Deep Single Image Defocus Deblurring via Gaussian Kernel Mixture Learning

Yuhui Quan, Zicong Wu, Ruotao Xu, Hui Ji

Abstract—This paper proposes an end-to-end deep learning approach for removing defocus blur from a single defocused image. Defocus blur is a common issue in digital photography that poses a challenge due to its spatially-varying and large blurring effect. The proposed approach addresses this challenge by employing a pixel-wise Gaussian kernel mixture (GKM) model to accurately yet compactly parameterize spatially-varying defocus point spread functions (PSFs), which is motivated by the isotropy in defocus PSFs. We further propose a grouped GKM (GGKM) model that decouples the coefficients in GKM, so as to improve the modeling accuracy with an economic manner. Afterward, a deep neural network called GGKMNet is then developed by unrolling a fixed-point iteration process of GGKM-based image deblurring, which avoids the efficiency issues in existing unrolling DNNs. Using a lightweight scale-recurrent architecture with a coarse-to-fine estimation scheme to predict the coefficients in GGKM, the GGKMNet can efficiently recover an all-in-focus image from a defocused one. Such advantages are demonstrated with extensive experiments on five benchmark datasets, where the GGKMNet outperforms existing defocus deblurring methods in restoration quality, as well as showing advantages in terms of model complexity and computational efficiency.

Index Terms—Defocus Deblurring, Gaussian Kernel Mixture, Fixed-Point Iteration, Deep Unrolling Networks, Image Deblurring

1 Introduction

The sharpness of an object in an image captured by a regular optical camera depends on the object's distance from the camera's focal plane. Objects close to the focal plane are in focus and appear sharp, and the area around the focal plane where objects appear in focus is known as the depth of field (DoF). Objects outside the DoF appear blurry, with the level of blurriness increasing as the object moves further from the DoF. This phenomenon is known as defocus blur, and it is particularly noticeable in images captured with a shallow DoF, such as those taken with a large aperture. Defocus blur can erase important object details, making it necessary to remove for consequent vision tasks. This paper addresses single image defocus deblurring (SIDD), which aims to generate an all-in-focus image from a defocus-blurred image with multiple out-of-focus regions. SIDD is useful in various machine vision applications, such as photo refocusing, tracking, and object recognition.

We consider a defocused image $\boldsymbol{y}$ and its corresponding all-in-focus image $\boldsymbol{x}$, related by the following model:

$$\boldsymbol{y} = \boldsymbol{B} \circ \boldsymbol{x} + \boldsymbol{\epsilon}, \tag{1}$$

where $\boldsymbol{\epsilon}$ represents measurement noise, and $\boldsymbol{B}$ is a linear blurring operator defined as

$$(\boldsymbol{B} \circ \boldsymbol{x})[m,n] \equiv \sum_i \sum_j \boldsymbol{b}_{m,n}[i,j]\boldsymbol{x}[m-i, n-j], \tag{2}$$

with $[m,n]$ denoting the pixel location. That is, each pixel at $[m,n]$ is associated with a defocus blur kernel $\boldsymbol{b}_{m,n}$. These kernels, also called point spread functions (PSFs), represent the blur created by imaging a perfect point with the same distance to the focal plane. Since scene depth varies in most images, the defocus PSFs in an image typically vary across different spatial locations.

Defocus PSFs in a pin-hole camera are isotropic. Most existing works parameterize pixel-wise PSFs by isotropic Gaussian kernelswith unknown standard deviation (*e.g.* [1], [2], [3], [4], [5]) or disk kernels with unknown radius (*e.g.* [6], [7]). For instance, using isotropic Gaussian kernels

$$\boldsymbol{g}(\sigma)[i,j] = \exp(-\frac{i^2+j^2}{\sigma^2}), \tag{3}$$

where σ represents the standard deviation, one can define

$$\boldsymbol{b}_{m,n} = \boldsymbol{g}(\sigma_{m,n}), \tag{4}$$

with a spatially-varying parameter $\sigma_{m,n}$. These single-parametric models are oversimplified, and their kernel sizes cannot be set to large values as this can result in excessive computational cost of $\boldsymbol{B}$. While a more complex parametric form with multiple unknown parameters can better represent real-world defocus PSFs, the resulting model can be more expensive to work with; see *e.g.* [8].

To obtain an all-in-focus image $\boldsymbol{x}$ from (1), it is necessary to estimate both the image $\boldsymbol{x}$ and the blurring operator $\boldsymbol{B}$ composed of pixel-wise parametric defocus PSFs. Most existing methods take a two-stage approach; see *e.g.* [3], [5], [6], [9]. The first stage estimates a dense defocus map indicating pixel-wise blur degrees to derive the operator $\boldsymbol{B}$, while the second stage recovers the image $\boldsymbol{x}$ using an existing robust non-blind deblurring method (*e.g.* [10], [11], [12], [13], [14]) with the estimated $\boldsymbol{B}$. However, such an approach has a long pipeline with many modules which may amplify

- *Y. Quan, Z. Wu, and R. Xu are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China. E-mail: csyhquan@scut.edu.cn (Y. Quan); cszicongwu@mail.scut.edu.cn (Z. Wu); xrt@scut.edu.cn (R. Xu).*
- *H. Ji is with the Department of Mathematics, National University of Singapore, Singapore. E-mail: matjh@nus.edu.sg (H. Ji).*
- *This work was supported in part by National Natural Science Foundation of China under Grant 62372186, and in part by Singapore MOE Academic Research Fund under Tier 1 Research Grant R-146000-315-114.*

the errors from PSF estimation noticeably. Additionally, the second stage, which needs to call the inversion process of $\boldsymbol{B}$ for many times, is computationally expensive for non-uniform blurring operators, making it less practical than for uniform blurring operators that can be efficiently computed via Fast Fourier transform (FFT).

In recent years, there has been a significant interest in using learned deep neural network (DNN) models for SIDD. Following the discussed two-stage approach, one can replace the handcrafted defocus map estimator by a learned DNN-based one; see *e.g.* [5]. Unfortunately, the issues of error magnification and high computational cost of the two-stage approach remains. An alternative is using end-to-end learning (*e.g.* [15], [16], [17], [18], [19], [20], [21], [22], [23], [24]). Most existing end-to-end learning methods for SIDD focus on introducing spatially-vary operations into the DNN to better handle uniform defocus blur (*e.g.* [16], [17], [21], [22]) or auxiliary tasks using additional source data (*e.g.* [16], [18], [20], [23], [24]). Nevertheless, there is still room for improvement in their performance and efficiency by leveraging the characteristics of defocus PSFs.

This paper presents a physics-aware end-to-end DNN for SIDD that takes advantage of isotropy, one known property of defocus PSFs. While the commonly-used single Gaussian or disk parametric form has encoded such a property, they oversimplify the shape of a defocus PSF. To address this, we propose a Gaussian Kernel Mixture (GKM) formulation that more accurately models real-world defocus PSFs while still leveraging their isotropy. The GKM model expresses a defocus PSF by a linear combination of multiple predefined isotopic Gaussian kernels with varying sizes and standard deviations, where spatially-varying combination coefficients are used to represent pixel-wise defocus PSFs. As the linear combination of isotropic Gaussian kernels can have a non-Gaussian shape while being isotropic, the GKM model can represent a larger class of defocus PSFs with strong isotropy, compared to single Gaussian model. In addition, it leads to a computationally efficient form of the defocus blurring operator to work with.

The expressivity of the GKM model relies on the number of Gaussian kernels employed. While increasing the number of Gaussian kernels can improve the accuracy and coverage for real-world defocus PSFs, it incurs both possible overfitting due to the increased freedom degree and substantial computational cost caused by the estimation of extra combination coefficients. To win the trade-off, we introduce a grouping strategy which clusters Gaussian kernels to create an image-adaptive non-Gaussian kernel basis, for the linear representation of pixel-wise defocus PSFs using spatially-varying coefficients. The resulting new model, termed as Grouped Gaussian Kernel Mixture (GGKM), presents a decoupled form of the GKM model. It decomposes the combination coefficients into two parts: 1) a spatially-varying part related to scene depth, in the form of coefficient matrices, analogous to those in the original GKM; and 2) an image-adaptive spatially-consistent part linked to the characteristics of the acquisition device, represented as coefficient vectors, serving as an extra component to predict. Crucially, the additional coefficient vectors are significantly smaller than the coefficient matrices. This feature of the GGKM model enables the inclusion of more Gaussian kernels for improved representation accuracy, while introducing only a minimal number of additional coefficients for prediction.

To improve computational efficiency when handling the spatially-varying blurring operator $\boldsymbol{B}$, we develop an unrolling DNN based on a fixed-point iteration scheme for deblurring. A coarse-to-fine estimation scheme is also included into the unrolling DNN architecture for better performance and faster inference. Different from widely-used unrolling DNNs, the proposed one does not require the inversion of the blurring operator, making it more efficient for non-uniform deblurring. The resulting DNN consists of two key modules for convolutions with Gaussian kernels and predicting the combination coefficients of Gaussian kernels, respectively. The former is implemented with an iterative scheme for acceleration, while the latter is implemented by a memory-based scale-recurrent structure. Extensive experimental results indicate that our proposed DNN outperforms existing SIDD methods, and it has several advantages in both model complexity and computational efficiency.

See below for a summary of the technical contributions of this paper:

- A new GGKM-based parametric model of defocus PSFs is introduced, which fits real-world PSFs more accurately yet with low model complexity;
- A new unrolling DNN is derived from a fixed-point iterative scheme for handling the computational efficiency issue in non-uniform deblurring; and
- A coarse-to-fine progressive estimation scheme implemented by a memory-based scale-recurrent mechanism is introduced for our unrolling DNN for improvement.

This paper is an extension of its conference version [25], which includes not only more detailed discussion but also new materials, such as new ideas, components, and additional experiments. The new materials mainly include:

- We propose GGKM, a grouped GKM model that improves the expressivity of GKM, while with marginal complexity/cost introduced. The GGKM model results in a decoupled representation of defocus PSFs;
- The DNN has been improved according to the GGKM model, which include new modules for predicting the coefficients in GKM and an accelerated module for performing group Gaussian convolutions;
- A self-supervised loss function defined by an auxiliary task of self-reconstruction has been introduced to better keep details of in-focus regions of the input image;
- Two additional datasets have been used for performance evaluation, and additional experiments have been conducted to provide a more in-depth analysis.
- Experimental results demonstrate that our proposed DNN achieves higher deblurring accuracy than the previous version, while only slightly increasing model and computational complexity.

The rest of the paper is organized as follows. Section 2 provides a literature review. Section 3 describes the main idea behind our proposed DNN, and Section 4 presents the details on the DNN structure. Section 5 is for performance evaluation, and Section 6 provides additional experimental analysis. Finally, Section 7 concludes the paper.

2 Related Work

2.1 Two-Stage SIDD

Typically, traditional methods for SIDD involve a two-stage scheme, which first estimates a dense defocus map to derive the blurring operator and then applies non-blind image deblurring. However, defocus map estimation is a challenging task. Various methods have been proposed for it using hand-crafted features such as sparse codes of learned blurred atoms [1], maximum ranks of gradient-domain patches with different orientations [2], and re-blurred gradient magnitudes from a scale-consistent edge map [4]. DNNs such as convolutional neural networks (CNNs) have also been used to estimate features related to defocus blur. For instance, Park *et al.* [3] combine both hand-crafted and deep features for defocus map estimation.

A dense defocus map is also typically estimated by a two-stage process. First, a sparse defocus map is estimated on edge pixels, and the estimated sparse values are propagated to all pixels to obtain a dense defocus map using methods such as matting Laplacian [26], weighted average [4], or regression tree [6]. Nevertheless, this propagation process can introduce errors that affect the accuracy of estimation. Recently, deep learning has shown promise for defocus map estimation; see *e.g.* [3], [27], [28].

Despite the unavailability of accurate defocus map estimators, existing two-stage methods still face two challenges. First, deriving local PSFs from a defocus map is difficult as it only provides a rough estimate of the blur degree of each pixel, which is only sufficient for deriving simple parametric PSF models (*e.g.*, Gaussian or disk kernels) that are known to be oversimplified for real-world PSFs. Unfortunately, non-blind deblurring is sensitive to even small errors in PSFs. Second, the computational cost of deblurring with a non-uniform blurring operator is high, as it requires an iterative scheme where each iteration involves an inversion process. Unlike a uniform blurring operator (*i.e.* convolution) that can be efficiently inverted using FFT, there is no fast numerical scheme for inverting a non-uniform blurring operator.

Our proposed end-to-end DNN overcomes the limitations of two-stage methods in two ways. Firstly, it estimates more accurate models for real-world PSFs without relying on defocus map estimators. Secondly, it does not require the inversion of a non-uniform blurring operator during deblurring, leading to a computationally efficient process.

2.2 End-to-End Learning for Defocus Deblurring

There has been a recent surge in research focused on using end-to-end learning for defocus deblurring. The pioneering work can be traced back to Abuolaim and Brown [15] which trained a U-shaped CNN to predict an all-in-focus image from two view images captured by a dual-view sensor. While this model significantly outperforms the two-stage methods, it shows noticeable performance degradation when inputted with a single defocused image, as it lacks explicit mechanisms to handle spatially-varying defocus blur in single rather than dual-view images.

