MPIB: An MPI-Based Bokeh Rendering Framework for Realistic Partial Occlusion Effects

Juewen Peng[1], Jianming Zhang[2], Xianrui Luo[1], Hao Lu[1], Ke Xian[1]*, and Zhiguo Cao[1]

[1] Key Laboratory of Image Processing and Intelligent Control, Ministry of Education, School of AIA, Huazhong University of Science and Technology, China
{juewenpeng,zgcao,xianruiluo,hlu,kexian}@hust.edu.cn
[2] Adobe Research
jianmzha@adobe.com
https://github.com/JuewenPeng/MPIB

Abstract. Partial occlusion effects are a phenomenon that blurry objects near a camera are semi-transparent, resulting in partial appearance of occluded background. However, it is challenging for existing bokeh rendering methods to simulate realistic partial occlusion effects due to the missing information of the occluded area in an all-in-focus image. Inspired by the learnable 3D scene representation, Multiplane Image (MPI), we attempt to address the partial occlusion by introducing a novel MPI-based high-resolution bokeh rendering framework, termed MPIB. To this end, we first present an analysis on how to apply the MPI representation to bokeh rendering. Based on this analysis, we propose an MPI representation module combined with a background inpainting module to implement high-resolution scene representation. This representation can then be reused to render various bokeh effects according to the controlling parameters. To train and test our model, we also design a ray-tracing-based bokeh generator for data generation. Extensive experiments on synthesized and real-world images validate the effectiveness and flexibility of this framework.

Keywords: Bokeh Rendering, Multiplane Image, Partial Occlusion

1 Introduction

Bokeh effect is commonly used in photography to create an appealing blur effect in out-of-focus areas and make the subject stand out from the picture. Various post-processing methods, which simulate the bokeh effect from an all-in-focus image, have been proposed so far. However, as shown in Fig. 1, neither the physically based ones (*e.g.*, SteReFo [3]) nor the deep learning-based ones (*e.g.*, DeepLens [39]) can well handle the issue of partial occlusion: the out-of-focus

* Corresponding author

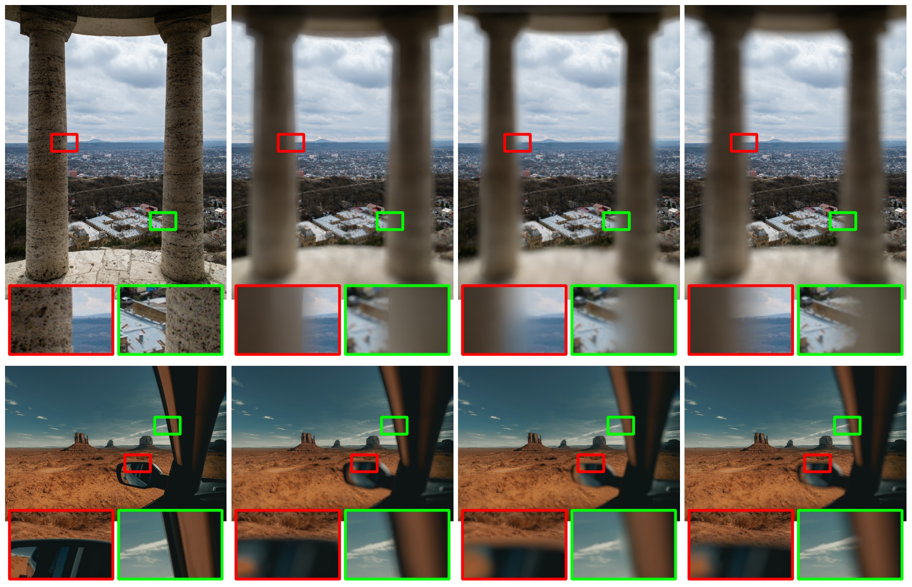

Fig. 1. MPIB renders more realistic partial occlusion effects than other methods at the boundary between foreground and background. Best viewed by zooming in.

object close to the camera becomes blurred and semi-transparent, revealing the occluded in-focus area.

In the field of computer graphics, Schedl and Wimmer [31] first attempt to solve this problem by decomposing the scene into different depth layers, and blurring each layer with a fixed-size filter before compositing them together. This strategy works well given the complete scene information, however, it is less effective for methods with a single image as input [3,48]. A recent idea relevant to this layering strategy is Multiplane Image (MPI) proposed for novel view synthesis [50]. MPI aims to learn a 3D scene representation with multiple RGBA planes from a single image, which can then be used to synthesize different novel views of the scene. Despite its desirable characteristics of explicitly modeling occluded areas for each plane, we observe that current MPI-based view synthesis methods [16,35,37,50] are still inadequate in restoring the occluded surface with complicated textures.

To apply MPI to bokeh rendering and address the above challenges, we first analyse the difference of MPI representation and layer compositing formulation between the view synthesis and bokeh rendering. Then, based on this analysis, we propose a novel MPI-based framework, termed MPIB, for synthetic bokeh effect. Specifically, we combine an MPI representation module with a background inpainting module to obtain a high-resolution 3D scene representation. The background inpainting module aims to synthesize convincing contents in the occluded areas and lighten the burden of the scene representation. Once the scene representation is obtained, it can be reused to render different bokeh effects with adjustable controlling parameters, such as blur amount and refocused disparity.

Due to the difficulty of capturing accurate pairs of all-in-focus image and bokeh image in the real world, we design a ray-tracing-based bokeh generator for training. We also use it to synthesize a test dataset for the initial validation. Since the judgement of the bokeh effect is really subjective, we further conduct a user study on images collected from websites and compare our results with the latest iPhone 13 Cinematic Mode. Experimental results show that MPIB renders realistic partial occlusion effects and yields substantial improvements over state-of-the-art methods.

In summary, our main contributions are as follows.

- We present an analysis on how to apply the MPI representation and the layer composting scheme to bokeh rendering.
- We propose an MPI-based framework for high-resolution bokeh rendering and realistic partial occlusion effects.
- We design a ray-tracing-based bokeh generator, which creates almost real bokeh effects and can be used to produce training and test data.

2 Related Work

Bokeh Rendering. Bokeh rendering techniques can be classified into physically based methods and neural rendering methods. In the early years, most physically based methods [1,14,31,40,46] entail 3D scene information and are time-consuming. Recent methods [2,3,7,26,32,33,38,41,44,48], which render bokeh effects from a single image and a corresponding depth map, are more efficient and practical. One classic idea is layered rendering [3,48], *i.e.*, decomposing the scene into multiple layers and independently blurring each layer before compositing them from back to front. However, they do not consider potential depth inaccuracy and object occlusion, resulting in unrealistic partial occlusion effects.

