Dr.Bokeh: DiffeRentiable Occlusion-aware Bokeh Rendering

YICHEN SHENG, Purdue University, USA

ZIXUN YU, Purdue University, USA

LU LING, Purdue University, USA
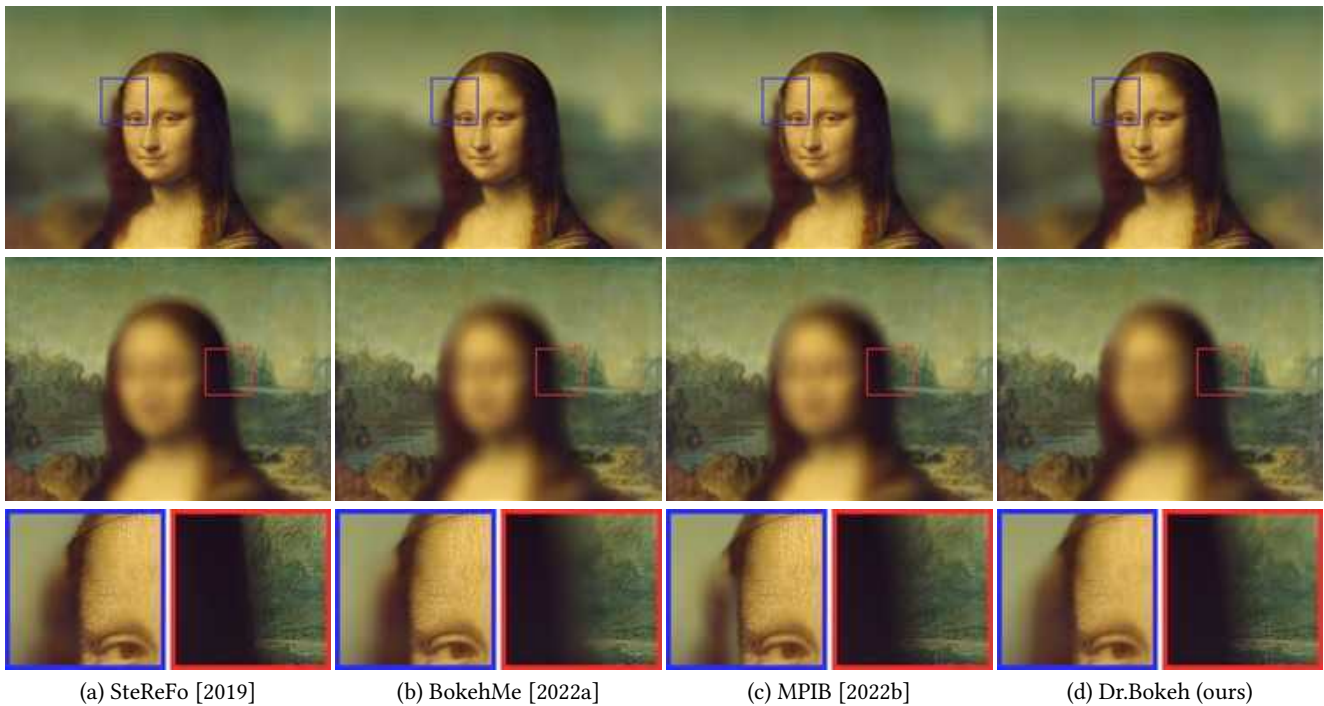
ZHIWEN CAO, Adobe, USA

CECILIA ZHANG, Adobe, USA

XIN LU, Adobe, USA

KE XIAN, Nanyang Technological University, Singapore

HAITING LIN, Adobe, USA

BEDRICH BENES, Purdue University, USA

(a) SteReFo [2019]　　　(b) BokehMe [2022a]　　　(c) MPIB [2022b]　　　(d) Dr.Bokeh (ours)

Fig. 1. By being occlusion-aware, Dr.Bokeh renders realistic bokeh effects directly from the bokeh rendering process without any post-processing. Compared with the scattering/gathering-based method SteReFo or learning-based method BokehMe, Dr.Bokeh renders natural partial occlusion (see red parts). MPIB learns to render a partial occlusion effect but breaks on unseen data (see blue parts). Dr.Bokeh is more robust than learning-based methods given the same inputs because the rendering process is not learned. **Best viewed by zooming in**.

Bokeh is widely used in photography to draw attention to the subject while effectively isolating distractions in the background. Computational methods simulate bokeh effects without relying on a physical camera lens. However, in the realm of digital bokeh synthesis, the two main challenges for bokeh synthesis are color bleeding and partial occlusion at object boundaries. Our primary goal is to overcome these two major challenges using physics principles that define bokeh formation. To achieve this, we propose a novel and accurate filtering-based bokeh rendering equation and a physically-based occlusion-aware bokeh renderer, dubbed Dr.Bokeh, which addresses the aforementioned challenges during the rendering stage without the need of post-processing or data-driven approaches. Our rendering algorithm first pre-processes the input RGBD to obtain a layered scene representation. Dr.Bokeh then takes the layered representation and user-defined lens parameters to render photo-realistic lens blur. By softening non-differentiable operations, we make Dr.Bokeh differentiable such that it can be plugged into a machine-learning framework. We perform quantitative and qualitative evaluations on

synthetic and real-world images to validate the effectiveness of the rendering quality and the differentiability of our method. We show Dr.Bokeh not only outperforms state-of-the-art bokeh rendering algorithms in terms of photo-realism but also improves the depth quality from depth-from-defocus.

CCS Concepts: • **Computing methodologies → Computational photography**; **Image-based rendering**.

Additional Key Words and Phrases: Computational photography, Depth-of-field, Differentiable rendering

ACM Reference Format:
Yichen Sheng, Zixun Yu, Lu Ling, Zhiwen Cao, Cecilia Zhang, Xin Lu, Ke Xian, Haiting Lin, and Bedrich Benes. 2023. Dr.Bokeh: DiffeRentiable Occlusion-aware Bokeh Rendering. In . ACM, New York, NY, USA, 14 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Bokeh is a physical effect produced by a camera lens, and it usually refers to the shape and quality of out-of-focus areas in an image. Such an effect can highlight the in-focus subject and enhance the aesthetic quality of the image. Various computational methods have been developed to create the bokeh effect from an all-in-focus photo, and they can be divided into classical and learning-based methods. Classical bokeh rendering methods take the all-in-focus RGB image and its depth map as inputs. Image filtering methods, like scattering and gathering operations, are commonly used to synthesize the bokeh effect. However, they often lead to the color bleeding problem (see Fig. 2 (a)) on object boundaries because the filtering-based methods fail to handle the boundary correctly. Other methods [Kraus and Strengert 2007; Wadhwa et al. 2018; Zhang et al. 2019a] split the image into layers based on depth discontinuity, render bokeh effects on each layer and carefully blend the rendered layers to fix the color bleeding problem. Still, they cannot represent continuous lens blur effects due to the discretization of layers. The learning-based methods [Peng et al. 2022a; Wang et al. 2018; Xiao et al. 2018] compute bokeh by training on artifact-free data. However, all the aforementioned methods fail to render the partial occlusion effects naturally (see Fig. 2 (c)) when the background is in-focus as the occluded background geometries contribute to the boundary partial occlusion regions.

As bokeh is a physical phenomenon, we aim to fundamentally address the color bleeding and rendering partial occlusion challenges from the rendering process. We observe that the image filtering-based methods do not strictly follow the physics law, which leads to color bleeding and unnatural partial occlusion. First, from the light transportation perspective, the radiance from the background should be fully occluded by the foreground object when the foreground object is in focus (see Sec. 3.2). Failure to model this in rendering leads to the color-bleeding artifact. Second, in the background in-focus case, the foreground object only partially occludes the radiance behind the boundaries. Missing this modeling leads to the unnatural partial occlusion problem. The first case shows the boundary occlusion *"hard"* occlude the radiance behind, while the second case shows the boundary occlusion *"soft"* occlude the radiance behind. No matter which case, we find that the **occlusion** is the critical factor not addressed by image filtering methods. However, it is crucial in addressing color bleeding and rendering partial occlusion effects for realistic bokeh rendering.

We introduce Dr.Bokeh, a novel differentiable (Dr.) bokeh rendering algorithm that correctly handles boundary occlusion in the image-based filtering-based rendering process, is free of the color bleeding artifacts, and renders natural partial occlusion effects (see Fig. 2 (b) and (d)). Dr.Bokeh does not need to be trained and can directly replace existing bokeh renderer in existing pipelines. Although the rendering process is not learning-based, the input information for Dr.Bokeh, e.g., depth and inpainted background, is needed and commonly acquired by learning-based methods. To support the data-driven framework, we soften the non-differentiable operations in Dr.Bokeh and make it fully differentiable so that Dr.Bokeh can be directly used in data-driven pipelines for end-to-end training. We validate the rendering quality and differentiability of Dr.Bokeh by extensive quantitative and qualitative evaluations on synthetic and real-world datasets. An example in Fig. 2 shows how Dr.Bokeh correctly handles the color bleeding problem and renders boundary partial occlusion effects. By correctly handling the boundary occlusion, Dr.Bokeh can help the depth-from-defocus community by improving the depth quality as Dr.Bokeh follows the physics law.

The main contributions of this work can be summarized as follows: (1) Dr.Bokeh, a novel occlusion-aware filtering-based bokeh renderer by introducing geometric occlusion terms. It addresses the color bleeding problem and renders the natural partial-occlusion effects directly in the rendering stage without training. (2) A differentiable implementation allows plug-and-play bokeh rendering in data-driven pipelines. A carefully designed loss that helps the Dr.Bokeh address the depth ambiguity problem in depth-from-defocus problem.

2 RELATED WORK

Our work is closely related to lens blur rendering, differentiable rendering, and image inpainting.

2.1 Lens blur

Existing methods of modeling lens blur can be classified into 3D rendering and image space rendering.

Lens blur in 3D: Classical graphics rendering [Pharr et al. 2016; Potmesil and Chakravarty 1981; Rokita 1996] creates the optical lens blur through ray tracing of the 3D scene from a configured camera setting with a lens (e.g., a thin lens model). The rendered images naturally contain optical bokeh that is physically accurate to how the camera lens is modeled. Real-time methods [Franke et al. 2018; Göransson and Karlsson 2007; Kass et al. 2006; Scheuermann et al. 2004] for efficient DOF rendering have been explored. Different point spread functions [Lee et al. 2009, 2010, 2008; Lei and Hughes 2013; Wu et al. 2013; Xu et al. 2014; Yan et al. 2015] on layered representations are proposed to render the lens blur in hardware rendering pipelines efficiently. Detailed effects, like lens blur on view-dependent surfaces [Lee et al. 2009] or lens aberration effects [Lee et al. 2010; Wu et al. 2013], or challenging dynamic scenes [Jeong et al. 2020] can also be efficiently synthesized. Light field [Vaidyanathan et al. 2015] or multiview images [Liu and Rokne 2016] render natural partial occlusion effects.