An increasing number of methods have been proposed which handle spatially-varying defocus blur through dynamic filtering, *i.e.*, introducing explicit spatially-varying operations into the CNN for SIDD. Lee *et al.* [16] predicted pixel-wise filters for deblurring deep defocused features. Ruan *et al.* [17] incorporated dynamic residual blocks into a multi-scale framework to restore the latent sharp image in a cascaded fashion. Zhang and Zhai [21] incorporated an attention disentanglement mechanism into a generative adversarial DNN to better treat blurry and clear regions separately. Wang *et al.* [22] proposed a U-shaped transformer for various image restoration tasks including SIDD. However, the predicted filters or attention maps lack constraints, limiting the generalization performance.

To exploit physical constraints from the defocus blurring model, Son *et al.* [19] use kernel-sharing parallel atrous convolutions to simulate invariant shapes of inverse defocus PSFs with different scales. In comparison to these methods, both our previous conference work [25] and this paper improve the generalization performance by exploiting the isotropy of defocus PSFs via the GKM/GGKM model for better constraints. The isotropy of defocus PSFs has also been leveraged in a recent study of Li *et al.* [29], which employed rotational replication of 1D predicted kernels for reblurring, but not for deblurring. This approach, coupled with an optical flow loss, can improve the robustness to misalignment in training data. Quan *et al.* [30] learned recursive kernel atoms to represent defocus PSFs with a blob shape and developed an unrolling DNN based on Neumann series. However, these learned kernel atoms may overfit the misalignments or blur characteristics of training data, limiting the performance on misalignment-free test data and unseen blur characteristics. In comparison, the proposed GGKM model not only mitigates possible risking via fixing inner-group atoms to Gaussian kernels, but also allows inter-group atoms to be adaptive in test time by predicting the combination coefficients of Gaussian kernel atoms on a test image. As result, our GGKMNet achieves better noise robustness and generalization performance, as suggested in our experiments.

The utilization of reblurring (self-reconstruction) loss is a common practice in training DNNs of SIDD, as evidenced in [29], [30], [31]. The approach in [31] employs a known spatially-invariant PSF for reblurring, while ours utilizes a CNN and has no need to access any PSF. The reblurring loss functions in [16], [30] are supervised, utilizing ground truths (GTs) as inputs for reblurring and being defined on the predicted deblurring filters. In contrast, ours is self-supervised without the need to access GTs, and it is defined based on extracted image features. The reblurring loss used in [29] serves a different purpose compared to ours. It is employed to prevent the deblurring DNN learning spatial deformation of mis-aligned data, utilizing a deblurred/blurred image pair as input. In contrast, ours employs the deblurred image and intermediate blur-related features as input, with the goal of regularizing the feature learning of PSF estimation.

There are some end-to-end methods using additional data sources for building auxiliary training tasks so as to improve the performance of SIDD. For instance, defocus maps are used in [18], [20], [23] and dual-view images are used in [16], [24]. In comparison to these methods, ours employs a reconstruction-based self-supervised auxiliary task on the blurred image itself, without requiring additional data sources.

2.3 DNNs for Non-Uniform Motion Deblurring

Deep learning has shown promise in solving blind image deblurring problems in dynamic scenes with moving objects. Examples include [32], [33], [34], [35], [36], [37], [38]. Many of these studies employ coarse-to-fine recurrent neural network (RNN) structures and attention mechanisms tailored for motion deblurring. In comparison, the coarse-to-fine structure of our DNN is derived from the multi-scale extension of the fixed-point iteration of GKM-based deblurring, with a scale-recurrent attention mechanism to predict the coefficients in GKM. In addition, existing DNNs for dynamic scene deblurring typically employ specialized techniques to exploit the specific properties of motion blur and moving objects, such as temporal cues from training data (*e.g.* [35], [36]), which are not applicable to handing defocus blur. Applying these DNNs to SIDD is sub-optimal and has limitations, as defocus blur differs much from motion blur. For instance, the PSF of defocus blur has roughly isotropic support while that of motion blur has highly curvy support, and defocus blur does not have the transparency effect of a moving object as motion blur does.

2.4 Unrolling DNNs for Image Deblurring

Optimization unrolling is a popular approach for designing DNNs for image recovery, such as image deblurring [14], [39], [40], [41], [42]. All these unrolling DNNs for image deblurring are designed for the uniform image deblurring, where the inversion of the uniform blurring operator (convolution) can be efficiently computed via FFT. They are not suitable for non-uniform image deblurring like SIDD, as there is no known fast numerical scheme for the inversion of a non-uniform blurring operator. Our proposed approach unrolls a fixed-point iterative scheme that does not require inversion of the blurring operator. Together with the GKM-based parametric model of defocus PSFs, it results in a computationally-efficient unrolling DNN for SIDD.

3 Main Idea of Proposed Approach

3.1 GKM Model for Defocus PSFs

A defocus PSF typically exhibits a convex shape with strong isotropy. Our proposed GKM model exploits this physical characteristic to offer an efficient parametric form for defocus PSFs, which expresses the defocus PSF $\boldsymbol{b}_{m,n}$ as the linear combination of predefined Gaussian kernels as follows:

$$\boldsymbol{b}_{m,n} = \sum_{k=1}^{K} \boldsymbol{\gamma}_k[m,n]\boldsymbol{g}\left(\sigma_k\right), \tag{5}$$

where $\boldsymbol{\gamma}_k$ denotes the matrix of mixing coefficients for the k-th Gaussian kernel $\boldsymbol{g}(\sigma_k)$. See Fig. 1 (left part) for an illustration of the GKM model.

The GKM model enjoys several benefits. Firstly, it can represent a large class of non-Gaussian isotropic kernels by using a weighted summation of isotropic Gaussian kernels, which fits real-world defocus PSFs more accurately than single-parametric Gaussian or disk kernels. Indeed, the GKM model simplifies to a single-parameter Gaussian kernel when $K = 1$. Secondly, the linear and simpler form of the GKM model makes it less challenging and less expensive to work with than existing complex non-linear parametric forms, such as the generalized Gaussian function [8]. Thirdly, the GKM model reduces the dimension of parameter space from $\mathcal{O}(S^2)$ to $\mathcal{O}(K)$ when compared to predicting free-form pixel-wise defocus PSFs of size $S \times S$. This dimension reduction is significant, especially for spatially-varying defocus PSFs of wide support.

3.2 GGKM: Improving GKM Model via Grouping

The choice for the total number of Gaussian kernels (*i.e.* K) is crucial for the expressivity of the GKM model. Depending on the characteristics of aperture and lens, isotropic defocus PSFs can have complex shapes that require combining a large number of Gaussian kernels for accurate representation. Nevertheless, selecting a large value for K can cause excessive complexity of model and computation, *e.g.*, there are K coefficient matrices to predict accordingly.

In order to achieve higher expressivity while minimizing additional cost, we utilize a grouping strategy for the GKM model. We begin by constructing P groups of Gaussian kernels, each consisting of Q Gaussian kernels with the same size, but different kernel sizes among groups. Let

$$\boldsymbol{g}_{p,q} = \boldsymbol{g}(\sigma_{p,q}) \in \mathbb{R}_{+}^{S_p \times S_p} \tag{6}$$

denote the q-th Gaussian kernel of the p-th group, with the standard deviation $\sigma_{p,q}$, for $p = 1, \cdots, P$, $q = 1, \cdots, Q$. The kernel size S_p increases along p, which fits the need to represent varying-size defocus PSFs and depict different blur degrees. We then define the GGKM model as follows:

$$\boldsymbol{b}_{m,n} = \sum_{p=1}^{P} \boldsymbol{\beta}_p[m,n]\tilde{\boldsymbol{g}}_p, \text{ with } \tilde{\boldsymbol{g}}_p = \sum_{q=1}^{Q} \boldsymbol{\alpha}_p[q]\boldsymbol{g}_{p,q}, \tag{7}$$

where $\boldsymbol{\alpha}_p$ is the vector of mixing coefficients for the inner-group Gaussian kernels $\boldsymbol{g}_{p,q}$ and $\boldsymbol{\beta}_p$ the matrix of mixing coefficients for the composite kernels $\tilde{\boldsymbol{g}}_p$. That is, the Gaussian kernels in each group are first linearly combined to compose non-Gaussian kernels $\tilde{\boldsymbol{g}}_p$, and then the composite kernels are used as the basis to represent the pixel-wise defocus PSFs. See Fig. 1 (right part) for an illustration.

The GGKM model (7) is indeed a decoupled form of the GKM model. We can rewrite it as

$$\boldsymbol{b}_{m,n} = \sum_{p=1}^{P}\sum_{q=1}^{Q} \boldsymbol{\alpha}_p[q]\boldsymbol{\beta}_p[m,n]\boldsymbol{g}_{p,q}. \tag{8}$$

A defocus PSF is mainly determined by two factors: the scene depth of corresponding pixel and the physical characteristics of the aperture and lens in the acquisition device. For the defocused pixels in the same image, the scene depth varies spatially while the acquisition device is the same. Accordingly, the coefficients of the Gaussian kernel $\boldsymbol{g}_{p,q}$ are decoupled into a spatially-varying part $\boldsymbol{\beta}_p[m,n]$ related to the scene depth and a spatially-invariant part $\boldsymbol{\alpha}_p[q]$ related to the acquisition device. As kernel sizes are the same in each group, the scene depth-related coefficients $\boldsymbol{\beta}_p[m,n]$ are set to be independent of q. Such a decoupling scheme is economic, which allows using PQ Gaussian kernels for defocus PSF representation while introducing only P coefficient vectors as the additional parameters to predict.

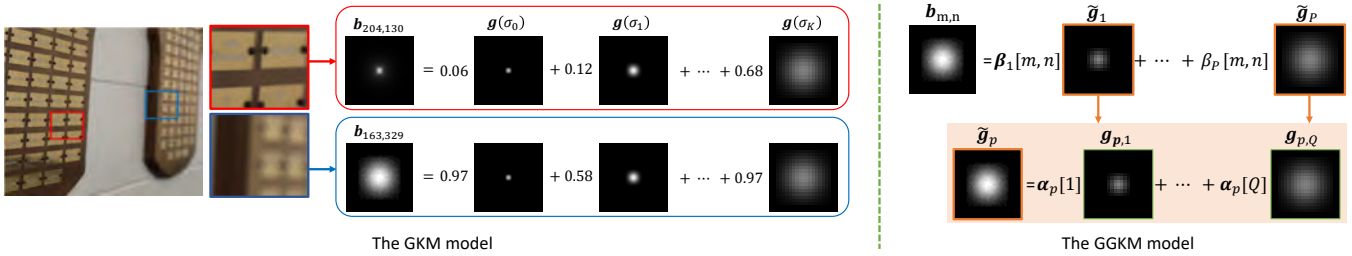


Fig. 1. Illustration of GKM model and GGKM model.

In addition, from the perspective that the kernel basis $\tilde{\boldsymbol{g}}_p$ is constructed using predicted mixing coefficients $\boldsymbol{\alpha}_p$ on each test image, the GGKM model has better adaptivity to specific images, compared to the GKM model.

3.3 Fixed-Point Iteration Unrolling with GGKM Model

Let $\boldsymbol{\delta}_{m,n}$ denote the Dirac delta function centered at location $[m,n]$, and $\otimes$, $\odot$ denote the operations of 2D convolution and point-wise multiplication respectively. Based on the GGKM model (7), we rewrite (1) by

$$\begin{aligned}\boldsymbol{B}\circ\boldsymbol{x} &= \sum_{m,n}\boldsymbol{\delta}_{m,n}\odot(\boldsymbol{b}_{m,n}\otimes\boldsymbol{x})\\ &= \sum_{m,n}\sum_{p=1}^{P}\boldsymbol{\delta}_{m,n}\odot((\boldsymbol{\beta}_p[m,n]\tilde{\boldsymbol{g}}_p)\otimes\boldsymbol{x})\\ &= \sum_{p=1}^{P}\boldsymbol{\beta}_p\odot((\sum_{q=1}^{Q}\boldsymbol{\alpha}_p[q]\boldsymbol{g}_{p,q})\otimes\boldsymbol{x}).\end{aligned} \tag{9}$$

The spatially-varying defocus blurring operator is now represented using only point-wise weighted summations and a number of convolutions, which is computationally efficient.

Once the blurring operator $\boldsymbol{B}$ is estimated from (9), the problem becomes non-blind image deblurring to recover the latent image $\boldsymbol{x}$. Optimization unrolling is a widely-used approach to design a deblurring DNN with explicit exploitation of the blurring operator, but existing unrolling DNNs for non-blind image deblurring, such as [14], [39], [40], [41], [42], focus on processing images with uniform blur effect. The main difference among these methods is the choice of the iteration scheme for unrolling, such as gradient descent [40] or half quadratic splitting (HQS) [39]. Take HQS for example, it involves inverting $\boldsymbol{B}$ at each iteration for updating the latent image estimate. In the case of uniform blur, $\boldsymbol{B}$ degenerates to a convolution, which can be effectively computed via FFT. However, this is not the case for non-uniform blur, where the computational cost of inverting $\boldsymbol{B}$ can be overwhelming. While gradient descent-based methods avoid FFT-based iteration, they need to call $\boldsymbol{B}^\top$, the transpose of $\boldsymbol{B}$, in one or more inner iterations. It is non-trivial to compute $\boldsymbol{B}^\top$ for non-uniform blurring.