To solve these problems, many neural rendering methods have been proposed recently. Xiao *et al.* [42] specialize in using a perfect depth map to render realistic bokeh effects in low resolution. Considering the difficulty of obtaining perfect depth maps in the real world, Wang *et al.* [39] propose a robust rendering system consisting of the depth prediction, lens blur, and guided upsampling modules. To further simplify the rendering process, many end-to-end networks [5,10,11,9,28] are proposed. They simulate the bokeh effect of DSLR cameras from a single wide depth-of-field image without inputting the depth map or any other controlling parameters and show a compelling performance on the bokeh dataset EBB! [9]. However, the simplicity comes at a cost. These networks are lack of flexibility. They cannot adjust bokeh effects, such as different blur amounts and focal planes. In this work, we focus on controllable bokeh rendering with an all-in-focus image, a potentially imperfect disparity map and some controlling parameters as input.

MPI Representation. Since Zhou *et al.* [50] first propose MPI to reconstruct the camera frustum from stereo images and synthesize novel views, this representation has been widely used in novel view synthesis methods [16,20,35,37,51] due to the appealing property of differentiability and explicitly modeling occluded contents. Srinivasan *et al.* [35] provide a theoretical analysis of MPI limits, and use the appearance flow [51] to improve the realism in occluded areas. Tucker *et al.* [37] apply the MPI representation to the single-view inputting case which is more practical but more challenging. Li *et al.* [16] propose an encoder-decoder architecture which is a continuous depth generalization of MPI by introducing the idea of neural radiance fields (NeRF) [21]. Different from the above works, we embed an additional disparity map into the model aside from the all-in-focus image and apply a lightweight guided upsampling network. It aims to enhance the generalization of the model and obtain the high-resolution scene representation with accurate and sharp object boundaries. In addition, we supervise our model with bokeh images instead of synthesized images in different views.

Image Inpainting. The goal of inpainting is to fill in missing contents of an image. Compared with traditional patch-based [4] and nearest neighbor-based [8] methods, neural networks have much stronger ability to capture spatial context of images and generate plausible contents for unseen parts, espcially after the introduction of adversarial training [6]. Therefore, deep learning-based inpainting has attracted a lot of attention and many CNN-based methods have been proposed [12,25,43]. Recent works attempt to replace the regular convolutions with partial [18], gated [45] or fourier [36] convolutions, and design new architectures [19,22,52] to address irregular masks and high-resolution inpainting. In this work, we utilize the off-the-shelf inpainting model [36] to facilitate our MPI representation and create more convincing partial occlusion effects.

3 MPIB: An MPI-Based Bokeh Rendering Framework

As shown in Fig. 2, our framework consists of 3 modules: background inpainting, MPI representation, and bokeh rendering. In the following, we first analyse how to apply MPI in bokeh rendering and introduce our rendering formula (Sec. 3.1). Then, we describe the structures of our background inpainting module (Sec. 3.2) and MPI representation module (Sec. 3.3). Finally, we provide the training details, including the proposed ray-tracing-based bokeh generator (Sec. 3.4).

3.1 MPI Representation and Layer Compositing Formulation

The well-known multiplane image (MPI) representation, introduced by Zhou *et al.* [50], consists of a set of fronto-parallel planes. Each plane encodes an RGB color image c_i and an alpha map α_i, *i.e.*, $\{(c_i, \alpha_i) \,|\, i = 1, 2, ..., N\}$, where N is the number of MPI planes. When applying the MPI representation to bokeh rendering, we make minor modifications. As proven in [38,44], the blur radius r of each pixel is

$$r = A \left| \frac{1}{z} - \frac{1}{z_f} \right| = A \left| d - d_f \right|, \tag{1}$$

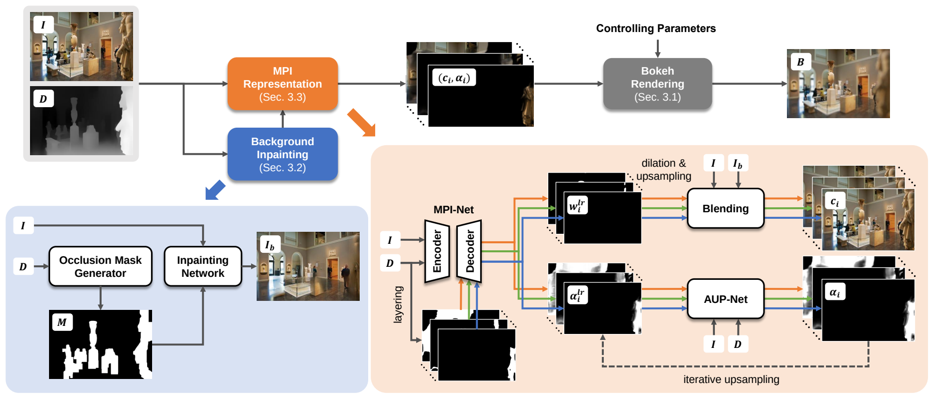

Fig. 2. Our framework MPIB takes an all-in-focus image and a potentially imperfect disparity map as input to obtain a 3D scene representation of multiple RGBA planes. This representation is generated by an MPI representation module and a background inpainting module. Then, the scene representation can be reused to produce multiple bokeh images according to different controlling parameters.

where A reflects the overall blur amount of the image. z is the depth of the pixel and z_f is the refocused depth. We replace the depth z with the disparity (inverse depth) d to simplify the formula. One can see that r is the linear variation of d. For uniform sampling of the blur amount across different planes, we replace depth discretization in general MPI representation with disparity discretization.

