2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) | 978-1-6654-4899-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/CVPRW53098.2021.00272

Stacked Deep Multi-Scale Hierarchical Network for Fast Bokeh Effect Rendering from a Single Image

Saikat Dutta
IIT Madras
Chennai, India
saikat.dutta779@gmail.com

Sourya Dipta Das
Jadavpur University
Kolkata, India
dipta.juetce@gmail.com

Nisarg A. Shah
IIT Jodhpur
Jodhpur, India
shah.2@iitj.ac.in

Anil Kumar Tiwari
IIT Jodhpur
Jodhpur, India
akt@iitj.ac.in

Abstract

The Bokeh Effect is one of the most desirable effects in photography for rendering artistic and aesthetic photos. Usually, it requires a DSLR camera with different aperture and shutter settings and certain photography skills to generate this effect. In smartphones, computational methods and additional sensors are used to overcome the physical lens and sensor limitations to achieve such effect. Most of the existing methods utilized additional sensor's data or pretrained network for fine depth estimation of the scene and sometimes use portrait segmentation pretrained network module to segment salient objects in the image. Because of these reasons, networks have many parameters, become runtime intensive and unable to run in mid-range devices. In this paper, we used an end-to-end Deep Multi-Scale Hierarchical Network (DMSHN) model for direct Bokeh effect rendering of images captured from the monocular camera. To further improve the perceptual quality of such effect, a stacked model consisting of two DMSHN modules is also proposed. Our model does not rely on any pretrained network module for Monocular Depth Estimation or Saliency Detection, thus significantly reducing the size of model and run time. Stacked DMSHN achieves state-of-the-art results on a large scale EBB! dataset with around 6x less runtime compared to the current state-of-the-art model in processing HD quality images.

1. Introduction

The word "Bokeh" originated from the Japanese word "boke" which means blur. In photography, Bokeh effect refers to the pleasing or aesthetic quality of the blur produced in the out-of-focus parts of an image produced by a camera lens. These images are usually captured with DSLR cameras using a large focal length and a large aperture size. Unlike DSLR cameras, mobile phone cameras have limitations on its size and weight. Thus, mobile phones cannot generate the same quality of bokeh as DSLR can do as the mobile cameras have a small and fixed-size aperture that produces images with the scene mostly in its focus, depending on depth map of the scene. One approach to solve these limitations is to computationally simulate the bokeh effect on the mobile devices with small camera. There are some works where additional depth information obtained from a dual pixel camera [19] or stereo camera (one main camera and one calibrated subcamera) [15, 3, 14] is incorporated in the model to simulate a more realistic bokeh image. However, these systems suffer from a variety of disadvantages like (a) addition of a stereo camera or depth sensor increases the size and the cost of the device and power consumption during usage (b) structured light depth sensors can be affected by poor resolution or high sensitivity to interference with ambient light (c) it performs badly when the target of interest is substantially distant from the camera, it is difficult to estimate good depth map for further regions leading because of the small stereo baseline of stereo camera (d) these approaches can't be used to enhance pictures already taken with monocular cameras as post processing since depth information is often not saved along with the picture.

To address these problems, we propose Deep Multi-Scale Hierarchical Network (DMSHN) for Bokeh effect rendering from the monocular lens without using any specialized hardware. Here, the proposed model synthesizes bokeh effect under the "coarse-to-fine" scheme by exploit-

978-1-6654-4899-4/21/$31.00 ©2021 IEEE
DOI 10.1109/CVPRW53098.2021.00272

ing multi-scale input images at different processing levels. Each lower level acts in the residual manner by contributing its residual image to the higher level thus with intermediate feature aggregation. In this way, low level encoders can provide additional global intermediate features to higher level for improving saliency, and higher level encoders can provide more local intermediate features to improve fidelity of generated bokeh images. Our model does not depend on any Monocular Depth Estimation or Saliency Detection pretrained network module, hence reducing the number of parameters and runtime considerably. We have also explored a stacked version of DMSHN, namely Stacked DMSHN, where two DMSHN modules were connected horizontally to boost the performance. It achieves state-of-the-art results in Bokeh Effect Rendering on monocular images. Though this improvement in accuracy in stacked DMSHN comes with increased runtime and model parameters, still DMSHN and Stacked DMSHN can process HD quality images in 50 fps and 25 fps, respectively, which is significantly faster than other methods in the literature. We have also shown the runtime of our models deployed in mid-range smartphone devices to demonstrate its efficiency. In our experiments, we have used a large-scale *EBB!* dataset [7] containing more than 10,000 images collected in the wild with the DSLR camera.

2. Related Work

Many works in Bokeh Effect Rendering leveraged depth information from images captured by two cameras. Liu *et al.* [14] presented a bokeh simulation method by using depth estimation map through stereo matching. They have also designed a convenient bokeh control interface that uses a little user interaction for identifying the salient object and control the bokeh effect by setting a specific kernel. Busam *et al.* [3] also proposed a stereo vision-based fast and efficient algorithm to perform the refocusing by using high-quality disparity map via efficient stereo depth estimation. Recently, Luo *et al.* [15] proposed a novel deep neural architecture named wavelet synthesis neural network (WSN), to produce high-quality disparity maps on smartphones by using a pair of calibrated stereo images. However, these methods are ineffective in the case of monocular cameras or in post-processing of previously captured images.

