CoCoGaussian: Leveraging Circle of Confusion for Gaussian Splatting from Defocused Images

Supplementary Material

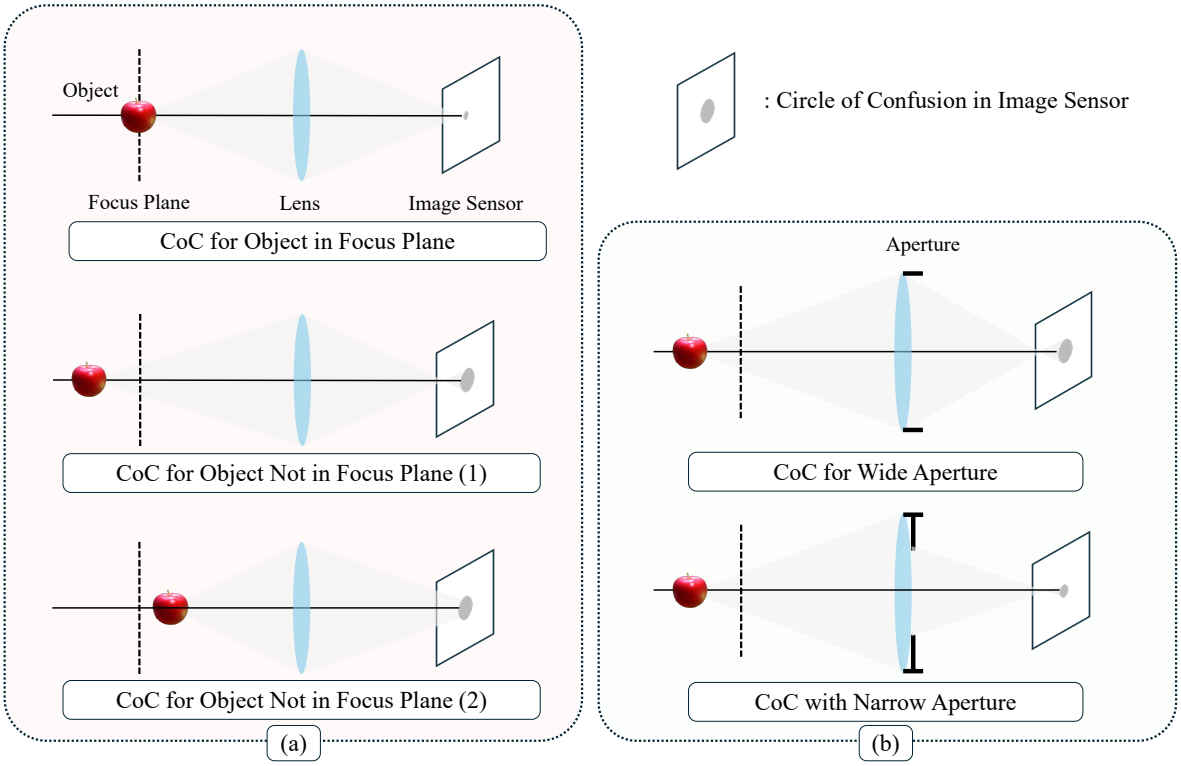

Figure 1. **(a)** CoC sizes based on the position of object relative to the focus plane, and **(b)** CoC sizes based on the aperture size.

1. Implementation Details

CoCoGaussian is built upon 3DGS [1] and Deblurring 3DGS [2], trained with a total of 30k iterations with the number of CoC Gaussians, M, set to 5. For coarse geometry in the early training stages, h_θ is not trained during the first 2k iterations and begins training afterward. We set $\delta\mathbf{q}_{\max}$ and $\delta\mathbf{s}_{\max}$ to 1.1. Additionally, after h_θ has been coarsely trained for 4k iterations, CNN $\mathcal{F}$ starts training. Note that, prior to the training of $\mathcal{F}$, the $(M+1)$ output images are averaged to obtain a blurry image. Additionally, we apply a positional encoding layer [5], γ, to 3D points (*i.e.*, $\mathbf{x}_{cam}$ and μ_B):

