# Adaptive Mask-based Pyramid Network for Realistic Bokeh Rendering

Konstantinos Georgiadis[*2], Albert Saà-Garriga[*1], Mehmet Kerim Yucel[*1], Anastasios Drosou[2], and Bruno Manganelli[1]

[1] Samsung Research UK
[2] Centre for Research and Technology Hellas (CERTH), Information Technologies Institute, Thessaloniki, Greece

**Abstract.** Bokeh effect highlights an object (or any part of the image) while blurring the rest of the image, and creates a visually pleasant artistic effect. Due to the sensor-based limitations on mobile devices, machine learning (ML) based bokeh rendering has gained attention as a reliable alternative. In this paper, we focus on several improvements in ML-based bokeh rendering; i) on-device performance with high-resolution images, ii) ability to guide bokeh generation with user-editable masks and iii) ability to produce varying blur strength. To this end, we propose Adaptive Mask-based Pyramid Network (AMPN), which is formed of a Mask-Guided Bokeh Generator (MGBG) block and a Laplacian Pyramid Refinement (LPR) block. MGBG consists of two lightweight networks stacked to each other to generate the bokeh effect, and LPR refines and upsamples the output of MGBG to produce the high-resolution bokeh image. We achieve i) via our lightweight, mobile-friendly design choices, ii) via the stacked-network design of MGBG and the weakly-supervised mask prediction scheme and iii) via manually or automatically editing the intensity values of the mask that guide the bokeh generation. In addition to these features, our results show that AMPN produces competitive or better results compared to existing methods on the EBB! dataset, while being faster and smaller than the alternatives.

**Keywords:** Bokeh Rendering, Image Refocusing, Laplacian Pyramid

## 1 Introduction

Bokeh effect is one of the fundamental photography techniques, where an object (or a region) in the image is effectively highlighted by blurring out the rest of the image. Traditionally, such an effect is achieved via focusing the camera on an area and taking the photo using a wide aperture lens. Although it is achievable with appropriate cameras with fast lenses with large apertures, or even with mobile devices with stereo setups, bokeh rendering may not be feasible for all mobile devices due to sensory/hardware constraints. Synthetically generating

---

[*] The authors have contributed equally.

the bokeh effect through machine-learning (ML) based methods are therefore viable alternatives, especially for setups without adequate hardware.

There has been a rising interest in synthetic bokeh rendering, especially following the release of EBB! dataset [8] and bokeh rendering challenges [9, 11]. Starting with earlier methods focusing on portrait images [24, 25], later methods leveraged advances in image-to-image methods and generative models [8, 5, 3, 21, 22, 20, 33, 17, 18, 27, 4]. Depth/disparity [27, 21, 8, 4] and saliency maps [21, 9], as well as segmentation maps [34] have found use as the principal guiding component for bokeh rendering, however, such methods require additional ground-truth information during training and require additional components in inference time. Conversely, methods not using any guidance at all [5, 18, 22, 3] are essentially learning an inflexible mapping, where in-focus areas will be implicitly learned by the network and can not be changed by the user. Furthermore, ML-based methods essentially fit to the blur strength of the training data and can generate only fixed bokeh styles [20], leading to limited user interaction.

Thinking from a mobile use case standpoint, we focus on three areas of improvement; i) fast on-device runtimes with high-resolution images, ii) ability to guide bokeh generation with user-editable masks without additional compute in inference time and iii) ability to render bokeh effect with varying blur strengths. There are fast, on-device methods for bokeh rendering [3, 4, 5, 8], but none of them meets the criteria ii) and iii). There are methods leveraging external guidance for bokeh rendering (i.e. criteria ii)) [27, 21, 8, 4, 9, 34], but they require additional processing to generate such guidance (i.e. depth models, saliency models, etc). A recent work [20] focuses on the ability to generate varying blur strengths (criteria iii)), but their solution is not tailored for on-device performance and therefore not suitable for mobile use cases.

In light of these above mentioned criteria, we propose Adaptive Mask-based Pyramid Network (AMPN) for mobile-friendly, realistic bokeh rendering. AMPN is formed of two main building blocks; i) Mask-Guided Bokeh Generator (MGBG) and ii) Laplacian Pyramid Refinement (LPR). MGBG consists of two stacked networks, where the first one is responsible for mask prediction and the other one is for bokeh generation, guided by the predicted mask. LPR essentially refines the low-resolution bokeh output of MGBG and upsamples it to the input resolution. Using lightweight designs in both blocks lets us operate efficiently, whereas LPR lets us produce high fidelity, high-resolution outputs without the need of third party solutions for upsampling/super-resolution (meeting criteria i). Furthermore, if a user is providing its mask, we can completely remove the mask prediction network during inference, making our pipeline even more compact.

