Constrained Predictive Filters for Single Image Bokeh Rendering

Bolun Zheng*, Quan Chen*, Shanxin Yuan†, Xiaofei Zhou†, Hua Zhang†, Jiyong Zhang, Chenggang Yan, Gregory Slabaugh, *Senior Member, IEEE*

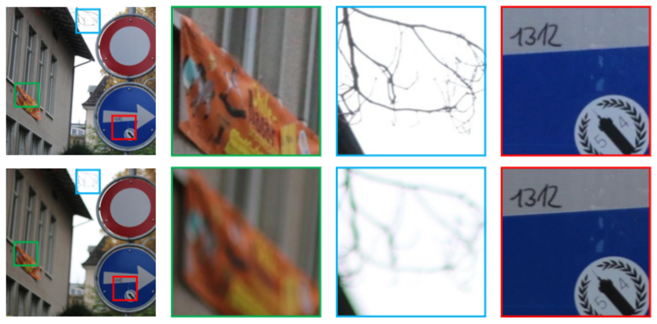
***Abstract*—Bokeh rendering is a technique used to take pictures with out-of-focus areas to highlight regions of interest. Due to limitations in hardware and shooting condition, rendering a bokeh image from a full-focus image has attracted a lot of interest. In this paper, we model bokeh rendering as the combination of salient region retention and bokeh blurring, and propose a neural network to generate a realistic bokeh image from a single full-focus image through end-to-end training. Specifically, we propose a gate fusion block to estimate the salient area, and introduce a constrained predictive filter for salient region retention and bokeh blurring within a unified architecture. Further, we utilize a pixel coordinate-based map to enhance the training. Experimental results illustrate the effectiveness of our model. The comparison with state-of-the-art methods (PyNET [18], DMSHN [12], BGGAN [40], etc.) shows that our model produces better bokeh effects and retains salient objects.**

***Index Terms*—Bokeh rendering, constrained predictive filtering, coordinate map, saliency retention.**

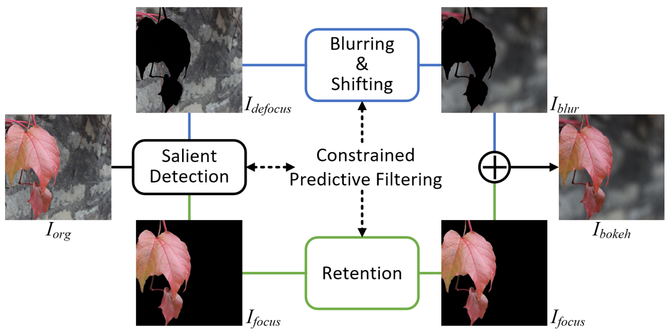
I. Introduction

Over the past few years, image enhancement methods have advanced considerably. In particular, smartphone image enhancement has witnessed increasing interest from both the vision and graphics communities, with numerous methods being published, such as image quality enhancement [52, 24, 50], image de-blurring [9, 6, 31], photo segmentation [3, 55] and image denoising [10, 13]. One topic that gained large popularity over the past years is bokeh rendering.

Bokeh, or depth of field (DoF) effect, is often used in computational photography for aesthetic purposes. This shallow focus technique is used to create images with prominent out-of-focus regions [8] when using a digital single-lens reflex camera (DSLR) with a wide aperture. In the out-of-focus regions, the blur effect on the image edges is caused not only by Gaussian blur, but also by some edge offsets. Fig. 1(a) shows a full-focus image and a real bokeh image, with different out-of-focus and in-focus regions zoomed in, captured with a DSLR camera. Unfortunately, this optical process is not feasible for many smartphone users due to hardware limitations, as smartphone cameras usually have smaller sensors with small apertures, producing full-focus images. Recently some devices have adopted various hardware approaches, for example using multiple cameras or dual pixel sensors [44] to promote synthetic bokeh effect rendering. However, in our case, we propose to render bokeh effect from a full-focus image without any special hardware requirements, and thus our model can be used with any existing device.



(a)



(b)

Fig. 1. (a) Bokeh effect comparison. The top row shows a bokeh-free image and bottom row shows a corresponding bokeh image. (b) The pipeline of the proposed method to produce the bokeh effect from a single image. I_{org} is the full focus image that requires bokeh rendering. The focus part I_{focus} and defocus part $I_{defocus}$ are separated from I_{org} via saliency detection. Then I_{focus} is preserved while $I_{defocus}$ is rendered to I_{blur} using blurring and shifting filters.

Traditional bokeh rendering methods depend on a high-precision depth map to theoretically calculate the blurring kernels, and produce the bokeh effect by filtering the input image with the calculated kernels. However how to obtain a high-precision depth map from a single image input is challenging. Besides filtering with theoretically calculated kernels also suffers from the complicated applications for robust and bokeh rendering. Recently, deep learning based methods

B.Zheng and Q.Chen contributed equally to this paper.

B.Zheng, Q.Chen, X.Zhou, J.Zhang, and C.Yan are currently with the school of automation, Hangzhou Dianzi University.

H.Zhang is currently with the school of computer science, Hangzhou Dianzi University.

Shanxin Yuan (email: shanxinyuan@gmail.com), Xiaofei Zhou (email: zxforchid@outlook.com) and Hua Zhang (email: zhangh@hdu.edu.cn) are the corresponding authors of this paper.

S.Yuan is currently with Huawei Noah's Ark Lab.

G.Slabaugh is currently with Queen Mary University of London.

show promising performance on single image bokeh rendering task, and have won the latest two bokeh effect synthesis competitions [22, 23]. Currently, major flagship and mid-range mobile devices have dedicated hardware to accelerate deep learning models [21]. The performance of such hardware is already comparable with mid-range Nvidia GPUs [20], making it possible to run deep learning methods for bokeh rendering of images on smartphones.

In this paper, we model the bokeh rendering task with two sub-tasks: salient content retention, and bokeh blurring. We propose a multi-scale predictive filtering network (MPFN) to solve the two sub-tasks simultaneously within a unified architecture. Specifically, we propose constrained predictive filters (CPF) with three types of kernels shown in Fig. 5, including a retaining kernel K_R, blurring kernel K_B, and shifting kernel K_S; where K_R is for salient content retention, and both K_B and K_S are for bokeh blurring. We also design the gate fusion block (GFB) to detect the focused region to assist the constrained filters in processing RGB images. In addition, we propose a 2-channel coordinate map (CM) to solve the position consistency of input patches in training. We concatenate the CM with the input RGB image along the channel dimension as the final input to the network. To demonstrate the effectiveness of our MPFN, we conduct extensive ablation experiments to verify the impact of each module. Besides, the comparison with state-of-art methods proves that our model synthesize better bokeh effects and retains focused objects accurately.

In conclusion, our main contributions are as follows:

- We propose a novel multi-scale predictive filter network (MPFN) model for bokeh rendering with two novel modules: gate fusion block (GFB) and constrained predictive filter block (CPFB).
- We propose a constrained predictive filter for bokeh rendering. The CPF introduces three types of filter kernels to precisely retain the salient content of the image and blur the remaining areas. We also propose a gate fusion module for detecting salient regions and fusing the features from salient and non-salient regions.
- We propose a coordinate map (CM) to solve the spatial consistency of input patches in training. We also demonstrate that introducing the coordinate map (CM) can restore the side effects caused by inaccurate prior knowledge (e.g. depth map and saliency map) for single image bokeh rendering.

The rest of this paper is organized as follows: In Section II, we briefly review the related work. In Section III, we introduce our method and describe the architecture of our MPFN in details. Experimental results are presented in Section IV. In Section V, we discuss the limitation of our method and future work. Finally, we conclude this paper in Section VI.

II. RELATED WORK

Infusing bokeh effects into a full-focus image generates aesthetically pleasing images, drawing the observer's gaze to the regions in focus. Several methods [41, 29, 45, 25] have been proposed for bokeh rendering using both conventional and learning-based techniques.

Non-learning Method. Riguer *et al.* [41] propose a gathering-based method for bokeh effect rendering, and apply several filters of different shapes and weights to obtain diverse effects. Kodama *et al.* [29] improve upon [41] by replacing the convolution operation in the spatial domain with a multiplication operation in the frequency domain to accelerate the rendering procedure. Lee *et al.* [33] introduce a scattering-based method for simulating more accurate bokeh for defocused highlights, which are obtained using the texturing extension of GPU point sprites. Buhler *et al.* [5] propose a method based on distributed ray tracing, where a user-specified probability density function represents the light intensity distribution within the circle of confusion. Bokeh rendering and super-resolution are selectively integrated in [45] into one scheme. An anisotropic filter is designed to render the bokeh. Jeong *et al.* [25] improve the effect of real-time rendering from extrinsic (scene-related) and intrinsic (relating solely to optical systems) appearance.

Deep Learning Method. With the development of deep learning, more attention has been paid to its use for automatic bokeh rendering [39, 40, 37, 27]. Ignatov *et al.* [19] propose an efficient multi-level CNN approach to render bokeh effects on a mobile GPU, named PyNET. To obtain a more efficient algorithm on mobile devices, Dutta *et al.* [12] proposed a stacked encoder-decoder structure similar to PyNET. Besides, Qian *et al.* [40] firstly introduce a GAN-based model to obtain the bokeh rendering effect, and greatly improve the visual perception of the rendered images. However, the full convolution models use fixed receptive fields to process all areas of the image with the same parameters, which is inefficient in processing focused regions, because the theoretically focused regions does not need to be processed. Yang *et al.* [22] adopt a selective kernel convolution and construct a two-stage *BokehNets* approach, which allows neurons to adaptively adjust their receptive field size based on input and achieves the best SSIM results in AIM2019 Bokeh Effect challenge. To distinguish the convolution parameters between the focused regions and the background regions, Purohit *et al.* [39] propose a dense filter network DDDF based on dynamic filtering mechanism[4]. Due to the limitation of computational cost, the filtering receptive field of the DDDF is limited and the generated bokeh blur is incomplete. To balance the filtering receptive field and computational cost, we propose a multi-scale predictive filtering network. Different from the conventional filters, the constrained filter kernels generated by our model process the focused regions and the background regions simultaneously.

