




# Efficient and Interpretable Deep Blind Image Deblurring Via Algorithm Unrolling

Yuelong Li , *Student Member, IEEE*, Mohammad Tofighi , *Student Member, IEEE*, Junyi Geng , *Student Member, IEEE*, Vishal Monga , *Senior Member, IEEE*, and Yonina C. Eldar , *Fellow, IEEE*

**Abstract**—Blind image deblurring remains a topic of enduring interest. Learning based approaches, especially those that employ neural networks have emerged to complement traditional model based methods and in many cases achieve vastly enhanced performance. That said, neural network approaches are generally empirically designed and the underlying structures are difficult to interpret. In recent years, a promising technique called algorithm unrolling has been developed that has helped connect iterative algorithms such as those for sparse coding to neural network architectures. In this article, we propose a neural network architecture based on this idea. We first present an iterative algorithm that may be considered as a generalization of the traditional total-variation regularization method in the gradient domain. We then unroll the algorithm to construct a neural network for image deblurring which we refer to as Deep Unrolling for Blind Deblurring (DUBLID). Key algorithm parameters are learned with the help of training images. Our proposed deep network DUBLID achieves significant practical performance gains while enjoying interpretability and efficiency at the same time. Extensive experimental results show that DUBLID outperforms many state-of-the-art methods and in addition is computationally faster.

**Index Terms**—Image deblurring, deconvolution, deep learning, algorithm unrolling, interpretable deep networks.

## I. INTRODUCTION

**B**LIND image deblurring refers to the process of recovering a sharp image from its blurred observation without explicitly knowing the blur function. In real world imaging, images frequently suffer from degraded quality as a consequence of blurring artifacts. Blind deblurring algorithms are designed to remove such artifacts. These distortions may come from different sources, such as atmospheric turbulence, diffraction, optical defocusing, camera shaking, and more [1]. In the computational imaging literature, motion deblurring is an important topic because camera shakes are common during the photography

procedure. In recent years, this topic has attracted growing attention due to the popularity of smartphone cameras. On such platforms, the motion deblurring algorithm plays a crucial role because effective hardware solutions such as professional camera stabilizers are difficult to deploy due to cost and space restrictions.

In this work we focus on motion deblurring in particular because of its practical importance. However, our development does not make assumptions on blur type and hence may be extended to cases other than motion blur. Motion blurs occur as a consequence of relative movements between the camera and the imaged scene during exposure. Assuming the scene is planar and the camera motion is translational, the image degradation process may be modelled as a discrete convolution [1]:

$$\mathbf{y} = \mathbf{k} * \mathbf{x} + \mathbf{n}, \quad (1)$$

where  $\mathbf{y}$  is the observed blurry image,  $\mathbf{x}$  is the latent sharp image,  $\mathbf{k}$  is the unknown point spread function (blur kernel), and  $\mathbf{n}$  is random noise which is often modelled as Gaussian. Blind motion deblurring corresponds to estimating  $\mathbf{x}$  from the knowledge of  $\mathbf{y}$  alone; this estimation problem is also commonly called blind deconvolution.

**Related Work:** The majority of existing blind motion deblurring methods are based on iterative optimization. Early works can be traced back to several decades ago [1]–[5]. These methods are only effective when the blur kernel is relatively small. In the last decade, significant breakthroughs have been made both practically and conceptually. As both the image and the kernel need to be estimated, there are infinitely many pairs of solutions forming the same blurred observation rendering blind deconvolution an ill-posed problem. A popular remedy is to add regularizations so that many blind deblurring algorithms essentially reduce to solving regularized inverse problems. A vast majority of these techniques hinge on sparsity-inducing regularizers, either in the gradient domain [6]–[13] or more general sparsifying transformation domains [14]–[16]. Examples of recent advanced transform domains include dark channel [17] and edge profile [18]. Variants of such techniques may arise indirectly from a statistical estimation perspective, such as [19]–[22].

From a conceptual perspective, Levin *et al.* [23] study the limitations and remedies of the commonly employed Maximum a Posterior (MAP) approach, while Perrone *et al.* [24] extend their study with a particular focus on Total-Variation (TV) regularization. Despite some performance improvements achieved along their developments, the iterative optimization

Manuscript received July 1, 2019; revised November 4, 2019 and December 8, 2019; accepted December 10, 2019. Date of publication January 6, 2020; date of current version February 3, 2020. This work was supported by NSF Career Award. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Javier Portilla. (*Corresponding author: Yuelong Li.*)

Y. Li, M. Tofighi, and V. Monga are with the Department of Electrical Engineering, The Pennsylvania State University, University Park, PA, 16802 USA (e-mail: yul200@psu.edu; tofighi@psu.edu; vmonga@engr.psu.edu).

J. Geng is with the Department of Aerospace Engineering, The Pennsylvania State University, University Park, PA, 16802 USA (e-mail: jxg1052@psu.edu).

Y. C. Eldar is with the Department of Electrical Engineering, Faculty of Math and Computer Science, Weizmann Institute of Science, Rehovot, 76100, Israel (e-mail: yonina.eldar@weizmann.ac.il).

This article has supplementary downloadable material available at <http://ieeexplore.ieee.org>, provided by the authors.

Digital Object Identifier 10.1109/TCI.2020.2964202

approaches generally suffer from several limitations. First, their performance depends heavily on appropriate selection of parameter values. Second, handcrafted regularizers play an essential role, and designing versatile regularizers that generalize well to a variety of real datasets can be a challenging task. Finally, hundreds and thousands of iterations are often required to reach an acceptable performance level and thus these approaches can be slow in practice.

Complementary to the aforementioned approaches, learning based methods for determining a non-linear mapping that deblurs the image while adapting parameter choices to an underlying training image set have been developed. For example, in [25] a discriminative prior is learned from real data. Principally important in this class are techniques that employ deep neural networks. The history of leveraging neural networks for blind deblurring actually dates back to the last century [26]. In the past few years, there has been a growing trend in applying neural networks to various imaging problems [27], and blind motion deblurring has followed that trend. Xu *et al.* [28] use large convolution kernels with carefully chosen initializations in a Convolutional Neural Network (CNN); Yan *et al.* [29] concatenate a classification network with a regression network to deblur images without prior information about the blur kernel type. Zhang *et al.* [30] proposed a network which is composed of three deep CNNs and a recurrent neural network (RNN). Tao *et al.* [31] develop a network based on an encoder-decoder structure. Chakrabarti *et al.* [32] work in the frequency domain and employ a neural network to predict Fourier transform coefficients of image patches; Xu *et al.* [33] employ a CNN for edge enhancement prior to kernel and image estimation. These works often outperform iterative optimization algorithms especially for linear motion kernels; however, the structures of the networks are often empirically determined and their actual functionality is hard to interpret.

Other aspects of deblurring have been investigated such as spatially varying blurs [34], [35], including some recent neural network approaches [36]–[39]. Other algorithms benefit from device measurements [40]–[42] or leverage multiple images [43], [44]. A comparative survey of single image deblurring methods is performed in [45].

In the seminal work of Gregor *et al.* [46], a novel technique called algorithm unrolling was proposed. Despite its focus on approximating sparse coding algorithms, it provides a principled framework for expressing traditional iterative algorithms as neural networks, and offers promise in developing interpretable network architectures. Specifically, each iteration step may be represented as one layer of the network, and concatenating these layers form a deep neural network. Passing through the network is equivalent to executing the iterative algorithm a finite number of times. The network may be trained using back-propagation [47], and the trained network can be naturally interpreted as a parameter optimized algorithm. An additional benefit is that prior knowledge about the conventional algorithms may be transferred. There has been exciting recent exploration of neural network architectures by unrolling iterative algorithms for problems such as super-resolution and clutter/noise suppression [48]–[51]. In blind deblurring, Schuler *et al.* [52]

employ neural networks as feature extraction modules towards a trainable deblurring system. However, the network portions are still empirical.

An idea similar to algorithm unrolling is to integrate neural networks with iterative deblurring algorithms; the neural networks may be regarded as discriminative priors [53] or as pre/post-processing modules [54], [55].

**Motivations and Contributions:** Conventional iterative algorithms have the merits of interpretability, but acceptable performance levels demand much longer execution time compared to modern neural network approaches. Despite previous efforts, the link between both categories remains largely unexplored for the problem of blind deblurring, and a method that simultaneously enjoys the benefits of both is lacking. In this paper, we make the following contributions:<sup>1</sup>

- **Deep Unrolling for Blind Deblurring (DUBLID):** We propose an interpretable neural network structure called DUBLID. We first present an iterative algorithm that may be considered as a generalization of the traditional total-variation regularization method in the gradient domain, and subsequently unroll the algorithm to construct a neural network. Key algorithm parameters are learned with the help of training images using backpropagation, for which we derive analytically simple forms that are amenable to fast implementation. Since the DUBLID network architecture is custom derived and departs from known standard deep neural networks, many backpropagation derivatives had to be hand-coded so that their analytical forms are crucial for reproducibility.
- **Performance and Computational Benefits:** Through extensive experimental validation<sup>2</sup> over *six* benchmark datasets, we verify the superior performance of the proposed DUBLID, both over conventional iterative algorithms and more recent neural network approaches. Both traditional linear and more recently developed non-linear kernels are used in our experiments. Besides quality gains, we show that DUBLID is computationally simpler. In particular, the carefully designed interpretable layers enables DUBLID to learn with far fewer parameters than state of the art deep learning approaches — hence leading to very low inference time.

There have been recent efforts that unroll iterative algorithms based on variable splitting such as Yang *et al.* [57], who develop a deep network for compressive sensing inspired by the well-known Alternating Direction Method of Multipliers (ADMM). Our approach is unique and differs significantly from such past efforts in several aspects: 1) the optimization problem we formulate and solve is distinct (see (6)–(10)) and involves determining representation filters, 2) The neural network architecture that results using the steps of the algorithmic solution – see Figs. 1–3 – is unique and differs from previously proposed networks, and 3.) a new, customized training procedure

<sup>1</sup>A preliminary version of this work has appeared in IEEE ICASSP 2019 [56].

<sup>2</sup>To ensure reproducibility, we share our code and datasets that are used to generate all our experimental results freely. ([Online]. Available: [https://scholarsphere.psu.edu/concern/generic\\_works/8sf268475](https://scholarsphere.psu.edu/concern/generic_works/8sf268475) m).

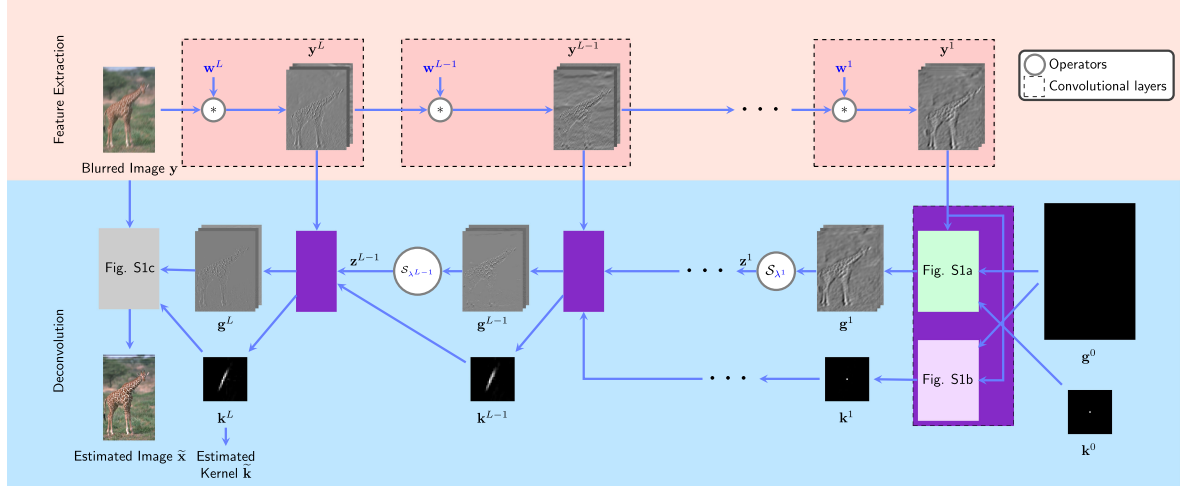


Fig. 1. DUBLID: using a cascade of small  $3 \times 3$  filters instead of large filters (as compared to the network in Fig. S2 in the supplementary document) reduces the dimensionality of the parameter space, and the network is easier to train. Intermediate data (hidden layers) on the trained network are also shown. It can be observed that, as  $l$  increases,  $g^l$  and  $y^l$  evolve in a coarse-to-fine manner. The parameters that are learned from real datasets are colored in blue.

(including a distinct loss minimization algorithm and analytically derived custom back-propagation rules) is developed in Section III-B.

