Motion Blur Kernel Estimation via Deep Learning

Xiangyu Xu, Jinshan Pan, Yu-Jin Zhang, *Senior Member, IEEE*, and Ming-Hsuan Yang, *Senior Member, IEEE*

***Abstract*— The success of the state-of-the-art deblurring methods mainly depends on the restoration of sharp edges in a coarse-to-fine kernel estimation process. In this paper, we propose to learn a deep convolutional neural network for extracting sharp edges from blurred images. Motivated by the success of the existing filtering-based deblurring methods, the proposed model consists of two stages: suppressing extraneous details and enhancing sharp edges. We show that the two-stage model simplifies the learning process and effectively restores sharp edges. Facilitated by the learned sharp edges, the proposed deblurring algorithm does not require any coarse-to-fine strategy or edge selection, thereby significantly simplifying kernel estimation and reducing computation load. Extensive experimental results on challenging blurry images demonstrate that the proposed algorithm performs favorably against the state-of-the-art methods on both synthetic and real-world images in terms of visual quality and run-time.**

***Index Terms*— Blind image deblurring, kernel estimation, deep convolutional neural network, sharp edges.**

I. Introduction

BLIND image deblurring has been an active research topic in the computer vision and image processing communities within the last decades as it entails tackling challenging tasks in problem formulation and optimization for real-world applications. The goal of blind image deblurring is to recover a blur kernel and a latent sharp image from a blurred input. When the blur is linear shift invariant, the image formation process can be modeled as

$$y = x * k + n, \tag{1}$$

where y is an observed blurred image; x is the corresponding latent image; k denotes the blur kernel; n accounts for noise; and $*$ represents the convolution operator.

Manuscript received February 10, 2017; revised July 2, 2017 and September 1, 2017; accepted September 3, 2017. Date of publication September 18, 2017; date of current version October 20, 2017. This work was supported in part by the NSF CAREER under Grant 1149783, in part by the NSF of China under Grant 61673234, Grant 61732007 and Grant U1636124, in part by the 973 Program of China under Grant 2014CB347600, in part by the NSF of Jiangsu Province under Grant BK20140058, in part by the National Key Research and Development Program of China under Grant 2016YFB1001001, and in part by gifts from Adobe and Nvidia. The work of X. Xu was supported by a scholarship from China Scholarship Council. The associate editor coordinating the review of this manuscript and approving it for publication was Mr. Pierre-Marc Jodoin. *(Corresponding author: Yu-Jin Zhang.)*

X. Xu and Y.-J. Zhang are with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (e-mail: xu-xy13@mails.tsinghua.edu.cn; zhang-yj@mail.tsinghua.edu.cn).

J. Pan is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: sdluran@gmail.com).

M.-H. Yang is with the School of Engineering, University of California at Merced, Merced, CA 95343 USA (e-mail: mhyang@ucmerced.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2017.2753658

Most state-of-the-art deblurring methods rely on implicit or explicit sharp edge restoration for blur kernel estimation [1]–[12]. Most recently, numerous deep learning methods have been developed for image enhancement [13]–[17]. We discuss the most related methods and put this work in proper context.

A. Related Work

1) Implicit Sharp Edge Restoration for Deblurring: This line of image deblurring methods mainly utilizes statistical priors of natural images for sharp edge restoration [1], [2], [7], [8], [11], [18]. As natural image gradients can be modeled well by a heavy-tailed distribution [19], Fergus *et al*. [1] use a mixture of Gaussians to estimate the prior for blind deblurring. Similarly, Levin *et al*. [20] adopt the hyper-Laplacian prior in image deconvolution. Shan *et al*. [2] concatenate two piece-wise continuous functions to fit the logarithmic gradient distribution of natural images for deblurring. However, as demonstrated in [4], deblurring methods based on sparse priors are likely to favor blurred images over clear ones under the naive maximum a posterior (MAP) framework. To address this issue, Krishnan *et al*. [7] present a normalized sparse prior which tends to favor clear images instead of blurred ones. Xu *et al*. [11] analyze the success of implicit as well as explicit sharp-edge-based deblurring methods and develop an L_0-regularized gradient prior for blind image deblurring.

Image priors based on the patch recurrence property have also been developed [21] where these priors favor clear images over blurred ones. In addition, statistical priors for text images have been proposed for deblurring [8], [22]. While these priors have been shown to be effective for image deblurring, the use of such statistical models entails solving non-convex problems. Furthermore, the kernel estimation process is complex and computationally expensive as coarse-to-fine optimization formulations are adopted.

2) Explicit Sharp Edge Restoration for Deblurring: To predict sharp edges for blur kernel estimation, numerous methods have been developed within the MAP framework [3], [10]. However, existing edge selection methods often use heuristic operations such as bilateral [23] and shock [24] filters, which increase the computational complexity. As such, these approaches are likely to fail when sharp edges cannot be identified via image filters. Instead of using image filters, Sun *et al*. [9] learn a dictionary of sharp edge patches from clear images to predict sharp edges. In addition, recent approaches exploit visual information contained both in the blurred input and example images in an external dataset. However, querying sharp edge patches or example images in

