Bokehlicious: Photorealistic Bokeh Rendering with Controllable Apertures

Supplementary Material

In the supplementary material, we first show the Aperture Attention Block (AAB) used in the Residual Groups (RGs) of our Bokehlicious architecture in Sec. A, then we present the hyperparameter study of our method in Sec. B. Next, Sec. C and Sec. D provide visualization of feature activations within our network.

We also provide detailed descriptions of the benchmarking methods and their training procedures in Sec. E.

Additional dataset samples with varying apertures are shown in Sec. F. To align with standard single-aperture practices, qualitative comparisons are presented in Sec. G, along with results on the conventional *EBB! Val294* [4] benchmark in Sec. H and EBB400 [17] in Sec. I. We also show the impact of loss proportions in Sec. J.

We compare our purely neural single-step approach to controllable aperture bokeh rendering with previous multi-step architectures. The full version of Tab. 5 showing the performance at all apertures represented in RealBokeh is provided in Sec. K. The uncropped versions of our qualitative comparison in Fig. 7 can be found in Sec. L, with additional qualitative samples in Sec. M. Examples of smooth aperture control, interpolating between the known f-stops from the training data, are shown in **accompanying videos**, including application on smartphone images from [24] in a zero-shot way.

Finally, we show additional comparisons on real-world portrait photography in Sec. N, general real-world applications in Sec. O, and explore our potential in deblurring scenarios in Sec. P, respectively.

A. Aperture Attention Block (AAB)

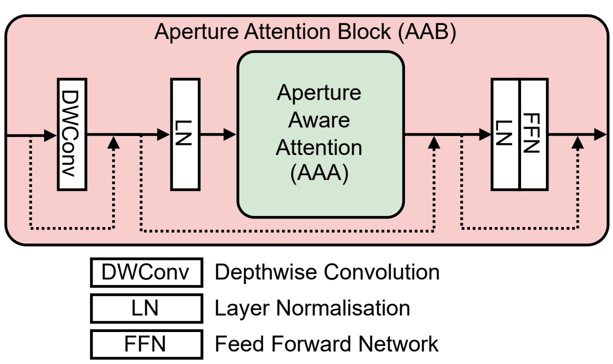


Figure A. Diagram of the Aperture Attention Block (AAB) from Fig. 4, it embeds our Aperture Aware Attention (AAA) mechanism as described in Fig. 5 and sec 4.3. This block architecture was adopted from Fan *et al*. [6].

B. Hyperparameter Study

The results of our study on the effects of different hyperparameter choices within our proposed architecture are shown in Fig. B. Here, a baseline configuration called Bokehlicious-dev is used and marked in red. In particular, it is implemented with a CNN width of 16 channels, four Residual Groups, each containing five blocks with three AAA heads on a 96-dimensional embedding.

For experiments a) - e), each configuration was trained with crops of $384 \times 384px$ resolution on *RealBokeh*. The experiment f) on the training resolution used RealBokeh$_{bin}$. All experiments were trained until convergence using Adam with a learning rate of $5e-4$.

a) Embedding Dimensions: The dimension of the embedding used by our transformer backbone has a significant impact on the computational complexity of Bokehlicious. Interestingly, our architecture remains relatively robust when using a very small embedding size such as 16.

b) CNN Width: The width of the CNN encoder and decoder has a more limited effect on the computational cost, compared to the dimension of the transformer embedding. Likewise, our architecture is robust to thin CNN encoder/decoder modules.

c) Number of Residual Groups: The computational complexity of our method naturally scales linearly with the number of groups. Our results indicate that Bokehlicious should be implemented with at least two groups.

d) Number of Attention Blocks: Analogously to the number of groups, the computational complexity increases linearly. Our findings indicate that a minimum of three attention blocks per group is imperative to achieve satisfactory output fidelity.

e) Number of Attention Heads: For a 96-dimensional embedding, it is advisable to employ three or four attention heads.

f) Training Resolution: Naturally, as a training patch needs to include the full extent of a Bokeh blur kernel, our Bokehlicious architecture suffers massively when small training sizes are used. The study suggests that this criterion is likely to be satisfied at $384px$ or $512px$, as the enhancement beyond these resolutions is relatively small.

Based on the results of this study, we chose the parameters of our proposed Bokehlicious-M as defined in Sec. 4.1.

C. Exploration of Deep Layers

In Fig. C, we provide a visualization of the AAA activation maps for all RGs within the transformer backbone of

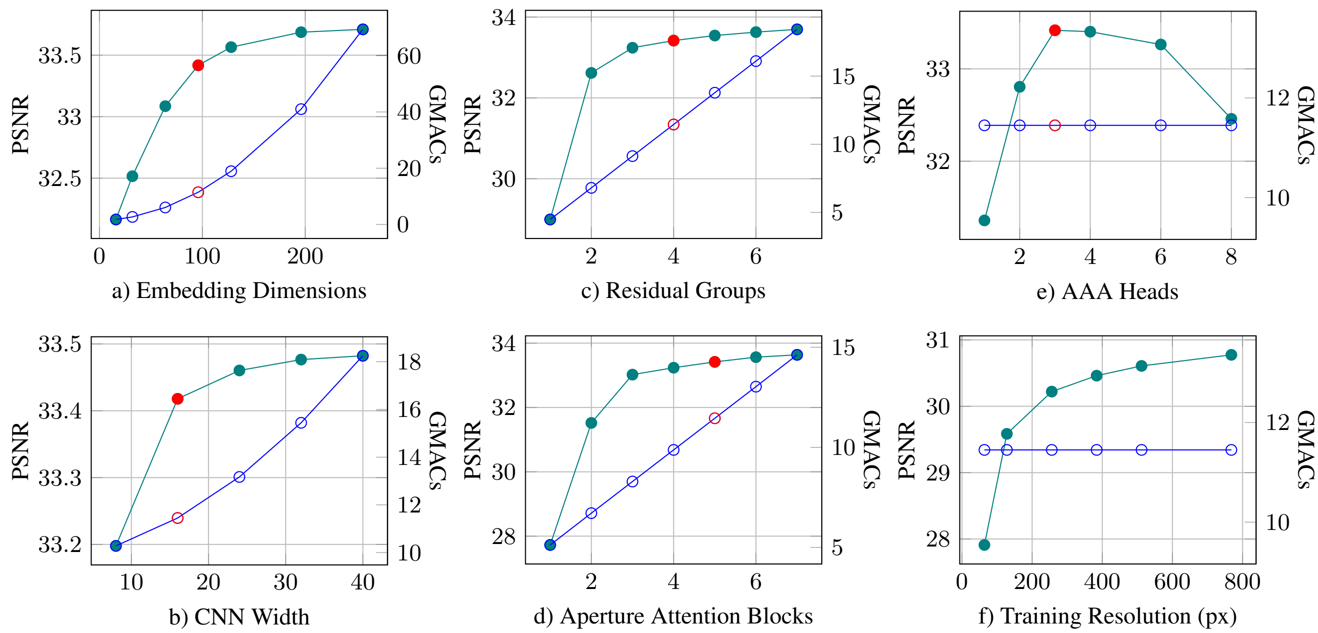


Figure B. **Results of our hyperparameter study.** The PSNR fidelity is denoted by the teal plots, GMACs complexity at $256 \times 256px$ is denoted by the blue plots, the baseline Bokehlicious-dev is marked in red. Note that experiment f) used RealBokeh$_{bin}$.