In novel view synthesis, one can reconstruct the scene I by continuously using the "over" alpha compositing operation [15,27] to MPI planes with a back-to-front order:

$$I = \sum_{i=1}^{N} \left(c_i \alpha_i \prod_{j=i+1}^{N} (1 - \alpha_j) \right) . \tag{2}$$

When applying Eq. 2 to the bokeh rendering, we blur each layer with a fixed-size kernel K_i, which is adaptive to the controlling parameters. The rendered bokeh image B can be formulated as

$$B = \sum_{i=1}^{N} \left((c_i \alpha_i * K_i) \prod_{j=i+1}^{N} (1 - \alpha_j * K_j) \right) , \tag{3}$$

where $*$ is the convolution operation. To eliminate the artifacts caused by discretization, we add extra weight normalization to Eq. 3 as follows:

$$B = \frac{\sum_{i=1}^{N} \left((c_i \alpha_i * K_i) \prod_{j=i+1}^{N} (1 - \alpha_j * K_j) \right)}{\sum_{i=1}^{N} \left((\alpha_i * K_i) \prod_{j=i+1}^{N} (1 - \alpha_j * K_j) \right)} . \tag{4}$$

Typically, the transformation from scene irradiance to RGB values is nonlinear [44], so we conduct the gamma transformation to c_i and the inverse gamma transformation to B following common practice.

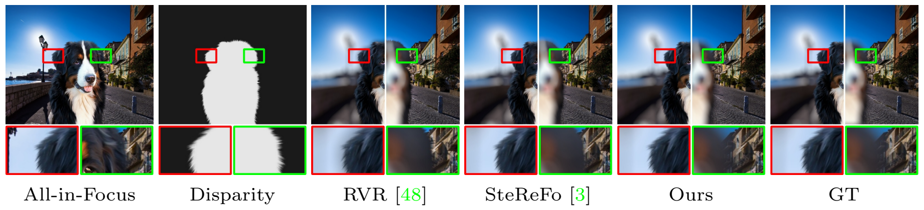

Fig. 3. Toy experiment. The scene is synthesized from two images with constant disparities. "GT" refers to the result of the ray-tracing-based bokeh generator proposed in Sec. 3.4. For each rendered bokeh image, the left half is refocused on the foreground while the right half is refocused on the background.

Previous layered rendering methods, such as RVR [48] and SteReFo [3], adopt a similar compositing strategy. However, they only consider visible parts of the image and calculate the alpha map of each plane manually. For clarity, we conduct a toy experiment in Fig. 3 where only two planes are used. We compare our approach with RVR, SteReFo, and the bokeh generator proposed in Sec. 3.4 (regarded as the ground truth). Note that the result of our approach is in an ideal situation with known background information. One can see that our result is most similar to the ground truth, and the occluded background is partly visible at the boundary of the blurred foreground, demonstrating the plausibility of our rendering formula and the importance of predicting occluded contents.

3.2 Background Inpainting Module

In background inpainting module, we combine an off-the-shelf inpainting model LaMa [36] with an occlusion mask generator to produce a background image. Since occlusion often occurs where the disparity of the scene changes significantly and the occluded area is on the side with a larger disparity, we design the occluded mask generator according to this principle.

At the beginning, we use the Sobel operator to calculate a 2-channel gradient map $G = \{G_x, G_y\}$ of the input disparity map D. By thresholding the gradient magnitude, we obtain a depth discontinuity mask and regard it as the initial occlusion mask M. Then, we remove short segments of M and G to reduce the impact of noise. Subsequently, we iteratively extend M in the direction of gradient increase. In each iteration, we first perform ℓ_2 normalization to G to obtain unit normal vectors G^n. Then, we forward warp G^n via the softmax splatting operation [23] using the same G^n as the optical flow to get the inner ring of G^n, and represent it with G^w. At last, we update G by

$$G = M \cdot G^n + (1 - M) \cdot G^w \,, \tag{5}$$

and update M to refer to the nonzero areas of G. In practice, as the input disparity map may not align well with the all-in-focus image at occluding boundaries, we finally dilate M by several pixels as processed in [34] to prevent the foreground color leakage and reduce inpainting artifacts. For ease of understanding, we visualize the process of producing an occlusion mask in Fig. 4.

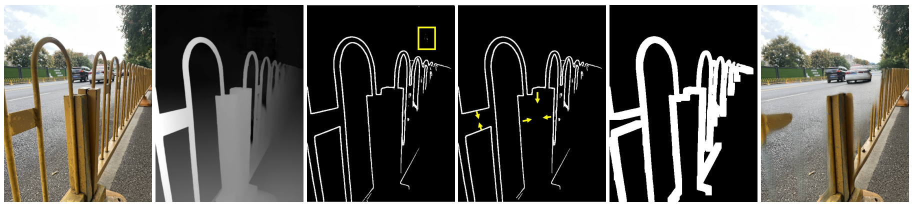

Fig. 4. Generation of the occlusion mask and the inpainted background image. "Mask_1" is calculated by thresholding the gradient of the disparity map. "Mask_2" removes the short segments of "Mask_1". "Mask_3" extends the area of "Mask_2" in the direction of disparity increase. "Inpainted BG" is predicted by the inpainting method LaMa [36] using the all-in-focus image and the produced occlusion mask as input.

Note that for complicated scenes, we can restrict the area of occlusion mask and predict more background images for MPI planes in different disparity levels. However, we show in Sec. 4.2 that one background image is sufficient for rendering real-world images in general.

3.3 High-resolution MPI Representation Module

In MPI representation module, we first propose an encoder-decoder network MPI-Net to obtain an initial MPI representation as with [16]. For each image, the encoder runs only once, while the decoder runs multiple times to generate N planes. However, there are two main differences to be aware of. (i) We embed the disparity map D into the decoder instead of the discrete disparity values to pass more prior knowledge to the network, which will lead to the better layering and stronger generalization. Specifically, assume D is ranged from 0 to 1, we divide the range by $\{[\frac{i-1}{N}, \frac{i}{N}] \,|\, i = 1, 2, ..., N\}$, and calculate the coarse zone mask for each plane. Subsequently, we apply a single convolution layer to each zone mask to increase channels before feeding them into the decoder. (ii) To reduce the difficulty of training MPI-Net, we assume that the RGB image c_i of each plane is the per-pixel weighted average of the original all-in-focus image I and the inpainted background image I^b. Thus, for each plane, MPI-Net only predicts an alpha map α_i and a blend weight map w_i, and c_i can be produced by

$$c_i = w_i \cdot I^b + (1 - w_i) \cdot I \,. \tag{6}$$

To adapt our model to high-resolution input, we also propose a lightweight guided upsampling network AUP-Net to iteratively upsample the alpha map α_i by a factor of 2. Note that AUP-Net is guided by the high-resolution all-in-focus image I and its corresponding disparity map D. As for the blend weight map w_i, we just perform slight dilation and bilinear upsampling. The reason for the dilation here is similar to the reason we dilated the occlusion mask earlier in Sec. 3.2. Although this operation may cause a small amount of visible information loss on c_i, it effectively prevents foreground colors from bleeding into background planes and reduces the risk of unpleasant boundary artifacts.