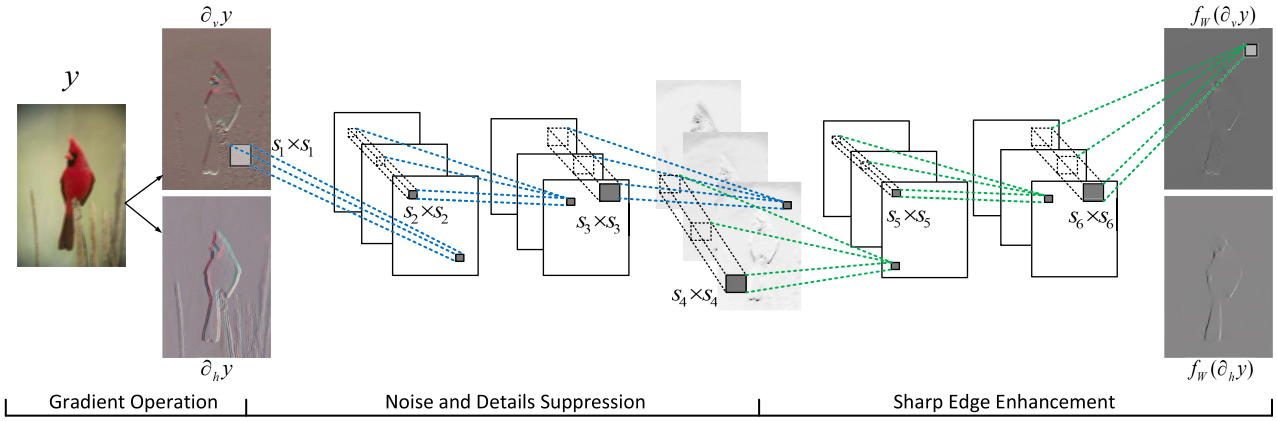


Fig. 1. Overview of the proposed 6-layer CNN for sharp edge restoration in the gradient domain. Given a blurred image y, the proposed method first computes image gradients in vertical and horizontal directions, $\partial_v y$ and $\partial_h y$. With image gradients as input, the first three-layer CNN model is used to retain the main structures and remove the extraneous details of image gradients. The effect of this stage is equivalent to applying a filter to gradient images to remove minor details. The following three-layer CNN model is used to enhance the extracted structure. The effect of this stage is in spirit similar to a shock filter. The detailed analysis of our network is presented in Section IV-A.

a large dataset is computationally expensive. Different from these methods, we propose an edge prediction approach using a convolutional neural network (CNN) model. The proposed algorithm does not require any heuristic step or coarse-to-fine strategy, thereby reducing the complexity and run-time cost of the whole estimation process.

3) Deep Learning for Deblurring: Recently, deep learning models have been applied to image restoration [13], [14], [16], [17], [25]–[28]. Dong *et al.* [25] train an end-to-end CNN model for single image super-resolution. Xu *et al.* [27] propose a CNN to approximate edge-aware image filters. For non-blind image deblurring, Schuler *et al.* [13] use a multi-layer perceptron to help remove artifacts in the Wiener deconvolution process. Xu *et al.* [17] develop a CNN model by separating blur kernels into 1D convolutional kernels. For blind image deblurring, Hradiš *et al.* [14] train a deep CNN model in an end-to-end manner specifically for text images. Schuler *et al.* [16] propose a neural network to estimate blur kernels for generic image deblurring. Nevertheless, this method needs to train different networks for kernels of different sizes, thereby limiting its application domains as the motion blurs in real cases are rather complex. Chakrabarti [28] adopts a locally-connected neural network to predict weights of deconvolution filters. As this network operates on image patches, heuristic thresholding is used to estimate sharp edges and blur kernels. The proposed algorithm differs from the above-mentioned deblurring methods in that it directly learns sharp edges from generic blurred images without heuristic steps or limitations to specific kernel sizes.

B. Our Contributions

In this work, we develop an effective CNN model (see Figure 1) to extract sharp edges from blurred images for kernel estimation. The proposed model learns a mapping function between blurred images and corresponding sharp edges in an end-to-end fashion. It consists of two stages to suppress extraneous details and enhance sharp edges in spirit similar to the state-of-the-art edge prediction methods [3], [10]. The two-stage network architecture simplifies the learning process and leads to effective sharp edge restoration. Compared to the existing deblurring methods, the proposed algorithm does not require any ad-hoc edge selection steps or coarse-to-fine strategy in the kernel estimation process. Thus, it significantly reduces the complexity and run-time of the whole estimation process. Extensive experimental results demonstrate that the proposed algorithm performs favorably against the state-of-the-art methods in terms of visual quality and run-time.

The contributions of this work are summarized as follows. First, we propose a learning based framework for image deblurring which does not require heuristic steps or coarse-to-fine strategies, and significantly reduces the complexity and computational cost of the whole estimation process. Second, we demonstrate the relationship between existing explicit edge selection methods and the proposed CNN model, and analyze how the proposed network is able to extract sharp edges. Finally, to generate more realistic training data, we propose an approach where the intensity distribution is taken into account in the process of synthesizing blur kernels, which facilitates preserving sparse properties

II. Proposed Algorithm

A. Motivation

The state-of-the-art edge prediction approaches [3], [10] usually rely on heuristic filtering methods to select sharp edges from recovered intermediate latent images. The filtering methods are used in a two-stage process: 1) suppression of minor details in an intermediate latent image by the bilateral filter and 2) enhancement of strong structures by the shock filter. However, the heuristic sharp edge prediction step needs to be carried out iteratively in a coarse-to-fine manner with gradient thresholding, which increases the computational complexity of blur kernel estimation. Furthermore, proper thresholds need to be selected for such methods to perform well. To address these issues, we develop a CNN for the aforementioned

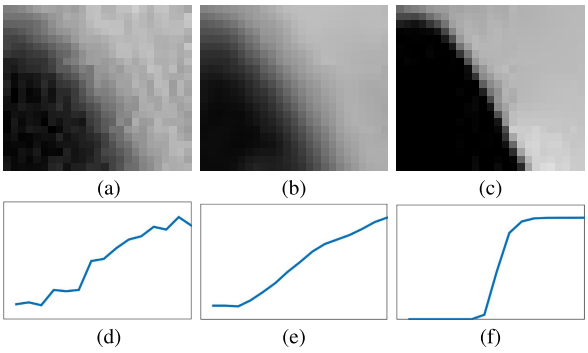


Fig. 2. Proposed edge restoration process. (a) Input. (b) Intermediate feature map. (c) Output. (d)-(f) 1D scanlines of (a)-(c). The first three-layer CNN model (noise and detail suppression in Figure 1) removes details from blurred image (a), and the following three-layer model (sharp edge enhancement in Figure 1) restores the sharp edges from (b).

two-stage process to restore sharp edges from blurred images. The proposed model first extracts the main structures from a blurred input and then enhances them to restore sharp edges. Figure 2 illustrates an intuitive example of this process using 1D signals. By using the restored edges, the blur kernels can be obtained by using existing kernel estimation methods.

