Efficient Multi-Lens Bokeh Effect Rendering and Transformation

Tim Seizinger¹*, Marcos V. Conde¹*, Manuel Kolmet², Tom E. Bishop², Radu Timofte¹

¹ Computer Vision Lab, CAIDAS & IFI, University of Würzburg, Germany
² Glass Imaging Inc., CA

Abstract

Many advancements of mobile cameras aim to reach the visual quality of professional DSLR cameras. Great progress was shown over the last years in optimizing the sharp regions of an image and in creating virtual portrait effects with artificially blurred backgrounds. Bokeh is the aesthetic quality of the blur in out-of-focus areas of an image. This is a popular technique among professional photographers, and for this reason, a new goal in computational photography is to optimize the Bokeh effect itself.

This paper introduces EBokehNet, a efficient state-of-the-art solution for Bokeh effect transformation and rendering. Our method can render Bokeh from an all-in-focus image, or transform the Bokeh of one lens to the effect of another lens without harming the sharp foreground regions in the image. Moreover we can control the shape and strength of the effect by feeding the lens properties i.e. type (Sony or Canon) and aperture, into the neural network as an additional input. Our method is a winning solution at the NTIRE 2023 Lens-to-Lens Bokeh Effect Transformation Challenge, and state-of-the-art at the EBB benchmark.

1. Introduction

Computational photography research and recent advancements of mobile cameras aim to reach the visual quality of full-frame DSLR cameras [8, 15]. One of the most popular effects in photography is Bokeh, which refers to the way the lens renders the out-of-focus blur in a photograph (Fig. 1) [14, 28]. Professional photographers can produce different Bokeh styles by using different lens designs and configurations. This effect is controlled by the optical design of a lens, its aperture setting, the distance to the subject, and the focal length of the lens. However, due to the limited optics of mobile cameras, these cannot produce re-
alistic Bokeh naturally. In this case, the effect is added as a post-processing; this is the main focus and application of most algorithms for Bokeh rendering.

Note that Bokeh and depth-of-field (DoF) are two different, yet related techniques in photography. DoF refers to the sharp areas of focus, while Bokeh is the artistic quality of out-of-focus areas. These are related, and often used interchangeably since Bokeh rendering can be seen as transforming wide to shallow depth-of-field images [14].

Classical approaches [3, 25, 36, 39, 46] can render and change Bokeh styles easily by controlling the shape and size of the blur kernel, which is usually an estimated point spread function (PSF). However, these methods often suffer from unpleasant artifacts such as chromatic aberration and depth discontinuities.

Deep learning-based methods [14, 28, 31, 37, 38] represent the state-of-the-art for this task, but they have difficulty to simulate real Bokeh styles e.g. bokeh balls, and only produce the effect present in the training data. Moreover, these methods cannot adjust different styles, and cannot apply large blur kernels to high-resolution (HR) images, as they are limited by the fixed receptive field of the neural network. This is an important point that limits their potential application in real scenarios.

A common approach to render Bokeh consists in segmenting out the foreground (e.g. person, face, or object of interest) in the image, and then processing the background [17, 34, 35, 47] independently. A similar approach is to blur the image based on a depth map [13, 28]. We can also find end-to-end deep learning solutions [14, 17, 33] capable of transforming wide to shallow depth-of-field images automatically, without using depth or segmentation maps.

Despite the active research in this topic, rendering photorealistic Bokeh is still a challenging task. Recently the new Bokeh Transformation task was introduced [7]. This task is defined as follows: for a given input image A (all-in-focus, out-of-focus or in-between) with known lens-type and aperture setting, knowing the target lens type and setting, we aim to produce or transform the corresponding effect B while preserving the foreground intact.

In this work, we present an efficient neural network capable of rendering or converting the Bokeh effect of one lens to the effect of another lens without harming the sharp foreground regions in the image. The proposed EBokehNet model achieves state-of-the-art results at the Bokeh Effect Transformation Dataset (BETD) [7] and the Everything is Better with Bokeh! (EBB) benchmark [14, 16].

2. Related Work

Classical Bokeh rendering methods require a single image and 3D information. The most practical ones use depth maps [2, 11, 25, 39, 42], while more advanced classical rendering requires the complete 3D scene information [28].