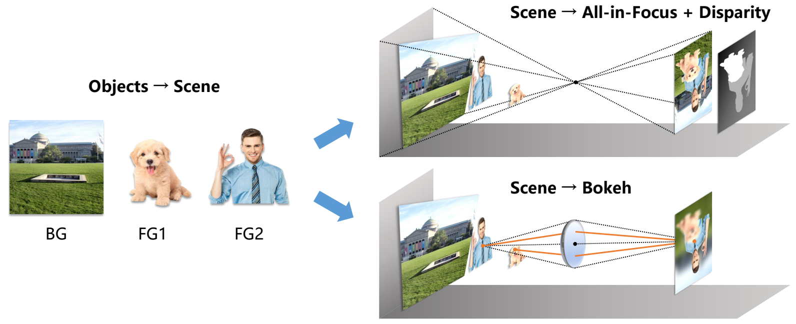

Fig. 5. Pipeline of ray-tracing-based bokeh generator. Given a background RGB image and two foreground RGBA images, we first construct a 3D scene through randomly setting the positions and disparities of different objects. Then, we can synthesize an all-in-focus image and a corresponding disparity map by compositing different object planes, and generate a bokeh image by backward tracing the path of light rays.

3.4 Model Training

Bokeh Generator and Training Data. To obtain ground-truth bokeh images, we design a ray-tracing-based bokeh generator (Fig. 5), which is capable of creating a bokeh image from a background RGB image and some foreground RGBA images with customized disparities and different controlling parameters. Specifically, for each image to be composited, we set its disparity map d as a plane equation of pixel coordinates (x, y):

$$d = \frac{1 - ax - by}{c}\,, \tag{7}$$

where a, b, c are three constants. We project all images to a 3D scene based on their disparities, and treat them as different objects. For each pixel (x, y) of the rendered result, we sample 2500 rays passing through the aperture, look for the intersection of each ray and the scene, and project the intersection to the sensor plane. The detailed calculation is in the supplementary material. Here, we directly provide the coordinates (x_n, y_n) of the projected intersection:

$$\begin{cases} x_n = x + \dfrac{1 - ax - by - cd_f}{aA\mu + bA\nu + c} A\mu\,, \\ y_n = y + \dfrac{1 - ax - by - cd_f}{aA\mu + bA\nu + c} A\nu\,, \end{cases} \tag{8}$$

where A is the blur parameter. d_f is the refocused disparity. (μ, ν) denotes the sampling point within the aperture, which meets $\mu^2 + \nu^2 \leq 1$. (x_n, y_n) is available if and only if it is inside the object. Specially, if (x_n, y_n) happens to be on the boundary, we suppose the ray with a proportion equal to the alpha value of (x_n, y_n) intersects the current plane, and the ray with the remaining proportion continues to propagate. In summary, we calculate the intersection of the ray and each plane from front to back, and the search process stops until the energy of the ray is exhausted. The final rendered result of (x, y) is the weighted average of the colors for all intersections.

In practice, we first randomly select 20k images from Places [49] as our background images. The foreground images originate from two sources. One is PhotoMatte85 [17] with 85 portrait RGBA images. The other is websites. We collect 300 images with pure background and extract their alpha maps by matting tools. During the training, each sample is synthesized online from 2 random foreground images and 1 background image with the resolution of 256×256. The disparity map is set within the range from 0 to 1. The ground-truth bokeh image is produced with the random blur parameter from 0 to 32, refocused disparity from 0 to 1, and gamma value from 1 to 4. To improve the generalization of the model, we augment the input disparity map with random noise, gaussian blur, dilation and erosion. We also use the intermediate synthesized results instead of the inpainted results as I^b to simplify and accelerate the training process.

Loss Functions. The training of MPI-Net and AUP-Net are independent. For MPI-Net, we use the following loss:

$$\mathcal{L}_{MPI} = \mathcal{L}_{bokeh} + 0.4\,\mathcal{L}_{disp} + 0.1\,\mathcal{L}_{alpha} + 0.1\,\mathcal{L}_{weight}\,, \tag{9}$$

where $\mathcal{L}_{bokeh}$ is a ℓ_1 loss, which encourages the rendered bokeh image B to match the ground truth B^* in both image space and gradient space. $\mathcal{L}_{disp}$ has the same form with $\mathcal{L}_{bokeh}$, which enforces the reconstructed disparity map D^{rc} to be consistent with the raw input disparity map D^* without augmentation. This term aims to assist MPI representation learning and improve the sensitivity of the model to object boundaries. As with [37], the reconstructed disparity is defined by

$$D^{rc} = \sum_{i=1}^{N} \left(d_i \alpha_i \prod_{j=i+1}^{N} (1 - \alpha_j) \right), \tag{10}$$

where $d_i = \frac{i-0.5}{N}$, which represents the average disparity of each plane. Next, $\mathcal{L}_{alpha}$ and $\mathcal{L}_{weight}$ are two ℓ_1 regularization losses, which constrain the predicted alpha maps and blend weight maps to be smooth, respectively.

For training AUP-Net, we use a similar loss:

$$\mathcal{L}_{AUP} = \mathcal{L}_{bokeh} + 0.4\,\mathcal{L}_{disp} + 0.1\,\mathcal{L}_{alpha} + 0.4\,\mathcal{L}_{self}\,. \tag{11}$$

$\mathcal{L}_{weight}$ is not used here because AUP-Net only processes alpha maps. Instead, we add a self-supervised loss $\mathcal{L}_{self}$ to accelerate the convergence and prevent the upsampled alpha map α_i and the input low-resolution alpha map α_i^{lr} from being too different.

$$\mathcal{L}_{self} = \mathcal{L}_{\ell_1}\left(\text{downsample}(\alpha_i), \alpha_i^{lr}\right). \tag{12}$$

Implementations. We implement our model by PyTorch [24]. The number of MPI planes is set to 32. In MPI-Net, we use ResNet-18 as the encoder, and use a decoder similar to [16]. AUP-Net is a lightweight network based on U-Net [30]. We show its detailed architecture in the supplementary material. When training AUP-Net, we forcely downsample the input by a factor of 2 before applying MPI-Net, and use AUP-Net to upsample the initial rendered result up to the original resolution. Both networks are trained for 40 epochs using the Adam optimizer [13] with a learning rate of 10^{-4}, and a batch size of 8. All experiments are conducted on four NVIDIA GeForce GTX 1080 Ti GPUs.