$$\gamma(\mathbf{x}_{cam}) = \left(\sin(2^k\pi\mathbf{x}_{cam}), \cos(2^k\pi\mathbf{x}_{cam})\right)_{k=0}^{L-1}, \quad (1)$$

$$\gamma(\mu_B) = \left(\sin(2^k\pi\mu_B), \cos(2^k\pi\mu_B)\right)_{k=0}^{L-1}, \quad (2)$$

where L denotes the number of the frequencies. The h_θ consists of 3 serial MLP layers, each with 64 hidden units, and parallelized 4 head layers for K, β, $\mathbf{d}$, and $\delta(\mathbf{q}, \mathbf{s})$. The CNN $\mathcal{F}$ comprises 4 convolutional layers with 64 channels each. To compensate for the sparse initial point cloud, we adopt the approach from Deblurring 3DGS, adding approximately 200k additional points after 2.5k iterations and pruning Gaussians based on depth. All experiments are conducted on either an NVIDIA RTX 3090 or NVIDIA V100 GPU.

2. Circle of Confusion

In this section, we explain the principles behind the generation of the Circle of Confusion (CoC) based on the focus plane and aperture size. As shown in Fig. 1 **(a)**, when a subject is precisely located on the focus plane, the radiance emitted from a point on the subject is projected onto the image sensor as a single point. However, when the subject is positioned away from the focus plane, the radiance passes through the lens and forms a circular point spread function on the image sensor. As this circle increases in size, the defocus effect becomes more pronounced. However, even if

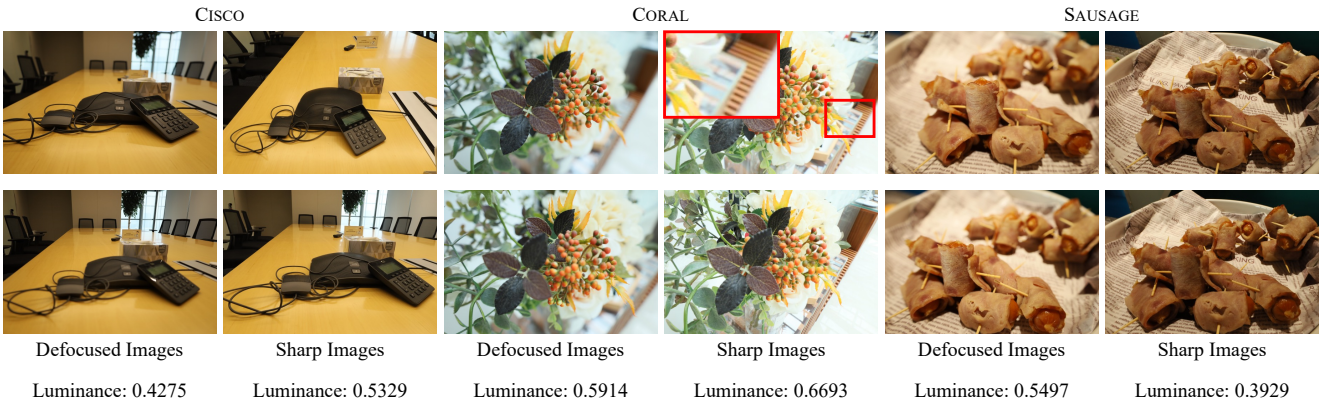

Figure 2. The Luminance Difference between Defocused and Sharp Images. All the luminance values are normalized ranging between 0 and 1.

the subject is off the focus plane, the circle is perceived as a single point if its radius remains below a certain threshold, known as the "acceptable CoC." Thus, even when a CoC exists on the image sensor, the resulting image is perceived as all-in-focus if the CoC is smaller than this threshold.