With the development of computer vision, depth estimation [14, 30, 34], defocus estimation [2, 32, 7] and saliency detection [54, 38, 43, 56] have made significant progress, which also provide more directions for solving bokeh rendering tasks. Many methods [48, 37, 11, 17] utilize these prior knowledge to synthesize bokeh effects. Luo *et al.* [37] utilize radiance estimation and defocus estimation [47] to predict the relationship between the amount of blur and the intensity of pixels in each image, and then synthesizes the bokeh effect with pre-defined blur kernels. Depth priors seem to be more suitable for bokeh rendering tasks. Methods [39, 19, 12]

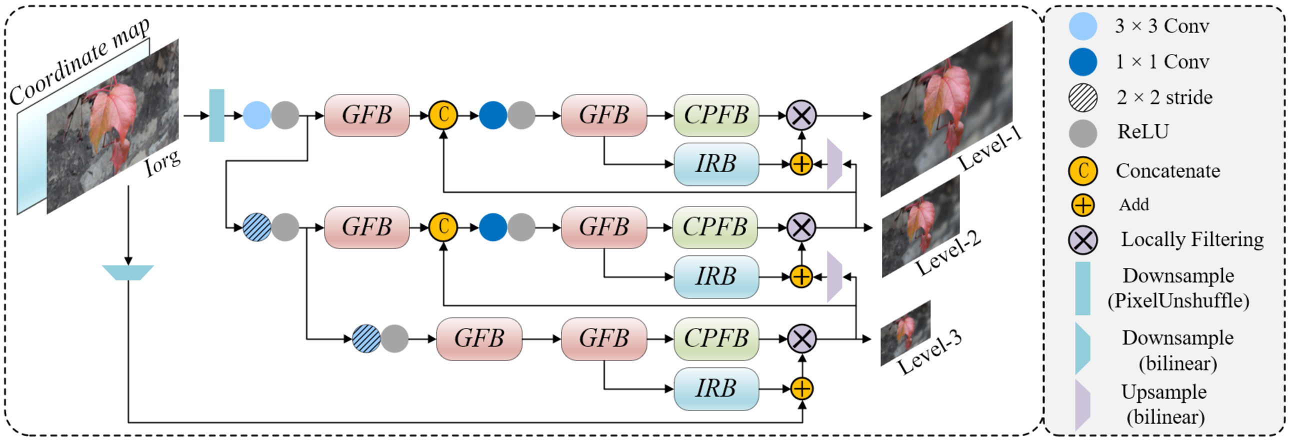


Fig. 2. The architecture of the multi-scale predictive filter network. The MPFN works in three levels and has three different types of blocks, which are the gate fusion block (GFB), constrained predictive filter block (CPFB) and image reconstruction block (IRB).

directly send the estimation depth maps [34] to the network as input, and the experiments prove that the inaccurate depth maps may reduce the performance of the model [12]. To using depth maps completely, Dutta *et al*. [11] treat the bokeh image as the weighted sum of the original image and a number of differently smoothed images, where the corresponding weight maps are predicted by a depth-estimation module. Similar to method [11], Huang *et al*. [17] propose a Depth-guided Bokeh Synthesis Network which can synthesis, refocus and adjust the level of bokeh of the images. Kaneko *et al*. [27] proposes an AR-GANs model which generates a deep DoF image and depth from a random noise to render a shallow DoF image via aperture rendering. However, this method cannot directly process existing full-focused images. Furthermore, Purohit *et al*. also send the estimated saliency maps [15] into the network as input. Xian *et al*. [48] integrate the depth estimation module and saliency detection module into the same network, and synthesis the bokeh effects using a physically motivated method. However, due to the large difference between the dataset used for prior knowledge training and the bokeh dataset, the accuracy of prior knowledge may not be reliable, and the imprecise prior may degrade models. Besides, there is currently no method to synthesize bokeh effects based solely on salience. To overcome the several challenges caused by prior knowledge, we propose a salient-based bokeh rendering network. We propose an unsupervised saliency detection module that fuses foreground and background features, to guide the constrained filter to generate more realistic bokeh effects.

III. Proposed Method

As shown in Fig. 1(a), comparing a full-focus image and a real bokeh image captured by a DSLR camera, we observe that the bokeh image consists of focused salient regions and defocused blurry regions. We formulate the bokeh rendering task as two sub-tasks: salient content retention, and bokeh blurring, as shown in Fig. 1(b). In this way, bokeh rendering from a single all-in-focus image can be formulated as:

$$I_{bokeh} = \Psi\left(I_{org}\right) + \Gamma\left(I_{org} - \Psi\left(I_{org}\right)\right), \tag{1}$$

where I_{org} is the full-focus image, I_{bokeh} is the rendered bokeh image, Ψ is a function of saliency detection that selectively retains salient content in the image, and Γ is a bokeh blurring function that simulates bokeh rendering effect. However, limited by the accuracy of Ψ, directly applying Eqn. 1 to produce bokeh images would probably retain the defocused areas, or blur the focused areas by mistake. To overcome this limitation, we proposed a constrained predictive filter (CPF) to address both sub-tasks, i.e., to handle foreground pixels retention, background pixels flexible blurring and shifting within a unified framework. Specifically, the CPF has three types of kernels, including a retaining kernel K_R, blurring kernel K_B, and shifting kernel K_S; where K_R is for salient content retention, and both K_B and K_S are for bokeh blurring. With the CPF, the Eqn. 1 can be rewritten as:

$$I_{\text{bokeh}} = F_{K_R}(I_{org}) + F_{K_B,K_S}(I_{org} - \Psi(I_{org})) \tag{2}$$

where F_K denotes the filtering operation using the corresponding filtering kernel K. So that we can further rewrite the above eqation as:

$$I_{\text{bokeh}} = F_{CPF}(I_{org}), CPF_{(i,j)} \in \{K_R, K_B, K_S\} \tag{3}$$

where F_{CPF} denotes the filtering operation using the constrained predictive filter, and $CPF_{(i,j)} \in \{K_R, K_B, K_S\}$ denotes the filter kernel located at (i, j) belongs to one of the three types $\{K_R, K_B, K_S\}$. Besides, we design a gate fusion block (GFB) to explicitly detect the salient regions, thus guiding the CPF to retain focused regions more accurately.

With the two modules as major components, we construct a multi-scale architecture named Multi-scale Predictive Filtering Network (MPFN) for synthesizing a coarse-to-fine bokeh effect. Our MPFN works in three scales and has three types of blocks, including GFB, CPFB and the image reconstruction block (IRB). The details of each block are described in Sections A and B. In addition, we also designed a Coordinate Map to enhance the model's training and inference, the details of which are described in Sec.C.

The architecture of MPFN is shown in Fig. 2. After concatenating the coordinate map and image I_{org} along the channel dimension, the input feature map with shape $h \times w \times c_{total}$ is transformed into I_{sub} with a shape of $\frac{h}{2} \times \frac{w}{2} \times 4c_{total}$ through a space-to-depth layer. Then through a 3×3 convolution followed by a ReLU activation layer, the channel dimension of I_{sub} becomes n_D. With the tensor I_{sub} as input, the following network has three levels. There is a convolutional layer with a stride of 2 at the beginning of level 2 and level 3 to downsample the features' dimension. Consequently, each level processes information at a different scale. Following Eqn. 1, each level sequentially executes the salient content retention, bokeh blur and the final output reconstruction. For level-2 and level-3, the output will be used as an additional input to the finer-level (level-2 and level-1) sub-network. There are two reasons for doing this: 1) for better learning of salient features, and 2) after the upsampling layer, the output is sent to the IRB for image reconstruction.

A. Salient Content Retention

As mentioned above, detecting the salient regions of the input image is required for CPF to generate satisfied bokeh effects. We propose the GFB for detecting salient features, see Fig. 3. Assuming the input of the GFB is x_{in}^{GFB}, a dense block first extracts deep features, denoted as Z_{deep}. The dense block has N densely connected [16] $3 \times 3\, n_D$-channel convolution with ReLU activation layers ($CD_N \sim CD_1$). Afterwards, we design a salient feature detection block to detect salient features Z_s from Z_{deep}. This block includes three 1×1 convolution with the ReLU activation layers ($CR_1 \sim CR_3$) and another $Sign$ activation layer. The CR_1 firstly fatten the Z_{deep}. Next, the CR_2 and CR_3 are sequentially introduced to produce the raw attention map, which will then be activated by $Sign$ layer to obtain the binary salient feature Z_s. Finally, we design a gating mechanism to evaluate the reliability of the salient features Z_s, and to fuse the features from salient and non-salient regions. It is formulated as:

$$GFB(x) = Z_{trans} \cdot Z_s + (1 - Z_s) \cdot x_{in}^{GFB} \tag{4}$$

where Z_{trans} denotes the deep features obtained from Z_{deep} going through the convolution layer CK. TABLE I lists the attributes of all learnable layers in GFB.

TABLE I
ATTRIBUTES OF LEARNABLE LAYERS IN GFB.

Layer	$CD_1 \sim CD_N$	CR_1	CR_2	CR_3	CK
kernel	3×3	1×1	1×1	1×1	1×1
Output Ch.	n_D	$n_D \cdot 4$	$n_D \cdot 2$	1	$n_D \cdot 2$

We visualize the estimated salient features in Fig. 4. The proposed salient feature detection block can accurately detect the salient regions especially around edge areas. For smooth areas, since retaining or blurring will produce a similar effect, any estimation is acceptable.

B. Constrained Predictive Filtering

Following the formulation defined in Eqn. 2, to efficiently handle both salient and non-salient features, we propose a constrained predictive filter to address both sub-tasks within a unified framework. The constrained predictive filter consists of two parts. The first part is the constrained filter generation module for generating specific types of filter kernels, and the second part is the predictive filtering module to filter the input image using the generated constrained filters, see Fig. 5 and Fig. 6 for illustration.