To address the difficulty, we propose to unroll a fixed-point iteration scheme for non-uniform deblurring, so that the resulting unrolling DNN only involves the call of $\boldsymbol{B}$. Recall that the function of the blurring operator $\boldsymbol{B}$ is keeping the low-frequency image components and attenuating the high-frequency ones. Let $\boldsymbol{I}$ denote the identity mapping. The mapping $\boldsymbol{I}-\boldsymbol{B}$ is then about attenuating the low-frequency image components and keeping the high-frequency ones. Neglecting the noise $\boldsymbol{\epsilon}$, we rewrite (1) by

$$\boldsymbol{x} = \boldsymbol{y} + (\boldsymbol{I}-\boldsymbol{B})\circ\boldsymbol{x}. \tag{10}$$

Incorporating the GKM-based blurring operator (9) and the fixed-point iteration (10), we have then an unrolling scheme for solving SIDD, which is given by

$$\begin{aligned}\boldsymbol{x}^{(t)} &= f(\boldsymbol{x}^{(t-1)}) = \boldsymbol{y}+\boldsymbol{x}^{(t-1)}-\boldsymbol{B}\circ\boldsymbol{x}^{(t-1)}\\ &= \boldsymbol{y}+\boldsymbol{x}^{(t-1)}-\sum_{p=1}^{P}\boldsymbol{\beta}_p\odot((\sum_{q=1}^{Q}\boldsymbol{\alpha}_p[q]\boldsymbol{g}_{p,q})\otimes\boldsymbol{x}^{(t-1)}),\end{aligned} \tag{11}$$

for $t=1,\ldots,T$. Without the need to calculate inversion or transpose of $\boldsymbol{B}$, the iteration in (11) is computationally efficient which only consists of convolutions and point-wise weighted summations.

It is worth mentioning that, the fixed-point iteration scheme (10) converges when $\boldsymbol{I}-\boldsymbol{B}$ is a contractive mapping, meaning the eigenvalues of $\boldsymbol{B}$ fall in $(0,1]$, which holds true when the defocus blur is uniform with a normalized Gaussian PSF, *i.e.*, the scene depths are constant in the view. While convergence is not guaranteed for more general blur, the scheme serves as the motivation for an unrolling DNN which has a limited number of iterations.

In summary, based on the GGKM model of defocus PSFs, we unroll the fixed-point iteration scheme (11) for solving SIDD with predicted coefficients in $\{\boldsymbol{\alpha}_1,\cdots,\boldsymbol{\alpha}_P\}$ and $\{\boldsymbol{\beta}_1,\cdots,\boldsymbol{\beta}_P\}$. The resulting process only involves forward passes of the blurring operator $\boldsymbol{B}$, which is computationally efficient. In the next section, we construct a DNN to utilize such a process for end-to-end SIDD.

4 Neural Network Design

4.1 Overview

Our proposed DNN for SIDD, named as Grouped Gaussian Kernel Mixture Network (GGKMNet), is implemented using the iteration (11) with a coarse-to-fine recurrent fashion.

Given an input image $\boldsymbol{y}$, we generate its multi-scale versions $\boldsymbol{y}^{(1)},\cdots,\boldsymbol{y}^{(T)}$ via bi-linear downsampling with factors $2^{T-1},\cdots,2^0$, respectively. Let $\boldsymbol{x}^{(0)}=\boldsymbol{y}^{(1)}$ and let $\uparrow_2$ denote the bi-linear upsampling operation by a factor of 2. Consider the following coarse-to-fine extension of (11):

$$\boldsymbol{x}^{(t)} = \boldsymbol{y}^{(t)}+\boldsymbol{x}^{(t-1)}_{\uparrow_2}-\sum_{p=1}^{P}\boldsymbol{\beta}_p^{(t)}\odot((\sum_{q=1}^{Q}\boldsymbol{\alpha}_p^{(t)}[q]\boldsymbol{g}_{p,q})\otimes\boldsymbol{x}^{(t-1)}_{\uparrow_2}), \tag{12}$$

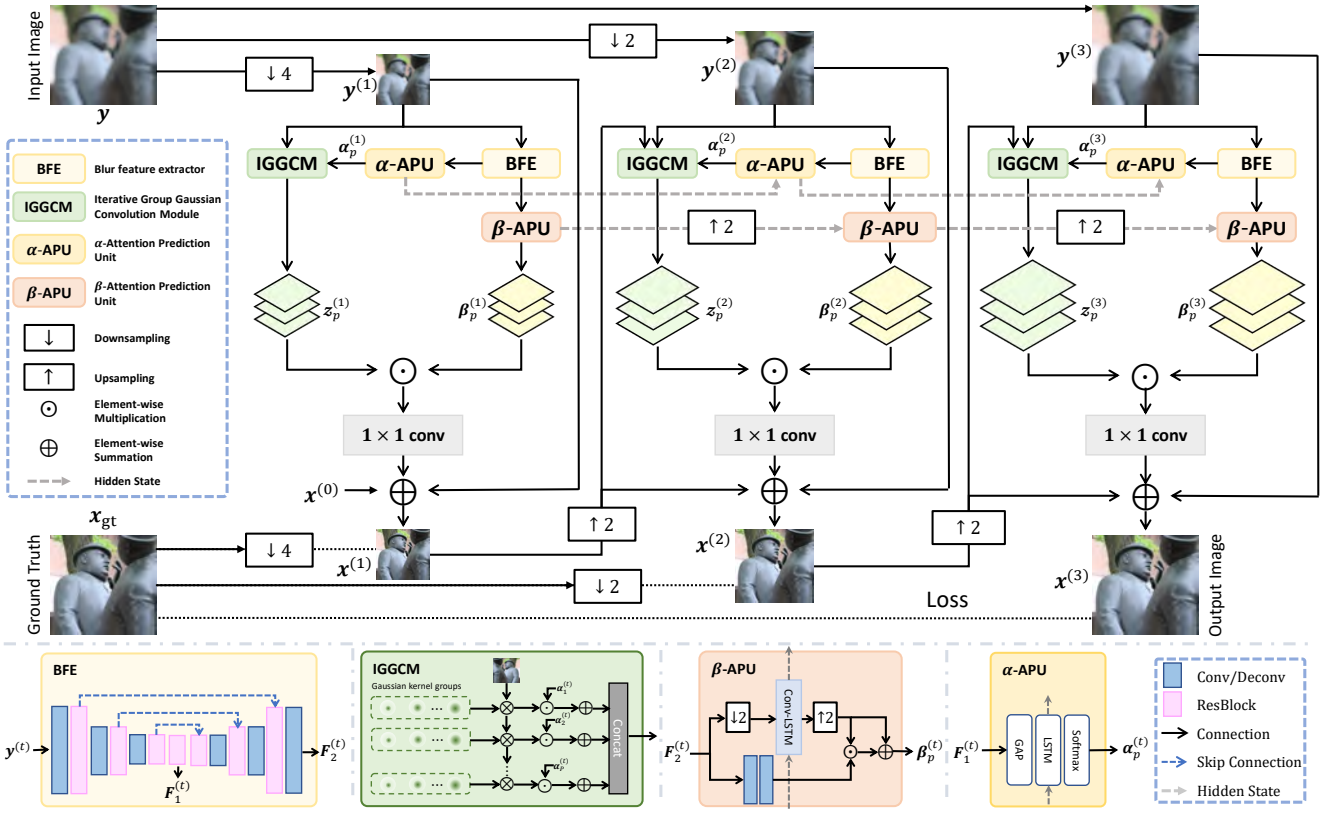


Fig. 2. Architecture of proposed GGKMNet for SIDD.

for $t = 1, \ldots, T$. For notational convenience, we define

$$\boldsymbol{z}_p^{(t)} = \tilde{\boldsymbol{g}}_p \otimes \boldsymbol{x}_{\uparrow 2}^{(t-1)}, \ \tilde{\boldsymbol{g}}_p = \sum_{q=1}^{Q} \boldsymbol{\alpha}_p^{(t)}[q] \boldsymbol{g}_{p,q} \quad p = 1, \cdots, P, \tag{13}$$

We then rewrite (12) as

$$\boldsymbol{x}^{(t)} = \boldsymbol{y}^{(t)} + \boldsymbol{x}_{\uparrow 2}^{(t-1)} - \sum_{p=1}^{P} \boldsymbol{\beta}_p^{(t)} \odot \boldsymbol{z}_p^{(t)}. \tag{14}$$

The GGKMNet uses T recurrent blocks to mimic (14), with the summation implemented by a 1×1 convolutional (Conv) layer. The output of the T-th block is used as the final output. Each recurrent block consists of two key modules: Mixing Coefficient Predictor (MCP) for predicting the mixing coefficients $\boldsymbol{\alpha}_p^{(t)}$ and $\boldsymbol{\beta}_p^{(t)}$, and Iterative Group Gaussian Convolution Module (IGGCM) for calculating the intermediate variables $\boldsymbol{z}_p^{(t)}$. In addition to $\boldsymbol{x}_{\uparrow 2}^{(t-1)}$, the IGGCM also fuses it with $\boldsymbol{y}^{(t)}$ as input, to achieve additional information. See Fig. 2 for a diagram of the GGKMNet.

Different from many existing unrolling DNNs designed for deblurring (*e.g.* [14], [39], [40], [41], [42]), the GGKMNet does not incorporate explicit denoising blocks for eliminating artifacts from intermediate image estimates. Nonetheless, the GGKMNet exhibits good performance and noise robustness in our experiments. The reasons are two-folded. First, the inversion process implemented by the GGKMNet utilizes learnable blocks, rather than the classic non-learnable estimators used in most existing works. This scheme allows incorporating implicit built-in artifact removal functions into the GGKMNet. Indeed, the combination of Gaussian kernels not only can express the defocus PSFs, but also may facilitate noise suppression (*e.g.*, Gaussian kernels themselves have certain denoising effects). This inherent noise suppression capability is achieved via supervised learning, where the GGKMNet is trained to match its predictions with noise-free GT images. Second, there is an implicit regularization effect due to the fact that the unrolled fixed-point iteration terminates after a predetermined number of steps, which is in the same way as early stopping that can induce regularization in optimization methods.

It is noted that, while in the GGKM model, the predicted coefficients $\boldsymbol{\alpha}_p^{(t)}$ and $\boldsymbol{\beta}_p^{(t)}$ appear to be strongly correlated to device and depth, respectively, but not to image content. However, for effective noise suppression and potential performance benefits from its adaptability to image content, the GGKM-inspired GGKMNet introduces moderate correlation to image content.

4.2 Mixing Coefficient Predictor

The MCP maps an input image $\boldsymbol{y}^{(t)}$ to the mixing coefficients $\boldsymbol{\alpha}_p^{(t)}$ and $\boldsymbol{\beta}_p^{(t)}$. The coefficient vectors $\boldsymbol{\alpha}_p^{(t)}$ can be viewed as the channel attention associated with the kernels $\boldsymbol{g}_{p,q}$, and the coefficient maps $\boldsymbol{\beta}_p^{(t)}$ can be viewed as the spatial-channel attention associated with the feature maps generated by the kernels $\tilde{\boldsymbol{g}}_p$. With the inspiration from existing attention modules [33], [43], [44], we adopt a lightweight design for MCP. As illustrated in Fig. 2, the MCP consists of a CNN-based Blur Feature Extractor (BFE) implemented by an encoder-decoder U-Net [45] for feature

extraction, and two RNN-based Attention Prediction Units, $\boldsymbol{\alpha}$-APU and $\boldsymbol{\beta}$-APU, implemented by long short-term memory (LSTM) [46] and convolutional LSTM (Conv-LSTM) [47] respectively for predicting the mixing coefficients $\boldsymbol{\alpha}_p$ and $\boldsymbol{\beta}_p$. The use of memory in APUs can effectively exploit feature dependencies among scales to improve the coarse-to-fine estimation. As both $\boldsymbol{\alpha}_p$ and $\boldsymbol{\beta}_p$ are highly related to low-level image features and blur features, their predictions share the common features extracted by the BFE for computational efficiency. The $\boldsymbol{\alpha}$-APU and $\boldsymbol{\beta}$-APU are run in parallel rather than sequentially connected, due to the consideration of computational efficiency. Indeed, with the shared features provided by BFE, $\boldsymbol{\alpha}_p$ and $\boldsymbol{z}_p$ do not provide much extra information to the prediction of each other.