Additionally, the size of the aperture is another key factor influencing the size of the CoC. As illustrated in Fig. 1 **(b)**, radiance emitted from a subject passes through the lens and the aperture. With a larger aperture, the radiance forms a larger CoC on the image sensor. With a smaller aperture, only a portion of the light passing through the lens reaches the sensor, resulting in a smaller CoC. Consequently, a smaller aperture produces CoCs smaller than acceptable CoC, leading to an all-in-focus image. However, a smaller aperture also reduces the amount of light reaching the image sensor during the same exposure time, requiring a longer exposure to collect sufficient light. Therefore, capturing an all-in-focus image necessitates (1) a small aperture size and (2) stability to prevent camera movement during the extended exposure time.

Difference from DoF-NeRF [8]. DoF-NeRF is the first study to apply a physical CoC to 3D scene representation. As it uses NeRF [5], a ray tracing-based method, as its backbone, it has the advantage of directly modeling the CoC that reaches the image sensor for a single ray. Specifically, each ray is represented as a CoC on the image sensor, and the color derived from the ray is divided by area of the CoC, ensuring uniform pixel color within a single CoC. However, this approach has three major limitations: (1) it assumes a uniform point spread function (PSF), significantly reducing the flexibility of learning in real-world scenario, (2) it heavily relies on the CoC based on estimated depth, even when derived from uncertain depth, and (3) as an implicit neural representation, its training and rendering are extremely slow.

In contrast, while our CoCoGaussian also models the CoC in different way, it overcomes these three limitations: (1) By generating multiple Gaussians to form the CoC and performing a weighted sum of the resulting images using a CNN $\mathcal{F}$, our approach provides much greater learning flexibility for the PSF. (2) While 3DGS also suffers from challenges with uncertain depth, we propose methods to make CoCoGaussian robust to such depth inaccuracies, as described in Sec. 4.3 of the main paper. (3) Since our backbone, 3DGS, is an explicit rasterization-based method, it guarantees fast training and rendering speeds. Thus, while CoCoGaussian draws inspiration from DoF-NeRF, the contributions are clearly distinct and independent.

3. Deblur-NeRF [4] Real-World Dataset

As shown in Tab. 1 of the main paper, not only our method but also other methods on the Deblur-NeRF [4] Real-World dataset exhibit relatively poor PSNR and SSIM scores compared to their LPIPS performance. This discrepancy arises from inherent issues within the dataset itself, primarily the luminance differences between the defocused images used for training and the sharp images used for evaluation. As illustrated in Fig. 2, the CISCO and CORAL scenes have higher luminance in the sharp images, while the SAUSAGE scene has higher luminance in the defocused images. These differences result in lower PSNR and SSIM scores. However, LPIPS evaluates high-level features that align with human visual perception, making it the most reliable metric for this dataset. Consequently, as shown in Tab. 1, CoCoGaussian achieves the best LPIPS score, highlighting the comprehensive performance of our approach.

4. Additional Ablation Study

In this section, we conduct two ablative experiments. The first focuses on qualitative results related to the CoC scaling

factor β, and the second evaluates the quantitative results based on the number of CoC Gaussian sets M.

CoC Scaling Factor. As shown in Tab. 3 of our main paper, excluding the CoC scaling factor when modeling CoC Gaussians results in slightly lower performance. This factor is designed to enable robust training of CoC Gaussian positions, even with imperfectly optimized depth, which often occurs in scenes involving reflection or refraction. We visualize scenes with reflections and refractions in Fig. 3. The top scene models a transparent cup where light refracts, and the rendered result without CoC scaling shows significant artifacts on the cup. Similarly, in the bottom scene, where a metallic pipe causes light reflection, the absence of CoC scaling leads to many floaters in the affected areas. This indicates that the CoC Gaussians without CoC scaling factor are overfitted to the training images, resulting in poorly optimized base Gaussian positions μ_B. Furthermore, this overfitting may potentially affect the covariance of the Gaussians and their SH coefficients. In other words, objects with reflections or refractions often exhibit inconsistent radiance depending on the view direction, making it challenging for 3DGS [1] to properly optimize for such textures. This inherent limitation of 3DGS causes the CoC Gaussians to overly rely on inaccurately optimized depth, leading to suboptimal outputs. By incorporating the CoC scaling factor during training, we enable robust modeling of CoC Gaussians even in scenes with challenging reflective or refractive surfaces.

