Single Image Defocus Deblurring
Using Kernel-Sharing Parallel Atrous Convolutions

Hyeongseok Son   Junyong Lee   Sunghyun Cho   Seungyong Lee

POSTECH

{sonhs, junyonglee, s.cho, leesy}@postech.ac.kr

Abstract

This paper proposes a novel deep learning approach for single image defocus deblurring based on inverse kernels. In a defocused image, the blur shapes are similar among pixels although the blur sizes can spatially vary. To utilize the property with inverse kernels, we exploit the observation that when only the size of a defocus blur changes while keeping the shape, the shape of the corresponding inverse kernel remains the same and only the scale changes. Based on the observation, we propose a kernel-sharing parallel atrous convolutional (KPAC) block specifically designed by incorporating the property of inverse kernels for single image defocus deblurring. To effectively simulate the invariant shapes of inverse kernels with different scales, KPAC shares the same convolutional weights among multiple atrous convolution layers. To efficiently simulate the varying scales of inverse kernels, KPAC consists of only a few atrous convolution layers with different dilations and learns per-pixel scale attentions to aggregate the outputs of the layers. KPAC also utilizes the shape attention to combine the outputs of multiple convolution filters in each atrous convolution layer, to deal with defocus blur with a slightly varying shape. We demonstrate that our approach achieves state-of-the-art performance with a much smaller number of parameters than previous methods.

1. Introduction

Defocus blur of an image occurs when the light ray from a point in the scene forms a circle of confusion (COC) on the camera sensor. The aperture shape and lens design of the camera determine the blur shape, and the blur size varies upon the depth of a scene point and intrinsic camera parameters. In a defocused image, the spatial variance of the blur size is large, while that of the blur shape is relatively small. Single image defocus deblurring remains a challenging problem as it is hard to accurately estimate and remove defocus blur spatially varying in both size and shape.

The conventional two-step approach [2, 23, 5, 18, 4, 11, 14] reduces the complexity of defocus deblurring by assuming an isotropic kernel for the blur shape, such as disc [2, 5] or Gaussian [23, 18, 11, 14]. Based on the assumption, the approach first estimates a defocus map containing the per-pixel blur size of a defocused image, then uses the defocus map to perform non-blind deconvolution [8, 15, 12] on the image. However, real-world defocused images may have more complex kernel shapes than disc or Gaussian, and the discrepancy often hinders accurate defocus map estimation and successful defocus deblurring.

Recently, Abuolaim and Brown [1] proposed DPDNet, the first end-to-end defocus deblurring network, that learns to directly deblur a defocused image without relying on a restrictive blur model. They also presented a defocus deblurring dataset that includes stereo images attainable from a dual-pixel sensor camera. Thanks to the end-to-end learning and the strong supervision provided by the dual-pixel dataset, DPDNet outperforms two-step approaches on deblurring of real-world defocused images. Still, the deblurred results tend to include ringing artifacts or remaining blur, as the conventional encoder-decoder architecture of DPDNet confines its capability in handling spatially variant blur [34].

In this paper, we propose a novel deep learning approach for single image defocus deblurring based on inverse kernels. It was shown that deconvolution of an image with a given blur kernel can be performed by convolving the image with an inverse kernel [30], where the inverse kernel could be computed from the given blur kernel using Fourier transform. Xu *et al.* [29] trained a deep network to learn uniform deconvolution by introducing the property of pseudo inverse kernel into the network. Similarly, we train our network to learn the deconvolution operation by capitalizing the specific characteristics of inverse kernels needed for defocus deblurring. However, due to the spatially varying nature of defocus blur, the inverse kernel required for defocus deblurring also changes per pixel. Training a deep network to learn deconvolution operations of varying inverse kernels would be challenging even with the guidance of defocused

and sharp image pairs.

To reduce the complexity, we use the property of defocus blur that the blur shape is similar in a defocused image although the blur size can drastically change. However, instead of assuming any specific blur shape as in the two-step approach, we exploit our observation on inverse kernels; When only the size of a blur changes while keeping the shape, the shape of the corresponding inverse kernel remains the same and the size changes in the same way as the blur (Sec. 3.1). Then, we may constrain our network to simulate inverse kernels with a single shape but with different sizes to cover spatially varying defocus blur. However, it is hard to directly simulate inverse kernels with all possible sizes in practice. Instead, we design our network to carry a few convolutional layers to cover inverse kernels with a discrete set of sizes, and aggregate the outputs of the layers for handling blur with arbitrary size. As a result, compared to the conventional two-step approach and the recent deep learning approach [1], our method can perform defocus deblurring more accurately by exploiting the properties of defocus blur in the form of inverse kernel.

To implement the network design, we propose a novel *kernel-sharing parallel atrous convolutional (KPAC) block*. The KPAC block consists of multiple atrous convolution layers [9, 3] with different dilation rates and additional layers for the scale and shape attentions. The multiple atrous convolution layers share the same convolution kernels representing the invariant shape of an inverse kernel, and different dilation rates of the layers correspond to inverse kernels with a discrete set of sizes. To simulate deconvolution using inverse kernels of other sizes, the KPAC block equips a spatial attention mechanism [28], which we call *the scale attention*, to aggregate the outputs of the atrous convolution layers. By combining the per-pixel scale attention with multiple atrous convolution layers, the KPAC block can handle the spatially varying size of defocus blur. In addition, the shape of defocus blur may slightly change in a defocused image due to the non-linearity of a camera image pipeline. To handle the variance, we include a channel attention mechanism [33], which we call *the shape attention*, in a KPAC block to support the slight shape change of the inverse kernel.

An important benefit of our KPAC block is its small number of parameters, enabled by kernel sharing of the multiple atrous convolution layers. As a result, our defocus deblurring network is lighter-weighted than the previous work [1], showing the better performance (Sec. 5.2).