Having strong user-guidance often requires strong supervision (i.e. ground-truths) in (guidance) mask prediction, which may not be available in every dataset. Our pipeline, on the other hand, learns mask prediction in a weakly supervised way due to the stacked network paradigm we adopt in MGBG (meeting criteria ii)). We train using wide/shallow depth-of-field images using existing datasets without mask ground-truth. The level of mask guidance we achieve is visually pleasant (see Figure 1), it empowers users with greater flexibility (i.e.
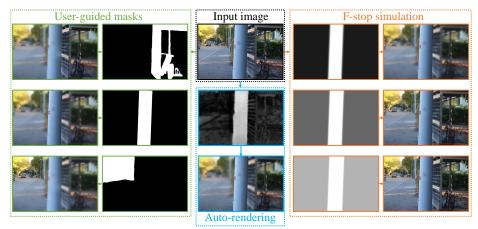
| User-guided masks | Input image | F-stop simulation |
| --- | --- | --- |

Auto-rendering

**Fig. 1.** Our AMPN method renders bokeh images conditioned on masks, which can be provided by users and thus our results can focus on selected area of an image (left side). Furthermore, by changing the intensity of the guidance masks, we can simulate different f-stops (right side). Our weakly-supervised mask prediction network learns masks without direct supervision (middle images), which can be replaced by user-provided masks in inference.

mask editing, providing custom masks) and even extends the use case beyond bokeh rendering to mask-based, learnable image refocusing. Note that our bokeh generation can still be guided with depth and/or saliency maps. Our mask prediction network also brings the controllable blur strength feature; by changing the background intensities of our guidance mask, we show that we can simulate different f-stops (meeting criteria iii), see Figure 7). Our main contributions can be summarized as follows:

1. We propose AMPN for realistic bokeh rendering, which renders bokeh at low resolution via the MGBG block and performs refinement/upsampling with the LPR block. AMPN produces competitive or better results compared to existing methods on EBB!val294 set and performs faster than alternatives, making it ideal for mobile device deployment.
2. We show that the design of AMPN enables training of a mask prediction network in a weakly-supervised way, which lets users control their bokeh rendering with a mask; either obtained by the mask prediction network, edited by the user or even produced by the user.
3. We also show that by simply changing the intensity values of the guidance mask, we can control the strength of the blur in the bokeh rendered image.

## 2   Related work

**Synthetic Bokeh Rendering.** Synthetic bokeh rendering can be largely divided into two categories [20] as classical and ML-based approaches. Classical

rendering is shown to provide a degree of control and flexibility, but relies heavily on availability of accurate 3D information [32, 26, 2]. Here, we focus on the ML-based approaches due to their relevance to our method.

Following the portrait based methods [24, 25], more recent methods utilize prior knowledge, such as depth or saliency maps, along with the in-focus input image for high-fidelity bokeh rendering. In [8], the authors use an inverted pyramid-shaped architecture, which processes images at different scales, thus learning more diverse features. Their approach relies on depth estimation maps, produced by a pretrained Megadepth model [15]. In [21], saliency maps, as well as depth maps are used for a spatially-aware blurring process. The maps are concatenated with the output of a space-to-depth module, before being fed to a densely connected encoder/decoder network that produces the bokeh image.

Parallel lines of work try to avoid depending on external information such as depth or saliency maps. [3] proposes a fast generator architecture with a feature pyramid network and two discriminators, operating at global and patch levels. The authors of [5] propose a fast, stacked multi-scale hierarchical network that process images at different scales to obtain local and global features. In [17], the authors propose a multi-module network, where each module focus on separate components such as defocus estimation, radiance, rendering and upsampling. [22] uses a fast Glass-Net generator and Multi-Receptive-Field discriminator, and also re-implement instance normalization layers for further speed-ups on a mobile device. Authors of [18] propose a transformer-based architecture for bokeh rendering. They show that the increased receptive field, as well as pretraining the model on image restoration tasks, help improve the results. A recent work [20] propose a combination of classical and neural rendering, to achieve high-resolution photo-realistic bokeh images. Their framework is adjustable, where the blur size, focal plane and aperture shape can be chosen. Our approach combines the flexibility (i.e. user-editable mask guided bokeh rendering, varying blur strength) and mobile-friendliness of existing methods.

**Operating at High-resolutions.** A key aspect of mobile vision tasks, such as dense regression/prediction (i.e. bokeh rendering, image editing, etc), is that they have a low tolerance to visual artefacts. Especially with the continuously improving mobile display technology, high-resolution images are becoming the norm, which makes the error tolerance even lower and requires accurate, high-resolution outputs. However, simply operating at high-resolution is not an option, especially due to the limited resources available on mobile devices.