In one of the earliest works in Monocular Bokeh Effect Rendering, Shen *et al.* [18] used a Fully Convolutional Network to perform portrait image segmentation and using this segmentation map, they generated depth-of-field image with uniformly blurred background on portrait images. Wadhwa *et al.* [19] proposed a system to generate synthetic shallow depth-of-field images on mobile devices by incorporating a portrait segmentation network and depth map from the camera's dual-pixel auto-focus system. But their system's limitation is that their bokeh rendering method is not photorealistic as actual bokeh photos taken from DSLR cameras because of using an approximated disk blur kernel to blur the background. Xu *et al.* [22] also proposed a similar approach where they had used two different deep neural networks to get depth map and portrait segmentation map from a single image. Then, they improved those initial estimates using another deep neural network and trained a depth and segmentation guided Recursive Neural Network to approximate and accelerate the bokeh rendering.

Lijun *et al.* [13] presented a memory efficient deep neural network architecture with a lens blur module, which synthesized the lens blur and guided upsampling module to generate bokeh images at high resolution with user controllable camera parameters at interactive speed of rendering. Dutta [5] formulated bokeh image as a weighted sum of the input image and its different blurred versions obtained by using different sizes of Gaussian blur kernels. The weights for this purpose were generated using a fine-tuned depth estimation network. Purohit *et al.* [17] designed a model consisting of a densely connected encoder and decoder taking benefits of joint Dynamic Filtering and intensity estimation for the spatially-aware background blurring. Their model utilized pretrained networks for depth estimation and saliency map segmentation to guide the bokeh rendering process. Zheng *et al.* [8] used a multi-scale predictive filter CNN consisting of Gate Fusion Block, Constrained Predictive Filter Block, and Image Reconstruction Block. They trained their model using image patches with concatenated pixel coordinate maps with respect to the full-scale image. Xiong *et al.* [8] used an ensemble of modified U-Net consisting of residual attention mechanism, multiple Atrous Spatial Pyramid Pooling blocks, and Fusion Modules. Yang *et al.* [8] used two stacked bokehNet with additional memory blocks to capture global and local features. The generated feature maps from each model are concatenated and fed to a Selective Kernel Network [11].

Ignatov *et al.* [7] presented a multi-scale end-to-end deep learning architecture, PyNet, for natural bokeh image rendering by utilizing both input image captured with narrow aperture and pre-computed depth map of the same image. In this paper, we propose a fast, lightweight efficient network, Stacked DMSHN for Single-image Bokeh effect rendering. Our model doesn't require additional depth maps which makes it suitable for post-processing photos.

3. Proposed Method

We used Stacked Deep Multi-Scale Hierarchical Network (DMSHN) for Bokeh Effect Rendering. The model diagram of Stacked DMSHN is shown in Fig. 1. Stacked DMSHN consists of two DMSHN base networks which are cascaded one after another. The details of DMSHN architecture are described in the following.

Base Network: We used Deep Multi-Scale Hierarchi-

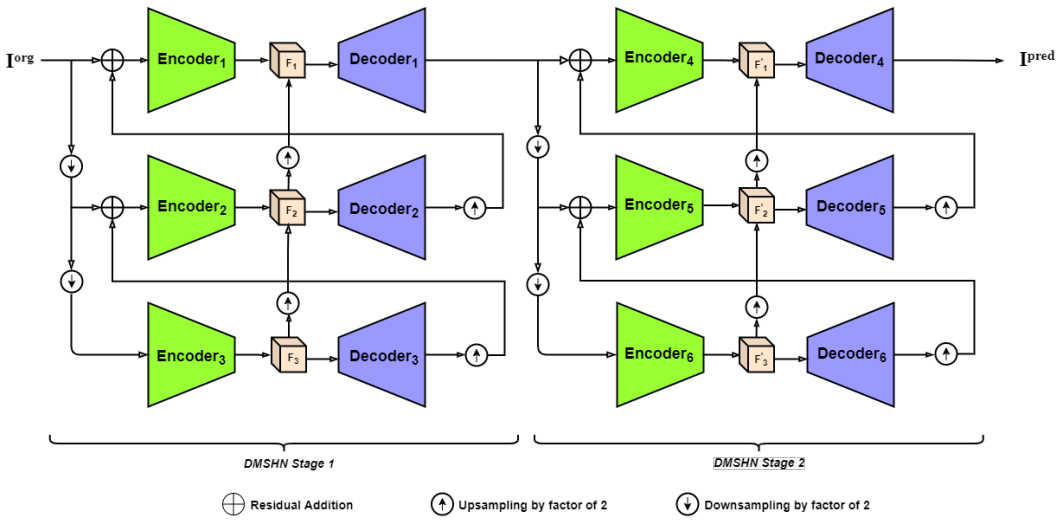

Figure 1. Stacked Deep Multi-Scale Hierarchical Network.