B. CNN for Sharp Edge Restoration

Based on the analysis in Section II-A, we propose a CNN to restore sharp edges from blurred images. Using the image formation from (1), the proposed network, as shown in Figure 1, is defined by

$$f^0(\partial y) = \partial y \tag{2}$$

$$f_n^l(\partial y) = \sigma\Big(\sum_m f_m^{l-1} * w_{m,n}^l + b_n^l\Big), \quad l = 1, 2, \ldots, 5 \tag{3}$$

$$f_W(\partial y) = \phi\Big(\sum_m f_m^5 * w_m^6 + b^6\Big), \tag{4}$$

where f_n^l represents the n-th feature map of layer l, and ∂y denotes the image gradient computed from the blurry image y. In addition, w^l and b^l are the convolution kernel and bias of layer l, respectively, and index (m, n) denotes the mapping from the m-th feature map of the current layer to the n-th feature map of the next layer. In this model, w^l is of size $c_{l-1} \times s_l \times s_l \times c_l$, and b^l is of size $c_l \times 1$, where c_l and s_l are the number and the size of the filters in layer l. The function $\sigma(\cdot)$ denotes the rectified linear unit (ReLU) [29]. We normalize the range of image gradients to $[-2, 2]$ and use $\phi(x) = 2\tanh(x)$ as the activation function to constrain filter responses.

The proposed CNN consists of two stages (see Figure 1). The first stage based on the first 3 layers is used to remove extraneous details and retain the main image structures. The second stage based on the following 3 layers is used to enhance the extracted structures and obtain sharp edges. We present detailed analysis of the proposed CNN model in Section IV-A.

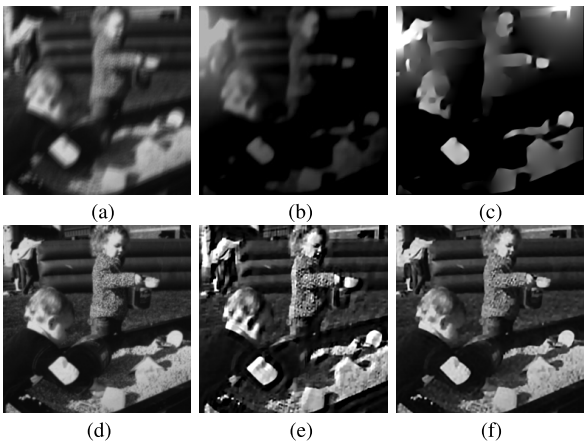


Fig. 3. Learning the network $f_W(\cdot)$ in the intensity and gradient domains. The result in (c) is shown with the Poisson reconstruction and contrast transformation. The network $f_W(\cdot)$ trained in the gradient domain is able to generate sharp edges and thus leads to better deblurred results. (a) Blurred image. (b) $f_W(B)$. (c) $f_W(\partial B)$. (d) Clear image. (e) Result using (b). (f) Result using (c).

C. Learning Process

1) Generating Training Data: To train the first stage network, we use the bilateral filtered images as the ground truth training data to reduce the influence of noise and extraneous details. For the second stage, we need an appropriate representation of sharp edges for training a CNN. As the L_0 sparse representation has been shown to be effective for image deblurring [11], we use the L_0 filter [30] to extract strong structures from x. Thus, each L_0 filtered result can be viewed as the ideal sharp edges for the image x. Note that both the bilateral and L_0 filters are directly applied to clear images.

To generate training data, we randomly collect a set patches $\{x_i\}$ from clear natural images. Based on the degradation process in (1), to obtain the blurred patch y_i, we blur each clean patch x_i using a set of blur kernels $\{h_j\}$ (introduced in Section II-C3) and further add 1% Gaussian noise to the blurred patches. We denote $T_1(x_i)$ as the intermediate result generated by the bilateral filter, and $T_2(x_i)$ as the sharp edge patch generated by the L_0 filter [30]. As the goal of this step is to predict sharp edges, we train the network in the gradient domain. Thus, the gradient operators are then applied to y_i, $T_1(x_i)$ and $T_2(x_i)$ to obtain ∂y_i, $\partial T_1(x_i)$ and $\partial T_2(x_i)$. As the sharp edge extraction can generate the same effect when we rotate the input image by 90 degrees, we train the network only on the gradients in the vertical direction, and share the weights for gradients in both directions. We also train the network in the intensity domain. However, we find that the network trained in the intensity domain is not able to restore sharp edges as shown in Figure 3(b). Therefore, we do not recover as clear images as the network trained in the gradient domain (Figure 3(e) and (f)).

2) Training the Proposed Network: As shown in Figure 1, the output of the third layer is composed of c_3 feature maps denoted by $\{f_m^3, m = 1, 2, \ldots, c_3\}$. However, $\partial T_1(x)$, which is the ground truth training data of the first stage, has only one channel. A straight-forward method to train the two-stage

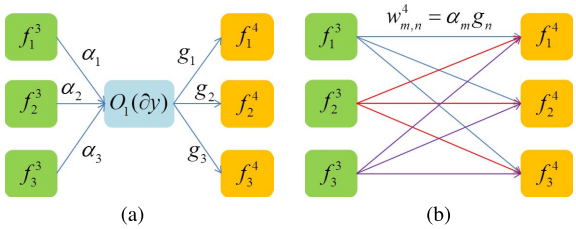


Fig. 4. A toy example of combining the two sub-networks. The formulation is introduced in (12). The capacity of the network increases after the combination (from 6 free parameters in (a) to 9 parameters in (b)). (a) Before combination. (b) After combination.

network is to use one channel for the third layer (i.e., $c_3 = 1$). But the model with only one channel in the intermediate layer is not efficient and usually leads to undesirable results. As such, we first take the weighted average of the feature maps as the output of the first stage by

$$O_1(\partial y) = \sum_{m=1}^{c_3} \alpha_m f_m^3, \tag{5}$$

where $\{\alpha_m\}$ are the learnable coefficients.