Number of CoC Gaussians. We conduct ablative experiments on the number of CoC Gaussians, M, with the results as shown in Tab. 1. For PSNR and SSIM, the scores vary inconsistently as M changes. This inconsistency arises primarily from differences in light exposure between defocused images for training and sharp images for evaluation, which depends on the aperture size and exposure time used to equalize the amount of light. In the DoF-NeRF [8] dataset, defocused images are captured with an aperture of f/4 and an exposure time of 1/13 seconds, while sharp images are captured with an aperture of f/11 and an exposure time of 0.8 seconds. The change from f/4 to f/11 reduces light by a factor of 1/8 due to a 3-stop aperture decrease. Meanwhile, the exposure time increases by a factor of approximately 10.4, from 1/13 seconds to 0.8 seconds. Accounting for both aperture and exposure time, sharp images receive 1.3 times more light than defocused images, making the latter slightly darker. Consequently, higher PSNR and SSIM scores do not necessarily indicate better quality.

On the other hand, for LPIPS, larger values of M generally result in better performance. Since LPIPS evaluates quality based on high-level features, such as geometric differences, rather than pixel-level or luminance-based differences, it aligns more closely with human visual perception. As a result, a lower LPIPS score is a more reliable measure of image quality compared to higher PSNR or SSIM scores. Further details are provided in Sec. 3.

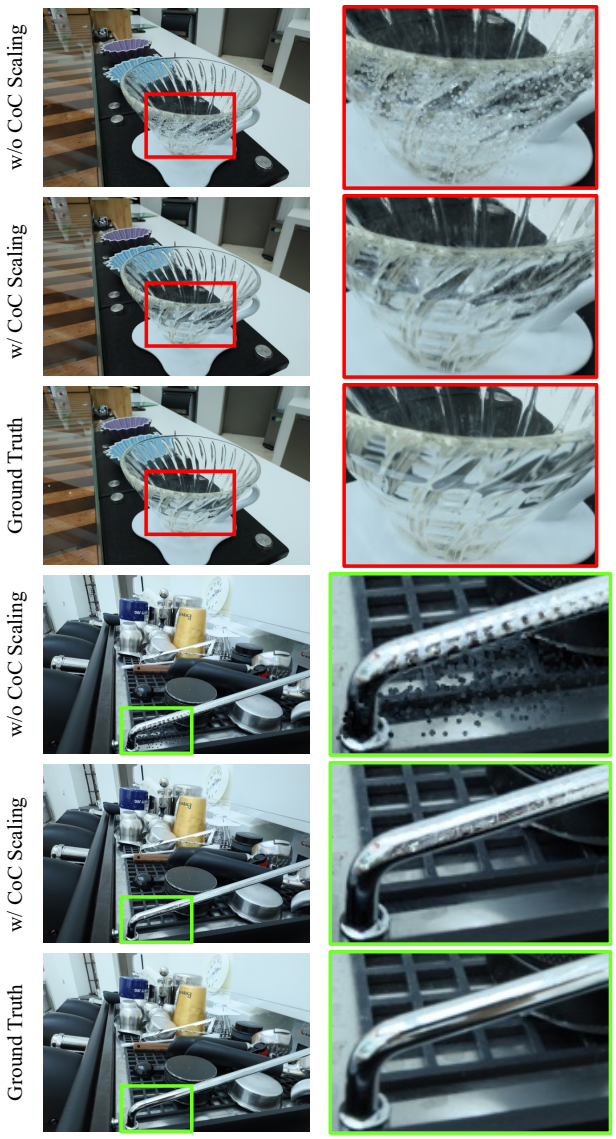

Figure 3. Qualitative Ablation of CoC Scaling Factor. The figures are from CUP and TOOLS scenes of Deblur-NeRF [4] real-world dataset.