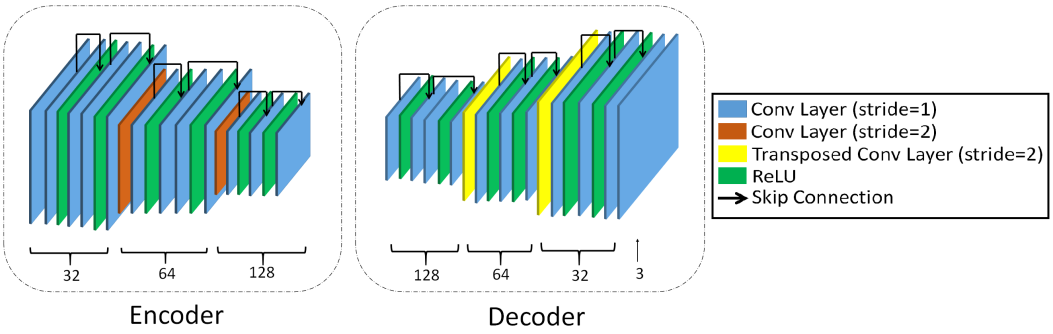

Figure 2. Encoder and Decoder Architecture. Numbers below the curly braces denote number of output channels of the corresponding layers.

cal Network (DMSHN) as the base network. The network works on a 3-level Image pyramid. Let the input images at i^{th} level be denoted as I_i^{org}.

$$I_1^{org} = I^{input} \tag{1}$$

$$I_2^{org} = down(I_1^{org}) \tag{2}$$

$$I_3^{org} = down(I_2^{org}) \tag{3}$$

where $down(.)$ is bilinear downsampling by factor of two.

In each level of the network, we have an encoder and a decoder. Let's denote these modules as $Encoder_i$ and $Decoder_i$ respectively for i^{th} level.

At the bottom-most level encoder, $Encoder_3$, takes bokeh-free image I_3^{org} as input. The features F_3 generated by $Encoder_3$ is then fed into $Decoder_3$. So, the generated feature map F_3 is then fed to $Decoder_3$ to generate residual features res_3.

$$F_3 = Encoder_3(I_3^{org}) \tag{4}$$

$$res_3 = Decoder_3(F_3) \tag{5}$$

res_3 is upsampled by a factor of 2 and then added to I_2^{org} in the next level and fed to $Encoder_2$. The encoded feature map G_2 is then added to upscaled feature F_3 to produce F_2. Reusing the encoded features from the lower level helps the network leverage the global context information learned by the encoder in the previous level. Residual connection between encoders of different levels is useful in correct localization and reconstruction of the foreground. F_2 is then passed to $Decoder_2$ which generates residual features res_2.

$$G_2 = Encoder_2(I_2^{org} + up(res_3)) \tag{6}$$

$$F_2 = G_2 + up(F_3) \tag{7}$$

$$res_2 = Decoder_2(F_2) \tag{8}$$

where $up(.)$ is bilinear upsampling by factor of 2. The upscaled residual feature map res_2 is added to I_1^{org} in the top-most level and passed to $Encoder_1$ to generate encoded feature map G_1. G_1 is added with upscaled F_2 to generate F_1. F_1 is further fed to $Decoder_1$ to generate the final bokeh image I^{pred}.

$$G_1 = Encoder_1(I_1^{org} + up(res_2)) \quad (9)$$

$$F_1 = G_1 + up(F_2) \quad (10)$$

$$I^{pred} = Decoder_1(F_1) \quad (11)$$

For the task of non-homogeneous Image Dehazing [2], DMSHN [4] was coined for ablation studies. In [4], the authors have shown the inferiority of DMSHN with respect to a Patch-hierarchical network. However we have shown in this paper that this type of multi-scale architecture is more suitable for solving problems where capturing the global context and saliency in the image is important, by incorporating it to Bokeh Effect Rendering.

Encoder and Decoder Architecture: The encoder and the decoder consists of 3 level and each level consists of two Residual blocks. Convolutional layers with stride 2 is used to decrease spatial resolution of the feature maps in the encoder and Transposed convolution is used for increasing feature map resolution in the decoder. ReLU is used as activation function and kernel size used in convolutional layers is 3×3 everywhere. Same architecture of encoder and decoder is used at all the levels of our network. The architecture diagram of encoder and decoder is shown in Fig. 2.

Instead of increasing depth of DMSHN model vertically, stacking two or more DMSHN module horizontally can significantly improve the performance [23]. Cascading multiple base networks can help refining the rendered bokeh images. We cascaded two pretrained DMSHN models in this approach and finetune the whole network. Let the first network be denoted as net_1 and the second one as net_2. The final output I^{pred} is given by,

$$I^{pred} = net_2(net_1(I_1^{org})) \quad (12)$$

Our Stacked DMSHN model has a few similarities with Gridnet [6]. However the key differences are: (a) Feature maps at the same level of Gridnet has same channel and spatial dimension, which is not the case for Stacked DMSHN (b) 2nd and 3rd levels of DMSHN blocks are not connected in Stacked DMSHN.

