Supplementary Material for ExBluRF: Efficient Radiance Fields for Extreme Motion Blur

Blur Model	Radiance Fields	N	PSNR	SSIM	LPIPS	Blur Model	Radiance Fields	N	PSNR	SSIM	LPIPS
2D Kernel	Neural	5	20.39	0.472	0.462	Cam. Traj.	Neural	5	21.21	0.522	0.391
		11	21.29	0.533	0.396			11	22.01	0.632	0.297
		21	21.22	0.539	0.368			21	22.41	0.672	0.239
Blur Model	Radiance Fields	N	PSNR	SSIM	LPIPS	Blur Model	Radiance Fields	N	PSNR	SSIM	LPIPS
2D Kernel	Voxel	5	20.18	0.465	0.518	Cam. Traj.	Voxel	5	23.24	0.613	0.411
		11	22.12	0.556	0.442			11	26.42	0.755	0.304
		21	21.57	0.537	0.437			21	26.96	0.753	0.328

Table 1: Ablation of the effectiveness of the proposed components on "Factory" scene of synthetic dataset. N refers to the number of sampling for blur reconstruction (kernel size). The 2D kernel-based blur model with neural radiance fields and the kernel size $N = 5$ (left-top) is the original configuration and official implementation of DeblurNeRF [3]. Our camera trajectory-based blur model and voxel-based radiance fields are decomposed to evaluate quantitative metrics for novel view synthesis.

In the supplementary material, we provide more discussions, comparison results, and other details that could not be included in the main manuscript due to lack of space. The contents are summarized below:

S1. Additional Ablation Study

We quantitatively compare the performance gain of the proposed camera trajectory-based blur model to the 2D kernel-based model of DeblurNeRF [3] in Tab. 1. In the same radiance fields model and the number of sampling N, our blur model outperforms the 2D kernel model with significant margins. Since the challenging camera motion consists of a complex trajectory with severe orientation changes, 2D kernel estimation may suffer from enlarging the kernel window size. In addition, the pixel-wise kernel estimation is difficult to leverage the prior knowledge that the camera is moving on a single trajectory and rays are sequentially accumulated. Therefore, the deblurring performance is less improved with the increasing kernel size as reported in [3]. On the other hand, our model directly formulates the motion blur in 6-*degree of freedom* (DOF) camera poses, which accords with the physical motion blur operation of the camera. The comparison by interchanging radiance fields model shows that the camera-based blur model is proper to optimize extreme motion blurred images regardless of the architecture of radiance fields. Furthermore, the camera trajectory-based blur model shows better compatibility with voxel-based radiance fields where PSNR and SSIM are substantially improved only with the proposed blur model.

S2. Limitations and Future Works

ExBluRF leverages the voxel-based radiance fields to increase the number of sampling for blur reconstruction with efficient computational cost. Regardless of deblurring performance, the voxel-based radiance fields can fail to render the rays that is out of the boundary of voxel grids. In addition, the optimization of the proposed method begins with the initial camera poses that are obtained from COLMAP [7]. Even though the proposed camera trajectory-based blur model has the ability to correct inaccurate initial poses, the optimization cannot be started on the scene where COLMAP fails to a total extent. These limitations of ExBluRF lead to our future plans: 1) Deformable sparse voxel-based radiance fields that are able to

extend the boundaries from the initial cube-shaped voxel grids. 2) Pose-free end-to-end optimization of ExBluRF from extreme motion blurred images.

S3. Description for Supplementary Video

We report a video that shows novel view synthesis. The novel views are generated by spiral motion for the real scenes, which is the most widely used visualization for NeRF-based approaches. ExBluRF reconstructs startlingly sharp 3D scenes from extreme motion blurred images with much fewer 3D artifacts compared to DeblurNeRF [3].

S4. Gradient to Camera Trajectory

We extend an implementation of CUDA [5] kernel to backpropagate the gradient to our camera trajectory-based blur model. From the equation of trilinear interpolation and differentiable volume rendering of voxel grid, we derive the gradient from the photo-consistency loss to the ray's origin and direction. Note that, the ray origin and direction are a function of camera pose in our model, and the backpropagtion of this part is computed by the automatic differentiation of PyTorch [6].