4 Experiments

4.1 Bokeh Rendering on Synthesized Dataset

Dataset. Current public bokeh datasets including EBB! [9], Aperture [48] and a Unity synthesized dataset [42] are unsuitable for evaluations of controllable bokeh rendering. The first two datasets are manually captured by a DSLR camera with unknown controlling parameters. There exist color inconsistency and scene misalignment between the wide and shallow DoF image pairs, and almost all images are focused on the front subject. For the last synthesized dataset, all scenes are built from randomized object geometries, so there is a huge gap between them and real-world images. In addition, the maximum blur size of this dataset is too small. As a result, we use our ray-tracing-based generator to synthesize a test dataset, which contains 100 new scenes (not the same with our training data) with the resolution of 1024×1024. Each scene is synthesized from 1 background image and 3 foreground images. Unlike [42], all images are derived from the real world. We provide two versions for disparity settings. In the first version, the disparity of each object is set to a constant value, while in the second version, it is set to a smoothly varying plane. Besides, for each all-in-focus image, we synthesize 16 bokeh images with 4 blur parameters from 20 to 80 and 4 refocused disparities, which correspond to the disparities of the 4 objects. The gamma values are set to 2.2 for all settings.

Metrics. To measure performance, we use LPIPS [47], PSNR and SSIM as metrics. Since the difficulty of bokeh rendering is mainly concentrated at occluding boundaries and the human eye is more sensitive to these areas, we additionally introduce two metrics PSNR_{ob} and SSIM_{ob}, which only reflect the accuracy at occluding boundaries. More details are in the supplementary material.

Compared Methods. We compare MPIB with 4 physically based rendering methods, including 2 pixel-wise rendering methods: Gather [38] and Scatter [38], and 2 layered rendering methods: RVR [48] and SteReFo [3]. We also test 2 neural rendering methods: DeepLens [39] and DeepFocus [42]. Note that as DeepFocus [42] cannot handle large blur sizes, directly applying it to high resolution will cause the collapse of the model. Thus, we downsample the input before feeding it into the network, and the rendered result is bilinearly upsampled without refining. More discussion about DeepFocus is in the supplementary material. To be

Table 1. Quantitative results on the synthesized dataset. N_{bg} denotes the number of the inpainted images used in our background inpainting module. The best performance is in **boldface**, and the second best is underlined.

Method	Constant disparity for each object					Varying disparity for each object				
	LPIPS↓	PSNR↑	PSNR_{ob}↑	SSIM↑	SSIM_{ob}↑	LPIPS↓	PSNR↑	PSNR_{ob}↑	SSIM↑	SSIM_{ob}↑
Gather [38]	0.042	32.6	25.4	0.978	0.896	0.044	33.0	25.9	0.979	0.903
Scatter [38]	0.029	33.9	26.7	0.983	0.921	0.027	34.8	27.8	0.985	0.934
RVR [48]	0.103	28.8	21.9	0.951	0.783	0.113	28.9	22.0	0.951	0.787
SteReFo [3]	0.038	32.8	26.4	0.976	0.923	0.040	32.5	26.3	0.973	0.928
DeepLens [39]	0.068	29.5	24.9	0.945	0.897	0.055	29.8	24.9	0.957	0.911
DeepFocus [42]	0.083	33.4	30.3	0.938	0.923	0.063	35.2	31.9	0.968	0.952
Ours ($N_{bg} = 1$)	0.011	36.7	30.0	0.989	0.951	0.019	36.8	30.5	0.986	0.956
Ours ($N_{bg} = 3$)	**0.008**	**39.3**	**33.0**	**0.991**	**0.963**	**0.017**	**38.8**	**33.0**	**0.988**	**0.966**

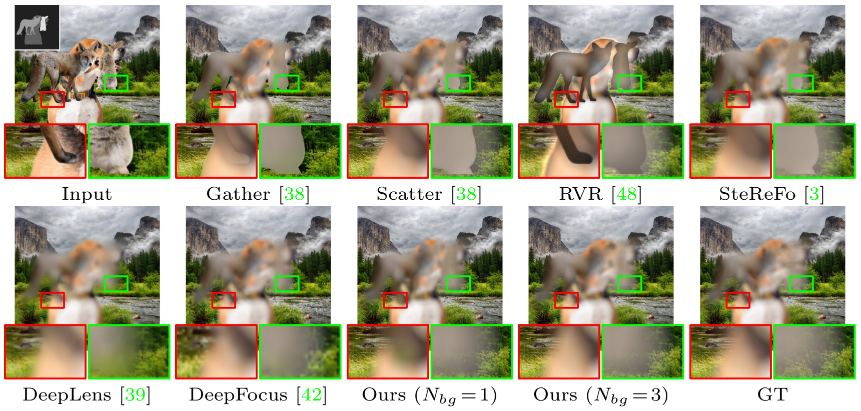

Fig. 6. Qualitative results on the synthetic dataset.

fair, we discard the depth estimation modules in SteReFo [3] and DeepLens [39], and only preserve their rendering modules. Then, all methods can take the same all-in-focus image, disparity map and controlling parameters as input.

Comparison with State-of-the-Art. We show quantitative and qualitative results in Table 1 and Fig. 6. One can observe: (i) Among the physically based rendering methods, pixel-wise methods (Gather [38], Scatter [38]) perform well on the overall PSNR and SSIM due to the accurate bokeh rendering in depth-continuous areas, but cause serious color bleeding artifacts at occluding boundaries. For layered methods, SteReFo [3] is much better than RVR [48] as SteReFo uses the weight normalization during the rendering. Without this operation, halo artifacts will occur at occluding boundaries, just like RVR. (ii) Neural rendering methods render relatively smooth bokeh effects at boundaries. However, DeepLens [39] does not perform very well in metrics. The reason may be that its training data is synthesized by VDSLR [44], which applies manual color enhancement and is inconsistent with real rendering rules. DeepFocus [42] gets high PSNR_{ob} and SSIM_{ob}, but it processes images in low resolution, which leads to

Table 2. Pairwise comparison results of the user study. A number of more than 50% indicates that our approach gets more votes than the other method.