Blur feature extractor The BFE comprises three encoder blocks and three decoder blocks, connected in a sequential manner. Each encoder block consists of a Conv layer and a residual block, where the latter consists of two Conv layers separated by a ReLU activation function. The first encoder block generates a 32-channel feature map, and each subsequent encoder block doubles the channel number and reduces the spatial dimensions via a Conv layer with a stride of 2. The last encoder block has the same structure as the first one. For the decoder blocks, each of them consists of a residual block and a transposed Conv layer which doubles the spatial dimensions and halves the channel number.

Attention prediction unit for $\boldsymbol{\alpha}_p$ The $\boldsymbol{\alpha}$-APU applies a series of operations to the features generated at the bottleneck (*i.e.* the fourth encoder block) of BFF, denoted by $\boldsymbol{F}_1^{(t)}$ in Fig. 2. First, global average pooling (GAP) is used to generate a global description, which is then input into two LSTM layers separated by a ReLU activation function. The first LSTM layer reduces the feature dimension by a factor of 4, while the second layer expands it to PQ. To capture blur from different scales and improve restoration across scales, we link the hidden units of the LSTM layers across the scale dimension t. Finally, we use a Softmax function to normalize the predicted $\boldsymbol{\alpha}_p$ group-wise.

Attention prediction unit for predicting $\boldsymbol{\beta}_p$ There are two paths in the $\boldsymbol{\beta}$-APU, which predict two intermediate variables $\bar{\boldsymbol{\beta}}_1$ and $\bar{\boldsymbol{\beta}}_2$ respectively. The first path mainly contains a scale-recurrent Conv-LSTM layer whose hidden state recurrently captures the dependencies among the blur degrees in the same location at different scales. This enables the model to progressively improve its estimation of the mixing coefficients. It also applies downsampling/upsampling before/after it for inducing local smoothness on the predicted $\bar{\boldsymbol{\beta}}_1$, given that the blur degrees of adjacent regions tend to be the same. The second path contains two Conv layers but without any downsampling operation so as to preserve details in the prediction of $\bar{\boldsymbol{\beta}}_2$. Finally, the $\boldsymbol{\beta}$-APU combines the two intermediate variables using the expression $\tanh(\bar{\boldsymbol{\beta}}_1 \odot \bar{\boldsymbol{\beta}}_2 + \bar{\boldsymbol{\beta}}_1)$ to produce the output.

4.3 Iterative Group Gaussian Convolution Module

The IGGCM produces $\boldsymbol{z}_p^{(t)}$ as output. The calculation of $\boldsymbol{z}_p^{(t)}$ via directly convolving the input with the group kernel $\tilde{\boldsymbol{g}}_p$ can be computationally expensive when $\tilde{\boldsymbol{g}}_p$ is large. Exploiting the properties of Gaussian convolution, we use an iterative scheme for acceleration. Recall that the convolution of two Gaussian kernels, $\boldsymbol{g}(\sigma_1)$ and $\boldsymbol{g}(\sigma_2)$, results in a Gaussian kernel $\boldsymbol{g}(\sqrt{\sigma_1^2+\sigma_2^2})$. Given two group kernels $\tilde{\boldsymbol{g}}_{p_1}$ and $\tilde{\boldsymbol{g}}_{p_2}$, we have that their convolution

$$\tilde{\boldsymbol{g}}_{p_1} \otimes \tilde{\boldsymbol{g}}_{p_2} = \sum_{q_1,q_2=1}^{Q} \boldsymbol{\alpha}_{p_1}[q_1]\boldsymbol{\alpha}_{p_2}[q_2](\boldsymbol{g}_{p_1,q_1} \otimes \boldsymbol{g}_{p_2,q_2}), \quad (15)$$

is a larger group kernel associated with Gaussian kernel atoms $\boldsymbol{g}_{p_1,q_1} \otimes \boldsymbol{g}_{p_2,q_2}$. In other words, the convolution with a group kernel can be done by the convolution with smaller group kernels. Therefore, we construct the group kernels $\tilde{\boldsymbol{g}}_p$ iteratively:

$$\tilde{\boldsymbol{g}}_p = \tilde{\boldsymbol{g}}_{p-1} \otimes \bar{\boldsymbol{g}}_p, \quad p = 2, \cdots, P,\ q = 1, \cdots, Q, \quad (16)$$

where $\bar{\boldsymbol{g}}_p$ is a smaller group kernel. As a result, convolving with $\tilde{\boldsymbol{g}}_p$ is done using the convolution result with a smaller group kernel $\bar{\boldsymbol{g}}_p$. This iteration scheme can largely reduce the computational cost. In implementation, we define the Gaussian kernels for $\bar{\boldsymbol{g}}_p$ to have the size of $(2p-1)\times(2p-1)$ and stand deviations of $\frac{1}{2}(p+PQ)/PQ+q$ with $p>1$.

Depending on kernel sizes, a possible way for further acceleration is utilizing that $\boldsymbol{z}_p^{(t)} = \sum_{q=1}^{Q} \boldsymbol{\alpha}_p^{(t)}[q](\boldsymbol{g}_{p,q} \otimes \boldsymbol{x}_{\uparrow_2}^{(t-1)})$, *i.e.*, first convolving with Gaussian kernels $\boldsymbol{g}_{p,q}$, followed by a weighted summation. Unlike the non-Gaussian group kernel $\tilde{\boldsymbol{g}}_p$, the 2D Gaussian kernel $\boldsymbol{g}_{p,q}$ is separable and its convolution can be factorized into two 1D convolutions, increasing the efficiency.

4.4 Loss Function for Model Training

The GGKMNet is trained using a multi-component loss function targeting different aspects, defined by

$$\ell_{\text{total}} = \ell_{\text{MSE}} + \lambda_1 \ell_{\text{FR}} + \lambda_2 \ell_{\text{SSIM}} + \lambda_3 \ell_{\text{SR}}, \quad (17)$$

where $\lambda_1, \lambda_2, \lambda_3 \in \mathbb{R}_+$ control the relative importance of each component, and are all set to 0.1 in this work. See below for the discussion of each component in (17).

MSE loss ℓ_{MSE} In common with many other multi-scale deblurring DNNs (*e.g.* [32], [33]), we use the Euclidean loss to measure the difference between the estimate $\boldsymbol{x}^{(t)}$ and the GT $\boldsymbol{x}_{\text{GT}}^{(t)}$ at each scale. The loss is given by

$$\ell_{\text{MSE}} = \sum_{t=1}^{T} \omega^{(t)} ||\boldsymbol{x}^{(t)} - \boldsymbol{x}_{\text{GT}}^{(t)}||_2^2,\ \omega^{(t)} \in \mathbb{R}_+. \quad (18)$$

Frequency reconstruction loss ℓ_{FR} Reducing the distance in frequency domain is important for image deblurring, as it helps to restore the corrupted high-frequency parts [37]. To this end, we include an FFT-based multi-scale frequency reconstruction loss as an auxiliary loss, which is defined by

$$\ell_{\text{FR}} = \sum_{t=1}^{T} \omega^{(t)} ||\text{FFT}(\boldsymbol{x}^{(t)}) - \text{FFT}(\boldsymbol{x}_{\text{GT}}^{(t)})||_1. \quad (19)$$

SSIM loss ℓ_{SSIM} It is a common practice to maximize the structural similarity between the recovered image and the GT for better visual quality [48]. The SSIM (Structural Similarity Index) loss is commonly used for this purpose, and we define it at each scale as follows:

$$\ell_{\text{SSIM}} = \sum_{t=1}^{T} \omega^{(t)} (1 - \text{SSIM}(\boldsymbol{x}^{(t)}, \boldsymbol{x}_{\text{GT}}^{(t)})). \quad (20)$$

Self-reconstruction loss $\ell_{\mathbf{SR}}$ To further regularize the learning process, we define an auxiliary task in a self-supervised manner, which reconstructs the defocused image $\boldsymbol{y}^{(t)}$ back by applying a sub-NN to the extracted blur features in BFE, the coefficients $\boldsymbol{\beta}^{(t)}$ predicted by $\boldsymbol{\beta}$-APU, and the image estimate $\boldsymbol{x}^{(t)}$. The motivation of introducing this task is as follows. If the features and latent image are accurately predicted, they should be able to reconstruct the blurred image. Further, different from uniform blur, spatially-varying defocus blur will lead to different degree of detail loss in different regions. For a defocused image, there are nearly in-focus regions that preserve most image details, but such regions are unknown. To faithfully keep these un-degraded image details, the deblurred image should contain them so that it can be transformed back to the whole defocused image.

The sub-NN for the reconstruction task is called Self Reconstruction Module (SRM), which is only required during training and thus does not bring computational overhead to the inference process. At the scale t, the SRM concatenates the feature maps produced at the bottleneck of BFE (*i.e.*, $\boldsymbol{F}_1^{(t)}$ in Fig. 2), the coefficients produced by the $\boldsymbol{\beta}$-APU, and the image estimate as input (all the three are of the same size), and it outputs a synthesized defocused image, denoted by $\tilde{\boldsymbol{y}}^{(t)}$. We then use the multi-scale L1 loss for measuring the reconstruction accuracy:

$$\ell_{\text{SR}} = \sum_{t=1}^{T} \omega^{(t)} ||\tilde{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{(t)}||_1. \tag{21}$$

Given that the input for self-reconstruction comprises the predicted image and blur-related features, rather than the GT all-in-focus image, it is important to account for the possibility of outliers in the synthesized image $\tilde{\boldsymbol{y}}^{(t)}$. Thus, we utilize the ℓ_1 distance in (21) to enhance robustness against outliers. We implement the SRM as a shallow CNN consisting of three sequential Conv layers separated by ReLU, with a Sigmoid activation function for final output. It is quite simple and thus not shown in Fig. 2.

Setting of $\omega^{(t)}$ The weights $\omega^{(t)}$ for each loss item at different scales are shared among all the four loss functions above. In implementation, we set $\omega^{(t)} = 2^{1-t}$.

5 Performance Evaluation

5.1 Experimental Setting

Implementation details Through all the experiments, we set the hyperparameters $P = 21, Q = 4, T = 3$ for the GGKMNet.[1] All learnable DNN's parameters are initialized with the Xavier method [49] before training. During training, we use the Adam optimizer [50] with a batch size of 4 and train for 5000 epochs. The initial learning rate is set to 10^{-4} for the first 2000 epochs and then reduced by a factor of 2 for every 1000 epochs. Similar to existing work, data augmentation is done via cropping patches of size 256×256 and random horizontal/vertical flipping. The PyTorch code of our preliminary conference work [25] has been published on github.com/csZcWu/GKMNet, and the code of this work will be also published through the same link upon the paper's acceptance.

1. The GKMNet proposed in our previous conference work [25] employs 21 Gaussian kernels for the GKM model.

TABLE 1
Characteristics of datasets used for performance evaluation.

Dataset	#Images	Resolution	Acquisition Method	GT	Split
DPDD	500	1120×1680	Canon EOS 5D Mark IV	✓	✓
LFDOF	11986	688×1008	Lytro Illum	✓	✓
RealDOF	50	≈1536×2320	Dual DSLR System	✓	n/a
RTF	22	360×360	Lytro	✓	n/a
CUHK	704	≈470×610	Internet	n/a	n/a

Benchmark datasets Five publicly available datasets are used for performance evaluation, including

1) DPDD [15]: This dataset contains 500 defocused/in-focus image pairs in resolution 1120×1680, captured by a Canon EOS 5D Mark IV camera. It provides an official training/test split.
2) LFDOF [18]: This dataset contains 11986 defocused/in-focus image pairs in resolution 688×1008, captured using a Lytro Illum light field camera. It provides an official training/test split.
3) RealDOF [16]: This dataset contains 50 real defocused and corresponding in-focus image pairs in a resolution of 1536×2320, captured by two digital single lens reflex (DSLR) cameras with a beam splitter. We use this dataset for generalization test.
4) RTF [6]: This dataset contains 22 images in resolution 360×360, captured using a Lytro light field camera. We use this dataset for generalization test.
5) CUHK-BD [51]: This is a blur detection dataset that provides 704 real defocused images in a size of roughly 470×610 and their binary masks of blur/sharp regions. Since no in-focus GT is provided, we use this dataset for qualitative comparison only.

The characteristics of these datasets are also summarized in Table 1. For DPDD and LFDOF, we train and evaluate our model using their official training/test splits, respectively. The two trained models are also evaluated using the other three datasets for generalization ability analysis.

Baselines Totally eleven SIDD methods are selected for performance comparison, including 1) two 2-stage methods: JNB [1] and EBDB [4]; and 2) nine DNN-based methods: DPDNet-S [15], KPAC [19], IFANet [16], AIFNet [18], GKMNet [25], MDP [24], DRBNet [17], MPRNet* [29] and NRKNet [30]. The results of these methods are sourced from existing literature whenever feasible, and in cases where necessary, we retrain the models using their publicly available codes. Note that for the LFDOF dataset, we train IFANet by disabling its dual-view disparity term in the original loss function, as LFDOF lacks dual-view data. The MRPNet* is the MPRNet [52] trained by the robust scheme of [29] for handing the misalignment issue in DPDD. The GKMNet is the model proposed in our preliminary conference work [25]. In addition, two scale-recurrent DNNs of dynamic scene deblurring, SRN [33] and AttSRN [53], are also retrained on the datasets of SIDD for comparison. The results of those DNN-based methods are quoted from existing literature whenever the experimental settings match, or obtained by running their published training/test codes

TABLE 2
Quantitative comparison on DPDD. **Bold** and underline indicate the best and second-best results in deblurring accuracy, respectively.