To summarize, our contributions include:

- Novel deep learning approach for single image defocus deblurring based on inverse kernels,

- Novel Kernel-Sharing Parallel Atrous Convolutional (KPAC) block designed upon the properties of spatially varying inverse kernels for defocus deblurring,

- Light-weight single image defocus deblurring network, which shows state-of-the-art performance.

2. Related Work

2.1. Defocus deblurring

Conventional approaches [2, 23, 5, 18, 4, 11, 14] perform defocus deblurring in two steps of defocus map estimation and non-blind deconvolution. As they use existing non-blind deconvolution methods [15, 12] for deblurring, they focus on improving the accuracy of defocus map estimation based on parametric blur models such as disc and Gaussian blur. Various methods were proposed for defocus map estimation using hand-crafted features such as edge gradients [11], sparse coded features [23], machine learning features [5], combination of hand-crafted and deep learning features [18], and an end-to-end deep learning model [14]. These two-step approaches often fail to produce faithful deblurring results due to the restricted blur model as well as the errors of defocus map estimation.

Recently, Abuolaim and Brown [1] proposed the first end-to-end model for deep learning-based defocus deblurring and a dataset for supervised training. The model outperforms conventional two-step approaches, and shows that a dual-pixel image input significantly improves defocus deblurring performance. However, their network structure does not explicitly consider the spatially varying nature of defocus blur with large variance in size but small variance in shape, and the performance has rooms for improvements.

2.2. Inverse kernel

Convolution of an inverse kernel can be used to perform deblurring on an image [27, 31, 30]. However, a naïve inverse kernel that could be obtained from Wiener deconvolution [27] usually introduces unwanted artifacts such as ringing and amplified noise. To suppress artifacts, previous works took a progressive approach using an image pyramid [31], regularization using sparse priors [15, 30], a neural network for postprocessing [25], and feature space processing [6]. Xu *et al.* [29] and Ren *et al.* [20] directly used a CNN to perform non-blind deconvoluton by adapting a separable property of a large inverse kernel, and showed that their approaches are effective for suppressing artifacts. Distinct from [29, 20] that simulate a single inverse kernel for uniform deconvolution, our approach simulates spatially varying inverse kernels needed for defocus deblurring.

3. Key Idea

In this section, we first present our observation on inverse kernels for deconvolution, and propose inverse kernel-based deconvolution to deal with spatially varying defocus blur

(a) blurred by $\frac{1}{5^2}k_{\uparrow 5}$ (b) deblurred by $(\frac{1}{5^2}k_{\uparrow 5})^\dagger$ (c) deblurred by $\frac{1}{5^2}(k^\dagger_{\uparrow 5})$

Figure 1. Deblurring examples using the two kernels in Eq. (4). Wiener deconvolution is used for computing the inverse kernels. The results in (b) and (c) are almost equivalent. Their PSNRs to the original sharp image are 21.64dB and 21.59dB, respectively, and the PSNR between them is 51.09dB.

(Sec. 3.1). We then present experiments to verify the observation and the proposition (Sec. 3.2).

3.1. Inverse kernel-based deconvolution for spatially varying defocus blur

The inverse kernel-based deconvolution approach is closely related to convolution neural networks (CNN) by nature, as they are based on convolutional operations in the spatial domain. We aim to design a network architecture that takes the benefits of both CNN and inverse kernels for defocus deblurring. To this end, we first introduce the concept and general derivation of pseudo inverse kernel, as in previous works [29, 20].

We consider a simple blur model defined with convolution operation $*$ as

$$y = k * x, \qquad (1)$$

where k is a blur kernel, and y and x are a blurry image and a latent sharp image, respectively. The spatial convolution can be transformed to an element-wise multiplication in the frequency domain as

$$F(y) = F(k) \cdot F(x), \qquad (2)$$

where $F(\cdot)$ denotes the discrete Fourier transform. Then, the latent sharp image can be derived with convolution operation as

$$x = F^{-1}(1/F(k)) * y = k^\dagger * y, \qquad (3)$$

where $F^{-1}(\cdot)$ denotes the inverse discrete Fourier transform, and $k^\dagger$ is the spatial pseudo inverse kernel.

We observe that the shape of the corresponding inverse kernel $k^\dagger$ remains the same when the spatial support size of a blur kernel k changes, i.e.,

$$(\frac{1}{s^2}k_{\uparrow s})^\dagger = \frac{1}{s^2}(k^\dagger_{\uparrow s}), \qquad (4)$$

where $\uparrow s$ denotes general upsampling operation with a scale factor s in the spatial domain (refer to the supplementary material for more details on kernel upsampling). In Eq. (4), $\frac{1}{s^2}k_{\uparrow s}$ is the kernel k' with the upsampled resolution, where $\frac{1}{s^2}$ is used for normalizing the kernel weights. Then, the inverse kernel of k' is the upsampled version of $k^\dagger$ with the same scale factor s.

(a) blurred by $\frac{1}{3.5^2}k_{\uparrow 3.5}$ (b) deblurred by $\frac{1}{3.0^2}(k^\dagger_{\uparrow 3.0})$ (c) deblurred by $\frac{1}{4.0^2}(k^\dagger_{\uparrow 4.0})$ (d) blended using $0.5*(b)+0.5*(c)$ (e) deblurred by $\frac{1}{3.5^2}(k^\dagger_{\uparrow 3.5})$

Figure 2. Linear combination of deblurring results from different inverse kernels.