Another related category of deblurring techniques is based on discriminative learning [25], [58]. However, these methods essentially adopt a standard model (such as conditional random fields and conventional deep networks) as a drop-in replacement of a particular sub-problem in a certain iterative optimization algorithm. In contrast, our approach unrolls the entire iterative optimization algorithm into a network, and the network architecture is distinct from standard models. In practice, the proposed DUBLID network can be trained end-to-end to obtain an optimized set of parameters and only needs to be executed once, which significantly enhances its computational efficiency.

The rest of the paper is organized as follows. Generalized gradient domain deblurring is reviewed in Section II-A. We identify the roles of (gradient/feature extraction) filters and other key parameters, which are usually assumed *fixed*. Based on a half-quadratic optimization procedure to solve the aforementioned gradient domain deblurring, we develop a new unrolling method that realizes the iterative optimization as a neural network in Section III. In particular, we show that the various linear and non-linear operators in the optimization can be cascaded to generate an interpretable deep network, such that the number of layers in the network corresponds to the number of iterations. The fixed filters and parameters are now *learnable* and a custom back-propagation procedure is proposed to optimize them based on training images. Experimental results that provide insights into DUBLID as well as comparisons with state of the art methods are reported in Section IV. Section V concludes the paper.

## II. GENERALIZED BLIND DEBLURRING: A FILTERED DOMAIN REGULARIZATION PERSPECTIVE

### A. Blind Deblurring in the Filtered Domain

A common practice for blind motion deblurring is to estimate the kernel in the image gradient domain [7]–[10], [19], [20], [24].

Because the gradient operator  $\nabla$  commutes with convolution, taking derivatives on both side of (1) gives

$$\nabla \mathbf{y} = \mathbf{k} * \nabla \mathbf{x} + \mathbf{n}', \quad (2)$$

where  $\mathbf{n}' = \nabla \mathbf{n}$  is Gaussian noise. Formulation in the gradient domain, as opposed to the pixel domain, has several desirable strengths: the kernel generally serves as a low-pass filter, and low-frequency components of the image are barely informative about the kernel. Intuitively, the kernel may be inferred along edges rather than homogeneous regions in the image. Therefore, a gradient domain approach can lead to improved performance in practice [20] as the gradient operator effectively filtered out the uninformative regions. Additionally, from a computational perspective, gradient domain formulations help in better conditioning of the linear system resulting in more reliable estimation [8].

The model (2) alone, however, is insufficient for recovering both the image and the kernel; thus regularizers on both are needed. The gradients of natural images are generally sparse, i.e., most of their values are of small magnitude [19], [23]. This fact motivates the developments of various sparsity-inducing regularizations on  $\nabla \mathbf{x}$ . Among them one of particular interest is the  $\ell_1$ -norm (often called TV) thanks to its convexity [5], [24], [59]. To regularize the kernel, it is common practice to assume the kernel coefficients are non-negative and of unit sum. Consolidating these facts, blind motion deblurring may be carried out by solving the following optimization problem [24]:

$$\begin{aligned} \min_{\mathbf{k}, \mathbf{g}_1, \mathbf{g}_2} \quad & \frac{1}{2} \left( \|D_x \mathbf{y} - \mathbf{k} * \mathbf{g}_1\|_2^2 + \|D_y \mathbf{y} - \mathbf{k} * \mathbf{g}_2\|_2^2 \right) \\ & + \lambda_1 \|\mathbf{g}_1\|_1 + \lambda_2 \|\mathbf{g}_2\|_1 + \frac{\epsilon}{2} \|\mathbf{k}\|_2^2, \\ \text{subject to} \quad & \|\mathbf{k}\|_1 = 1, \quad \mathbf{k} \geq 0, \end{aligned} \quad (3)$$

where  $D_x \mathbf{y}$ ,  $D_y \mathbf{y}$  are the partial derivatives of  $\mathbf{y}$  in horizontal and vertical directions respectively. The notation  $\|\cdot\|_p$  denotes the



$\ell_p$  vector norm, while  $\lambda_1, \lambda_2, \varepsilon$  are positive constant parameters to balance the contributions of each term. The  $\geq$  sign is to be interpreted elementwise. The solutions  $\mathbf{g}_1$  and  $\mathbf{g}_2$  of (3) are estimates of the gradients of the sharp image  $\mathbf{x}$ , i.e., we may expect  $\mathbf{g}_1 \approx D_x \mathbf{x}$  and  $\mathbf{g}_2 \approx D_y \mathbf{x}$ .

In practice, numerical gradients of images are usually computed using discrete filters, such as the Prewitt and Sobel filters. From this viewpoint,  $D_x \mathbf{y}$  and  $D_y \mathbf{y}$  may be viewed as filtering  $\mathbf{y}$  through two derivative filters of orthogonal directions [60]. Therefore, a straightforward generalization of (3) is to use more than two filters, i.e., pass  $\mathbf{y}$  through a filter bank. This generalization increases the flexibility of (3), and appropriate choice of the filters can significantly boost performance. In particular, by steering the filters towards more directions other than horizontal and vertical, local features (such as lines, edges and textures) of different orientations are more effectively captured [61]–[63]. Moreover, the filter can adapt its shapes to enhance the representation sparsity [64], [65], a desirable property to pursue.

Suppose we have determined a desired collection of  $C$  filters  $\{\mathbf{f}_i\}_{i=1}^C$ . By commutativity of convolutions, we have

$$\mathbf{f}_i * \mathbf{y} = \mathbf{f}_i * \mathbf{k} * \mathbf{x} + \mathbf{n}'_i = \mathbf{k} * (\mathbf{f}_i * \mathbf{x}) + \mathbf{n}'_i, \quad i = 1, 2, \dots, C, \quad (4)$$

where the filtered noises  $\mathbf{n}'_i = \mathbf{f}_i * \mathbf{n}$  are still Gaussian. To encourage sparsity of the filtered image, we formulate the optimization problem (which may similarly be regarded as a generalization of [24]):

$$\min_{\mathbf{k}, \{\mathbf{g}_i\}_{i=1}^C} \sum_{i=1}^C \left( \frac{1}{2} \|\mathbf{f}_i * \mathbf{y} - \mathbf{k} * \mathbf{g}_i\|_2^2 + \lambda_i \|\mathbf{g}_i\|_1 \right) + \frac{\varepsilon}{2} \|\mathbf{k}\|_2^2, \quad \text{subject to } \|\mathbf{k}\|_1 = 1, \quad \mathbf{k} \geq 0. \quad (5)$$

### B. Efficient Minimization via Half-Quadratic Splitting

Problem (5) is non-smooth so that traditional gradient-based optimization algorithms cannot be considered. Moreover, to facilitate the subsequent unrolling procedure, the algorithm needs to be simple (to simplify the network structure) and converge quickly (to reduce the number of layers required). Based on these requirements, we adopt the half-quadratic splitting algorithm [66]. This algorithm is simple but effective, and has been successfully employed in many previous deblurring techniques [13], [16], [58].

The basic idea is to perform variable-splitting and then alternating minimization on the penalty function. To this end, we first cast (5) into the following approximation model:

$$\min_{\mathbf{k}, \{\mathbf{g}_i, \mathbf{z}_i\}_{i=1}^C} \sum_{i=1}^C \left( \frac{1}{2} \|\mathbf{f}_i * \mathbf{y} - \mathbf{k} * \mathbf{g}_i\|_2^2 + \lambda_i \|\mathbf{z}_i\|_1 + \frac{1}{2\zeta_i} \|\mathbf{g}_i - \mathbf{z}_i\|_2^2 \right) + \frac{\varepsilon}{2} \|\mathbf{k}\|_2^2, \quad \text{subject to } \|\mathbf{k}\|_1 = 1, \quad \mathbf{k} \geq 0, \quad (6)$$

by introducing auxiliary variables  $\{\mathbf{z}_i\}_{i=1}^C$ , where  $\zeta_i, i = 1, \dots, C$  are regularization parameters. It is well known that as  $\zeta_i \rightarrow 0$  the sequence of solutions to (6) converges to that of (5) [67]. In a similar manner to [13], we then alternately minimize over  $\{\mathbf{g}_i\}_{i=1}^C, \{\mathbf{z}_i\}_{i=1}^C$  and  $\mathbf{k}$  and iterate until convergence.<sup>3</sup> Specifically, at the  $l$ -th iteration, we execute the following minimizations sequentially:

$$\mathbf{g}_i^{l+1} \leftarrow \arg \min_{\mathbf{g}_i} \frac{1}{2} \|\mathbf{f}_i * \mathbf{y} - \mathbf{k}^l * \mathbf{g}_i\|_2^2 + \frac{1}{2\zeta_i} \|\mathbf{g}_i - \mathbf{z}_i^l\|_2^2, \quad \forall i,$$

$$\mathbf{z}_i^{l+1} \leftarrow \arg \min_{\mathbf{z}_i} \frac{1}{2\zeta_i} \|\mathbf{g}_i^{l+1} - \mathbf{z}_i\|_2^2 + \lambda_i \|\mathbf{z}_i\|_1, \quad \forall i,$$

$$\mathbf{k}^{l+1} \leftarrow \arg \min_{\mathbf{k}} \sum_{i=1}^C \frac{1}{2} \|\mathbf{f}_i * \mathbf{y} - \mathbf{k} * \mathbf{g}_i^{l+1}\|_2^2 + \frac{\varepsilon}{2} \|\mathbf{k}\|_2^2,$$

$$\text{subject to } \|\mathbf{k}\|_1 = 1, \quad \mathbf{k} \geq 0. \quad (7)$$

Many existing blind motion deblurring methods [13], [16], [24] solve a variant of (3) by alternating through two directions, i.e., between kernel estimation and non-blind deblurring. However, this scheme typically leads to a very deep unrolled network (which we will discuss in Section III) as many layers have to be allocated for the non-blind deblurring module. Consequently, the network can be hard to train and computationally less efficient in practice. We therefore adopt a different approach, by alternating through three directions, without compromising the interpretability. From a practical perspective,  $\{\mathbf{g}_i\}_{i=1}^C$  and  $\{\mathbf{z}_i\}_{i=1}^C$  may be updated with a better estimate of the kernel  $\mathbf{k}$ , and the algorithm could potentially converge faster. Correspondingly, the network can have reduced number of layers.

As it is, Problem (5) is non-convex over the joint variables  $\mathbf{k}$  and  $\{\mathbf{g}_i\}_{i=1}^C$  and proper initialization is crucial to get good solutions. However, it is difficult to find appropriate initializations that perform well under various practical scenarios. An alternative strategy that has been commonly employed is to use different parameters per iteration [11], [13], [24], [66]. For example,  $\lambda_i$ 's are typically chosen as a large value from the beginning, and then gradually decreased towards a small constant. In [66] the values of  $\zeta_i$ 's decrease as the algorithm proceeds for faster convergence. In numerical analysis and optimization, this strategy is called the continuation method and its effectiveness is known for solving non-convex problems [68]. By adopting this strategy, we choose different parameters  $\{\zeta_i^l, \lambda_i^l\}_{i,l}$  across the iterations. We take this idea one step further by optimizing the filters across iterations as well, i.e. we design filters  $\{\mathbf{f}_i^l\}_{i,l}$ . Consequently, the alternating

<sup>3</sup>In the non-blind deconvolution literature, a formal convergence proof has been shown in [66] while for blind deconvolution, empirical convergence has been frequently observed as shown in [11], [13].

<sup>4</sup>For notational brevity, we will consistently use  $i$  to index the filters and  $l$  to index the layers (iteration) henceforth. The notations  $\{\cdot\}_i$  and  $\{\cdot\}_l$  collect every filter and layer components, respectively.

minimization scheme in (7) becomes:

$$\mathbf{g}_i^{l+1} \leftarrow \arg \min_{\mathbf{g}_i} \frac{1}{2} \|\mathbf{f}_i^l * \mathbf{y} - \mathbf{k}^l * \mathbf{g}_i\|_2^2 + \frac{1}{2\zeta_i^l} \|\mathbf{g}_i - \mathbf{z}_i^l\|_2^2, \quad \forall i, \quad (8)$$

$$\mathbf{z}_i^{l+1} \leftarrow \arg \min_{\mathbf{z}_i} \frac{1}{2\zeta_i^l} \|\mathbf{g}_i^{l+1} - \mathbf{z}_i\|_2^2 + \lambda_i^l \|\mathbf{z}_i\|_1, \quad \forall i, \quad (9)$$

$$\mathbf{k}^{l+1} \leftarrow \arg \min_{\mathbf{k}} \sum_{i=1}^C \frac{1}{2} \|\mathbf{f}_i^l * \mathbf{y} - \mathbf{k} * \mathbf{g}_i^{l+1}\|_2^2 + \frac{\epsilon}{2} \|\mathbf{k}\|_2^2, \quad (10)$$

subject to  $\|\mathbf{k}\|_1 = 1, \quad \mathbf{k} \geq 0$ .

