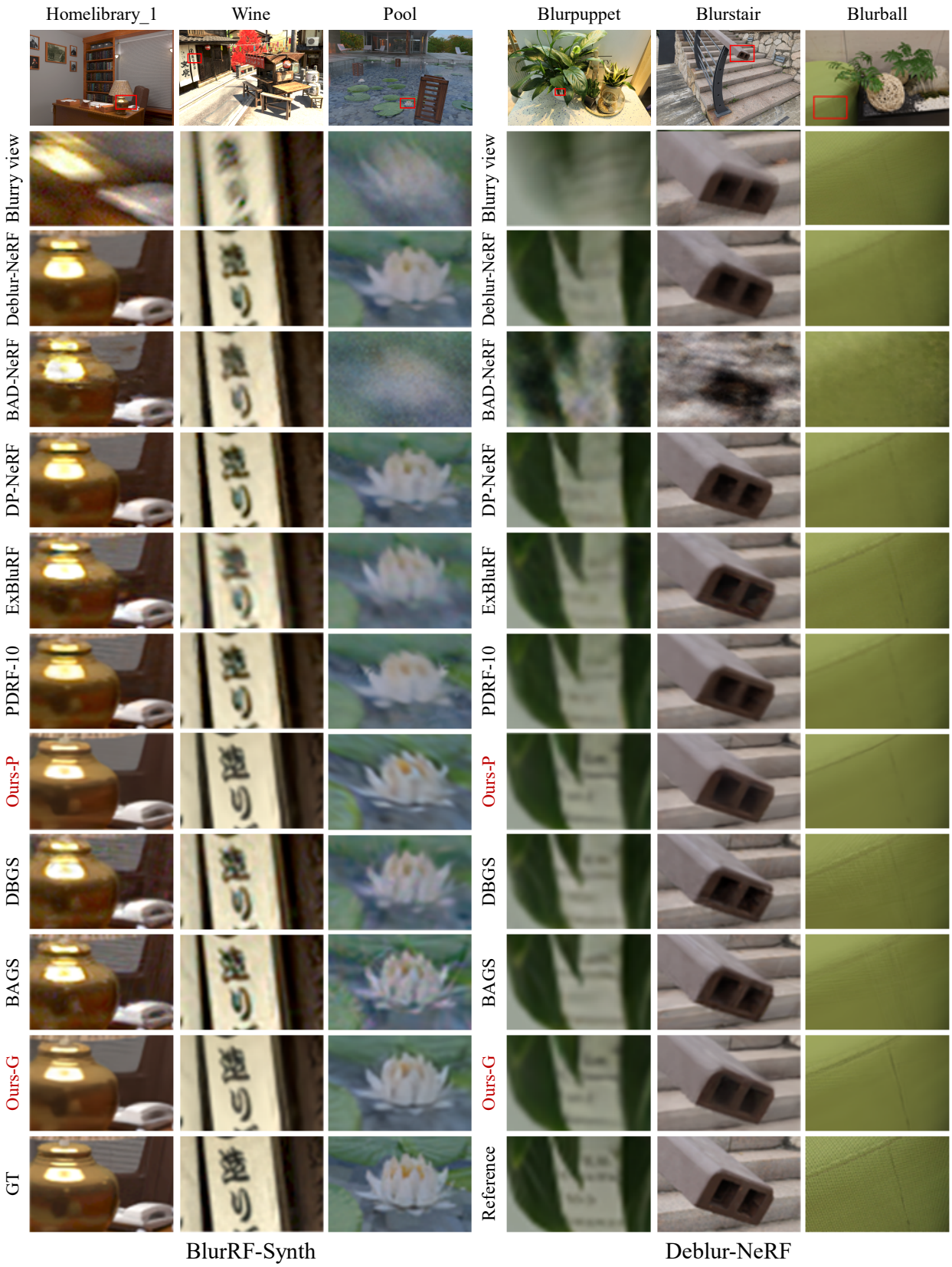

Figure S5. Qualitative comparison of novel-view synthesis on camera motion blur datasets. The left three columns show results on the test set of BlurRF-Synth, and the right three columns show real-world scenes of Deblur-NeRF [7].