Based on the observation, to handle defocus blur with varying sizes but with the same shape, we may use inverse kernels sharing a single shape but with different sizes. However, as we aim to utilize a CNN for defocus deblurring, it is not practical to implement a network that carries the inverse kernels of all possible sizes. To reduce the complexity, we may approximate deconvolution of an image by combining the results of applying inverse kernels with a discrete set of sizes to the image. Similarly to a classical approach [2] that handles non-uniform blur using a linear combination of differently deblurred images, spatially varying deconvolution for defocus blur can be approximated as

$$x \approx \sum_i \{\alpha_i \cdot (\frac{1}{s_i^2}k^\dagger_{\uparrow s_i} * y)\}, \qquad (5)$$

where $s_i \in \{1, \cdots, n\}$ is an upsampling factor, and α_i is the per-pixel weight map for the result image obtained using the inverse kernel with an upsampling factor s_i.

3.2. Experimental validation of the key idea

In this section, we experimentally validate the observed property of an inverse kernel for defocus deblurring (Eq. (4)) and the approach for spatially varying deconvolution based on the property (Eq. (5)). In the validation, we use Wiener deconvolution [27] to compute a spatial inverse kernel and Lanczos upsampling [7] to scale inverse kernels. We use Wiener inverse kernel as it has a finite spatial support analogous to the finite receptive field of a CNN, due to the involvement of signal-to-noise ratio (SNR) as regularization [29].

Regarding Eq. (4), Fig. 1 shows an example that two kernels on both sides of Eq. (4) produce equivalent deblurring results. We also include the proof of Eq. (4) in the supplementary material. Regarding Eq. (5), Fig. 2 shows that a linear combination of resulting images from the inverse kernels with a discrete set of sizes can approximate deconvolution with an inverse kernel of a different size. Figs. 2b and 2c are the deblurred results obtained by convolving inverse kernels of different sizes. While neither kernel fits the actual blur size, we can still obtain a visually decent result (Fig. 2d) with an approximation accuracy[1] of 87.4%, by simply applying Eq. (5) with $\alpha_i = 0.5$. When we optimize α_i using the method of non-negative least squares, the accuracy

---

[1]The accuracy is computed by $1 - MAE(1, \hat{x}/x_s)$, where MAE is the mean absolute error, $/$ is pixel-wise division, x_s is the deconvolution result using an inverse kernel of a target scale (e.g., Fig. 2e), and $\hat{x}$ is the approximated deconvolution result computed using Eq. (5) (e.g., Fig. 2d).
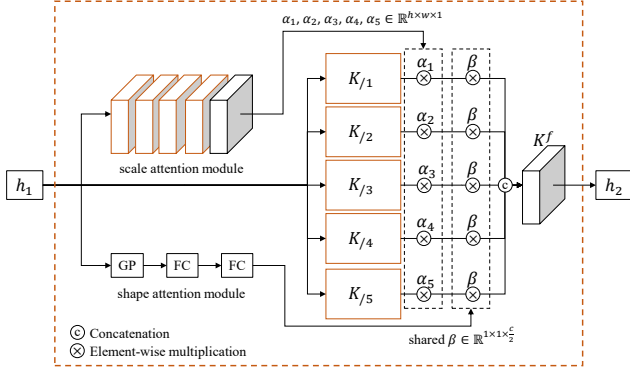
Figure 3. Architecture of Kernel-sharing Parallel Atrous Convolutional (KPAC) block. GP and FC denote global average pooling and a fully connected layer, respectively.

further increases up to 92.5%. These experiments confirm the validity of expanding Eqs. (4) and (5) to a CNN architecture, where the remaining errors could be compensated through deep learning process (Sec. 4). Refer to the supplementary material for more examples.

4. Network Design

Based on the key idea, we design a network reflecting the property of inverse kernels for defocus deblurring. Instead of directly adopting the linear combination in Eq. (5), we extend the concept of combination with a convolution layer that aggregates the results of multiple inverse kernels, to fully exploit the non-linear nature of deep learning.

Let $h_1 \in \mathbb{R}^{h \times w \times c}$ and $h_2 \in \mathbb{R}^{h \times w \times c}$ be the input and output feature maps that correspond to images y and x in Eq. (3), respectively. We modify Eq. (5) to represent defocus deblurring using multiple convolution layers as

$$h_2 = K^f * \bigparallel_{i=1}^{n} \{\alpha_i \cdot (K_{s_i} * h_1)\}, \quad (6)$$

where $K_{s_i} \in \mathbb{R}^{s_i k \times s_i k \times c \times \frac{c}{2}}$ are the convolutional kernels representing an inverse kernel with an upsampling factor s_i. $K^f \in \mathbb{R}^{3 \times 3 \times \frac{nc}{2} \times c}$ is for non-linear aggregation of multiple scales of the outputs of $\{K_{s_i}\}$. $\alpha_i \in \mathbb{R}^{h \times w \times 1}$ is a per-pixel weight map for scale s_i, and $\parallel$ denotes concatenation operation. Based on this modification, we propose a kernel-sharing parallel atrous convolutional (KPAC) block.

4.1. Kernel-sharing Parallel Atrous Convolutional (KPAC) block

Our KPAC block consists of multiple kernel-sharing atrous convolution layers, and modules for the scale and shape attentions (Fig. 3).