(a). Scattering/Gathering   (b). Dr.Bokeh   (c). Scattering/Gathering   (d). Dr.Bokeh
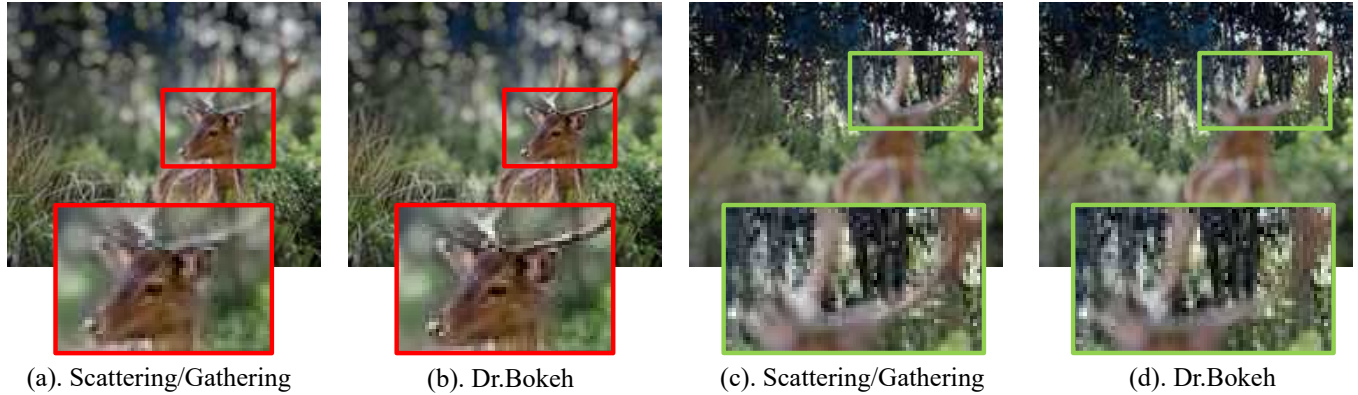
Fig. 2. **Color bleeding and partial occlusion:** Color bleeding is a common artifact meaning the pixels in the out-of-focus scatter to in-focus regions. Partial occlusion is a semi-transparent effect on the out-of-focus boundary regions, mostly in the background in-focus case. (a). naïve scattering/gathering suffers from the color bleeding problem. (b). Dr.Bokeh solves this issue directly in the rendering process via correctly handling on-focal occlusion. (c). Existing scattering/gathering method does not consider occluded pixels, leading to an unnatural partial occlusion effect. (d). Dr.Bokeh addresses this challenge via inpainting occluded pixels and correctly handling non-focal occlusion. **Best viewed by zoom-in.**

However, a 3D scene is not always available, and it is computationally expensive to render a fully-converged rendering as the camera sampling space increases.

Image-space lens blur: Recently, lens blur has been more efficiently rendered in the image space by applying a depth-dependent blur to in-focus pixels. Classical methods [Kraus and Strengert 2007; Wadhwa et al. 2018; Yang et al. 2016; Zhang et al. 2019a] use an RGBD image and create shallow depth-of-field effects using kernel scattering or gathering operations. Naïve scattering or gathering-based methods result in color bleed artifacts due to inadequate occlusion handling. Wadhwa et al.[2018], Busam et al.[2019] and Zhang et al.[2019b] propose a fix to the boundary errors by carefully blending the blurred layers. Approached from a different angle, our method delves into the fundamentals of the light propagation process. It proposes a better light transport simulation that naturally avoids color bleeding and produces realistic partial occlusion effects.

Existing learning-based methods primarily leverage light field rendering and neural rendering. Light-field based lens blur rendering [Kalantari et al. 2016; Srinivasan et al. 2018] predicts scene depth and constructs the 4D light field by warping the all-in-focus image using the depth. The shallow depth-of-field image can be approximated by aggregating the light-field images. However, this approach is memory intensive as a high-quality lens blur effect requires a large set of light field images. Another class of methods leverages the differentiability of the gathering operation to learn a layered representation and render defocus blur [Busam et al. 2019; Luo et al. 2020; Srinivasan et al. 2018]. These approaches assume each layer has a single depth value and apply a fixed blur kernel per layer. Other learning-based methods [Ignatov et al. 2020a; Nalbach et al. 2017; Peng et al. 2022b; Wang et al. 2018; Xiao et al. 2018] directly produce a defocused image using deep neural networks. Recent work of Peng et al. [2022a] first uses a classical method to render lens blur and then a neural network to fix the artifacts. Although the neural network corrects color bleeding, the artifacts and corrections are bounded by the training data. Our physically-based

occlusion-aware rendering method can directly render realistically defocused images without any post-processing fix (see Fig. 1).

Although these methods provide background-focused lens blur rendering, the lack of explicit prediction of the occluded backgrounds leads to unnatural partial occlusion effects (Fig. 1). A recent work of Peng et al. [2022b] proposes to explicitly inpaint the occluded background and apply adaptive gathering operations on the multiplane image (MPI) [Tucker and Snavely 2020] layers to make the network learn shallow depth-of-field rendering on multiple focal planes. Due to the generalization limitation of the learning process and the discretization of layered depth, small inconsistent artifacts between two depth layers lead to a noticeable "leaky" artifact (see Fig. 1). Our method simulates smooth lens blur based on continuous depth. Our blur rendering process does not need to be trained, which makes Dr.Bokeh more robust than learning-based methods. Also, Dr.Bokeh is differentiable and can be directly plugged into classical lens blur or data-driven pipelines.

2.2 Differentiable Rendering

Differentiable rendering makes the rendering process suitable for inverse problems [Jakob et al. 2022; Li et al. 2018; Liu et al. 2019; Zhang et al. 2020, 2019c; Zhao et al. 2020]. Instead of handcrafting the rendering equations, others [Eslami et al. 2018; Lombardi et al. 2019; Sheng et al. 2022, 2021, 2023] propose to leverage the neural renderer directly. We refer readers for more details to [Kato et al. 2020; Tewari et al. 2020].

Existing methods for lens blur rendering [Busam et al. 2019; Kaneko 2021; Peng et al. 2022b; Srinivasan et al. 2018] rely on the light field or adaptive gathering operators on discrete depth layers to render blur results. As the process is fully differentiable, depth maps can be obtained via blur supervision. The estimated depth is derived from discrete layers, leading to quantization errors and consequently lacking sufficient depth details.

Our method can directly replace the blur rendering modules in those methods and output continuous depth. Gur and Wolf [2019]
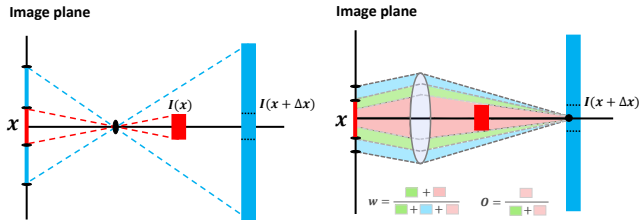
Fig. 3. **Occlusion-based bokeh rendering:** The left image shows that the red and blue objects are projected to the image plane for a pinhole camera. Each pixel in the image space has a corresponding 3D point under the pinhole camera assumption. The right image shows that the point $x + \Delta x$ (originally occluded by x in pinhole camera view) will be visible and contributes to the pixel where x was projected from the scattering perspective. The cone regions of all the colors visualize the regions that $x + \Delta x$ can cover. w is the percentage of energy for $x + \Delta x$ to be scattered to pixel x. O is the percentage of energy for $x + \Delta x$ to be scattered to pixel x considering the occlusion by the front red object.

propose a differentiable scattering-based bokeh rendering layer with a Gaussian blur kernel to learn continuous depth estimation. The Gaussian blur kernel is not physically accurate and does not consider the geometry occlusion, leading to inaccurate depth and artifacts on boundaries. Dr.Bokeh follows physics laws and achieves better depth estimation results.

2.3 Image Inpainting

Our work is related to image inpainting in terms of how occluded pixels are handled. Compared with traditional methods [Criminisi et al. 2003; Hays and Efros 2007], CNN-based methods [Iizuka et al. 2017; Pathak et al. 2016; Yang et al. 2017] supervised by a GAN loss [Goodfellow et al. 2020] generate plausible contents using the spatial context. Conventional convolution and different network architectures [Liu et al. 2020; Nazeri et al. 2019; Zhu et al. 2021] have been extended by different convolutions [Liu et al. 2018; Suvorov et al. 2022; Yu et al. 2019] to deal with free-form inpainting mask.

We directly utilize the off-the-shelf inpainting method [Suvorov et al. 2022] to fill in occluded contents.

3 IMAGE SPACE BOKEH RENDERING

We introduce an occlusion-aware filtering-based bokeh rendering method, taking into account of *occlusion* that is missing in point splatting-based methods (scattering or gathering) (Sec. 3.1). An accurate occlusion calculation requires a full 3D scene, either given or reconstructed, which is rarely available in consumer photography. Instead, we approach the occlusion term from a 2D image space, specifically targeting bokeh rendering. We base our modeling on different types of occlusions (Sec. 3.2). Then we propose our layered bokeh rendering equation by exactly modeling the in-focal occlusion and approximating the non-focal occlusion, which is fundamentally difficult to retrieve (Sec. 3.3).
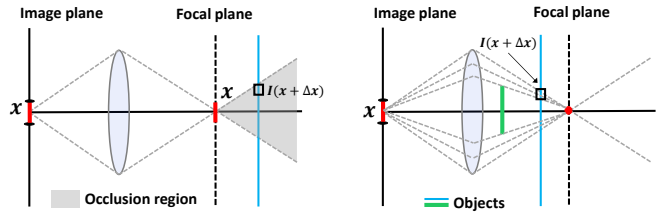

Fig. 4. **Two types of occlusion:** In the occlusion by the focal point (left), the parts behind the red visible object (although as small as a pixel) should not scatter to the position x in the image plane during the bokeh rendering process. In the occlusion by non-focal objects (right) like the green stick, some objects originally occluded by the green objects under the pinhole camera view can contribute to the position x in the image plane.