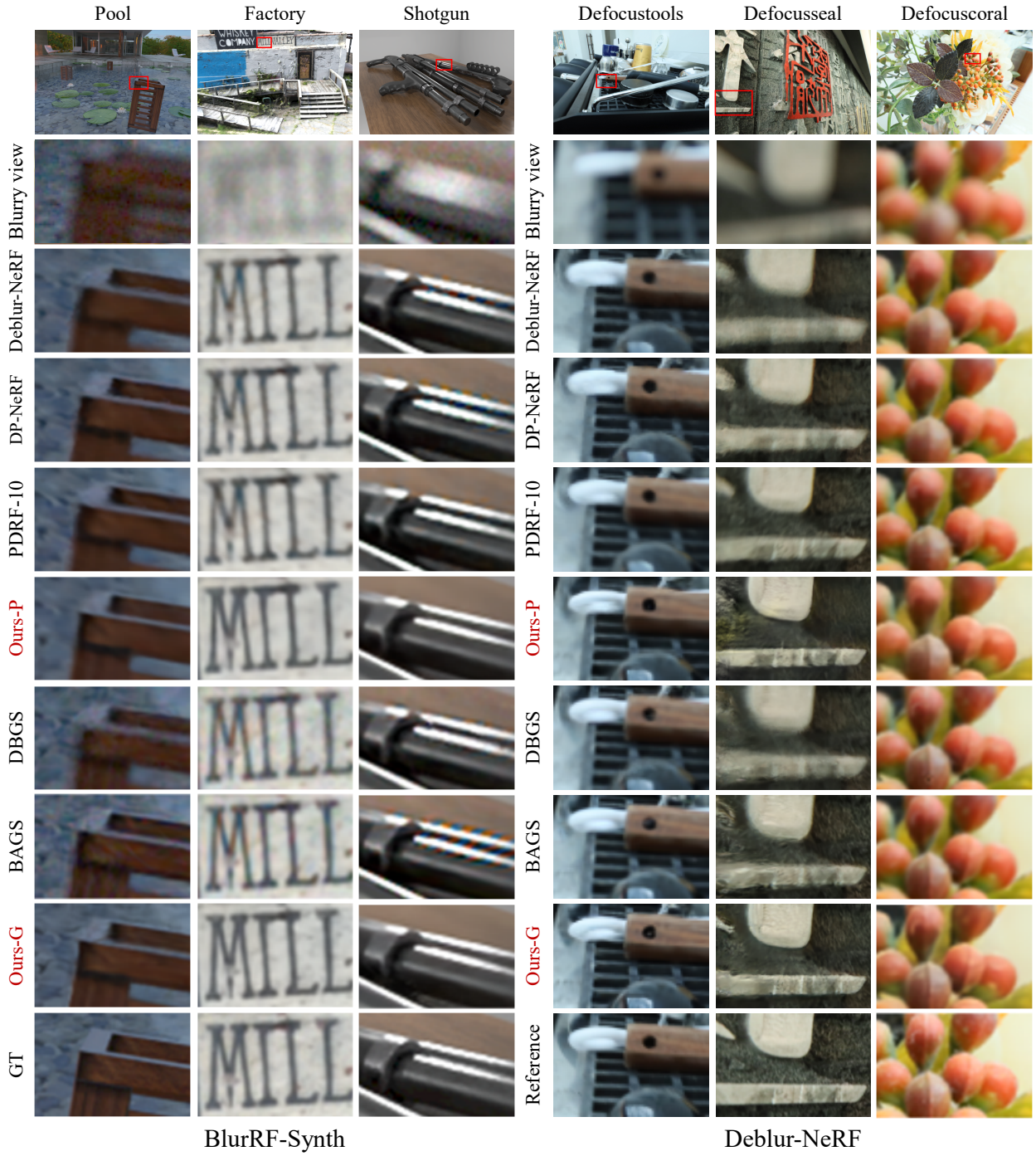

Figure S6. Qualitative comparison of novel-view synthesis on defocus blur datasets. The left three columns show results on the test set of BlurRF-Synth, and the right three columns show real-world scenes of Deblur-NeRF [7].