Gradients of trilinear interpolation. Let $\mathbf{X}_k = \mathbf{o} + s_k \cdot \mathbf{d}$ denotes the 3D location of k^{th} ray sample point with the step size s_k. The gradient of the k^{th} sample point to the ray's orgin and direction are derived by:

$$\begin{aligned} \frac{\partial \mathbf{X}_k}{\partial \mathbf{o}} &= \mathbb{1}, \\ \frac{\partial \mathbf{X}_k}{\partial \mathbf{d}} &= s_k. \end{aligned} \tag{1}$$

The volume density σ_k and color $\mathbf{c}_k$ of $\mathbf{X}_k$ is trilinear interpolated from voxel grids $\mathcal{G}_\sigma(\mathbf{X}_k)$ and $\mathcal{G}_{sh}(\mathbf{X}_k)$. Let $\mathbf{v}_\sigma^{x,y,z} \in \mathbb{R}$ and $\mathbf{v}_{sh}^{x,y,z} \in \mathbb{R}^9$ denote the interpolated volume density and spherical harmonic (SH) coefficients of $\mathbf{X}_k$, as shown in Fig. 1. The gradient of voxel grids' trilinear interpolation to $\mathbf{X}_k$ are:

$$\begin{aligned} \frac{\partial \mathcal{G}_\sigma(\mathbf{X}_k)}{\partial \mathbf{o}} &= \frac{\partial \mathcal{G}_\sigma(\mathbf{X}_k)}{\partial \mathbf{X}_k} \cdot \frac{\partial \mathbf{X}_k}{\partial \mathbf{o}} \\ &= \begin{bmatrix} \mathbf{v}_\sigma^{\lceil x \rceil, y, z} - \mathbf{v}_\sigma^{\lfloor x \rfloor, y, z} \\ \mathbf{v}_\sigma^{x, \lceil y \rceil, z} - \mathbf{v}_\sigma^{x, \lfloor y \rfloor, z} \\ \mathbf{v}_\sigma^{x, y, \lceil z \rceil} - \mathbf{v}_\sigma^{x, y, \lfloor z \rfloor} \end{bmatrix}, \\ \frac{\partial \mathcal{G}_\sigma(\mathbf{X}_k)}{\partial \mathbf{d}} &= \frac{\partial \mathcal{G}_\sigma(\mathbf{X}_k)}{\partial \mathbf{X}_k} \cdot \frac{\partial \mathbf{X}_k}{\partial \mathbf{d}} \\ &= \begin{bmatrix} \mathbf{v}_\sigma^{\lceil x \rceil, y, z} - \mathbf{v}_\sigma^{\lfloor x \rfloor, y, z} \\ \mathbf{v}_\sigma^{x, \lceil y \rceil, z} - \mathbf{v}_\sigma^{x, \lfloor y \rfloor, z} \\ \mathbf{v}_\sigma^{x, y, \lceil z \rceil} - \mathbf{v}_\sigma^{x, y, \lfloor z \rfloor} \end{bmatrix} \cdot s_k, \end{aligned} \tag{2}$$

where $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ denote ceil and floor operation to designate the voxel value covering $\mathbf{X}_k$, respectively. Note that,

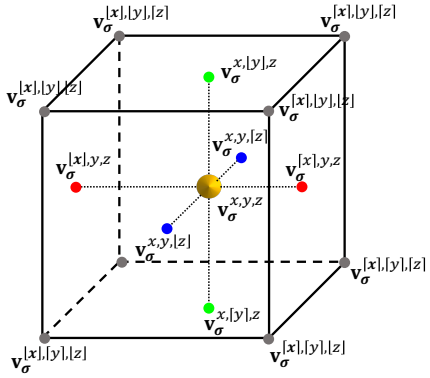


Figure 1: Visualization of trilinear interpolation from the nearest voxel grids to the ray sample point.

the same differentiation is applied to $\frac{\partial \mathcal{G}_{sh}(\mathbf{X}_k)}{\partial \mathbf{o}} \in \mathbb{R}^{3\times 9}$ and $\frac{\partial \mathcal{G}_{sh}(\mathbf{X}_k)}{\partial \mathbf{d}} \in \mathbb{R}^{3\times 9}$ without loss of generality.