With a set of D image patch pairs $\{\partial y_i, \partial T_1(x_i)\}$, the first stage of the network can be learned by minimizing

$$\frac{1}{D}\sum_i \|O_1(\partial y_i) - \partial T_1(x_i))\|_1 + \lambda \|O_1(\partial y_i)\|_1, \tag{6}$$

where the L_1 loss with total variation (TV) regularization is used to enforce sparsity on image gradients and λ is the regularization weight. As the L_1 norm metric function is not differentiable at zero, we use the Charbonnier function $\rho(z) = (z^2 + \epsilon^2)^{1/2}$ to approximate it. Thus, the objective function is rewritten as

$$\frac{1}{D}\sum_i \rho(O_1(\partial y_i) - \partial T_1(x_i)) + \lambda \rho(O_1(\partial y_i)). \tag{7}$$

To train the second stage network, the output of the last three layers is computed by:

$$f_n^4(\partial y) = \sigma(g_n * O_1(\partial y) + b_n^4), \tag{8}$$

$$f_n^5(\partial y) = \sigma(\sum_m f_m^4 * w_{m,n}^5 + b_n^5), \tag{9}$$

$$O_2(\partial y) = \phi(\sum_m f_m^5 * w_m^6 + b^6), \tag{10}$$

where g_n is the learnable convolution kernel of size $s_4 \times s_4$. Similarly, with a set of D image patch pairs $\{\partial y_i, \partial T_2(x_i)\}$, the second stage of the network can be learned by minimizing

$$\frac{1}{D}\sum_i \rho(O_2(\partial y_i) - \partial T_2(x_i)) + \lambda \rho(O_2(\partial y_i)). \tag{11}$$

After both the sub-networks are trained, we combine them by computing the convolution kernel of the fourth layer as

$$w_{m,n}^4 = \alpha_m g_n, \quad m = 1, 2, \ldots, c_3, \ n = 1, 2, \ldots, c_4. \tag{12}$$

This combining process is able to increase the capacity of the network (see Figure 4), thus facilitating sharp edge prediction.

Algorithm 1 Training the Proposed Network

Input: training dataset $\{\partial y_i, \partial T_1(x_i), \partial T_2(x_i)\}$;
for each image patch **do** ▷ first stage
  compute the output of the first stage $O_1(\partial y_i)$ with (2), (3), and (5);
  compute the loss between $O_1(\partial y_i)$ and $T_1(x_i)$ with (7), and update the first stage network weights using backward propagation;
end for
for each image patch **do** ▷ second stage
  compute the output of the first stage $O_1(\partial y_i)$ with (2), (3), and (5);
  compute the output of the second stage $O_2(\partial y_i)$ with $O_1(\partial y_i)$, (8), (9), and (10);
  compute the loss between $O_2(\partial y_i)$ and $T_2(x_i)$ with (11), and update the second stage network weights using backward propagation;
end for
combine the two sub-networks using (12);
for each image patch **do** ▷ fine-tuning
  compute the output of the whole network $f_W(\partial y_i)$ with (2), (3), and (4);
  compute the loss between $f_W(\partial y_i)$ and $T_2(x_i)$ with (13), and update the whole network weights using backward propagation;
end for
Output: network model weights W.

Finally, the whole network is fine-tuned in an end-to-end manner to minimize

$$\frac{1}{D}\sum_i \rho(f_W(\partial y_i) - \partial T_2(x_i)) + \lambda \rho(f_W(\partial y_i)). \tag{13}$$

The main steps of the training process are shown in Algorithm 1.

3) Synthesizing Blur Kernels: One of the main issues in generating training data is how to synthesize blur kernels close to real-world scenarios. Although several real blur kernels have been generated by Levin *et al.* [4], this dataset is not sufficient for training the proposed network. Schmidt *et al.* [31] propose a method to generate blur kernels by sampling random 3D trajectories with a linear motion model. These trajectories are then projected and rasterized to generate blur kernels of square sizes. However, this method does not preserve the sparse properties of blur kernels. We plot the distributions of the synthetic kernels generated by this scheme and the real blur kernels from [4] in Figure 5. The results show that the distribution of the generated blur kernels using the approach by Schmidt *et al.* [31] is significantly different from that of the real ones.

To generate more realistic synthetic blur kernels, we sample from the intensity distribution of real kernels [4] and 3D trajectories obtained in a way similar to the method by Schmidt *et al.* [31]. We use the Kullback-Leibler divergence to remove sampling results which deviate significantly from the average real kernel distribution. The green curve shown

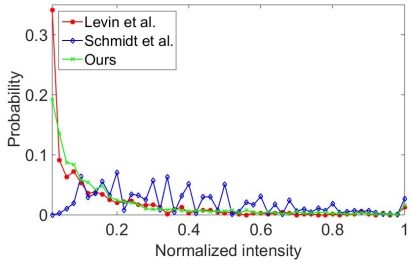


Fig. 5. Intensity distributions of blur kernels by different methods. The distribution of our synthetic blur kernels is more close to that of real blur kernels.

(a) (b) (c)

Fig. 6. Some synthetic kernels generated by different methods. (a) real kernels obtained from [4]. (b) synthetic blur kernels generated by [31]. (c) synthetic kernels generated by our method.

in Figure 5 demonstrates that the distribution of our synthetic blur kernels is similar to that of real blur kernels. Figure 6 shows some blur kernels synthesized by the proposed method.