3.1 Occlusion-aware Bokeh Rendering

The filtering-based bokeh rendering method with the occlusion term (see Fig. 3) has the form:

$$B(x) = \sum_{y \in \Omega(x)} I(y)w(y,x)O(y,x), \qquad (1)$$

where $B(x)$ is the pixel value for rendered bokeh image, $\Omega(x)$ is the set of all neighborhood pixels of x contributing to the defocus blur, $I(y)$ denotes the in-focus pixel values of y, $w(y,x)$ is the energy weight term for pixel y to scatter to x, and $O(y,x)$ is the geometry occlusion term for pixel y to scatter to x.

Existing methods [Busam et al. 2019; Peng et al. 2022a; Yang et al. 2016] often denote I as the all-in-focus image. In the generalized rendering equation that we propose, $I(y)$ includes not only the visible neighborhood pixels but also the occluded pixels that contribute to $B(x)$ (see Fig. 3).

The energy term $0 \le w \le 1$ is a unitless value that represents the fraction of energy $I(y)$ scattered to $B(x)$. The w term is determined jointly by the energy distribution of the scattering region S and the lens shape K as described below. To compute the energy distribution of the scattering region S, we first calculate the blur kernel radius k at depth z_p using Lensmaker's formula:

$$k = \alpha\, L\, f \left| \frac{1}{z_p} - \frac{1}{z_f} \right|, \qquad (2)$$

where α is a configurable scaling factor, L is the lens size, f is the focal length, and z_f is the scene depth at focal length. Existing gathering-based methods [Busam et al. 2019; Wadhwa et al. 2018] often discretize the scene into layers and apply convolution with blur kernels whose radius is computed with Eqn. (2). If $||x - y|| \ge k$, $w(y) = 0$, meaning there is no contribution from the pixel $I(y)$. These methods make different assumptions on the distribution for pixels with $||x - y|| \le k$: uniform distribution [Wadhwa et al. 2018; Yang et al. 2016], distance-based [Wadhwa et al. 2018], radial-based [Busam et al. 2019] have been used to compute the energy term w. The lens shape is another factor that affects w. Instead of using a circular disk kernel [Wadhwa et al. 2018], stylized lens shapes like a star-, heart-, or triangular-shaped kernels have been used for this term [Yang et al. 2016].
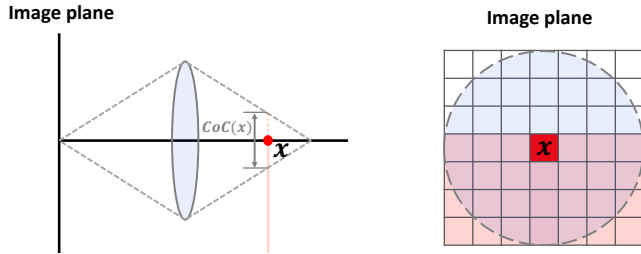
Image plane

Image plane

Fig. 5. **Non-focal occlusion:** Given an image plane, a thin lens and a rectangular-shaped pink object in the scene, the left image shows its side view. The right image is the local region for the projected pixel position for x in the image plane. The region covered by the blue circle visualizes the CoC region. The pink regions are the rectangular-shaped object. The inter-layer occlusion is computed by integrating the occluders over the CoC regions for pixel x and is used to occlude the contributions of layers behind.

Finally, the newly introduced occlusion term $0 \le O(x, y) \le 1$ describes whether $I(y)$ is occluded when it scatters to pixel x. The occlusion term is critical in bokeh rendering, especially when handling boundary regions, which are sensitive to color bleeding artifacts caused by blending pixels from incorrect depth layers. As shown in Fig. 4, ignoring the occlusion term leads to boundary artifacts when focusing on the foreground. It also leads to unnatural partial occlusion artifacts when focusing on the background. Existing methods [Busam et al. 2019; Wadhwa et al. 2018] address the color bleeding problem by pasting back the segmented in-focus object to the final rendered image to override the error regions or using a neural network [Peng 2022a] to correct the errors. These methods, however, still fail to model natural partial occlusion effects in the challenging cases of focusing on the background. Peng et al. [2022b] uses a data-driven method to learn to render realistic partial occlusion. Similar to any deep learning methods, Peng et al. [2022b] is not robust on unseen data and may break due to generalization issues. We propose to address both challenges fundamentally from the rendering process without any post-processing or training of a neural network.

3.2 Occlusion Types

We classify the occlusion into two types (see Fig. 4): (1) **on-focal occlusion**: the occluder is on the focal plane when it blocks all the contributions from behind, and (2) **non-focal occlusion**: the occluder is not on the focal plane, and only block a portion of rays. Note that the objects invisible to the camera in the all-in-focus image may still contribute to the final lens blur result (Fig. 4 in the thin-lens camera cases). As shown in Fig. 2, correctly handling **on-focal occlusion** makes Dr.Bokeh free of color bleeding artifacts, and correctly handling the **non-focal occlusion** helps Dr.Bokeh render natural partial occlusion effects.

The first type of occlusion is straightforward and explains why some pixels within the scattering region should not scatter to the foreground when the foreground is in-focus. The second type of occlusion needs to be modeled with more care. When the occluder is not at the focal plane (Fig. 4), the occluded points, which are invisible in the all-in-focus image, become partially visible, while the

occluder becomes semitransparent around its blurry boundary. This type of occlusion requires correctly modeling the scene geometries, the lens parameters, and the relative distances between two pixels in the image plane, which is a complex function to model in the image space. However, under the planar surface assumption, we can approximate the occlusion in the way shown in Fig. 5. If we assume the scene is composed of planes aligned with the camera view, similar to the pink rectangle in Fig. 5, all the points on the pink rectangle are on the same focal plane. They block the rays coming from behind. In this special case, the percentage of blocked rays by point x and its neighbors equals the percentage of pixels the object covers in the CoC region, as shown in Fig. 5. Therefore, by searching around the circle of confusion (CoC) region of x, we can approximate the percentage of rays coming from behind that will be occluded by the pink object. In practice, Dr.Bokeh applies the aforementioned method to approximate non-focal occlusion computation for non-planar layers. Experiments show that our non-focal occlusion approximation still renders realistic bokeh effects.

3.3 Layered Occlusion-aware Bokeh Rendering Equation

Given the observations in Sec. 3.2, we propose a layered occlusion-aware bokeh rendering equation:

$$B_l(x) = \sum_{l=1}^{n} V_l(x) \Pi_{k=1}^{l-1} (1 - V_k(x)) \frac{\sum_{y \in \Omega} I_l(y) w_l(y, x) O_l(y, x)}{\sum_{y \in \Omega} w_l(y, x) O_l(y, x)}, \quad (3)$$

where $O_l(y)$ and $V_l(x)$ are the in-layer occlusion and inter-layer occlusion, n is the total amount of layers. The layered occlusion-aware bokeh rendering equation Eqn. (3) computes the scattering results with in-layer occlusion for each layer, then blends the layers with inter-layer occlusion terms from front to back according to the inter-layer occlusion. All the blending weights sum up to one to ensure energy conservation. This new rendering method corrects the classical scattering-/gathering-based methods by enabling spatially varying defocus blur with continuous change of blur radius and handles correctly both in-layer occlusion and inter-layer occlusion at the discontinuity boundaries.

Without entire 3D scene geometries, it is challenging to reconstruct all occluded layers given only single image input. We propose to approximate the layer representation by peeling the scene at an object level, i.e., one layer is defined by pixels that belong to the same object.

The in-layer occlusion term $O_l(y)$ ensures each layer has the correct on-focal occlusion to avoid any color bleeding problem. The inter-layer occlusion term $V_l(x)$ ensures the correct handling of non-focal occlusion and blending of different layers, resulting in natural partial occlusion effects. In detail, the $V_l(x)$ term decides the occlusion percentage of the radiance behind and also the amount of energy coming from layer l. In Fig. 5, $V_l(x)$ for the pink rectangle layer reweights the energy from the rectangle layer scattering and the energy coming from layers behind such that their coefficients sum up to be one, which ensures energy conservation.

The **on-focal occlusion** is denoted by $O_l(y, x) \in \{0, 1\}$, a unitless binary value that describes the in-layer occlusion between the

neighborhood pixels y and x:

$$O_l(y, x) = \begin{cases} 0, & \text{if } d_x = 0 \text{ and } d_x > d_y \\ 1, & \text{otherwise} \end{cases} \tag{4}$$

where d_x is the relative disparity (inverse of depth): $d_x = 1/z_x - 1/z_f$. As shown in Sec. 3.2, the in-layer occlusion only happens when the point corresponding to pixel x in 3D is on the focal plane and the corresponding 3D point of neighborhood pixel $x + dx$ is behind the 3D point of the pixel x. In practice, we model $O_l(y, x)$ as a probability instead of a binary value. There are several advantages: 1) The continuous value $O_l(y, x)$ better models the real physics as points not exactly on the focal plane but near the focal plane should partially occlude some amount of radiance from behind; 2) The probability is a "soft" value that models a smooth boundary occlusion. 3) the softened $O_l(y, x)$ is differentiable. The equation details are discussed in Sec. 4.1

The **non-focal occlusion** is denoted by $V_l(x) \in [0, 1]$ is a unitless value that describes the inter-layer *visibility* between layer l and all the layers behind:

$$V_l(x) = \frac{1}{A_{\Omega'}} \sum_{y \in \Omega'} a_l(y), \tag{5}$$

where Ω' is the set of all the in-layer neighborhood pixels within the circle of confusion (CoC) region for x, $A_{\Omega'}$ is the area of Ω' and a_l is the alpha value for the layer l. The CoC region can be computed by Eqn. (2).

The energy term $w_l(y, x)$ for the pixel y in layer l is:

$$w_l(y, x) = \frac{S_l(y, x)K(y)a(y)}{A_l(y)} = \frac{\mathbb{1}(\|y - x\| < r)\mathbb{1}(\|y - x\| < k)a(y)}{\pi r^2}, \tag{6}$$

where $a(y)$ is the alpha value of y, r is the scatter radius, k is the lens size, and $A_l(y)$ is the area of the CoC of y. The norm is L2-norm measuring the Euclidean distance, and w is similar to [Lee et al. 2008] as it considers the energy attenuation for different scatter radius r. But we further model the lens shape term K to support stylized lens shape. By default, K is a perfect circle as described in Eqn. (6).