Gradients of volume rendering. Along the ray sample points $\{\mathbf{X}_k\}_{k=1}^{N_k}$, the color of the ray $\hat{C}(r)$ is computed by the differentiable volume rendering [2] as follows:

$$\begin{aligned} \hat{\mathbf{C}}(\mathbf{r}) &= \sum_{k=1}^{N_k} w_k \cdot \mathbf{c}_k, \\ \text{where} \quad w_k &= \exp(-\sum_{j=1}^{k-1} \sigma_j \delta_j) \cdot (1 - \exp(-\sigma_k \delta_k)), \end{aligned} \tag{3}$$

where δ_k is the distance between the $(k-1)^{th}$ and k^{th} sample points. The color of lay is computed by the weighted sum of $\{\mathbf{c}_k\}_{k=1}^{N_k}$, where the weights $\{w_k\}_{k=1}^{N_k}$ are a function of $\{\sigma_k = \mathcal{G}_\sigma(\mathbf{X}_k)\}_{k=1}^{N_k}$. From Eq. 3, the gradient of w_k to $\{\sigma_k\}_{k=1}^{N_k}$ is derived by:

$$\frac{\partial w_k}{\partial \sigma_j} = \begin{cases} -\delta_j \exp(-\sum_{j=1}^{k-1} \sigma_j \delta_j)(1 - \exp(-\sigma_k \delta_k)) & j < k \\ \delta_k \exp(-\sum_{j=1}^{k-1} \sigma_j \delta_j) \exp(-\sigma_k \delta_k) & j = k \end{cases} \tag{4}$$

Additionally, the colors of ray sample points are computed from a SH function that is $\{\mathbf{c}_k = S(\mathbf{d})^\intercal \cdot \mathcal{G}_{sh}(\mathbf{X}_k)\}_{k=1}^{N_k}$. Therefore, the gradients of $\mathbf{c}_k$ to spherical harmonic function and coefficients are:

$$\begin{aligned} \frac{\partial \mathbf{c}_k}{S(\mathbf{d})} &= \mathcal{G}_{sh}(\mathbf{X}_k), \\ \frac{\partial \mathbf{c}_k}{\mathcal{G}_{sh}(\mathbf{X}_k)} &= S(\mathbf{d}). \end{aligned} \tag{5}$$

Gradients from Loss Function. Finally, the gradient from the photo-consistency loss to the ray's origin and directions derived by chain rule as follows:

$$\mathcal{L}_{color} = ||\hat{C}(\mathbf{r}) - C_{gt}(\mathbf{r})||_2^2, \tag{6}$$

$$
\begin{aligned}
\frac{\partial \mathcal{L}_{color}}{\partial \mathbf{o}} &= \frac{\partial \mathcal{L}_{color}}{\partial \hat{C}(\mathbf{r})} \cdot \frac{\partial \hat{C}(\mathbf{r})}{\partial \mathbf{o}} \\
&= \frac{\partial \mathcal{L}_{color}}{\partial \hat{C}(\mathbf{r})} \Big(\sum_{k=1}^{N_k} \frac{\partial \hat{C}(\mathbf{r})}{\partial w_k} \cdot \frac{\partial w_k}{\partial \mathbf{o}} + \frac{\partial \hat{C}(\mathbf{r})}{\partial \mathbf{c}_k} \cdot \frac{\partial \mathbf{c}_k}{\partial \mathbf{o}} \Big) \\
&= \frac{\partial \mathcal{L}_{color}}{\partial \hat{C}(\mathbf{r})} \Big(\sum_{k=1}^{N_k} \frac{\partial \hat{C}(\mathbf{r})}{\partial w_k} \cdot \frac{\partial w_k}{\mathcal{G}_\sigma(\mathbf{X}_k)} \cdot \frac{\mathcal{G}_\sigma(\mathbf{X}_k)}{\partial \mathbf{o}} \\
&\qquad + \sum_{k=1}^{N_k} \frac{\partial \hat{C}(\mathbf{r})}{\partial \mathbf{c}_k} \cdot \frac{\partial \mathbf{c}_k}{\mathcal{G}_{sh}(\mathbf{X}_k)} \cdot \frac{\mathcal{G}_{sh}(\mathbf{X}_k)}{\partial \mathbf{o}} \Big),
\end{aligned} \tag{7}
$$

