Deep Defocus Map Estimation using Domain Adaptation

Junyong Lee[1] POSTECH
Sungkil Lee[2] Sungkyunkwan University
Sunghyun Cho[3] DGIST
Seungyong Lee[1] POSTECH

[1]{junyonglee, leesy}@postech.ac.kr [2]sungkil@skku.edu [3]scho@dgist.ac.kr

Abstract

In this paper, we propose the first end-to-end convolutional neural network (CNN) architecture, Defocus Map Estimation Network (DMENet)*, for spatially varying defocus map estimation. To train the network, we produce a novel depth-of-field (DOF) dataset,* SYNDOF*, where each image is synthetically blurred with a ground-truth depth map. Due to the synthetic nature of SYNDOF, the feature characteristics of images in SYNDOF can differ from those of real defocused photos. To address this gap, we use* domain adaptation *that transfers the features of real defocused photos into those of synthetically blurred ones. Our* DMENet *consists of four subnetworks: blur estimation, domain adaptation, content preservation, and sharpness calibration networks. The subnetworks are connected to each other and jointly trained with their corresponding supervisions in an end-to-end manner. Our method is evaluated on publicly available blur detection and blur estimation datasets and the results show the state-of-the-art performance.*

1. Introduction

A *defocus map* contains the amount of defocus blur or the size of circle of confusion (COC) per pixel for a defocus-blurred (in short, defocused) image. Estimation of a defocus map from a defocused image can greatly facilitate high-level visual information processing, including saliency detection [12], depth estimation [40], foreground/background separation [22], and deblurring [39]. A typical approach for defocus map estimation first detects edges from a blurred image, then measures the amounts of blur around the edges, and finally interpolates the estimated blur amounts at edges to determine the blur amounts in homogeneous regions.

The previous edge-driven approach has a few limitations. First, the edges in a blurred image are often ambiguous, leading to inaccurate detection. Second, blur estimation for edges are inherently prone to errors, as a pixel at object boundary with depth discontinuity contains a mixture of different COCs in a defocused image [18]. Third, this instability of blur estimation at edges would result in less reliable prediction in homogeneous regions. That is, the blur amounts estimated at different parts of an object boundary could be incoherent and then their interpolation toward the homogeneous object interior would produce only smooth but less accurate blur estimation. For example, the estimated blur amounts of an object with a single depth may not be constant, because the blur amounts separately measured at the opposite edges could not be the same when the edges have different depth discontinuities with nearby objects.

In this paper, we present *DMENet* (Defocus Map Estimation Network), the first end-to-end CNN framework, which directly estimates a defocus map given a defocused image. Our work is distinguished from the previous ones for its clear definition on which COC we try to estimate among the mixture of COCs, where we infer the COC size of a pixel using the depth value in the corresponding pinhole image. The network trained with our COC definition leads to more robust estimation of blur amounts, especially at object boundaries. The network also better handles homogeneous regions by enlarging its receptive field so that object edges and interior information are used together to resolve ambiguity. As a result, our network significantly improves the blur estimation accuracy in the presence of mixtures of COCs.

To enable such network learning, a high-quality dataset is crucial. However, currently available datasets [29, 4] are not enough, as they are either for blur detection [29], instead of blur estimation, or of a small size [4]. To this end, we generate a defocus-blur dataset, which we call "SYNDOF" dataset. It would be almost impossible, even manually, to generate ground-truth defocus maps for defocused photos. So we use pinhole image datasets, where each image is accompanied by a depth map, to synthesize defocused images with corresponding ground-truth defocus maps.

One limitation of our dataset is that defocus blurs are synthetic, and there could be *domain difference* [9] between the characteristics of real and synthetic defocused images. To resolve this, we design our network to include domain adaptation, which is capable of adapting the features of real defocused images to those of synthetic ones, so that the network can estimate the blur amounts of real images with the

training of defocus blur estimation using synthetic images.

To summarize, our contributions include:

- the first end-to-end CNN architecture that directly estimates accurate defocus maps without edge detection;
- *SYNDOF* defocus-blur dataset that contains synthetic defocused images with ground-truth defocus maps;
- domain adaptation that enables learning through a synthetic dataset for real defocused images.

2. Related Work

Defocus map estimation For defocus map estimation, most of previous works first estimate blur amounts around explicitly detected edges and then propagate them to the surrounding homogeneous regions. Zhuo *et al*. [40] and Karaali *et al*. [13] use image gradients as local blur cues, and calculate the ratio of the blur cues between edges of the original and re-blurred images. Tang *et al*. [32] estimate a sparse blur map with spectrum contrast near image edges. Shi *et al*. [29] utilize frequency-domain features, learned features, and image gradients to estimate blur amounts. Shi *et al*. [30] adopt a sparse representation to detect just noticeable blurs, which cannot handle large blurs. Xu *et al*. [37] use the rank of a local patch as a cue for blur amount. Park *et al*. [24] build feature vectors consisting of hand-crafted features as well as deep-features taken from a pre-trained blur classification network, and then feed the feature vectors to another network to regress the blur amounts on edges. All these methods commonly rely on features defined only around image edges, and so blur amounts interpolated from the edges for homogeneous regions could be less accurate.

Recently, machine learning techniques have been utilized to densely estimate defocus maps. Andrès *et al*. [4] create a dataset where a ground-truth defocus map is labeled with the radius of point-spread-function (PSF) at each pixel which minimizes the error on a defocused image. They train regression tree fields (RTF) to estimate blur amount of each pixel. However, the method cannot be easily generalized due to insufficient training images and is not robust at pixels around depth boundaries where ground-truth blur amounts cannot be accurately measured. Zhang *et al*. [38] create a dataset by manually labeling each pixel of a defocused image into four levels of blur: high, medium, low, and no blur, for training a CNN classification. Their method shows the state-of-the-art performance on CUHK blur detection dataset [29], but it cannot estimate the exact blur amount, which is essential for applications, such as deblurring and depth estimation.