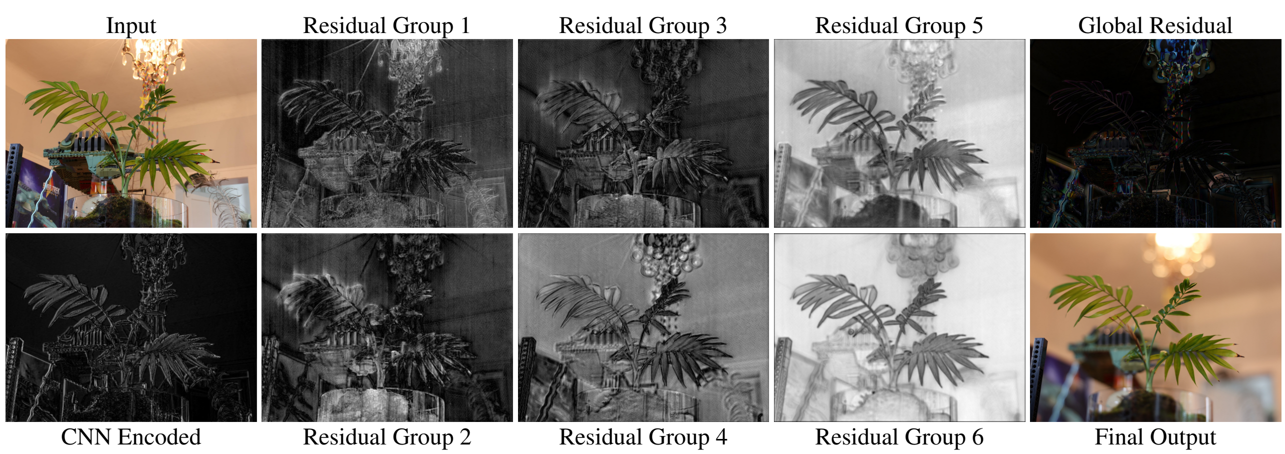
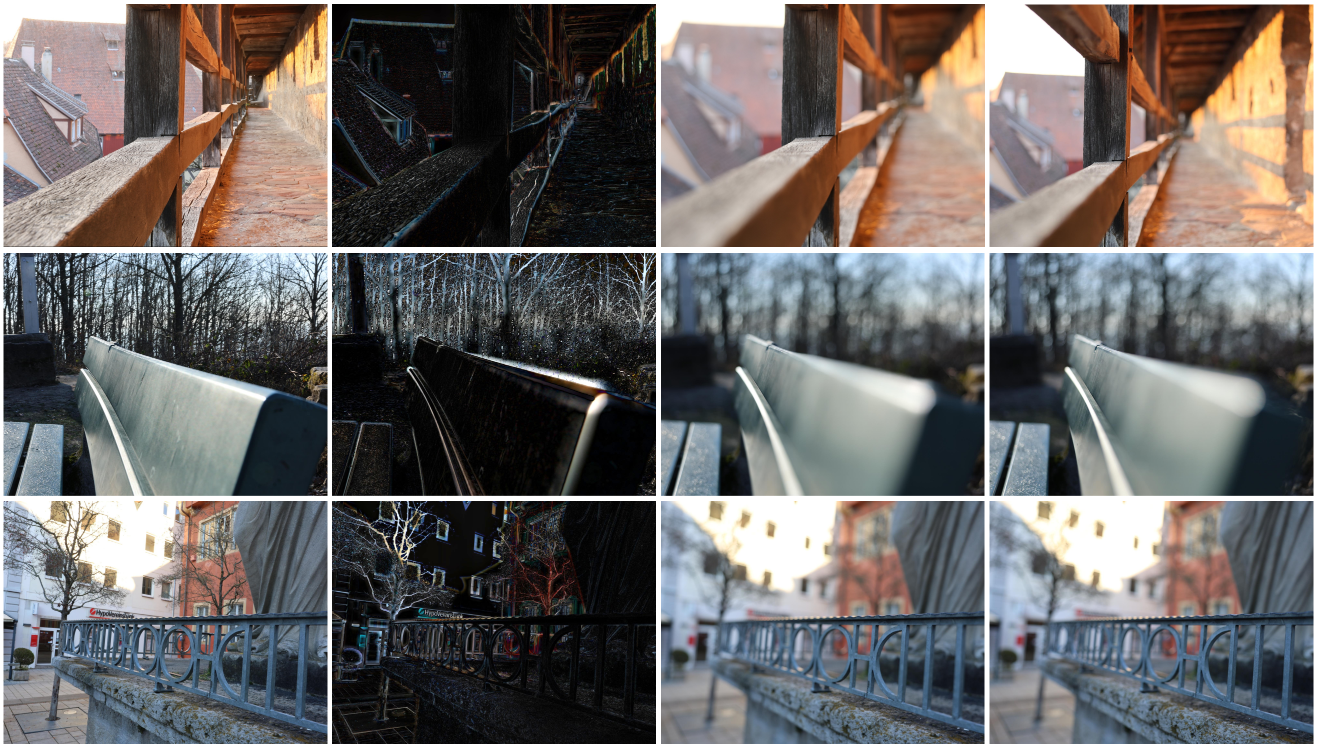


Figure C. Activation maps for the **encoded features**, **each residual group** and the resulting **global residual**.

our method. The behavior is particularly interesting, with the first three blocks focusing around high-intensity light sources, which is the area that will suffer major modifications during Bokeh Rendering. In addition, groups 1-3 have an activation of higher magnitude for the foreground objects, since in-focus contents are characterized by increased sharpness in the rendered shallow DoF image. In group 4 of the proposed transformer structure, we can observe the focus shifting to the Bokeh “balls” appearing around the light sources. We can observe that the geometry of this region is being refined from block to block. The latter groups seem to focus specifically on areas of the image background, since this is the area that is most affected by the Bokeh effect.

We also provide an RGB representation of the residual learned by the proposed model, in which we can observe the image areas affected the most by the Bokeh rendering. Naturally, the region corresponding to the leaves that are in-focus receives the slightest domain shift, while contents localized either closer or deeper than the position of the projected focal plane receive higher modifications. Unsurprisingly, high-intensity segments corresponding to out-of-focus areas are affected the most, since the Bokeh effect corresponding to them is stronger in manifestation. We can also observe that, in background areas with little high-frequency detail, there is barely any activation. Logically, this is because their changes are minimal.



Input Residual Output GT

Figure D. Additional **examples for the RGB Residual** corresponding to the Bokeh renderings of our method
.

D. Additional Analysis of the Residual

In Fig. D, we provide additional visual examples of the RGB representation of the learned residual for an input image and the corresponding Bokeh rendering performed by our model. In the upper example, the focus point is the intersection between the horizontal and vertical pillars. Naturally, this is also the lowest magnitude area in the residual.

The same can be observed in the other two examples. The horizontal edges are in the examples are the easiest to follow since they also provide insight regarding the depth of the represented scene. The important detail we want to emphasize here is the width of the area that denotes the segment around the edges that suffers the most severe change during rendering. Naturally, this is correlated with the strength of the defocus effect, which depends on the optical system used for acquisition.

As our training images were acquired with a Canon 28-70mm *f/2.0* L lens, the observed geometry can be correlated with its optical system. It shows a non-linear progression starting from the focus point position and increases in intensity as the distance to the focal plane increases.

E. Training Details of Benchmark Methods

As mentioned in Sec. 5, we used the official code bases for all methods in our benchmarks on Bokeh Rendering in Sec. 5.1 and Sec. 2. If any method required an additional depth channel as input, the SOTA monocular depth estimation method *DepthAnything* [27] was used. Whenever possible, we used a training resolution of $512 \times 512px$ and a batch size of four, following our own training methodology. In case of an exception, we have noted it in the network description accordingly.

Some methods in our benchmark at least partially adopt a classical rendering approach [17, 18, 21]. These algorithms provide multiple parameters that have to be manually tuned by the user to achieve a pleasant output. All methods require a focus distance D and strength factor K, An optimal D for each *scene* and an optimal K for each *sample-pair* is determined following the procedure used by [17], with the maximum possible K being 250.