A naive approach is to perform the vision task at low resolution and then perform upsampling/super-resolution [14]. However, this approach requires a second model, which should work well on the same distribution and has to work sequentially with the vision model, which lowers the overall feasibility. Authors of [16] propose the LPTN framework where upsampling and the vision task is performed jointly, where low-frequency components are translated with a lightweight network and high-frequency details are refined using both low and high-frequency components. Our LPR block is based on LPTN and its successors [12], with key differences explained later in relevant sections.
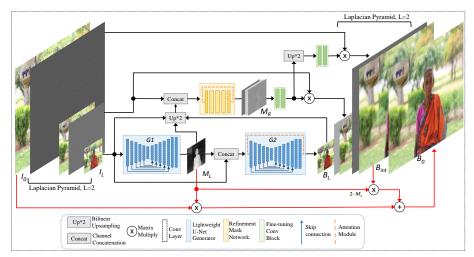
**Fig. 2.** The diagram of our proposed AMPN pipeline. First, we decompose the input image $I_0$ with a Laplacian Pyramid and acquire the low-resolution image $I_L$. Through our MGBG (light blue blocks) block, we first (in a weakly-supervised manner) discover the area of focus $M_L$ with $G1$, and then in $G2$ we use this area of focus to guide the bokeh rendering in low-resolution to produce $B_L$. Within our LRP block (refinement mask network and fine-tuning conv blocks), we progressively upsample and refine the low-resolution bokeh image and use mask-based blending (red arrows) to produce the final high-resolution bokeh image $B_0$.

## 3   Adaptive Mask-based Pyramid Network

In this section, we motivate the need for a new bokeh rendering approach, introduce AMPN and its components in detail.

### 3.1   Motivation

Our aim to design a pipeline that meets the three criteria mentioned in earlier sections; i) fast processing with high-resolution outputs, ii) user-editable mask guided bokeh rendering and iii) ability to edit blur strength. We note that bokeh, by definition, is guided by depth information. Therefore, achieving criteria ii) provides a functionality beyond bokeh rendering, which is essentially image refocusing guided by a mask. In a sense, our aim is to learn from bokeh rendering datasets while being able to perform the more abstract task of mask-based image refocusing. Furthermore, achieving iii) makes our bokeh rendering hardware-independent; essentially we will not overfit to a specific bokeh style [20]. To summarize, our aim is beyond having an accurate, fast bokeh rendering model, but also to produce a highly interactive bokeh rendering experience.

### 3.2  Overview

Our AMPN pipeline is composed of two main blocks; the Mask-Guided Bokeh Generator (MGBG) block and a Laplacian Pyramid Refinement (LPR) block. The input image $I_0$ is decomposed, through a Laplacian pyramid, into its high-frequency components, denoted by $H = [h_0, h_1, ..., h_{L-1}]$, and its low-frequency residual image $I_L$, where $L$ refers to the levels in the Laplacian pyramid [16]. $I_L$ is then fed into the MGBG block, which generates the low-resolution mask $M_L$ and then the low-resolution bokeh image $B_L$. Next, $I_L$, $M_L$ and $B_L$ are upsampled with bilinear interpolation to match the size of the lowest size high-frequency component $h_{L-1}$, and then the four of them are concatenated and used as input to the LPR block. LPR generates a refinement mask $M_R$, which is combined progressively with the high frequency components $H$ of each level, producing the high-resolution image $B_{int}$. $B_{int}$ and $I_0$ are processed with $M_L$ to produce the final bokeh image $B_0$. The entire network is trained end-to-end, using high-resolution inputs and outputs. The overall diagram of AMPN is shown in Figure 2. We now explain in more detail how the MGBG and LPRL blocks work and how they are trained.

### 3.3  Mask-Guided Bokeh Generator Block

The goal of the MGBG block is to generate the bokeh image in low-resolution. A natural approach here would be to use a single network that takes in the input image and simply produce the output image (i.e. optionally by using additional information such as depth and saliency maps). Since our aim is to also achieve criteria ii), we use a stacked-network formation using two networks, which we call $G1$ and $G2$. For both $G1$ and $G2$, we use accurate and performant architectures (criteria i)); we leverage the architecture of [30] which is a combination of MobileNetv2 [23] and FBNet [29] components.

$G1$ and $G2$ have their own separate tasks; $G1$ takes as input the input image $I_L$ and outputs a grayscale mask $M_L$, whereas $G2$ takes in $I_L$ and $M_L$ as input (as a 4-channel tensor) and generates the low-resolution bokeh rendered image $B_L$. We note that there is no ground-truth for $M_L$, therefore $G1$ is trained without any explicit supervision. $G1$, by leveraging the paired training samples, discovers the areas of refocus. The core advantages of having $G1$ are threefold: first, unlike other methods that do not use any guidance, $G1$ makes our network interpretable as it learns the area of refocus in the form of a mask (see Figure 5). Second, since $G2$ is conditioned on the output of $G1$, we can control the refocus area (criteria ii), see Figure 1). Finally, we can even remove $G1$ in inference and ask the user to provide any type of mask; user-generated, depth, saliency, etc (see Figure 6). This makes our method lightweight and applicable beyond bokeh rendering (i.e. image refocusing).