Domain adaptation Domain adaptation [5] was developed to address the generalization ability of a learning based approach to other domains that it has not been trained for. Ganin *et al*. [5] propose an adversarial learning framework for domain adaptation. Given a source domain with labeled data and a target domain with unlabeled data, their framework trains a label classifier for source domain data as well as a domain classifier to classify different domains. They showed that a classifier trained using their framework generalizes well to a target domain.

Several approaches have been developed since then. Tzeng *et al*. [33] use an adversarial discriminative loss function, and Long *et al*. [21] propose a residual domain classifier. Hoffman *et al*. [10] extend the domain adaptation framework for semantic segmentation. Chen *et al*. [2] propose class-wise domain adaptation for semantic segmentation of road scenes. Hoffman *et al*. [9] present cycle consistent adversarial domain adaptation for better adaptation performance. Bousmalis *et al*. [1] propose to learn a transformation in the pixel-space that transforms a source domain image to appear as if drawn from a target domain.

We use domain adaptation to address the gap between our synthetically generated training images and real ones. While previous studies used domain adaptation mostly for binary or multi-label classification tasks such as semantic segmentation, we adopt it for image-to-image regression.

3. The *SYNDOF* Dataset

3.1. Data Collection

We first collected both synthetic and real images with their associated depth maps; we did not use 3D scene models to avoid time-consuming high quality rendering. Our images are from MPI Sintel Flow (MPI) [35], SYNTHIA [27], and Middlebury Stereo 2014 (Middlebury) [28] datasets. MPI dataset is a collection of game scene renderings, SYNTHIA dataset contains synthetic road views, and Middlebury dataset consists of real indoor scene images with accurate depth measurements.

MPI and SYNTHIA datasets include sequences of similar scenes, and thus we kept only dissimilar images in terms of peak-signal-to-noise-ratio (PSNR) and structural similarity index (SSIM), ending up with 2,006 distinct sample images in total. Then, we repeated the process of randomly selecting an image from the sample set to generate a defocused image with random sampling of camera parameters and the focal distance. The total number of defocused images we generated is 8,231. Table 1 shows the details.

3.2. Thin Lens Model

Given the color-depth pairs, we generated defocused images using the thin-lens model [25], which is a standard for defocus blur in computer graphics (Fig. 1). Let the focal length be F (mm), the object-space focal distance S_1 (mm), and the f-number N. The image-space focal distance is $f_1 = \frac{FS_1}{S_1 - F}$, and the aperture diameter is $D = \frac{F}{N}$. Then, the image-space COC diameter $c(x)$ of a 3D point located at the

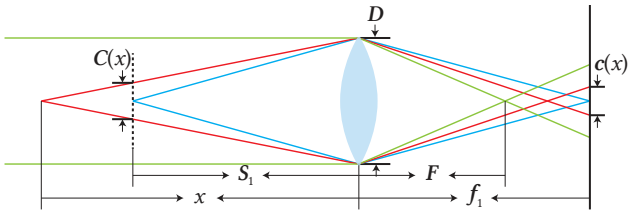


Figure 1: The thin-lens model.

object distance x is defined as:

$$c(x) = \alpha \frac{|x - S_1|}{x}, \text{ where } \alpha = \frac{f_1}{S_1} D. \quad (1)$$

3.3. Defocused Image Generation

To apply defocus blur to an image, we first extract the minimum and maximum depth bounds, x_{near} and x_{far}, from the depth map, respectively. Then, we randomly sample S_1 from the range of $[x_{near}, x_{far}]$. When computing $c(x)$ using Eq. (1), we only need α that abstracts physical parameters. In practice, x is not near zero (implying very close to the lens), having a certain limit. To facilitate meaningful yet random generation of capture conditions, we limit the COC size up to c_{max}. Thereby, the upper bound of α, denoted by α_{up}, is:

$$\alpha_{up} = c_{max} \cdot \min\left(\frac{x_{far}}{|x_{far} - S_1|}, \frac{x_{near}}{|x_{near} - S_1|}\right). \quad (2)$$

Now α is randomly sampled within $[0, \alpha_{up}]$. We then apply Gaussian blur to the image with kernel standard deviation σ, where we empirically define $\sigma(x) = \frac{c(x)}{4}$.

To blur an image based on the computed COC sizes, we first decompose the image into discrete layers according to per-pixel depth values, where the maximum number of layers is limited to 350. Then, we apply Gaussian blur to each layer with $\sigma(x)$, blurring both image and mask of the layer. Finally, we alpha-blend blurred layer images in the back-to-front order using the blurred masks as alpha values. In addition to defocused images, we generate labels (i.e., defocus maps), which trivially record $\sigma(x)$ as the amounts of per-pixel blur. This layer-driven defocus blur is similar to the algorithm of [15], but we bypass the matting step as we do not put different depths into the same layer.

Our SYNDOF dataset enables a network to accurately estimate a defocus map due to the following properties. First, our defocus map is densely (per-pixel and not binary) labeled. The dense labels respect the scene structure, including object boundaries and depth discontinuities, and resolve ambiguities in homogeneous regions. Second, object positions in the original sharp image are used when pixels are labeled with blur amounts in a defocused image. Then, even if the network encounters a mixture of COCs (called the partial occlusion [20]), a blurry pixel is supervised to have the COC size that it had in the sharp image. Note that the other COCs in the mixture are irrelevant at the pixel, as they come from nearby foreground or hidden surfaces (not revealed in the sharp image) [19]. This clarification of which COCs are estimated in a defocus map is a drastic improvement over the previous studies.

Datasets	# samples	# outputs	Type
MPI	1,064	4,346	synthetic
SYNTHIA	896	3,680	synthetic
Middlebury	46	205	real
Total	2,006	8,231	

Table 1: Collection summary of our *SYNDOF* dataset.

4. Defocus Map Estimation

4.1. Overview

Network design Our *DMENet* has a novel architecture for estimating a defocus map from a defocused image (Fig. 2). The network consists of four subnetworks: blur estimation (***B***), domain adaptation (***D***), content preservation (***C***), and sharpness calibration networks (***S***).