D. Kernel Estimation

After obtaining the salient edges $\partial e = f_W(\partial y)$ from a blurry image y, we estimate the blur kernel by alternately solving

$$k = \arg\min_k \|\partial e * k - \partial y\|_2^2 + \gamma \|k\|_2^2, \tag{14}$$

and

$$\widetilde{x} = \arg\min_{\widetilde{x}} \|\widetilde{x} * k - y\|_2^2 + \eta \|\partial \widetilde{x}\|_0, \tag{15}$$

where γ and η are parameters for the regularization terms, and $\widetilde{x}$ is the intermediate latent image. Similar to [32], we use $\widetilde{x}$ to update the salient edges ∂e in (14).

For (14), we note that it is a least squares problem, and the closed-form solution is

$$k = \mathcal{F}^{-1}\left(\frac{\overline{\mathcal{F}(\partial_h e)}\mathcal{F}(\partial_h y) + \overline{\mathcal{F}(\partial_v e)}\mathcal{F}(\partial_v y)}{\mathcal{F}(\partial_h e)^2 + \mathcal{F}(\partial_v y)^2 + \gamma}\right), \tag{16}$$

where $\mathcal{F}(\cdot)$ and $\mathcal{F}^{-1}(\cdot)$ denote the FFT and inverse FFT operators, and $\overline{\mathcal{F}(\cdot)}$ is the complex conjugate of $\mathcal{F}(\cdot)$.

As (15) involves the L_0 norm, it can be efficiently solved using the half-quadratic splitting technique [30].

1) Final Latent Image Estimation: Once the blur kernel is determined, the final latent image can be estimated by a number of non-blind deconvolution methods. In this paper, we use the hyper-Laplacian prior with $L_{0.8}$ norm to recover the latent image by

$$x = \arg\min_x \|x * k - y\|_2^2 + \beta \|\partial x\|_{0.8}, \tag{17}$$

which can be optimized in an iterative re-weighted least squares process [20]. The main steps for kernel estimation and deblurring are summarized in Algorithm 2.

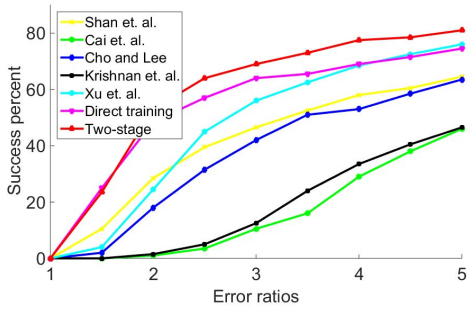


Fig. 7. Quantitative evaluation on the proposed synthetic dataset with several state-of-the-art single image blind deblurring methods: Shan *et al.* [2], Cho and Lee [3], Krishnan *et al.* [7], Cai *et al.* [33], and Xu *et al.* [11].

Algorithm 2 Blur Kernel and Latent Image Estimation

1: **Input:** blurred input y, optimized network weights W;
2: compute the salient edges $\partial e = f_W(\partial y)$ according to (2), (3), and (4);
3: **for** $loop = 1 : \tau$ **do**
4: solve for k using (14);
5: solve for $\widetilde{x}$ using (15);
6: $\partial e \leftarrow \partial x$;
7: **end for**
8: estimate the latent image x using (17);
9: **Output:** blur kernel k and latent image x.

E. Non-Uniform Deblurring

Camera shakes including rotation and translation usually lead to spatially variant blur effects on images. As shown in [34] and [11], this process is usually modeled as

$$\mathbf{y} = \sum_m t_m \mathbf{H}_m \mathbf{x} + \mathbf{n}, \tag{18}$$

where $\mathbf{y}$, $\mathbf{x}$ and $\mathbf{n}$ are the corresponding vector forms of y, x, and n, respectively; m indexes camera pose samples and $\mathbf{H}_m$ is a transformation matrix which corresponds to either camera rotation or translation for pose m; and t_m denotes the time that the camera stays at pose m and serves as a weight in this function. Note that the equations for non-uniform deblurring are expressed in lexicographic notation for clarity.

With the predicted salient edges $\partial \mathbf{e}$, the blur kernel $\mathbf{k} = \{t_m\}$ can be obtained by alternately solving

$$\mathbf{k} = \arg\min_{\mathbf{k}} \|\sum_m t_m \mathbf{H}_m \partial \mathbf{e} - \partial \mathbf{y}\|_2^2 + \gamma \|\mathbf{k}\|_2^2, \tag{19}$$

and

$$\widetilde{\mathbf{x}} = \arg\min_{\widetilde{\mathbf{x}}} \|\sum_m t_m \mathbf{H}_m \widetilde{\mathbf{x}} - \mathbf{y}\|_2^2 + \eta \|\partial \widetilde{\mathbf{x}}\|_0. \tag{20}$$

For (20), we use the half-quadratic splitting L_0 minimization method [30] to solve it. For the kernel estimation model (19), we use the same optimization method in [11] to update blur kernel $\mathbf{k}$. Similar to the method discussed in Section II-D, the hyper-Laplacian prior with the $L_{0.8}$ norm is used for non-blind deblurring.

Fig. 8. An example from the proposed synthetic dataset. Compared with the state-of-the-art prior-based deblurring method [11], the proposed method generates a clearer image with less artifacts. (a) Blurred image. (b) $f_W(\partial B)$. (c) Cho and Lee [3]. (d) Xu *et al.* [11]. (e) Ours.