Method	PSNR(dB)	SSIM	LPIPS	#MACs(B)	#Para(M)	Time(s)
Input	23.890	0.725	0.349	-	-	-
JNB	23.840	0.715	0.315	-	-	-
EBDB	23.450	0.683	0.336	-	-	-
DPDNet-S	24.348	0.747	0.277	484.8	35.25	0.261
AIFNet	24.213	0.742	0.309	984.7	41.56	0.471
MDP	25.347	0.763	0.268	1080.8	46.86	0.534
KPAC	25.221	0.774	0.226	98.5	2.06	0.082
IFANet	25.366	0.789	0.217	362.9	10.48	0.274
DRBNet	25.485	0.792	0.254	346.6	11.7	0.201
MRPNet*	25.730	0.792	0.232	5347.5	20.1	2.200
SRN	24.612	0.612	0.256	763.0	10.25	0.338
AttSRN	25.224	0.781	0.219	763.8	10.28	0.369
NRKNet	26.109	**0.810**	**0.210**	552.8	6.09	0.329
GKMNet	25.468	0.789	0.219	147.8	1.47	0.145
GKMNet*	25.711	0.790	0.219	147.8	1.47	0.145
GGKMNet	26.039	0.806	0.215	158.4	1.68	0.154
GGKMNet†	**26.272**	**0.810**	0.215	518.9	5.93	0.322

TABLE 3
Quantitative comparison on LFDOF. **Bold** and underline indicate the best and second-best results in deblurring accuracy, respectively.

Method	PSNR(dB)	SSIM	LPIPS	#MACs(B)	#Para(M)	Time(s)
Input	25.874	0.777	0.320	-	-	-
AIFNet	29.677	0.884	0.202	984.7	41.56	0.471
MDP	28.069	0.834	0.185	1080.8	46.86	0.534
KPAC	28.942	0.857	0.174	98.5	2.06	0.082
IFANet	29.787	0.872	0.154	362.9	10.48	0.274
DRBNet	30.253	0.883	**0.147**	346.6	11.7	0.201
SRN	28.971	0.851	0.176	763.0	10.25	0.338
AttSRN	29.621	0.867	0.169	763.8	10.28	0.369
NRKNet	30.481	0.884	**0.147**	552.8	6.09	0.329
GKMNet	29.081	0.867	0.171	147.8	1.47	0.145
GKMNet*	29.125	0.867	0.170	158.4	1.68	0.154
GGKMNet	29.936	0.873	0.166	158.4	1.68	0.154
GGKMNet†	**30.552**	**0.886**	0.154	518.9	5.93	0.322

on the same training/test data as those we use. For a comprehensive comparison, we also retrain GKMNet, denoted by GKMNet* using the same loss function as employed in GGKMNet, where the self-reconstruction loss is applied to the U-Net backbone of GKMNet, paralleling GGKMNet.

Metrics Three image quality metrics are employed for evaluation: two standard metrics, Peak Signal to Noise Ratio (PSNR) and SSIM [54], and a perceptual quality metric, Learned Perceptual Image Patch Similarity (LPIPS) [55], which was also used in [15]. In addition, we evaluate the model/computational complexity by three metrics: number of MACs (multiply-accumulate operations), number of parameters, and average inference time for deblurring an image of 1280×720 pixels. All the experiments are conducted on a single NVIDIA RTX 2080Ti GPU.

5.2 Evaluation on DPDD and LFDOF

Evaluation in terms of complexity The complexity of different DNNs is compared in Table 2. It is evident that our proposed GGKMNet only marginally increases the model complexity and computational cost over the GKMNet presented in our previous conference work. Moreover, compared to the other models, the GGKMNet exhibits relatively low model complexity and low computational cost. We also replace iterative group Gaussian convolutions in the IGGCM by the standard Gaussian convolutions used in GKMNet [25]. It results in additional 13G of MACs and 0.152 seconds in running time. This indicates the design of IGGCM does bring some acceleration. In the subsequent experiments, to facilitate a fairer comparison with NRKNet and DRBNet, two state-of-the-art methods, we construct a larger GGKMNet model, denoted by GGKMNet†, via augmenting the number of residual blocks in BFE. Importantly, GGKMNet† maintains a model size comparable to NRKNet, yet being notably smaller than DRBNet.

Evaluation on DPDD See Table 2 for the quantitative comparison on DPDD. The GGKMNet† outperforms all other compared methods in PSNR and SSIM. The performance of EBDB and JNB is much worse than that of GGKMNet†, probably due to that the errors in their defocus maps are magnified in the consequent deconvolution process. This indicates the advantage of end-to-end learning over two-stage methods. In addition, the GGKMNet† performs better than the SRN and AttSRN, while enjoying smaller model sizes. This indicates the necessity to develop specific DNN architectures for optimizing the performance of SIDD. Further, the noticeable improvement of GGKMNet† over DDPDNet-S indicates the importance of exploiting physical characteristics of defocus blur for SIDD, and, the superior performance of GGKMNet† over other DNNs (*e.g.* NRKNet) has clearly demonstrated the effectiveness of our specific DNN design. See also Fig. 3 for a visual comparison on some deblurred images, where the GGKMNet† produces results of better visual quality than other compared methods.

Indeed, as a smaller version of GGKMNet†, the GGKMNet is already very competitive, with superior performance to almost all the baseline methods. In addition, the GKMNet, the preliminary version of GGKMNet, can also already make use of a light-weight model to compete with other methods. In comparison to GKMNet, the GGKMNet gains over 0.5dB improvement in PSNR, but with only a little increase in model size and running time, demonstrating the advantage of GGKMNet over GKMNet.

Evaluation on LFDOF See Table 3 for the quantitative comparison on LFDOF. In comparison to GKMNet, the GGKMNet gains more than 0.4dB improvement in PSNR. On both datasets, while showing improvement over GGKMNet, GKMNet* still performs notably worse than GGKMNet, indicating the superiority of GGKM over GKM for defocus PSF representation. Overall, the GGKMNet performs competitively in terms of all the three metrics, with a quite light-weight model. While it performs worse than DRBNet in PSNR, the GGKMNet has a significantly smaller model size (*i.e.* around 1/7). However, the GGKMNet, a larger version of GGKMNet, outperforms DRBNet noticeably in PSNR and SSIM, while its model size is only around one half of that of DRBNet. The GGKMNet† also performs better than other compared methods in terms of PSNR and SSIM, and it is the second best in terms of LPIPS. All these results have again validated the effectiveness of GGKMNet. See also Fig. 3 for a visual inspection on some deblurred images. The GGKMNet† is better at restoring the all-in-focus images

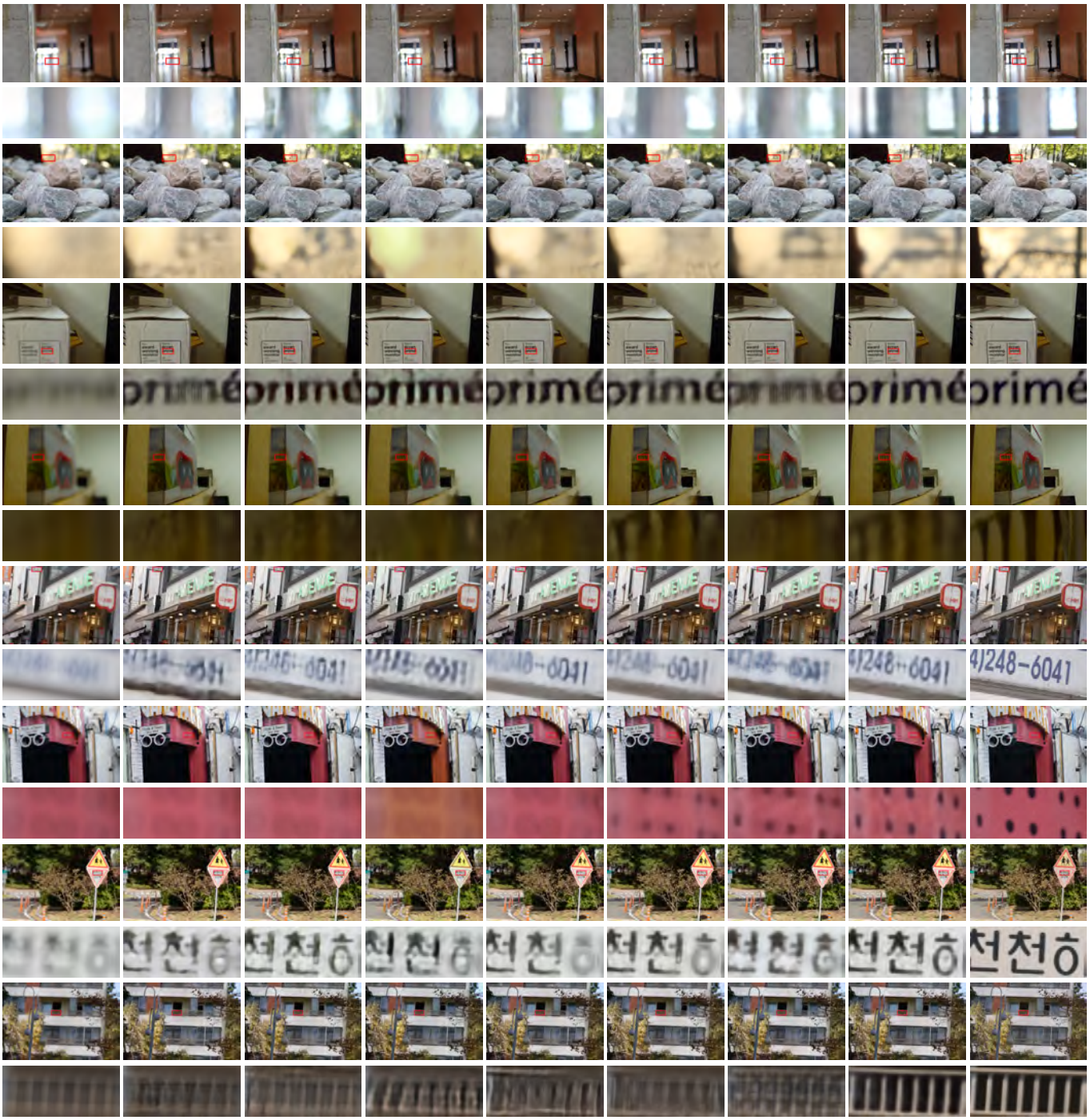


Fig. 3. Visual comparison of deblurred images from selected methods on DPDD (first two), LFDOF (middle two) and RealDOF (last four).

with clearer texture and text than other compared methods.

5.3 Generalization Ability Analysis

Defocus blur is closely related to the physical properties of the aperture and lens in the acquisition device. In real-world situations, it is crucial for a model trained on a particular camera to generalize to images captured by other cameras. To assess the generalization ability, we evaluate the models trained on DPDD and LFDOF using another three datasets including RealDOF, RTF, and CUHK-BD.