Method	Refocusing on the foreground				Refocusing on the background			
	Scatter[38]	SteReFo[3]	DeepLens[39]	DeepFocus[42]	Scatter[38]	SteReFo[3]	DeepLens[39]	DeepFocus[42]
Ours	**89.4%**	**72.7%**	**83.4%**	**86.1%**	**82.9%**	**76.7%**	**77.2%**	**82.8%**

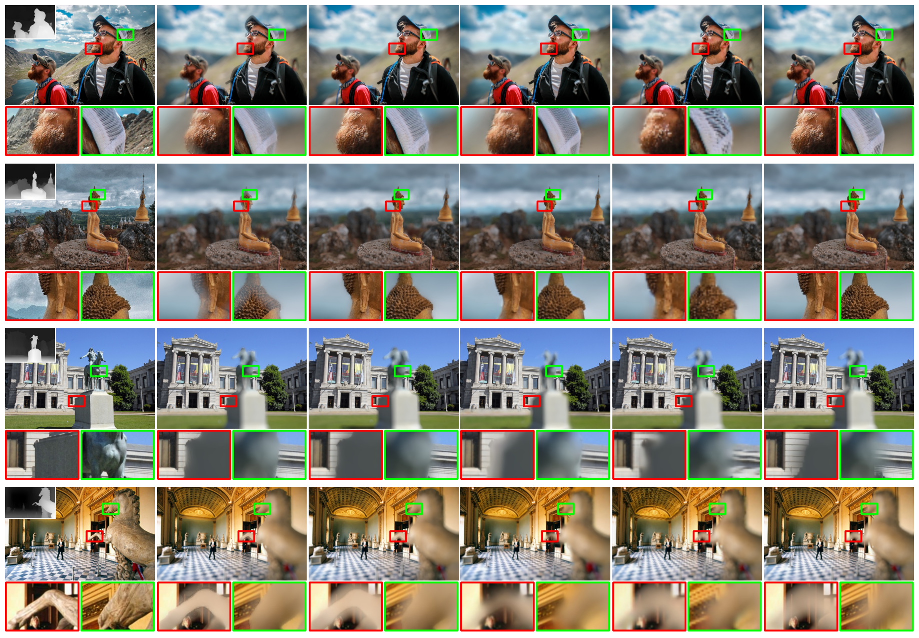

Fig. 7. Qualitative results of the user study. The first 2 rows are refocused on the foreground. The last 2 rows are refocused on the background.

poor LPIPS and unpleasant fuzziness and aliasing on refocused plane. (iii) All of the above methods cannot fill the occluded areas with convincing contents, and the rendered results look fake, particularly when the blur amount is large. (iv) Our approach outperforms other methods numerically and creates much more realistic partial occlusion effects. Besides, generating more background images can handle well the continuous occlusion, leading to better performance.

4.2 Bokeh Rendering on Real-World Images

In this section, all disparity inputs are predicted by DPT [29], and we use only one inpainted image in the background inpainting module.

User Study. Since numerical metrics cannot reflect the perceptual quality of the rendered result, we conduct a user study on real-world images. Specifically,

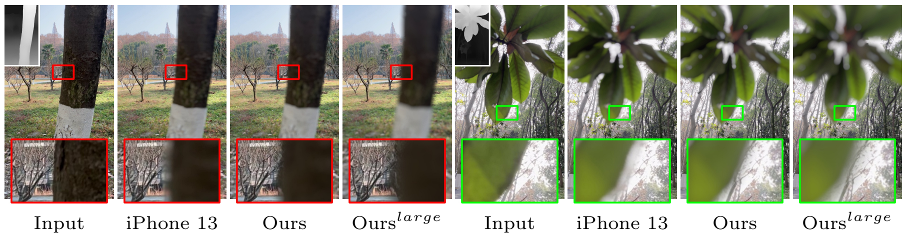

Fig. 8. Comparison with iPhone 13 Cinematic Mode. Image resolution is 1080×1920. The superscript *large* denotes applying the larger blur amount in the rendering.

we collect 50 all-in-focus images from websites, including both indoor and outdoor scenes. The average resolution of images is 1690×1354. For each image, we manually select 2 focal points, one on the foreground and the other on the background. Then, we use different methods to produce bokeh images under the same controlling parameters. During the test, we show each participant 4 images at a time, including an all-in-focus image, a darker all-in-focus image with the focal point labeled, and two bokeh images in random order produced by our approach and a method randomly selected from Scatter [38], SteReFo [3], DeepLens [39] and DeepFocus [42]. The participant is required to choose the method with more realistic bokeh effects or choose none if it is hard to judge. In addition, each participant needs to complete at least 10 tests before submitting results. Finally, 99 people participate in this survey, and there are 2097 valid pairwise comparisons. The data of the 2 focal points is counted separately. We show the comparison results in Table 2, where the number represents the preference of our approach over the other method. One can observe that whether the image is refocused on the foreground or the background, our approach is most favored. Some qualitative results are visualized in Fig. 7. See the supplementary material for more experimental details and visual results.

It is also worth noting that for other rendering methods, their rankings in this user study differ from the results on the synthetic dataset. For example, Scatter [38] gets high metrics on the synthetic dataset, but it is the least preferred in the user study due to the obvious boundary artifacts. This phenomenon demonstrates that the current metrics may fail to accurately measure the perceptual quality of rendered bokeh results.

Comparison with iPhone 13 Cinematic Mode. We further compare MPIB with the latest feature of iPhone 13 - Cinematic Mode, which can freely change the focus point after capturing. From Fig. 8, the upper limit of blur amount for iPhone 13 Cinematic Mode is small, and our approach creates more clear and natural bokeh effects in transition areas of the foreground and background.

4.3 Ablation Study

To investigate the impact of different components, we present ablation studies on the synthetic dataset. The results are shown in Table 3.

Table 3. Ablation study on the synthetic dataset. The best performance is in **boldface**. Our settings are underlined.