We summarize the complete algorithm in Algorithm 1, where  $\delta$  in Step 1 is the impulse function. Problem (8) can be efficiently solved by making use of the Discrete Fourier Transform (DFT) and its solution is given in Step 5 of Algorithm 1, where  $*$  is the complex conjugation and  $\odot$  is the Hadamard (element-wise) product operator. The operations are to be interpreted elementwise when acting on matrices and vectors. We let  $\widehat{\cdot}$  denote the DFT and  $\mathcal{F}^{-1}$  be the inverse DFT. The closed-form solution to problem (9) is well known and can be found in Step 6, where,  $\mathcal{S}_\lambda(\cdot)$  is the soft-thresholding operator defined as:  $\mathcal{S}_\lambda(x) = \text{sgn}(x) \cdot \max\{|x| - \lambda, 0\}$ .

Subproblem (10) is a quadratic programming problem with a simplex constraint. While in principle, it may be solved using iterative numerical algorithms, in the blind deblurring literature an approximation scheme is often adopted. The unconstrained quadratic programming problem is solved first (again using DFT) to obtain a solution; its negative coefficients are then thresholded out, and finally normalized to have unit sum (Steps 8–10 of Algorithm 1). We define  $[x]_+ = \max\{x, 0\}$ . This function is commonly called the Rectified Linear Unit (ReLU) in neural network terminology [69]. Note that in Step 9 of Algorithm 1, we are adopting a common practice [8], [16] by thresholding the kernel coefficients using a positive constant (which is usually set as a constant parameter  $\beta$  multiplying the maximum of the kernel coefficients); to avoid the non-smoothness of the maximum operation, we use the log-sum-exp function as a smooth surrogate.

In the algorithm,  $L$  refers to the number of outer iterations, which subsequently becomes the number of network layers in Section III-A. While in traditional iterative algorithms it is commonly determined by certain termination criteria, this approach is difficult to implement for neural networks. Therefore, in this work we choose it through cross-validation as is done in [46], [48].

The quality of the outputs of Algorithm 1 and the convergence speed depend crucially on the filters  $\{\mathbf{f}_i^l\}_{i,l}$  and parameters  $\{\zeta_i^l, \lambda_i^l\}_{i,l}$ , which are difficult to infer due to the huge variety of real world data. Under traditional settings, they are usually determined by hand crafting or domain knowledge. For example, in [16], [24]  $\{\mathbf{f}_i^l\}_{i,l}$  are taken as Prewitt filters while  $\{\lambda_i^l\}_l$ 's are chosen as geometric sequences. Their optimal values thus remain unclear. To optimize the performance, we learn (adapt) the filters and parameters by using training images in a deep

---

**Algorithm 1:** Half-quadratic Splitting Algorithm for Blind Deblurring with Continuation.

---

**Input:** Blurred image  $\mathbf{y}$ , filter banks  $\{\mathbf{f}_i^l\}_{i,l}$ , positive constant parameters  $\{\zeta_i^l, \lambda_i^l\}_{i,l}$ , number of iterations<sup>5</sup>  $L$  and  $\epsilon$ .

**Output:** Estimated kernel  $\tilde{\mathbf{k}}$ , estimated feature maps  $\{\tilde{\mathbf{g}}_i\}_{i=1}^C$ .

```

1: Initialize  $\mathbf{k}^1 \leftarrow \delta; \mathbf{z}_i^1 \leftarrow 0, i = 1, \dots, C$ .
2: for  $l = 1$  to  $L$  do
3:   for  $i = 1$  to  $C$  do
4:      $\mathbf{y}_i^l \leftarrow \mathbf{f}_i^l * \mathbf{y}$ ,
5:      $\mathbf{g}_i^{l+1} \leftarrow \mathcal{F}^{-1} \left\{ \frac{\zeta_i^l \widehat{\mathbf{k}}^l \odot \widehat{\mathbf{y}}_i^l + \widehat{\mathbf{z}}_i^l}{\zeta_i^l \|\widehat{\mathbf{k}}^l\|^2 + 1} \right\}$ ,
6:      $\mathbf{z}_i^{l+1} \leftarrow \mathcal{S}_{\lambda_i^l \zeta_i^l} \{\mathbf{g}_i^{l+1}\}$ ,
7:   end for
8:    $\mathbf{k}^{l+1} \leftarrow \mathcal{F}^{-1} \left\{ \frac{\sum_{i=1}^C \widehat{\mathbf{z}}_i^{l+1} \odot \widehat{\mathbf{y}}_i^l}{\sum_{i=1}^C \|\widehat{\mathbf{z}}_i^{l+1}\|^2 + \epsilon} \right\}$ ,
9:    $\mathbf{k}^{l+1} \leftarrow \left[ \mathbf{k}^{l+1} - \beta^l \log \left( \sum_i \exp \left( \mathbf{k}_i^{l+1} \right) \right) \right]_+$ ,
10:   $\mathbf{k}^{l+1} \leftarrow \frac{\mathbf{k}^{l+1}}{\|\mathbf{k}^{l+1}\|_1}$ ,
11:   $l \leftarrow l + 1$ .
12: end for
```

---

learning set-up via back-propagation. A detailed visual comparison between filters commonly used in conventional algorithms and filters learned through real datasets by DUBLID is provided in Section IV-B.

We note that previous works on algorithm unrolling [46], [49] commonly construct the corresponding neural network without strict adherence to the underlying iterative algorithms. Instead, they often take initial inspiration from those algorithms and apply custom modifications to the networks to enhance their practical merits. Our proposed optimization of parameters across iterations, chiefly the filters, is similarly motivated and its merits are confirmed via an ablation study in Section IV-B. Indeed, altering the parameters (across iterations) is also a common strategy employed in many iterative blind deblurring algorithms such as [11], [13], [16].

After Algorithm 1 concludes, we obtain the estimated feature maps  $\{\tilde{\mathbf{g}}_i\}_i$  and the estimated kernel  $\tilde{\mathbf{k}}$ . Because of the low-pass nature of  $\tilde{\mathbf{k}}$ , using it alone is inadequate to reliably recover the sharp image  $\mathbf{x}$  and regularization is needed. We may infer from (4) that, as  $\tilde{\mathbf{k}}$  approximates  $\mathbf{k}$ ,  $\tilde{\mathbf{g}}_i$  should approximate  $\mathbf{f}_i * \mathbf{x}$ . Therefore, we retrieve  $\mathbf{x}$  by solving the following optimization problem:

$$\begin{aligned} \tilde{\mathbf{x}} \leftarrow \arg \min_{\mathbf{x}} \frac{1}{2} \|\mathbf{y} - \tilde{\mathbf{k}} * \mathbf{x}\|_2^2 + \sum_{i=1}^C \frac{\eta_i}{2} \|\mathbf{f}_i^L * \mathbf{x} - \tilde{\mathbf{g}}_i\|_2^2 \\ = \mathcal{F}^{-1} \left\{ \frac{\widehat{\tilde{\mathbf{k}}}^* \odot \widehat{\mathbf{y}} + \sum_{i=1}^C \eta_i \widehat{\mathbf{f}}_i^L \odot \widehat{\tilde{\mathbf{g}}}_i}{\|\widehat{\tilde{\mathbf{k}}}\|^2 + \sum_{i=1}^C \eta_i \|\widehat{\mathbf{f}}_i^L\|^2} \right\}, \end{aligned} \quad (11)$$

where  $\eta_i$ 's are positive regularization parameters.

### III. ALGORITHM UNROLLING FOR DEEP BLIND IMAGE DEBLURRING (DUBLID)

#### A. Network Construction via Algorithm Unrolling

Each step of Algorithm 1 is in analytic form and may be implemented using a series of basic functional operations. In particular, step 5 and step 8 in Algorithm 1 can be implemented according to the diagrams in Fig. S1a and Fig. S1b, respectively (please refer to the supplementary document). The soft-thresholding operation in step 6 may be implemented using two ReLU operations by recognizing that.<sup>5</sup>  $S_\lambda(x) = [x - \lambda]_+ - [-x - \lambda]_+$ . Similarly, (11) may be implemented according to Fig. S1c. Therefore, each iteration of Algorithm 1 admits a diagram representation, and repeating it  $L$  times yields an  $L$ -layer neural network (as shown in Fig. S2 in the supplementary document) which corresponds to executing Algorithm 1 with  $L$  iterations. For notational brevity, we concatenate the parameters in each layer and let  $\mathbf{f}^l = (\mathbf{f}_i^l)_{i=1}^C$ ,  $\zeta^l = (\zeta_i^l)_{i=1}^C$ ,  $\lambda^l = (\lambda_i^l)_{i=1}^C$  and  $\eta = (\eta_i)_{i=1}^C$ . We also concatenate  $\mathbf{y}^l$ 's,  $\mathbf{z}^l$ 's and  $\mathbf{g}^l$ 's by letting  $\mathbf{y}^l = (\mathbf{y}_i^l)_{i=1}^C$ ,  $\mathbf{z}^l = (\mathbf{z}_i^l)_{i=1}^C$  and  $\mathbf{g}^l = (\mathbf{g}_i^l)_{i=1}^C$ , respectively.

When the blur kernel has a large size (which may happen due to fast motion), it is desirable to alter the spatial size of the filter banks  $\{\mathbf{f}_i\}_i$  in different layers. In blind deblurring, kernel recovery is frequently performed stage-wise in a coarse-to-fine scheme: the kernel is initially estimated from a high-level summary of the image features, and then progressively refined by introducing lower-level details. For example, in [9] an initial kernel is obtained by masking out detrimental gradients, followed by iterative refinements. Other works [8], [16], [24] use a multi-scale pyramid implementation by decomposing the images into a Gaussian pyramid and progressively refine the kernel in each scale. We may integrate this scheme into Algorithm 1 by choosing large filters in early iterations, so that they are capable of capturing rather high-level features, and gradually decrease their size to let fine details emerge. Translating this into the network, we may expect the following relationship among the sizes of kernels in different layers:

$$\text{size of } \mathbf{f}_i^1 \geq \text{size of } \mathbf{f}_i^2 \geq \text{size of } \mathbf{f}_i^3 \geq \dots$$

In practice, large filters may be difficult to train due to the large number of parameters they contain. To address this issue, we produce large filters by cascading small  $3 \times 3$  filters, following the same principle as [71]. Formally, we set  $\mathbf{f}_i^L = \mathbf{w}_{i1}^L$  where  $\{\mathbf{w}_{i1}^L\}_{i=1}^C$  is a collection of  $3 \times 3$  filters, and recursively obtain  $\mathbf{f}_i^l$  by filtering  $\mathbf{f}_i^{l+1}$  through  $3 \times 3$  filters  $\{\mathbf{w}_{ij}^l\}_{i,j=1}^C$ :

$$\mathbf{f}_i^l \leftarrow \sum_{j=1}^C \mathbf{w}_{ij}^l * \mathbf{f}_j^{l+1}, \quad i = 1, 2, \dots, C.$$

Embedding the above into the network, we obtain the structure depicted in Fig. 1. Note that  $\mathbf{y}^l$  can now be obtained more efficiently by filtering  $\mathbf{y}^{l+1}$  through  $\mathbf{w}^l$ . Also note that  $\{\mathbf{w}_{ij}^l\}_{i,j=1}^C$  are to be learned as marked in Fig. 1. In addition, the conventional

<sup>5</sup>Its gradient can therefore be computed in a similar fashion to the widely used ReLU operator [70]

scale-space deblurring scheme can be implemented by choosing  $\mathbf{w}_{ij}^l$  as the Gaussian blur kernel when  $i = j$  and zeros otherwise. Experimental justification of cascaded filtering is provided in Fig. 3.