5. Computational Efficiency and Speed

We compare our GPU usage, training time, and rendering speed with BAGS [7], a state-of-the-art method, on the Deblur-NeRF real-world dataset using an NVIDIA RTX 3090. As shown in the Tab. 2, CoCoGaussian achieves comparable resource consumption and training time while delivering superior performance.

Table 1. Quantitative Results for the Number of CoC Gaussian Sets M. The orange and yellow cells respectively indicate the highest and second-highest values.

Methods	AMIYA			BOOK			CAMERA			DESK		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
$M=2$	30.06	0.9313	0.0915	30.74	0.9277	0.0626	29.44	0.9276	0.0725	30.07	0.9163	0.0652
$M=3$	31.02	0.9389	0.0882	31.89	0.9377	0.0576	29.36	0.9270	0.0723	30.57	0.9273	0.0592
$M=4$	30.71	0.9395	0.0858	30.45	0.9207	0.0600	29.63	0.9283	0.0707	29.68	0.9193	0.0532
$M=5$	30.59	0.9406	0.0867	31.58	0.9304	0.0585	30.21	0.9372	0.0609	29.94	0.9188	0.0559
$M=6$	31.21	0.9411	0.0829	30.94	0.9255	0.0573	29.39	0.9266	0.0698	29.48	0.9151	0.0524
Methods	KENDO			PLANT			SHELF			**AVERAGE**		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
$M=2$	25.77	0.8414	0.1303	31.21	0.8899	0.0901	32.02	0.9351	0.0455	29.90	0.9099	0.0797
$M=3$	25.83	0.8441	0.1264	30.78	0.8814	0.0859	29.91	0.9131	0.0462	29.91	0.9099	0.0765
$M=4$	25.76	0.8246	0.1277	29.59	0.8642	0.0830	31.20	0.9245	0.0480	29.57	0.9030	0.0755
$M=5$	26.26	0.8560	0.1160	31.01	0.8881	0.0693	31.38	0.9181	0.0431	30.14	0.9127	0.0701
$M=6$	26.67	0.8607	0.1117	27.58	0.8661	0.0894	31.42	0.9243	0.0460	29.53	0.9085	0.0728

Table 2. Computational Efficiency and Speed. * indicates that the rendering speed is identical to that of the corresponding model.

Methods	GPU Memory (GB)	Training Time (min)	Rendering Speed
BAGS [7]	4.7	47	*Mip-Splatting [9]
CoCoGaussian	5.3	45	*3DGS [1]

After the training phase, CoCoGaussian renders sharp images using only the $\mathbf{G}_B$ through a naive 3DGS. In the other words, the rendering speed and memory cost are the same as 3DGS alone. Therefore, our method is feasible for real-time applications that rely on 3DGS rendering speeds.

6. CoC Visualization

We visualize the CoC for various types of images in Fig. 4 and Fig. 5. To simplify the visualization, we randomly sample a subset of positions from numerous Gaussians. The points in Fig. 4 represent the positions of Gaussians for defocused images. For images where the focus plane is close to the camera, CoCoGaussian generates CoC sizes that are very small for shallow depths and progressively larger for deeper depths. However, when the focus plane is farther from the camera, the CoC sizes reconstructed at greater depths are smaller. This demonstrates the effectiveness of our modeling, accurately reflecting the principles of defocus blur.

In contrast, Fig. 5 visualizes the CoC for an all-in-focus scene [5], where the CoC sizes remain uniformly small regardless of depth. This suggests that the learned aperture size is very small, ensuring precise modeling of all-in-focus images. In other words, CoCoGaussian can model small apertures and capture subtle defocus effects that are imperceptible to the human eye, which contributes to its superior performance compared to naive 3DGS [1], as demonstrated in Tab. 4 of the main paper. In summary, our model can accurately represent 3D scenes for both defocused and all-in-focus images, highlighting its versatility and robustness.