In addition to these crucial basic settings, there are additional user options that influence the look of the rendered Bokeh. All methods offer a gamma setting G which can be used to influence the contrast of the rendered Bokeh, with BokehMe [17] also offering an enhanced highlight rendering toggle H. To simulate optimal user behavior, we optimize both parameters G and H on a scene-by-scene basis. Here, with an optimal D and K known, we take the *f/2.0* sample as a reference and generate all outputs for G settings between 2 and 5 in a 0.25-step interval and pick the G that results in the best PSNR fidelity. In the case of BokehMe [17] there is an additional image for each G with activated H and the optimal combination of both is selected. This procedure for finding suitable user options is performed separately for each method and generally improves their performance by at least 1dB PSNR compared to a default value of $G = 2.2$.

The following are descriptions of the methods we have included in our Bokeh Rendering benchmark.

GRL [12] is a SOTA general image restoration model that uniquely models image hierarchies within their global, regional, and local ranges by combining a variety of transformer attention mechanisms. The specific version of the GRL architecture that we employed is GRL-S. Following the authors, we used Adam with a learning rate of $2e-4$, but the batch size had to be reduced to one due to memory limitations.

SwinIR [13] is a popular baseline image restoration model that implements a swin [14] transformer for image restoration. We adapt the lightweight version of this architecture and use Adam with a learning rate of $2e-4$ and trained with a reduced resolution of $384px$ and a batch size of two.

MambaIR [8] is a novel SOTA image restoration model that implements the idea of selective structured state space models for long-range dependency modeling. We adapt the configuration proposed for the real image de-noising task. Moreover, we had to reduce its embedding dimension from 48 to 32 and the number of blocks by half due to memory limits at high training and inference resolutions. This network was optimized with a learning rate of $2e-4$ using Adam, as suggested by the authors.

NAFNet [3] is a popular CNN-based baseline architecture for a wide variety of image restoration tasks that has previously been adapted for Bokeh rendering [11, 20]. We adopted the configuration of the model defined by the authors with a width of 32 channels and trained with a learning rate of $1e-3$ using Adam.

Restormer [30] is another popular transformer-based image restoration architecture that has been applied to a variety of tasks, including defocus deblurring and Bokeh rendering [29].

D2F [15] is a multi-step approach combining three sub-modules tailored specifically to Bokeh rendering. These modules are for defocus estimation, low-resolution weighted layered rendering with hand-crafted kernels following [2] and a deep poison fusion module for upscaling the image to its original resolution. Although an official training code for this method is not available for reference, we followed the procedure described by the authors [15], but using our own loss target.

BRViT [16] is a transformer-based SOTA Bokeh rendering method built on a Resnet-50 [9] feature extractor. Following the procedure described by the authors, we initially pretrain the network on input replication and then on the actual Bokeh rendering task. Adam with a learning rate of $1e-5$ and a batch size of one was used to optimize the model.

PyNET [10] is the pioneering neural Bokeh rendering architecture. We followed the bottom-up layer-wise training procedure as described by the authors using Adam, but with a reduced batch size to two. Note that since *RealBokeh* does not provide depth information, we used the version without depth guidance.

DMSHN [5] is an efficient CNN based Bokeh rendering method. We implemented the stacked version of this architecture as it shows the best results in the original proposal and used Adam for optimization.

DeepLens [25] is a multi-module neural Bokeh rendering framework that assembles separate depth prediction and neural lens blur models with guided upsampling for improved efficiency. Due to its integrated nature, it only requires the RGB image as input and, similarly to our method, does not rely on external resources during inference time. But unlike our proposal, DeepLens requires a dataset with additional accurate ground truth depth data for its intricate training protocol. As the collection of such a dataset in real-world conditions is problematic, its Bokeh is learned from the synthetic ray tracing based generator of [28].

Dr.Bokeh [21] is a multi-step rendering framework requiring depth input with additional salient detection and background-inpainting modules in its rendering pipeline. We followed the suggestion of the authors and used LDF [26] for salient detection and big-LaMa [23] for background in-painting.

MPIB [18] is a multi-step rendering framework requiring depth input and combining a Multiplane Image (MPI) [31] scene representation module with a background-inpainting [23] module.

BokehMe [17] is a multi-step rendering framework requiring depth input. It combines a simple classical rendering algorithm with a concurrent neural rendering. This neural renderer is used in difficult depth-discontinuous image areas where otherwise the classical renderer would bleed the bokeh from the background into the foreground.

F. Additional RealBokeh Examples

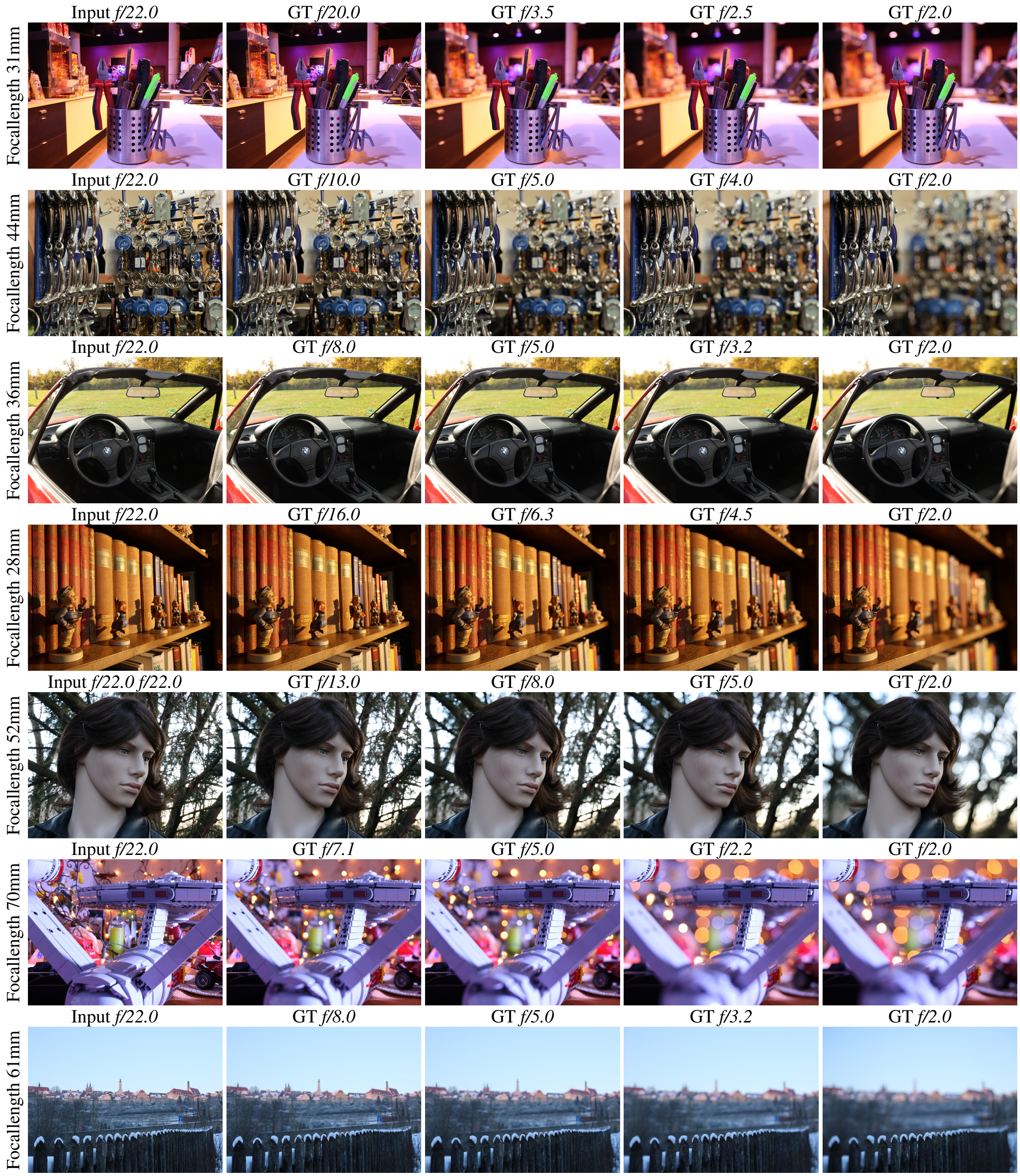


Figure E. **More examples from RealBokeh.** Note the high quality of **spatial and color alignment** between different aperture samples and the **high diversity of scene contents**.