$$
\begin{aligned}
\frac{\partial \mathcal{L}_{color}}{\partial \mathbf{d}} &= \frac{\partial \mathcal{L}_{color}}{\partial \hat{C}(\mathbf{r})} \cdot \frac{\partial \hat{C}(\mathbf{r})}{\partial \mathbf{d}} \\
&= \frac{\partial \mathcal{L}_{color}}{\partial \hat{C}(\mathbf{r})} \Big(\sum_{k=1}^{N_k} \frac{\partial \hat{C}(\mathbf{r})}{\partial w_k} \cdot \frac{\partial w_k}{\partial \mathbf{d}} + \frac{\partial \hat{C}(\mathbf{r})}{\partial \mathbf{c}_k} \cdot \frac{\partial \mathbf{c}_k}{\partial \mathbf{d}} \Big) \\
&= \frac{\partial \mathcal{L}_{color}}{\partial \hat{C}(\mathbf{r})} \Big(\sum_{k=1}^{N_k} \frac{\partial \hat{C}(\mathbf{r})}{\partial w_k} \cdot \frac{\partial w_k}{\mathcal{G}_\sigma(\mathbf{X}_k)} \cdot \frac{\mathcal{G}_\sigma(\mathbf{X}_k)}{\partial \mathbf{d}} \\
&\qquad + \sum_{k=1}^{N_k} \frac{\partial \hat{C}(\mathbf{r})}{\partial \mathbf{c}_k} \cdot \frac{\partial \mathbf{c}_k}{\mathcal{G}_{sh}(\mathbf{X}_k)} \cdot \frac{\mathcal{G}_{sh}(\mathbf{X}_k)}{\partial \mathbf{d}} \Big) \\
&\qquad + \sum_{k=1}^{N_k} \frac{\partial \hat{C}(\mathbf{r})}{\partial \mathbf{c}_k} \cdot \frac{\partial \mathbf{c}_k}{S(\mathbf{d})} \cdot \frac{S(\mathbf{d})}{\partial \mathbf{d}} \Big),
\end{aligned} \tag{8}
$$

where $\frac{\partial \mathcal{L}_{color}}{\partial \hat{C}(\mathbf{r})} = 2(\hat{C}(\mathbf{r}) - C_{gt}(\mathbf{r}))$ is derivative of mean squared error (MSE). All the other terms are derived in Sec. S4. Note that the SH function $S(\cdot)$ is only dependent on the ray's direction $\mathbf{d}$. We implemented the described differentiation in a CUDA [5] kernel for a single ray, and the blur operation with camera trajectory is implemented in auto-grad functions of PyTorch [6].

S5. All Comparison Results

We report all the quantitative and qualitative results compared on our main manuscript in Tab. 2, 3, 4 and Fig. 2, 3, 4, 5. ExBluRF shows the best deblurring on qualitative results, where the differences are larger in extremely blurred scenes. However, for some real scenes, ExBluRF shows quantitatively lower performance compared to DeblurNeRF [3]. As mentioned in Sec. S2, this is due to the limitation of the voxel-based radiance fields that cannot generate colors when the test view has unseen regions in training views. This is applied equally to neural radiance fields, but neural radiance fields fill these regions in an implicit manner even though the colors are slightly inaccurate.

References

[1] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. In *ECCV*, 2022. 4

[2] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH*, 18(3):165–174, 1984. 2

[3] Li Ma, Xiaoyu Li, Jing Liao, Qi Zhang, Xuan Wang, Jue Wang, and Pedro V. Sander. Deblur-nerf: Neural radiance fields from blurry images. In *CVPR*, 2022. 1, 2, 3, 4, 7, 8