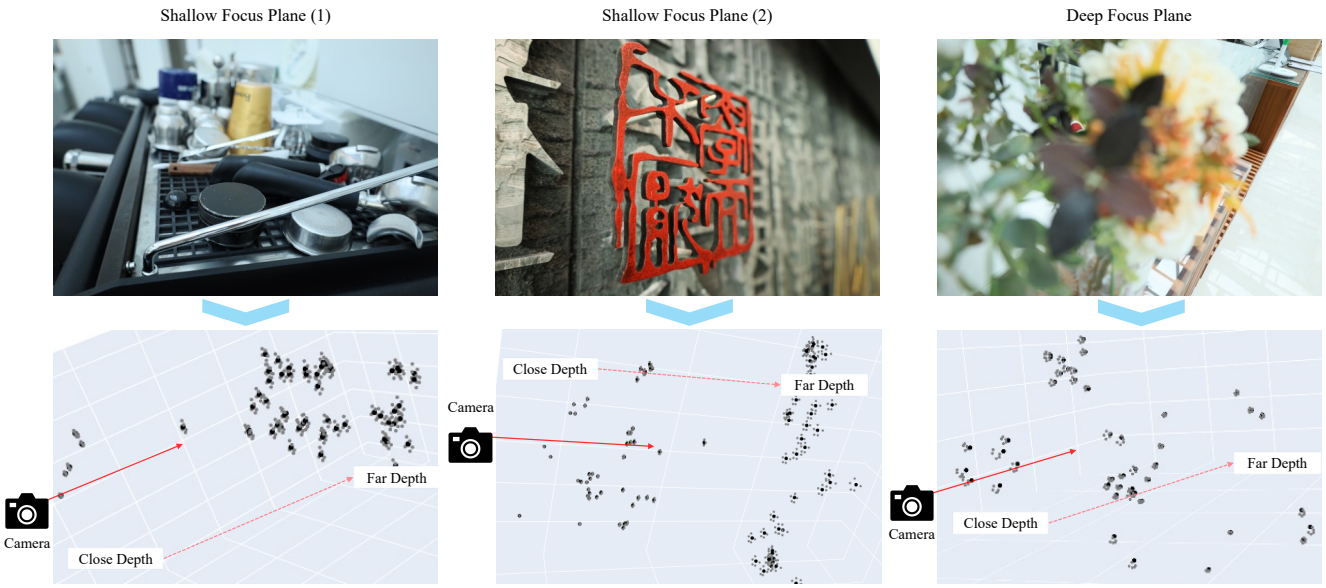

Figure 4. Gaussian Positions of Defocused Images. The black and gray dots indicate base and CoC Gaussians, respectively.

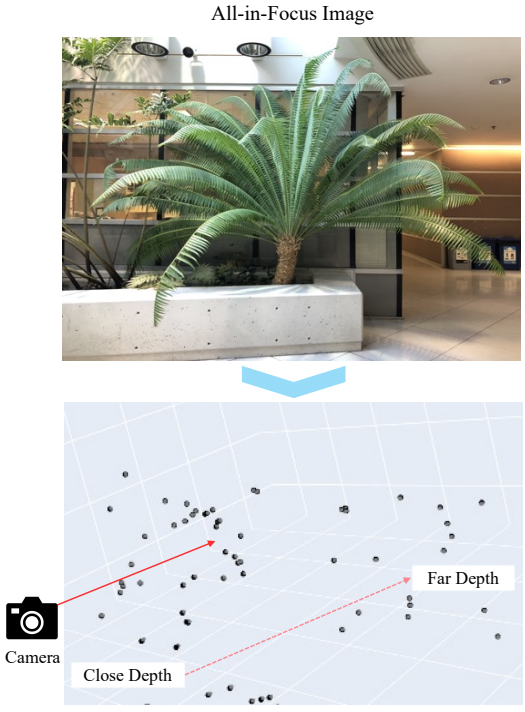

Figure 5. Gaussian Positions of All-in-Focus Images.