The 4-channel input to $G2$ ($I_L$ and $M_L$) is passed through the $G2$ network that predicts an RGB image $I_{int}$. We also leverage a dual-attention mechanism that operates over the RGB component of the input (i.e. $I_L$) and $I_{int}$ (see Figure 3). Separate attention modules process $I_L$ and $I_{int}$, and their results are summed
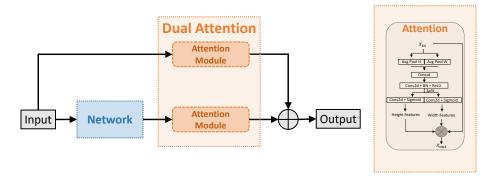
**Fig. 3.** Our dual attention module. The *network* block represents $G2$ network of the MBGB block, as well as the refinement network of the LPR block. The attention module is implemented via [7], details of which are shown on the right.

to produce the low-resolution bokeh image $B_L$. The inspiration for this dual-attention mechanism comes from modern super-resolution approaches [10], where long-distance residual connections are used. Instead of using residual connections directly from $I_L$, we leverage attention modules for visually pleasing results. We use [7] to implement the dual-attention modules. Note that each attention module is learned jointly with the entire pipeline, and they do not share weights. We also note that similar to $G1$, $G2$ is learned without direct supervision. We enforce our losses at the final, high-resolution bokeh image $B_0$ (i.e. the output of the LPR block).

### 3.4   Laplacian Pyramid Refinement Block

The goal of the LPR block is twofold; upsample the low-resolution bokeh rendered image $B_L$ back to the original resolution and refine/improve the results while doing so. Our LPR block is based on [16], with several key differences.

**Preliminaries.** The Laplacian pyramid [1] decomposes the image into low and high frequency components, from which the original image can be reconstructed. At each level of the pyramid, a fixed kernel is used to calculate the weighted average of the neighbouring pixels of the image, resulting in a downsampled version. The downsampled version is then upsampled again, and a high-frequency residual component $h$ is calculated by subtracting the upsampled image from the original one. This process is repeated for each pyramid level, and with the inverse operation, the original image is reconstructed. For each level, we refer to high-frequency residuals as $H = [h_0, h_1, ..., h_{L-1}]$.

**LPR.** Having produced $M_L$ and $B_L$ via the MGBG block, we now aim to upsample $B_L$ to produce our final results $B_0$. We first take $I_L$, $M_L$ and $B_L$ and upsample them to reach the spatial resolution of the lowest (pyramid level 2) high-frequency residual $h_{L-1}$, and then concatenate these four. This concatenated tensor is fed to the refinement network, which produces the refinement

mask $M_R$. We note that we use the same dual-attention mechanism used in G2 (see Figure 3) in the refinement network as well; input attention processes the $B_L$ and the refinement network outputs an RGB image, which are then summed to produce $M_R$. $M_R$ is then processed with fine-tuning convolutional blocks (formed of two convolutional layers, and a LeakyReLu in between), and multiplied with $h_{L-1}$ to produce the output high-frequency residual of the first level. The same process is repeated for every level with the upsampled $M_R$ as the input, except we use the refinement network only on the first level. In LPR, we use a 2-level Laplacian pyramid, however we note that this can be extended to any number of levels depending on the capacity/output-resolution requirements. At the end, we end up with two output high-frequency residuals, which are added to progressively upsampled $B_L$ in each level, until we produce $B_{int}$ in the original resolution. As the final step, we take in the original high-resolution input $I_0$ and multiply it with $M_L$, and sum its results with $B_{int}$ multiplied with $1 - M_L$ to produce the final bokeh rendered image $B_0$. We use masking so that our entire pipeline can focus on where matters (i.e. non-focus areas), since the input masking operation copies the focus area directly from the input image.

**Differences with LPTN.** Our LPR block differs from LPTN in several aspects; i) In addition to $I_L$ and $B_L$, LPR also leverages $M_L$ in the upsampling, which guides the refinement/upsampling process, ii) we propose the dual-attention mechanism in the refinement network which improves it accuracy and iii) we use the mask $M_L$ to both leverage $I_0$ and better blend/integrate $I_0$ and $B_{int}$ spatially in generating the final output $B_0$.