#### B. Training

In a given training set, for each blurred image  $\mathbf{y}_t^{\text{train}} (t = 1, \dots, T)$ , we let the corresponding sharp image and kernel be  $\mathbf{x}_t^{\text{train}}$  and  $\mathbf{k}_t^{\text{train}}$ , respectively. We do not train the parameter  $\varepsilon$  in step 8 of Algorithm 1 because it simply serves as a small constant to avoid division by zeros. We re-parametrize  $\zeta_i^l$  in step 6 of Algorithm 1 by letting  $b_i^l = \lambda_i^l \zeta_i^l$  and denote  $\mathbf{b}^l = (\mathbf{b}_i^l)_{i=1}^C, l = 1, \dots, L$ . The network outputs  $\tilde{\mathbf{x}}_t, \tilde{\mathbf{k}}_t$  corresponding to  $\mathbf{y}_t^{\text{train}}$  depend on the parameters  $\mathbf{w}^l, \mathbf{b}^l, \zeta^l, \beta^l, l = 1, 2, \dots, L$ . In addition,  $\tilde{\mathbf{x}}_t$  depends on  $\eta$ . We train the network to determine these parameters by solving the following optimization problem:

$$\begin{aligned} & \min_{\{\mathbf{w}^l, \mathbf{b}^l, \zeta^l, \beta^l\}_l, \eta, \{\tau_t\}_t} \sum_{t=1}^T \frac{\kappa_t}{2} \text{MSE} \\ & \quad \times \left( \tilde{\mathbf{k}}_t(\{\mathbf{w}^l, \mathbf{b}^l, \zeta^l, \beta^l\}_l), \mathcal{T}_{\tau_t} \{\mathbf{k}_t^{\text{train}}\} \right) \\ & \quad + \frac{1}{2} \text{MSE}(\tilde{\mathbf{x}}_t(\{\mathbf{w}^l, \mathbf{b}^l, \zeta^l, \beta^l\}_l, \eta), \\ & \quad \mathcal{T}_{-\tau_t} \{\mathbf{x}_t^{\text{train}}\}) \\ & \text{subject to } b_i^l \geq 0, \zeta_i^l \geq 0, \eta_i \geq 0, \beta^l \geq 0, l = 1, \dots, L, \\ & \quad i = 1, \dots, C, \quad (12) \end{aligned}$$

where  $\kappa_t > 0$  is a constant parameter which is fixed to  $\frac{\kappa_0}{\max_i |(\mathbf{k}_i^{\text{train}})|^2}$ , and we determined  $\kappa_0 = 10^5$  through cross-validation.  $\text{MSE}(\cdot, \cdot)$  is the (empirical) Mean-Square-Error loss function, and  $\mathcal{T}_\tau\{\cdot\}$  is the translation operator in 2D that performs a shift by  $\tau \in \mathbb{R}^2$ . The shift operation is used here to compensate for the inherent shifting ambiguity in blind deconvolution [24]. Specifically, the same blurred image  $\mathbf{y}$  can be produced by shifting the blur kernel by  $\tau$  and reversely shift the sharp image by  $\tau$ , i.e.,

$$\mathbf{y} = \mathbf{k} * \mathbf{x} = \mathcal{T}_\tau\{\mathbf{k}\} * \mathcal{T}_{-\tau}\{\mathbf{x}\},$$

if we do not consider the image boundary.

In the training process, when working on each mini-batch, we alternate between minimizing over  $\{\tau_t\}_t$  and the network parameters  $\{\mathbf{w}^l, \mathbf{b}^l, \zeta^l, \beta^l\}_l, \eta$ . Specifically, we first determine the optimal  $\{\tau_t\}_t$  efficiently via a grid search in the Fourier domain. We then update the network parameters by taking one stochastic gradient descent step; the analytic derivation of the gradient is included in Appendix A.<sup>6</sup> Effectively, this performs partial minimization of the loss w.r.t. the network parameters. Nevertheless, the updated parameters may not lie

<sup>6</sup>It is worth mentioning that both the loss function (12) and all the learnable parameters are real. Therefore, despite the introduction of complex coefficients in intermediate operations by invoking DFT, the gradients w.r.t. to each of the learnable parameters are real in principle. In our implementation, we drop out the imaginary part after taking the inverse DFT whenever the original signal is supposed to be real, which is consistent with [57].

within the feasible set; i.e., the non-negativity constraints over  $\{b_i^l, \zeta_i^l, \beta_i^l\}_{i,l}$ ,  $\{\eta_i\}_i$  may be violated. Therefore, we threshold out the negative coefficients of these parameters afterwards to enforce the non-negativity constraints. This strategy is similar to the gradient projection approach in nonlinear programming. We use the Adam algorithm [72] in the stochastic gradient descent step to accelerate the training speed. The learning rate is set to  $1 \times 10^{-3}$  initially and decayed by a factor of 0.5 every 20 epochs. We terminate training after 160 epochs. The parameters  $\{b_i^l\}_{i,l}$  are initialized to 0.02,  $\{\zeta_i^l\}_{i,l}$  initialized to 1,  $\{\beta_i^l\}_l$  initialized to 0, and  $\{\eta_i\}_i$  initialized to 20, respectively. These values are again determined through cross-validation. The upper part (feature extraction portion) of the network in Fig. 1 resembles a CNN with linear activations (identities) and thus we initialize the weights according to [73].

### C. Handling Color Images

For color images, the red, green and blue channels  $\mathbf{y}_r$ ,  $\mathbf{y}_g$ , and  $\mathbf{y}_b$  are blurred by the same kernel, and thus the following model holds instead of (1):

$$\mathbf{y}_c = \mathbf{k} * \mathbf{x}_c + \mathbf{n}_c, \quad c \in \{r, g, b\}.$$

To be consistent with existing literature, we modify  $\mathbf{w}^L$  in Fig. 1 to allow for multi-channel inputs. More specifically,  $\mathbf{y}^L$  is produced by the following formula:

$$\mathbf{y}_i^L = \sum_{c \in \{r, g, b\}} \mathbf{w}_{ic}^L * \mathbf{y}_c, \quad i = 1, \dots, C.$$

It is easy to check that, with  $\mathbf{w}^L$  and  $\mathbf{y}^L$  being replaced, all the components of the network can be left unchanged except for the module in Fig. S1 (see supplementary document). This is because (4) no longer holds and is modified to the following:

$$\sum_{c \in \{r, g, b\}} \mathbf{w}_{ic} * \mathbf{y}_c = \mathbf{k} * \left( \sum_{c \in \{r, g, b\}} \mathbf{w}_{ic} * \mathbf{x}_c \right) + \mathbf{n}'_i, \\ i = 1, \dots, C,$$

where  $\mathbf{n}'_i = \sum_{c \in \{r, g, b\}} \mathbf{w}_{ic} * \mathbf{n}$  represents Gaussian noise.

Problem (11) then becomes:

$$\{\widetilde{\mathbf{x}}_r, \widetilde{\mathbf{x}}_g, \widetilde{\mathbf{x}}_b\} \leftarrow \arg \min_{\mathbf{x}_r, \mathbf{x}_g, \mathbf{x}_b} \sum_{c \in \{r, g, b\}} \frac{1}{2} \left\| \mathbf{y}_c - \widetilde{\mathbf{k}} * \mathbf{x}_c \right\|_2^2 \\ + \sum_{i=1}^C \frac{\eta_i}{2} \left\| \sum_{c \in \{r, g, b\}} \mathbf{w}_{ic} * \mathbf{x}_c - \widetilde{\mathbf{g}}_i \right\|_2^2,$$

whose solution is given as follows:

$$\widetilde{\mathbf{x}}_r = \mathcal{F}^{-1} \left\{ \frac{\mathbf{m}_{rr} \odot \mathbf{b}_r + \mathbf{m}_{rg} \odot \mathbf{b}_g + \mathbf{m}_{rb} \odot \mathbf{b}_b}{\mathbf{d}} \right\}, \\ \widetilde{\mathbf{x}}_g = \mathcal{F}^{-1} \left\{ \frac{\mathbf{m}_{rg}^* \odot \mathbf{b}_r + \mathbf{m}_{gg} \odot \mathbf{b}_g + \mathbf{m}_{gb} \odot \mathbf{b}_b}{\mathbf{d}} \right\}, \\ \widetilde{\mathbf{x}}_b = \mathcal{F}^{-1} \left\{ \frac{\mathbf{m}_{rb}^* \odot \mathbf{b}_r + \mathbf{m}_{gb}^* \odot \mathbf{b}_g + \mathbf{m}_{bb} \odot \mathbf{b}_b}{\mathbf{d}} \right\},$$

where

$$\mathbf{m}_{rr} = \mathbf{c}_{gg} \odot \mathbf{c}_{bb} - |\mathbf{c}_{gb}|^2, \quad \mathbf{m}_{rg} = \mathbf{c}_{rb} \odot \mathbf{c}_{gb}^* - \mathbf{c}_{bb} \odot \mathbf{c}_{rg}, \\ \mathbf{m}_{rb} = \mathbf{c}_{rg} \odot \mathbf{c}_{gb} - \mathbf{c}_{gg} \odot \mathbf{c}_{rb}, \quad \mathbf{m}_{gg} = \mathbf{c}_{bb} \odot \mathbf{c}_{rr} - |\mathbf{c}_{rb}|^2, \\ \mathbf{m}_{gb} = \mathbf{c}_{rg}^* \odot \mathbf{c}_{rb} - \mathbf{c}_{rr} \odot \mathbf{c}_{gb}, \quad \mathbf{m}_{bb} = \mathbf{c}_{rr} \odot \mathbf{c}_{gg} - |\mathbf{c}_{rg}|^2.$$

Here,

$$\mathbf{c}_{cc'} = \sum_{i=1}^C \eta_i \widehat{\mathbf{w}}_{ic}^* \odot \widehat{\mathbf{w}}_{ic'} + |\widehat{\mathbf{k}}|^2 \delta_{cc'}, \quad c, c' \in \{r, g, b\}, \\ \mathbf{b}_c = \widehat{\mathbf{k}}^* \odot \widehat{\mathbf{y}}_c + \sum_{i=1}^C \eta_i \widehat{\mathbf{w}}_{ic}^* \odot \widehat{\mathbf{g}}_i, \quad c \in \{r, g, b\}, \\ \mathbf{d} = \left( \mathbf{c}_{gg} \odot \mathbf{c}_{rr} - |\mathbf{c}_{rg}|^2 \right) \odot \mathbf{c}_{bb} + 2\Re\{\mathbf{c}_{rb}^* \odot \mathbf{c}_{rg} \odot \mathbf{c}_{gb}\} \\ - |\mathbf{c}_{gb}|^2 \odot \mathbf{c}_{rr} - |\mathbf{c}_{rb}|^2 \odot \mathbf{c}_{gg},$$

and  $\delta_{cc'}$  is the Kronecker delta function. These analytical formulas may be represented using diagrams similar to Fig. S1 (see supplementary document) and embedded into a network.

## IV. EXPERIMENTAL VERIFICATION

### A. Experimental Setups

#### 1) Datasets, Training and Test Setup:

- **Training for linear kernels:** For the images we used the Berkeley Segmentation Data Set 500 (BSDS500) [74] which is a large dataset of 500 natural images that is explicitly divided into disjoint training, validation and test subsets. Here we use 300 images for training by combining the training and validation images.

We generated 256 linear kernels by varying the length and angle of the kernels (refer to Section IV-C for details).

- **Training for nonlinear kernels:** We used the Microsoft COCO [75] dataset which is a large-scale object detection, segmentation, and captioning dataset containing 330 K images.

*Nonlinear kernels:* We generated around 30,000 real world kernels by recording camera motion trajectories (refer to Sec. IV-D for details).

- **Testing for the linear kernel experiments:** We use 200 images from the test portion of BSDS500 as test images and we randomly choose four kernels of different angle and length as test kernels.
- **Testing for the nonlinear kernel experiments:** We test on two benchmark datasets specifically developed for evaluating blind deblurring with non-linear kernels: 1.) 4 images and 8 kernels by Levin *et al.* [23] and 2.) 80 images and 8 kernels from Sun *et al.* [12].

#### 2) Comparisons Against State-of-the-Art Methods:

We compare against six methods:

- Perrone *et al.* [24] - a representative iterative blind image deblurring method based on total-variation minimization, which demonstrated state-of-the-art performance amongst traditional iterative methods. (TPAMI 2016)



TABLE I  
EFFECTS OF DIFFERENT VALUES OF LAYER  $L$

Number of layers	6	8	10	12
PSNR (dB)	26.55	26.94	27.30	27.35
RMSE ( $\times 10^{-3}$ )	1.96	2.06	1.67	1.66

TABLE II  
EFFECTS OF DIFFERENT VALUES OF NUMBER OF FILTERS  $C$

Number of filters	8	16	32
PSNR (dB)	26.55	27.30	27.16
RMSE ( $\times 10^{-3}$ )	1.99	1.67	1.93

- Chakrabarti *et al.* [32] - one of the first neural network blind image deblurring methods. (CVPR 2016)
- Nah *et al.* [37] - a recent deep learning method based on the state-of-the-art ResNet [76]. (CVPR 2017)
- Kupyn *et al.* [77] - a recent deep learning method based on the state-of-the-art generative adversarial networks (GAN) [78]. (CVPR 2018)
- Tao *et al.* [31] - a recent deep learning method based on the scale-recurrent network. (CVPR 2018)
- Xu *et al.* [33] - a recent state of the art deep learning method focused on motion kernel estimation. (TIP 2018)

3) *Quantitative Performance Measures*: To quantitatively assess the performance of various methods in different scenarios, we use the following metrics:

- Peak-Signal-to-Noise Ratio (PSNR);
- Improvement in Signal-to-Noise-Ratio (ISNR), which is given by  $\text{ISNR} = 10 \log_{10}(\frac{\|\mathbf{y} - \mathbf{x}\|_2^2}{\|\tilde{\mathbf{x}} - \mathbf{x}\|_2^2})$  where  $\tilde{\mathbf{x}}$  is the reconstructed image;
- Structural Similarity Index (SSIM) [79];
- (Empirical) Root-Mean-Square Error (RMSE) computed between the estimated kernel and the ground truth kernel.

We note that for selected methods, RMSE numbers (against the ground truth kernel) are *not* reported in Tables III, IV and V because those methods directly estimate just the deblurred image (and not the blur kernel).