G. Additional Qualitative Comparisons for methods without aperture control

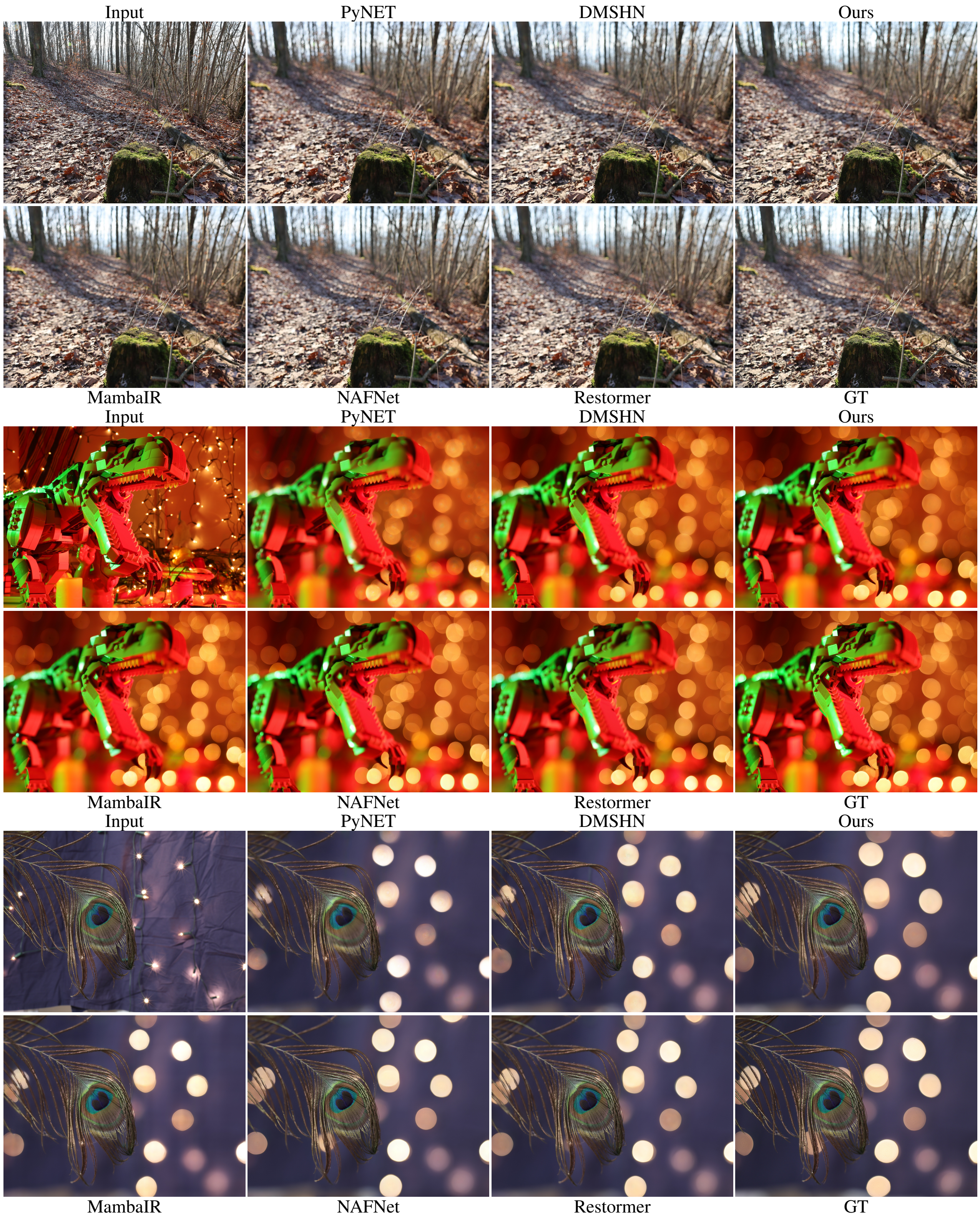


Figure F. **Additional Qualitative Comparisons between top methods** of our Benchmark in Tab. 4. Note how our method is better at **retaining fine foreground details** such as the small branches in the first scene, while our method produces a **more accurate Bokeh** in the second and third scene, particular when multiple Bokeh kernels interact with each other. Please zoom in to compare details.