Evaluation on RealDOF The quantitative results on RealDOF are listed in Table 4 for comparison. Our proposed GGKMNet†, no matter trained on DPDD or LFDOF, achieves the highest PSNR and SSIM values among all the compared methods. It also achieves the best LPIPS value when trained on LFDOF. These results have clearly indicated the strong generalization capability of GGKMNet†, which mainly comes from its highly-interpretable architecture. Specially, LFDOF is captured by a light-field camera, while RealDOF is captured by a DLSR system. We can see that both the GGKMNet and GGKMNet† trained on LFDOF

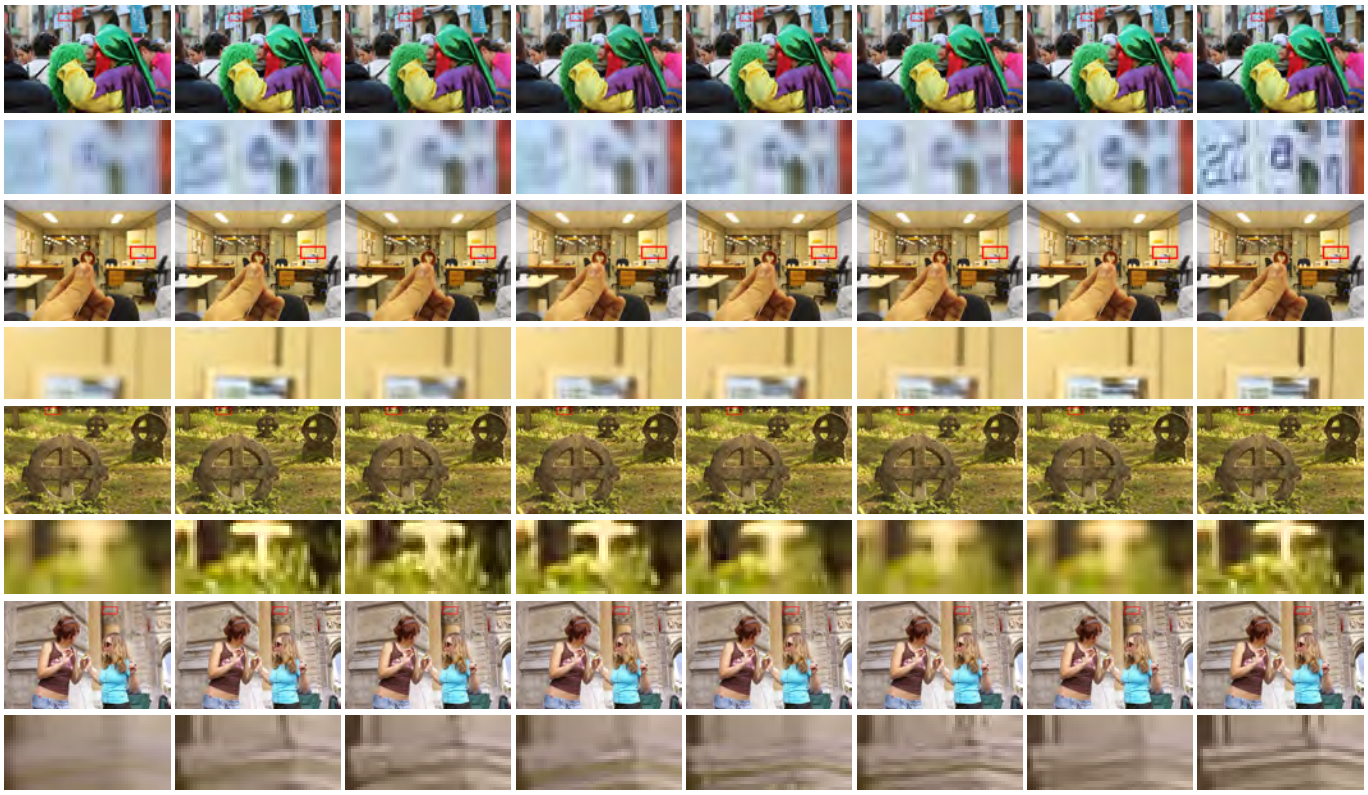


Fig. 4. Visual comparison of deblurred results on images selected from CUHK-BD. Upper half: results from DPDD-trained models; Bottom half: results from LFDOF-trained models.

TABLE 4
Quantitative comparison on RealDOF. **Bold** and underline indicate the best and second-best results.

Method	Trained on DPDD			Trained on LFDOF		
	PSNR(dB)	SSIM	LPIPS	PSNR(dB)	SSIM	LPIPS
Input	22.333	0.633	0.524	22.333	0.633	0.524
AIFNet	23.093	0.680	0.413	22.623	0.667	0.461
MDP	23.500	0.681	0.444	22.726	0.680	0.453
KPAC	23.975	0.762	0.338	22.550	0.671	0.457
IFANet	24.712	0.748	**0.306**	22.504	0.669	0.483
DRBNet	24.884	0.751	0.376	22.910	0.691	0.437
NRKNet	25.148	0.768	0.338	23.528	0.716	0.431
GGKMNet	24.942	0.763	0.356	23.901	0.729	0.399
GGKMNet†	**25.355**	**0.770**	0.320	**24.108**	**0.735**	**0.385**

TABLE 5
Quantitative comparison on RTF. **Bold** and underline indicate the best and second-best results.

Method	Trained on DPDD			Trained on LFDOF		
	PSNR(dB)	SSIM	LPIPS	PSNR(dB)	SSIM	LPIPS
Input	23.608	0.591	0.296	24.200	0.717	0.248
AIFNet	24.041	0.758	0.289	27.552	0.882	0.176
MDP	24.012	0.738	0.312	25.580	0.809	0.228
KPAC	24.618	0.777	0.236	25.959	0.803	0.230
IFANet	24.924	0.801	0.227	26.437	0.838	0.238
DRBNet	24.463	0.773	0.311	26.717	0.853	0.200
NRKNet	25.931	0.829	0.215	28.047	0.889	0.145
GGKMNet	25.895	0.827	0.225	27.908	0.883	0.169
GGKMNet†	**26.012**	**0.846**	**0.210**	**28.308**	**0.905**	**0.140**

still performs well on RealDOF, indicating their generalizability across different types of cameras. It is interesting to see that, in comparison to those trained on DPDD, the models of IFANet and DRBNet trained on LFDOF show significant performance degradation (*i.e.* more than 2dB in PSNR), and that of KPAC also suffers from a PSNR drop of 1.5dB. While the GGKMNet and GGKMNet† also show some performance drop, the PSNR decrement is smaller (*i.e.* around 1dB and 1.2dB respectively). See also Fig.3 for a visual inspection on some deblurred images from different methods. The GGKMNet† produces clearer image with more structures and details, compared to other methods.

Evaluation on RTF Table 5 presents a quantitative comparison of the performance on RTF. Since RTF and LFDOF are both captured by light-filed cameras, the models trained on LFDOF performs noticeably better on RTF, in comparison to those trained on DPDD. In both training settings, the GGKMNet† achieves the best results in all metrics. These results have again demonstrated the strong generalization capability of GGKMNet†.

Evaluation on CUHK-BD The CUHK-BD dataset does not provide GTs. Therefore, we perform a qualitative comparison on the deblurred results of some of its images. See Fig. 4 for a visual inspection. We select the models trained on DPDD for comparison. Though the defocus PSFs of the images synthesized from dual-pixel images differ from the ones in real-world images of CUHK-BD collected from Internet, the GGKMNet† performs well on those real-world images. Specifically, we can see that the GGKMNet successfully restores fine details with fewer visual artifacts,

TABLE 6
Performance drop on noisier DPDD test set, measured by the difference between the performance of re-trained models on noisy data and the performance of original models on noise-free data.

Noise	IFANet			NRKNet			GGKMNet		
$\hat{\sigma}$	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
3	0.481	0.031	0.095	0.962	0.035	0.086	0.343	0.025	0.074
5	0.632	0.036	0.108	1.045	0.041	0.096	0.471	0.032	0.091
15	0.918	0.060	0.145	1.452	0.070	0.144	0.937	0.060	0.154

TABLE 7
Performance drop on noisier RTF, measured by the difference between the performance of re-trained models on noisy data and the performance of original models on noise-free data.

Noise	IFANet			NRKNet			GGKMNet		
$\hat{\sigma}$	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
3	0.675	0.021	0.085	1.651	0.011	0.100	0.300	0.003	0.030
5	0.771	0.028	0.092	1.863	0.012	0.101	0.572	0.019	0.091
15	1.471	0.087	0.115	2.824	0.073	0.140	1.611	0.088	0.114

in comparison to other compared methods. For instance, it is evident that GGKMNet† restores clearer characters in the first example of Fig. 4, compared to the other methods.

Impact of mixd dataset training It is interesting to examine whether training on a mixture of multiple datasets leads to generalization improvement. Thus, we combine DPDD and LFDOF to retrain GGKMNet. The performance gain is 0.148dB/0.002/0.006 on RealDOF over our DPDD-trained model and 0.004dB/0.009/0.001 on RTF compared to our LFDOF-trained model, in terms of PSNR/SSIM/LPIPS. Overall, there is no significant improvement. This phenomenon, where mixed dataset training brings no noticeable improvement, is also seen in [17].

5.4 Noise Robustness Analysis

As mentioned in Section 4.1, the GGKMNet does not include an explicit denoising module like existing unrolling DNNs. It is necessary to test its robustness to noise. Following [6], [16], we add Gaussian white noise with standard deviations drawn from [0,255$\cdot\frac{7}{100}$] to the blurred images of DPDD for training. Then we have a new model retrained on the nosier DPPD samples. For evaluation, we corrupt the test images of DDPP by Gaussian white noise with standard deviations $\hat{\sigma} = 3, 5, 15$, respectively. Using the same way, three noisier versions of RTF are also generated for test. The IFANet and NRKNet are selected for comparison. See Table 6 and Table 7 for the result. For the convenience of comparison, we report the performance drop of three methods caused by noise, which is measured by the difference between the quantitative results of the original model on original test data (*i.e.*, the values reported in Table 2 and 5) and that of the re-trained models on the noisier data. In the presence of additional noise, our GGKMNet shows certain performance drop, with an overall similar level to IFANet. In contrast, the performance drop of NRKNet is noticeably larger. This is probably attributable to that the NRKNet imposes the less restive/effective constraints on defocus PSFs. See Fig. 5 for the deblurring results from the NRKNet and GGKMNet on a blurry noisy image with $\sigma = 15$. The NRKNet generates many noticeable artifacts on the bricks in the deblurred image. In comparison, GGKMNet's result looks much better.

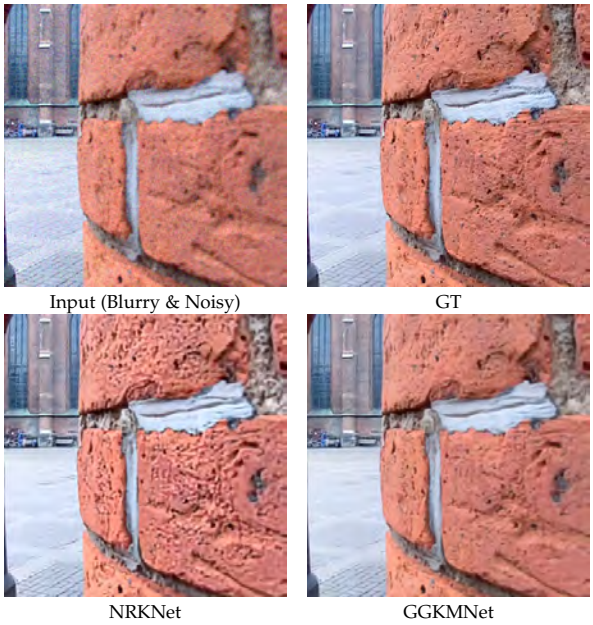


Fig. 5. Deblurring results of NRKNet and GGKMNet on a blurry noisy image with noise level $\hat{\sigma} = 15$.

6 ANALYSIS AND DISCUSSION

6.1 Ablation Study

To assess the performance contribution of each component in the GGKMNet, we form several baselines by removing or replacing some component(s) from it. The quantitative results of these baselines are listed in Table 8 for comparison.

GGKM model vs. GKM model We remove the grouping strategy from the GGKM model, resulting in a similar model to the GKMNet of our previous conference work [25]. Concretely, we remove the $\boldsymbol{\alpha}$-APU from MCP, and replace the IGGCM with the simple GCM used in GKMNet [25] which consists of K individual Gaussian convolutions. The resulting model is denoted as "w/o Group (K)". We set K to 21 and 84, corresponding to the group number and total kernel number used in GGKMNet, respectively. For $K = 21$, we use the 21 Gaussian kernels pre-defined in [25]. It yields a model with a similar complexity to GGKMNet, but performing noticeably worse due to the lowered expressivity of the model of defocus PSFs. For $K = 84$, we use the 84 previously pre-defined Gaussian kernels of the original GGKMNet. It yields a model with close performance to GGKMNet, but resulting in significantly higher model/computational complexity, *i.e.*, approximately tripling the model size and quadrupling the computational cost. Such result indicate that the proposed grouping strategy can gain performance improvement via increasing the number of Gaussian kernels, while bringing marginal model/computational complexity.

w/ vs. w/o scale-recurrent architecture We construct a baseline GGKMNet(1) by using only the original image scale in GGKMNet with $T = 1$, which yields a slightly smaller model with much worse results. In other words,

TABLE 8
Results in ablation study using DPDD.

Model	PSNR(dB)	SSIM	LPIPS	#Para(M)	#MACs(B)
w/o Group (21)	25.798	0.794	0.241	1.68	152.3
w/o Group (84)	26.012	0.807	0.218	4.60	673.4
w/o LSTM (N)	23.881	0.741	0.371	4.72	136.1
w/o LSTM (S)	25.678	0.797	0.256	1.55	136.1
GGKMNet(1)	24.275	0.752	0.283	1.68	115.3
MCP-Net	22.686	0.672	0.542	1.55	136.0
Fully-Learned (R)	25.861	0.801	0.256	1.68	158.7
Fully-Learned (G)	26.072	0.808	0.216	1.68	158.7
w/o ℓ_{FR}	25.857	0.795	0.239	1.68	158.4
w/o ℓ_{SSIM}	25.944	0.791	0.225	1.68	158.4
w/o ℓ_{SR}	25.864	0.788	0.238	1.68	158.4
GGKMNet	26.039	0.806	0.215	1.68	158.4

the multi-scale architecture of GGKMNet with coarse-to-fine estimation leads to significant improvement over the single-scale version, while introducing only a few additional model parameters. We also construct another baseline "w/o LSTM" by replacing each LSTM layer in the MCP by a fully-connected layer with the same size, as well as replacing each Conv-LSTM layer by a Conv layer with the same kernel size. We test two cases: weight sharing over scales on the newly-added layers is enabled and disabled respectively, denoted by "S" and "N". The improvement from using LSTM and Conv-LSTM layers over simple convolutions is significant, no matter using cross-scale weight sharing or not. The reason is probably that the LSTM and Conv-LSTM layers can effectively relate and exploit the features learned from different scales to guide the deblurring process. All the results above have demonstrated the effectiveness of the scale-recurrent architecture in our proposed GGKMNet.