[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 *ECCV*, 2020. 5, 6, 7, 8

[5] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020. 2, 3

[6] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 2019. 2, 3

[7] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 1

[8] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *CVPR*, 2022. 4

Method	Bench			Camellia			Dragon			Jars			
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	
Restormer [8]+NeRF	24.28	0.596	0.511	23.03	0.494	0.610	27.57	0.539	0.579	23.72	0.510	0.605	
NAFNet [1]+NeRF	25.58	0.593	0.517	23.95	0.516	0.571	29.79	0.593	0.589	25.79	0.581	0.504	
DeblurNeRF [3]	30.07	0.749	0.347	27.76	0.655	0.365	**31.92**	**0.708**	0.418	27.73	0.669	0.456	
ExBluRF (Ours)	**30.99**	**0.782**	**0.223**	**28.83**	**0.705**	**0.265**	30.65	0.701	**0.417**	**29.67**	**0.378**	**0.308**	
Method	Jars2			Postbox			Stone_lantern			Sunflowers			Average		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Restormer [8]+NeRF	21.92	0.563	0.563	23.70	0.548	0.511	24.23	0.619	0.544	26.21	0.695	0.443	24.33	0.570	0.546
NAFNet [1]+NeRF	24.82	0.641	0.470	24.72	0.580	0.429	26.32	0.676	0.456	27.34	0.714	0.414	26.03	0.612	0.494
DeblurNeRF [3]	27.26	0.682	0.475	28.48	0.712	0.345	27.08	0.700	0.468	30.66	0.800	0.342	28.87	0.709	0.402
ExBluRF (Ours)	**29.93**	**0.770**	**0.240**	**30.22**	**0.768**	**0.230**	**28.45**	**0.731**	**0.380**	**32.36**	**0.837**	**0.268**	**30.17**	**0.757**	**0.284**

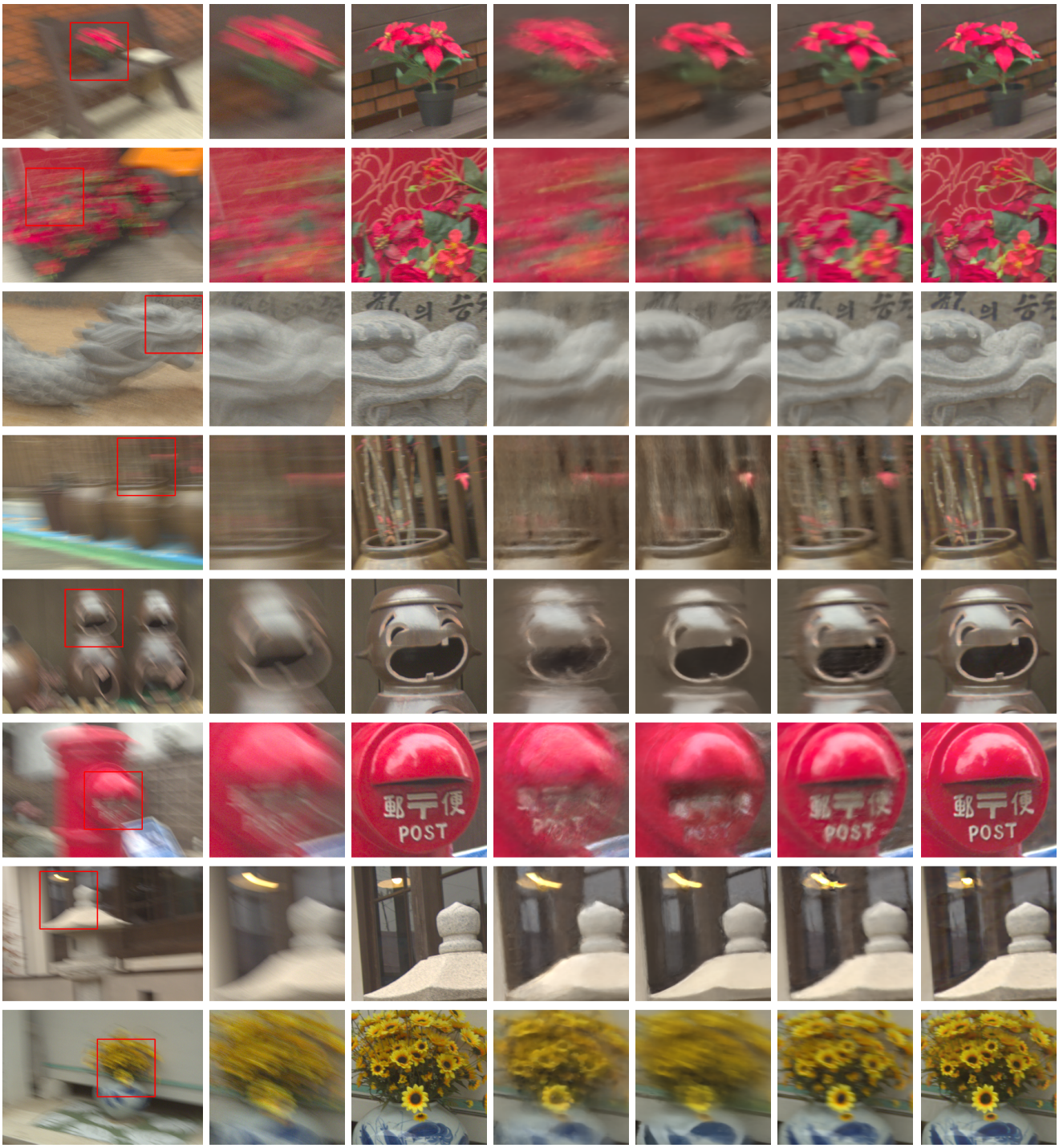
Table 2: Quantitative comparison of novel view synthesis on ExBlur dataset (proposed).