### 3.5   Losses

We use common regression losses, as well as perceptual-aware losses during our training. In total, we use three loss functions; the reconstruction loss $L_1$, the learned perceptual image patch similarity (LPIPS) [31] loss $L_{LPIPS}$ and the structural similarity loss $L_{SSIM}$ [28]. The final loss is a combination of the above losses, which is defined as:

$$L_{total} = 10 \cdot L_1 + 2 \cdot L_{LPIPS} + L_{SSIM}$$

The weights for each loss are based on [22]. As noted before, the loss is applied only on the final bokeh image $B_0$, therefore $G1$ or $G2$ networks in the MGBG block do not have direct supervision. During our experiments, we experimented with applying the losses also over the output of $G2$ ($B_L$) and even over the outputs of each pyramid level (i.e. $B_{int}$), however, we did not see tangible improvements in either configuration.

## 4   Experiments

### 4.1   Dataset

We train our model on the *Everything looks Better with Bokeh! (EBB!)* dataset [8]. It consists of 5K aligned image pairs of wide/shallow depth-of-field, 4600 of

which are used as the training set and 200 images are used for the validation and test sets, respectively. All the images in the dataset have a height of 1024 pixels, however, the width varies between images.

Since *EBB!* is a challenge dataset [9, 11], the ground-truths for validation and test sets are not publicly available. For a fair comparison, we use the *val294* [4] set for evaluation. The *val294* set is based on the EBB!'s train set, where the first 4400 images are used for training, while the rest is used for evaluation.

### 4.2    Experimental setup

**Implementation Details.** We use PyTorch [19] throughout our experiments. Following [30], we initialize the encoders of $G1$ and $G2$ with weights obtained by training on ImageNet. The rest of the learnable parameters (i.e. LPR modules and the decoders of $G1$ and $G2$) are initialized with [6]. All experiments are conducted on a PC with NVIDIA GeForce RTX 3090 GPU and AMD EPYC 7352 CPU. We train our model for 500 epochs, with a batch size of 8, utilizing the Adam optimizer and $2 \cdot 10^{-4}$ learning rate. The trainings are performed with input and output images with the size 1024x1536.
**Evaluation metrics.** We use Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM) [28] and Learned Perceptual Image Patch Similarity (LPIPS) [31] as our evaluation metrics. These metrics are the most widely used metrics used in the synthetic bokeh rendering literature, therefore we choose them to establish a fair comparison with existing methods. We note that many methods also leverage user surveys as another evaluation criteria; despite the value they bring, such surveys are generally not comparable nor reproducible. Furthermore, we believe the key contributions of our method, such as user-editable mask guidance and controllable blur strengths, can be well-represented with qualitative examples. Therefore, we leave user surveys as future work.

### 4.3    Comparison with State-of-the-art

We compare our method with several state-of-the-art methods, such as PyNet [8], DMSHN [5], SKN [13], DBSI [4] and BRViT [18]. We choose these methods based on the availability of their source codes and whether they report results on *Val294* set on EBB!. We use pretrained models and source codes for evaluation (if available), or original results reported in relevant papers for other methods.
**Quantitative comparison.** The results are shown in Table 1. Our method produces competitive results in each metric. In PSNR, our method trails behind others. Our method does quite well in SSIM and LPIPS metrics, it is the second best in SSIM and the best in LPIPS by a considerable margin. We note that BRViT (1st in SSIM) [18] uses quite a large model based on Vision Transformers, and also performs an initial pretraining stage. In general, our model performs competitively, and even outperforms others in perceptual metrics.
**Performance comparison.** We also compare our runtime performance against other state-of-the-art method. For a fair comparison, we only compare against methods that have publicly available source codes or pretrained models, so that

| Method | PSNR↑ | SSIM↑ | LPIPS↓ |
|---|---|---|---|
| SKN [13] | 24.66 | 0.8521 | 0.3323 |
| DBSI [4] | 23.45 | 0.8675 | 0.2463 |
| PyNet [8] | **24.93** | 0.8788 | 0.2219 |
| DMSHN [5] | 24.65 | 0.8765 | 0.2289 |
| Stacked DMSHN [5] | 24.72 | 0.8793 | 0.2271 |
| BRViT [18] | 22.88 | 0.8516 | 0.2558 |
| BRViT [18] ‡ | <u>24.76</u> | **0.8904** | <u>0.1924</u> |
| Ours | 24.50 | <u>0.8847</u> | **0.1718** |

**Table 1.** Comparison with other methods on *Val294* set. SKN results are taken from [18]. BRViT [18] ‡ performs initial pretraining. The best and second best results are in bold and underlined, respectively, for each metric.

| Method | CPU (sec) | # of params | GFLOPs |
|---|---|---|---|
| PyNet [8] | <u>4.089</u> | 47.5M | 5300 |
| Stacked DMSHN [5] | 12.657 | <u>21.7M</u> | <u>480.7</u> |
| BRViT [18] | 30.288 | 123.1M | 650 |
| Ours | **1.591** | **5.4M** | **45.9** |