### B. Ablation Study of the Network

To provide insights into the design of our network, we first carry out a series of experimental studies to investigate the influence of two key design variables: 1.) the number of layers  $L$ , and 2.) the number of filters  $C$ . We monitor the performance using PSNR and RMSE. The results in Tables I and II are for linear kernels with the same training-test configuration as in Section IV-A. The trends over  $L$  and  $C$  were found to be similar for non-linear kernels as well.

We first study the influence of different number of layers  $L$ . We alter  $L$ , i.e. the proposed DUBLID is learned as in Section III-B, each time as  $L$  is varied. Table I summarizes the numerical scores corresponding to different  $L$ . Clearly, the network performs better with increasing  $L$ , consistent with common observations [76], [80]. In addition, the network performance

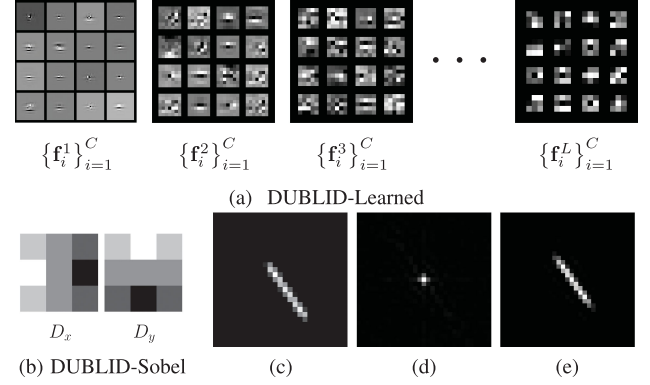


Fig. 2. Comparison of learned filters with analytic Sobel filters: (a) DUBLID learned filters. (b) Sobel filters that are commonly employed in traditional iterative blind deblurring algorithms. (c) An example motion blur kernel. (d) Reconstructed kernel using Sobel filters and (e) using learned filters.

improves marginally after  $L$  reaches 10. We thus fix  $L = 10$  subsequently. For all results in Table I, the number of filters  $C$  is fixed to 16.

We next study the effects of different values of  $C$  in a similar fashion. The network performance over different choices of  $C$  is summarized in Table II. It can be seen that the network performance clearly improves as  $C$  increases from 8 to 16. However, the performance slightly drops when  $C$  increases further, presumably because of overfitting. We thus fix  $C = 16$  henceforth.

To corroborate the network design choices made in Section III-A, we illustrate DUBLID performance for different filter choices. We first verify the significance of learning the filters  $\{\mathbf{w}^l\}_l$  (and in turn  $\{\mathbf{f}_i^l\}_{i,l}$ ) and compare the performance with a typical choice of analytical filters, the Sobel filters, in Fig. 2. Note that by employing Sobel filters, the network reduces to executing TV-based deblurring but for a small number of iterations, which coincides with the number of layers  $L$ . For fairness in comparison, the fixed Sobel filter version of DUBLID (called DUBLID-Sobel) is trained exactly as in Section III-B to optimize other parameters. As Fig. 2 reveals, DUBLID-Sobel is unable to accurately recover the kernel. Indeed, such phenomenon has been observed and analytically studied by Levin *et al.* [23], where they point out that traditional gradient-based approaches can easily get stuck at a delta solution. To gain further insight, we visualize the learned filters as well as the Sobel filters in Fig. 2(a) and Fig. 2(b). The learned filters demonstrate richer variations than known analytic (Sobel) filters and are able to capture higher-level image features as  $l$  grows. This enables the DUBLID network to better recover the kernel coefficients subsequently. Quantitatively, the PSNR achieved by DUBLID-Sobel for  $L = 10$  and  $C = 16$  on the same training-test set up is 18.60 dB, which implies that DUBLID achieves a 8.7 dB gain by explicitly optimizing filters in a data-adaptive fashion.

Finally, we show the effectiveness of cascaded filtering. To this end, we compare with the alternative scheme of fixing the size of  $\{\mathbf{f}_i^l\}_{i,l}$  by restricting  $\{\mathbf{w}^l\}_l$  to be of size  $1 \times 1$  whenever  $l < L$ . The results are shown in Fig. 3. By employing learnable



TABLE III  
QUANTITATIVE COMPARISON OVER AN AVERAGE OF 200 IMAGES AND 4 KERNELS. THE BEST SCORES ARE IN BOLD FONTS

Metrics	DUBLID	Perrone <i>et al.</i> [24]	Tao <i>et al.</i> [31]	Nah <i>et al.</i> [37]	Xu <i>et al.</i> [33]	Kupyn <i>et al.</i> [77]
PSNR (dB)	<b>27.30</b>	22.23	25.32	24.82	24.02	23.98
ISNR (dB)	<b>4.45</b>	2.06	2.42	1.92	1.12	1.05
SSIM	<b>0.88</b>	0.76	0.83	0.80	0.78	0.78
RMSE ( $\times 10^{-3}$ )	<b>1.67</b>	5.21	—	—	2.40	—

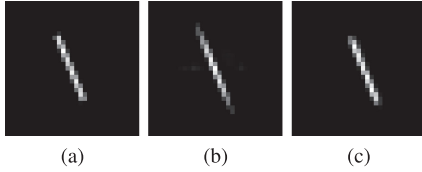


Fig. 3. The effectiveness of cascaded filtering: (a) a sample motion kernel. (b) Reconstructed kernel by fixing all  $f_i^l$ 's to be of size  $3 \times 3$ , which can be implemented by enforcing  $w_{ij}^l$  to be of size  $1 \times 1$  whenever  $l < L$ . (c) Reconstructed kernel using the cascaded filtering structure in Fig. 1.

filters, the network becomes capable of capturing the correct directions of blur kernels as shown in Fig. 3(b). In the absence of cascaded filtering though, the recovered kernel is still coarse — a limitation that is overcome by using cascaded filtering, verified in Fig. 3(c).

### C. Evaluation on Linear Kernels

We use the training and validation portions of the BSDS500 [74], [81] dataset as training images. The linear motion kernels are generated by uniformly sampling 16 angles in  $[0, \pi]$  and 16 lengths in  $[5, 20]$ , followed by 2D spatial interpolation. This gives a total number of 256 kernels. We then generate  $T = 256 \times 300$  blurred images by convolving each kernel with each image and adding white Gaussian noise with standard deviation 0.01 (suppose the image intensity is in the range  $[0, 1]$ ). Examples of training samples (images and kernels) are shown in Fig. 4. We use 200 images from the test portion of the BSDS500 dataset [81] for evaluation. We randomly choose angles in  $[0, \pi]$  and lengths in  $[5, 20]$  to generate 4 test kernels. The images and kernels are convolved to synthesize 800 blurred images. White Gaussian noise (again with standard deviation 0.01) is also added.

Note that some of the state of the art methods are only designed to recover the kernels, including [33] and [24]. To get the deblurred image, the non-blind method in [13] is used consistently. The scores are averaged and summarized in Table III. The RMSE values are computed over kernels, and smaller values indicate more accurate recoveries. For all other metrics on images, higher scores generally imply better performance. We do not include results from Chakrabarti *et al.* [32] here because that method works on grayscale images only. Table III confirms that DUBLID outperforms competing state-of-the-art algorithms by a significant margin.

Fig. 5 shows four example images and kernels for a qualitative comparison. Other than DUBLID, Perrone *et al.* [24] and Nah *et al.* [37], Tao *et al.* [31] are also included as representatives



Fig. 4. Examples of images and kernels used for training. (a) The sharp images are collected from the BSDS500 dataset [81]. (b) The blur kernels are linear motion of different lengths and angles. (c) The blurred images are synthesized by convolving the sharp images with the blur kernels.

of iterative methods and deep learning methods, respectively. Although [24] can roughly infer the directions of the blur kernels, the recovered coefficients clearly differ from the groundtruth as evidenced by the spread-out branches. Consequently, destroyed local structures and false colors are observed in the reconstructed images. Nah *et al.*'s method [37] does not suffer from false colors, yet the recovered images appear blurry. In contrast, DUBLID recovers kernels close to the groundtruth, and produces significantly fewer visually objectionable artifacts in the recovered images.

Finally, consistent with [20] and [12], we present the cumulative histogram of error ratios [23] in Fig. 7 and Fig. 8, respectively, as a more comprehensive quantitative comparison. The error ratios are computed by dividing blind deconvolution errors with the corresponding non-blind deconvolution errors. Generally, an error ratio close to 1 indicates decent performance, while an error ratio above 3 indicates visually implausible results. Overall, DUBLID achieves leading performance, as it has the highest bars in most scenarios.

### D. Evaluation on Non-Linear Kernels

It has been observed in several previous works [23], [82] that realistic motion kernels often have non-linear shapes due to irregular camera motions, such as those shown in Fig. 9.

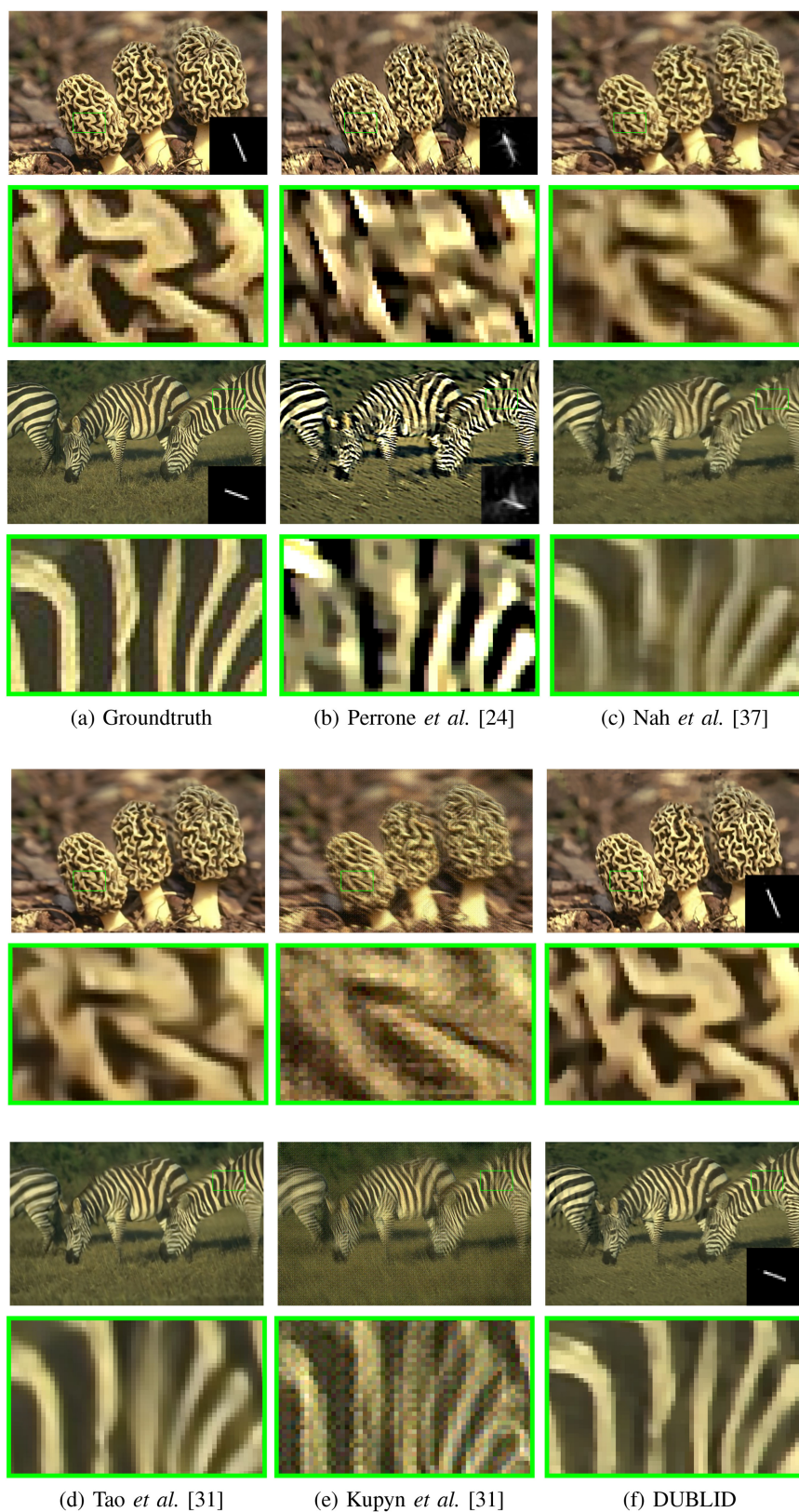


Fig. 5. Qualitative comparisons on the BSDS500 [81] dataset. The blur kernels are placed at the right bottom corner. DUBLID recovers the kernel at higher accuracy and therefore the estimated images are more faithful to the groundtruth.