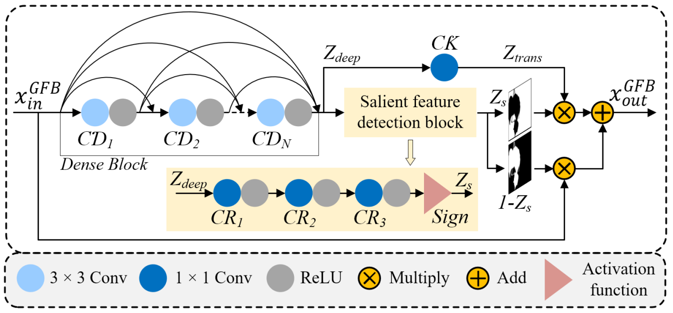


Fig. 3. The structure of the gate fusion block.

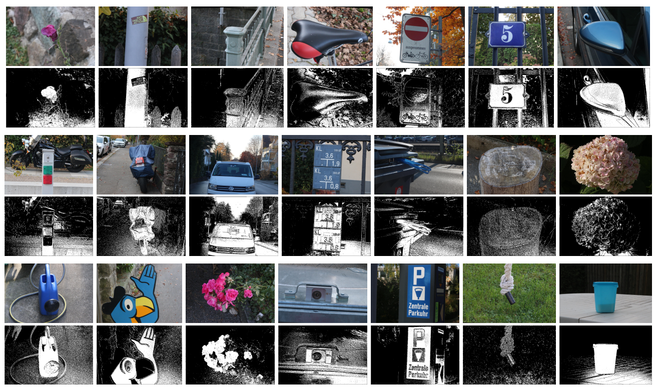


Fig. 4. The salient regions $(1 - Z_s)$ estimated by the GFB. Following Eqn. 4, the bright regions denote the salient regions. Rows 1, 3 and 5 are the full-focus inputs and rows 2, 4 and 6 are the corresponding salient features. As indicated in Fig. 3, the dark areas denote the salient regions. Note that the salient features are binary masks

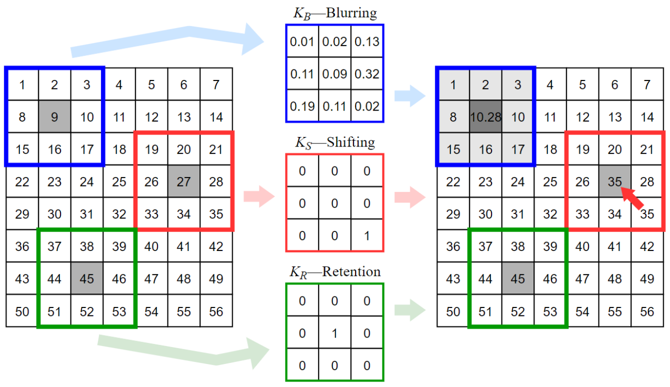


Fig. 5. Description of filter kernels. There are three types of filter kernels. The blue box K_B represents the blurring filter, which is used to blur the pixel in the domain, while the red box K_S implements a local shifting of the pixel, and the green box K_R is used to retain the current pixel.

Constrained Filter Generation. The constrained filter generation module F_G takes an input $I_{org} \in \mathbb{R}^{c \times h \times w}$, where c, h and w are number of channels, height and weight of the

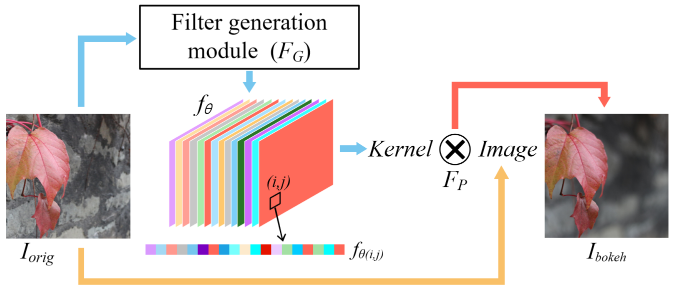


Fig. 6. Constrained predictive filter. The constrained predictive filter generation module generates filters with different kinds kernels based on each pixel point, and the predictive filtering module processes the input full-focus image I_{org} with the filters to obtain the final image I_{bokeh} with bokeh rendering effect.

input I_{org}, respectively. It generates filters f_θ parameterized by parameters $\theta \in \mathbb{R}^{k^2 \times h \times w}$, where k is the filter size:

$$f_{\theta(i,j)} = F_G\left(I_{\text{org}}(i,j)\right) \tag{5}$$

Fig. 5 illustrates the filtering based operations. Given a single channel image X, we propose three types of filtering kernels to achieve the corresponding operations: K_R is for retaining a pixel, K_B is for flexible blurring, and K_S is for pixel shifting. Specifically, K_R, whose the center element is 1 and other elements are 0s, retains the current pixel. K_B, whose elements are between 0 and 1 and sum to 1, achieves adaptive blur in the neighborhood. K_S, with one non-central element as 1 and other elements 0s, implements a pixel shift to the center in the neighborhood. The corresponding operations can be expressed as:

$$Y(i,j) = F\left(X(\Omega_r(i,j)), \theta(i,j)\right) \tag{6}$$

where $F(\cdot,\cdot)$ denotes the filtering operation, r is the radius of the filter kernel, $\Omega_r(i,j) = \{(m,n) \mid m,n \in [i-r, i+r]\}$ denotes a $(2r+1)\times(2r+1)$ sized region centered on (i,j), and $\theta(i,j) \in \{K_R, K_B, K_S\}$ denotes the filter kernel located at (i,j). Obviously, the K_R, K_B and K_S meet the following unified conditions:

$$\sum_{m=0}^{2r+1}\sum_{n=0}^{2r+1} \theta(m,n) = 1 \tag{7}$$

$$0 \le \theta(m,n) \le 1, \forall (m,n) \in \Omega_r(i,j) \tag{8}$$

Therefore, we propose to constrain filter parameters with $Softmax$ function when learning the kernel elements.

Predictive Filtering. As shown in Fig. 6, the predictive filtering module F_P takes the image I_{org} and constrained filters f_θ as input and estimates the filtered result I_{bokeh}, which can be expressed as:

$$I_{bokeh}(i,j) = F_P\left(I_{\text{org}}(i,j), f_{\theta(i,j)}\right) \tag{9}$$

Given a filtering kernel, the effective filtering area is limited by the kernel size k. A larger kernel size leads to a larger effective filtering area, but also increases the computation cost. Therefore, we propose a multi-scale filtering architecture to achieve a large filtering kernel with multiple small filtering kernels. Assuming there are in total S scales in the network, and the size of filtering kernel from the i-th ($i \in [1,S]$, $i = 1$

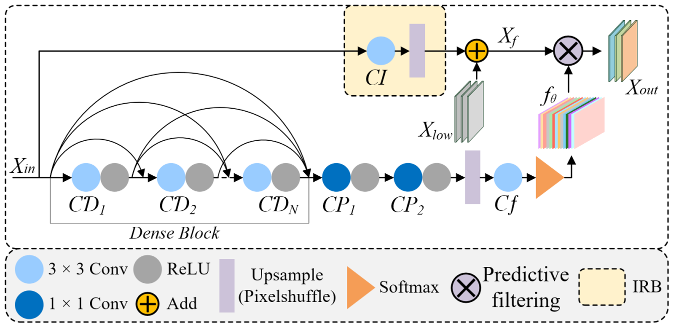


Fig. 7. The structure of constrained predictive filtering block and image reconstruction block.

denotes the original scale) scale is k_i, then the final effective filtering area k_{total} is $k_{\text{total}} = \sum_{i=1}^{S} k_i \cdot 2^{S-i}$. Since both the model scale and the filter kernel size affect the final filtering range, we consider these two factors in the model design.

CNN Modules. We design a CNN-based module named constrained predictive filtering block (CPFB) following Eqn. 4 - 9. To better work with the multi-scale filtering architecture, we additionally introduce an image reconstruction block (IRB) to generate the filtering input. Fig. 7 illustrates the detailed structures of CPFB and IRB.

Given the input X_{in}, one branch generates constrained predictive filters. We first extract the deep features X_{deep} through a dense block similar to the GFB. After that, two 3×3 convolutional layers (CP_1, CP_2) and a pixel-shuffle layer are used, and the width and height of X_{deep} are doubled and the channel of X_{deep} is n_D. Finally, a $3\times3\,k^2$-channel convolutional layer (Cf) followed by Softmax activation is applied to generate the constrained filters f_θ, where the number of channels of the output depends on the filter kernel size k.

The other branch is used to reconstruct the image. The input of the IRB is also X_{in}. We adopt a 3×3 convolution CI and the pixel-shuffle layer to reconstruct the image X_f. Additionally, the coarser scale output X_{low} is added to X_f for enhancing the details. Finally, image X_f and constrained predictive filters f_θ are sent to the predictive filtering module to get the image X_{out} with a bokeh rendering effect, which can be expressed as:

$$X_{\text{out}} = \sum_{c\in\{R,G,B\}} \sum \left(\left(X_f^c \otimes W\right)\cdot f_\theta\right) \tag{10}$$

where X_f^c is the single channel feature map and W is the filtering window. The detailed structure of the predictive filtering module is shown in Fig. 8. We designed a filtering window W with the shape of $k^2\times k\times k$, which can be defined as:

$$W_{(x,y,z)} = \begin{cases} 1 & (y = [x/k], z = (x \bmod k)) \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

TABLE II lists the attributes of all learnable layers in the CPFB and IRB.