Multiple kernel-sharing atrous convolutions In Eq. (6), convolutional kernels $\{K_{s_i}\}$ should represent inverse kernels of the same shape but with different sizes as observed in Eq. (4). We may share the weights of $\{K_{s_i}\}$ to enforce

(a) deblurred by $k^\dagger_{\uparrow/5}$ (b) $\frac{1}{5^2}(k^\dagger_{\uparrow 5})$ (c) $k^\dagger_{\uparrow/5}$
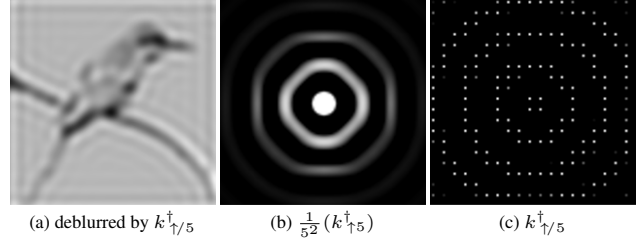
Figure 4. All settings are the same as in Fig. 1. (a) deblurring result using a dilated inverse kernel. (a) is almost same as (b) and (c) in Fig. 1, and the PSNR between (a) and Fig. 1(c) is 53.74dB. (b) & (c) shapes of the scaled and dilated inverse kernels.

the constraint of the same inverse kernel shape. However, in practice, the weight sharing is not straightforward because of the different sizes of $\{K_{s_i}\}$.

We resolve this problem with a simple but effective solution based on another important observation; For a blurred region, which is spatially smooth, filtering operations on sparsely sampled pixels (with a dilated kernel) and on densely sampled pixels (with a rescaled kernel) produce similar results. Thus, the upsampling operation for $\frac{1}{s^2}(k^\dagger_{\uparrow s})$ in Eq. (4) can be replaced by a dilation operation, yielding an inverse kernel applied to sparsely sampled pixels. The dilation operation does not change the number of filter weights but only scales the spatial support of the kernel without resampling, enabling direct weight sharing for the convolutional kernels $\{K_{s_i}\}$.

Fig. 4 shows an example that $\frac{1}{s^2}(k^\dagger_{\uparrow s})$ and $k^\dagger_{\uparrow/s}$ produce almost equivalent deblurring results, where $\uparrow/s$ denotes the dilation operation. We also experimentally verified that a modified version of Eq. (5) using $k^\dagger_{\uparrow/s}$ produces results almost equivalent to Fig. 2. Refer to the supplementary material for the experiment on dilated inverse kernels.

Based on the observation, our KPAC block includes multiple atrous convolution layers with different dilation rates, placed in parallel (Fig. 3). The atrous convolution kernels $K_{/s_i}$ are composed of the same number of kernel weights regardless of scales and share the same kernel weights, satisfying the constraint of shared inverse kernel shape. That is, we substitute the standard convolution in Eq. (6) by an atrous convolution layer, obtaining

$$h_2 = K^f * \bigparallel_{i=1}^{n} \{\alpha_i \cdot (K_{/s_i} * h_1)\}, \quad (7)$$

where $K_{/s_i} \in \mathbb{R}^{k \times k \times c \times \frac{c}{2}}$ denotes the kernel with a dilation rate s_i in the kernel-sharing atrous convolution layer.

Scale attention Our KPAC block represents an inverse kernel with an arbitrary scale by aggregating resulting feature maps from multiple scales of atrous convolution layers. To dynamically determine pixel-wise weights for aggregation, we use a scale attention $\alpha_i \in \mathbb{R}^{h \times w \times 1}$ based on spatial attention mechanism [28]. As the scale of defocus deblurring is determined by a combination of atrous convolution
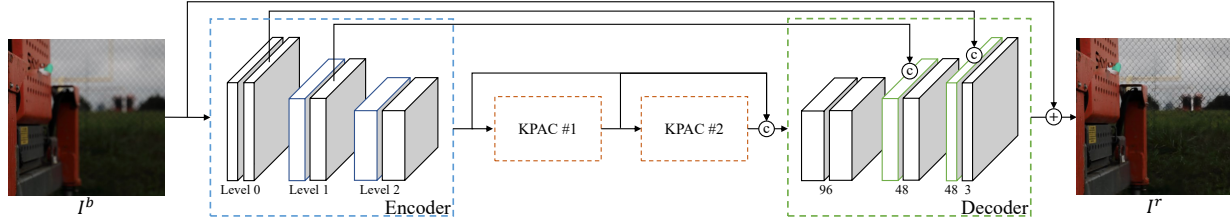
Figure 5. Overall network architecture. Refer to the supplementary material for the detailed architecture of our network.

layers, a different scale attention map should be applied to the resulting feature map of each atrous convolution layer.

Shape attention   Although we have assumed that inverse kernels share the same shape, the shape of defocus blur and then the corresponding shape of the inverse kernel can change in a defocused image due to the non-linearity in a camera imaging pipeline. As an atrous convolution layer consists of multiple convolution filters, different combinations of the filters can represent different inverse kernels. To support shape variations of inverse kernels, we use a shape attention module based on channel attention mechanism [33] that determines combination weights for filters on the resulting feature maps. As the atrous convolution layers in our KPAC block should share the inverse kernel shape, they share the channel-wise weight vector $\beta \in \mathbb{R}^{1 \times 1 \times \frac{c}{2}}$ from the shape attention module.

4.2. Defocus deblurring network

For effective multi-scale processing in the feature space, we adopt an encoder-decoder structure [21] for our defocus deblurring network (Fig. 5). The network consists of three parts: encoder, KPAC blocks, and decoder. Except for the final convolution, all convolution layers include LeakyReLU [16] for the non-linear activation layer.

As our KPAC block is designed upon inverse kernels defined in the linear space, one question naturally follows whether it is proper for the KPAC block to run in the non-linear feature space. We found that the KPAC block still works with non-linear features as CNNs are locally linear [17, 13]. It has also been shown in a recent work [6] that Wiener deconvolution, which is a kind of inverse filters, can be successfully extended to the feature space due to the piecewise linearity of the feature space of CNN.

The proposed KPAC block may not operate exactly the same as conventional inverse kernel-based approaches for deblurring, as it does not explicitly employ inverse kernels. However, the architecture is still constrained with atrous convolutional layers with shared kernels and non-linear aggregation of resulting features, which are designed upon the property of inverse kernels. As a result, the KPAC block would learn restoration kernels that are more robust and effective for defocus deblurring. In addition, while a single KPAC block is designed to model the entire process of deblurring, we can stack multiple KPAC blocks to exploit the iterative nature for removing residual blurs.