The *blur estimation* network ***B*** is the main component of our *DMENet* and supervised with ground-truth synthetic defocus maps from the *SYNDOF* dataset to predict blur amounts given an image. To enable network ***B*** to measure the blur amounts on real defocused images, we attach the *domain adaptation* network ***D*** to it, which minimizes domain differences between synthetic and real features. The *content preservation* network ***C*** supplements network ***B*** to avoid a blurry output. The *sharpness calibration* network ***S*** allows real domain features to induce correct sharpness in a defocus map by informing network ***B*** whether the given real domain feature corresponds to a sharp or blurred pixel. The details of our network structure are in the supplementary material.

Training Our ultimate goal is to train the blur estimation network ***B*** to estimate blur amounts of real images. To achieve this, we jointly train networks ***B***, ***D***, and ***S*** parametrized by θ_B, θ_D, and θ_S, respectively, with three different training sets. Note that network ***C*** is fixed during our training. $\mathcal{D}_S = \{\ldots, \langle I_S^n, y^n \rangle, \ldots\}$ is a training set of synthetic defocused images with ground truth defocus maps, where I_S^n and y^n are the n-th image and the corresponding defocus map, respectively. $\mathcal{D}_R = \{\ldots, I_R^n, \ldots\}$ is a training set of real defocused images with no labels. Lastly, $\mathcal{D}_B = \{\ldots, \langle I_B^n, b^n \rangle, \ldots\}$ is a training set of real defocused images I_B^n with ground truth binary blur maps b^n, where b^n is labeled as sharp or blurred at each pixel.

Given the training datasets, we alternatingly train θ_B and θ_S with a loss L_g, and θ_D with a loss L_d, following the com-

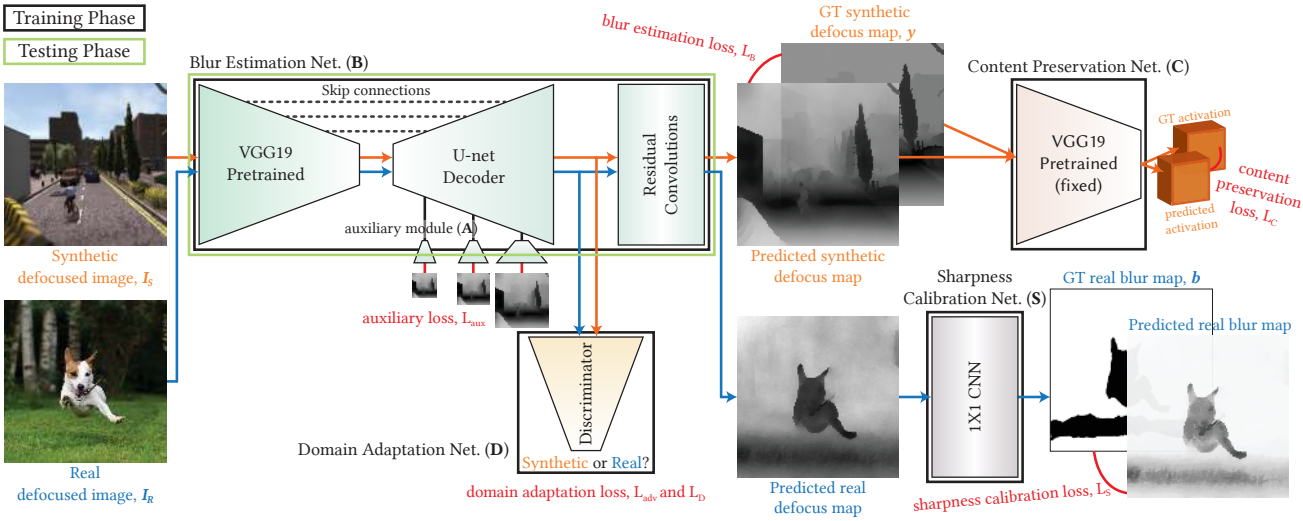


Figure 2: Architecture of *DMENet*. During training, we utilize all four subnetworks: blur estimation (***B***), domain adaptation (***D***), content preservation (***C***) and sharpness calibration (***S***) networks. They are jointly trained to learn blur amounts from synthetic defocused images while minimizing the domain difference between synthetic and real defocused images. For testing, we only utilize network ***B*** for estimating a defocus map given a real defocused image.

mon practice of adversarial training. Loss L_g is defined as:

$$L_g = \frac{1}{|\mathcal{D}_S|}\sum_{n=1}^{|\mathcal{D}_S|}\left\{L_B(I_S^n, y^n; \theta_B) + \lambda_C L_C(I_S^n, y^n; \theta_B)\right\} \quad (3)$$
$$+\frac{1}{|\mathcal{D}_B|}\sum_{n=1}^{|\mathcal{D}_B|}\{\lambda_{adv} L_{adv}(I_B^n; \theta_B) + \lambda_S L_S(I_R^n, b^n; \theta_B, \theta_S)\}$$
$$+\frac{\lambda_{adv}}{|\mathcal{D}_R|}\sum_{n=1}^{|\mathcal{D}_R|} L_{adv}(I_R^n; \theta_B),$$

where $|\mathcal{D}|$ is the number of elements in a set $\mathcal{D}$. L_B, L_C, L_S, and L_{adv} are blur map loss, content preservation loss, sharpness calibration loss, and adversarial loss, respectively, which will be discussed later. λ_c, λ_S, and λ_{adv} are hyperparameters to balance the loss terms. Loss L_d is defined as:

$$L_d = \frac{\lambda_D}{|\mathcal{D}_S|}\sum_{n=1}^{|\mathcal{D}_S|} L_D(I_S^n, 1; \theta_D) \quad (4)$$
$$+\frac{\lambda_D}{|\mathcal{D}_R| + |\mathcal{D}_B|}\left\{\sum_{n=1}^{|\mathcal{D}_R|} L_D(I_R^n, 0; \theta_D) + \sum_{n=1}^{|\mathcal{D}_B|} L_D(I_B^n, 0; \theta_D)\right\},$$

where L_D is discriminator loss and λ_D is a hyperparameter for balancing between L_g and L_d.