C. Coordinate Map

Due to memory limitations, the training image pairs are usually cropped into multiple patches to be sent to the network

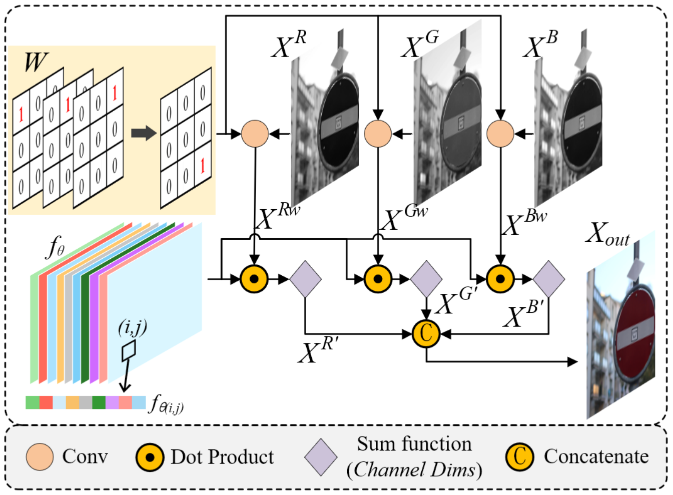


Fig. 8. The structure of Predictive Filtering module.

TABLE II
ATTRIBUTES OF LEARNABLE LAYERS IN THE CPFB AND IRB.

Layer	$CD_1 \sim CD_N$	CP_1	CP_2	Cf	CI
kernel	3×3	1×1	1×1	3×3	3×3
Output Ch.	n_D	$n_D \cdot 2$	$n_D \cdot 4$	k_i^2	12

for image manipulation tasks. However, this limitation can be a great challenge for bokeh rendering. As shown in Fig. 9, patches in different parts of the image may require different processing to retain focus of salient objects or blur non salient regions. We observe that the patch's position is important, and propose a position-based coordinate map to solve this problem. The coordinate map is a 2-channel tensor with the same height and width of the input image. The element at the (x, y) in the coordinate map denotes the relative position from the center of the image, which can be expressed as:

$$C(x,y) = (\frac{x}{W}, \frac{y}{H}) - 0.5 \tag{12}$$

where C denotes the coordinate map, and W and H are the width and height of the input image. We concatenate the coordinate map and the input image along the channel dimension to formulate a 5-channel input for the network. In this way, no matter which parts are cropped from the image, the corresponding coordinate map will provide the relative position to guide the network to produce a more accurate bokeh effect.

D. Loss Function

To train our MPFN, we construct a multi-scale supervision architecture to guide the network to make predictive filtering level-by-level. For each level, we adopt the L1 loss as the base loss function. However, as the L1 loss is a point-wise loss, it cannot provide edge information to guide the predictive filtering to make appropriate pixel retention or bokeh rendering. Inspired by [51, 53], we adopt the Advanced Sobel Loss (ASL)

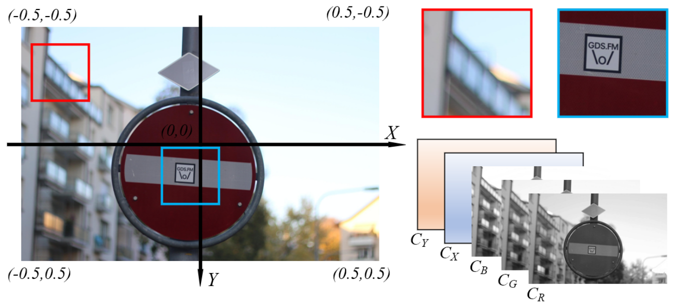


Fig. 9. An illustration of the coordinate map. The blue box represents the in-focus content and is located in the center of the image, with coordinates close to (0,0). The red box represents defocused content with coordinates away from the origin. The coordinate map and the image are concatenated to form a 5-channel image.

and combine it with the L1 loss to formulate the loss function for each level, which can be expressed as:

$$\mathcal{ASL}\,(I_{\text{out}}^s, I_G^s) = \frac{1}{N}\sum \mid Sobel^*(I_{out}^s) - Sobel^*(I_G^s) \mid \tag{13}$$

$$loss\,(I_{\text{out}}^s, I_G^s) = \mathcal{L}1\,(I_{\text{out}}^s, I_G^s) + \lambda \cdot \mathcal{ASL}\,(I_{\text{out}}^s, I_G^s) \tag{14}$$

where I_G^s and I_{out}^s denote the groundtruth and the output of MPFN at level s, $Sobel^*$ denotes the advanced Sobel filtering referring, $\mathcal{L}1$ and $\mathcal{ASL}$ denote the L1 loss and ASL loss, and λ is a hyper-parameter to balance the L1 loss and ASL, which is set to 0.25. Then the total multi-scale supervision loss can be obtained as:

$$Loss = \sum_{s=1}^{S} loss\,(I_{\text{out}}^s\,, I_G^s) \tag{15}$$

IV. EXPERIMENTS

In this section, we conduct extensive ablation studies and compare the model with the state-of-the-art methods both quantitatively and qualitatively on the *Everything is Better with Bokeh!* (EBB!) dataset, which is also used in the AIM2019 and AIM2020 bokeh effect challenges [22, 23]. The dataset contains 4694 bokeh and bokeh-free image pairs, and we separate the first 100 pairs as a testing set, called the Test100 set, for the quantitative evaluation, and the remaining 4594 image pairs for training. Note that the Test100 set is not used for training. Then, the dataset also contains 200 bokeh-free images without public ground-truth, denoted as Test200 set, only for qualitative evaluation. In addition, we also collect 36 full-focus images using a Huawei Mate30 cellphone to construct an additional testing set (Real36) to further evaluate all compared models. Fig. 10 exhibits all images of the Real36 dataset. Specifically, this dataset doesn't have groundtruth images.

A. Implementation Detail

For the MPFN model, we adopt the following settings, with n_D=64, N=5. The model uses standard Tensorflow packages [1], and runs on a machine with an Intel i7-10700K CPU, 128GB RAM, and a Nvidia GTX 2080Ti GPU with 11GB memory. For the training dataset, the initial learning rate is

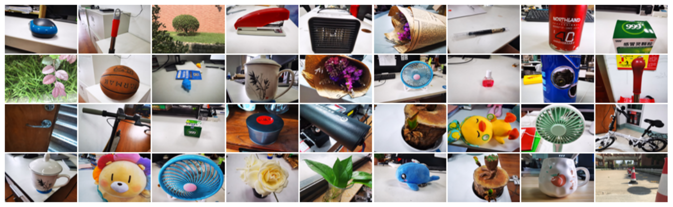


Fig. 10. The complete full-focus Real36 dataset captured with a Huawei Mate30.

set to 10^{-4} and is halved every 30 epochs. The Adam [28] is used as our training optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$. All images (including the corresponding coordinate maps) are cropped to 256×256 patches, with the batch size set to 16. It takes around 80 epochs to train MPFN. In the inference phase, the full-focus images and corresponding coordinate map are concatenated along the channel dimension, then sent to the MPFN for processing.

To fairly evaluate all compared models, we adopt PSNR, SSIM[46] and LPIPS[49] as the basic quantitative metrics. Further, we also adopt the Mean Opinion Score (MOS) by a user study for explicit image quality assessment. To avoid random subjective bias, we first let the 12 participants score the generated bokeh images from Test100 with the real bokeh images as reference using a five point scale (0 - almost identical from the ground truth, 4 - mostly different). Then, they are asked to score the generated bokeh images from Test200 and Real36 with no reference (0 - almost natural rendering, 4 - mostly identical from the input).

B. Ablation Studies

To verify the effectiveness of components and settings in the MPFN, we conduct extensive ablation studies, including the evaluations of model levels, model structures, prior knowledge, constrained predictive filter sizes and loss functions. Moreover, as the unified filtering architecture is one of the major contribution of this work, we design a visualization method to indicate the blurring degrees of predicted filters, which can be expressed as:

$$k_{visual}^{(i,j)} = \sum_{m=0}^{2r+1} \sum_{n=0}^{2r+1} \theta(m,n) \times \frac{(|m-r| + |n-r|)}{2r+1} \tag{16}$$

$$\forall (m,n) \in \Omega_r(i,j), 0 \leq k_{visual}^{(i,j)} \leq 1 \tag{17}$$

The value of $k_{visual}^{(i,j)}$ represents the degree of blurring of the constrained filter kernel to the pixel (i,j). When $k_{visual}^{(i,j)} \to 0$, the pixel (i,j) is tended to be retained. While the pixel (i,j) is tended to be blurred when $k_{visual}^{(i,j)} \to 1$.

Model levels. As described in previous sections, our proposed model has three levels for extracting information at different scales. In order to verify this setting we tested variants of our method with different numbers of levels. The model with only level 1 is called MPFN-1, the model with level 1 and 2 is called MPFN-2, and the model we use for final training is MPFN-3. In addition, we use the same subdivision method that fused the information from level 2 into level 1,

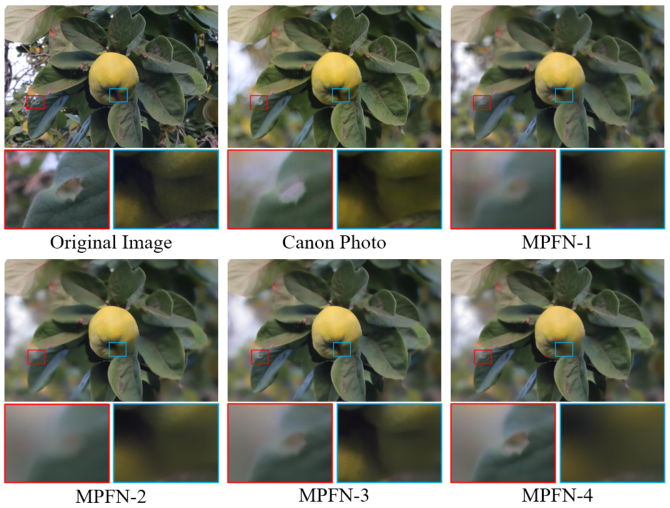


Fig. 11. Visual results obtained with different levels.