Early works such as Bertalmío et al. [3] use a point-spread function (PSF) to simulate realistic bokeh. Yang et al. [42] use simple ray tracing to render the effect.

Due to its complexity, it is common to split this task into: depth estimation [13], semantic segmentation [5], and classical rendering [25, 29, 34–36, 46]. This task decomposition also implies decoupling the image into foreground and background, and execute rendering from back to front.

These modular approaches are flexible, however, they might struggle at depth discontinuities. Furthermore their overall performance depends highly on the individual performance of each module e.g. the quality of the estimated depth maps, the quality of the background-foreground segmentation, the power of the classical rendering.

During the recent years we can observe a trend towards using deep learning to simulate the rendering process as an end-to-end operation. Early works such as Nalbach et al. [27] and Xiao et al. [40] train convolutional neural networks (CNN) to produce a bokeh effect from an all-in-focus image and its accurate depth map. Wang et al. [37] proposes an automatic rendering system comprised of depth estimation, lens blur, and guided upsampling to generate high-resolution depth-of-field (DoF) images from a single image. Most recently, Peng et al. proposes BokehMe [28], a framework that combines neural and classical rendering techniques and achieves state-of-the-art results.

Other deep learning-based methods [14, 17, 21, 23, 31, 33, 38] do not require any prior information such as depth maps, which are not easy to capture in real-world scenes. These methods usually follow a encoder-decoder architecture [32], and map the all-in-focus input images into shallow DoF images in an end-to-end manner.

Despite the promising results, these deep learning-based rendering methods lack controllability. The trained neural network can produce only the style of the effect present in the training data, and the blur range is limited by the kernels’ receptive field.

We aim to improve mobile photography, therefore it is also important to address the method complexity considering the computational limitations of mobile devices. Ignatov et al. [14, 16, 17] studied efficient Bokeh rendering for mobile devices, being able to deploy the models on different target platforms [17]. These challenges use the popular large-scale Everything is Better with Bokeh! (EBB) dataset [14] containing more than 10 thousand images collected in the wild.

By controlling the aperture size of the lens, pairs of images with wide (aperture f/16) and shallow (aperture f/1.8) depth-of-field were taken, resulting in a normal sharp photo and one exhibiting a strong Bokeh effect.

In this work we propose a novel neural rendering method able to control the effect by feeding the lens properties into the neural network as an additional input.
3. Efficient Bokeh Rendering

We design our network EBokehNet for Bokeh effect rendering and transformation considering the following desired features: (i) the network should allow to control the strength and style of the Bokeh effect. This is fundamental to tackle the novel Bokeh Effect Transformation task. (ii) The model must be efficient in order to be usable, to achieve this we adopt the already efficient NAFNet architecture [4] and simplify it further by reducing the number of encoder-decoder blocks. (iii) The model should be able to render or convert the Bokeh effect of one lens to the effect of another lens without modifying the sharp foreground regions in the image. This ultimately implies a SOTA performance.

Model Design We illustrate the architecture of EBokehNet in Fig. 2. We follow a classical U-Net [32] encoder-decoder structure inspired in NAFNet [4], yet reducing notably the number of blocks.

We propose a new modified Baseline Block that incorporates LayerNorm [1], GELU activations [12], pixel-wise convolutions, inverted residual blocks [8], and additional residual connections. Following [4] we use two core elements that ensure efficiency and performance: (i) downsampling using strided convolutions, (ii) upsampling using pixelshuffle [44].

The core modifications allow to integrate the encoded lens information, and thus we can condition the deep features and the overall model behaviour.

We inject the lens information (type and bokeh strength) at different stages of the decoder, by doing this we are able to control the style and strength of the Bokeh effect by “conditioning” the network’s features. This idea was successfully applied for flexible compression removal [18, 41].