H. Qualitative comparisons on *EBB! Val294*

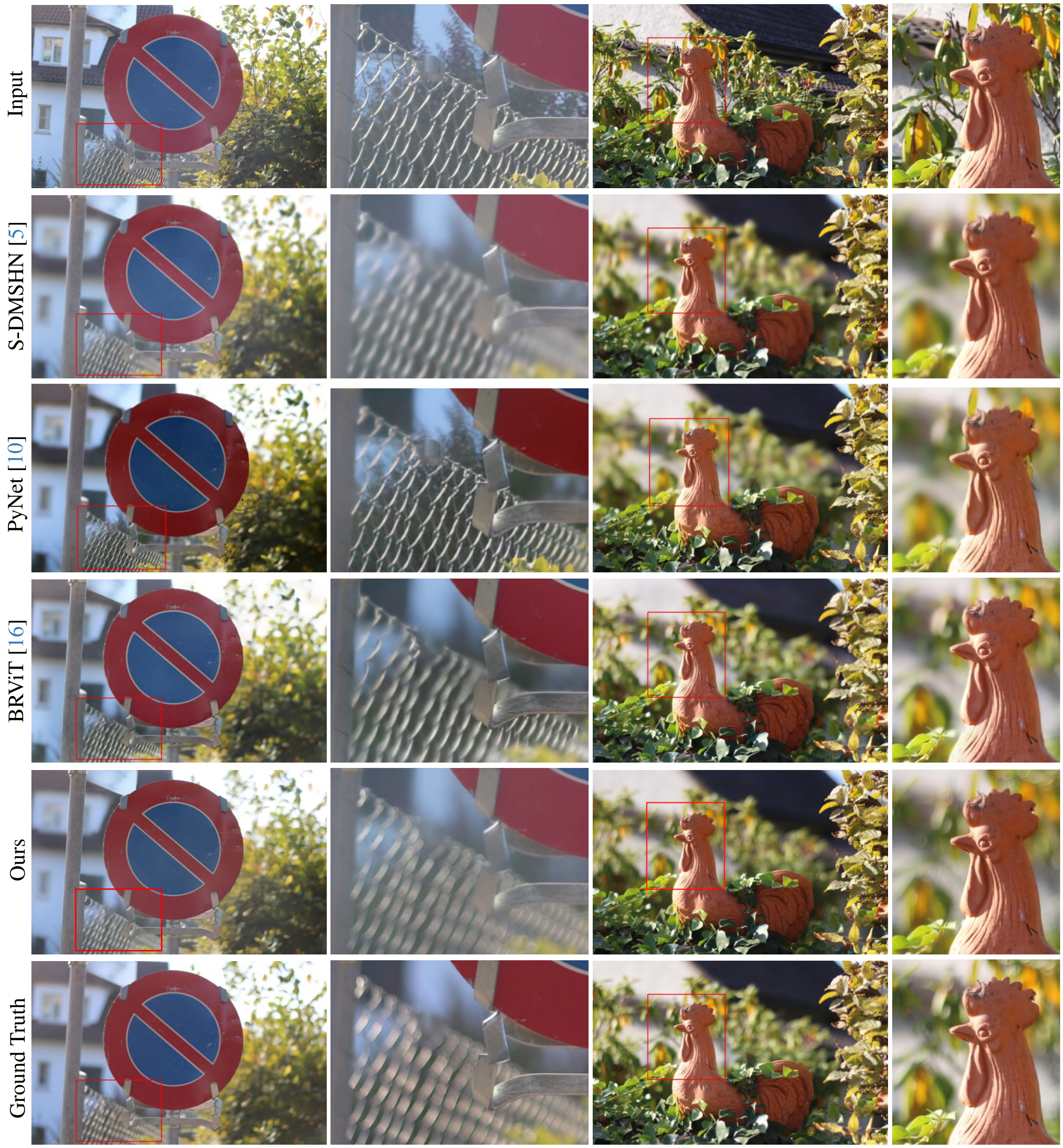


Figure G. **Visual comparison with additional methods of Bokehlicious on *EBB! Val294*.** Note that the Bokeh generated by our method **almost perfectly matches the visual appearance of the reference**. The left example also shows inconsistent colors between input and ground truth, one of the many problems of *EBB!*. Note that our method additionally does not change to the colors of the input.

I. Additional Qualitative Examples from EBB400

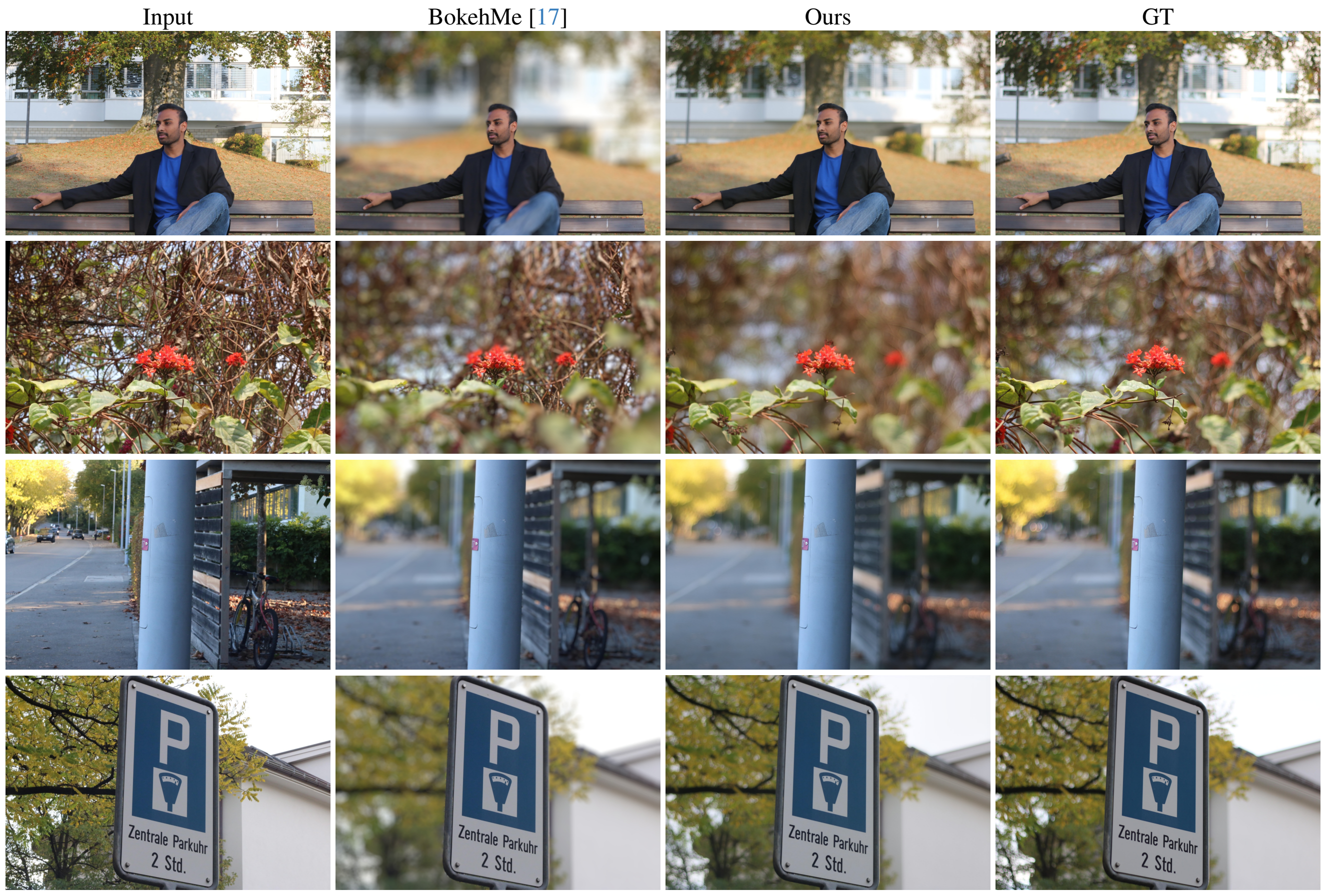


Figure H. **Additional qualitative samples on EBB400.** Note how ours achieves **better separation** between foreground and background, while also **closely matching style** of the Bokeh in the GT image. Please zoom in to compare details.

J. Effect of the loss weight λ

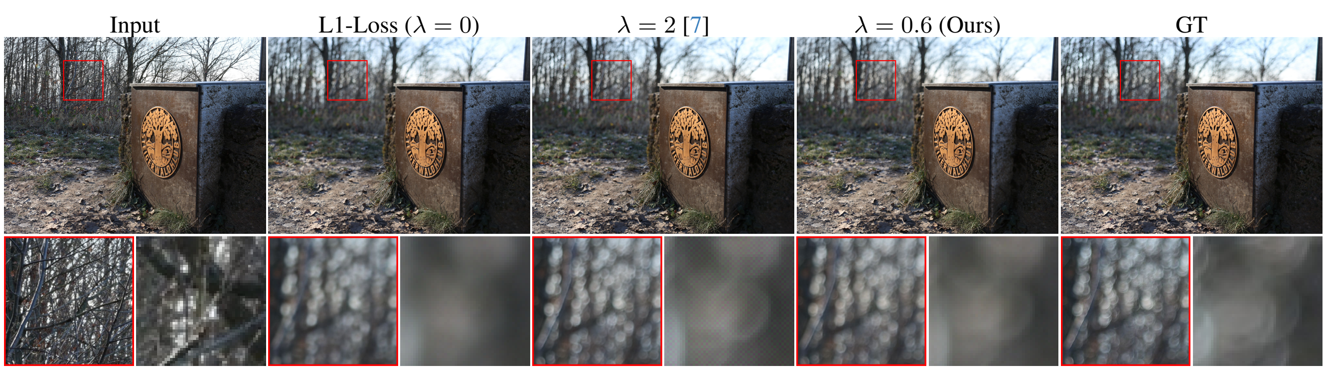


Figure I. Visual comparison of different choices for parameter λ in our proposed loss function Eq. (5). Note how a **large** λ results in a effect that is visually close to the GT, but unfortunately introduces **artifact patterns** (right crop). Our choice of $\lambda = 0.6$ combines the **artifact free** rendering of the L1 loss target, while **maintaining visual fidelity**. Please zoom in to compare details.