TABLE III
PERFORMANCE COMPARISON OF MPFN WITH DIFFERENT LEVELS. THE AVERAGE RUNNING TIME IS MEASURED ON A 1536×1024 SIZED IMAGE.

Model	MPFN-1	MPFN-2	MPFN-3	MPFN-4
PSNR/SSIM	23.79/0.8716	24.09/0.8823	**24.25/0.8861**	24.16/0.8822
LPIPS	0.2559	0.2389	**0.2297**	0.2362
MOS	2.01	1.64	**1.35**	1.45
Time(s)	**0.009**	0.012	0.014	0.019

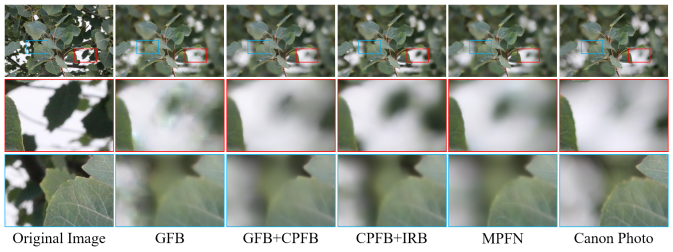


Fig. 12. Visual results obtained with model using different structures.

further complicating the model by designing level 4 and fusing the output image into level 3, called MPFN-4. All models are trained with the same training parameters and datasets, and the Test100 set is used to calculate the PSNR and SSIM. As shown in TABLE 11, the proposed MPFN-3 achieves the best PSNR and SSIM on the Test100 set. Moreover, the visual results provided in Fig. 11 demonstrate that the MPFN-3 achieves more accurate bokeh blurring and produces more realistic details of in-focus parts than other models. Therefore, we adopt the MPFN-3 as the benchmark in the following experiments.

GFB vs. CPFB vs. IRB. To investigate the effectiveness of these blocks, we constructed another three models to performed ablation experiments. The M-GFB denotes the model constructed with only GFBs. The M-GFB-CPFB denotes the model constructed with GFBs and CPFBs. The M-CPFB-IRB denotes the model constructed with CPFBs and IRBs. Fig. 15 illustrates the details three models.

TABLE IV
PERFORMANCE COMPARISON OF MPFN WITH DIFFERENT SETTINGS.

Model Structure			PSNR	SSIM	LPIPS
GFB	CPFB	IRB			
✓			24.12	0.8815	0.2327
✓	✓		24.06	0.8840	0.2383
	✓	✓	24.02	0.8807	0.2414
✓	✓	✓	**24.25**	**0.8861**	**0.2297**
CM	DM	SM	PSNR	SSIM	LPIPS
			24.04	0.8836	0.2370
✓			**24.25**	**0.8861**	**0.2297**
	✓		23.99	0.8812	0.2430
		✓	24.14	0.8847	0.2341
✓	✓		24.14	0.8829	0.2336
✓		✓	24.22	0.8849	0.2359
	✓	✓	24.13	0.8858	0.2327
✓	✓	✓	24.19	0.8817	0.2359
GFB Activation			PSNR	SSIM	LPIPS
Sigmoid	Sign				
✓			24.19	0.8861	0.2290
	✓		**24.25**	**0.8861**	**0.2297**
CPFB Activation			PSNR	SSIM	LPIPS
ReLU	Softmax				
✓			24.18	0.8845	0.2348
	✓		**24.25**	**0.8861**	**0.2297**

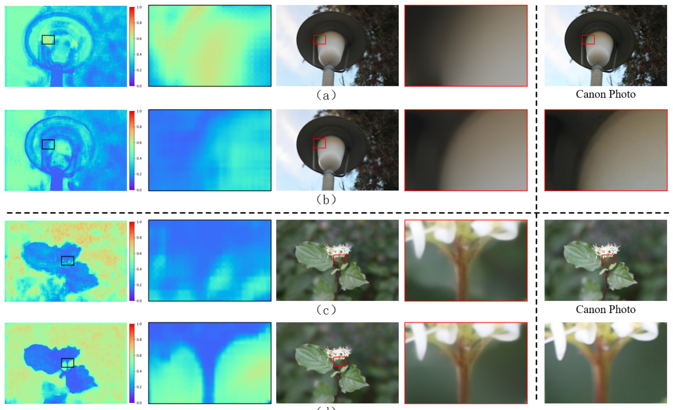


Fig. 13. Visual constrained filter kernels and results obtained with MPFN and M-CPFB-IRB.

Referring to TABLE IV, removing IRB substantially reduces the PSNR, SSIM and LPIPS performances. The IRB fully utilizes the features enhanced by GFBs to reconstruct the images to be filtered, which could better associate with the CPFB to produce accurate bokeh effect. Comparing M-GFB with M-GFB-CPFB, introducing CPFB temporally leads a slight performance reduction. However, from the visualized results shown in Fig. 12, the CPFB clearly contributes to produce a more smooth and natural blurring effect on defocused areas. Moreover, GFB plays a significant role for saliency detection.

TABLE V
PERFORMANCE COMPARISON OF MPFN WITH DIFFERENT FILTER SIZES.

Kernel size	3-5-7	5-7-9	7-9-11	9-11-13
PSNR/SSIM	24.23/0.8841	24.19/0.8833	24.24/0.8855	**24.25/0.8861**
LPIPS	0.2335	0.2369	0.2356	**0.2297**
MOS	1.37	1.32	1.31	**1.20**

TABLE VI
PERFORMANCE COMPARISON OF MPFN TRAINED WITH DIFFERENT LOSS FUNCTIONS.

Loss	L2	L1	L1 + λ·Perceptual	L1 + λ·Laplace	L1 + λ·SSIM	L1 + λ·ASL
λ	-	-	1.0	0.5	0.2	0.25
PSNR	24.01	24.04	24.06	24.07	24.15	**24.25**
SSIM	0.8759	0.8739	0.8739	0.8815	0.8806	**0.8861**
LPIPS	0.2295	0.2330	**0.1993**	0.2408	0.2243	0.2297
MOS	1.59	1.46	1.37	1.48	1.46	**1.23**

Removing GFB leads a sharply performance reduction. From the visualized results of predicted filters shown in Fig. 13, the CPFB struggles to predict accurate filters and fails to preserve the salient content and blur the background. In general, all three blocks are carefully designed and significant for rendering bokeh effect.

CM vs. Depth map vs. Saliency map. As mentioned above, we proposed a coordinate map to assist the network to sense the patch's position in the training stage, which is important to bokeh rendering. Compared to another priors depth map (DM) and saliency map (SM), both DM and SM are obtained from the input image by using specifically trained models[34, 36]. The priors brought by corresponding models certainly affect the MPFN. We listed the performances of introducing different combinations of priors (CM, DM and SM) to MPFN in TABLE IV. Singly Introducing CM or SM brings performance gains in all indexes. However, the additional DM suffers from the inaccurate depth and unknown focus-depth, cannot further improve the performance. We also provide the visualization of predicted filters and final outputs by introducing different combinations of priors, which is shown in Fig. 14. The SM could provide a good saliency guidance, but may also mislead the MPFN to produce inaccurate filters. This limitation is mainly caused by the model producing a SM cannot sense the depth information from the input image. Though introducing DM could partially alleviate this limitation, the predicted filters still suffer from the inaccurate guidance from the SM. Besides, it should be noticeable that introducing DM or SM certainly requires additional computation cost, while our CM is almost free.

Sizes of constrained predictive filters. According to k_{total}, we propose the multi-scale filtering model and the filter kernel size together affect the final effective filtering area, so we select a series of filter kernel sizes for experiments to prove that the parameters we finally selected are optimal. Specifically, the filter kernel sizes corresponding to level 1, level 2 and level 3 are defined as k_1, k_2 and k_3, and we set a series of different values for k_1, k_2 and k_3 shown in

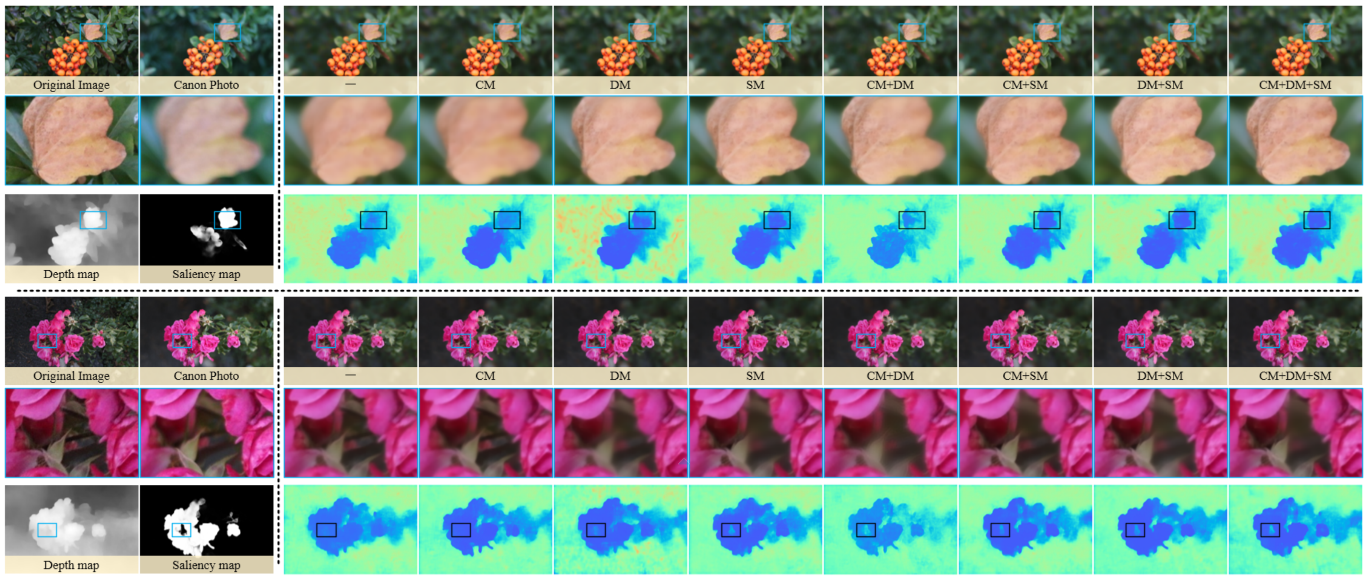


Fig. 14. Visual results obtained with MPFN using different prior knowledge.