Method	Cozyroom			Factory			Pool			Tanabata			Trolley			Average		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Restormer [8]+NeRF	22.42	0.680	0.252	17.87	0.339	0.574	26.21	0.637	0.410	19.76	0.527	0.426	19.50	0.522	0.435	21.15	0.541	0.419
NAFNet [1]+NeRF	20.37	0.596	0.400	18.23	0.356	0.522	26.27	0.636	0.403	19.50	0.518	0.422	19.56	0.525	0.439	20.78	0.526	0.437
DeblurNeRF [3]	24.48	0.778	0.218	20.39	0.472	0.461	28.45	0.743	0.270	22.79	0.698	0.315	23.81	0.745	0.239	23.98	0.687	0.301
ExBluRF (Ours)	**28.55**	**0.876**	**0.121**	**26.96**	**0.753**	**0.328**	**30.05**	**0.811**	**0.251**	**26.91**	**0.847**	**0.212**	**25.92**	**0.810**	**0.267**	**27.81**	**0.823**	**0.227**

Table 3: Quantitative comparison of novel view synthesis on the synthetic dataset.

Method	Ball			Basket			Buick			Coffee			Decoration					
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓			
Restormer [8]+NeRF	25.01	0.681	0.325	24.21	0.743	0.268	21.99	0.661	0.278	27.73	0.845	0.201	23.00	0.713	0.283			
NAFNet [1]+NeRF	26.28	0.716	0.291	26.57	0.797	0.229	23.17	0.700	0.213	29.37	0.861	0.157	23.92	0.730	0.247			
DeblurNeRF [3]	27.42	0.770	0.230	27.31	0.834	0.144	24.44	0.762	0.178	30.47	0.889	0.129	**24.07**	**0.767**	**0.177**			
ExBluRF (Ours)	**28.35**	**0.791**	**0.205**	**28.86**	**0.875**	**0.126**	**25.95**	**0.809**	**0.154**	**32.44**	**0.915**	**0.101**	23.51	0.740	0.275			
Method	Girl			Heron			Parterre			Puppet			Stair			Average		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Restormer [8]+NeRF	20.16	0.725	0.282	19.59	0.572	0.331	23.75	0.676	0.317	22.69	0.654	0.278	24.06	0.555	0.328	23.22	0.682	0.289
NAFNet [1]+NeRF	21.91	0.778	0.217	21.51	0.640	0.292	21.71	0.647	0.336	24.44	0.721	0.225	23.40	0.503	0.317	24.23	0.709	0.252
DeblurNeRF [3]	**22.30**	**0.792**	**0.171**	**22.63**	**0.682**	**0.212**	**25.83**	**0.761**	**0.209**	24.80	0.734	0.163	25.69	0.645	0.213	25.49	0.763	**0.182**
ExBluRF (Ours)	21.80	0.770	0.205	22.52	0.676	0.281	24.97	0.768	0.227	**25.12**	**0.758**	**0.181**	**25.78**	**0.647**	**0.229**	**25.93**	**0.775**	0.198

Table 4: Quantitative comparison of novel view synthesis on the real dataset of DeblurNeRF [3].