We calculate the Bokeh strength factor as follows:

\[
L_{\text{bokeh}} = \frac{1}{F_{\text{tgt}}^2} - \frac{1}{F_{\text{src}}^2}
\]

\[
BF = L_{\text{bokeh}} \ast \text{disparity} \ast \frac{100}{2}
\]

\[
BF_{\text{norm}} = \frac{BF + 1}{2}
\]

considering the aperture of the source \(F_{\text{src}}\) and target \(F_{\text{tgt}}\) lens, and the disparity. This is injected at the 1st encoding block as shown in Fig. 2. The disparity value indicates the relative distance of the foreground to the background, and serves as another indicator of the “amount of blur”. This information is provided for each sample in the BETD dataset [7], in the cases were this information is not
available e.g. EBB dataset [14], our model allows to disable such inputs and conditions (Fig. 2).

Additionally, similar to CoordConv [24], we also employ 2D positional encoding (PE) in some blocks, this provides extra spatial context and guide the network in rendering optical vignetting and cats-eye bokeh in the corners. We use a 2-channel coordinates map as illustrated in Fig. 3. Since the real and synthetic Bokeh effect from BETD [7] is associated with space-varying PSFs, we found PE a very powerful method to enhance features.

Note that we use five blocks with PE at the final decoding stage using high-resolution features (see Fig. 2), we found this especially important to improve performance notably.

4. Experimental Results

We evaluate our models using the novel Bokeh Effect Transformation Dataset (BETD) [7], and the “Everything is Better with Bokeh!” (EBB) benchmark [14, 16].

BETD The dataset [7] contains 20000 and 200 image pairs for the training and test sets, respectively. The average resolution of the images is $1584 \times 1056$. The training set consists on synthetic images generated using an estimated PSF of professional lenses. The test set contains 100 synthetic images, and 100 real captures. Both the simulated and real images are based on Sony Alpha 7R II and Sony Alpha 7R IV professional cameras with a Sony 50mm lens set to f/1.8 and f/16 apertures and a Canon EF 50mm lens set to f/1.8 and f/1.4 apertures. For each real or synthetic pair, we have the corresponding metadata for the source and target images e.g. Sony50mmf1.8BS $\rightarrow$ Canon50mmf1.4BS. We also use the provided disparity value that indicates the “amount” of blur as the “bokeh strength”.

EBB used in [14,16,17] is a large-scale EBB! dataset containing 5K shallow / wide depth-of-field image pairs collected in the wild with the Canon 7D DSLR camera and 50mm f/1.8 fast lens.

4.1. Bokeh Effect Transformation

We provide the results on the BETD [7] benchmark in Tab. 1 and qualitative samples in Fig. 5 and Fig. 6.

Our method EBokehNet achieves state-of-the-art performance while being $50 \times$ smaller in comparison to the others [20, 43]. Even our smaller version EBokehNet-s with less blocks and depth, achieves very competitive results. We believe this is because: (i) novel block with positional encoding, (ii) the new design of our baseline block based on NAFNet [4], and (iii) the efficient integration of lens information into the model.

We define our small version EBokehNet-s as the smartphone model. Following [14, 16] we design this shallow variant for mobile devices. This has blocks with 16 channels instead of 32, and less number of blocks in the middle and decoding part. Since the model is extremely compact (1 Million parameters) we can train using full-resolution images, which we found is an advantage.

In Fig. 4 we show the performance comparison of the two model variants when we vary the bokeh strength e.g. render strong Bokeh $f/16 \rightarrow f/1.8$, transform it $f/1.8 \rightarrow f/1.4$, or recover sharp regions by “removing” Bokeh $f/1.8 \rightarrow f/16$.

Both models allow high-resolution image processing without patching or tiling strategies. Also note that we do not employ the provided alpha masks or any sort of segmentation, the foreground and background separation is completely implicitly learned.

Results per transformation Removal. Is a transformation from shallow to wide DoF e.g. $f/16 \rightarrow f/1.8$. This is the most challenging sub-task since it is similar to deblurring [8], an ill-posed problem. While the perceived increase in background busyness and detail is captured, the details do not necessarily match the ground-truth and the perceived style of the target lens is rarely captured.