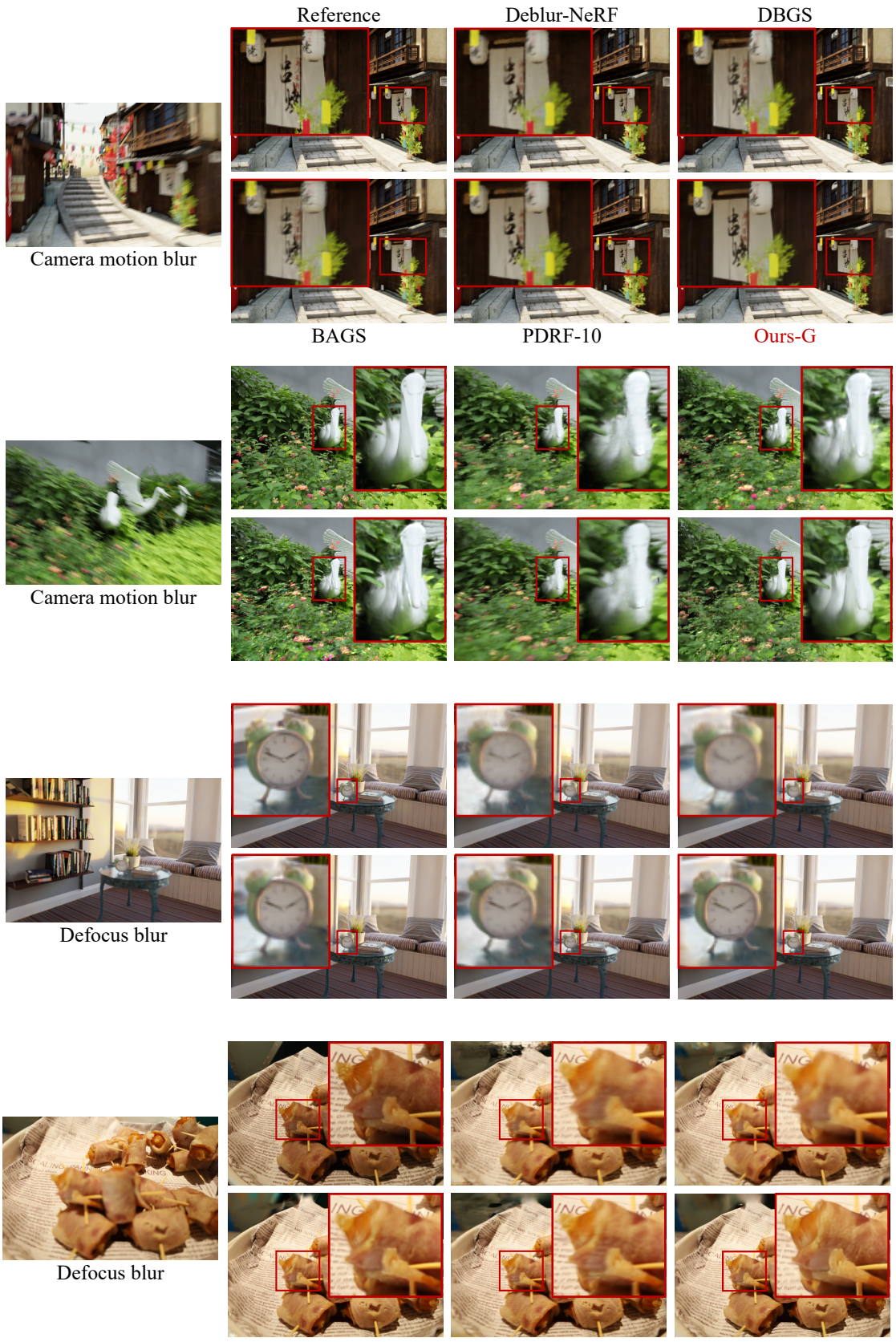

Figure S7. Qualitative comparison of novel-view synthesis on benchmark datasets [7].