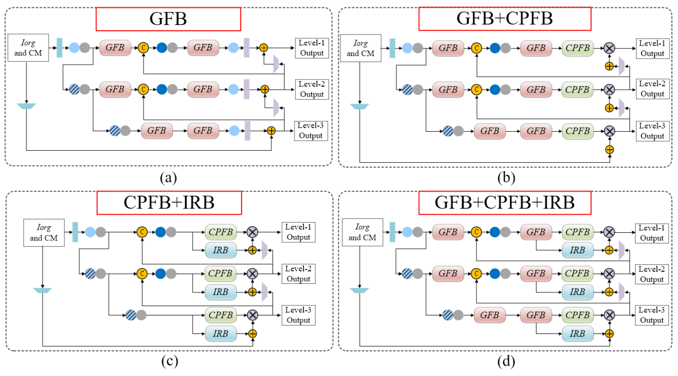


Fig. 15. llustration of the model combining different blocks. (a) M-GFB. (b) M-GFB-CPFB. (c) M-CPFB-IRB. (d) Full MPFN.

TABLE VII
PERFORMANCE COMPARISON OF MPFN MODEL AND OTHER RELATED METHODS ON TEST100 SET.

Method	PSNR	SSIM	LPIPS	MOS	Parameters	FLOPs	GPU Time(s)	CPU Time(s)	GPU-M Time(s)
EDSR[35]	23.72	0.8571	0.2517	1.69	1.5M	1.08T	0.007	12.344	1.490
U-net[42]	23.96	0.8695	0.2290	1.45	31M	0.78T	**0.006**	**6.386**	2.798
Yang[23]	23.96	0.8811	**0.2275**	1.15	5.4M	0.45T	0.028	6.808	3.515
PyNET[19]	23.93	0.8757	0.2527	1.23	47.5M	1.34T	0.223	6.861	1.063
Stacked-DMSHN[12]	23.90	0.8793	0.2320	0.96	10.8M	0.62T	0.019	7.929	**0.943**
MPFN	**24.25**	**0.8861**	0.2297	**0.93**	8.7M	0.75T	0.014	9.738	1.594

TABLE VIII
MOS COMPARISON OF MPFN AND OTHER RELATED METHODS ON TEST200 AND REAL36.

	EDSR [35]	U-net [42]	Yang [23]	PyNET [19]	Stacked-DMSHN[12]	BGGAN [40]	MPFN
Test200	2.10	1.94	1.75	1.75	1.45	1.35	**1.28**
Real36	1.92	1.67	1.34	1.33	1.33	/	**1.24**

TABLE V. It's clear that the larger filter kernel sizes produce more human-eye pleased results. Meanwhile, because the filter generation and the filtering operation works on the original scales, it also consumes more computation resource. To make a balance, we adopt the filter kernel sizes of 9, 11 and 13 to formulate our final model in following evaluations.

Loss Function. To demonstrate the validity of the loss function we adopted, we compare it with several related loss functions, such as perceptual loss based on VGG16 [26], Laplace loss and SSIM loss. All loss functions are applied using the strategy as shown in Eqn. 15 and finally measured by the MAE function. To balance the outputs of these losses and L1 loss, we assigned a hyper-parameter λ. To prove that the L1 Loss is more suitable for our task than the L2 Loss, we trained with the L2 Loss and compared the results using only L1 Loss. As shown in TABLE VI, the structural information provided by the ASL leads to a significant improvement of on all metrics. Though the perceptual loss can achieve the best LPIPS performance, it cannot accurately capture the degree of blurring and gets a comparatively low MOS value. On the other hand, both the ASL and Laplace losses can provide high-frequency information. However because the Laplace filter places a much higher weight on the center than neighbors, it produces similar results to the L1 loss. Generally, the ASL loss we adopted is simple and effective for bokeh rendering.

C. Comparison With the State-of-the-Art

In this section, we compare the proposed method with several state-of-the-art methods on both quantitative and qualitative metrics and average running times. The average running time is measured on Test100. We use a Nvidia 2080Ti GPU, Intel i7-10700K CPU, Snapdragon 865 mobile processor as the platforms to measure the running times on GPU, CPU and mobile GPU (GPU-M) respectively.

Comparison on EBB! dataset. We first compare the results of the several state-of-the-art methods tested on Test100.

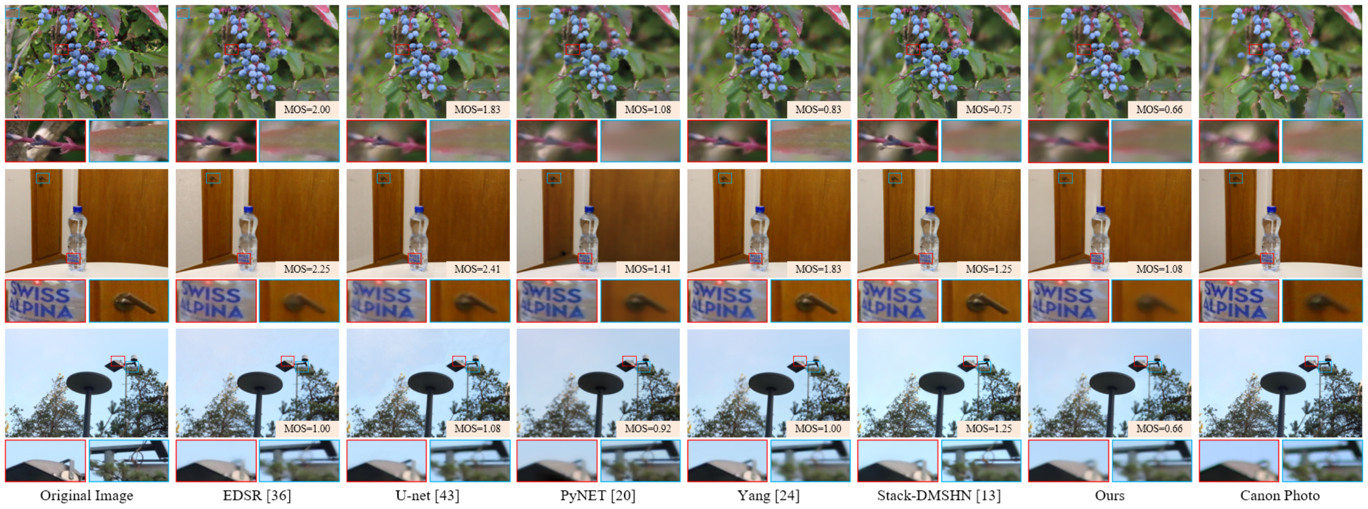


Fig. 16. Visual results obtained with 6 different methods. From left to right, the original wide depth-of-field images, EDSR [35], U-net [42], PyNET [18], Yang [23], Stacked-DMSHN [12], our MPFN and the target Canon photo.

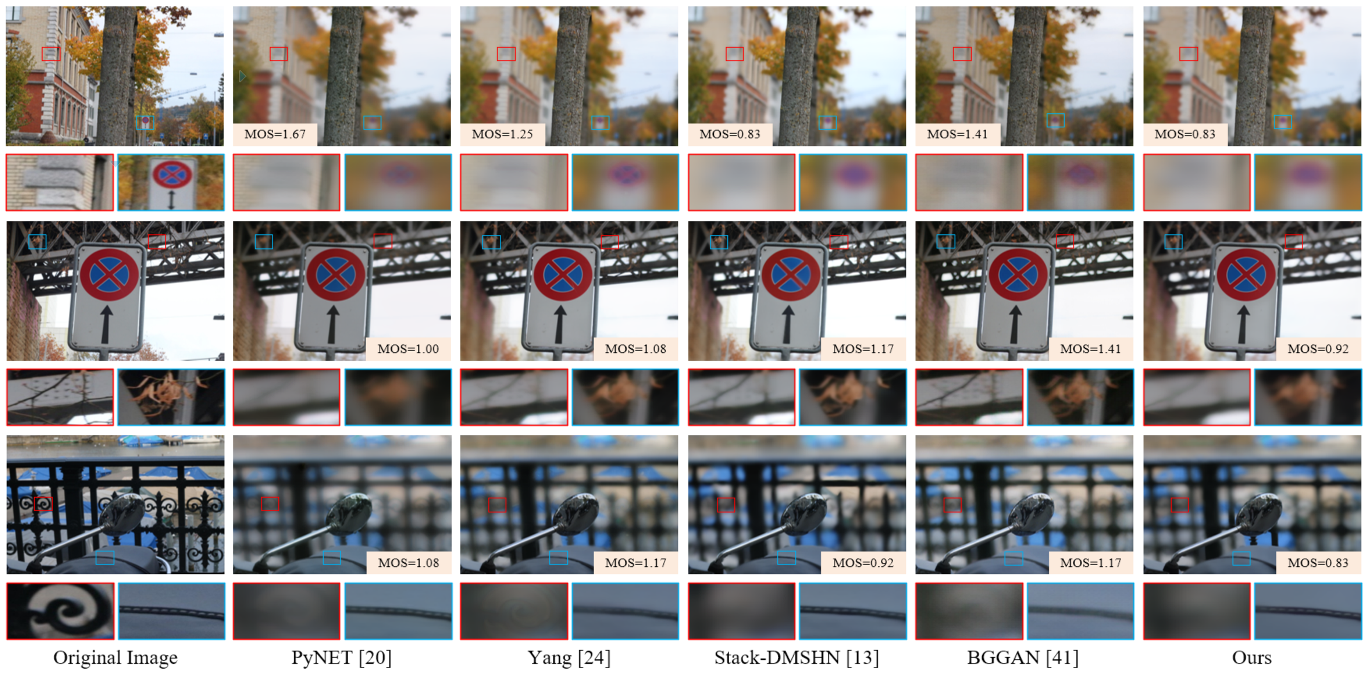


Fig. 17. Visual results obtained with 5 different methods for the bokeh rendering task on Test200. From left to right, the original wide depth-of-field images, PyNET [18], Yang [23], Stacked-DMSHN [12], BGGAN [40] and our MPFN.