4 DIFFERENTIABLE BOKEH RENDERING

Differentiability would allow the bokeh renderer to fit into any neural network for end-to-end training. However, the occlusion-aware bokeh rendering method in Sec. 3.3 is not differentiable. We introduce a fully differentiable bokeh rendering method in Sec. 4.1, discuss the derivative details in Sec. 4.2 and show an application that benefits from it: depth from defocus cues in Sec. 4.3.

4.1 Soften the Non-differentiable Operations

The occlusion-aware bokeh equation Eqn. (1) includes two non-differentiable terms: the occlusion term O and the scattering term S_l. The terms are non-differentiable as it involves non-differentiable operations similar to step function or Dirac delta function as shown in Fig. 6. We approximate those non-differentiable terms with differentiable operations, e.g., a step function can be approximated with a soft-step function. The occlusion term O is the Dirac delta
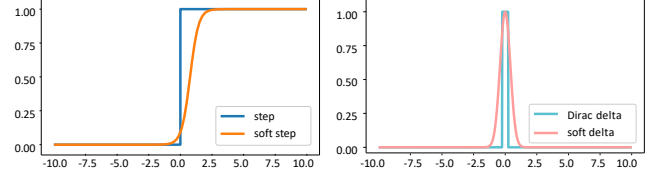

Fig. 6. **Softening of the two non-differentiable operations:** Softening of the non-differentiable step function (left). Note we shift the soft-step function to the right to make sure that for $x = 0$, y is close to zero. The right image shows softening of the non-differentiable delta function.

function δ_x with a value infinity at zero and zero everywhere else. We replace the O_l by:

$$O_l(y, x) = 1 - exp(-3d_x^2)\left(\frac{1}{2}\tanh(10(d_y - d_x - 0.1)) - \frac{1}{2}\right). \tag{7}$$

The scattering term S_l is a step function(if the neighborhood $x + \Delta x$ can scatter to x then is one, otherwise zero) we replace it by a differentiable function:

$$S_l(y, x) = 1/\left(1 + 10\exp\left(-3(\alpha|d_y| + 1 - \|d_y - d_x\|_2^2)\right)\right), \tag{8}$$

where α is a camera parameter controlling the blur radius. The coefficients in Eqns. (4.1, 4.1) are empirically selected to fit the original function and are reasonable to the bokeh rendering process.

4.2 Derivatives of the Bokeh Rendering Method

The current machine learning frameworks like Pytorch [Paszke et al. 2019] provide automatic differentiation mechanisms for basic mathematical operations. Dr.Bokeh cannot directly be implemented using provided auto-differentiable layers and need custom forward/backward calculation in the CUDA layer. Also, the derivatives, especially w.r.t. disparity, are complicated as most of the terms in the $B(x)$ involve disparity. So we derive the derivatives in this section.

For simplicity, we only derive the partial derivative for a single layer, which is enough for the implementation. Multiple layers can be easily derived based on the per-layer partial derivatives. The partial derivatives of each RGB channel are similar. We use I to denote the three channels. According to the chain rule, the partial derivatives with respect to RGB I, depth d, and alpha a are:

$$\frac{\partial L}{\partial B(x)}\frac{\partial B(x)}{\partial I(x)} = a(x)\sum_{y \in \Omega(x)}\frac{\frac{\partial L}{\partial B(y)}w(x, y)O(x, y)}{\sum_{y' \in \Omega(y)}w(y', y)O(y', y)}. \tag{9}$$

The partial derivative for d is:

$$\frac{\partial L}{\partial B(x)}\frac{\partial B(x)}{\partial d(x)} = \sum_{y \in \Omega}\frac{\partial L}{\partial B(y)}\frac{I(y)(W(y, x) - w(y, x)O(y, x))}{W(y, x)^2}$$
$$\cdot \left(\frac{\partial w(y, x)}{\partial d(x)}O(y, x) + \frac{\partial O(y, x)}{\partial d(x)}w(y, x)\right). \tag{10}$$
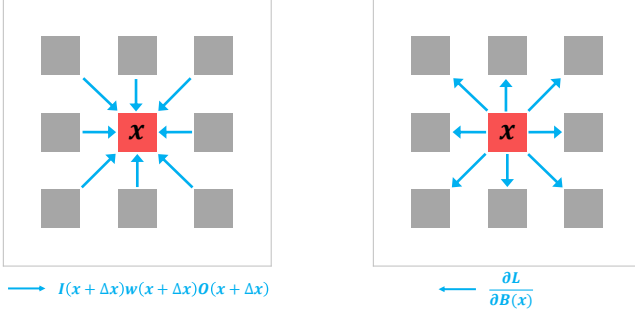
The partial derivative for a is:

Fig. 7. **Per-pixel loss is not enough:** In the left image, neighborhood pixels jointly contribute to x. There exists a case that the x is correct, i.e. the sum of the contributions is correct, but all the neighborhood values are totally wrong, e.g., neighborhood values are shuffled. In this case, there is no loss for pixel x. Then there is no backward gradient for the neighborhood pixels as shown in the right image, even though the neighborhood pixels are wrong.

$$\frac{\partial L}{\partial B(x)}\frac{\partial B(x)}{\partial a(x)} = \sum_{y \in \Omega(x)} \frac{\partial L}{\partial B(y)} I(y) O(x,y) \frac{\partial w(x,y)}{\partial a(x)}$$
$$\cdot \frac{W(y) - w(x,y)O(x,y)}{W(y)^2}, \quad (11)$$

where the $W(y)$ is:

$$W(y) = \sum_{y' \in \Omega(y)} w(y',y) O(y',y). \quad (12)$$

Note that $w(y,x)$ is the energy term for pixel y to scatter to x, and $w(x,y)$ is the energy term for x to scatter to y. So $w(y,x)$ is not equivalent to $w(x,y)$. O is similar to w that $O(y,x)$ is not equivalent to $O(x,y)$. More details of related terms can be found in Appendix.

4.3 Depth from Defocus

Depth from defocus is a depth estimation method that utilizes the correlation between depth and defocuses blur to train a neural network to estimate depth supervised by blur data [Gur and Wolf 2019; Srinivasan et al. 2018]. Dr.Bokeh can replace the bokeh rendering module in the previous depth from defocus methods and achieves better depth quality than the state-of-the-art methods.

Depth from defocus needs a special loss function design, and we propose the loss in the following form:

$$L(y,\hat{y}) = \lambda_1 L_1(y,\hat{y}) + \lambda_2 G(\hat{y}) + \lambda_3 H_{SSIM}(y,\hat{y}), \quad (13)$$

where λ_i are coefficients, L_1 is the norm, G is a regularization term in gradient space for smoothness, and H_{SSIM} is the hierarchy SSIM loss to supervise Dr.Bokeh to learn a better depth. We follow existing works [Gur and Wolf 2019; Srinivasan et al. 2018] to have the regularization term G as:

$$G = \frac{1}{N}\sum_{i=1}^{N} |\partial_x D_i| e^{-|\partial_x I_i|} + |\partial_y D_i| e^{-|\partial_y I_i|}, \quad (14)$$

where N is the N layers of the pyramid of the image, and G is a smooth and regularization term in monocular-depth estimation.
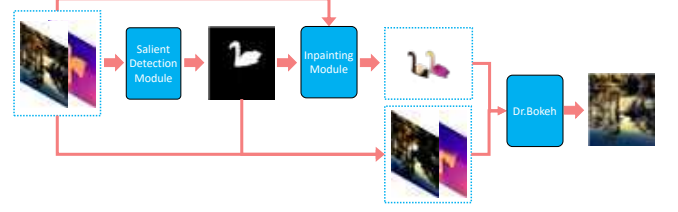

Fig. 8. **Dr.Bokeh rendering pipeline:** The rendering pipeline takes the RGBD as input. A salient object detection module extracts the salient object. Then the pipeline computes the occluded RGBD values behind the salient objects. Then given the foreground RGBAD and background RGBD, Dr.Bokeh renders a realistic bokeh image.

Except for the gradient loss, we noticed that per-pixel loss, such as $L1$ or $L2$ norm cannot supervise the neural network efficiently due to the ambiguity introduced by the bokeh computation process. As the example shown in Fig. 7, the per-pixel loss fails to supervise the network to optimize the neighborhood values. The reason is that pixel scattering or gathering is a patch-level operation, which means that the per-pixel loss signal is not enough in describing the patch-level error. To guide the network not only care about per-pixel results but also regional results, we propose adding a hierarchy SSIM term to learn a better depth. The default SSIM has a patch size of 11. As the maximum scattering range is highly likely to be larger than 11, we propose to use a hierarchy SSIM loss: a set of SSIM loss with different patch sizes to give the pixel the regional feedback instead of per-pixel feedback.

5 IMPLEMENTATION AND RESULTS

Since implementing the backward differential computation is non-trivial, we will release our code to allow for the reproducibility of our work. Here we introduce our forward bokeh rendering pipeline and relevant implementation details in Sec. 5.1. Quantitative and qualitative evaluations of bokeh rendering and differentiability are discussed in Sec. 5.2 and 5.3.

5.1 Implementation

Forward-Bokeh Rendering: For a $W \times H$ pixel image with L layers and searching neighborhood range R, the computation complexity for Dr.Bokeh is $O(W \times H \times L \times R^2)$. We implement Dr.Bokeh with CUDA acceleration and integrate it in Pytorch as a new computation layer. In practice, for a 256×256 image with $L = 2$ and $R = 21$, it takes 19 ms on average. A larger L improves the bokeh rendering quality, but qualitative results in Sec. 5.2.2 and supplementary materials show that two layers are enough for many real-world cases. Our bokeh rendering pipeline (Fig. 8) includes a salient object detection module, an inpainting module, and the Dr.Bokeh renderer. An off-the-shelf salient object detection model [Wei et al. 2020] provides the layered scene representation for all examples in this paper. Please note the salient object detection module can be replaced with any other segmentation or matting network depending on the input category for good performance, e.g., an image matting network that predicts a detailed matting layer for portrait images. The salient object mask is then used to guide background RGBD inpainting, and

we use LaMa [Suvorov et al. 2022] for high-resolution inpainting to generate all the results. Dr.Bokeh takes user-defined camera parameters, including the focal plane distance, blur radius, and lens shape, to synthesize the bokeh image.

Backward-Derivatives: To make Dr.Bokeh fit into data-driven pipelines, we implement the differentiable operation in Pytorch [Paszke et al. 2019]. Although Pytorch is auto-differentiable, the computation in Dr.Bokeh involves many operations not supported by the auto-differentiable computation layers in Pytorch. We implement the derivative computation from scratch in CUDA and integrate the computation layer as a new layer in Pytorch.