7. Per-Scene Quantitative Results

We present the performance for individual scenes across all datasets in Tabs. 3 to 5. CoCoGaussian achieves the best LPIPS scores in all scenes except for the CORAL scene in the Deblur-NeRF Real-World dataset. As discussed in Secs. 3 and 4, this indicates that CoCoGaussian exhibits the most superior high-level feature representation.

Table 3. Per-Scene Quantitative Results on Deblur-NeRF [4] Synthetic Dataset.

Methods	Factory			Cozyroom			Pool			Tanabata			Trolley		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
NeRF [5]	25.36	0.7847	0.2351	30.03	0.8926	0.0885	27.77	0.7266	0.3340	23.90	0.7811	0.2142	22.67	0.7103	0.2799
3DGS [1]	24.52	0.8057	0.1842	30.09	0.9024	0.0692	20.14	0.4451	0.5094	23.08	0.7981	0.1710	22.26	0.7400	0.2281
Deblur-NeRF [4]	28.03	0.8628	0.1127	31.85	0.9175	0.0481	30.52	0.8246	0.1901	26.26	0.8517	0.0995	25.18	0.8067	0.1436
PDRF-10 [6]	30.90	0.9138	0.1066	32.29	0.9305	0.0518	30.97	0.8408	0.1893	28.18	0.9006	0.0819	28.07	0.8799	0.1210
DP-NeRF [3]	29.26	0.8793	0.1035	32.11	0.9215	0.0386	31.44	0.8529	0.1563	27.05	0.8635	0.0779	26.79	0.8395	0.1170
Deblurring 3DGS [2]	27.39	0.8922	0.1160	31.29	0.9201	0.0505	31.27	0.8565	0.1556	27.04	0.9029	0.0872	27.53	0.8843	0.1167
BAGS [7]	30.87	0.9334	0.0724	32.45	0.9312	0.0289	31.78	0.8645	0.0932	29.19	0.9278	0.0405	28.97	0.9070	0.0804
Ours	30.15	0.9300	0.0489	33.02	0.9410	0.0213	31.96	0.8788	0.0803	29.65	0.9396	0.0263	29.41	0.9165	0.0620

Table 4. Per-Scene Quantitative Results on Deblur-NeRF [4] Real-World Dataset.

Methods	Cake			Caps			Cisco			Coral			Cupcake		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
NeRF [5]	24.42	0.7210	0.2250	22.73	0.6312	0.2801	20.72	0.7217	0.1256	19.81	0.5658	0.2155	21.88	0.6809	0.2689
3DGS [1]	20.16	0.5903	0.2082	19.08	0.4355	0.4329	20.01	0.6931	0.1781	19.50	0.5519	0.3111	21.53	0.6794	0.2081
Deblur-NeRF [4]	26.27	0.7800	0.1282	23.87	0.7128	0.1612	20.83	0.7270	0.0868	19.85	0.5999	0.1160	22.26	0.7219	0.1214
PDRF-10 [6]	27.06	0.8032	0.1622	24.06	0.7102	0.2854	20.68	0.7239	0.0943	19.61	0.5894	0.2335	22.95	0.7421	0.1862
DP-NeRF [3]	26.16	0.7781	0.1267	23.95	0.7122	0.1430	20.73	0.7260	0.0840	20.11	0.6107	0.0960	22.80	0.7409	0.1178
Deblurring 3DGS [2]	26.91	0.8039	0.1136	24.45	0.7391	0.1509	20.55	0.7227	0.0816	18.99	0.5534	0.2767	22.11	0.7356	0.1021
BAGS [7]	26.53	0.7996	0.1113	24.15	0.7422	0.1391	20.31	0.7230	0.0759	19.63	0.6016	0.1114	21.52	0.6971	0.1214
Ours	26.65	0.8043	0.1037	24.62	0.7472	0.1334	20.83	0.7359	0.0680	19.57	0.5925	0.1105	22.48	0.7515	0.0739