Interpretable vs. non-Interpretable DNN design We use the MCP, the full learnable part of GGKMNet, as a whole DNN for SIDD. The $\boldsymbol{\alpha}$-APU is removed and the $\boldsymbol{\beta}$-APU is attached with an 1×1 Conv layer with Sigmoid activation for outputting an image. The resulting model is denoted by "MCP-Net", which yields much worse results than GGKMNet. Another two baselines, denoted by "Fully-Learned (G)" and "Fully-Learned (R)", are constructed by viewing all Gaussian kernels in GGKMNet as learnable kernels, initialized by the original Gaussian kernels and random values, respectively. Learning beyond Gaussian-initialized kernels leads to very minor improvement over using fixed Gaussian kernels. Indeed, the learned kernels look like the original Gaussian kernels, as shown in Fig. 6. In contrast, learning with random kernel initialization yields a noticeable PSNR. This demonstrated the effectiveness of using Gaussian kernels for modeling defocus PSFs.

w/ vs. v/o loss components We retrain the model by removing ℓ_{FR}, ℓ_{SSIM}, ℓ_{SR} respectively. Without one of these loss functions, there is some performance degradation. For instance, without using ℓ_{SR} for model training, there is around 0.18dB PSNR drop observed, similar to that of without using ℓ_{FR}. Note that the performance degradation caused by discarding one of the loss functions is not very large. This is because the GGKM-based unrolling architecture of our GGMNet has achieved effective regularization by exploiting physics constraints of defocus blur.

TABLE 9
Average PSNR(dB) of synthesized defocused images on RTF using different defocus PSF models.

Model	Gaussian	Generalized Gaussian	GKM	GGKM
PSNR(dB)	34.832	35.617	91.680	100.256

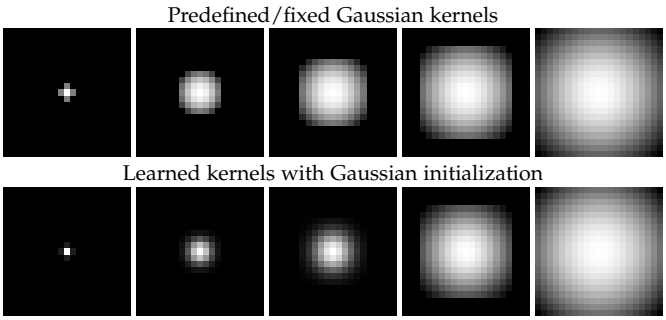


Fig. 6. Visualization of predefined and learned kernels. From left to right is $\boldsymbol{g}_{0,0}$, $\boldsymbol{g}_{5,0}$, $\boldsymbol{g}_{10,0}$, $\boldsymbol{g}_{13,1}$, $\boldsymbol{g}_{20,3}$, respectively.

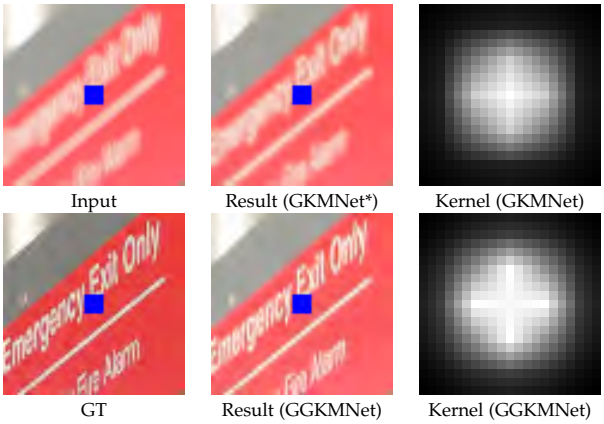


Fig. 7. Deblurred images and estimated defocus kernels of two models.

6.2 GGKM vs. GKM and Single Gaussian Models

This study evaluates the accuracy of different models for defocus PSFs: the proposed GKM/GGKM-based model versus the often-used single Gaussian model. As there is no GT defocus PSFs available in the datasets, we indirectly evaluate the accuracy in terms of the synthesis quality of defocused images from these models. Given a clear/defocused image pair ($\boldsymbol{x}_{\text{GT}}$, $\boldsymbol{y}$), we synthesize the blurred image $\boldsymbol{y}$ using $\boldsymbol{B}_{\boldsymbol{\theta}} \circ \boldsymbol{x}$ via solving $\min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \boldsymbol{B}_{\boldsymbol{\theta}} \circ \boldsymbol{x}\|$. The blurring operator $\boldsymbol{B}_{\boldsymbol{\theta}}$ parameterized $\boldsymbol{\theta}$ is defined by some of the tested defocus PSF models. We use (9) for the GGKM model, (5) for the GKM model with 21 kernels (same as the group number of GGKM), and (4) for the single Gaussian kernel-based model. In addition, the generalized Gaussian model of [8] is also chosen for test. The average PSNR between $\boldsymbol{y}$ and $\boldsymbol{B}_{\boldsymbol{\theta}} \circ \boldsymbol{x}$ on three randomly-selected images from RTF is reported in Table 9.[2] Both the GGKM and GKM models yield noticeably better synthesis quality than both the single and generalized Gaussian models. Further, the synthesis accuracy of GGKM is higher than that of GKM. All these results have verified

2. As the computational time is huge in the single and generalized Gaussian cases, we only sample three images for test.

TABLE 10
Quantitative results of GGKMNet using different values for the group number P and inner-group Gaussian kernel number Q.

P	Q	PSNR(dB)	SSIM	LPIPS	#Para(M)	#MACs(B)
10	1	25.644	0.790	0.253	1.65	151.3
21	1	25.809	0.794	0.241	1.68	152.2
84	1	26.012	0.807	0.218	4.60	673.4
21	4	26.039	0.806	0.215	1.68	158.4

TABLE 11
Quantitative results of GGKMNet using different multi-scale settings. The PSNR is calculated on DPDD.

		GKMNet			GGKMNet		
S	R	PSNR(dB)	#Para(M)	#MACs(B)	PSNR(dB)	#Para(M)	#MACs(B)
1	3	25.114	4.22	389.9	25.723	5.02	345.8
2	1	25.236	1.41	137.1	25.810	1.68	144.1
2	2	25.377	2.81	346.1	25.965	3.35	288.2
2	3	25.127	4.22	423.8	25.767	5.02	432.2
3	1	25.468	1.41	147.8	26.039	1.68	158.4
3	2	25.529	2.81	296.6	26.121	3.35	317.2

the better effectiveness of GGKM over existing models in modeling defocus PSFs.

For further demonstrating the benefit of GGKM over GKM, we use Fig. 7 to show the deblurred images and the defocus PSFs predicted by GKMNet* and GGKMNet. It can be seen that GKMNet* produced a less accurate PSF, characterized by a smaller size in the blurred region, thus yielding a blurrier result. In contrast, the PSF predicted by GGKMNet is larger, resulting in a sharper deblurred image.

6.3 Effect of Group Number and Kernel Number

Table 10 lists the results of GGKMNet using different values for P and Q. When a group scheme is not utilized with $Q = 1$, the performance is improved with an increase in the number of kernels. However, such improvement comes at the cost of a proportional increase in computational resources. As shown in the 3rd and 4th rows of the table, the model using 4 groups with 21 inner-group kernels yields a similar performance as that of the single-group case of $P = 84$, but reduces the number of parameters and MACs to approximately a quarter of the single-group case, resulting in a more acceptable computational increment. These results indicate the effectiveness of our proposed GGKM model.

6.4 Analysis on Variants of Coarse-to-Fine Unrolling

Recall that the fixed point iteration (12) is applied to multiple iterations at the same scale, while our actual architecture also simultaneously increases the scale. We could further write our architecture as S scales and R iterations per scale, where we use $S = 3$ and $R = 1$ in GGKMNet. To further analysis the effect of such a multi-scale estimation scheme for GGKMNet, Table 11 lists the results using different values of S and R. It shows that using single or fewer scales with multiple iterations noticeably decreases the performance and increases the inference time. Using an additional iteration in the three-scale case brings a small improvement but nearly doubles the complexity and time. Thus, our coarse-to-fine scheme is a better implementation

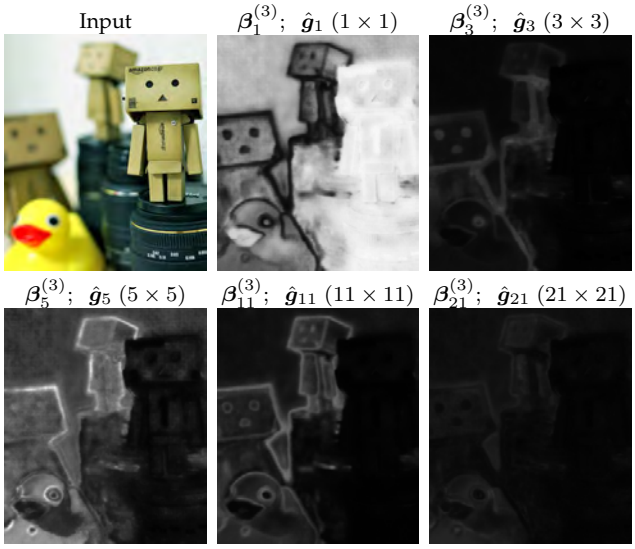


Fig. 8. Visualization of coefficient maps $\boldsymbol{\beta}_p$.

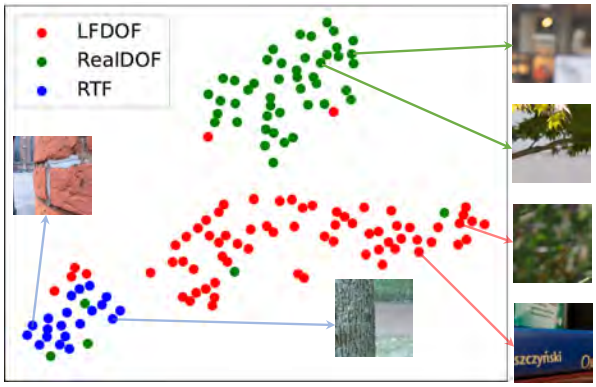
for high performance and low computational cost. The reason that using a single iteration per scale in our approach can achieve high performance is probably due to the use of LSTM and Conv-LSTM across different scales. These cross-scale memory modules enhance the feature flow from one scale to another. As a result, the iteration at each scale can fully exploit the iteration result from previous scales for accurate prediction, and thus one iteration at each scale suffices to achieve good results.

6.5 Visualization of Predicted Coefficient Maps

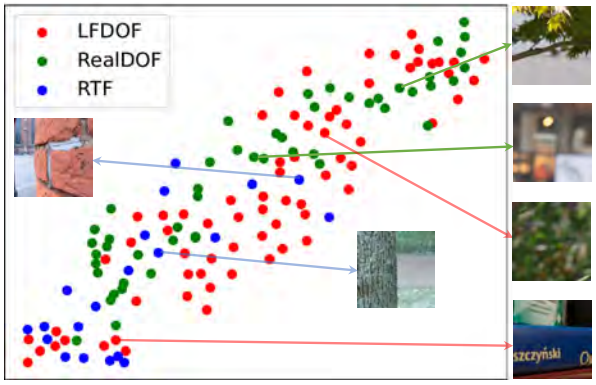
To further support the interpretability of our GGKMNet, we use Fig. 8 to visualize the coefficient maps $\boldsymbol{\beta}_p$ generated by the MCP, corresponding to the Gaussian-composite kernels $\hat{\boldsymbol{g}}_p$ of various sizes. It can be seen that the predicted values of the coefficient maps are closely related to the blur degrees and scene depths of the pixels. For instance, the in-focus robot toy shows significantly stronger activations in $\boldsymbol{\beta}_1$ than other toys that are out of focus, while displaying considerably weaker activations in other coefficient maps.

For further validation, we employ the t-SNE technique to visualize the distribution of the predicted coefficient vectors $\boldsymbol{\alpha}_p$ and coefficient maps $\boldsymbol{\beta}_p$ across three datasets, as shown in Fig. 9. Specifically, for an input image, all its $\boldsymbol{\alpha}_p$ vectors are aggregated into a feature vector, and all $\boldsymbol{\beta}_p$ maps at the finest scale (*i.e.* $\boldsymbol{\beta}_p^{(T)}$) are vectorized and then concatenated into a feature vector. To eliminate possible impact of image size, we crop all images to a uniform size 360×360. Subsequently, the feature vectors of all images are projected onto a 2D space using t-SNE, separately for $\boldsymbol{\alpha_p}$ and $\boldsymbol{\beta_p}$. The perplexity and iteration number used for t-SNE are set to 15 and 5000, respectively. For clarity, unique colors are assigned to the 2D points representing images from the same dataset, and we also display images from different datasets. Each dataset encompasses a broad spectrum of blur levels, image contents, and scene depths, and these datasets share some commonalities in these properties.