Figure 4. Performance comparison of the small and large variants of EBokehNet on BETD [7] depending on the strength of the bokeh transformation. Negative factor means Bokeh is removed (shallow to wide DoF), around 0 it is transformed e.g. Sony50mmf1.4 $\rightarrow$ Canon50mmf1.4, and for positive factors bokeh is rendered by the network.
<table>
<thead>
<tr>
<th>Method</th>
<th># Params. (M)</th>
<th>Synthetic + Real</th>
<th>Real</th>
<th>Foreground/Background</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PSNR ↑</td>
<td>SSIM ↑</td>
<td>LPIPS ↓</td>
</tr>
<tr>
<td>NAFBET [20]</td>
<td>115</td>
<td>35.264</td>
<td>0.9362</td>
<td>0.0985</td>
</tr>
<tr>
<td>SBTNet [7]</td>
<td>265</td>
<td>34.572</td>
<td>0.9361</td>
<td>0.0966</td>
</tr>
<tr>
<td>CBTNet [7]</td>
<td>182</td>
<td>32.326</td>
<td>0.9333</td>
<td>0.1076</td>
</tr>
<tr>
<td>BokehOrNot [43]</td>
<td>21</td>
<td>32.288</td>
<td>0.9327</td>
<td>0.1130</td>
</tr>
<tr>
<td>SGLMS [7]</td>
<td>7</td>
<td>32.076</td>
<td>0.9324</td>
<td>0.1076</td>
</tr>
<tr>
<td>IR-SDE [26]</td>
<td>78</td>
<td>30.866</td>
<td>0.9297</td>
<td>0.1301</td>
</tr>
<tr>
<td>DoubleGAN [17]</td>
<td>5</td>
<td>27.970</td>
<td>0.9213</td>
<td>0.1542</td>
</tr>
<tr>
<td>Synthetic</td>
<td>-</td>
<td>28.599</td>
<td>0.9128</td>
<td>0.2181</td>
</tr>
<tr>
<td>EBokehNet-s</td>
<td>1</td>
<td>34.543</td>
<td>0.9350</td>
<td>0.1039</td>
</tr>
<tr>
<td>EBokehNet</td>
<td>20</td>
<td><strong>35.521</strong></td>
<td><strong>0.9362</strong></td>
<td><strong>0.0993</strong></td>
</tr>
</tbody>
</table>

Table 1. NTIRE 2023 Lens-to-Lens Bokeh Effect Transformation (BETD) [7] results. The methods are ranked by PSNR/SSIM. The models were tested on unseen real captures and synthetic rendered content. We also provide the rounded number of parameters of each method. Synthetic indicates the metrics for the raw source image. Our method EBokehNet achieves state-of-the-art performance while being extremely smaller in comparison to others. Moreover we can appreciate that the method is not harming the sharp foreground regions.

Figure 5. Real captures from the BETD [7]. These images were captured using the same DSLR camera. The proposed EBokehNet is able to do a bidirectional conversion between both setups Sony ↔ Canon. Images courtesy of Glass Imaging, Inc.
Transformation. The method achieves great performance, the bokeh style is very well adjusted to the target lens. However, we found that the method struggles to generalize to real images, which were not present in the training data and have different properties.

Rendering. The method can emulate the bokeh style and strength, and the results match the target lens settings. There is a slight degradation in performance when the bokeh strength is too different from the source (see Fig. 4).

Real vs Synthetic Data We found a clear gap generalization gap between the synthetic and real content. Despite the synthetic content was rendered using accurate estimated PSFs, the properties of real images differ notably e.g. illumination, distance to the objective (implicit depth). We provide extensive qualitative results in Fig. 8 and Fig. 5.

Table 2. Quantitative results on the EBB [14, 16] Val294 testset. Some numbers are borrowed from [23, 38].