K. Full Table for Controllable Bokeh Rendering

Method	***f/2.0***			*f/2.2*			*f/2.5*			***f/2.8***			*f/3.2*		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Input	19.667	0.6447	0.5131	19.114	0.6663	0.4967	20.163	0.6771	0.4774	20.388	0.7015	0.4603	20.400	0.6836	0.4676
DeepLens [25]	23.069	0.8254	0.3449	22.292	0.8267	0.3526	23.379	0.8291	0.3358	23.177	0.8353	0.3243	23.833	0.8356	0.3168
Dr.Bokeh [21]	25.222	0.8970	0.2332	24.242	0.8947	0.2400	24.759	0.8972	0.2370	25.295	0.9027	0.2242	25.501	0.9053	0.2070
MPIB [18]	25.274	0.8980	0.2345	24.250	0.8952	0.2469	24.837	0.8970	0.2441	24.140	0.9023	0.2262	25.651	0.9066	0.2094
BokehMe [17]	26.151	0.9030	0.2144	25.612	0.8992	0.2192	26.450	0.9054	0.2089	26.811	0.9111	0.1988	27.032	0.9120	0.1884
Ours-M	29.636	0.9254	0.1173	29.369	0.9227	0.1158	29.903	0.9261	0.1120	30.872	0.9407	0.0939	30.858	0.9375	0.0952
Ours-L	30.883	0.9335	0.1095	30.355	0.9310	0.1068	31.021	0.9336	0.1049	32.180	0.9480	0.0882	32.224	0.9461	0.0886

Method	*f/3.5*			***f/4.0***			*f/4.5*			*f/5.0*			***f/5.6***		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Input	20.099	0.6883	0.4618	21.325	0.7251	0.4028	21.720	0.7097	0.3906	22.706	0.7426	0.3513	23.249	0.7581	0.3358
DeepLens [25]	23.284	0.8336	0.3216	24.359	0.8460	0.2842	24.580	0.8335	0.2711	25.790	0.8606	0.2435	25.535	0.8517	0.2473
Dr.Bokeh [21]	25.589	0.9037	0.2234	26.157	0.9026	0.2053	26.659	0.9028	0.1881	27.419	0.9130	0.1795	27.463	0.9099	0.1786
MPIB [18]	25.229	0.9037	0.2146	25.975	0.9044	0.1958	26.788	0.9069	0.1779	26.896	0.9112	0.1721	27.001	0.9045	0.1780
BokehMe [17]	26.825	0.9081	0.1963	27.272	0.9109	0.1771	27.865	0.9103	0.1611	28.189	0.9181	0.1510	28.451	0.9163	0.1512
Ours-M	31.016	0.9373	0.0924	31.750	0.9441	0.0801	31.965	0.9436	0.0740	32.470	0.9426	0.0768	32.969	0.9461	0.0721
Ours-L	31.999	0.9424	0.0899	32.810	0.9485	0.0796	33.191	0.9493	0.0720	33.520	0.9486	0.0730	34.126	0.9527	0.0690

Method	*f/6.3*			*f/7.1*			***f/8.0***			*f/9.0*			*f/10.0*		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Input	23.579	0.7657	0.3229	25.446	0.8206	0.2455	25.750	0.8102	0.2360	25.775	0.8090	0.2382	26.727	0.8285	0.2135
DeepLens [25]	26.318	0.8604	0.2306	27.471	0.8848	0.1874	28.048	0.8855	0.1738	28.172	0.8816	0.1790	28.882	0.8920	0.1645
Dr.B [21]	27.664	0.9084	0.1821	28.044	0.9224	0.1537	28.845	0.9238	0.1430	29.091	0.9246	0.1471	29.868	0.9330	0.1381
MPIB [18]	27.278	0.9090	0.1721	28.429	0.9246	0.1456	29.147	0.9263	0.1297	28.964	0.9192	0.1431	30.029	0.9301	0.1319
BokehMe [17]	28.587	0.9178	0.1544	29.770	0.9351	0.1229	30.456	0.9361	0.1127	30.081	0.9312	0.1231	31.127	0.9422	0.1174
Ours-M	33.364	0.9489	0.0656	33.388	0.9513	0.0623	34.660	0.9591	0.0457	34.453	0.9592	0.0463	36.165	0.9686	0.0363
Ours-L	34.487	0.9547	0.0634	34.462	0.9567	0.0593	35.466	0.9622	0.0466	35.464	0.9643	0.0453	37.156	0.9723	0.0371

Method	***f/11.0***			*f/13.0*			*f/14.0*			***f/16.0***			*f/18.0*		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Input	28.494	0.8651	0.1598	29.078	0.8874	0.1257	30.628	0.9046	0.1096	32.277	0.9274	0.0759	35.201	0.9559	0.0399
DeepLens [25]	30.045	0.9274	0.1201	30.835	0.9378	0.1028	31.383	0.9388	0.0982	32.031	0.9430	0.0827	33.374	0.9554	0.0571
Dr.Bokeh [21]	29.896	0.9329	0.1230	30.642	0.9417	0.1089	30.915	0.9430	0.0999	31.527	0.9470	0.0895	32.037	0.9560	0.0658
MPIB [18]	30.240	0.9293	0.1176	31.299	0.9415	0.0989	31.543	0.9412	0.0951	32.574	0.9468	0.0824	33.723	0.9570	0.0562
BokehMe [17]	31.275	0.9419	0.1013	31.526	0.9470	0.0914	32.595	0.9517	0.0861	32.941	0.9534	0.0729	34.585	0.9641	0.0522
Ours-M	36.044	0.9667	0.0343	35.201	0.9604	0.0381	37.135	0.9693	0.0284	36.979	0.9655	0.0271	38.202	0.9732	0.0195
Ours-L	36.956	0.9701	0.0342	35.890	0.9631	0.0381	37.693	0.9717	0.0289	37.437	0.9671	0.0287	38.809	0.9750	0.0198

Table A. **Performance on RealBokeh**. This is the extension of Tab. 5.