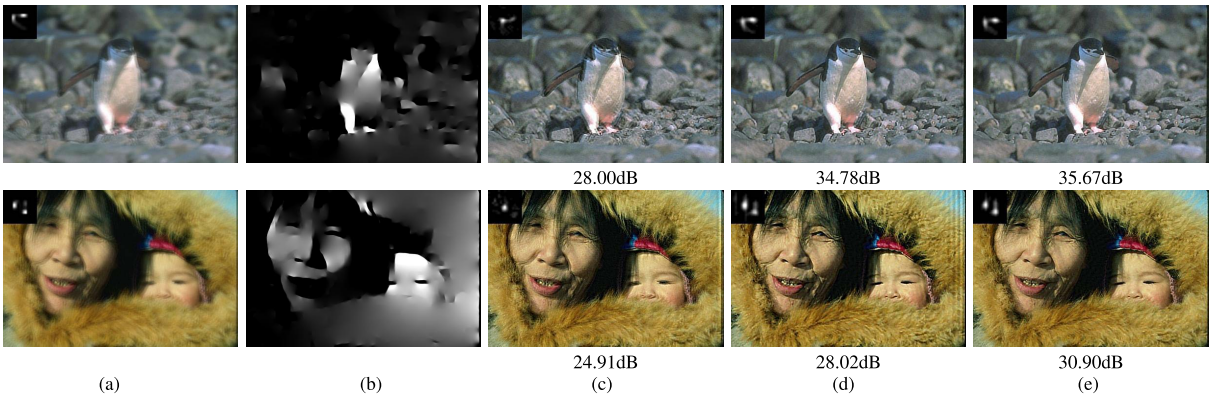


Fig. 9. Sample images from the proposed synthetic dataset. Compared with the state-of-the-art edge-selection-based deblurring method [3], the proposed method generates clearer images with less artifacts. (images best viewed on a high-resolution display) (a) Blurred image. (b) $f_W(\partial B)$. (c) Cho and Lee [3]. (d) Xu *et al.* [11]. (e) Ours.

III. Experimental Results

We first describe the implementation details and parameter settings, and then compare the proposed algorithm with the state-of-the-art deblurring methods [2], [3], [7], [11], [18], [33]. The default parameters of all the evaluated methods are used.

A. Implementation Details and Parameter Settings

For the proposed network (Section II-B), we set the size of convolutional filters as $s_1 = 9, s_2 = 1, s_3 = 3, s_4 = 5, s_5 = 1, s_6 = 3$, and the numbers of filters as $c_n = 128$ where $n = 1, 2, \ldots, 5$. As the input and output of our network are gradients of gray images, c_0 and c_6 are both set as 1. We pad the image boundary at each convolution layer such that the output has the same size with that of the input. The parameter λ and ϵ in the training objective function are set to be 0.005 and 10^{-6}. For kernel estimation, the parameters η, γ, τ and β are set to be 0.002, 1, 15 and 0.001, respectively.

To generate training data, we randomly crop one million 128×128 patches $\{x_i\}$ from the training set of the BSDS500 database [35]. We use the L_0-smoothing weight of 0.02 to remove extraneous details and retain sharp edges in the ground truth data $T_2(x)$. To use more informative data in the training stage, we remove the image patches where fewer than 10% pixels have gradients in the horizontal and vertical directions with an absolute value 0.1 or above. Each image patch is blurred with a single synthetic blur kernel. The size of our synthetic blur kernels is randomly sampled within the range of [17, 31] pixels. Furthermore, 1% Gaussian noise is added to the blurred patches. The above training set is empirically determined to be sufficient for training the proposed network.

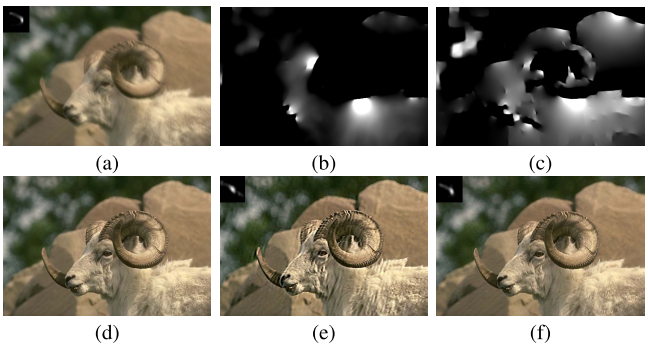


Fig. 10. Effectiveness of the two-stage training strategy. (a) is the blurred image and blur kernel; (d) is the clear image; (b) and (e) are the edge map and the deblurred result without using two-stage training; (c) and (f) are the edge map and the deblurred result using two-stage training.

Similar to [36], the weights of filters in each layer are initialized using a Gaussian distribution with zero mean and