Blurry View | Input | Ground Truth | Restormer+[4] | NAFNet+[4] | DeblurNeRF | ExBluRF

Figure 2: Qualitative comparison of deblurring on ExBlur dataset (proposed).

Blurry View | Input | Ground Truth | Restormer+[4] | NAFNet+[4] | DeblurNeRF | ExBluRF

Figure 3: Qualitative comparison of deblurring on the synthetic dataset.

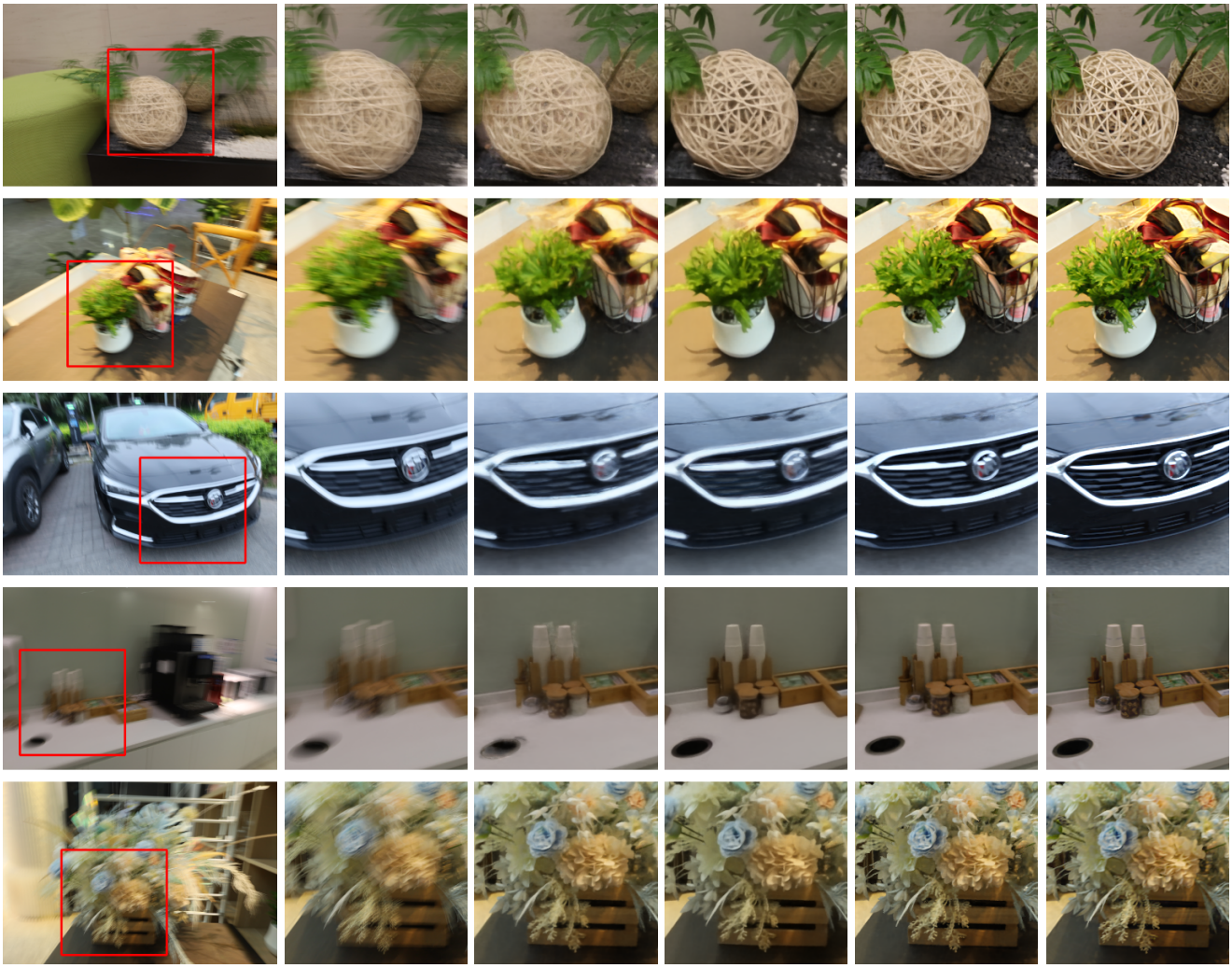


Figure 4: Qualitative comparison of deblurring on the real dataset of DeblurNeRF [3].