Training   For training the defocus deblurring network, we use the mean absolute error (MAE) between a network output and the corresponding ground-truth sharp image as a loss function. We also employ the perceptual loss [10] for restoring more realistic textures. For the perceptual loss, we use the feature map extracted at the 'conv4_4' layer in the pre-trained VGG-19 network [24]. When the perceptual loss is used, it is combined with the MAE loss, where the balancing factor is 7×10^{-4} for the perceptual loss.

5. Experiments

We implemented and evaluated our models using Tensorflow 1.10.0 with NVIDIA Titan Xp GPU. Our final model has two KPAC blocks with a kernel size $k = 5$ and the number of atrous convolution layers $n = 5$, as we empirically found it to work well in most cases. We use negative slope coefficient $\lambda = 0.2$ for LeakyReLU layers. We use the Adam optimizer [19] with $\beta_1 = 0.9$ and $\beta_2 = 0.99$ to train our models. We train our models for 200k iterations with the fixed learning rate of 1×10^{-4}. We tested a model trained for more iterations with learning rate decay, but its improvement was marginal in PSNR. For evaluation in Sec. 5.2, we train our models with the perceptual loss [10]. For those models, we initialize them with pre-trained models trained with the MAE loss for 200k iterations. Then, we fine-tune the networks with both MAE and perceptual losses for additional 100k iterations with the fixed learning rate of 5×10^{-5}. We use the batch size of 4. Each image in a batch is randomly cropped to 512×512.

Dataset   We use the DPDD dataset [1] for evaluation of our models. The dataset provides 500 image pairs of a real-world defocused image and the corresponding all-in-focus ground-truth image captured by a Canon EOS 5D Mark IV. The dataset consists of training, validation, and testing sets of 350, 74, and 76 pairs of images, respectively. In our experiments, we train and evaluate our models using the training and testing sets, respectively. While the dataset also provides dual-pixel data, we do not use them in our experiments. The dataset provides 16-bit images in the PNG format. We convert them to 8-bit images for our experiments.

5.1. Analysis

Effect of weight sharing   Our KPAC blocks learn spatially varying inverse kernels whose shapes remain the
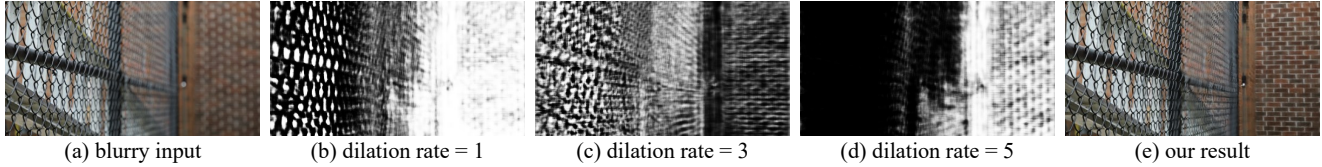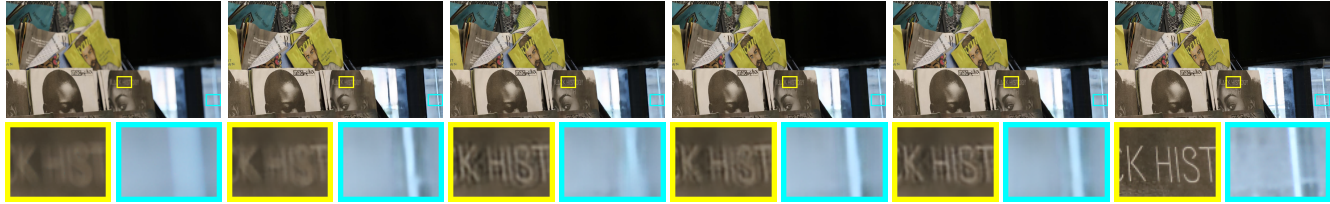
| (a) blurry input | (b) dilation rate = 1 | (c) dilation rate = 3 | (d) dilation rate = 5 | (e) our result |

Figure 6. Visualization of scale attention maps for different atrous convolution layers in the first block of our 2-block network.

| (a) Input | (b) baseline | (c) no attention | (d) w/ scale | (e) w/ scale & shape | (f) GT |

Figure 7. Qualitative examples of the ablation study. The baseline denotes a network model using conventional residual convolution blocks without our KPAC blocks. 'scale' and 'shape' denote scale attention and shape attention, respectively.

	PSNR (dB)	Parameters (M)
w/o weight sharing	24.78	2.50
w/ weight sharing	25.21	1.58

Table 1. Effect of weight sharing. The weight sharing improves the deblurring quality while reducing the number of parameters.

KPAC components			Baseline	PSNR (dB)	Parameters (M)
ACs	Scale	Shape			
			✓	24.59	2.26
✓				24.74	1.33
✓		✓		24.98	1.33
✓	✓			25.03	1.58
✓	✓	✓		25.21	1.58

Table 2. Ablation study. ACs: atrous convolution layers. Scale: scale attention. Shape: shape attention.

same, but their sizes vary. For effective learning of such inverse kernels, our network shares the convolution weights across multiple atrous convolution layers. In this experiment, we verify the effect of the weight sharing between atrous convolution layers by comparing the performance of models with and without the weight sharing. Both models have two KPAC blocks with 5×5 kernels. Table 1 shows the deblurring quality and the number of parameters of each model. As shown in the table, our model with the weight sharing not only reduces the number of learning parameters, but also improves the deblurring quality, as its weight sharing structure properly constrains and guides the learning process.