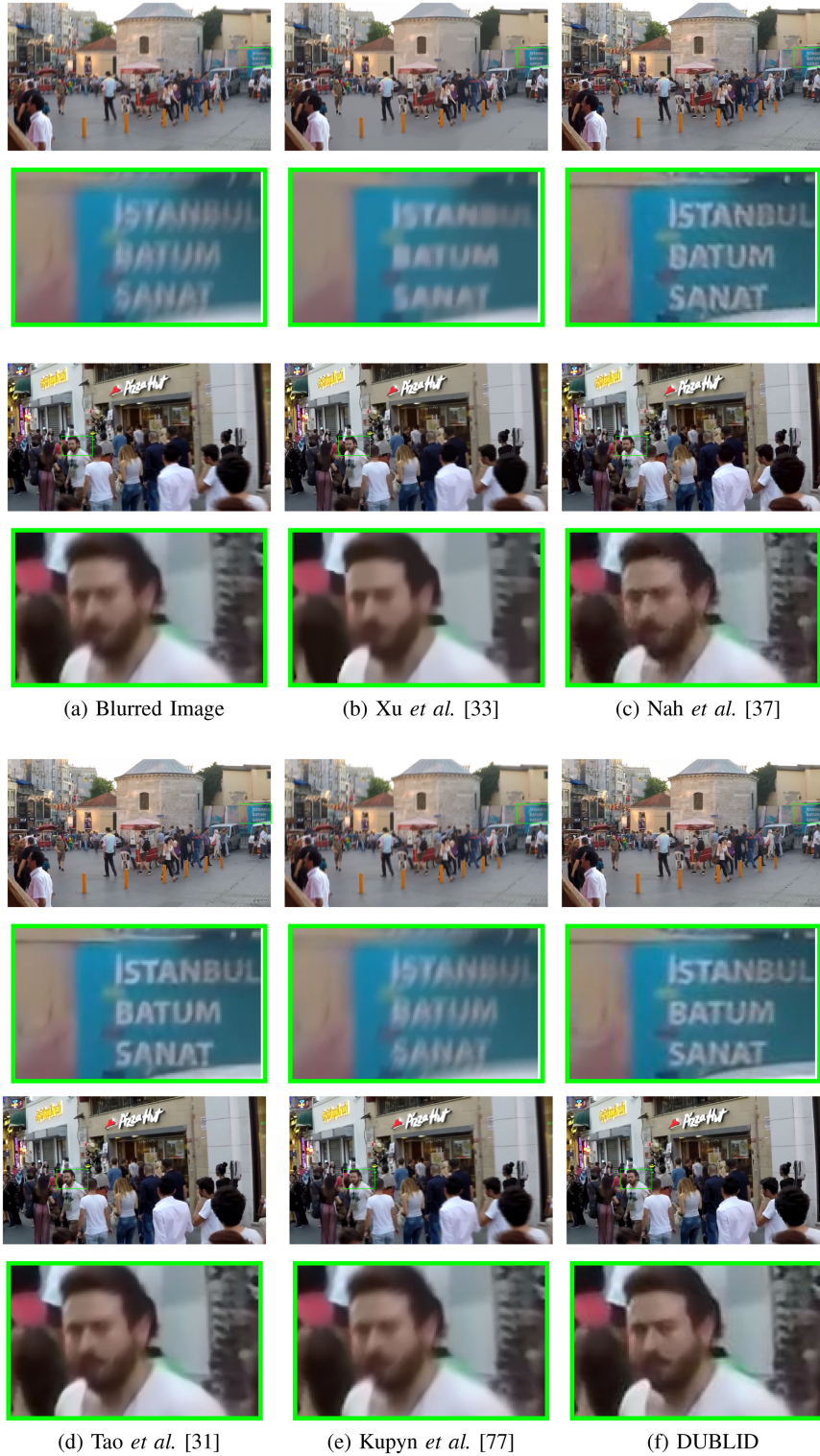


Fig. 6. Qualitative comparisons on the GoPro [37] dataset. Despite the fact that DUBLID is designed for uniform blur, its performance in real world non-uniform blurred images is acceptable.

Therefore, the capability to handle such kernels is crucial for a blind motion deblurring method.

We generate training kernels by interpolating the paths provided by [82] and those created by ourselves: specifically, we

record the camera motion trajectories using the Vicon system, and then interpolate the trajectories spatially to create motion kernels. We further augment these kernels by scaling over 4 different scales and rotating over 8 directions. In this way,

TABLE IV  
QUANTITATIVE COMPARISON OVER AN AVERAGE OF 4 IMAGES AND 8 KERNELS FROM [23]

	DUBLID	Perrone <i>et al.</i> [24]	Tao <i>et al.</i> [31]	Nah <i>et al.</i> [37]	Chakrabarti <i>et al.</i> [32]	Xu <i>et al.</i> [33]	Kupyn <i>et al.</i> [77]
PSNR (dB)	<b>27.15</b>	26.79	25.61	24.51	23.21	26.75	23.98
ISNR (dB)	<b>3.79</b>	3.63	2.45	1.35	0.06	3.59	0.43
SSIM	<b>0.89</b>	<b>0.89</b>	0.85	0.81	0.81	<b>0.89</b>	0.80
RMSE ( $\times 10^{-3}$ )	3.87	<b>3.83</b>	—	—	4.33	3.98	—

TABLE V  
QUANTITATIVE COMPARISON OVER AN AVERAGE OF 80 IMAGES AND 8 NONLINEAR MOTION KERNELS FROM [12]

	DUBLID	Perrone <i>et al.</i> [24]	Tao <i>et al.</i> [31]	Nah <i>et al.</i> [37]	Chakrabarti <i>et al.</i> [32]	Xu <i>et al.</i> [33]	Kupyn <i>et al.</i> [77]
PSNR (dB)	<b>29.91</b>	29.82	26.43	26.98	29.86	26.55	25.84
ISNR (dB)	<b>4.11</b>	4.02	0.30	0.86	4.06	0.43	0.15
SSIM	<b>0.93</b>	0.92	0.83	0.85	0.91	0.87	0.83
RMSE ( $\times 10^{-3}$ )	<b>2.33</b>	2.68	—	—	2.72	2.79	—

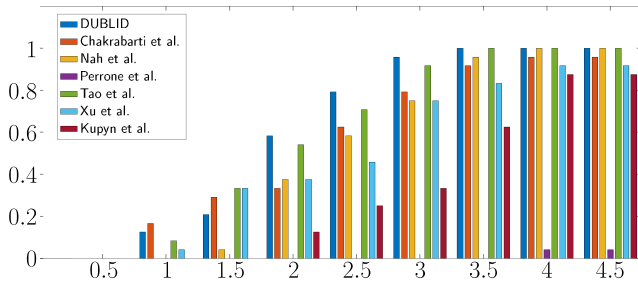


Fig. 7. Cumulative histogram of error ratios for the Levin *et al.*'s dataset [23]. Bar height equals the percentage of images with error ratio smaller or equal to the values in the horizontal axis. High bars indicate better performance.

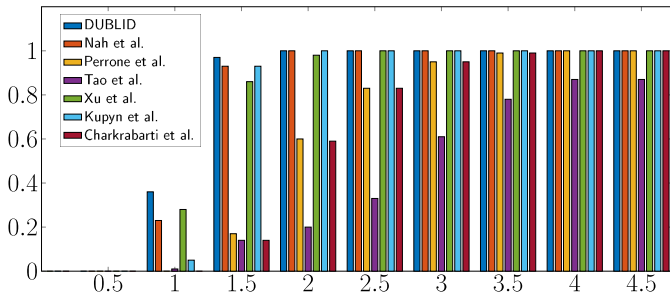


Fig. 8. Cumulative histogram of error ratios for the Sun *et al.*'s dataset [12]. Bar height equals the percentage of images with error ratio smaller or equal to the values in the horizontal axis. High bars indicate better performance.



Fig. 9. Examples of realistic non-linear kernels [23].

we build around 30,000 training kernels in total.<sup>7</sup> The blurred images for training are synthesized by randomly picking a kernel and convolving with it. Gaussian noise of standard deviation

<sup>7</sup>To re-emphasize, all learning based methods use the same training-test configuration for fairness in comparison.

0.01 is then added. We use the standard image set from [23] (comprising 4 images and 8 kernels) and from [12] (comprising 80 images and 8 kernels) as the test sets. The average scores for both datasets are presented in Table IV and Table V, respectively. In both datasets, DUBLID emerges overall as the best method. The method of Chakrabarti *et al.* [32] performs second best in Table V. In Table IV, Perrone *et al.* [24] and the recent deep learning method of Xu *et al.* [33] perform comparably and mildly worse than DUBLID. DUBLID however achieves the deblurring at a significantly lower computational cost as verified in Section IV-F.

Visual examples are shown in Fig. 10 for qualitative comparisons. It can be clearly seen that DUBLID is capable of more faithfully recovering the kernels, and hence produces reconstructed images of higher visual quality. In particular, DUBLID preserves local details better as shown in the zoom boxes of Fig. 10 while providing sharper images than Nah *et al.* [37], Chakrabarti *et al.* [32] and Kupyn *et al.* [77]. Finally, DUBLID is free of visually objectionable artifacts observed in Perrone *et al.* [24] and Xu *et al.* [33].

#### E. Evaluation on Real-World Blur Datasets

In some cases, blur in real-world images can be non-uniform if the camera motion is not planar or there are moving objects in the scene [45]. While a comprehensive handling of non-uniform blur is beyond the scope of this paper, we present a few results to demonstrate the potential of DUBLID.

We use the same training configuration as in Sec. IV-D and evaluate on the test portion of GoPro dataset [37] with 11 video sequences of 100 frames. In Fig. 6, we observe that the deblurred image through DUBLID is comparable to that of Tao *et al.* [31], which performs the best among all the compared methods. The quantitative results are presented in Table VI. From Table VI, we can observe that, although DUBLID is neither specifically designed nor trained towards non-uniform blur, it achieves performance close to the best performing method (Tao *et al.*). Overall, DUBLID achieves the highest performance (both visually and quantitatively) in all the cases under uniform blur,



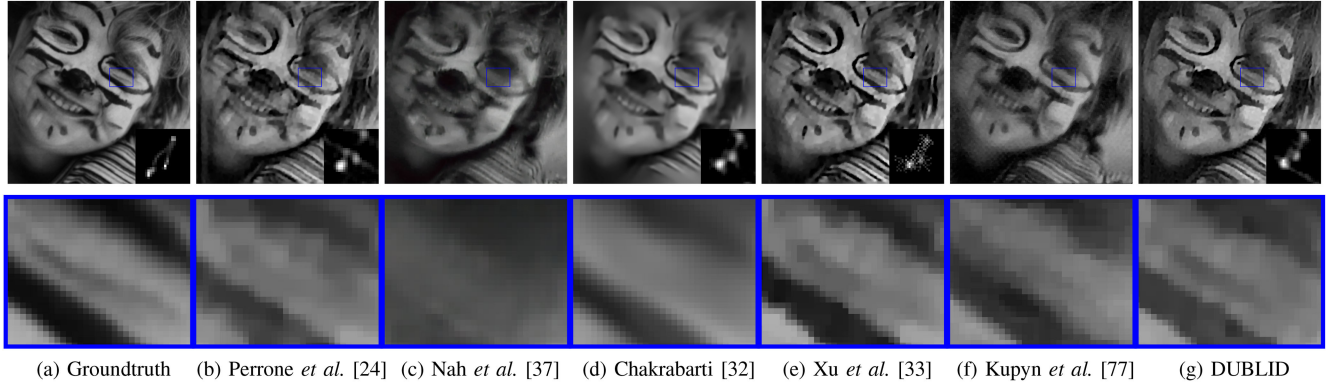


Fig. 10. Qualitative comparisons on the dataset from [23]. The blur kernels are placed at the right bottom corner. DUBLID generates fewer artifacts and preserves more details than competing state of the art methods.

TABLE VI  
QUANTITATIVE COMPARISON OVER THE TEST IMAGES  
FROM GoPro [37] DATASET

	DUBLID	Tao [31]	Nah [37]	Xu [33]	Kupyn [77]
PSNR (dB)	29.96	30.26	29.08	26.39	27.64
SSIM	0.93	0.93	0.91	0.89	0.88

TABLE VII  
QUANTITATIVE COMPARISON OVER KOHLER *ET AL.*'S DATASET [82].  
THE BEST SCORES ARE IN BOLD FONT

	PSNR (dB)	SSIM	LPLIPS	Time
Kim <i>et al.</i> [80]	24.68	0.794	0.533	1 hr
Sun <i>et al.</i> [12]	25.22	0.774	0.514	20 min
Nah <i>et al.</i> [37]	26.48	0.808	0.495	3.09 sec
Tao <i>et al.</i> [31]	26.75	0.837	0.481	1.87 sec
Li <i>et al.</i> [85]	<b>30.14</b>	0.889	0.357	379 sec
DUBLID	29.83	<b>0.892</b>	<b>0.305</b>	<b>0.08 sec</b>

TABLE VIII  
QUANTITATIVE COMPARISON OVER LAI *ET AL.*'S DATASET [45]

	Whyte [35]	Xu [11]	Pan [16]	Kupyn [77]	Tao [31]	Li [85]	DUBLID
PSNR (dB)	15.72	18.51	19.64	16.36	16.16	20.85	20.32

and a performance close to state of the art for non-uniform blur. We believe this gap can also be overcome via explicit extensions of DUBLID for non-uniform blur, such as generalizing the model in (1), which forms a viable future research direction.