The vanilla implementation suffers from the vanishing gradient problem for large depth inputs as the two sides of the soft-step function in Fig. 6 are flat. The optimization results heavily depend on the initialization and may stay at the local minimum without further improvement. To address this issue, we adaptively "leak" the gradient of two sides of the soft step functions similar to leaky ReLU. In practice, we notice this slight change significantly improves the optimization results. As shown in Fig. 12, there are two large blob-like areas in the second-row example for GaussPSF [Gur and Wolf 2019] because GaussPSF applies a Gaussian kernel for blur computation. Although it provides easy gradient computations, GaussPSF also suffers from the vanishing gradient problem. Using our "leaky" soft step, Dr.Bokeh can be robust to bad network initializations and learn a better depth.

Depth-From-Defocus Implementation: Following the Aperture [Srinivasan et al. 2018] and GaussPSF [Gur and Wolf 2019], we train a CNN to predict the depth. The only difference is that we replace the bokeh rendering module with Dr.Bokeh.

5.2 Lens Blur Rendering Evaluation

Quantitatively evaluating lens blur quality is still challenging given a single RGB or RGBD image as input and no benchmark exists yet. So we follow existing works [Peng et al. 2022a,b; Wang et al. 2018] to create a synthetic benchmark (Sec. 5.2.1) except that we render high-quality lens blur results by ray tracing through a real thin-lens. In addition, quantitative evaluations on lens blur are not always reliable [Ignatov et al. 2020b; Peng et al. 2022b], so we collected real-world images and applied a user study to evaluate our method qualitatively (Sec. 5.2.2). We also provide more qualitative results in supplementary materials.

5.2.1 Synthetic Benchmark for Quantitative Evaluation.

Dataset: Existing works [Peng et al. 2022b; Wang et al. 2018] setup the scene by compositing multiple layered images and utilizing an approximated pseudo ray tracer to render the lens blur ground truth. Instead, we implemented a renderer that ray traces through a real thin lens to generate the lens blur ground truth in order to evaluate the effectiveness of Dr.Bokeh (see Fig. 9). The lens is modeled as the intersection of two identical spheres of radius R_c, such that the radius of the intersection circle is the aperture radius $R_a = L/2$. The thickness of the lens is computed as $d = 2\sqrt{R_c^2 - R_a^2}$, which gives the lens' focal length f together with the lens' refractive index η,
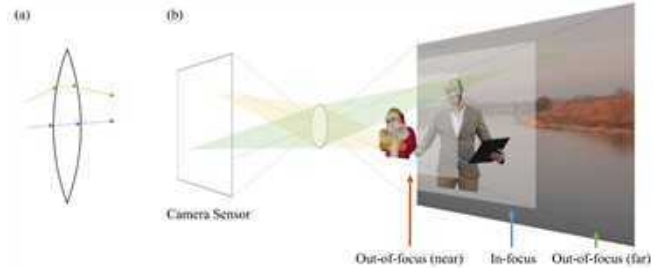

Fig. 9. We simulate bokeh by computing how rays scatter and focus through a spherical lens system. (a) Side-view of how rays get refracted into and out of the lens. Rays hitting somewhere near the center of the lens are barely distorted (blue), while the ones hitting the peripheral get distorted more (brown). (b) Setup of the rendering scene. The billboards are resized to cover the exact FOV of the camera sensor. A point on the image plane shoots many rays that go through the lens aperture and get refracted. As a result, in-focus billboards will be rendered sharply, while ones that are closer/farther will be blurry.

using the lensmaker's equation [Greivenkamp 2004]:

$$\frac{1}{f} = (\eta - 1)\left(\frac{2}{R_c} + \frac{(\eta - 1)d}{\eta R_c^2}\right).$$

The camera is set forth with a chosen FOV, and the image plane is placed at distance $D_I > f$. The color of a pixel is computed by tracing rays from the pixel through various random points on the lens. More details can be found in the appendix.

The scene (5-layer billboards) setup is similar to the dataset by DeepLens [Wang et al. 2018] and MPIB [Peng et al. 2022b]. The foreground objects are randomly sampled from Adobe Matting Dataset [Xu et al. 2017] and AIM-500 [Li et al. 2021]. The background scenes are randomly sampled from the landmark dataset [Weyand et al. 2020]. The benchmark includes 100 scenes with different blur radiuses and focal planes. Each scene has an all-in-focus image, a ground truth depth, a layered ground truth scene representation, and a bokeh ground truth.

Metric: We apply the RMSE metric and a scale-invariant RMSE (RMSE-s) [Sun et al. 2019] as we noticed that different methods have different gamma correction implementations. We also apply the SSIM and ZNCC for perception evaluation.

Comparison to related work: We compare Dr.Bokeh to a gathering-based method SteReFo [Busam et al. 2019], a fully learning-based method DeepLens [Wang et al. 2018] and a hybrid of classical and learning-based method BokehMe [Peng et al. 2022a] and MPIB [Peng et al. 2022b]. Different methods take different kernel parameters. So we search all the blur kernels and pick the best result from each method. All methods take the same depth as input.

Each step in the lens blur rendering pipeline affects the rendering quality. But different methods have different pipelines, which makes the quantitative evaluation easy to be unfair. For example, DeepLens predicts its own depth and then predicts the lens blur. MPIB and Dr.Bokeh involves background inpainting, which looks reasonable perceptually but is easy to have large quantitative errors. To be fair to all the methods, the quantitative evaluation only measures the rendering step quality in all the pipelines instead of measuring the

Table 1. Result on the synthetic benchmark. Comparing with SteReFo [Busam et al. 2019], DeepLens [Wang et al. 2018], BokehMe [Peng et al. 2022a], and MPIB [Peng et al. 2022b]. Dr.Bokeh outperforms state-of-the-art methods in all the metrics.

Method	RMSE ↓	RMSE-s ↓	SSIM ↑	PSNR ↑	ZNCC ↑
SteroFo	0.0179 ± 0.0092	0.0178 ± 0.0092	0.9753 ± 0.0168	35.9201 ± 4.2145	0.9966 ± 0.0028
DeepLens	0.0461 ± 0.0188	0.0403 ± 0.0147	0.9476 ± 0.0263	27.3505 ± 3.6262	0.9827 ± 0.0125
BokehMe	0.0144 ± 0.0077	0.0143 ± 0.0077	0.9708 ± 0.0248	37.7721 ± 4.1209	0.9976 ± 0.0027
MPIB	0.0152 ± 0.0075	0.0151 ± 0.0074	0.9702 ± 0.0256	37.2024 ± 3.8708	0.9974 ± 0.0027
Dr.Bokeh	**0.0133 ± 0.0077**	**0.0133 ± 0.0076**	**0.9757 ± 0.0211**	**38.7288 ± 4.7439**	**0.9979 ± 0.0024**

(a) SteReFo   (b) DeepLens   (c) BokehMe   (d) MPIB   (e) Dr.Bokeh (ours)   (f) Ground Truth
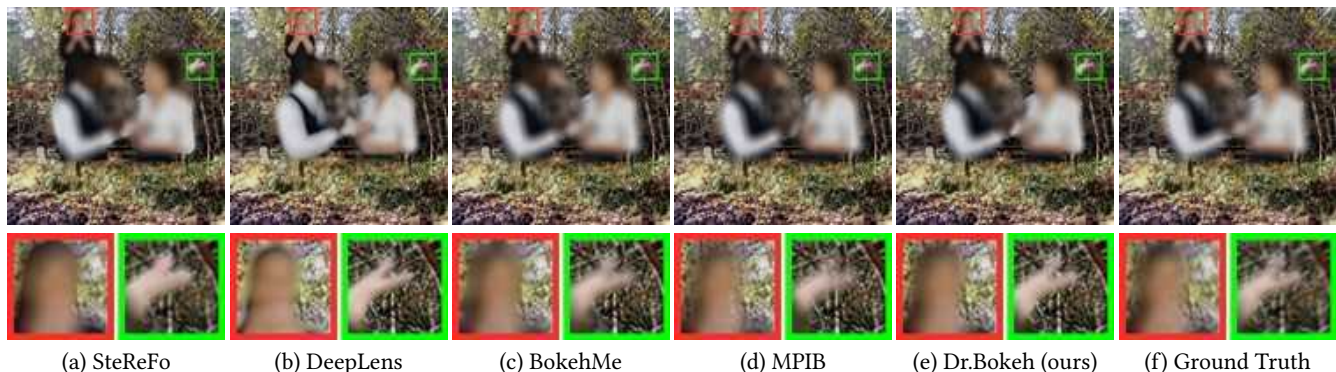
Fig. 10. **Qualitative Comparison of Depth of Field (DoF) Results on Synthetic Benchmarks:** The scattering/gathering-based method (SteRoFo) exhibits unnatural partial occlusion. Learning-based methods (DeepLens and BokehMe) struggle to render natural partial occlusion in the absence of explicit modeling. Although the state-of-the-art method MPIB was trained to address the partial occlusion challenge, our method Dr.Bokeh achieves the best DoF quality without necessitating training. Best viewed by zooming in.

whole pipeline. We use the ground truth depth for all the methods. For DeepLens, we replace the predicted depth with the ground truth depth. For fairness, we replace the predicted occluded pixels with the ground truth pixels for MPIB and Dr.Bokeh.

We show quantitative and qualitative results in Tab. 1 and Fig. 10. Although the gathering-based method SteRoFo achieves a high SSIM value, it is easy to observe its unnature partial occlusion results in Fig. 10 (a). DeepLens does not perform well in metrics. The potential reason is that the synthesized training data for DeepLens fails to have realistic foreground blurs. Although MPIB was designed to learn better partial occlusion effects, BokehMe still performs slightly better than MPIB in the metrics. The reason may be that MPIB is a fully learning-based method and does not generalize well to unseen datasets compared to the hybrid method BokehMe. However, MPIB qualitatively renders more natural partial occlusion effects, as shown in Fig. 10 (d). Our method Dr.Bokeh performs the best in the quantitative metrics and can render realistic partial occlusion effects.

5.2.2 Real-data for Qualitative Evaluation.

Dataset: Existing works [Ignatov et al. 2020b; Peng et al. 2022b] suggest that the numerical metric for evaluating lens blur is unreliable, which also shows up in our quantitative evaluations. Thus, we further apply a qualitative evaluation to evaluate our method. We collect real-world images with different subjects, background scenarios, and lighting as testing data for the user study. The user study contained 40 questions of two-alternative forced-choice (2AFC). Each

question is a pair of lens blur results generated by Dr.Bokeh and a comparison method (SteReFo, DeepLens, BokehMe, and MPIB).