4. Experiments

4.1. System description

We implemented the proposed models in Python and PyTorch [16]. Our models were trained on a machine with Intel Xeon CPU with 16 GB RAM and NVIDIA 1080Ti GPU card with approximately 12 GB GPU Memory.

4.2. Dataset Description

Everything is Better with Bokeh (*EBB!*) Dataset [7] is used in this work. In this dataset, the images were taken from a wide range of locations during the daytime, and in varying lighting and weather conditions. This dataset contains 5094 pairs of Bokeh-free and Bokeh images. The training set consists of 4694 image pairs whereas the validation and test set contains 200 image pairs each. The Bokeh-free images were captured using a narrow aperture (f/16) and the corresponding bokeh images were captured using high aperture (f/1.8). The average image resolution of this dataset is 1024×1536. Since the Validation and test set ground truth images are not available yet, we use *Val294* set [5] was used for evaluation and the rest of the dataset is used for training the models.

4.3. Training and Testing Details

We rescaled the images to 1024×1024 for training. Data augmentation techniques e.g., horizontal and vertical flipping were used to increase the training set size. We use Adam optimizer [10] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for training the network with a batch size of 2. The initial learning rate is set to 10^{-4} and gradually decreased to 10^{-6}.

During inference time, the input images are resized to 1024×1536 and fed into the network. The output from the network is rescaled back to its original dimension using bilinear interpolation.

4.4. Loss functions

The proposed network was trained in two stages. We used a linear combination of L1 loss and SSIM loss [25] in the first stage. L1 loss helps in pixelwise reconstruction of the synthesized bokeh image. SSIM loss is used to improve perceptual quality of the generated image, because it focuses on the similarity of local structure. So, the loss function used in the first stage is given by,

$$\mathcal{L}_{st1} = \mathcal{L}_1 + \alpha.\mathcal{L}_{SSIM} \quad (13)$$

where $\mathcal{L}_1 = ||I^{pred} - I^{gt}||_1$, $\mathcal{L}_{SSIM} = 1 - SSIM(I^{pred}, I^{gt})$ and I^{gt} is the ground truth bokeh image. In the second stage, we use Multi-scale SSIM (MS-SSIM) loss [25] with default scale value of 5, to further fine tune the model. MS-SSIM loss is based on MS-SSIM [21] which considers structural similarity at multiple levels of image pyramid. So, the loss in second stage is given by,

$$\begin{aligned} \mathcal{L}_{st2} &= \mathcal{L}_{MS\text{-}SSIM} \\ &= 1 - MS\text{-}SSIM(I^{pred}, I^{gt}) \end{aligned} \quad (14)$$

In our experiments, α is chosen to be 0.1.

For the training of Stacked DMSHN network, we loaded the weights of pretrained DMSHN model and finetuned the whole network using $\mathcal{L}_{MS\text{-}SSIM}$.

4.5. Results

4.5.1 Evaluation metrics:

We used Peak signal-to-noise ratio (PSNR), Structural Similarity (SSIM) [20] and Learned Perceptual Image Patch

Figure 3. Comparison with other methods. From left: (a) Input Image (b) SKN [8] (c) DBSI [5] (d) PyNet [7] (e) DMSHN (ours) (f) Stacked DMSHN (ours) (g) Ground Truth.

Similarity metrics (LPIPS) [24] for comparative evaluation. For higher similarity with respect to ground truth, higher values of PSNR and SSIM, and lower values of LPIPS scores are desired. Although LPIPS is often used as Perceptual Metric in Image Restoration problems, this is not a reliable metric in case of Bokeh Effect Rendering as discussed in [7]. Thus, we also evaluated our models on Mean Opinion Score (MOS) based on a user study.

User Study: 25 users having good experience in photography were asked to rate images with bokeh effect generated from different methods. The users were presented with 20 sets of images from *Val294* where each set contained bokeh ground truth image and rendered bokeh images. Scoring was done on a scale of 0 to 4 where 0 stands for "almost similar" and 4 is "mostly different" (following [7]). The users were suggested to give a low rating (high quality) if (a) All edges, text present in the salient object region are intact in the generated bokeh image (b) There is no particular artifacts (bright spot, blur circle) present in the generated bokeh image. (c) There are no changes in chromatic feature or brightness of the image. (d) The salient region from the generated bokeh image is the same as the ground truth image.

4.5.2 Comparison with other methods

We compared the performance of our method with state-of-the-art methods, e.g. PyNet [7], Depth-aware Blending of Smoothed Images (DBSI) [5] and Selective Kernel networks (SKN) [11]. As the source code is not available for Depth-guided Dense Dynamic Filtering network (DDDF) [17], we only show comparison with other meth-

ods[1]. Table 1 shows that our Stacked DMSHN model performs better than DMSHN. Both DMSHN and Stacked DMSHN performs better than SKN [8] and DBSI [5], whereas Stacked DMSHN achieves similar perceptual quality to that of PyNet [7]. Qualitative comparison (shown in Fig. 3) reveals that the Stacked DMSHN is at par with PyNet and better than other methods in sharp reconstruction of the foreground.