**Table 2.** Performance comparison with other state-of-the-art methods, on a desktop CPU. Refer to the implementation details section for the evaluation setup.

we can compare them in our system in a fair fashion. The results are reported in Table 2. In constrained scenarios, such as the desktop CPU, we outperform others with a significant margin. Furthermore, the last column shows that our method uses significantly fewer parameters. Finally, it is visible that our method has the fewest GFLOPs among other alternatives, proving its efficiency in terms of computation complexity. We note that our method is not likely to saturate high-end desktop GPUs, therefore it might not be as efficient as other larger methods that can utilize them better. Therefore, we highlight that our method is aimed at achieving good performance in resource-limited environments, such as mobile devices.

**Qualitative comparison.** Example results produced by our method are shown in Figure 4. $G1$ network in our MGBG module successfully learns, in a weakly supervised way, the areas of focus in the ground-truth data. Our learned masks are not strictly binary, which we later show to be useful for simulating different f-stops. Qualitative comparison against existing methods are shown in Figure 5. Our method performs competitively against the alternatives, producing visually pleasing images in multiple cases. We note that competing methods are slower and have significantly larger capacity compared to us, therefore our method might produce slightly worse results in some cases.

### 4.4   Ablation studies

**Component Analyses.** We present the component analyses of our model architecture to show the contribution of the building blocks. The results are shown

**Fig. 4.** Qualitative results of our proposed method. (a) Input images, (b) $M_L$ masks predicted by $G1$ of the MGBG block, (c) rendered bokeh images and (d) ground-truth masks.



(a) Input Image    (b) PyNet [8]    (c) S-DMSHN [5]    (d) BRViT [18]    (e) Ours    (f) Ground Truth
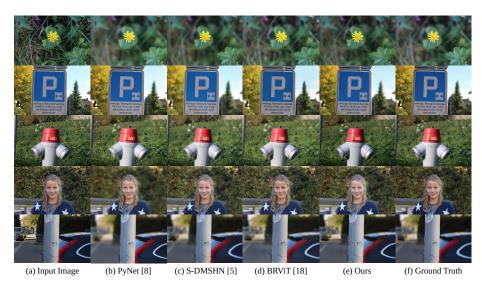
**Fig. 5.** Qualitative comparison with other methods.

in Table 3. First, we test our pipeline without the refinement mask model in the LPR block. This variant (2nd row in Table 3) is clearly the fastest variant, on mobile GPU and desktop CPU alike, showing that the refinement mask network is a bottleneck in performance. However, we lose out quite a bit on LPIPS without the refinement mask model, which justifies its addition to our pipeline. Second, we remove $G2$ from the MGBG block; in this variant (3rd row in Table 3), we essentially force LPR to both learn bokeh rendering and upsample/refine the results. This variant does surprisingly well and performs slightly faster than the full pipeline. Third, we remove the dual attention mechanism from the re-

| Components | PSNR↑ | SSIM↑ | LPIPS↓ | D CPU (s) | D GPU (s) | M GPU (s) | # of params |
|---|---|---|---|---|---|---|---|
| G1 + G2 + LPR | **24.50** | **0.8847** | <u>0.1718</u> | 1.591 | 0.059 | 0.330 | 5.4M |
| G1 + G2 + LPR (w/o ref.) | 24.24 | 0.8814 | 0.1931 | **0.731** | 0.040 | **0.118** | 5.2M |
| G1, no G2 + LPR | 23.91 | 0.8692 | 0.1903 | 1.483 | 0.042 | 0.297 | 2.8M |
| G1 + G2 + LPR (w/o att.) | <u>24.25</u> | <u>0.8825</u> | **0.1717** | 1.581 | 0.052 | 0.319 | 5.2M |
| no G1, G2 + LPR | − | − | − | 1.4732 | **0.039** | 0.312 | 2.8M |

**Table 3.** Ablation study on our model pipeline on *Val294* set. *w/0 ref.* indicates LPR without the refinement mask model. *w/0 att.* indicates LPR without dual attention in the refinement mask model. **D** and **M** stand for desktop and mobile runtimes, respectively. The desktop hardware is detailed in the implementation details section. Mobile runtimes are averaged on 50 runs, and acquired using Samsung Galaxy S22 Ultra with Exynos 2200 processor.

finement mask module in the LPR block. This variant (4th row in Table 3) is the close second in terms of evaluation metrics, and it is slightly faster than the full pipeline. Finally, we remove $G1$ from MGBG and simply use external masks to guide $G2$. This variant (5th row in Table 3) has similar runtimes with the 3rd variant. We note that this variant is a desirable configuration since external mask guidance is a more interactive use case, and also we can not report metrics with this variant since we use external masks. Finally, our full pipeline (1st row in Table 3) produces the best SSIM and LPIPS metrics, showing the effectiveness of our design choices. We note that the other three variants shown in Table 3 (2nd, 3rd and 4th rows) are still competitive to other methods.