During training, the networks ***D***, ***C***, and ***S*** differently affect ***B*** depending on the domain of the input. In the case of synthetically blurred images with ground truth defocus maps, $\langle I_S, y\rangle \in \mathcal{D}_S$, we directly minimize the difference between y and the predicted defocus map $\boldsymbol{B}(I_S)$ using the blur map loss L_B that measures the mean squared error (MSE). We also minimize the content preservation loss L_C to reduce blurriness in the prediction $\boldsymbol{B}(I_S)$ using the network ***C***.

Real defocused images with binary blur maps, $\langle I_B, b\rangle \in \mathcal{D}_B$, are used to calibrate sharpness measurement from domain transferred features. With the supervision of b, the sharpness calibration loss L_S guides network ***S*** to classify whether an estimated defocus map $\boldsymbol{B}(I_B)$ has correct blur amounts, eventually calibrating network ***B*** to estimate correct degrees of sharpness from domain transferred features.

Finally, $I_S \in \mathcal{D}_S$, $I_B \in \mathcal{D}_B$, and $I_R \in \mathcal{D}_R$ are used together to minimize domain difference between features extracted from synthetic and real defocused images. For images I_S, the ground-truth domain labels are *synthetic*, while I_B and I_R are labeled as *real*. We minimize the discriminator loss L_D and the adversarial loss L_{adv} in an adversarial way, in which we train network ***D*** to correctly classify the domains of features from different inputs, while train network ***B*** to confuse ***D***.

In the remaining of this section, we describe the four networks and their associated loss functions in more detail.

4.2. Blur Estimation

The blur estimation network ***B*** is the core module in our *DMENet*. We adopt a fully convolutional network (FCN) [10], which is based on the U-net architecture [26] with slight changes. We initialize the encoder using a pretrained VGG19 [31] for better feature representations at the initial stage of training. The decoder uses up-sampling convolution instead of deconvolution to avoid checkerboard artifacts [23]. We also apply scale-wise auxiliary loss at each up-sampling layer to guide multi-scale prediction of a defocus map. This structure induces our network not only to be robust on vari-

ous object scales, but also to consider global and local contexts with large receptive fields. After the last up-sampling layer of the decoder, we attach convolution blocks with short skip connections to refine domain adapted features.

We use mean squared error (MSE) for the loss function L_B to estimate the overall structure of a defocus map and densely predict blur amounts in regions. Given a synthetic defocused image I_S of size $W \times H$, L_B is defined as:

$$L_B = \frac{1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} (\boldsymbol{B}(I_S; \theta_B)_{i,j} - y_{i,j})^2 + \lambda_{aux} L_{aux}, \tag{5}$$

where $\boldsymbol{B}(I_S; \theta_B)_{i,j}$ is the amount of blur of I_S predicted by network $\boldsymbol{B}$ at pixel (i, j) with learning parameters θ_B. $y_{i,j}$ is the corresponding ground-truth defocus value. L_{aux} is the scale-wise auxiliary loss defined as:

$$L_{aux} = \sum_{\ell=1}^{L_B} \frac{1}{W_l H_l} \sum_{i=1}^{W_\ell} \sum_{j=1}^{H_\ell} (\boldsymbol{B}_\ell(I_S; \theta_B, \theta_{aux})_{i,j} - y_{\ell,i,j})^2, \tag{6}$$

where $\boldsymbol{B}_\ell(I_S; \theta_B, \theta_{aux}) = \boldsymbol{A}_\ell(\boldsymbol{B}_\ell(I_S; \theta_B); \theta_{aux})$ is the output at the ℓ-th up-sampling level of network $\boldsymbol{B}$ converted to a defocus map by a small auxiliary network $\boldsymbol{A}_\ell$ parameterized by θ_{aux}. Each auxiliary network $\boldsymbol{A}_\ell$ consists of two convolutional layers, where the number of kernels in the first layer varies with level ℓ. λ_{aux} is a balance parameter. $W_\ell \times H_\ell$ is the size of a defocus map at the ℓ-th level. y_ℓ is the ground-truth defocus map resized to $W_\ell \times H_\ell$. L_B is the number of up-sampling layers in $\boldsymbol{B}$.

4.3. Domain Adaptation

Our domain adaptation network $\boldsymbol{D}$ compares the features of real and synthetic defocused images captured by the blur estimation network $\boldsymbol{B}$. We use adversarial training for network $\boldsymbol{D}$ so that the two domains have the same distributions in terms of extracted features. In principle, $\boldsymbol{D}$ is a discriminator in the GAN framework [7], but in our case, it makes the characteristics of the captured features of real and synthetic defocused images indistinguishable. We design $\boldsymbol{D}$ as a CNN with four convolution layers, each of which is followed by a batch normalization layer [11] and leaky rectified-linear-unit (ReLU) activation [36].

Discriminator loss We first train the network $\boldsymbol{D}$ as a discriminator to classify features from synthetic and real domains with the discriminator loss L_D, defined as

$$\begin{aligned} L_D = (z-1) \cdot \log(1 - \boldsymbol{D}(\boldsymbol{B}_{last}(I; \theta_B), \theta_D)) \\ -z \cdot \log(\boldsymbol{D}(\boldsymbol{B}_{last}(I; \theta_B), \theta_D)), \end{aligned} \tag{7}$$

where z is a label indicating whether the input feature comes from a real or synthetic defocused image, i.e., whether the input image I is real or synthetic; $z = 0$ if the feature is real and $z = 1$ otherwise. $\boldsymbol{B}_{last}(I, \theta_B)$ returns the feature maps of the last up-sampling layer of $\boldsymbol{B}$ for image I. Note that here we only train the parameters of the discriminator, θ_D.