Method	PSNR	SSIM	LPIPS	MOS
SKN [8]	24.66	0.8521	0.3323	3.71
DBSI [5]	23.45	0.8675	0.2463	1.89
PyNet [7]	24.93	0.8788	0.2219	1.11
DMSHN (ours)	24.65	0.8765	0.2289	1.52
Stacked DMSHN (ours)	24.72	0.8793	0.2271	1.17

Table 1. Quantitative Comparison with other methods on Val294 set. The scores in Red and Blue represent best and second best scores respectively.

4.6. Ablation study

Importance of residual connections between encoded features: To show the significance of skip connections between encoded features at different levels, we train one variant of DMSHN where such residual connections are removed. It can be observed from Fig. 4 that residual connections between the encoded features is crucial in correctly detecting the foreground and increasing the overall quality of the rendered bokeh image. Qualitative comparison between the two variants in Table 2 shows significant improvement in performance when the residual connections are used.

Method	PSNR	SSIM	LPIPS	MOS
DMSHN (w/o res.)	22.73	0.8150	0.2953	3.34
DMSHN (with res.)	**24.65**	**0.8765**	**0.2289**	**1.52**

Table 2. Importance of Residual Connections between encoded features.

Effect of the second stage of training: The network is first trained with a combination of $\mathcal{L}_1$ and $\mathcal{L}_{SSIM}$ and then finetuned with $\mathcal{L}_{MS\text{-}SSIM}$. We compare the results of our network with Stage-2 training and without Stage-2 training. We have also experimented with a combination of $\mathcal{L}_1$ and $\mathcal{L}_{MS\text{-}SSIM}$ in second stage training. Table 3 shows that stage-2 training with $\mathcal{L}_1$ and $\mathcal{L}_{MS\text{-}SSIM}$ improves SSIM, LPIPS and MOS metrics over no stage-2 training, whereas using $\mathcal{L}_{MS\text{-}SSIM}$ alone in stage-2 training improves all the four metrics. From the first and second row of Fig. 5, it can be inferred that stage-2 training removes artifacts in the background, and the third and fourth row shows that $\mathcal{L}_{MS\text{-}SSIM}$ helps in the better reconstruction of the foreground.

[1]For qualitative comparison on *EBB!* Test data, please refer to Supplementary material.

Method with Specification	PSNR	SSIM	LPIPS	MOS
DMSHN (without stage-2)	24.41	**0.8765**	0.2322	1.71
DMSHN (with stage-2) $\mathcal{L}_{st2} = \mathcal{L}_1 + 0.1 * \mathcal{L}_{MS\text{-}SSIM}$	24.34	0.8748	0.2299	1.67
DMSHN (with stage-2) $\mathcal{L}_{st2} = \mathcal{L}_{MS\text{-}SSIM}$	**24.65**	**0.8765**	**0.2289**	**1.52**

Table 3. Effect of different Loss functions in Stage-2 training.

DMSHN vs Stacked DMSHN: By stacking two DMSHN networks helps recovering important details in the foreground image in the rendered bokeh image as shown in Fig. 6. Table 1 shows the quantitative improvement of Stacked DMSHN over DMSHN.

Inclusion of depth maps: In order to see if depth maps are useful in our networks, normalized depth maps were computed from the input images using a state-of-the-art Monocular Depth estimation network MegaDepth [12]. The estimated depth map was resized and concatenated to encoder input at each scale.

Table 4 shows that inclusion of depth maps improves the performance of DMSHN. Although LPIPS improves by a small margin in case of Stacked DMSHN after incorporating depth maps, PSNR, SSIM and MOS scores do not improve. It indicates that inaccuracies in depth map estimation doesn't help Stacked DMSHN further in Bokeh rendering. Qualitative comparison in Fig. 7 shows Stacked DMSHN without depth map produces best perceptual results.

Method	PSNR	SSIM	LPIPS	MOS
DMSHN	24.65	0.8765	0.2289	1.52
DMSHN (with depth)	24.68	0.8780	0.2264	1.39
Stacked DMSHN	**24.72**	**0.8793**	0.2271	**1.17**
Stacked DMSHN (with depth)	24.67	0.8780	**0.2263**	1.21

Table 4. Effect of Inclusion of Depth maps in our network.

4.7. Efficiency and Deployment in Mobile Devices:

The proposed models, DMSHN and Stacked DMSHN are lightweight as DMSHN has 5.42 million, whereas Stacked DMSHN has 10.84 million trainable parameters. DMSHN and Stacked DMSHN models take 0.02 and 0.04 seconds to process an HD image, respectively on our system. Table 5 shows parameter and runtime comparison with other existing models. DMSHN is faster than all other models. Our final model, Stacked DMSHN is 0.23 seconds faster than the current state-of-the-art PyNet [7]. It is also important to note that PyNet takes precomputed depth maps as input, generating which takes additional time if it is not readily available, whereas Stacked DMSHN can process the input image directly.