Comparison to related works: The metric and the comparing works are the same as the quantitative evaluation: 41 people (75% male, 25% female, 46% no photography experience, 26% some experience, 28% experienced) participated in our user study. We discard all replies that were too short (under three minutes) or always clicked on the same side. Our results show that 81%, 74%, 68%, 61% of participants support that the image generated by Dr.Bokeh is more realistic than SteReFo, DeepLens, BokehMe, and MPIB. In particular, the T-test value in the more realistic lens blur effect for an image generated by Dr.Bokeh and BokehMe, DeepLens, and SteReFo are 5.75, 8.12, and 11.63 and are significant at 0.001 levels, which indicates the Dr.Bokeh is significantly better than those reference methods from a user perception perspective.

Fig. 11 demonstrates the qualitative comparisons with all the previous works. For foreground in-focus cases, Dr.Bokeh can preserve the sharp boundary consistently, while the learning-based methods still have generalization issues such as unnatural partial occlusion effects and leaky background artifacts (see Fig. 11 the second and third examples). For background in-focus cases, Dr.Bokeh has natural partial occlusion effects and is more robust than SOTA learning-based method MPIB. More qualitative comparison results can be found in the supplementary materials.

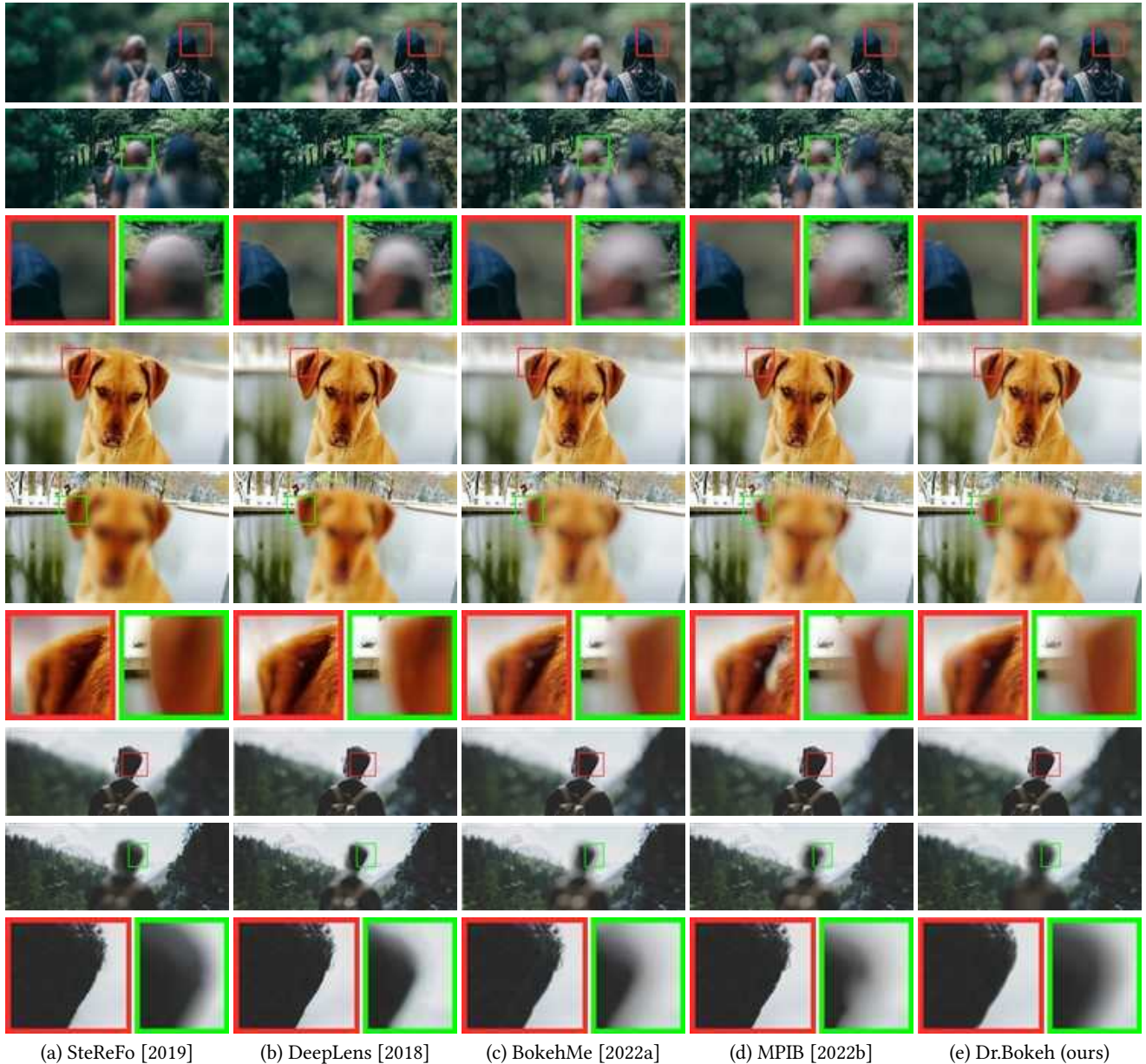|				
(a) SteReFo [2019]	(b) DeepLens [2018]	(c) BokehMe [2022a]	(d) MPIB [2022b]	(e) Dr.Bokeh (ours)

Fig. 11. **Qualitative comparisons on real-world images:** Classical methods (SteReFo) are competitive for foreground in-focus but fail to render natural boundary partial occlusion. Learning-based methods (DeepLens and BokehMe) suffer from unnatural partial occlusion effects. MPIB learns to render the partial occlusion effect but has leaking artifacts due to generalization issues (see the second and the third-row examples). Dr.Bokeh renders natural partial occlusion effects and is more robust for either foreground or background in-focus cases given the same inputs.

5.3 Differentiable Evaluation

Comparison to Related Work: We relate our differentiable lens blur rendering to two previous works: Aperture Supervision (Aperture) [Srinivasan et al. 2018] and Guassian-based PSF (GaussPSF)[Gur and Wolf 2019]. Aperture trains a neural network to predict depth layers by blur image supervision. GaussPSF replaces the bokeh rendering module in Aperture with differentiable Gaussian kernels

and trains the neural network in a similar routine. Compared to GaussPSF, our occlusion-aware Dr.Bokeh is a more accurate differentiable bokeh rendering module in terms of lens blur physics. For a fair comparison in the following benchmarks, we use the same depth estimation network [Xian et al. 2020] for all the comparison methods and the same loss functions Eqn. (13) for all methods.

Benchmark: We evaluate the differentiability of Dr.Bokeh on the real-world benchmark: *Light Field Dataset* [Srinivasan et al. 2017].

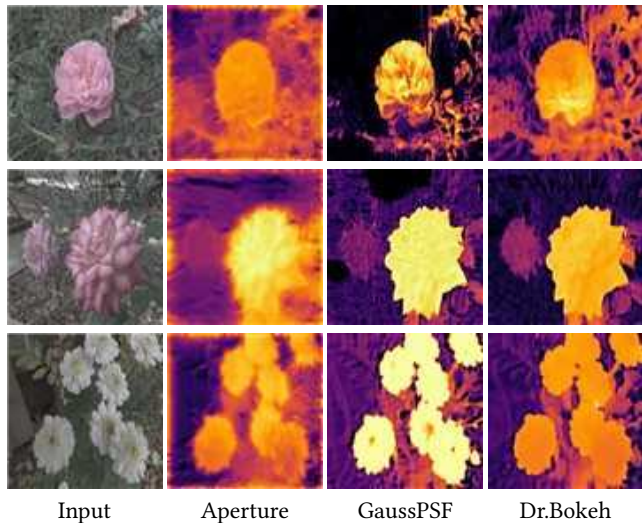Input   Aperture   GaussPSF   Dr.Bokeh

Fig. 12. **Depth optimization for one pair data:** The first column is the all-in-focus input image. The second column shows results by Aperture [2018]; the third column by GaussPSF [2019], and the last column results by Dr.Bokeh. The depth map optimized by Dr.Bokeh has more details and is more accurate.

There is no depth ground truth for the real datasets, so we only quantitatively evaluate the final rendered bokeh images and qualitatively show depth qualities from all the methods. The bokeh images rendered from the light-field camera are good bokeh approximations. There are 3,343 images. Similar to previous works, we split the dataset into 3,006 training images and 337 testing images.

Table 2. Result on the light field benchmark. Comparing with aperture supervision [Srinivasan et al. 2018] and GaussPSF [Gur and Wolf 2019]. Note the metrics are measured on the DoF image due to the lack of depth ground truth.

Method	RMSE ↓	SSIM ↑	PSNR ↑
Aperture	0.0133 ± 0.0038	0.9774 ± 0.0104	37.8396 ± 2.2216
GaussPSF	0.0132 ± 0.0032	0.9767 ± 0.0101	37.8391 ± 1.9961
Dr.Bokeh	**0.0123 ± 0.0029**	**0.9807 ± 0.0086**	**38.4422 ± 1.9447**

Depth Quality: Tab. 2 shows the quantitative evaluation results. Dr.Bokeh outperforms all the previous works in all metrics, which shows that the more accurate blur renderer improves the learning process. The quantitative evaluation is measured on bokeh images and we show the qualitative results of the generated depth map in Fig. 12 and 13. The depth map can either be obtained by direct optimization over an all-in-focus image and a bokeh image pair or by training a neural network to predict the depth based on a large-scale defocus dataset. The direct optimization over one-pair data can clearly show the depth quality supervised by the differentiable rendering layer, while the depth predicted by the trained neural network can illustrate the overall performance of the differentiable layer in the data-driven pipeline.