Adversarial loss We then train network $\boldsymbol{B}$ to minimize the domain difference between features of synthetic and real defocused images. Given a real defocused image I_R, we define the adversarial loss L_{adv} for domain adaptation as:

$$L_{adv} = -\log(\boldsymbol{D}(\boldsymbol{B}_{last}(I_R; \theta_B), \theta_D)), \tag{8}$$

where we fix parameters θ_D and only train θ_B.

Here the main goal is to train the blur estimation network $\boldsymbol{B}$ so that it treats real and synthetic defocused images as they are from the same domain. As our domain adaptation network $\boldsymbol{D}$ becomes stronger as the domain classifier, network $\boldsymbol{B}$ has to generate more indistinguishable features for real and synthetic domains, minimizing the domain difference in terms of extracted features.

4.4. Content Preservation

Our blur estimation loss L_B is a MSE loss and has a nature of producing blurry outputs, as it takes the smallest value with the average of desirable targets [17]. To reduce the artifact, we use a content preservation loss [6] that measures the distance in a feature space ϕ, rather than in the image space itself. We define our content preservation network $\boldsymbol{C}$ as the pre-trained VGG19 [31]. During training, network $\boldsymbol{B}$ is optimized to minimize:

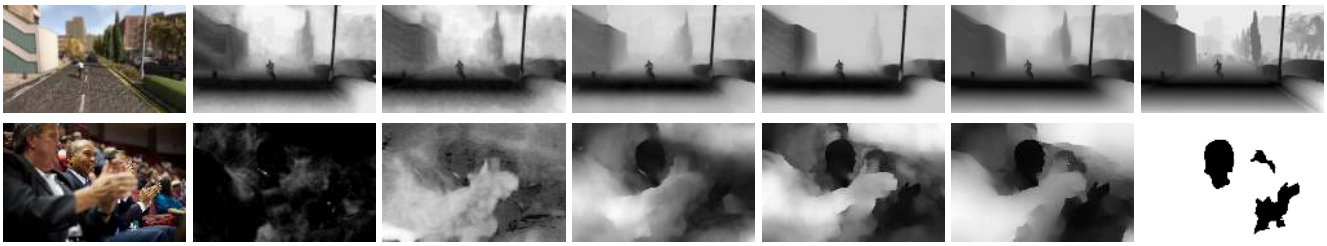
$$L_C = \frac{1}{W_\ell H_\ell} \sum_{i=1}^{W_\ell} \sum_{j=1}^{H_\ell} (\phi_\ell(\boldsymbol{B}(I_S; \theta_B))_{i,j} - \phi_\ell(y)_{i,j})^2, \tag{9}$$

where $W_\ell \times H_\ell$ is the size of a feature map $\phi_\ell(\cdot)$ at the last convolution layer in the ℓ-th max pooling block of VGG19.

4.5. Sharpness Calibration

Our domain adaptation network $\boldsymbol{D}$ concentrates on modulating the overall distributions of extracted features among real and synthetic defocused images, and it does not specifically align the amounts of blurs corresponding to the features between the two domains. In other words, the blur amounts learned by our blur estimation network $\boldsymbol{B}$ for synthetic defocused images cannot be readily applied to real defocused images, and we need to calibrate the estimated blur amounts for the two domains. To resolve this problem, our sharpness calibration network $\boldsymbol{S}$ provides additional information for the features captured from real defocused images in network $\boldsymbol{B}$ by correlating them with the blur information available in a blur detection dataset, where each pixel in an image is labeled as either sharp or blurred.

For a given real defocused image from the dataset, we train network $\boldsymbol{S}$ to classify the output of network $\boldsymbol{B}$ in terms of the correctness of estimated blurriness. The prediction is considered to be correct, only when a pixel estimated as sharp belongs to a sharp region in the input image. We build the network $\boldsymbol{S}$ with 1×1 convolutional layers, each of which



(a) Input (b) $DMENet_B$ (c) $DMENet_{BD}$ (d) $DMENet_{BDC}$ (e) $DMENet_{BDCS}^{w/o\ L_{aux}}$ (f) $DMENet_{BDCS}$ (g) ground-truth

Figure 3: Outputs generated with incremental additions of subnetworks in our network. The top row shows the defocus maps estimated from a synthetic defocus-blur image and the bottom row shows the results given a real DOF image. We can observe that each subnetwork improves the quality of the output. In (g), the ground-truths are a defocus map in *SYNDOF* dataset (top) and a binary blur mask in CUHK dataset (bottom).

is followed by batch normalization and leaky ReLU layer, enforcing the network $\boldsymbol{B}$ to densely estimate the blurriness. We apply a sigmoid cross entropy loss for optimization:

$$L_S = \frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H}\left\|\frac{1}{1+\exp(-\boldsymbol{S}(\boldsymbol{B}(I_B;\theta_B);\theta_S)_{i,j})} - \boldsymbol{b}_{i,j}\right\|_2^2, \quad (10)$$

where $\boldsymbol{b}$ is a ground truth binary blur map.

We used 1×1 kernels to maintain the same size of receptive field between the networks $\boldsymbol{B}$ and $\boldsymbol{S}$. Otherwise, as the receptive field of $\boldsymbol{S}$ becomes larger, gradients passed from $\boldsymbol{S}$ to $\boldsymbol{B}$ would be propagated to larger regions than the receptive fields of $\boldsymbol{B}$. A larger kernel for $\boldsymbol{S}$ eventually leads $\boldsymbol{B}$ to generate a smudged defocus map. Refer to the supplementary material for more details.

5. Experiments

This section reports our experiments that assess the performance of *DMENet* in generating defocus maps. We first summarize the setting of our experiments, and then discuss the influence of reciprocal connections between the subnetworks, $\boldsymbol{B}$, $\boldsymbol{D}$, $\boldsymbol{C}$, and $\boldsymbol{S}$. We then compare our results with the state-of-the-art methods on CUHK dataset [29] and RTF dataset [4], followed by a few applications of our *DMENet*. Details of evaluation results are accompanied in the supplementary material.