Our models are also deployable in mobile devices. We converted our PyTorch models to Tensorflow Lite (TFLite)

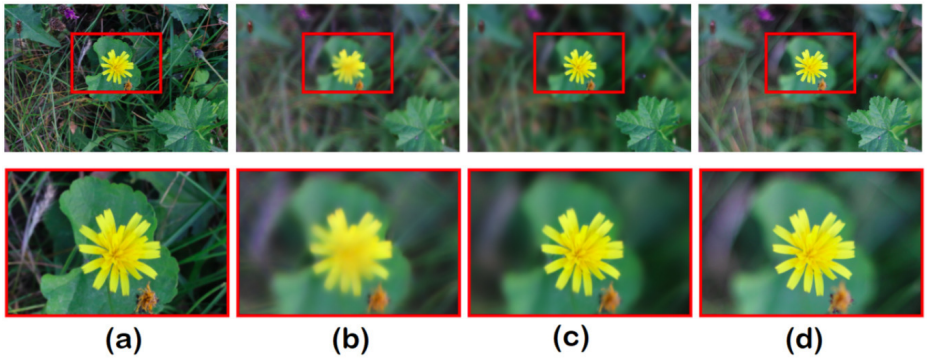

Figure 4. Effect of residual connection between encoders of different levels. From left: (a) Input Image (b) DMSHN (w/o res.) (c) DMSHN (with res.) (d) Ground Truth.

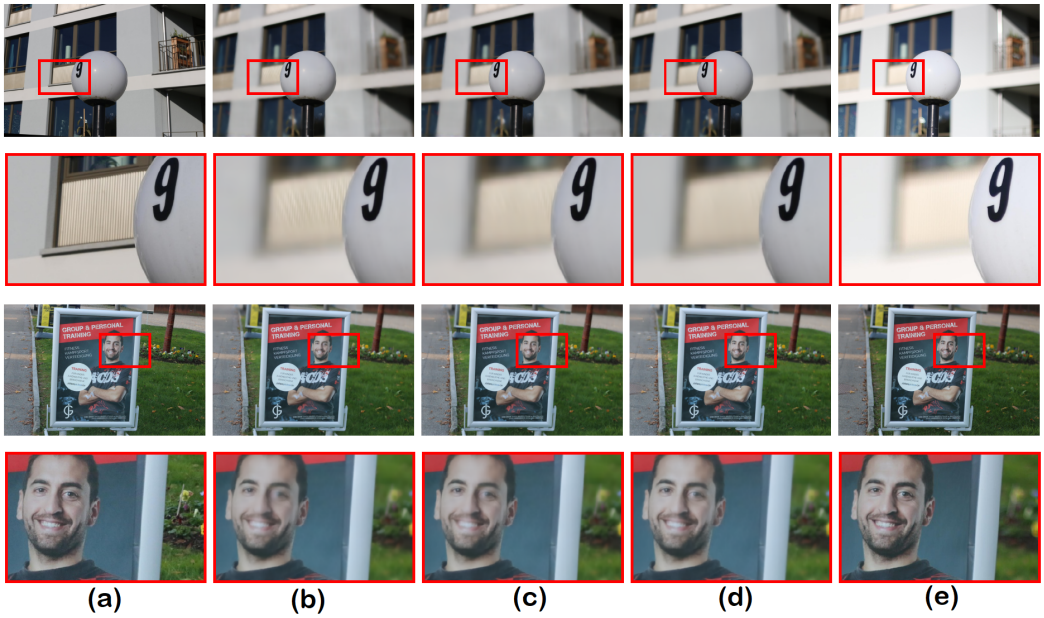

Figure 5. Effect of stage-2 training. From left: (a) Input Image (b) DMSHN (w/o stage-2) (c) DMSHN ($\mathcal{L}_1 + 0.1.\mathcal{L}_{MS\text{-}SSIM}$) (d) DMSHN ($\mathcal{L}_{MS\text{-}SSIM}$) (e) Ground Truth.

Method	Parameters (M)	Runtime (s)
SKN [8]	5.37	0.055
DDDF [17]	N/A	2.5
DBSI [5]	5.36	0.048
PyNet [7]	47.5	0.27
DMSHN (ours)	5.42	0.020
Stacked DMSHN (ours)	10.84	0.040

Table 5. Parameter and Runtime comparison with state-of-the-art models. Runtime for DDDF is reported from [17] and rest of the runtimes were measured on our system. Red and Blue represent best and second best values respectively.

[1] models and ran them on AI Benchmark Android application [9]. Here, we have selected three mainstream mid-range mobile chipsets from three different smartphone manufacturers. The configurations of these chipsets are as follows.

Config-1: Qualcomm Snapdragon 660 AIE processor, Adreno 512 GPU and 4GB RAM.