**Using different masks.** As explained in earlier sections, our pipeline is flexible as it can be guided by any mask. This mask can be generated by $G1$ of the MGBG block, but also can be generated by a user. Furthermore, one can guide the bokeh generation with depth maps or saliency maps. We provide several scenarios where
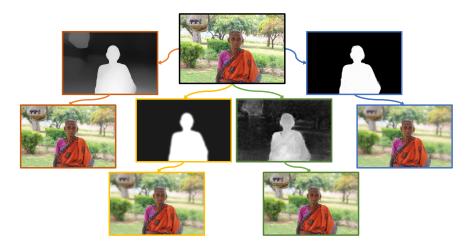


**Fig. 6.** Example outputs using different *types* of masks; <span style="color:orange">orange</span>, <span style="color:yellow">yellow</span>, <span style="color:green">green</span> and <span style="color:blue">blue</span> masks are depth map, user-generated mask, $G1$-generated mask and saliency map, respectively.

we generate the bokeh effect with different types of mask guidance, see Figure 6 for examples. Our method manages to refocus images successfully with various types of mask guidance.

**Simulating different f-stops.** One of the important features of our method is that it can simulate different f-stops/blur strengths. We discover that by modifying the values of the input mask to our model ($G2$), we can achieve a certain level of control over various depth-of-focus effects. Specifically, we do not alter the mask intensities on the in-focus area, but rather alter the intensity values of the background (out-of-focus) areas. Coupled with the fact that we can use any mask to guide our bokeh/refocusing, this also provides another level of interactive experience, which further empowers the user. We provide several examples in Figure 7.
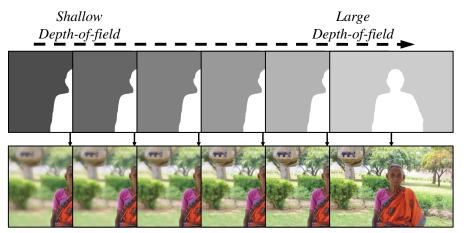


**Fig. 7.** Different f-stop simulation. Each column shows the mask and the related bokeh rendering result. By changing the mask intensity values of the out-of-focus areas, we manage to simulate different f-stops.

## 5    Conclusion

In this paper, we focus on the task of ML-based synthetic bokeh rendering. We focus on three areas of improvement; i) accurate on-device performance with high-resolution images, ii) ability to guide bokeh-generation with user-editable masks and iii) ability to produce blurs with varying strength profiles. We propose the AMPN pipeline, which is formed of the MGBG and the LPR blocks. The MGBG block performs bokeh rendering at low resolutions for low memory footprint and fast processing, whereas LPR progressively upsamples and refines the low-resolution bokeh image generated by MGBG. Owing to its two-network design, the first network of MGBG discovers in-focus areas of images without supervision, in the form a mask. Since the second network of MGBG conditions the bokeh generation on this (weakly-supervised) discovered mask, we can

edit/change the mask freely to guide bokeh generation. This level of control lets us achieve the more abstract functionality of mask-guided image refocusing. Furthermore, by changing the intensities of the mask, we show that we can simulate various blur strengths. Finally, we show that our method produces competitive or better results compared to various alternatives on the EBB! dataset, while running faster.

## References

1. Burt, P.J., Adelson, E.H.: The laplacian pyramid as a compact image code. In: Readings in computer vision, pp. 671–679. Elsevier (1987)
2. Busam, B., Hog, M., McDonagh, S., Slabaugh, G.: Sterefo: Efficient image refocusing with stereo vision. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops. pp. 0–0 (2019)
3. Choi, M.S., Kim, J.H., Choi, J.H., Lee, J.S.: Efficient bokeh effect rendering using generative adversarial network. In: 2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia). pp. 1–5. IEEE (2020)
4. Dutta, S.: Depth-aware blending of smoothed images for bokeh effect generation. Journal of Visual Communication and Image Representation **77**, 103089 (2021)
5. Dutta, S., Das, S.D., Shah, N.A., Tiwari, A.K.: Stacked deep multi-scale hierarchical network for fast bokeh effect rendering from a single image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2398–2407 (2021)
6. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015)
7. Hou, Q., Zhou, D., Feng, J.: Coordinate attention for efficient mobile network design. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 13713–13722 (2021)
8. Ignatov, A., Patel, J., Timofte, R.: Rendering natural camera bokeh effect with deep learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 418–419 (2020)
9. Ignatov, A., Patel, J., Timofte, R., Zheng, B., Ye, X., Huang, L., Tian, X., Dutta, S., Purohit, K., Kandula, P., et al.: Aim 2019 challenge on bokeh effect synthesis: Methods and results. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). pp. 3591–3598. IEEE (2019)
10. Ignatov, A., Romero, A., Kim, H., Timofte, R.: Real-time video super-resolution on smartphones with deep learning, mobile ai 2021 challenge: Report. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2535–2544 (2021)
11. Ignatov, A., Timofte, R., Qian, M., Qiao, C., Lin, J., Guo, Z., Li, C., Leng, C., Cheng, J., Peng, J., et al.: Aim 2020 challenge on rendering realistic bokeh. In: European Conference on Computer Vision. pp. 213–228. Springer (2020)
12. Lei, B., Guo, X., Yang, H., Cui, M., Xie, X., Huang, D.: Abpn: Adaptive blend pyramid network for real-time local retouching of ultra high-resolution photo. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2108–2117 (2022)
13. Li, X., Wang, W., Hu, X., Yang, J.: Selective kernel networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 510–519 (2019)