Noticing that several recently proposed methods adopt the U-net[42]-like and EDSR [35]-like architectures, we included U-net and EDSR in the comparison. All the models were retrained and tested with the same dataset, and the results are shown in TABLE VII and TABLE VIII. Our proposed model MPFN achieves the best PSNR, SSIM and MOS results, and the third best LPIPS, which is only slightly lower than Yang [23] and U-net. For the most important MOS results, our MPFN clearly surpasses all compared methods on Both Test100 and Test200 datasets. From the user study, our method clearly achieves a better blurring effect in the defocused area and produces a smoother bokeh effects than the compared methods.

Moreover we also conduct a detailed comparison to evaluate performances of compared methods on focused regions and bokeh regions seperately. We use the estimated focused features to segment the generated bokeh image and groundtruth, so we get the corresponding images of the focused regions and the bokeh regions. As shown in TABLE IX, our MPFN achieves the best PSNR in both the focused and bokeh areas, indicating that our method is efficient in reconstructing both the focused and defocused regions. Furthermore, compared to other bokeh specific methods, our method gains more in the bokeh area than in the focused area, which means that the blur effect we generate has less pixel difference from the real bokeh images.

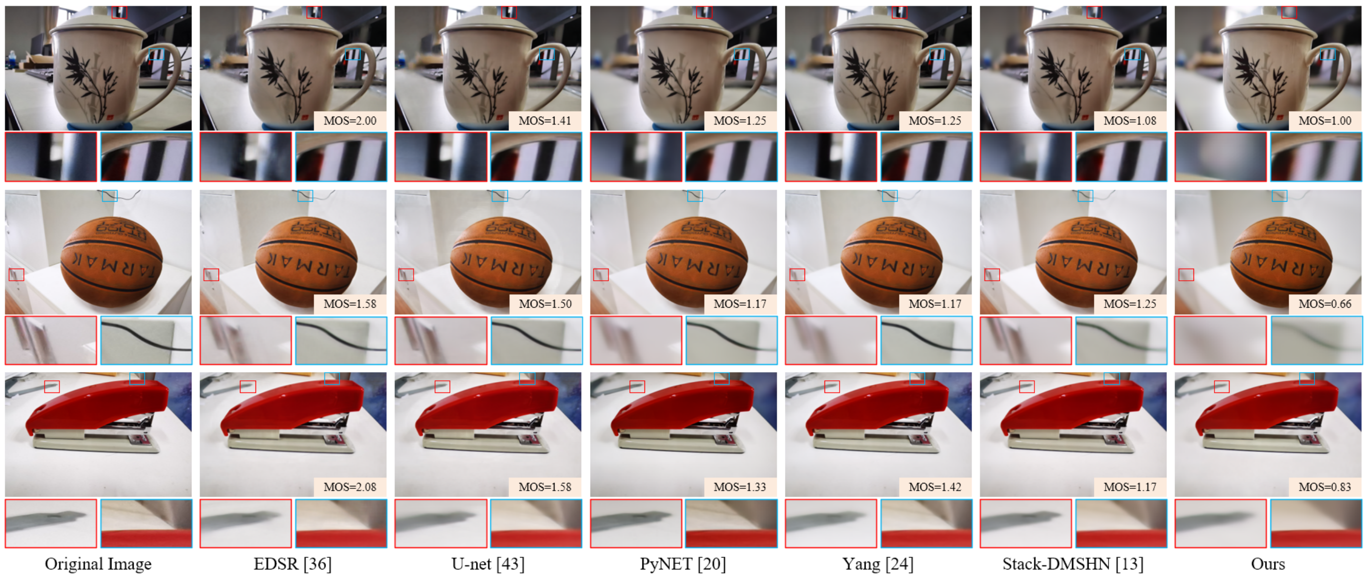


Fig. 18. Visual results in new real data captured with a Huawei Mate30. From left to right, the original images, U-net [42], PyNET [18], Yang [23], and our MPFN.

TABLE IX
PERFORMANCE COMPARISON OF MPFN IN FOCUSED REGIONS AND BOKEH REGIONS ON TEST100 DATASET.

Method	EDSR[35]	U-net[47]	Yang[22]	PyNET[17]	Stacked-DMSHN[12]	MPFN
Focused regions	31.47	31.66	31.75	31.80	31.63	**31.95**
Bokeh regions	24.94	25.20	25.16	25.09	25.14	**25.50**

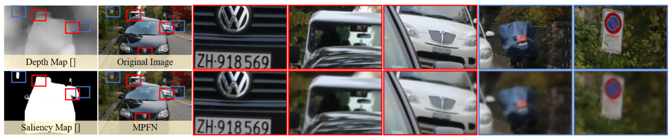


Fig. 19. Some failure results. The red boxes indicate similar saliency regions at different depths, and the blue boxes indicate defocus areas at corresponding depths.

Because bokeh rendering is a visually oriented task, we also visualize some results produced by compared methods on Test100 and Test200 in Figs. 16 and 17, to further highlight the advantages of our MPFN. From the visual comparison, our MPFN produces the best bokeh rendering effect without artifacts and better preserves the details of the focused parts, such as sharp edges and accurate colors. It is worth mentioning that our MPFN is the most efficient one among bokeh rendering methods running on the GPU device, even though our parameters and FLOPs are not optimal. However due to the multiple dense blocks and the unoptimized predictive filtering modules, our MPFN struggles to efficiently run on the CPU device and mobile devices. Note that we process half resolution floating point 16-bit of images on mobile devices. Optimization for these devices is left for future work.

Generalization to new real data. To test the generalization ability of our model, we test our model and the other compared methods on Real36 dataset, which are diverse and different from EBB! dataset and not used in the training stage. As shown in Fig. 18, our MPFN accurately blurs the defocus area while preserving the details of salient objects. From the MOS results shown in TABLE VIII, our model achieves the best MOS score on realistic images, and exhibits good generalization ability.

V. Limitation and Future Work

Although the proposed method can synthesize satisfactory bokeh effects, but sometimes suffers from the depth-dependent scenes. As shown in Fig. 19, when input image contains multiple similar saliency regions at different depths, our MPFN fails to fairly blur the both out-of-focus salient object and background. Therefore, how to effectively utilize both depth and saliency information to produce a fair blurring effects for both salient objects and background is left as one of the future works. On the other hand, we should also pay attention to the computation efficiency that the computation cost of MPFN is only affordable for several flagship processors. Therefore further improving the computation efficiency allowing the algorithm running on most of consumer processor is another future work.

VI. Conclusion

In this paper, we present an effective computational approach for the realistic single image bokeh effect rendering task. To retain the focus area and simulate bokeh rendering in a unified architecture, we propose a multi-scale constrained predictive filtering network using three types of filtering kernels. In order to obtain accurate edge information after filtering, we design a gate fusion block for salient feature detection and fusion of focus and defocused features. Furthermore, we pay attention to the positional information of pixels and propose the coordinate map to be sent the network together with full-focus images for training. Extensive ablation studies validate the importance of the multi-scale strategy, coordinate map, and the components of MPFN. Experiments on the Test100, Test200 and Real36 datasets show that our model outperforms state-of-the-art methods.

VII. Acknowledgment

This work is supported by The National Nature Science Foundation of China (62001146, 61901145), National Natural Science Major Foundation of Research Instrumentation of PR China under Grants 61427808, Zhejiang Province Nature Science Foundation of China (LR17F030006, Q19F010030), 111 Project, No. D17019.

References

[1] Martin Abadi et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *arXiv preprint arXiv:1603.04467* (2016).

[2] Abdullah Abuolaim and Michael S Brown. "Defocus Deblurring Using Dual-Pixel Data". In: *European Conference on Computer Vision*. Springer. 2020, pp. 111–126.

[3] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495.

[4] B De Brabandere et al. "Dynamic Filter Networks". In: *International Conference on Neural Information Processing Systems*. 2016.

[5] Juan Buhler and Dan Wexler. "A phenomenological model for bokeh rendering". In: *ACM SIGGRAPH 2002 conference abstracts and applications*. 2002, pp. 142–142.

[6] Meng Chang et al. "Beyond Camera Motion Blur Removing: How to Handle Outliers in Deblurring". In: *IEEE Transactions on Computational Imaging* 7 (2021), pp. 463–474. DOI: 10.1109/TCI.2021.3076886.

[7] Laurent D'Andrès et al. "Non-parametric blur map regression for depth of field extension". In: *IEEE Transactions on Image Processing* 25.4 (2016), pp. 1660–1673.

[8] Harold Davis. *Practical Artistry: Light & Exposure for Digital Photographers*. " O'Reilly Media, Inc.", 2008.

[9] Mauricio Delbracio and Guillermo Sapiro. "Hand-Held Video Deblurring Via Efficient Fourier Aggregation". In: *IEEE Transactions on Computational Imaging* 1.4 (2015), pp. 270–283. DOI: 10.1109/TCI.2015.2501245.

[10] Weisheng Dong et al. "Deep Spatial–Spectral Representation Learning for Hyperspectral Image Denoising". In: *IEEE Transactions on Computational Imaging* 5.4 (2019), pp. 635–648. DOI: 10.1109/TCI.2019.2911881.

[11] Saikat Dutta. "Depth-aware Blending of Smoothed Images for Bokeh Effect Generation". In: *arXiv preprint arXiv:2005.14214* (2020).

[12] Saikat Dutta et al. "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 (CVPR) Workshops*. June 2021, pp. 2398–2407.