L. Uncropped Qualitative Examples from Fig.7

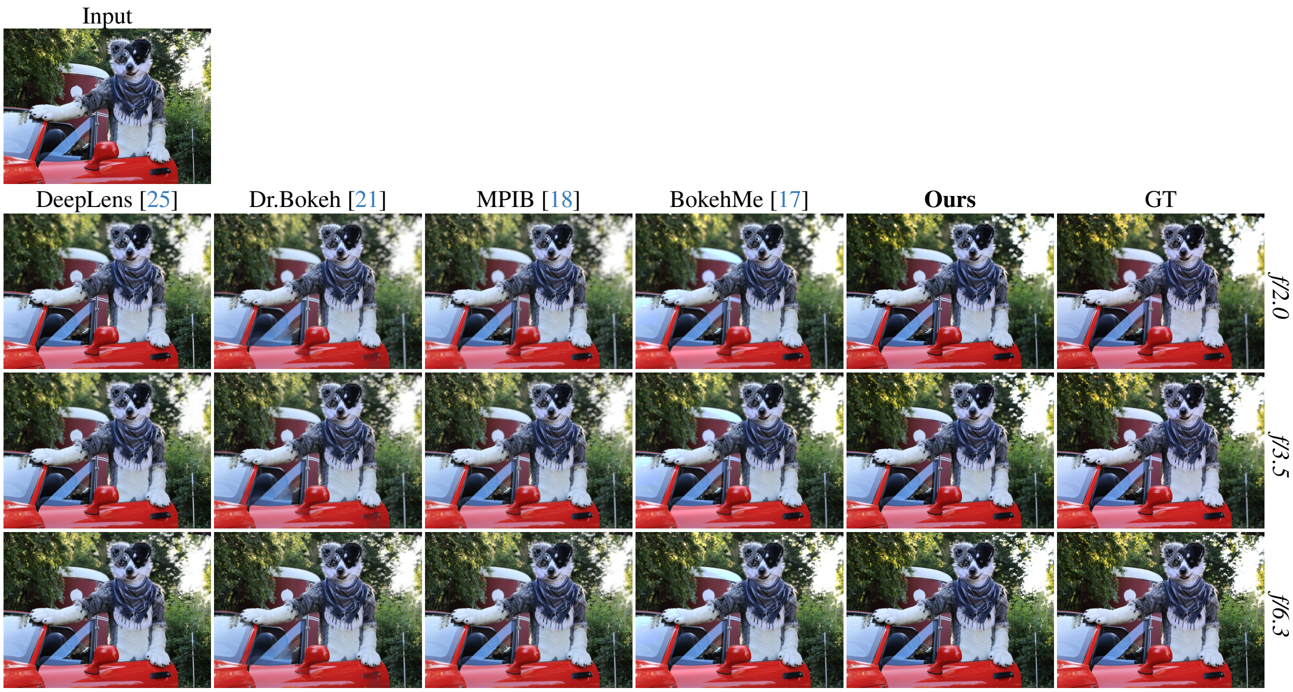


Figure J. **Uncropped version of the first qualitative comparison in Fig.7.** In the first example our method maintains the **critical sharpness** on the complex fur structure of the subject while **accurately rendering highlight intensity and and color**. One can also observe how the multi-step nature of other approaches can cause undesirable behavior, such as the door handle being removed by the background inpainting module of Dr.Bokeh [21]. Please zoom in to compare details.

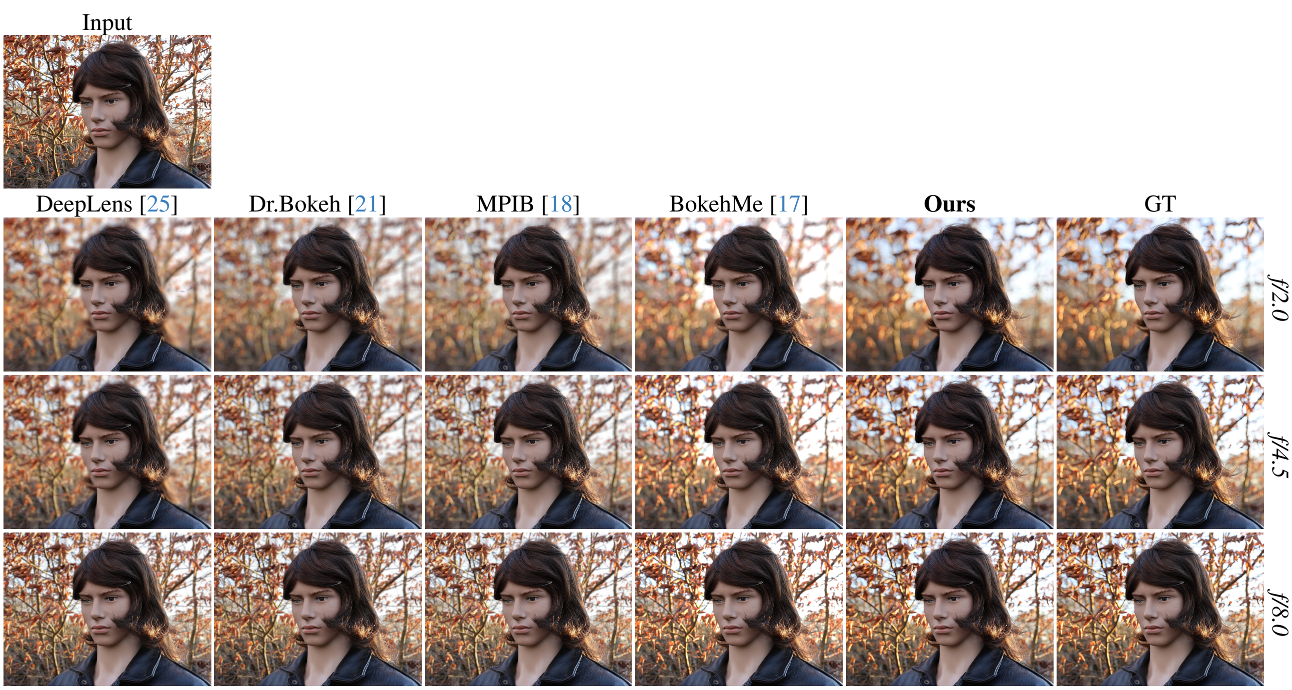


Figure K. **Uncropped version of the second qualitative comparison in Fig.7.** In the second example our method **precisely renders hair** while the contrast and saturation of the background remains **accurate** to the ground truth. Excluding BokehMe [17] this lack of image contrast is especially apparent in the competing solutions. Please zoom in to compare details.