14. Li, Y., Zhang, K., Timofte, R., Van Gool, L., Kong, F., Li, M., Liu, S., Du, Z., Liu, D., Zhou, C., et al.: Ntire 2022 challenge on efficient super-resolution: Methods and results. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1062–1102 (2022)
15. Li, Z., Snavely, N.: Megadepth: Learning single-view depth prediction from internet photos. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2041–2050 (2018)
16. Liang, J., Zeng, H., Zhang, L.: High-resolution photorealistic image translation in real-time: A laplacian pyramid translation network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9392–9400 (2021)
17. Luo, X., Peng, J., Xian, K., Wu, Z., Cao, Z.: Bokeh rendering from defocus estimation. In: European Conference on Computer Vision. pp. 245–261. Springer (2020)
18. Nagasubramaniam, H., Younes, R.: Bokeh effect rendering with vision transformers (2022)
19. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
20. Peng, J., Cao, Z., Luo, X., Lu, H., Xian, K., Zhang, J.: Bokehme: When neural rendering meets classical rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16283–16292 (2022)
21. Purohit, K., Suin, M., Kandula, P., Ambasamudram, R.: Depth-guided dense dynamic filtering network for bokeh effect rendering. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). pp. 3417–3426. IEEE (2019)
22. Qian, M., Qiao, C., Lin, J., Guo, Z., Li, C., Leng, C., Cheng, J.: Bggan: Bokeh-glass generative adversarial network for rendering realistic bokeh. In: European Conference on Computer Vision. pp. 229–244. Springer (2020)
23. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4510–4520 (2018)
24. Shen, X., Hertzmann, A., Jia, J., Paris, S., Price, B., Shechtman, E., Sachs, I.: Automatic portrait segmentation for image stylization. In: Computer Graphics Forum. vol. 35, pp. 93–102. Wiley Online Library (2016)
25. Shen, X., Tao, X., Gao, H., Zhou, C., Jia, J.: Deep automatic portrait matting. In: European conference on computer vision. pp. 92–107. Springer (2016)
26. Wadhwa, N., Garg, R., Jacobs, D.E., Feldman, B.E., Kanazawa, N., Carroll, R., Movshovitz-Attias, Y., Barron, J.T., Pritch, Y., Levoy, M.: Synthetic depth-of-field with a single-camera mobile phone. ACM Transactions on Graphics (ToG) **37**(4), 1–13 (2018)
27. Wang, L., Shen, X., Zhang, J., Wang, O., Lin, Z., Hsieh, C.Y., Kong, S., Lu, H.: Deeplens: shallow depth of field from a single image. ACM Transactions on Graphics (TOG) **37**(6), 1–11 (2018)
28. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004)
29. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: Fbnet: Hardware-aware efficient convnet design via differentiable neu-

ral architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10734–10742 (2019)

30. Yucel, M.K., Dimaridou, V., Drosou, A., Saa-Garriga, A.: Real-time monocular depth estimation with sparse supervision on mobile. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2428–2437 (2021)

31. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)

32. Zhang, X., Matzen, K., Nguyen, V., Yao, D., Zhang, Y., Ng, R.: Synthetic defocus and look-ahead autofocus for casual videography. ACM Transactions on Graphics (TOG) **38**(4), 1–16 (2019)

33. Zheng, B., Chen, Q., Yuan, S., Zhou, X., Zhang, H., Zhang, J., Yan, C., Slabaugh, G.: Constrained predictive filters for single image bokeh rendering. IEEE Transactions on Computational Imaging **8**, 346–357 (2022)

34. Zhu, B., Chen, Y., Wang, J., Liu, S., Zhang, B., Tang, M.: Fast deep matting for portrait animation on mobile phone. In: Proceedings of the 25th ACM international conference on Multimedia. pp. 297–305 (2017)