To further confirm the effectiveness of our method under large motions and real world blurs, we show comparisons with state-of-the-art methods on two additional datasets by Kohler *et al.* [82] and Lai *et al.* [45]. The numerical scores are summarized in Tables VII and VIII. We are only including PSNR values in Table VIII because in some of the competing works, only PSNR scores are and code is unavailable. For competing methods, it may also be verified that our measures are entirely consistent with what has been already reported before in published work. Note that Tables VII and VIII focus on comparing DUBLID against state of the art known methods for these datasets – hence new methods may be seen in these Tables. For faithful reproduction and consistency with past

work, the training-test configuration and evaluation strategies for the results in Table VII and VIII are consistent with the ones in [31] and [83] respectively. In addition to PSNR and SSIM scores, we include for the Kohler *et al.* [82] dataset, the Learned Perceptual Image Patch Similarity (LPIPS) [84], which measures the perceptual similarity between image pairs by comparing their features extracted through deep networks. Typically, lower LPIPS scores indicate higher visual similarity to the groundtruth and hence better performance. Visual examples are provided further in Fig. 11. Compared to Pan *et al.* and Tao *et al.*, the results for DUBLID clearly deliver fewer blurry or ringing artifacts, as also evidenced by its superior numerical scores. As with Li *et al.* [83], DUBLID achieves competitive visual quality and comparable numerical scores. Specifically, the PSNR value for DUBLID is slightly below that of Li *et al.*, while its SSIM and LPIPS values, which arguably provide better measures of visual quality, are superior. In addition, the execution speed for DUBLID is almost 5000 times faster than Li *et al.*, which implies that it is better suited for practical applications.

#### F. Computational Comparisons Against State-of-the-Art

Table IX summarizes the execution (inference) times of each method for processing a typical blurred image of resolution  $480 \times 320$  and a blur kernel of size  $31 \times 31$ . The number of parameters for DUBLID is estimated as follows: for  $3 \times 3$  filters  $w_{ij}$ , there are a total of  $L = 10$  layers and in each layer there are  $C^2 = 16 \times 16$  filters, which contribute to  $3 \times 3 \times 16 \times 16 \times 10 \approx 2.3 \times 10^4$  parameters. Other parameters have negligible dimensions compared with  $w_{ij}$  and thus do not contribute significantly.

We include measurements of running time on both CPU and GPU. The – symbol indicates inapplicability. For instance, Chakrabarti *et al.* [32] and Nah *et al.* [37] only provide GPU implementations of their work and likewise Perrone *et al.*'s iterative method [24] is only implemented on a CPU. Specifically, the two benchmark platforms are: 1.) Intel Core i7–6900 K, 3.20 GHz CPU, 8 GB of RAM, and 2.) an NVIDIA TITAN X GPU. The results in Table IX deliver two messages. First, the deep/neural

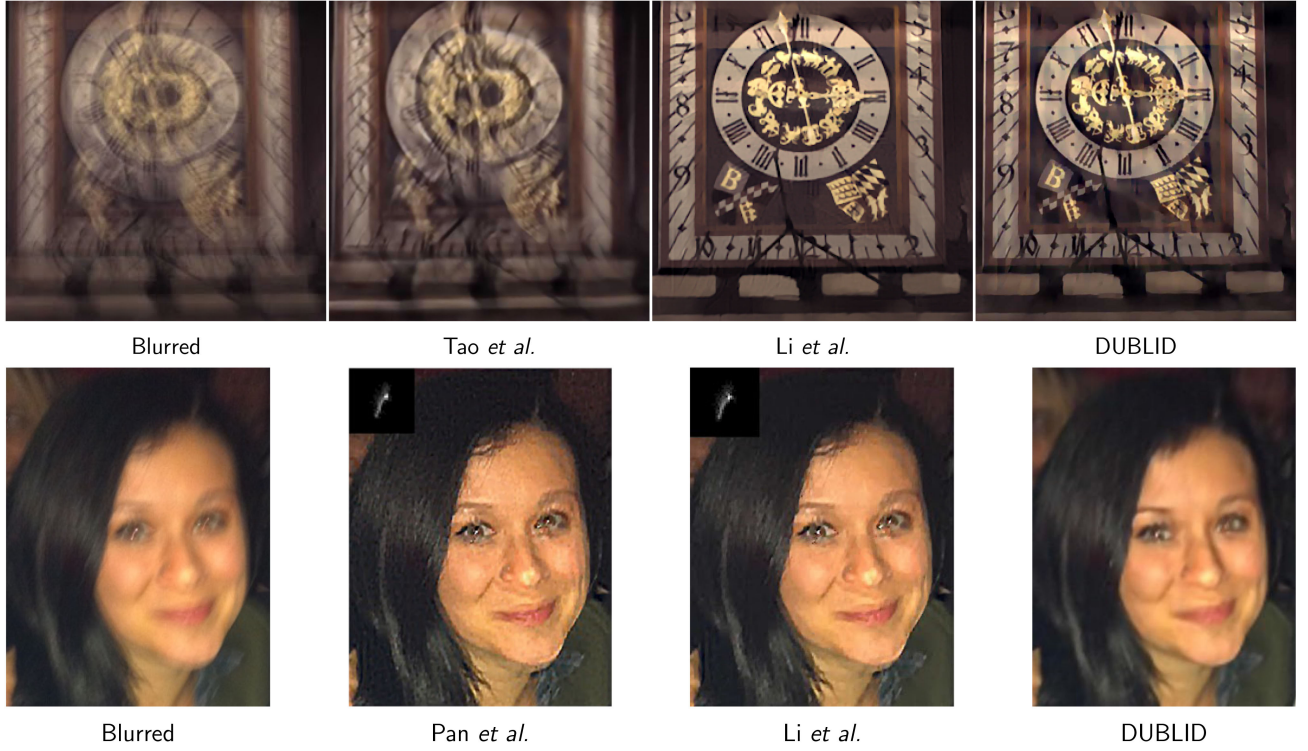


Fig. 11. Qualitative comparisons on datasets from Kohler *et al.*'s [82] (top) and Lai *et al.* [45] (bottom). Kohler *et al.*'s dataset includes images blurred by large motion, while Lai *et al.*'s dataset comprises real blurred images.

TABLE IX  
RUNNING TIME COMPARISONS OVER DIFFERENT METHODS. THE IMAGE SIZE IS  $480 \times 320$  AND THE KERNEL SIZE IS  $31 \times 31$

	DUBLID	Tao <i>et al.</i> [31]	Chakrabarti <i>et al.</i> [32]	Nah <i>et al.</i> [37]	Perrone <i>et al.</i> [24]	Xu <i>et al.</i> [33]	Kupyn <i>et al.</i> [77]
CPU Time (s)	<b>1.47</b>	—	—	—	1462.90	6.89	10.29
GPU Time (s)	<b>0.05</b>	0.15	227.80	7.32	—	2.01	0.13
Number of Parameters	<b><math>2.3 \times 10^4</math></b>	$3.8 \times 10^6$	$1.1 \times 10^8$	$2.3 \times 10^7$	—	$6.0 \times 10^6$	$1.2 \times 10^7$

network based methods are faster than their iterative algorithm counterparts, which is to be expected. Second, amongst the deep neural net methods DUBLID runs significantly faster than the others on both GPU and CPU, largely because it has significantly fewer parameters as seen in the final row of Table IX. Note that the number of parameters for competing deep learning methods are computed based on the description in their respective papers.

## V. CONCLUSION

We propose a Deep Unrolling approach for BLind Deblurring (DUBLID). Our method is based on recasting a generalized TV-regularized algorithm into a neural network, and optimizing its parameters via a custom designed backpropagation procedure. Unlike most existing neural network approaches, our technique has the benefit of interpretability, while sharing the performance benefits of modern neural network approaches. While some existing techniques excel for the case of linear kernels and others for non-linear, our method is versatile across a variety of scenarios and kernel choices — as is verified both visually and quantitatively. Furthermore, DUBLID requires much fewer

parameters leading to significant computational benefits over iterative methods as well as competing deep learning techniques.

## APPENDIX GRADIENT COMPUTATION BY BACK-PROPAGATION

Here we develop the back-propagation rules for computing the gradients of DUBLID. We will use  $\mathbf{F}$  to denote the DFT operator and  $\mathbf{F}^*$  its adjoint operator,  $\mathbf{1}$  is a vector whose entries are all ones, and  $\mathbf{I}$  refers to the identity matrix. The symbol  $\mathcal{I}_{\Omega}$  means an indicator vectors and  $\text{diag}(\cdot)$  embeds the vector into a diagonal matrix. The operators  $\mathbf{P}_g$  and  $\mathbf{P}_k$  are projections that restrict the operand into the domain of the image and the kernel, respectively. Let  $\mathcal{L}$  be the cost function defined in (12).

We derive its gradients w.r.t. its variables using the chain rule as follows:

$$\begin{aligned}\nabla_{\mathbf{w}_i^l} \mathcal{L} &= \nabla_{\mathbf{w}_i^l} \mathbf{y}_i^l \nabla_{\mathbf{y}_i^l} \mathcal{L} = \mathbf{R}_{\mathbf{w}_i^l} \mathbf{F} \text{diag} \left( \widehat{\mathbf{y}_i^{l+1}} \right) \mathbf{F}^* \nabla_{\mathbf{y}_i^l} \mathcal{L}, \\ \nabla_{\zeta_i^l} \mathcal{L} &= \nabla_{\zeta_i^l} \mathbf{z}_i^{l+1} \nabla_{\mathbf{z}_i^{l+1}} \mathcal{L}\end{aligned}$$

$$\begin{aligned}
&= \left[ \frac{\hat{\mathbf{k}}^l \odot (\hat{\mathbf{k}}^l \odot \hat{\mathbf{g}}_i^l - \hat{\mathbf{y}}_i^l)}{\left( \|\hat{\mathbf{k}}^l\|^2 + \zeta_i^l \right)^2} \right]^T \\
&\quad \times \mathbf{F}^* \left( \mathcal{I}_{\{\|\mathbf{p}_{\mathbf{g}}^{l+1}\| > b_i^l\}} \odot \nabla_{\mathbf{z}_i^{l+1}} \mathcal{L} \right), \\
\nabla_{b_i^l} \mathcal{L} &= \nabla_{b_i^l} \mathbf{z}_i^{l+1} \nabla_{\mathbf{z}_i^{l+1}} \mathcal{L} \\
&= \left( \mathcal{I}_{\{\mathbf{g}_i^{l+1} < -b_i^l\}} - \mathcal{I}_{\{\mathbf{g}_i^{l+1} > b_i^l\}} \right)^T \nabla_{\mathbf{z}_i^{l+1}} \mathcal{L},
\end{aligned}$$

where  $\mathbf{R}_{\mathbf{w}_i^l}$  is the operator that extracts the components lying in the support of  $\mathbf{w}_i^l$ . Again using the chain rule,

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{k}^l} &= \frac{\partial \mathcal{L}}{\partial \mathbf{z}_i^{l+1}} \frac{\partial \mathbf{z}_i^{l+1}}{\partial \mathbf{k}^l}, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{z}_i^l} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}_i^{l+1}} \frac{\partial \mathbf{z}_i^{l+1}}{\partial \mathbf{z}_i^l} + \frac{\partial \mathcal{L}}{\partial \mathbf{k}^l} \frac{\partial \mathbf{k}^l}{\partial \mathbf{z}_i^l}, \\
\frac{\partial \mathcal{L}}{\partial \mathbf{y}_i^l} &= \frac{\partial \mathcal{L}}{\partial \mathbf{z}_i^{l+1}} \frac{\partial \mathbf{z}_i^{l+1}}{\partial \mathbf{y}_i^l} + \frac{\partial \mathcal{L}}{\partial \mathbf{k}^{l+1}} \frac{\partial \mathbf{k}^{l+1}}{\partial \mathbf{y}_i^l} + \frac{\partial \mathcal{L}}{\partial \mathbf{y}_i^{l-1}} \frac{\partial \mathbf{y}_i^{l-1}}{\partial \mathbf{y}_i^l}. \quad (13)
\end{aligned}$$

Detailed derivation of each individual term in (13) can be found in the supplementary document.