M. Additional Qualitative Examples of Bokeh Rendering on RealBokeh

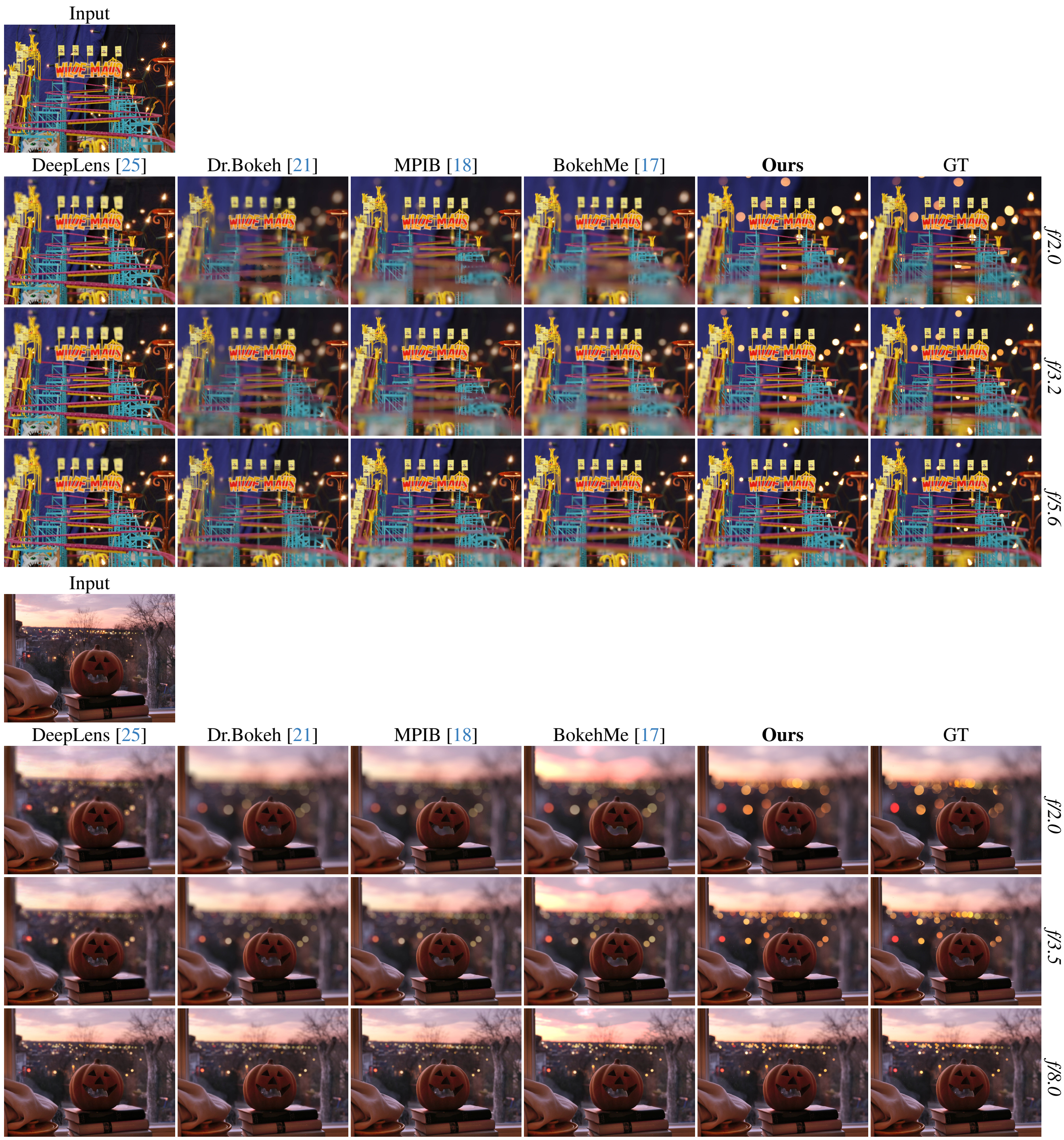


Figure L. Additional samples showing **Bokeh Rendering** on RealBokeh. In the first example our solution successfully **maintains critical sharpness** on the logo and flags while rendering a gradual falloff as the track gets closer to the camera. In the second ours renders more **accurate color and saturation** of the background lights. Note that DeepLens [25] is often unable to render strong bokeh effects and shows severe artifacts, this is in line with earlier evaluations of Peng *et al*. [17] on EBB400. Please zoom in to compare details.

N. Additional Comparisons with Syn-DoF

Input | Syn-DoF [24] | Ours (*f/4.0*) | Ours (*f/2.0*)

Figure M. More examples on **real-world portrait photography**. Note how our model produces more distinct Bokeh than Syn-DoF [24] (Google Portrait Mode) while improving the rendering of complex depth-discontinuities like hair and **enabling control** over the strength.

O. Application to Diverse Real World Images

Figure N. Additional results using **real-world** smartphone images [24]. Our model generalizes to diverse scenarios such as portraits, common objects, and complex scenes, without requiring depth map guidance.

P. Qualitative Comparison on RealDOF [1] defocus deblurring.

Figure O. **Defocus deblurring on the zero-shot RealDOF [1] Benchmark**. Our method generates results with **increased visual clarity** compared to previous SOTA methods. Please zoom in to compare details.

References

[1] Abdullah Abuolaim and Michael S Brown. Defocus deblurring using dual-pixel data. In *ECCV*, 2020. 14

[2] Benjamin Busam, Matthieu Hog, Steven McDonagh, and Gregory Slabaugh. Sterefo: Efficient image refocusing with stereo vision. In *ICCVW*, 2019. 4

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

[4] Saikat Dutta. Depth-aware blending of smoothed images for bokeh effect generation. *Journal of Visual Communication and Image Representation*, 77:103089, 2021. 1

[5] Saikat Dutta, Sourya Dipta Das, Nisarg A Shah, and Anil Kumar Tiwari. Stacked deep multi-scale hierarchical network for fast bokeh effect rendering from a single image. In *CVPRW*, 2021. 4, 7

[6] Qihang Fan, Huaibo Huang, Mingrui Chen, Hongmin Liu, and Ran He. Rmt: Retentive networks meet vision transformers. In *CVPR*, 2024. 1

[7] Konstantinos Georgiadis, Albert Saà-Garriga, Mehmet Kerim Yucel, Anastasios Drosou, and Bruno Manganelli. Adaptive mask-based pyramid network for realistic bokeh rendering. In *ECCVW*, 2023. 8

[8] Hang Guo, Jinmin Li, Tao Dai, Zhihao Ouyang, Xudong Ren, and Shu-Tao Xia. Mambair: A simple baseline for image restoration with state-space model. In *ECCV*, 2025. 4

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4

[10] Andrey Ignatov, Jagruti Patel, and Radu Timofte. Rendering natural camera bokeh effect with deep learning. In *CVPRW*, 2020. 4, 7

[11] Xiangyu Kong, Fan Wang, Dafeng Zhang, Jinlong Wu, and Zikun Liu. Nafbet: Bokeh effect transformation with parameter analysis block based on nafnet. In *CVPRW*, 2023. 4

[12] Yawei Li, Yuchen Fan, Xiaoyu Xiang, Denis Demandolx, Rakesh Ranjan, Radu Timofte, and Luc Van Gool. Efficient and explicit modelling of image hierarchies for image restoration. In *CVPR*, 2023. 4

[13] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *ICCV*, 2021. 4

[14] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 4

[15] Xianrui Luo, Juewen Peng, Ke Xian, Zijin Wu, and Zhiguo Cao. Defocus to focus: Photo-realistic bokeh rendering by fusing defocus and radiance priors. *Information Fusion*, 89: 320–335, 2023. 4

[16] Hariharan Nagasubramaniam and Rabih Younes. Bokeh effect rendering with vision transformers. *Authorea Preprints*, 2023. 4, 7

[17] Juewen Peng, Zhiguo Cao, Xianrui Luo, Hao Lu, Ke Xian, and Jianming Zhang. Bokehme: When neural rendering meets classical rendering. In *CVPR*, 2022. 1, 3, 4, 8, 9, 10, 11

[18] Juewen Peng, Jianming Zhang, Xianrui Luo, Hao Lu, Ke Xian, and Zhiguo Cao. Mpib: An mpi-based bokeh rendering framework for realistic partial occlusion effects. In *ECCV*, 2022. 3, 4, 9, 10, 11

[19] Lingyan Ruan, Bin Chen, Jizhou Li, and Miuling Lam. Learning to deblur using light field generated and real defocus images. In *CVPR*, 2022. 14

[20] Tim Seizinger, Marcos V Conde, Manuel Kolmet, Tom E Bishop, and Radu Timofte. Efficient multi-lens bokeh effect rendering and transformation. In *CVPRW*, 2023. 4

[21] Yichen Sheng, Zixun Yu, Lu Ling, Zhiwen Cao, Xuaner Zhang, Xin Lu, Ke Xian, Haiting Lin, and Bedrich Benes. Dr. bokeh: Differentiable occlusion-aware bokeh rendering. In *CVPR*, 2024. 3, 4, 9, 10, 11

[22] Hyeongseok Son, Junyong Lee, Sunghyun Cho, and Seungyong Lee. Single image defocus deblurring using kernel-sharing parallel atrous convolutions. In *ICCV*, 2021. 14

[23] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *WACV*, 2022. 4

[24] Neal Wadhwa, Rahul Garg, David E Jacobs, Bryan E Feldman, Nori Kanazawa, Robert Carroll, Yair Movshovitz-Attias, Jonathan T Barron, Yael Pritch, and Marc Levoy. Synthetic depth-of-field with a single-camera mobile phone. *ACM Transactions on Graphics (ToG)*, 37(4):1–13, 2018. 1, 12, 13

[25] Lijun Wang, Xiaohui Shen, Jianming Zhang, Oliver Wang, Zhe Lin, Chih-Yao Hsieh, Sarah Kong, and Huchuan Lu. Deeplens: shallow depth of field from a single image. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018. 4, 9, 10, 11

[26] Jun Wei, Shuhui Wang, Zhe Wu, Chi Su, Qingming Huang, and Qi Tian. Label decoupling framework for salient object detection. In *CVPR*, 2020. 4

[27] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024. 3

[28] Yang Yang, Haiting Lin, Zhan Yu, Sylvain Paris, and Jingyi Yu. Virtual dslr: High quality dynamic depth-of-field synthesis on mobile platforms. *Electronic Imaging*, 28:1–9, 2016. 4

[29] Zhihao Yang, Wenyi Lian, and Siyuan Lai. Bokehornot: Transforming bokeh effect with image transformer and lens metadata embedding. In *CVPRW*, 2023. 4

[30] 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

[31] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018. 4