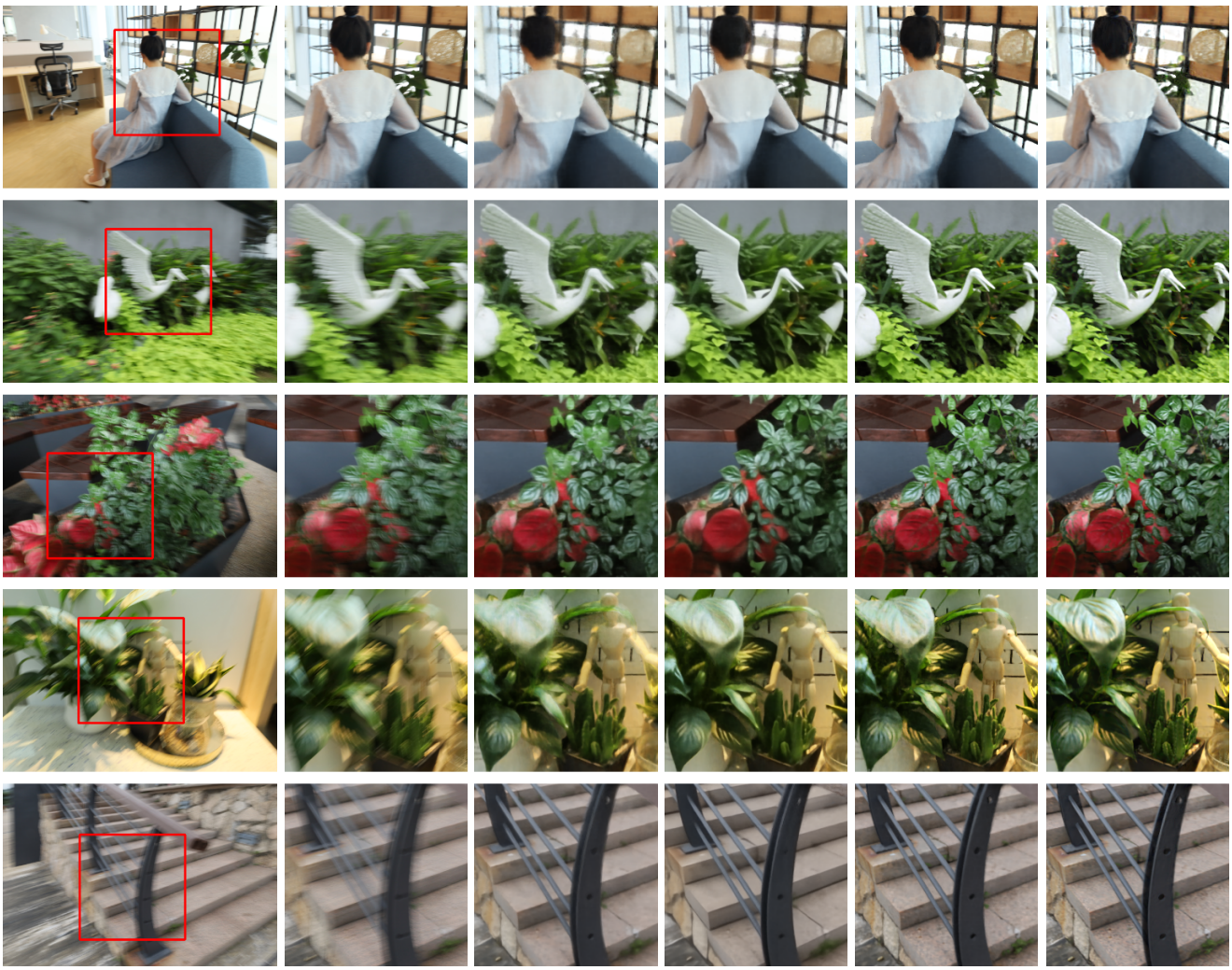


Figure 5: Qualitative comparison of deblurring on the real dataset of DeblurNeRF [3].