The results in Fig. 9(a) clearly illustrates that the predicted coefficient vectors $\boldsymbol{\alpha}_p$ of the images from the same



(a) Visualization of $\boldsymbol{\alpha}_p$



(b) Visualization of $\boldsymbol{\beta}_p$

Fig. 9. Visualization of $\boldsymbol{\alpha}_p$ and $\boldsymbol{\beta}_p$ via t-SNE using a perplexity 15 and iteration number 5000. The LFDOF-trained GGKMNet is used. As LFDOF contains significantly more samples than the other two datasets, we randomly selected an equal number of images from LFDOF to match the total number of images in RTF and RealDOF.

dataset exhibit a distinct distribution. Interestingly, the distribution of $\boldsymbol{\alpha}_p$ in LFDOF aligns more closely with that in RTF than that in RealDOF. This is likely attributable to the fact that both LFDOF and RTF are captured by light field cameras while RealDOF is captured by a DLSR camera. In contrast, the result in Fig. 9(b) show that the distribution of $\boldsymbol{\beta}_p$ is distinct from that of $\boldsymbol{\alpha}_p$, with the predicted coefficient maps $\boldsymbol{\beta}_p$ distributed uniformly across datasets.

7 Conclusion and Future Work

Defocus blur is an often-seen blur effect in digital photography, which has unique characteristics that distinguish it from other types of blur. This paper proposed the GGKM model, which parameterizes pixel-wise spatially-varying defocus PSFs based on their isotropy. A DNN is then constructed by unrolling a fixed-point iteration process of GGKM-based deblurring, with a built-in coarse-to-fine estimation scheme. The proposed DNN achieved better results than existing methods, while being lightweight.

Similar to most existing SIDD methods, ours does not account for possible misalignment between blurred images and GT images. Although the misalignment does not affect the isotropic shape property of defocus PSFs in a blurred image, it could impact the effectiveness of GGKM learning, especially in cases of severe misalignment. One of our future works will address potential misalignment in training data.

Additionally, developing a more robust approach to address a wider spectrum of real-world defocus PSFs, including less common and strongly anisotropic types, will also be a focus of our future research. Furthermore, as our experiments showsthat simply combining multiple datasets does not lead to noticeable improvement, we will investigate effective approaches to leveraging a mixture of training datasets for further performance improvement.

References

[1] J. Shi, L. Xu, and J. Jia, "Just noticeable defocus blur detection and estimation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 657–665.
[2] G. Xu, Y. Quan, and H. Ji, "Estimating defocus blur via rank of local patches," in *Proceedings of IEEE International Conference on Computer Vision*, 2017, pp. 5371–5379.
[3] J. Park, Y.-W. Tai, D. Cho, and I. So Kweon, "A Unified Approach of Multi-Scale Deep and Hand-Crafted Features for Defocus Estimation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1736–1745.
[4] A. Karaali and C. R. Jung, "Edge-Based Defocus Blur Estimation With Adaptive Scale Selection," *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1126–1137, Mar. 2018.
[5] J. Lee, S. Lee, S. Cho, and S. Lee, "Deep defocus map estimation using domain adaptation," in *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 222–12 230.
[6] L. D'Andrès, J. Salvador, A. Kochale, and S. Süsstrunk, "Non-parametric blur map regression for depth of field extension," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1660–1673, 2016.
[7] S. Cho and S. Lee, "Convergence analysis of MAP based blur kernel estimation," in *Proceedings of IEEEInternational Conference on Computer Vision*, 2017, pp. 4808–4816.
[8] Y.-Q. Liu, X. Du, H.-L. Shen, and S.-J. Chen, "Estimating generalized gaussian blur kernels for out-of-focus image deblurring," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, pp. 829–843, 2020.
[9] S. Anwar, Z. Hayder, and F. Porikli, "Deblur and deep depth from single defocus image," *Machine vision and applications*, vol. 32, no. 1, pp. 1–13, 2021.
[10] D. A. Fish, A. M. Brinicombe, E. Pike, and J. G. Walker, "Blind deconvolution by means of Richardson-Lucy algorithm." *Journal of The Optical Society of America A-optics Image Science and Vision*, vol. 12, pp. 58–65, 1995.
[11] D. Krishnan and R. Fergus, "Fast image deconvolution using hyper-laplacian priors," in *Proceedings of Advances in Neural Information Processing Systems*, 2009.
[12] H. Ji and K. Wang, "Robust image deblurring with an inaccurate blur kernel," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 1624–1634, 2011.
[13] M. Chen, Y. Quan, T. Pang, and H. Ji, "Nonblind image deconvolution via leveraging model uncertainty in an untrained deep neural network," *International Journal of Computer Vision*, vol. 130, no. 7, pp. 1770–1789, 2022.
[14] J. Dong, S. Roth, and B. Schiele, "DWDN: Deep Wiener Deconvolution Network for Non-Blind Image Deblurring," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.
[15] A. Abuolaim and M. S. Brown, "Defocus deblurring using dual-pixel data," in *Proceedings of European Conference on Computer Vision*. Springer, 2020, pp. 111–126.
[16] J. Lee, H. Son, J. Rim, S. Cho, and S. Lee, "Iterative filter adaptive network for single image defocus deblurring," in *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2034–2042.
[17] L. Ruan, B. Chen, J. Li, and M. Lam, "Learning to Deblur Using Light Field Generated and Real Defocus Images," in *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 304–16 313.
[18] L. Ruan, B. Chen, J. Li, and M.-L. Lam, "AIFNet: All-in-focus image restoration network using a light field-based dataset," *IEEE Transactions on Computational Imaging*, vol. 7, pp. 675–688, 2021.
[19] H. Son, J. Lee, S. Cho, and S. Lee, "Single image defocus deblurring using kernel-sharing parallel atrous convolutions," in *Proceedings of IEEE/CVF International Conference on Computer Vision*, 2021, pp. 2642–2650.

[20] H. Ma, S. Liu, Q. Liao, J. Zhang, and J.-H. Xue, "Defocus image deblurring network with defocus map estimation as auxiliary task," *IEEE Transactions on Image Processing*, vol. 31, pp. 216–226, 2021.

[21] J. Zhang and W. Zhai, "Blind attention geometric restraint neural network for single image dynamic/defocus deblurring," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[22] Z. Wang, X. Cun, J. Bao, W. Zhou, J. Liu, and H. Li, "Uformer: A general u-shaped transformer for image restoration," in *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 683–17 693.

[23] W. Zhao, F. Wei, Y. He, and H. Lu, "United Defocus Blur Detection and Deblurring via Adversarial Promoting Learning," in *Proceedings of European Conference on Computer Vision*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds. Cham: Springer Nature Switzerland, 2022, pp. 569–586.

[24] A. Abuolaim, M. Afifi, and M. S. Brown, "Improving single-image defocus deblurring: How dual-pixel images help through multi-task learning," in *Proceedings of IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 1231–1239.

[25] Y. Quan, H. Wu, Zicong, and H. Ji, "Gaussian kernel mixture network for single image defocus deblurring," in *Proceedings of Advances in Neural Information Processing Systems*, vol. 34, 2021.

[26] A. Levin, D. Lischinski, and Y. Weiss, "A Closed-Form Solution to Natural Image Matting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 228–242, Feb. 2008.

[27] W. Zhao, F. Zhao, D. Wang, and H. Lu, "Defocus blur detection via multi-stream bottom-top-bottom network," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, pp. 1884–1897, 2020.

[28] A. Karaali, N. Harte, and C. R. Jung, "Deep Multi-Scale Feature Learning for Defocus Blur Estimation," *IEEE Transactions on Image Processing*, vol. 31, pp. 1097–1106, 2022.

[29] Y. Li, D. Ren, X. Shu, and W. Zuo, "Learning single image defocus deblurring with misaligned training pairs," *arXiv preprint arXiv:2211.14502*, 2022.

[30] Y. Quan, Z. Wu, and H. Ji, "Neumann network with recursive kernels for single image defocus deblurring," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 5754–5763.

[31] J. Pan, J. Dong, Y. Liu, J. Zhang, J. Ren, J. Tang, Y.-W. Tai, and M.-H. Yang, "Physics-based generative adversarial models for image restoration and beyond," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 7, pp. 2449–2462, 2020.

[32] S. Nah, T. Kim, and K. M. Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 257–265.

[33] X. Tao, H. Gao, Y. Wang, X. Shen, J. Wang, and J. Jia, "Scale-recurrent network for deep image deblurring," in *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8174–8182.

[34] J. Zhang, J. Pan, J. Ren, Y. Song, L. Bao, R. W. Lau, and M.-H. Yang, "Dynamic scene deblurring using spatially variant recurrent neural networks," in *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2521–2529.

[35] Y. Yuan, W. Su, and D. Ma, "Efficient dynamic scene deblurring using spatially variant deconvolution network with optical flow guided training," in *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3555–3564.

[36] D. Park, D. U. Kang, J. Kim, and S. Chun, "Multi-temporal recurrent neural networks for progressive non-uniform single image deblurring with incremental temporal training," in *Proceedings of European Conference on Computer Vision*, 2020.

[37] S.-J. Cho, S.-W. Ji, J.-P. Hong, S.-W. Jung, and S.-J. Ko, "Rethinking Coarse-To-Fine Approach in Single Image Deblurring," in *Proceedings of IEEE/CVF International Conference on Computer Vision*, 2021, pp. 4641–4650.

[38] W. Ren, J. Zhang, J. Pan, S. Liu, J. S. Ren, J. Du, X. Cao, and M.-H. Yang, "Deblurring Dynamic Scenes via Spatially Varying Recurrent Neural Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 8, pp. 3974–3987, Aug. 2022.

[39] J. Zhang, J. Pan, W.-S. Lai, R. W. Lau, and M.-H. Yang, "Learning fully convolutional networks for iterative non-blind deconvolution," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3817–3825.

[40] D. Gong, Z. Zhang, Q. Shi, A. van den Hengel, C. Shen, and Y. Zhang, "Learning deep gradient descent optimization for image deconvolution," *Neural Networks*, 2020.

[41] W. Dong, P. Wang, W. Yin, G. Shi, F. Wu, and X. Lu, "Denoising prior driven deep neural network for image restoration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 10, pp. 2305–2318, 2018.

[42] D. K. P. Raied Aljadaany and M. Savvides, "Douglas-rachford networks: Learning both the image prior and data fidelity terms for blind image deconvolution," in *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 235–10 244.

[43] F.-J. Tsai, Y.-T. Peng, C.-C. Tsai, Y.-Y. Lin, and C.-W. Lin, "BANet: A Blur-Aware Attention Network for Dynamic Scene Deblurring," *IEEE Transactions on Image Processing*, vol. 31, pp. 6789–6799, 2022.

[44] Z. Zhong, Z. Q. Lin, R. Bidart, X. Hu, I. B. Daya, J. Li, and A. Wong, "Squeeze-and-attention networks for semantic segmentation," in *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 062–13 071.

[45] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2015.

[46] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[47] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W. chun Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems*, 2015.

[48] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss Functions for Neural Networks for Image Processing," Apr. 2018.

[49] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[50] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Proceedings of Advances in Neural Information Processing Systems*, 2017.

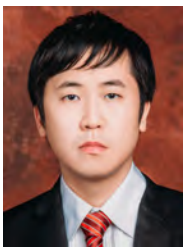
[51] J. Shi, L. Xu, and J. Jia, "Discriminative blur detection features," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2965–2972.

[52] A. Mehri, P. B. Ardakani, and A. D. Sappa, "Mprnet: Multi-path residual network for lightweight image super resolution," in *Proceedings of IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 2704–2713.

[53] Y. Xu, Y. Zhu, Y. Quan, and H. Ji, "Attentive deep network for blind motion deblurring on dynamic scenes," *Computer Vision and Image Understanding*, vol. 205, p. 103169, 2021.

[54] Z. Wang, A. Bovik, H. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, pp. 600–612, 2004.

[55] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 586–595.



Yuhui Quan received his Ph.D. degree in Computer Science from South China University of Technology in 2013. From 2013 to 2016, he served as a postdoctoral research fellow in Mathematics at the National University of Singapore. Currently, he is an associate professor in Computer Science at South China University of Technology. His research interests focus on image recovery, computational imaging, sparse representation, and machine learning.

Zicong Wu received his bachelor degree in Computer Science from South China University of Technology in 2020. He is currently a master student in Computer Science at South China University of Technology. He is working on image deblurring and deep learning.

Ruotao Xu received his Ph.D. degree in Computer Science from South China University of Technology in 2019. He is currently a postdoctoral research fellow in Computer Science at South China University of Technology. He is working on image processing and computer vision.

Hui Ji received his Ph.D. in Computer Science from the University of Maryland, College Park in 2006 and subsequently joined the National University of Singapore as a faculty member. He is currently a professor of applied mathematics at the National University of Singapore, and serve as the director of the Centre for Data Science and Machine Learning. His research interests include wavelets, inverse problems, computational vision, and machine learning.