Config-2: Exynos 9611 processor, Mali-G72 MP3 GPU and 4GB RAM.

Config-3: Qualcomm Snapdragon 855+ processor, Adreno 640 GPU and 8GB RAM.

The corresponding runtimes for processing images of half resolution of HD images on the devices can be found in Table 6. In comparison with other approaches, PyNet have the closest perceptual quality of our stacked DMSHN model, but it failed with an Out of Memory (OOM) error in these mid-range smartphones because of high memory consumption and instance normalization layers present in PyNet which are still not supported adequately by the TensorFlow Lite. However, our model is devoid of these problems.

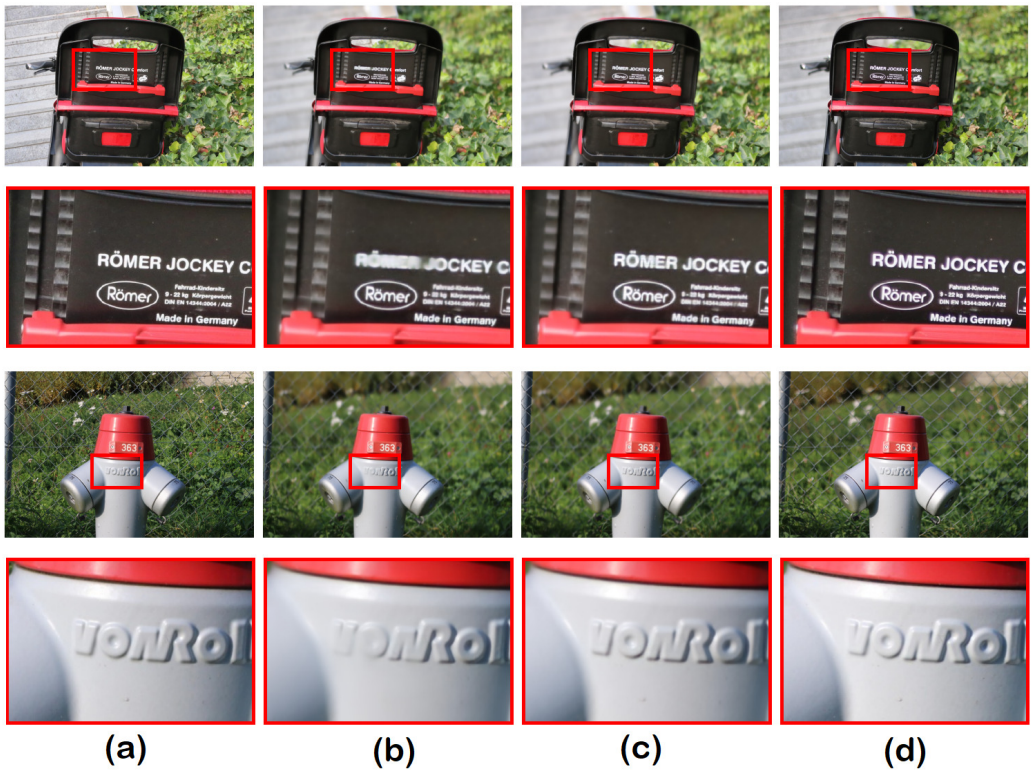

Figure 6. Effect of Stacking. From left: (a) Input Image (b) DMSHN (d) Stacked DMSHN (e) Ground Truth.

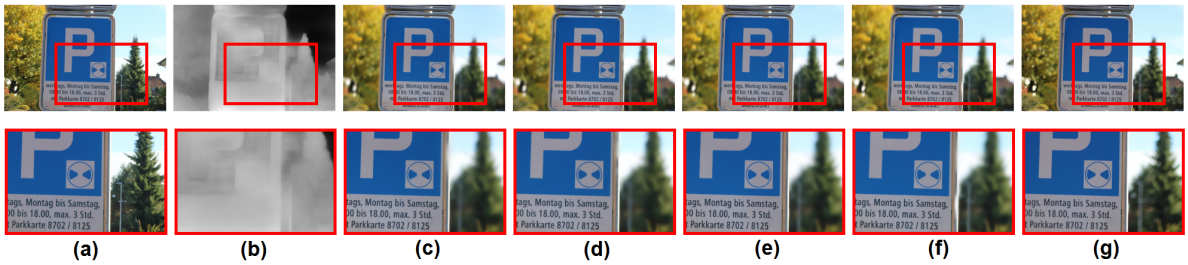

Figure 7. Effect of inclusion of depth maps. From left: (a) Input Image (b) Estimated Depth Map (c) DMSHN (d) DMSHN (with depth) (e) Stacked DMSHN (f) Stacked DMSHN (with depth) (g) Ground Truth

Method	Config-1	Config-2	Config-3
DMSHN	4.80	2.37	0.75
Stacked DMSHN	15.31	12.50	5.27

Table 6. Runtimes (in seconds) of our models on different smartphone configurations.