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR ↑</th>
<th>SSIM ↑</th>
<th>LPIPS [45] ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBokehNet</td>
<td>24.99</td>
<td>0.852</td>
<td>0.1912</td>
</tr>
<tr>
<td>SKN [22]</td>
<td>24.66</td>
<td>0.8521</td>
<td>0.3323</td>
</tr>
<tr>
<td>DBSI [9]</td>
<td>23.45</td>
<td>0.8675</td>
<td>0.2463</td>
</tr>
<tr>
<td>DMSHN [10]</td>
<td>24.72</td>
<td>0.8793</td>
<td>0.2271</td>
</tr>
<tr>
<td>DDF [30]</td>
<td>24.14</td>
<td>0.8713</td>
<td>0.2482</td>
</tr>
<tr>
<td>BGGAN [31]</td>
<td>24.39</td>
<td>0.8645</td>
<td>0.2467</td>
</tr>
<tr>
<td>BRADCN [23]</td>
<td>24.83</td>
<td>0.8737</td>
<td>0.1448</td>
</tr>
<tr>
<td>PyNet [14]</td>
<td>24.93</td>
<td>0.8788</td>
<td>0.2219</td>
</tr>
<tr>
<td>BEViT [38]</td>
<td>24.57</td>
<td>0.8880</td>
<td>0.1985</td>
</tr>
</tbody>
</table>

4.2. Bokeh Effect Rendering

First, we evaluate our model (pre-trained on the BETD [7] dataset and task) on the EBB dataset [14] in a zero-shot manner. We found that the model only works for a few images, the reason is because the training data in BETD [7] is clearly separable into foreground and background. Also the foreground in BETD is always a person (or a face), while at EBB we find a wide variety of objects.

Therefore, we fine-tune the model on the EBB dataset [14]. We notice that the model quickly learns how to separate foreground and background in real images, and can emulate strong natural Bokeh.

As we show in Tab. 2, our model achieves state-of-the-art results. In comparison to the baseline PyNET [14, 16] with 40M parameters, our method is 40× smaller. Moreover, we achieve better results than other complex methods that apply depth estimation or foreground segmentation, our approach is purely end-to-end with one efficient network. We can conclude that our method is easily transferable to real images with a wide variety of focused objects e.g. faces, persons, cars, animals, plants, etc. Pre-training on simple synthetic data -with a realistic blur model- improves the SOTA for Bokeh rendering on EBB [14].

4.3. Implementation Details

We train all the models using Adam optimizer [19] with Cosine Annealing learning rate scheduler with 10 epochs of linear warmup using a maximum learning rate of 1e-3 and minimum learning rate of 5e-5. We train the models to convergence, for the small model 220 epochs, and 280 for the large version. We use simple L1 loss.

Since the small model is quite efficient, we can high resolution crops of 1024x1024 on images for training, meanwhile we use 512 x 512 crops for the larger version. We apply standard augmentations consisting in flips and rotations. We set the mini-batch size to 14, and run a distributed training over 7 RTX 3090Ti GPUs via DDP. The complete train-
Figure 7. Rendering Bokeh on real-world images from the EBB dataset [14]. We show the rendered Bokeh effect from our EBokehNet in a large variety of scenes in-the-wild. We provide additional qualitative results and comparisons in our project site.

Fine-tuning on EBB. We use the aforementioned experimental setup with the following modifications. We start with a learning rate 5e-4. We keep the same architecture and disable the lens and Bokeh strength encoding, therefore we use only baseline blocks (see Fig. 2). We just need to train 50 epochs to achieve SOTA results. Since the images are not perfectly aligned, we train the model using a combination of fidelity and perceptual losses as follows:

$$L_{EBB} = 0.5 \times L_1 + 0.05 \times L_{SSIM} + 0.1 \times L_{VGG} \quad (1)$$

5. Conclusions

We introduce EBokehNet, a efficient state-of-the-art solution for Bokeh effect transformation and rendering. Our method can render Bokeh from an all-in-focus image, or transform the Bokeh of one lens to the effect of another lens without harming the sharp foreground regions in the image. Moreover, even being an end-to-end network, we can control the shape and strength of the effect by feeding the lens properties into the neural network as an additional input.

We prove the benefits of our method in the novel Bokeh Effect Transformation Dataset, and the real scenes from EBB, achieving state-of-the-art results in both benchmarks.

As future work we will study closer the gap between synthetic and real captures. We also aim to reduce further the complexity of the network and test it on real mobile devices.

Acknowledgements This work was partially supported by the Humboldt Foundation. Glass Imaging thanks Griffin Kettler and his models for assistance with the real images captured for the test portion of the BETD dataset.
Figure 8. *Qualitative results on the Bokeh Effect Transformation Dataset (BETD)* [7] testset.
References


1641