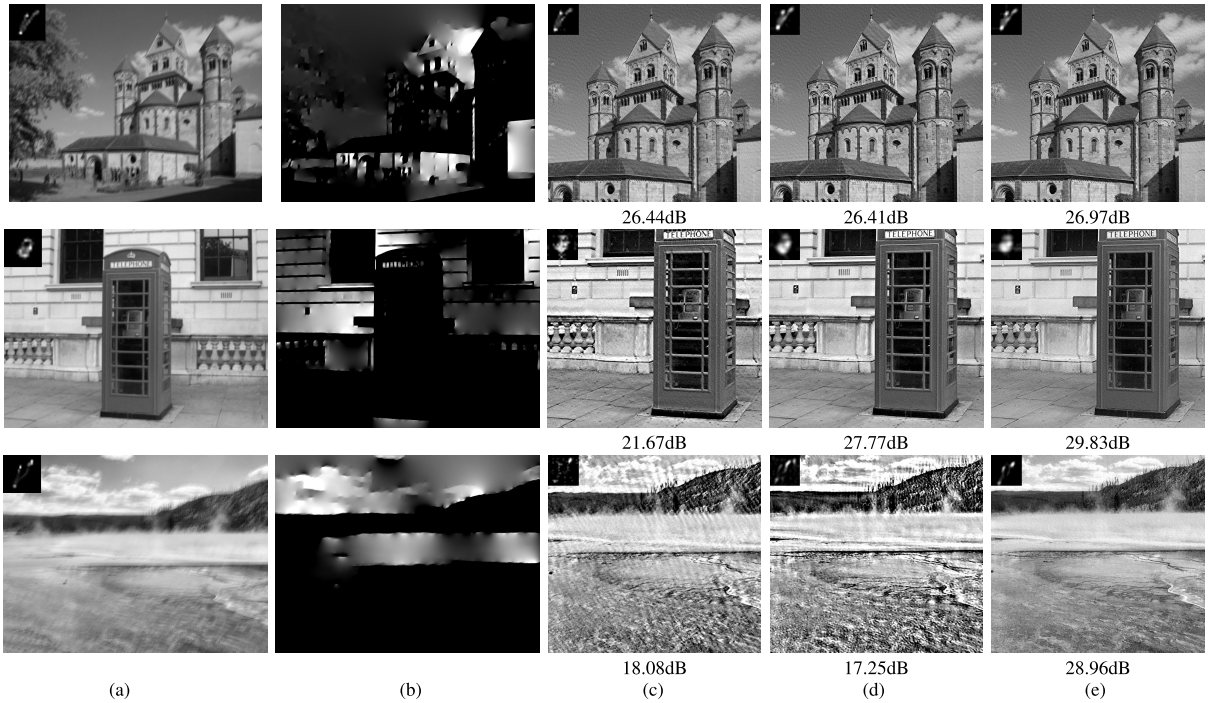


Fig. 11. Sample images from the dataset by Sun *et al.* [9]. Compared with the state-of-the-art methods [3] and [11], the proposed algorithm generates clearer images with fewer artifacts. Note that the blur kernels estimated by the proposed algorithm are close to the ground truth data while other methods tend to introduce noise (images best viewed on a high-resolution display). (a) Blurred image. (b) $f_W(\partial B)$. (c) Cho and Lee [3]. (d) Xu *et al.* [11]. (e) Ours.

variance of $2/n_{in}$, where n_{in} is the size of the respective convolutional filter. We use the Adam algorithm [37] with the default parameters to minimize the loss function of the proposed network. In the fine-tuning stage, we set the learning rate to be 5 times smaller and set β_1 as 0.5 while keeping other parameters unchanged. The training process takes roughly two days using a GTX Titan GPU.

B. Evaluations on Synthetic Images

We quantitatively and qualitatively evaluate the proposed algorithm on both the proposed and benchmark datasets [9].

1) Proposed Dataset: For comprehensive evaluation, we collect a dataset of 20 clear images sampled from the BSDS500 test set [35] and 10 ground truth kernels to generate a test set of 200 blurred images in a way similar to [4]. The images of the training and test datasets are not overlapped. The ground truth kernels are generated by the method proposed in Section II-C3 where the kernel size is from 17 to 27 pixels. In addition, we add 1% Gaussian noise to the blurred image to approximate real blur process as introduced in (1).

We evaluate the proposed algorithm against the state-of-the-art deblurring approaches [2], [3], [7], [11], [33] using the error metric proposed by Levin *et al.* [4]. For fair comparisons, we use the same non-blind deconvolution method from [20] for all the blur kernel estimation approaches to generate the deblurred images. Figure 7 shows the cumulative error ratio where higher curves indicate more accurate results. Note that the error ratio of 3 is considered as the threshold of visually plausible deblurred results [18]. As shown in Figure 7, the proposed deblurring algorithm performs favorably against the state-of-the-art methods.

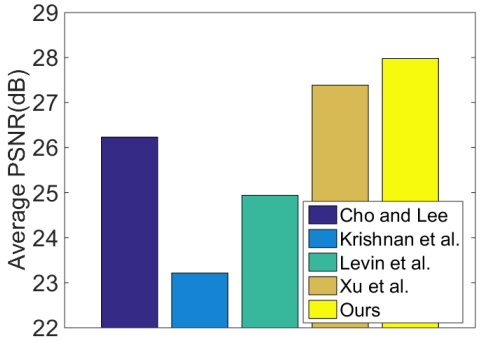
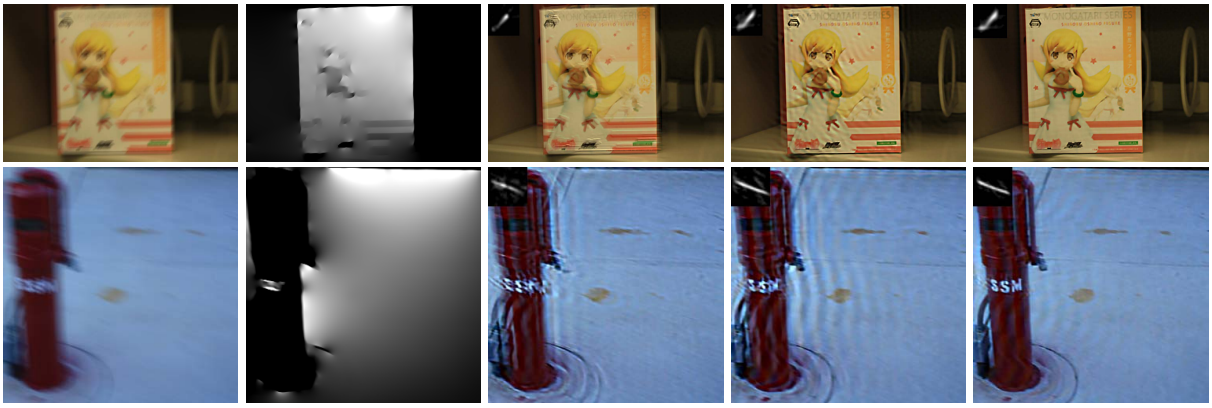


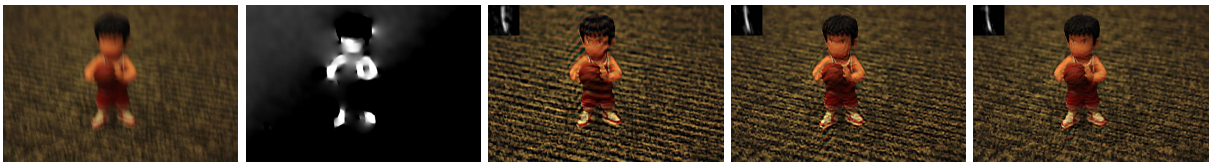
Fig. 12. Quantitative comparisons on the dataset from [9] with several state-of-the-art single image blind deblurring methods: Cho and Lee [3], Krishnan *et al.* [7], Levin *et al.* [18], and Xu *et al.* [11].