5. Conclusion

In this paper, we devised two end-to-end deep multi-scale networks, namely DMSHN and Stacked DMSHN for realistic bokeh effect rendering. Our models do not depend on any precomputed depth estimation maps or saliency maps and also do not require any other additional hardware (e.g. depth sensor or stereo camera) other than a monocular camera to aid the bokeh effect rendering. We see that Stacked DMSHN performs better than DMSHN both qualitatively and quantitatively. The Stacked DMSHN yields results of similar perceptual quality as PyNet [7] and performs better than other approaches in the literature. Along with that, our proposed methods are lightweight, efficient, runnable in real time and also much faster than other competing approaches with good perceptual quality. We also showed that our models are deployable in mid-range smartphones too and take significantly less time where our closest competitor, PyNet [7] is only able to run in high-end smartphones. In future, incorporation of dense connections in encoder and decoders and spatial attention can be explored to further improve the perceptual quality of rendered bokeh images.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016. 7

[2] Codruta O Ancuti, Cosmin Ancuti, Florin-Alexandru Vasluianu, and Radu Timofte. Ntire 2020 challenge on nonhomogeneous dehazing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 490–491, 2020. 4

[3] Benjamin Busam, Matthieu Hog, Steven McDonagh, and Gregory Slabaugh. Sterefo: Efficient image refocusing with stereo vision. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019. 1, 2

[4] Sourya Dipta Das and Saikat Dutta. Fast deep multi-patch hierarchical network for nonhomogeneous image dehazing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 482–483, 2020. 4

[5] Saikat Dutta. Depth-aware blending of smoothed images for bokeh effect generation. *Journal of Visual Communication and Image Representation*, page 103089, 2021. 2, 4, 5, 6, 7

[6] Damien Fourure, Rémi Emonet, Elisa Fromont, Damien Muselet, Alain Tremeau, and Christian Wolf. Residual conv-deconv grid network for semantic segmentation. In *BMVC 2017*, 2017. 4

[7] 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*, pages 418–419, 2020. 2, 4, 5, 6, 7, 8

[8] Andrey Ignatov, Jagruti Patel, Radu Timofte, Bolun Zheng, Xin Ye, Li Huang, Xiang Tian, Saikat Dutta, Kuldeep Purohit, Praveen Kandula, et al. Aim 2019 challenge on bokeh effect synthesis: Methods and results. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3591–3598. IEEE, 2019. 2, 5, 6, 7

[9] Andrey Ignatov, Radu Timofte, William Chou, Ke Wang, Max Wu, Tim Hartley, and Luc Van Gool. Ai benchmark: Running deep neural networks on android smartphones. In *Proceedings of the European conference on computer vision (ECCV)*, pages 0–0, 2018. 7

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 4

[11] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 510–519, 2019. 2, 5

[12] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2041–2050, 2018. 6

[13] W Lijun, S Xiaohui, Z Jianming, W Oliver, H Chih-Yao, K Sarah, and L Huchuan. Deeplens: Shallow depth of field from a single image. In *ACM Trans. Graph.(Proc. SIGGRAPH Asia)*, volume 37, pages 6–1, 2018. 2

[14] Dongwei Liu, Radu Nicolescu, and Reinhard Klette. Bokeh effects based on stereo vision. In *International Conference on Computer Analysis of Images and Patterns*, pages 198–210. Springer, 2015. 1, 2

[15] Chenchi Luo, Yingmao Li, Kaimo Lin, George Chen, Seok-Jun Lee, Jihwan Choi, Youngjun Francis Yoo, and Michael O Polley. Wavelet synthesis net for disparity estimation to synthesize dslr calibre bokeh effect on smartphones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2407–2415, 2020. 1, 2

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 4

[17] Kuldeep Purohit, Maitreya Suin, Praveen Kandula, and Rajagopalan Ambasamudram. Depth-guided dense dynamic filtering network for bokeh effect rendering. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3417–3426. IEEE, 2019. 2, 5, 7

[18] Xiaoyong Shen, Aaron Hertzmann, Jiaya Jia, Sylvain Paris, Brian Price, Eli Shechtman, and Ian Sachs. Automatic portrait segmentation for image stylization. In *Computer Graphics Forum*, volume 35, pages 93–102. Wiley Online Library, 2016. 2

[19] Neal Wadhwa, Rahul Garg, David E Jacobs, Bryan E Feldman, Nori Kanazawa, Robert Carroll, Yair Movshovitz-Attias, Jonathan T Barron, Yael Pritch, and Marc Levoy. Synthetic depth-of-field with a single-camera mobile phone. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018. 1, 2

[20] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 4

[21] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003. 4

[22] Xiangyu Xu, Deqing Sun, Sifei Liu, Wenqi Ren, Yu-Jin Zhang, Ming-Hsuan Yang, and Jian Sun. Rendering portraitures from monocular camera and beyond. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 35–50, 2018. 2

[23] Hongguang Zhang, Yuchao Dai, Hongdong Li, and Piotr Koniusz. Deep stacked hierarchical multi-patch network for

image deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5978–5986, 2019. 4

[24] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5

[25] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging*, 3(1):47–57, 2016. 4