Scale attention The atrous convolution layers in our KPAC block simulate inverse kernels of different sizes to effectively handle the spatially varying nature of defocus blur. To analyze how they are activated for defocus blur with different sizes, we visualize the scale attention maps of different atrous convolution layers (Fig. 6). The roles of different attention maps may not be strictly distinguished because of the nature of the learning process that implicitly learns the use of different layers. Nevertheless, we can observe a clear tendency that the attention maps of different dilation rates are activated for different blur sizes. For example, the attention map of the dilation rate 1 is activated for pixels with blur of almost any size. On the other hand, the attention map of the dilation rate 5 is activated only for pixels with large blur. This shows that our scheme properly works for handling spatially varying size of defocus blur.

Ablation study To quantitatively analyze the effect of each component in our KPAC block, we conduct an ab-

lation study (Table 2). We first prepare a baseline model, which uses naïve convolution blocks instead of our KPAC blocks. For the baseline model, we use a conventional residual block that consists of two convolution layers with the filter size of 3×3. For a fair comparison, the baseline model includes multiple convolution blocks so that its model size is similar to our model without the weight sharing. We also prepare four variants of the baseline model using two KPAC blocks with the kernel size of 5×5, then measure the deblurring performances of the models. Table 2 summarizes the ablation study result. As shown in the table, every component of our proposed approach increases the deblurring quality significantly. Fig. 7 presents a qualitative comparison, which shows that both scale attention and shape attention help our network better handle spatially varying blur and restore fine structures while the models with no attentions suffer from spatially varying blurs.

Atrous convolutions with different dilation rates We analyze the effect of dilation rates of atrous convolution layers in handling defocus blur with varying scales. In the test, we manually modulated the attention weight maps $[\alpha_{1\sim5}]$ to make our pretrained network use only the feature maps produced by atrous convolution layers of specified scales. With the similar tendency to Fig. 6, we observed that atrous convolutions with dilation rates of 1, 3 and 5 help remove blur of any, medium, and large sizes, respectively (Fig. 8).
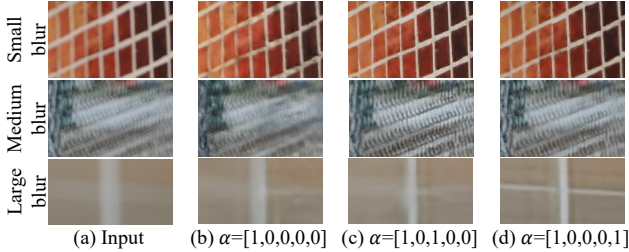
(a) Input   (b) α=[1,0,0,0,0]   (c) α=[1,0,1,0,0]   (d) α=[1,0,0,0,1]

Figure 8. Deblurring results with modulated scale attention maps $[\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5]$.

	Blurry input	Number of KPAC blocks			
		1	2	3	4
PSNR (dB)	23.92	24.82	25.21	25.25	25.14
SSIM (dB)	0.812	0.836	0.842	0.842	0.841
Params (M)	-	1.05	1.58	2.11	2.64

Table 3. Comparison on the effect of different numbers of KPAC blocks with 5×5 kernels.

Number of KPAC blocks Our KPAC block can be stacked together so that the network can iteratively remove defocus blur to achieve higher-quality deblurring results. We investigate the performance of different numbers of KPAC blocks. Table 3 shows that even a single KPAC block can effectively remove defocus blur and increase the PNSR by 0.90 dB. As we adopt more KPAC blocks, the PSNR increases although the improvement becomes smaller. After three KPAC blocks, the PSNR starts to decrease possibly due to the increased training complexity. Based on this experiment, we design our final model to have two KPAC blocks as two blocks provide relatively high deblurring quality with small model size.

5.2. Evaluation

Deblurring quality comparison with state-of-the-art We compare our method with state-of-the-art defocus deblurring methods, including both conventional two-step approaches [23, 11, 14] and the recent end-to-end deep learning-based approach [1]. For all the methods, we produced result images using the source code provided by the authors. For JNB [23], EBDB [11] and DMENet [14], we used the non-blind deconvolution method [12] for generating deblurred images using estimated defocus maps. For DPDNet [1], we used the source code and pre-trained models provided by the author. DPDNet provides two versions of models, each of which takes a single input image and dual-pixel data, which is a pair of sub-aperture images, respectively. We include both of them in our comparison. For evaluation, we measure PSNR and SSIM [26]. We also measure LPIPS [32] for evaluating the perceptual quality as done in [1].

We include two variants of our model, each of which has a different number of encoding levels, or a different number of downsampling layers in the encoder. By increasing the

(a) Input   (b) 2-level   (c) 3-level   (d) GT

Figure 9. Visual comparison between our 2- and 3-level models.

Model	PSNR↑	SSIM↑	LPIPS↓	Parameters (M)
JNB [23]	23.70	0.799	0.442	-
EBDB [11]	23.96	0.819	0.402	-
DMENet [14]	23.92	0.808	0.410	26.94
DPDNet (single) [1]	24.42	0.827	0.277	32.25
DPDNet (dual) [1]	25.12	**0.850**	**0.223**	32.25
Ours (2-level)	**25.24**	0.845	0.229	1.58
Ours (3-level)	**25.24**	0.842	0.225	2.06

Table 4. Quantitative comparison. The numbers of parameters of JNB and EBDB are not available as they are not deep learning-based methods. Note that DPDNet (dual) uses dual-pixel images.