[13] Jawook Gu and Jong Chul Ye. "AdaIN-Based Tunable CycleGAN for Efficient Unsupervised Low-Dose CT Denoising". In: *IEEE Transactions on Computational Imaging* 7 (2021), pp. 73–85. DOI: 10.1109/TCI.2021.3050266.

[14] Praful Hambarde and Subrahmanyam Murala. "S2DNet: Depth Estimation From Single Image and Sparse Samples". In: *IEEE Transactions on Computational Imaging* 6 (2020), pp. 806–817. DOI: 10.1109/TCI.2020.2981761.

[15] Qibin Hou et al. "Deeply supervised salient object detection with short connections". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 3203–3212.

[16] Gao Huang et al. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[17] Yihao Huang et al. "AdvBokeh: Learning to Adversarially Defocus Blur". In: *arXiv preprint arXiv:2111.12971* (2021).

[18] Andrey Ignatov, Jagruti Patel, and Radu Timofte. "Rendering Natural Camera Bokeh Effect with Deep Learning". In: (2020), pp. 0–0.

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

[20] Andrey Ignatov et al. "Ai benchmark: All about deep learning on smartphones in 2019". In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE. 2019, pp. 3617–3635.

[21] Andrey Ignatov et al. "Ai benchmark: Running deep neural networks on android smartphones". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018.

[22] Andrey Ignatov et al. "Aim 2019 challenge on bokeh effect synthesis: Methods and results". In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE. 2019, pp. 3591–3598.

[23] Andrey Ignatov et al. "AIM 2020 challenge on rendering realistic bokeh". In: *European Conference on Computer Vision*. Springer. 2020, pp. 213–228.

[24] Andrey Ignatov et al. "Dslr-quality photos on mobile devices with deep convolutional networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 3277–3285.

[25] Yuna Jeong et al. "Real-Time Dynamic Bokeh Rendering with Efficient Look-Up Table Sampling". In: *IEEE Transactions on Visualization and Computer Graphics* (2020).

[26] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution". In: *European conference on computer vision*. Springer. 2016, pp. 694–711.

[27] Takuhiro Kaneko. "Unsupervised Learning of Depth and Depth-of-Field Effect From Natural Images With

Aperture Rendering Generative Adversarial Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15679–15688.

[28] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[29] Kazuya Kodama, Hiroshi Mo, and Akira Kubota. "Virtual bokeh generation from a single system of lenses". In: *ACM SIGGRAPH 2006 Research posters*. 2006, 77–es.

[30] Himanshu Kumar et al. "Depth map estimation using defocus and motion cues". In: *IEEE Transactions on Circuits and Systems for Video Technology* 29.5 (2018), pp. 1365–1379.

[31] Orest Kupyn et al. "Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 8878–8887.

[32] Junyong Lee et al. "Deep defocus map estimation using domain adaptation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12222–12230.

[33] Sungkil Lee, Gerard Jounghyun Kim, and Seungmoon Choi. "Real-time depth-of-field rendering using point splatting on per-pixel layers". In: *Computer Graphics Forum*. Vol. 27. 7. Wiley Online Library. 2008, pp. 1955–1962.

[34] Zhengqi Li and Noah Snavely. "MegaDepth: Learning Single-View Depth Prediction from Internet Photos". In: *Computer Vision and Pattern Recognition (CVPR)*. 2018.

[35] Bee Lim et al. "Enhanced deep residual networks for single image super-resolution". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2017, pp. 136–144.

[36] Jiang-Jiang Liu et al. "A simple pooling-based design for real-time salient object detection". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 3917–3926.

[37] Xianrui Luo et al. "Bokeh rendering from defocus estimation". In: *European Conference on Computer Vision*. Springer. 2020, pp. 245–261.

[38] Haiyang Mei et al. "Exploring dense context for salient object detection". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2021).

[39] Kuldeep Purohit et al. "Depth-guided dense dynamic filtering network for bokeh effect rendering". In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE. 2019, pp. 3417–3426.

[40] Ming Qian et al. "BGGAN: Bokeh-Glass Generative Adversarial Network for Rendering Realistic Bokeh". In: *arXiv preprint arXiv:2011.02242* (2020).

[41] Guennadi Riguer, Natalya Tatarchuk, and John Isidoro. "Real-time depth of field simulation". In: *ShaderX2: Shader Programming Tips and Tricks with DirectX* 9 (2004), pp. 529–556.

[42] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[43] Luna Sun et al. "AMPNet: Average-and Max-Pool Networks for Salient Object Detection". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2021).

[44] Neal Wadhwa et al. "Synthetic depth-of-field with a single-camera mobile phone". In: *ACM Transactions on Graphics (ToG)* 37.4 (2018), pp. 1–13.

[45] Yingqian Wang et al. "Selective light field refocusing for camera arrays using bokeh rendering and superresolution". In: *IEEE Signal Processing Letters* 26.1 (2018), pp. 204–208.

[46] Z. Wang et al. "Image Quality Assessment: From Error Visibility to Structural Similarity". In: *IEEE Transactions on Image Processing* 13.4 (Apr. 2004), pp. 600–612. DOI: 10.1109/tip.2003.819861.

[47] Ke Xian et al. "Monocular relative depth perception with web stereo data supervision". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 311–320.

[48] Ke Xian et al. "Ranking-based salient object detection and depth prediction for shallow depth-of-field". In: *Sensors* 21.5 (2021), p. 1815.

[49] Richard Zhang et al. "The unreasonable effectiveness of deep features as a perceptual metric". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595.

[50] Hengrun Zhao et al. "CBREN: Convolutional Neural Networks for Constant Bit Rate Video Quality Enhancement". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2021), pp. 1–1. DOI: 10.1109/TCSVT.2021.3123621.

[51] Bolun Zheng et al. "Image demoireing with learnable bandpass filters". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 3636–3645.

[52] Bolun Zheng et al. "Implicit dual-domain convolutional network for robust color image compression artifact reduction". In: *IEEE Transactions on Circuits and Systems for Video Technology* 30.11 (2019), pp. 3982–3994.

[53] Bolun Zheng et al. "Learning Frequency Domain Priors for Image Demoireing". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[54] Wujie Zhou et al. "GFNet: Gate fusion network with Res2Net for detecting salient objects in RGB-D images". In: *IEEE Signal Processing Letters* 27 (2020), pp. 800–804.

[55] Yiren Zhou et al. "Computation and Memory Efficient Image Segmentation". In: *IEEE Transactions on Circuits and Systems for Video Technology* 28.1 (2018), pp. 46–61. DOI: 10.1109/TCSVT.2016.2600261.

[56] Lei Zhu et al. "Aggregating attentional dilated features for salient object detection". In: *IEEE Transactions*

on Circuits and Systems for Video Technology 30.10 (2019), pp. 3358–3371.

Bolun Zheng received the B.S. and Ph.D. degrees in electronic information technology and instrument from Zhejiang University in 2014 and 2019, respectively. He is currently a lecturer with Hangzhou Dianzi University. His research interests are computer vision, pattern recognition, image processing and embedded parallel computing.

Quan Chen received the B.S. degree from the Hangzhou Dianzi University, Zhejiang, China, in 2020. He is currently pursuing the M.E. degree with Hangzhou Dianzi University, Zhejiang. His research interests include intelligent information processing, machine learning and pattern recognition.

Shanxin Yuan is a Senior Research Scientist in Computer Vision for Huawei Technologies, R&D. He received the PhD degree from Imperial College London, the MSc degree from the University of Chinese Academy of Sciences, and the BSc degree from China Agricultural University. His research interests are machine learning and computer vision, and he is currently working on low-level vision. He was co-organizer of NTIRE 2020 workshop along with CVPR 2020 and AIM 2019 workshop along with ICCV 2019. He was the lead organizer of the HIM2017 challenge along with the HANDS17 workshop along with ICCV 2017. He regularly reviews for major computer vision conferences (CVPR, ICCV, ECCV, and NeurIPS) and related journals (TPAMI, IJCV, and TIP).

Xiaofei Zhou received the ph.D. degree from Shanghai University, Shanghai, China, in 2018. He is currently a lecturer with the School of Automation, Hangzhou Dianzi University, Hangzhou, China. His research interests include saliency detection and video segmentation.

Hua Zhang received the B.S. and Ph.D. degrees in biomedical engineering and instrumentation from Zhejiang University in 2003 and 2009, respectively. She is currently an associate professor at the School of Computer Science, Hangzhou Dianzi University. Her research interests are 3D video encoding/decoding, image quality assessment, and embedded system design.

Jiyong Zhang received the B.S. and M.S. degrees in computer science from Tsinghua University in 1999 and 2001, respectively, and the Ph.D. degree in computer sciences from the Swiss Federal Institute of Technology (EPFL), Lausanne, in 2008. He is currently a Distinguished Professor with Hangzhou Dianzi University. His research interests include intelligent information processing, machine learning, data sciences, and recommender systems.

Chenggang Yan received the B.S. degree in computer science from Shandong University in 2008, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences in 2013. He is currently a Professor with Hangzhou Dianzi University. Before that, he was an Assistant Research Fellow of Tsinghua University. His research interests include intelligent information processing, machine learning, image processing, computational biology, and computational photography. He was authored or coauthored over 30 referenced journal and conference papers. As the coauthor, he received the Best Paper Candidate from the International Conference on Multimedia and Expo 2011, and the Best Paper Award from the International Conference on Game Theory for Networks 2014 and the SPIE/COS Photonics Asia Conference 2014.

Gregory Slabaugh is Professor of Computer Vision and AI and Director of the Digital Environment Research Institute (DERI) at Queen Mary University of London. He earned a PhD in Electrical Engineering from Georgia Institute of Technology. Previously, he was Chief Scientist in Computer Vision (EU) for Huawei Technologies R&D, and other prior appointments include City, University of London, Medicsight, and Siemens. His research interests include computational photography, medical image computing, and applications of deep learning.