## REFERENCES

- [1] D. Kundur and D. Hatzinakos, "Blind image deconvolution," *IEEE Signal Process. Mag.*, vol. 13, no. 3, pp. 43–64, May 1996.
- [2] W. H. Richardson, "Bayesian-based iterative method of image restoration," *J. Opt. Soc. Amer.*, vol. 62, no. 1, pp. 55–59, 1972.
- [3] L. A. Shepp and Y. Vardi, "Maximum likelihood reconstruction for emission tomography," *IEEE Trans. Med. Imag.*, vol. 1, no. 2, pp. 113–122, Oct. 1982.
- [4] G. R. Ayers and J. Ch. Dainty, "Iterative blind deconvolution method and its applications," *Opt. Lett.*, vol. 13, no. 7, pp. 547–549, 1988.
- [5] T. F. Chan and C. K. Wong, "Total variation blind deconvolution," *IEEE Trans. Image Process.*, vol. 7, no. 3, pp. 370–375, Mar. 1998.
- [6] N. Joshi, R. Szeliski, and D. J. Kriegman, "PSF estimation using sharp edge prediction," in *Proc. IEEE Conf. CVPR*, Jun. 2008, pp. 1–8.
- [7] Q. Shan, J. Jia, and A. Agarwala, "High-quality motion deblurring from a single image," in *Proc. ACM SIGGRAPH*, 2008, pp. 1–10.
- [8] S. Cho and S. Lee, "Fast motion deblurring," in *Proc. ACM SIGGRAPH Asia*, 2009, pp. 1–6.
- [9] L. Xu and J. Jia, "Two-phase kernel estimation for robust motion deblurring," in *Proc. Eur. Conf. Comput. Vision*, 2010, pp. 157–170.
- [10] D. Krishnan, T. Tay, and R. Fergus, "Blind deconvolution using a normalized sparsity measure," in *Proc. IEEE Conf. CVPR*, 2011, pp. 233–240.
- [11] L. Xu, S. Zheng, and J. Jia, "Unnatural L0 sparse representation for natural image deblurring," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Jun. 2013, pp. 1107–1114.
- [12] L. Sun, S. Cho, J. Wang, and J. Hays, "Edge-based blur kernel estimation using patch priors," in *Proc. IEEE Int. Conf. Comput. Photography*, Apr. 2013, pp. 1–8.
- [13] J. Pan, Z. Hu, Z. Su, and M. H. Yang, "\$\mathcal{L}\_0\$-Regularized intensity and gradient prior for deblurring text images and beyond," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 2, pp. 342–355, Feb. 2017.
- [14] J.-F. Cai, H. Ji, C. Liu, and Z. Shen, "Framelet-Based blind motion deblurring from a single image," *IEEE Trans. Image Process.*, vol. 21, no. 2, pp. 562–572, Feb. 2012.
- [15] Sh. Xiang, G. Meng, Y. Wang, Ch. Pan, and Ch. Zhang, "Image deblurring with coupled dictionary learning," *Int. J. Comput. Vision*, vol. 114, no. 2–3, pp. 248–271, Sep. 2015.
- [16] J. Pan, D. Sun, H. Pfister, and M. H. Yang, "Deblurring images via dark channel prior," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 10, pp. 2315–2328, 1 Oct. 2018.
- [17] Y. Yan, W. Ren, Y. Guo, R. Wang, and X. Cao, "Image deblurring via extreme channels prior," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 4003–4011.
- [18] S. Vasu and A. N. Rajagopalan, "From local to global: Edge profiles to camera motion in blurred images," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 4447–4456.
- [19] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman, "Removing camera shake from a single photograph," in *Proc. ACM SIGGRAPH*, New York, NY, USA, 2006, pp. 787–794.
- [20] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Efficient marginal likelihood optimization in blind deconvolution," in *Proc. IEEE Conf. CVPR*, Jun. 2011, pp. 2657–2664.
- [21] S. Derin Babacan, R. Molina, M. N. Do, and A. K. Katsaggelos, "Bayesian blind deconvolution with general sparse image priors," in *Proc. Eur. Conf. Comput. Vision*, Oct. 2012, pp. 341–355.
- [22] D. Wipf and H. Zhang, "Revisiting Bayesian blind deconvolution," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3595–3634, 2014.
- [23] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Understanding blind deconvolution algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 12, pp. 2354–2367, Dec. 2011.
- [24] D. Perrone and P. Favaro, "A clearer picture of total variation blind deconvolution," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 6, pp. 1041–1055, Jun. 2016.
- [25] J. Pan, J. Dong, Y. Tai, Z. Su, and M. Yang, "Learning discriminative data fitting functions for blind image deblurring," in *Proc. IEEE ICCV*, Oct. 2017, pp. 1077–1085.
- [26] R. J. Steriti and M. A. Fiddy, "Blind deconvolution of images by use of neural networks," *Opt. Lett.*, vol. 19, no. 8, pp. 575–577, 1994.
- [27] A. Lucas, M. Iliadis, R. Molina, and A. K. Katsaggelos, "Using deep neural networks for inverse problems in imaging: Beyond analytical methods," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 20–36, Jan. 2018.
- [28] L. Xu, J. S. J. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 1790–1798.
- [29] R. Yan and L. Shao, "Blind Image Blur Estimation via Deep Learning," *IEEE Trans. Image Process.*, vol. 25, no. 4, pp. 1910–1921, Apr. 2016.
- [30] J. Zhang et al., "Dynamic scene deblurring using spatially variant recurrent neural networks," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 2521–2529.
- [31] X. Tao, H. Gao, X. Shen, J. Wang, and J. Jia, "Scale-recurrent network for deep image deblurring," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 8174–8182.
- [32] A. Chakrabarti, "A neural approach to blind motion deblurring," in *Proc. Eur. Conf. Comput. Vision*, Oct. 2016, pp. 221–235.
- [33] X. Xu, J. Pan, Y. J. Zhang, and M. H. Yang, "Motion blur kernel estimation via deep learning," *IEEE Trans. Image Process.*, vol. 27, no. 1, pp. 194–205, Jan. 2018.
- [34] Y. W. Tai, P. Tan, and M. S. Brown, "Richardson-Lucy deblurring for scenes under a projective motion path," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1603–1618, Aug. 2011.
- [35] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce, "Non-uniform deblurring for shaken images," *Int. J. Comput. Vision*, vol. 98, no. 2, pp. 168–186, Jun. 2012.
- [36] J. Sun, W. Cao, Z. Xu, and J. Ponce, "Learning a convolutional neural network for non-uniform motion blur removal," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2015, pp. 769–777.
- [37] S. Nah, T. H. Kim, and K. M. Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 257–265.
- [38] T. M. Nimisha, A. K. Singh, and A. N. Rajagopalan, "Blur-Invariant Deep Learning for Blind-Deblurring," in *Proc. IEEE Int. Conf. Comput. Vision*, Oct. 2017, pp. 4762–4770.
- [39] S. Su, M. Delbraccio, J. Wang, G. Sapiro, W. Heidrich, and O. Wang, "Deep Video Deblurring for Hand-held Cameras," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 237–246.
- [40] R. Raskar, A. Agrawal, and J. Tumblin, "Coded exposure photography: Motion deblurring using fluttered shutter," in *Proc. ACM SIGGRAPH*, 2006, pp. 795–804.
- [41] T. S. Cho, A. Levin, F. Durand, and W. T. Freeman, "Motion blur removal with orthogonal parabolic exposures," in *IEEE Int. Conf. Comput. Photography*, 2010, pp. 1–8.
- [42] N. Joshi, S. B. Kang, C. L. Zitnick, and R. Szeliski, "Image deblurring using inertial measurement sensors," in *Proc. ACM SIGGRAPH*, 2010, pp. 1–9.
- [43] J.-F. Cai et al., "Blind motion deblurring using multiple images," *J. Comput. Phys.*, vol. 228, no. 14, pp. 5057–5071, Aug. 2009.
- [44] F. Sroubek and P. Milanfar, "Robust multichannel blind deconvolution via fast alternating minimization," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1687–1700, Apr. 2012.



- [45] W.-S. Lai, J.-B. Huang, Z. Hu, N. Ahuja, and M.-H. Yang, "A comparative study for single image blind deblurring," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 1701–1709.
- [46] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 399–406.
- [47] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
- [48] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang, "Deep networks for image super-resolution with sparse prior," in *Proc. IEEE Int. Conf. Comput. Vision*, 2015, pp. 370–378.
- [49] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep convolutional neural network for inverse problems in imaging," *IEEE Trans. Image Process.*, vol. 26, no. 9, pp. 4509–4522, Sep. 2017.
- [50] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1256–1272, Jun. 2017.
- [51] O. Solomon *et al.*, "Deep unfolded robust PCA with application to clutter suppression in ultrasound," *IEEE Trans. Med. Imag.*, pp. 1–1, 2019.
- [52] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Scholkopf, "Learning to deblur," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 7, pp. 1439–1451, Jul. 2016.
- [53] L. Li, J. Pan, W.-S. Lai, C. Gao, N. Sang, and M.-H. Yang, "Learning a discriminative prior for blind image deblurring," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 6616–6625.
- [54] H. Son and S. Lee, "Fast non-blind deconvolution via regularized residual networks with long/short skip-connections," in *Proc. IEEE Int. Conf. Comput. Photography*, May 2017, pp. 1–10.
- [55] S. Vasu, V. R. Maligireddy, and A. N. Rajagopalan, "Non-blind deblurring: Handling kernel uncertainty with cnns," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Jun. 2018, pp. 3272–3281.
- [56] Y. Li, M. Tofghi, V. Monga, and Y. C. Eldar, "An algorithm unrolling approach to deep image deblurring," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2019, pp. 7675–7679.
- [57] Y. Yang, J. Sun, H. Li, and Z. Xu, "Admm-csnet: A deep learning approach for image compressive sensing," to appear in *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.
- [58] U. Schmidt, C. Rother, S. Nowozin, J. Jancsary, and S. Roth, "Discriminative Non-blind Deblurring," in *IEEE Conf. Comput. Vision Pattern Recognit.*, Jun. 2013, pp. 604–611.
- [59] Y. C. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications*, Cambridge, U.K.: Cambridge university press, 2012.
- [60] R. C. Gonzalez and R. E. Woods, "Digital Image Processing Second Edition," Beijing: Publishing House of Electronics Industry, vol. 455, 2002.
- [61] W. T. Freeman and E. H. Adelson, "The design and use of steerable filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 9, pp. 891–906, Sep. 1991.
- [62] J.-L. Starck, E. J. Candes, and D. L. Donoho, "The curvelet transform for image denoising," *IEEE Trans. Image Process.*, vol. 11, no. 6, pp. 670–684, Jun. 2002.
- [63] M. Unser, N. Chenouard, and D. Van De Ville, "Steerable Pyramids and Tight Wavelet Frames in," *IEEE Trans. Image Process.*, vol. 20, no. 10, pp. 2705–2721, Oct. 2011.
- [64] B. Maillh, S. Lesage, R. Gribonval, F. Bimbot, and P. Vandergheynst, "Shift-invariant dictionary learning for sparse representations: Extending K-SVD," in *Proc. EUSIPCO*, Aug. 2008, pp. 1–5.
- [65] Q. Barthelemy, A. Larue, A. Mayoue, D. Mercier, and J. I. Mars, "Shift amp; 2D rotation invariant sparse coding for multivariate signals," *IEEE Trans. Signal Process.*, vol. 60, no. 4, pp. 1597–1611, Apr. 2012.
- [66] Y. Wang, J. Yang, W. Yin, and Y. Zhang, "A new alternating minimization algorithm for total variation image reconstruction," *SIAM J. Imag. Sci.*, vol. 1, no. 3, pp. 248–272, Jan. 2008.
- [67] D. P Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, NY, USA: Academic press, 2014.
- [68] A. Blake and A. Zisserman, *Visual Reconstruction*, Cambridge, MA, USA: MIT Press, 1987.
- [69] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
- [70] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. Artif. Intell. Statist.*, 2011, pp. 315–323.
- [71] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015.
- [72] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.
- [73] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. ICAIS*, Mar. 2010, pp. 249–256.
- [74] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour Detection and Hierarchical Image Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011.
- [75] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vision*, Springer, 2014, pp. 740–755.
- [76] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Jun. 2016, pp. 770–778.
- [77] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. Matas, "Deblurgan: Blind motion deblurring using conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Jun. 2018, pp. 8183–8192.
- [78] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Conference Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
- [79] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
- [80] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Jun. 2016, pp. 1646–1654.
- [81] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. IEEE Int. Conf. Comput. Vision*, Jul. 2001, pp. 416–423.
- [82] R. Kohler, M. Hirsch, B. Mohler, B. Scholkopf, and S. Harmeling, "Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database," in *Proc. Eur. Conf. Comput. Vision*, 2012, pp. 27–40, Springer.
- [83] L. Li, J. Pan, W.-S. Lai, C. Gao, N. Sang, and M.-H. Yang, "Blind image deblurring via deep discriminative priors," *Int. J. Comput. Vision*, vol. 127, no. 8, pp. 1025–1043, Aug. 2019.
- [84] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 586–595.
- [85] L. Li, J. Pan, W.-S. Lai, C. Gao, N. Sang, and M.-H. Yang, "Learning a discriminative prior for blind image deblurring," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 6616–6625.