Experiment	Method	Constant disparity for each object					Varying disparity for each object				
		LPIPS↓	PSNR↑	PSNR_{ob}↑	SSIM↑	SSIM_{ob}↑	LPIPS↓	PSNR↑	PSNR_{ob}↑	SSIM↑	SSIM_{ob}↑
Inpainting Model	EC [22]	0.012	**36.8**	**30.1**	**0.989**	0.949	**0.019**	**36.8**	**30.6**	**0.986**	0.955
	MADF [52]	0.012	36.7	30.0	**0.989**	0.949	0.020	36.7	30.4	**0.986**	0.955
	LaMa [36]	**0.011**	36.7	30.0	**0.989**	**0.951**	**0.019**	**36.8**	30.5	**0.986**	**0.956**
Number of MPI Planes	8	0.027	34.1	28.5	0.977	0.930	0.090	31.6	27.5	0.932	0.902
	16	0.015	36.0	29.6	0.986	0.946	0.044	34.8	29.5	0.969	0.939
	32	**0.011**	36.7	30.0	**0.989**	0.951	0.019	36.8	**30.5**	0.986	0.956
	64	**0.011**	**36.8**	**30.1**	**0.989**	**0.952**	**0.018**	**36.9**	**30.5**	**0.989**	**0.959**
Weight Normalization	w/o	0.012	36.2	29.7	**0.989**	0.950	**0.019**	36.2	30.1	**0.986**	**0.956**
	w/	**0.011**	**36.7**	**30.0**	**0.989**	**0.951**	**0.019**	**36.8**	**30.5**	**0.986**	**0.956**

Inpainting Model. As we use the off-the-shelf inpainting model to generate background image, we compare 3 inpainting methods: EC [22], MADF [52], and LaMa [36]. From Table 3, one can observe that their performance is similar, showing that our framework can adapt to different inpainting models.

Number of MPI Planes. Increasing the number of MPI planes improves the representation ability of scene geometry and reduces the discretization error, but it also brings more runtime consumption at the same time. We finally choose 32 planes to achieve a trade-off between the performance and the time consumption.

Weight Normalization. As discussed in Sec. 4.1, the weight normalization plays an important role in traditional layered rendering methods. However, the performance gain of this operation is not that significant in our layer composition. The reason is that the learned alpha maps cover the invisible background parts and overlap in areas where the disparity changes smoothly, which makes the denominator of Eq. 4 more inclined to be 1.

5 Conclusion

We have presented an analysis on how to apply the MPI representation and the layer compositing formulation to bokeh rendering, and proposed an MPI-based framework MPIB to render multiple high-resolution bokeh effects and handle realistic partial occlusion effects. Despite the fact that this framework works well in general, it still has some limitations, such as the fracture of the connected area due to the plane discretization, and the boundary artifacts caused by terrible inpainting results or the continuous occlusion, which are shown in the supplementary material. We will address these issues in our future work.

Acknowledgements. This work was funded by Adobe.

References

1. Abadie, G., McAuley, S., Golubev, E., Hill, S., Lagarde, S.: Advances in real-time rendering in games. In: ACM SIGGRAPH 2018 Courses, pp. 1–1 (2018) 3
2. Barron, J.T., Adams, A., Shih, Y., Hernández, C.: Fast bilateral-space stereo for synthetic defocus. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4466–4474 (2015) 3
3. Busam, B., Hog, M., McDonagh, S., Slabaugh, G.: Sterefo: Efficient image refocusing with stereo vision. In: Proc. IEEE International Conference on Computer Vision Workshops (ICCVW). pp. 0–0 (2019) 1, 2, 3, 6, 10, 11, 12, 13
4. Criminisi, A., Perez, P., Toyama, K.: Object removal by exemplar-based inpainting. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). vol. 2, pp. II–II. IEEE (2003) 4
5. Dutta, S., Das, S.D., Shah, N.A., Tiwari, A.K.: Stacked deep multi-scale hierarchical network for fast bokeh effect rendering from a single image. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2398–2407 (2021) 3
6. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in Neural Information Processing Systems (NIPS) **27** (2014) 4
7. Hach, T., Steurer, J., Amruth, A., Pappenheim, A.: Cinematic bokeh rendering for real scenes. In: Proc. European Conference on Visual Media Production (CVMP). pp. 1–10 (2015) 3
8. Hays, J., Efros, A.A.: Scene completion using millions of photographs. ACM Transactions on Graphics (TOG) **26**(3), 4–es (2007) 4
9. Ignatov, A., Patel, J., Timofte, R.: Rendering natural camera bokeh effect with deep learning. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 418–419 (2020) 3, 10
10. Ignatov, A., Patel, J., Timofte, R., Zheng, B., Ye, X., Huang, L., Tian, X., Dutta, S., Purohit, K., Kandula, P., et al.: Aim 2019 challenge on bokeh effect synthesis: Methods and results. In: Proc. IEEE International Conference on Computer Vision Workshops (ICCVW). pp. 3591–3598. IEEE (2019) 3
11. Ignatov, A., Timofte, R., Qian, M., Qiao, C., Lin, J., Guo, Z., Li, C., Leng, C., Cheng, J., Peng, J., et al.: Aim 2020 challenge on rendering realistic bokeh. In: Proc. European Conference on Computer Vision Workshops (ECCVW). pp. 213–228. Springer (2020) 3
12. Iizuka, S., Simo-Serra, E., Ishikawa, H.: Globally and locally consistent image completion. ACM Transactions on Graphics (TOG) **36**(4), 1–14 (2017) 4
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proc. International Conference on Learning Representations (ICLR) (2014) 10
14. Lee, S., Eisemann, E., Seidel, H.P.: Real-time lens blur effects and focus control. ACM Transactions on Graphics (TOG) **29**(4), 1–7 (2010) 3
15. Levoy, M.: Efficient ray tracing of volume data. ACM Transactions on Graphics (TOG) **9**(3), 245–261 (1990) 5
16. Li, J., Feng, Z., She, Q., Ding, H., Wang, C., Lee, G.H.: Mine: Towards continuous depth mpi with nerf for novel view synthesis. In: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 12578–12588 (2021) 2, 4, 7, 10
17. Lin, S., Ryabtsev, A., Sengupta, S., Curless, B.L., Seitz, S.M., Kemelmacher-Shlizerman, I.: Real-time high-resolution background matting. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8762–8771 (2021) 9