Model	FLOPs (B)	running time (s)	Parameters (M)
DPDNet (single) [1]	1980	0.17	32.25
DPDNet (dual) [1]	1983	0.17	32.25
Ours (2-level)	358	0.09	1.58
Ours (3-level)	197	0.07	2.06

Table 5. Computational cost comparison. The average FLOPs and running times are measured on images of size 1280×720.

encoding levels, we can more easily handle large blur with small filters and with a small amount of computations. On the other hand, with fewer encoding levels, it is easier to restore fine-scale details. To inspect the difference between models with different numbers of encoding levels, we include two variants of our model, which have two and three levels, respectively. Both models are trained with both MAE and perceptual loss functions.

Table 4 reports the quantitative comparison. As shown in the table, the classical two-step approaches [23, 11, 14] perform worse than the recent deep-learning based approach [1]. While the DPDNet model with a single input image performs better than the classical approaches, our models outperform both the classical approaches and the DPDNet model with a single input image by a large margin. Moreover, our models outperform the dual-pixel-based DPDNet model even without the strong cue to defocus blur provided by dual-pixel data and with a much small number of parameters. This result clearly proves the effectiveness of our approach. In the supplementary material, we report the performance of the dual-pixel-based variant of our model, which outperforms the dual-pixel-based DPDNet model. Table 4 also shows that our 2- and 3-level models perform similarly. However, we found that the 3-level model tends to better handle extremely large blur as shown in Fig. 9 due to its larger receptive fields.

Fig. 10 shows a qualitative comparison. Our result is produced by the 3-level model. As the figure shows, our method produces sharper results with more details. Even compared to the result of the dual-pixel-based DPDNet [1],
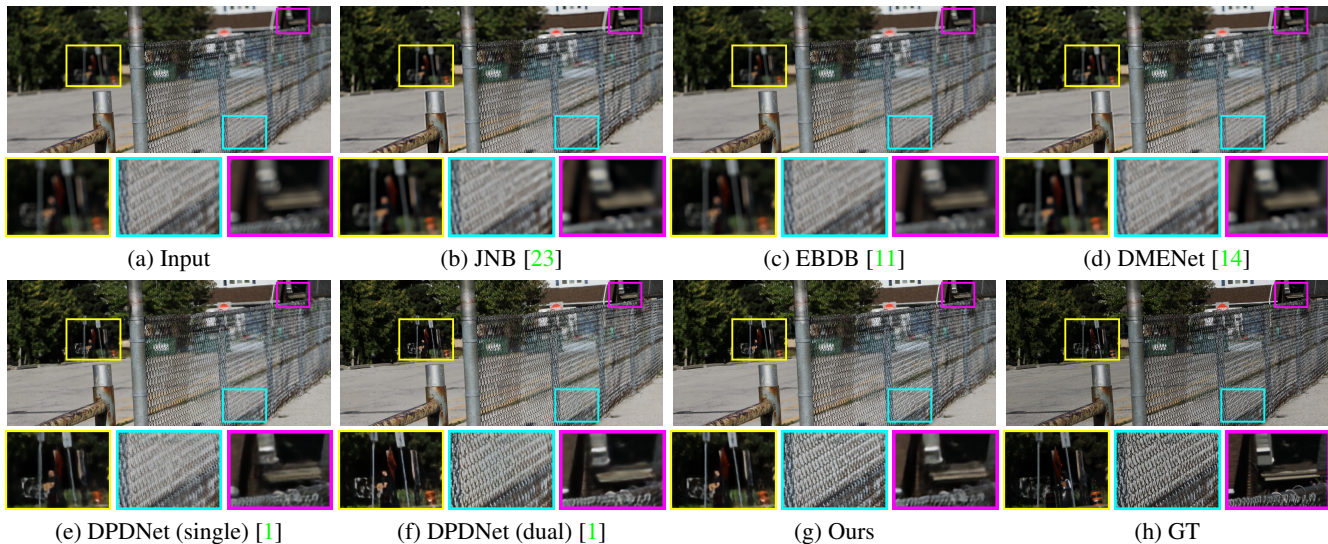
(a) Input   (b) JNB [23]   (c) EBDB [11]   (d) DMENet [14]

(e) DPDNet (single) [1]   (f) DPDNet (dual) [1]   (g) Ours   (h) GT

Figure 10. Qualitative comparison on the test set of the DPDD dataset [1]. Note that DPDNet (dual) uses dual-pixel stereo images.

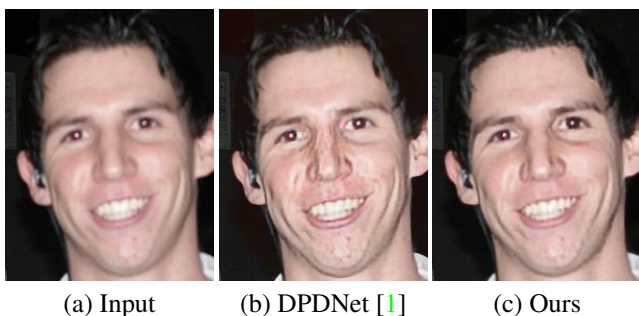(a) Input   (b) DPDNet [1]   (c) Ours

Figure 11. Defocus deblurring results on the defocused images in the CUHK blur detection dataset [22].

our result has comparably clear details.

Computational cost   We compare the computational cost of our models and DPDNet [1]. Classical two-step approaches rely on computationally heavy non-blind deconvolution algorithms, so we do not include them in this comparison. For the comparison, we measure FLOPs and the average running time per image of size 1280×720. Table 5 shows that our 3-level model requires small computational cost in FLOPs, which is 10 times smaller than [1]. The table also shows that our 3-level model is slightly faster than the 2-level model as features are more downsampled, even though it has more parameters.