Methods	Cups			Daisy			Sausage			Seal			Tools		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
NeRF [5]	25.02	0.7581	0.2315	22.74	0.6203	0.2621	17.79	0.4830	0.2789	22.79	0.6267	0.2680	26.08	0.8523	0.1547
3DGS [1]	20.55	0.6459	0.3211	20.96	0.6004	0.2629	17.83	0.4718	0.2855	22.25	0.5905	0.3057	23.82	0.8050	0.1953
Deblur-NeRF [4]	26.21	0.7987	0.1271	23.52	0.6870	0.1208	18.01	0.4998	0.1796	26.04	0.7773	0.1048	27.81	0.8949	0.0610
PDRF-10 [6]	26.39	0.8066	0.1370	24.49	0.7426	0.1024	18.94	0.5686	0.2126	26.36	0.7959	0.1927	28.00	0.8995	0.1395
DP-NeRF [3]	26.75	0.8136	0.1035	23.79	0.6971	0.1075	18.35	0.5443	0.1473	25.95	0.7779	0.1026	28.07	0.8980	0.0539
Deblurring 3DGS [2]	26.23	0.8230	0.1014	23.39	0.7288	0.0979	18.83	0.5609	0.1470	26.04	0.8087	0.0988	27.86	0.9069	0.0619
BAGS [7]	26.14	0.8194	0.0901	23.00	0.7332	0.0540	18.66	0.5721	0.1176	26.16	0.8050	0.0967	28.72	0.9148	0.0450
Ours	26.20	0.8317	0.0773	23.39	0.7425	0.0503	19.20	0.5864	0.1007	26.10	0.8243	0.0663	27.91	0.9149	0.0416

Table 5. Per-Scene Quantitative Results on DoF-NeRF [8] Real-World Dataset.

Methods	Amiya			Book			Camera			Desk		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
3DGS [1]	26.16	0.8718	0.1761	24.73	0.8480	0.1980	25.05	0.8429	0.1641	27.47	0.8760	0.1422
DP-NeRF [4]	28.64	0.8849	0.1421	28.51	0.8520	0.1669	26.50	0.8667	0.1332	27.57	0.8238	0.1712
Deblurring 3D-GS [6]	26.21	0.8728	0.1815	27.29	0.8754	0.1584	25.63	0.8486	0.1796	29.08	0.8478	0.1672
BAGS [3]	31.86	0.9453	0.0874	29.41	0.7515	0.2214	29.20	0.9248	0.0730	29.77	0.9175	0.0722
Ours	30.59	0.9406	0.0867	31.58	0.9304	0.0585	30.21	0.9372	0.0609	29.94	0.9188	0.0559

Methods	Kendo			Plant			Shelf			**Average**		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
3DGS [1]	20.63	0.6839	0.2715	26.64	0.7897	0.2047	29.37	0.8913	0.1150	25.72	0.8291	0.1817
DP-NeRF [4]	19.64	0.6166	0.3205	28.07	0.8158	0.1668	28.64	0.8415	0.1525	26.80	0.8145	0.1790
Deblurring 3D-GS [6]	22.74	0.7559	0.1904	26.66	0.7900	0.2081	28.44	0.8798	0.1102	26.58	0.8386	0.1708
BAGS [3]	26.19	0.8457	0.1291	30.66	0.8644	0.1128	32.03	0.9223	0.0741	29.87	0.8816	0.1100
Ours	26.26	0.8560	0.1160	31.01	0.8881	0.0693	31.38	0.9181	0.0431	30.14	0.9127	0.0701

References

[1] 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, 3, 4, 7

[2] Byeonghyeon Lee, Howoong Lee, Xiangyu Sun, Usman Ali, and Eunbyung Park. Deblurring 3d gaussian splatting. *arXiv preprint arXiv:2401.00834*, 2024. 1, 7

[3] 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. 7

[4] 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, 7

[5] 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 *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, pages 405–421, 2020. 1, 2, 4, 7

[6] 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. 7

[7] 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. *arXiv preprint arXiv:2403.04926*, 2024. 3, 4, 7

[8] 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, 3, 7

[9] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19447–19456, 2024. 4