5.1. Experimental Configuration

Training details We use Adam [14] for optimizing our network. The network is trained with the batch size of 4, and the learning rate is initially set to 0.0001 with exponential decaying rate of 0.8 for every 20 epochs. Our model converged after around 60 epochs. The loss coefficients in Eqs. (3), (4) and (5) are set to: $\lambda_{adv} = 1e{-}3$, $\lambda_D = 1.0$, $\lambda_C = 1e{-}4$, $\lambda_S = 2e{-}2$, and $\lambda_{aux} = 1.0$. We use $\ell = 4$ for ϕ_ℓ, which indicates the last convolution layer before the fourth max-pooling layer in VGG19. We jointly train all the networks in an end-to-end manner on a PC with an NVIDIA GeForce TITAN-Xp (12 GB).

Dataset For synthetic defocused images I_S used for training network $\boldsymbol{B}$ (Eq. (5)), $\boldsymbol{C}$ (Eq. (9)), and $\boldsymbol{D}$ (Eq. (7)), we use images of SYNDOF dataset. We limit the maximum size c_{max} of COC to 28. For real defocused images I_R for domain adaptation, we used 2,200 real defocused images collected from Flickr and 504 images from CUHK blur detection dataset [29]. For sharpness calibration, we also use the same 504 images from CUHK dataset for real defocused images I_B, which require binary blur maps. During training, we augment all images with random flip, rotation, and crop. For evaluation, we used 200 images of CUHK dataset and 22 images of RTF dataset [4], which are not used for training.

5.2. Evaluation on Subnetworks

Fig. 3 shows the effects of incremental addition of subnetworks to estimate defocus maps from synthetic (upper row) and real (lower row) images. Given a synthetic image, $DMENet_B$ estimates a defocus map reasonably well. However, for a real defocused image, the sole use of subnetwork $\boldsymbol{B}$ for blur estimation fails (Fig. 3b), confirming there is significant domain difference between features of synthetic and real defocused images. With our domain adaptation, $DMENet_{BD}$ starts to recognize the degree of blur for a real image to some extents (Fig. 3c), yet with blurry output. Adding content preservation subnetwork ($DMENet_{BDC}$) effectively removes blur artifacts from the estimated defocus map, enhancing the estimation in texture regions (Fig. 3d). Finally, with the sharpness calibration subnetwork $\boldsymbol{S}$, $DMENet_{BDCS}$ correctly classifies real-domain features corresponding to blurry or sharp regions (Fig. 3f). We also compare results of $DMENet_{BDCS}$ with and without the scale-wise auxiliary loss L_{aux} (Eq. (6)). Fig. 3e demonstrates that the network without the auxiliary module generates a less clear and inaccurate defocus map.

5.3. Evaluation on CUHK and RTF Datasets

We compare our results with the state-of-the-art methods [40, 30, 24, 13, 38]. For ours, we used the final model $DMENet_{BDCS}$. To quantitatively assess the quality, we mea-

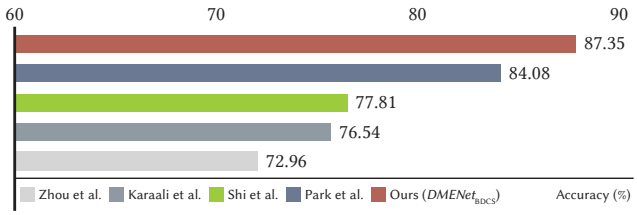


Figure 4: Accuracy comparison on CUHK dataset.

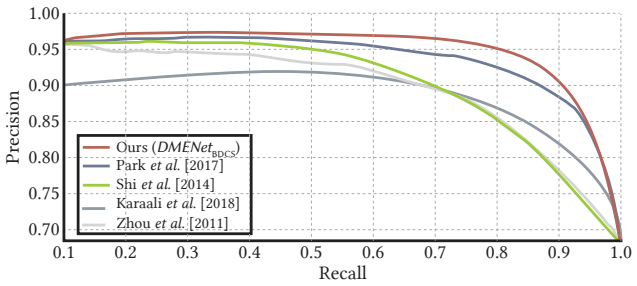


Figure 5: Precision-Recall comparison on CUHK dataset.

Figure 6: Qualitative comparison with [38]. From left to right: input, defocus map estimated by [38] and ours.

sured the accuracy and precision-recall of each method for 200 test images from CUHK blur detection dataset. As the dataset contains only binary blur maps for the ground-truths, we convert estimated defocus maps into binary blur maps. Following the method of Park *et al*. [24], the threshold τ for binarization is determined by $\tau = \alpha\, v_{\max} + (1-\alpha)\, v_{\min}$, where $v_{\max}$ and $v_{\min}$ are the maximum and minimum values in the estimated defocus map, respectively, and $\alpha = 0.3$.

Figs. 4 and 5 show quantitative comparison results. Our network significantly outperforms the previous methods in accuracy, which is the ratio of correctly classified pixels in a given image. Precision-recall curves also show superiority of our method in detecting blurred regions, where the curves are computed using defocus maps barbarized with different levels of τ that is slided from v_{min} to v_{max}.

Fig. 7 visually compares results generated by our network against previous methods, confirming the benefits of ours. First, our defocus maps show more continuous spectrum for the degrees of blur compared to others. In the first row of Fig. 7, our results exhibit less noise and smoother transitions with depth changes. Second, our network estimates more accurate blur for objects (*e.g.*, human, sky), as it is trained to consider scene contexts with mixture of COCs at object boundaries and ground-truth blur amounts on object interiors. In the second row of the figure, our result shows coherently labeled blur amounts while clearly respecting object boundaries. In

	[40]	[30]	[4]	[24]	[13]	Ours
MSE	0.037	0.082	0.033	0.024	0.064	**0.012**
MAE	0.143	0.241	0.106	0.129	0.199	**0.088**

Table 2: Evaluation of defocus map estimation result on RTF dataset in terms of mean squared error (MSE) and mean absolute error (MAE).

the third row, our method estimates consistent blur amounts both for the box surface and the symbol on it, while some other methods differently handle the symbol due to its strong edges. Lastly, our method is more robust in homogeneous regions. In the second and fourth rows, our results shows little smudginess around some objects, but they are still accurate in terms of relative depths. For instance, the sky should be farther than the mountain, which is not necessarily preserved with other methods.