Generalization to other images   Our models are trained using the DPDD dataset [1], which was generated using one camera. Thus, lastly, we inspect how well our model generalizes to images from other cameras. To this end, we use the CUHK blur detection dataset [22], which provides 704 defocused images without ground-truth all-in-focus images. The defocused images in the dataset are collected from various sources on the internet. As there are no ground-truth images, we qualitatively inspect the generalization ability. Fig. 11 shows that our method can successfully restore

fine details with less visual artifacts compared to the single image-based model of DPDNet [1]. Additional results can be found in the supplementary material.

6. Conclusion

This paper proposed a single image defocus deblurring framework based on inverse kernels. To effectively simulate spatially varying inverse kernels, we proposed the Kernel-Sharing Parallel Atrous Convolutions (KPAC) block. KPAC provides an effective way to handle spatially varying defocus blur with a small number of atrous convolution layers that share the same convolutional kernel weights but with different dilation rates. KPAC is also equipped with the per-pixel scale attention to further facilitate the handling of spatially varying blur. Thanks to the effective and light-weight structure of KPAC, we can simply stack multiple blocks of KPAC and achieve state-of-the-art deblurring performance. We experimentally validated the effectiveness of KPAC and showed that our method clearly outperforms previous methods with much fewer parameters.

Limitations and future work   While our method outperforms previous state-of-the-art methods, it may still fail in challenging cases such as large-scale blur, blur with irregular shapes, and bokeh with sharp boundaries (refer to the supplementary material for our deblurring results on images containing such cases). Handling these challenging cases would be an interesting future direction.

References

[1] A. Abuolaim and M.S. Brown. Defocus deblurring using dual-pixel data. In *Proc. ECCV*, 2020. 1, 2, 5, 7, 8

[2] Y. Bando and T. Nishita. Towards digital refocusing from a single photograph. In *Proc. Pacific Graphics*, 2007. 1, 2, 3

[3] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)*, 40(4):834–848, 2018. 2

[4] S. Cho and S. Lee. Convergence analysis of MAP based blur kernel estimation. In *Proc. ICCV*, pages 4818–4826, 2017. 1, 2

[5] 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. 1, 2

[6] J. Dong, S. Roth, and B. Schiele. Deep wiener deconvolution: Wiener meets deep learning for image deblurring. In *Proc. NeurIPS*, 2020. 2, 5

[7] Claude E. Duchon. Lanczos filtering in one and two dimensions. *Journal of Applied Meteorology and Climatology*, 18(8):1016–1022, 1979. 3

[8] D. A. Fish, A. M. Brinicombe, E. R. Pike, and J. G. Walker. Blind deconvolution by means of the richardson-lucy algorithm. *Journal of the Optical Society of America A (JOSA A)*, 12(1):58–65, 1995. 1

[9] M. Holschneider, R. Kronland-Martinet, J. Morlet, and Ph. Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Proc. Wavelets*, 1990. 2

[10] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proc. ECCV*, 2016. 5

[11] A. Karaali and C. Jung. Edge-based defocus blur estimation with adaptive scale selection. *IEEE Trans. Image Processing (TIP)*, 27(3):1126–1137, 2018. 1, 2, 7, 8

[12] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. In *Proc. NIPS*, 2009. 1, 2, 7

[13] Guang-He Lee, David Alvarez-Melis, and Tommi S. Jaakkola. Towards robust, locally linear deep networks. In *Proc. ICLR*, 2019. 5

[14] J. Lee, S. Lee, S. Cho, and S. Lee. Deep defocus map estimation using domain adaptation. In *Proc. CVPR*, 2019. 1, 2, 7, 8

[15] A. Levin, R. Fergus, F. Durand, and W. Freeman. Image and depth from a conventional camera with a coded aperture. *ACM Trans. Graphics (TOG)*, 26:70, 2007. 1, 2

[16] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, 2013. 5

[17] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Proc. NeurIPS*, 2014. 5

[18] 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. 1, 2

[19] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. De-Vito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *Proc. NIPSW*, 2017. 5

[20] Wenqi Ren, Jiawei Zhang, Lin Ma, Jinshan Pan, Xiaochun Cao, Wangmeng Zuo, Wei Liu, and Ming-Hsuan Yang. Deep non-blind deconvolution via generalized low-rank approximation. In *Proc. NeurIPS*, 2018. 2, 3

[21] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI*, 2015. 5

[22] J. Shi, L. Xu, and J. Jia. Discriminative blur detection features. In *Proc. CVPR*, 2014. 8

[23] J. Shi, L. Xu, and J. Jia. Just noticeable defocus blur detection and estimation. In *Proc. CVPR*, 2015. 1, 2, 7, 8

[24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. ICLR*, 2015. 5

[25] H. Son and S. Lee. Fast non-blind deconvolution via regularized residual networks with long/short skip-connections. In *Proc. ICCP*, 2017. 2

[26] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Processing (TIP)*, 13(4):600–612, 2004. 7

[27] Norbert Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. The MIT Press, 1964. 2, 3

[28] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural image caption generation with visual attention. In *Proc. ICML*, 2015. 2, 4

[29] Li Xu, Jimmy SJ Ren, Ce Liu, and Jiaya Jia. Deep convolutional neural network for image deconvolution. In *Proc. NeurIPS*, 2014. 1, 2, 3

[30] L. Xu, X. Tao, and J. Jia. Inverse kernels for fast spatial deconvolution. In *Proc. ECCV*, 2014. 1, 2

[31] Lu Yuan, Jian Sun, Long Quan, and Heung-Yeung Shum. Progressive inter-scale and intra-scale non-blind image deconvolution. *ACM Transactions on Graphics*, 27(3):1–10, 2008. 2

[32] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. CVPR*, 2018. 7

[33] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Proc. ECCV*, 2018. 2, 5

[34] S. Zhou, J. Zhang, J. Pan, H. Xie, W. Zuo, and J. Ren. Spatio-temporal filter adaptive network for video deblurring. In *Proc. ICCV*, 2019. 1