As shown in Fig. 12, Dr.Bokeh can obtain the best quality depth image supervised by the defocus image as Dr.Bokeh is more accurate
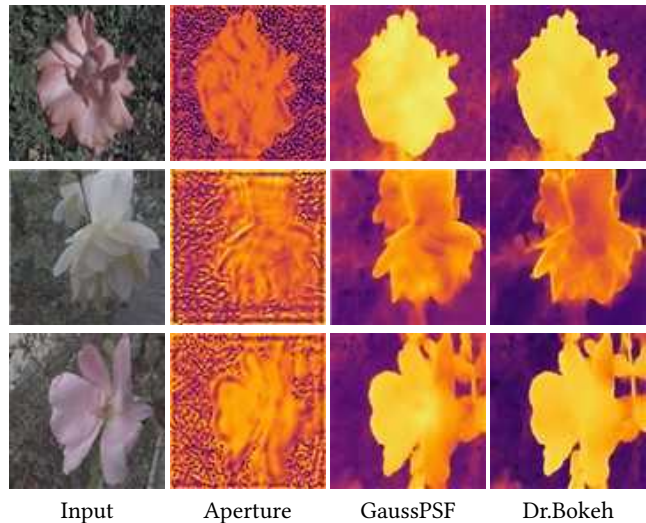

Input   Aperture   GaussPSF   Dr.Bokeh

Fig. 13. **Depth from defocus dataset**. The first column is the all-in-focus input image. The second column shows results by Aperture [2018]; the third column by GaussPSF [2019], and the last column by Dr.Bokeh. The depth map by Aperture is noisy. GaussPSF predicts smoother depth. Dr.Bokeh predicts smoother depth and keeps more boundary details.

Table 3. **Ablation study.** Correct handling of boundary occlusion improves the existing gathering-based depth form defocus performance. The proposed hierarchy SSIM loss further helps Dr.Bokeh learns better depth.

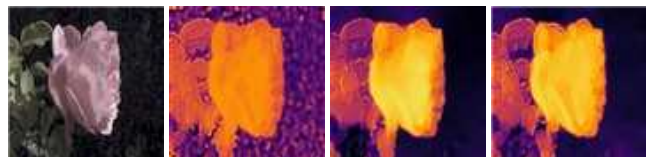| Method	RMSE ↓	SSIM ↑	PSNR ↑
w.o. occlusion	0.0139 ± 0.0034	0.9729 ± 0.0115	37.4016 ± 2.0212
w. occlusion	0.0136 ± 0.0033	0.9740 ± 0.0113	37.5888 ± 2.0473
L1 + Grad	0.0146 ± 0.0036	0.9673 ± 0.0136	36.9398 ± 2.0497
L1 + Grad+ SSIM	0.0136 ± 0.0033	0.9740 ± 0.0113	37.5888 ± 2.0473
L1 + Grad+ HSSIM	0.0123 ± 0.0031	0.9770 ± 0.0107	38.4487 ± 2.0902

Fig. 14. **Qualitative comparisons of different loss:** The first image is the RGB input. The second image is the result of L1 + Grad loss. The third image is the result of L1 + Grad + SSIM loss. The last image is the result of L1 + Grad + HSSIM loss.

in terms of the lens blur physics. Fig. 13 further shows that Dr.Bokeh helps the neural networks learn to predict the best quality depth compared with related works.

Ablation Study: We conduct two experiments to understand the contribution of the occlusion term and the proposed hierarchy SSIM (HSSIM) loss (Sec. 4.2). We first compare Dr.Bokeh with a similar differentiable rendering layer but without the occlusion term by training on the *light field* benchmark. Evaluations (see Tab. 3) on the benchmark show that the occlusion term helps the neural network

Fig. 15. **Failure cases:** the salient object detection cannot find a good target for complex scenes, which leads to unnatural partial occlusions.

training. Second, in the loss function experiment, we compare our loss function (Eqn. 13) with two similar versions: one is just a $L1$ loss with the gradient loss, and the other is the $L1$ loss with the gradient loss and the SSIM loss. Note the gradient loss is a regularization loss. Tab. 3 shows our loss function outperforms other baselines. As shown in Fig. 14, the L1 + Grad loss makes the depth map relatively noisy. The L1 + Grad + SSIM makes the results smoother but loses some details. Our L1 + Grad + HSSIM gets a smooth depth map while preserving the boundary details.

6 CONCLUSIONS

We have introduced Dr.Bokeh, a novel physically-based differentiable and occlusion-aware DoF rendering algorithm. Dr.Bokeh addresses the color bleeding problem and renders realistic partial occlusion for DoF effect synthesis directly in the rendering stage without any training on the rendering. Moreover, Dr.Bokeh is a fully differentiable DoF rendering module that allows it to be plug-and-play in data-driven pipelines. Qualitative and quantitative comparisons validate that Dr.Bokeh achieves the state-of-the-art lens blur quality in different focus settings and the state-of-the-art depth quality in the depth-from-defocus community.

Limitation and Future Work: Our method has several limitations. First, Dr.Bokeh relies on the layered image inputs generated by other methods. Therefore, the layered image quality affects the final bokeh quality, as shown in Fig. 15. Second, although the two-layer scene approximation works well in many cases shown in the qualitative evaluations, it fails when the two layers are not enough to represent a complex scene, as shown in Fig. 15. Third, lack of the synthesis of noise, which can often be observed in images captured by real-world devices like DSLR, the depth quality learned by Dr.Bokeh may be affected. Therefore, one of the future works is to propose a better scene understanding and robust layered scene representation. The differentiability of Dr.Bokeh may help this process. In addition, extending Dr.Bokeh with noise synthesis may further help the depth-from-defocus task.

REFERENCES

Benjamin Busam, Matthieu Hog, Steven McDonagh, and Gregory Slabaugh. 2019. SteReFo: Efficient Image Refocusing with Stereo Vision. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, Seoul, Korea (South), 3295–3304. https://doi.org/10.1109/ICCVW.2019.00411

Antonio Criminisi, Patrick Perez, and Kentaro Toyama. 2003. Object removal by exemplar-based inpainting. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, Vol. 2. IEEE, II–II.

SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. 2018. Neural scene representation and rendering. *Science* 360, 6394 (2018), 1204–1210.

Linus Franke, Nikolai Hofmann, Marc Stamminger, and Kai Selgrad. 2018. Multi-layer depth of field rendering with tiled splatting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 1 (2018), 1–17.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* 63, 11 (2020), 139–144.

Jhonny Göransson and Andreas Karlsson. 2007. Practical post-process depth of field. *GPU Gems* 3, 583-606 (2007), 2.

John E Greivenkamp. 2004. *Field guide to geometrical optics*. Vol. 1. SPIE press Bellingham, Washington.

Shir Gur and Lior Wolf. 2019. Single Image Depth Estimation Trained via Depth From Defocus Cues. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Long Beach, CA, USA, 7675–7684. https://doi.org/10.1109/CVPR.2019.00787

James Hays and Alexei A Efros. 2007. Scene completion using millions of photographs. *ACM Transactions on Graphics (ToG)* 26, 3 (2007), 4–es.

Andrey Ignatov, Jagruti Patel, and Radu Timofte. 2020a. Rendering natural camera bokeh effect with deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 418–419.

Andrey Ignatov, Radu Timofte, Ming Qian, Congyu Qiao, Jiamin Lin, Zhenyu Guo, Chenghua Li, Cong Leng, Jian Cheng, Juewen Peng, et al. 2020b. AIM 2020 challenge on rendering realistic bokeh. In *European Conference on Computer Vision*. Springer, 213–228.

Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. 2017. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)* 36, 4 (2017), 1–14.

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. 2022. DR. JIT: a just-in-time compiler for differentiable rendering. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–19.

Yuna Jeong, Seung Youp Baek, Yechan Seok, Gi Beom Lee, and Sungkil Lee. 2020. Real-time dynamic bokeh rendering with efficient look-up table sampling. *IEEE Transactions on Visualization and Computer Graphics* 28, 2 (2020), 1373–1384.

Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. 2016. Learning-based view synthesis for light field cameras. *ACM Trans. Graph.* 35, 6 (Nov. 2016), 1–10. https://doi.org/10.1145/2980179.2980251

Takuhiro Kaneko. 2021. Unsupervised Learning of Depth and Depth-of-Field Effect from Natural Images with Aperture Rendering Generative Adversarial Networks. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Nashville, TN, USA, 15674–15683. https://doi.org/10.1109/CVPR46437.2021.01542

Michael Kass, Aaron Lefohn, and John D Owens. 2006. Interactive depth of field using simulated diffusion on a GPU. (2006).

Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. 2020. Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057* (2020).

M. Kraus and M. Strengert. 2007. Depth-of-Field Rendering by Pyramidal Image Processing. *Computer Graphics Forum* 26, 3 (2007), 645–654. https://doi.org/10.1111/j.1467-8659.2007.01088.x _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2007.01088.x.

Sungkil Lee, Elmar Eisemann, and Hans-Peter Seidel. 2009. Depth-of-field rendering with multiview synthesis. *ACM Transactions on Graphics (TOG)* 28, 5 (2009), 1–6.

Sungkil Lee, Elmar Eisemann, and Hans-Peter Seidel. 2010. Real-time lens blur effects and focus control. *ACM Trans. Graph.* 29, 4 (July 2010), 1–7. https://doi.org/10.1145/1778765.1778802

Sungkil Lee, Gerard Jounghyun Kim, and Seungmoon Choi. 2008. Real-time depth-of-field rendering using point splatting on per-pixel layers. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 1955–1962.

Kefei Lei and John F Hughes. 2013. Approximate depth of field effects using few samples per pixel. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. 119–128.

Jizhizi Li, Jing Zhang, and Dacheng Tao. 2021. Deep automatic natural image matting. *arXiv preprint arXiv:2107.07235* (2021).

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–11.

Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. 2018. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European conference on computer vision (ECCV)*. 85–100.

Hongyu Liu, Bin Jiang, Yibing Song, Wei Huang, and Chao Yang. 2020. Rethinking image inpainting via a mutual encoder-decoder with feature equalizations. In *European Conference on Computer Vision*. Springer, 725–741.

Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7708–7717.

Xin Liu and Jon G Rokne. 2016. Depth of field synthesis from sparse views. *Computers & Graphics* 55 (2016), 21–32.

Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. 2019. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751* (2019).

Xianrui Luo, Juewen Peng, Ke Xian, Zijin Wu, and Zhiguo Cao. 2020. Bokeh Rendering from Defocus Estimation. In *Computer Vision – ECCV 2020 Workshops*, Adrien Bartoli and Andrea Fusiello (Eds.). Vol. 12537. Springer International Publishing, Cham, 245–261. https://doi.org/10.1007/978-3-030-67070-2_15 Series Title: Lecture Notes in Computer Science.

Oliver Nalbach, Elena Arabadzhiyska, Dushyant Mehta, Hans-Peter Seidel, and Tobias Ritschel. 2017. Deep Shading: Convolutional Neural Networks for Screen-Space Shading. *Computer Graphics Forum* 36, 4 (July 2017), 65–78. https://doi.org/10.1111/cgf.13225 arXiv:1603.06078 [cs].

Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Qureshi, and Mehran Ebrahimi. 2019. Edge-connect: Structure guided image inpainting using edge prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 0–0.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. 2016. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2536–2544.