We also report qualitative results compared with the most recent approach [38], whose implementation has not been publicized yet. Fig. 6 shows that our model can handle wider depth range of a scene. While our defocus map includes all the people who are located throughout the depth range in the scene, the result of [22] only deals with people within narrow depth range.

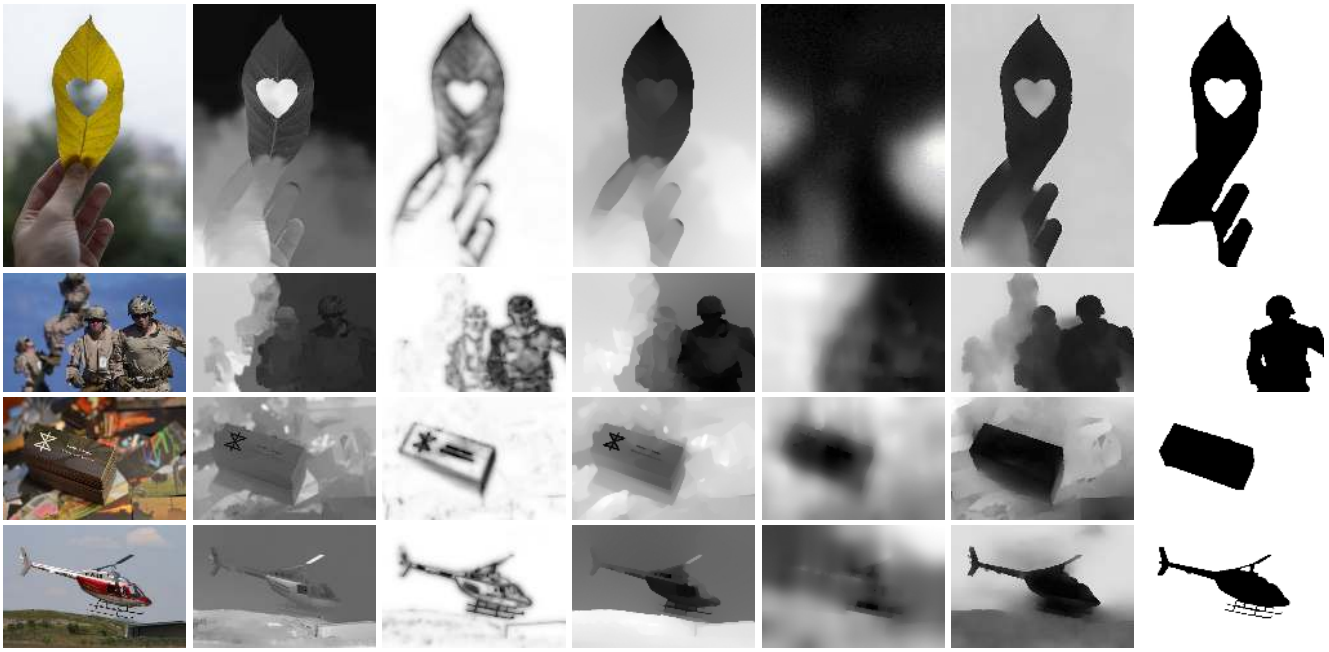
In addition, we conducted evaluation on RTF dataset [4], which consists of 22 real defocused images and ground truth defocus maps labeled with radii of disc PSFs. For all compared methods considering Gaussian PSF (including ours), we rescaled defocus maps using a conversion function that authors of [4] provide, which maps a Gaussian PSF into a disc PSF by measuring the closest fit. Our network shows the state-of-the-art accuracy on the dataset (Table 2). More detailed evaluation are in the supplementary material.

5.4. Applications

Defocus blur magnification Given an input image and its estimated defocus map, we can generate a magnified defocus-blur image (Fig. 8). We first estimate the blur amount $\sigma_{i,j}$ for each pixel using $DMENet_{BDCS}$. Then, we blur each pixel using $m \cdot \sigma_{i,j}$ for σ of Gaussian blur kernel, where m is a magnifying scale ($m = 8$ in Fig. 8). We used the same blur algorithm used for generating our *SYNDOF* dataset. The defocus blur magnification result demonstrates the accuracy of our estimated defocus map.

All-in-focus image generation Our estimated defocus map can be naturally utilized for deblurring (Fig. 9). From the estimated defocus map, we generate a Gaussian blur kernel for each pixel with the estimated σ. We then use non-blind image deconvolution technique leveraging hyper-Laplacian [16]; to handle spatially-varying deblur, we applied deconvolution to the decomposed layers, and compose deconvolved layer images.

Depth from blur Even without the presence of precise parameters related to the optical geometry (focus point, fo-



(a) (b) (c) (d) (e) (f) (g)

Figure 7: Qualitative comparison between $DMENet_{BDCS}$ and other methods: (a) Input and the defocus maps estimated by (b) Zhou *et al*. [40], (c) Shi *et al*. [30], (d) Park *et al*. [24], (e) Karaali *et al*. [13], (f) ours, and (g) ground-truth binary blur masks.

Figure 8: Defocus blur magnification using the defocus map estimated by $DMENet_{BDCS}$. From left to right: Input and our blur magnification result.

Figure 9: Deblurring using our defocus map estimated by $DMENet_{BDCS}$. From left to right: input and our deblurring result.

cal length, and aperture number), we can approximate the pseudo-depth using a scaled defocus map in a limited yet common scenario (*i.e*., a focus point is at either depth z_{near} or z_{far}). We used a light-field dataset [8, 34] to compare with the ground-truth depth map. Fig. 10 shows our estimated defocus map can provide a good approximation for the depth map.