18. Liu, G., Reda, F.A., Shih, K.J., Wang, T.C., Tao, A., Catanzaro, B.: Image inpainting for irregular holes using partial convolutions. In: Proc. European Conference on Computer Vision (ECCV). pp. 85–100 (2018) 4
19. Liu, H., Jiang, B., Song, Y., Huang, W., Yang, C.: Rethinking image inpainting via a mutual encoder-decoder with feature equalizations. In: Proc. European Conference on Computer Vision (ECCV). pp. 725–741. Springer (2020) 4
20. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Transactions on Graphics (TOG) **38**(4), 1–14 (2019) 4
21. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: Proc. European Conference on Computer Vision (ECCV). pp. 405–421. Springer (2020) 4
22. Nazeri, K., Ng, E., Joseph, T., Qureshi, F., Ebrahimi, M.: Edgeconnect: Structure guided image inpainting using edge prediction. In: Proc. IEEE International Conference on Computer Vision Workshops (ICCVW). pp. 0–0 (2019) 4, 14
23. Niklaus, S., Liu, F.: Softmax splatting for video frame interpolation. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5437–5446 (2020) 6
24. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. In: Advances in Neural Information Processing Systems Workshops (NIPSW) (2017) 9
25. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2536–2544 (2016) 4
26. Peng, J., Luo, X., Xian, K., Cao, Z.: Interactive portrait bokeh rendering system. In: Proc. IEEE International Conference on Image Processing (ICIP). pp. 2923–2927. IEEE (2021) 3
27. Porter, T., Duff, T.: Compositing digital images. In: ACM Transactions on Graphics (TOG). pp. 253–259 (1984) 5
28. Qian, M., Qiao, C., Lin, J., Guo, Z., Li, C., Leng, C., Cheng, J.: Bggan: Bokeh-glass generative adversarial network for rendering realistic bokeh. In: Proc. European Conference on Computer Vision (ECCV). pp. 229–244. Springer (2020) 3
29. Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. In: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 12179–12188 (2021) 12
30. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Proc. International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI). pp. 234–241. Springer (2015) 10
31. Schedl, D.C., Wimmer, M.: A layered depth-of-field method for solving partial occlusion. Journal of WSCG **20**(3), 239–246 (2012) 2, 3
32. Shen, X., Hertzmann, A., Jia, J., Paris, S., Price, B., Shechtman, E., Sachs, I.: Automatic portrait segmentation for image stylization. Computer Graphics Forum **35**(2), 93–102 (2016) 3
33. Shen, X., Tao, X., Gao, H., Zhou, C., Jia, J.: Deep automatic portrait matting. In: Proc. European Conference on Computer Vision (ECCV). pp. 92–107. Springer (2016) 3

34. Shih, M.L., Su, S.Y., Kopf, J., Huang, J.B.: 3d photography using context-aware layered depth inpainting. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8028–8038 (2020) 6
35. Srinivasan, P.P., Tucker, R., Barron, J.T., Ramamoorthi, R., Ng, R., Snavely, N.: Pushing the boundaries of view extrapolation with multiplane images. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 175–184 (2019) 2, 4
36. Suvorov, R., Logacheva, E., Mashikhin, A., Remizova, A., Ashukha, A., Silvestrov, A., Kong, N., Goka, H., Park, K., Lempitsky, V.: Resolution-robust large mask inpainting with fourier convolutions. In: Proc. IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 2149–2159 (2022) 4, 6, 7, 14
37. Tucker, R., Snavely, N.: Single-view view synthesis with multiplane images. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 551–560 (2020) 2, 4, 9
38. Wadhwa, N., Garg, R., Jacobs, D.E., Feldman, B.E., Kanazawa, N., Carroll, R., Movshovitz-Attias, Y., Barron, J.T., Pritch, Y., Levoy, M.: Synthetic depth-of-field with a single-camera mobile phone. ACM Transactions on Graphics (TOG) **37**(4), 1–13 (2018) 3, 4, 10, 11, 12, 13
39. Wang, L., Shen, X., Zhang, J., Wang, O., Lin, Z., Hsieh, C.Y., Kong, S., Lu, H.: Deeplens: Shallow depth of field from a single image. ACM Transactions on Graphics (TOG) **37**(6), 1–11 (2018) 1, 2, 3, 10, 11, 12, 13
40. Wu, J., Zheng, C., Hu, X., Xu, F.: Rendering realistic spectral bokeh due to lens stops and aberrations. The Visual Computer **29**(1), 41–52 (2013) 3
41. Xian, K., Peng, J., Zhang, C., Lu, H., Cao, Z.: Ranking-based salient object detection and depth prediction for shallow depth-of-field. Sensors **21**(5), 1815 (2021) 3
42. Xiao, L., Kaplanyan, A., Fix, A., Chapman, M., Lanman, D.: Deepfocus: Learned image synthesis for computational displays. ACM Transactions on Graphics (TOG) **37**(6), 1–13 (2018) 3, 10, 11, 12, 13
43. Yang, C., Lu, X., Lin, Z., Shechtman, E., Wang, O., Li, H.: High-resolution image inpainting using multi-scale neural patch synthesis. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6721–6729 (2017) 4
44. Yang, Y., Lin, H., Yu, Z., Paris, S., Yu, J.: Virtual dslr: High quality dynamic depth-of-field synthesis on mobile platforms. Electronic Imaging **2016**(18), 1–9 (2016) 3, 4, 5, 11
45. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Free-form image inpainting with gated convolution. In: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 4471–4480 (2019) 4
46. Yu, X., Wang, R., Yu, J.: Real-time depth of field rendering via dynamic light field generation and filtering. Computer Graphics Forum **29**(7), 2099–2107 (2010) 3
47. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 586–595 (2018) 10
48. Zhang, X., Matzen, K., Nguyen, V., Yao, D., Zhang, Y., Ng, R.: Synthetic defocus and look-ahead autofocus for casual videography. ACM Transactions on Graphics (TOG) **38**, 1 – 16 (2019) 2, 3, 6, 10, 11
49. Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., Torralba, A.: Places: A 10 million image database for scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **40**(6), 1452–1464 (2017) 9

50. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: Learning view synthesis using multiplane images. ACM Transactions on Graphics (TOG) **37**(4), 1–12 (2018) 2, 3, 4
51. Zhou, T., Tulsiani, S., Sun, W., Malik, J., Efros, A.A.: View synthesis by appearance flow. In: Proc. European Conference on Computer Vision (ECCV). pp. 286–301. Springer (2016) 4
52. Zhu, M., He, D., Li, X., Li, C., Li, F., Liu, X., Ding, E., Zhang, Z.: Image inpainting by end-to-end cascaded refinement with mask awareness. IEEE Transactions on Image Processing (TIP) **30**, 4855–4866 (2021) 4, 14