Juewen Peng, Zhiguo Cao, Xianrui Luo, Hao Lu, Ke Xian, and Jianming Zhang. 2022a. BokehMe: When Neural Rendering Meets Classical Rendering. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, New Orleans, LA, USA, 16262–16271. https://doi.org/10.1109/CVPR52688.2022.01580

Juewen Peng, Jianming Zhang, Xianrui Luo, Hao Lu, Ke Xian, and Zhiguo Cao. 2022b. MPIB: An MPI-Based Bokeh Rendering Framework for Realistic Partial Occlusion Effects. In *Computer Vision – ECCV 2022 (Lecture Notes in Computer Science)*, Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer Nature Switzerland, Cham, 590–607. https://doi.org/10.1007/978-3-031-20068-7_34

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically based rendering: From theory to implementation*. Morgan Kaufmann.

Michael Potmesil and Indranil Chakravarty. 1981. A lens and aperture camera model for synthetic image generation. *ACM SIGGRAPH Computer Graphics* 15, 3 (1981), 297–305.

Przemyslaw Rokita. 1996. Generating depth of-field effects in virtual reality applications. *IEEE Computer Graphics and Applications* 16, 2 (1996), 18–21.

Thorsten Scheuermann et al. 2004. Advanced depth of field. *GDC 2004* 8 (2004).

Yichen Sheng, Yifan Liu, Jianming Zhang, Wei Yin, A Cengiz Oztireli, He Zhang, Zhe Lin, Eli Shechtman, and Bedrich Benes. 2022. Controllable shadow generation using pixel height maps. In *European Conference on Computer Vision*. Springer, 240–256.

Yichen Sheng, Jianming Zhang, and Bedrich Benes. 2021. SSN: Soft shadow network for image compositing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4380–4390.

Yichen Sheng, Jianming Zhang, Julien Philip, Yannick Hold-Geoffroy, Xin Sun, He Zhang, Lu Ling, and Bedrich Benes. 2023. PixHt-Lab: Pixel Height Based Light Effect Generation for Image Compositing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16643–16653.

Pratul P. Srinivasan, Rahul Garg, Neal Wadhwa, Ren Ng, and Jonathan T. Barron. 2018. Aperture Supervision for Monocular Depth Estimation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Salt Lake City, UT, 6393–6401. https://doi.org/10.1109/CVPR.2018.00669

Pratul P Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. 2017. Learning to synthesize a 4D RGBD light field from a single image. In *Proceedings of the IEEE International Conference on Computer Vision*. 2243–2251.

Tiancheng Sun, Jonathan T Barron, Yun-Ta Tsai, Zexiang Xu, Xueming Yu, Graham Fyffe, Christoph Rhemann, Jay Busch, Paul E Debevec, and Ravi Ramamoorthi. 2019. Single image portrait relighting. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 79–1.

Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. 2022. Resolution-robust large mask inpainting with fourier convolutions. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*.

2149–2159.

Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. 2020. State of the art on neural rendering. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 701–727.

Richard Tucker and Noah Snavely. 2020. Single-View View Synthesis With Multiplane Images. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Seattle, WA, USA, 548–557. https://doi.org/10.1109/CVPR42600.2020.00063

Karthik Vaidyanathan, Jacob Munkberg, Petrik Clarberg, and Marco Salvi. 2015. Layered light field reconstruction for defocus blur. *ACM Transactions on Graphics (TOG)* 34, 2 (2015), 1–12.

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. 2018. Synthetic depth-of-field with a single-camera mobile phone. *ACM Trans. Graph.* 37, 4 (Aug. 2018), 1–13. https://doi.org/10.1145/3197517.3201329

Lijun Wang, Xiaohui Shen, Jianming Zhang, Oliver Wang, Zhe Lin, Chih-Yao Hsieh, Sarah Kong, and Huchuan Lu. 2018. DeepLens: Shallow Depth Of Field From A Single Image. *arXiv:1810.08100 [cs]* (Oct. 2018). http://arxiv.org/abs/1810.08100 arXiv: 1810.08100.

Jun Wei, Shuhui Wang, Zhe Wu, Chi Su, Qingming Huang, and Qi Tian. 2020. Label decoupling framework for salient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 13025–13034.

T. Weyand, A. Araujo, B. Cao, and J. Sim. 2020. Google Landmarks Dataset v2 - A Large-Scale Benchmark for Instance-Level Recognition and Retrieval. In *Proc. CVPR*.

Jiaze Wu, Changwen Zheng, Xiaohui Hu, and Fanjiang Xu. 2013. Rendering realistic spectral bokeh due to lens stops and aberrations. *The Visual Computer* 29 (2013), 41–52.

Ke Xian, Jianming Zhang, Oliver Wang, Long Mai, Zhe Lin, and Zhiguo Cao. 2020. Structure-Guided Ranking Loss for Single Image Depth Prediction. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lei Xiao, Anton Kaplanyan, Alexander Fix, Matthew Chapman, and Douglas Lanman. 2018. DeepFocus: learned image synthesis for computational displays. *ACM Trans. Graph.* 37, 6 (Dec. 2018), 1–13. https://doi.org/10.1145/3272127.3275032

Ning Xu, Brian Price, Scott Cohen, and Thomas Huang. 2017. Deep image matting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2970–2979.

Shibiao Xu, Xing Mei, Weiming Dong, Xun Sun, Xukun Shen, and Xiaopeng Zhang. 2014. Depth of field rendering via adaptive recursive filtering. In *SIGGRAPH Asia 2014 Technical Briefs*. 1–4.

Ling-Qi Yan, Soham Uday Mehta, Ravi Ramamoorthi, and Fredo Durand. 2015. Fast 4D sheared filtering for interactive rendering of distribution effects. *ACM Transactions on Graphics (TOG)* 35, 1 (2015), 1–13.

Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. 2017. High-resolution image inpainting using multi-scale neural patch synthesis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6721–6729.

Yang Yang, Haiting Lin, Zhan Yu, Sylvain Paris, and Jingyi Yu. 2016. Virtual DSLR: High Quality Dynamic Depth-of-Field Synthesis on Mobile Platforms. *ei* 28, 18 (Feb. 2016), 1–9. https://doi.org/10.2352/ISSN.2470-1173.2016.18.DPMI-031

Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. 2019. Free-form image inpainting with gated convolution. In *Proceedings of the IEEE/CVF international conference on computer vision*. 4471–4480.

Benxuan Zhang, Bin Sheng, Ping Li, and Tong-Yee Lee. 2019b. Depth of field rendering using multilayer-neighborhood optimization. *IEEE Transactions on Visualization and Computer Graphics* 26, 8 (2019), 2546–2559.

Cheng Zhang, Bailey Miller, Kan Yan, Ioannis Gkioulekas, and Shuang Zhao. 2020. Path-space differentiable rendering. *ACM transactions on graphics* 39, 4 (2020).

Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. 2019c. A differential theory of radiative transfer. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–16.

Xuaner Zhang, Kevin Matzen, Vivien Nguyen, Dillon Yao, You Zhang, and Ren Ng. 2019a. Synthetic defocus and look-ahead autofocus for casual videography. *ACM Trans. Graph.* 38, 4 (Aug. 2019), 1–16. https://doi.org/10.1145/3306346.3323015

Shuang Zhao, Wenzel Jakob, and Tzu-Mao Li. 2020. Physics-based differentiable rendering: from theory to implementation. In *ACM siggraph 2020 courses*. 1–30.

Manyu Zhu, Dongliang He, Xin Li, Chao Li, Fu Li, Xiao Liu, Errui Ding, and Zhaoxiang Zhang. 2021. Image inpainting by end-to-end cascaded refinement with mask awareness. *IEEE Transactions on Image Processing* 30 (2021), 4855–4866.

7 APPENDIX

7.1 Differentiabiliy

Here are all the terms related to the derivatives in the Eqns. 9, 10 and 11.

Yichen Sheng, Zixun Yu, Lu Ling, Zhiwen Cao, Cecilia Zhang, Xin Lu, Ke Xian, Haiting Lin, and Bedrich Benes

,

$$\frac{\partial w(x,y)}{\partial d(x)} = \frac{A(x,y)K(x)a(x)\frac{\partial S(x,y)}{\partial d(x)} - S(x,y)K(x)a(x)\frac{\partial A(x)}{\partial d}}{A(x,y)^2}$$

$$(15)$$

where $\frac{\partial S(x,y)}{\partial d(x)}$ and $\frac{\partial S(x,y)}{\partial d(x)}$ are the following in practice:

$$\frac{\partial S(x,y)}{\partial d(x)} = \frac{0.3e^{3(-(d(x)-d(y)))}}{(e^{3(-(d(x)-d(y)))} + 0.1)^2}$$

$$(16)$$

$$\frac{\partial O(x,y)}{\partial d(x)} = e^{-3d(x)^2}\frac{-20}{(10(d(y)-d(x)-0.1))^2} +$$

$$(-0.5-\tanh(10(d(y)-d(x)-0.1)))(-e^{-3d(x)^2}(-6|d(x)|\text{sign}(y)))$$

$$(17)$$

$$\frac{\partial w(x,y)}{\partial a(x)} = \frac{S(y,x)K(y)}{A(y)}$$

$$(18)$$

As mentioned in the paper, directly deriving and implementing the backward computation of Dr.Bokeh is complicated so we will release our code for reproducibility.

7.2 Ray Tracing for Synthetic Benchmark

Different rays hitting different points and collecting different colors is the main mechanism creating blurriness. The thin lens approximation equation ($1/f = 1/D_B + 1/D_I$) gives the recipe for computing the distance of an image D_B to be in focus when the image plane D_I is fixed. Furthermore, suppose an image was placed at a different distance D_B' (requiring a different image plane D_I' to focus). In that case, the following equation gives the radius of circle-of-confusion (r_c) on the image plane:

$$r_c = R_a\frac{|D_I - D_I'|}{D_I'}$$

This equation can also compute the distance D_B' required for a blur of radius r_c. When the image billboards $I_1, I_2, \ldots I_n$ are added to the scene to render, each image I_i is associated with a distance D_i to the lens according to the amount of desired blurriness. Each image is then texture mapped to a rectangle that covers exactly the camera field of view when placed at distance D_i (see Fig 9). The vertices of the image billboards are therefore aligned on the sides of a pyramid in the scene. If the aperture is small enough, the rendered image would be sharp as the camera is close to a pinhole camera. When the lens has some size, the circle of confusion enlarges, which renders out-of-focus image billboards blurry.