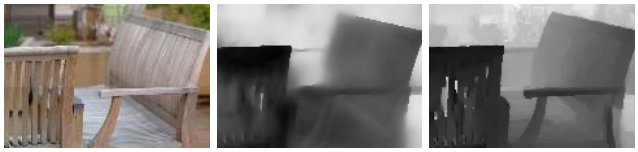


Figure 10: Depth from our defocus map estimated by $DMENet_{BDCS}$. From left to right: input, depth from our estimated defocus map and ground truth depth.

6. Conclusions

SYNDOF dataset can be improved with a more precise DOF rendering technique (*e.g*., distributed ray tracing [3]) and a more realistic optical model (*e.g*., a thick-lens or compound-lens model). A more systematic capture of training images to cover more varieties of real world defocused images is also an interesting direction for future work. Our network works best with LDR images, and strong highlights (*i.e*., bokeh) may not be properly handled. We plan to include the bokeh and HDR images as well in our *SYNDOF* dataset.

Acknowledgements We appreciate the constructive comments from the reviewers. This work was supported by the Ministry of Science and ICT, Korea, through IITP grant (IITP-2015-0-00174) and NRF grants (NRF-2017M3C4A7066316, NRF-2017M3C4A7066317). It was also supported by the DGIST Start-up Fund Program (2018010071).

References

[1] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proc. CVPR*, 2017.

[2] Y. Chen, W. Chen, Y. Chen, B. Tsai, Y. F. Wang, and M. Sun. No more discrimination: Cross city adaptation of road scene segmenters. In *Proc. ICCV*, 2017.

[3] R. L. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. *SIGGRAPH Computer Graphics*, 18(3):137–145, 1984.

[4] L. D'Andrès, J. Salvador, A. Kochale, and S. Susstrunk. Non-parametric blur map regression for depth of field extension. *IEEE Trans. Image Processing (TIP)*, 25(4):1660–1673, 2016.

[5] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proc. ICML*, 2015.

[6] L. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Proc. NIPS*, 2015.

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proc. NIPS*, 2014.

[8] M. Hirsch, D. Lanman, G. Wetzstein, and R. Raskar. Tensor displays. In *ACM SIGGRAPH 2012 Emerging Technologies*, pages 24:1–24:1, 2012.

[9] J. Hoffman, E. Tzeng, T. Park, J Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell. CyCADA: Cycle-consistent adversarial domain adaptation. In *Proc. ICML*, 2018.

[10] J. Hoffman, D. Wang, F. Yu, and T. Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *CoRR*, abs/1612.02649, 2016.

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, 2015.

[12] P. Jiang, H. Ling, J. Yu, and J. Peng. Salient region detection by ufo: Uniqueness, focusness and objectness. In *Proc. ICCV*, 2013.

[13] A. Karaali and C. Jung. Edge-based defocus blur estimation with adaptive scale selection. *IEEE Trans. Image Processing (TIP)*, 27(3):1126–1137, 2018.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2014.

[15] M. Kraus and M. Strengert. Depth-of-field rendering by pyramidal image processing. In *Proc. Erugraphics*, 2007.

[16] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. In *Proc. NIPS*, 2009.

[17] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proc. ICML*, 2016.

[18] S. Lee, E. Eisemann, and H. Seidel. Real-time lens blur effects and focus control. *ACM Trans. Graphics (TOG)*, 29(4):65:1–65:7, 2010.

[19] S. Lee, G. J. Kim, and S. Choi. Real-time depth-of-field rendering using point splatting on per-pixel layers. 27(7):1955–1962, 2008.

[20] S. Lee, G. J. Kim, and S. Choi. Real-time depth-of-field rendering using anisotropically filtered mipmap interpolation. *IEEE Trans. Visualization and Computer Graphics (TVGC)*, 15(3):453–464, 2009.

[21] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Unsupervised domain adaptation with residual transfer networks. In *Proc. NIPS*, 2016.

[22] M. McGuire, W. Matusik, H. Pfister, J. F. Hughes, and F. Durand. Defocus video matting. *ACM Trans. Graphics (TOG)*, 24(3):567–576, 2005.

[23] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[24] J. Park, Y. Tai, D. Cho, and I. S. Kweon. A unified approach of multi-scale deep and hand-crafted features for defocus estimation. In *Proc. CVPR*, 2017.

[25] M. Potmesil and I. Chakravarty. A lens and aperture camera model for synthetic image generation. *ACM Trans. Graphics (TOG)*, 15(3):297–305, 1981.

[26] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI*, 2015.

[27] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proc. CVPR*, 2016.

[28] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nesic, X. Wang, and P. Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *Proc. GCPR*, 2014.

[29] J. Shi, L. Xu, and J. Jia. Discriminative blur detection features. In *Proc. CVPR*, 2014.

[30] J. Shi, L. Xu, and J. Jia. Just noticeable defocus blur detection and estimation. In *Proc. CVPR*, 2015.

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[32] C. Tang, C. Hou, and Z. Song. Defocus map estimation from a single image via spectrum contrast. *Optic Letters*, 38(10):1706–1708, 2013.

[33] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *Proc. CVPR*, 2017.

[34] G. Wetzstein, D. Lanman, W. Heidrich, and R. Raskar. Layered 3d: Tomographic image synthesis for attenuation-based light field and high dynamic range displays. *ACM Trans. Graphics (TOG)*, 30(4):95:1–95:12, 2011.

[35] J. Wulff, D. J. Butler, G. B. Stanley, and M. J. Black. Lessons and insights from creating a synthetic optical flow benchmark. In *ECCV Workshop on Unsolved Problems in Optical Flow and Stereo Estimation*, 2012.

[36] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015.

[37] G. Xu, Y. Quan, and H. Ji. Estimating defocus blur via rank of local patches. In *Proc. ICCV*, 2017.

[38] S. Zhang, X. Shen, Z. Lin, R. Měch, J. P. Costeira, and J. M. F. Moura. Learning to understand image blur. In *Proc. CVPR*, 2018.

[39] C. Zhou and S. K. Nayar. What are good apertures for defocus deblurring? In *Proc. ICCP*, 2009.

[40] S. Zhuo and T. Sim. Defocus map estimation from a single image. *Pattern Recognition*, 44(9):1852–1858, 2011.