Figure 8 and 9 show sample deblurred images by the evaluated algorithms. The images deblurred by the sparse prior based method [11] contain significant artifacts. As these methods often require coarse-to-fine strategies to estimate kernels, the deblurred images are not sharp when the estimated results in the coarse stage are not accurate. As shown in Figure 9, the edge selection based method [3] does not perform well when background clutters are not removed and only few salient edges can be extracted. In contrast, the proposed algorithm learns to extract sharp edges from blurred images in an end-to-end manner without using heuristic steps. Thus, the restored results from the proposed model contain more useful edge information which leads to better blur kernel estimation and latent image restoration.



(a) (b) (c) (d) (e)

Fig. 13. Real captured blurry images. Our method generates clearer characters with fewer ringing artifacts. (a) Blurred image. (b) $f_W(\partial B)$. (c) Cho and Lee [3]. (d) Xu *et al*. [11]. (e) Ours.



(a) (b) (c) (d) (e)

Fig. 14. A real captured image with blurry background textures. Our method generates a clear image with fewer artifacts. (a) Blurred image. (b) $f_W(\partial B)$. (c) Cho and Lee [3]. (d) Xu *et al*. [11]. (e) Ours.

In addition, as the two-stage training strategy exploits the merits of typical sharp edge prediction processes which are based on details suppression and edge enhancement, the trained network achieves better performance than that by direct end-to-end training. As shown in Figure 7, the model trained by the two-stage strategy performs better than the model trained directly in the end-to-end manner by 5% at the error ratio of 3, which demonstrates the effectiveness of the two-stage model. We also show one example in Figure 10 for comparison. The proposed method generates a more informative edge map (Figure 10(e)) which in turn leads to a sharper deblurred image (Figure 10(f)).

2) Benchmark Dataset by Sun et al. [9]: We carry out experiments on the blurry image dataset developed by Sun *et al*. [9]. This dataset includes 640 images generated by 80 high resolution natural images from diverse scenes and 8 blur kernels from [4]. In addition, 1% Gaussian noise is added to each blurry image.

Figure 11 shows a few estimated kernels and deblurred images obtained by the proposed algorithm and the state-of-the-art methods [3], [7], [11], [18]. While the deblurred images by existing methods contain significant ringing artifacts, the proposed algorithm restores informative sharp edges from the blurry images which facilitate kernel estimation and lead to clearer results with fewer artifacts. Note that the kernels estimated by the proposed algorithm are close to the ground truth data whereas the results by the state-of-the-art methods contain noise. Figure 12 shows the quantitative evaluation on this dataset where the proposed algorithm performs favorably against the state-of-the-art methods in terms of average PSNR values.

C. Evaluations on Real Images

We evaluate the proposed algorithm against the state-of-the-art deblurring methods [3], [11] on real blurred images. Figure 13 shows two real captured images containing blurry characters. While the state-of-the-art methods do not generate clear images (Figure 13(c)-(e)), the proposed algorithm learns to extract sharp edges and recover latent images with clearer characters and fewer ringing artifacts.

Figure 14 shows another real image which contains blurry background textures caused by motion blur. Although the textured background contains numerous edges, the proposed algorithm restores sharp edges and generates a clearer latent image whereas the deblurred results by the state-of-the-art methods contain more artifacts.

1) Non-Uniform Image Deblurring: As discussed in Section II-E, the proposed method can also be applied to non-uniform deblurring. We evaluate the proposed algorithm against the state-of-the-art non-uniform deblurring methods [11], [34], [38]–[40] using real images. Figure 16 shows the deblurred result from [34] contains ringing effects along the roof. Compared to [38] and [11], the proposed algorithm generates a sharper image as shown in the zoomed-in regions of Figure 16. For the image in Figure 17, the results in [39] and [11] contain blurry edges and the deblurred image by [40] has significant ringing artifacts. In contrast, the proposed algorithm generates a clearer image.

D. Run-Time

As the proposed algorithm does not require any heuristic edge selection steps or coarse-to-fine strategies, the kernel

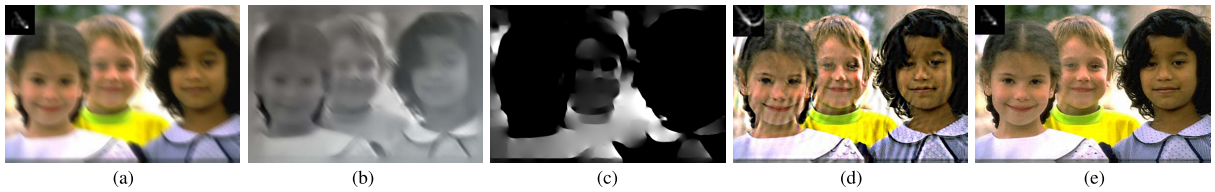


Fig. 15. Effectiveness of the proposed network. The three-layer CNN model used in image filtering [27] does not generate sharp edges for kernel estimation. (a) Blurred image. (b) Edges by model [27]. (c) Our learned edges. (d) Result using (b). (e) Result using (c).

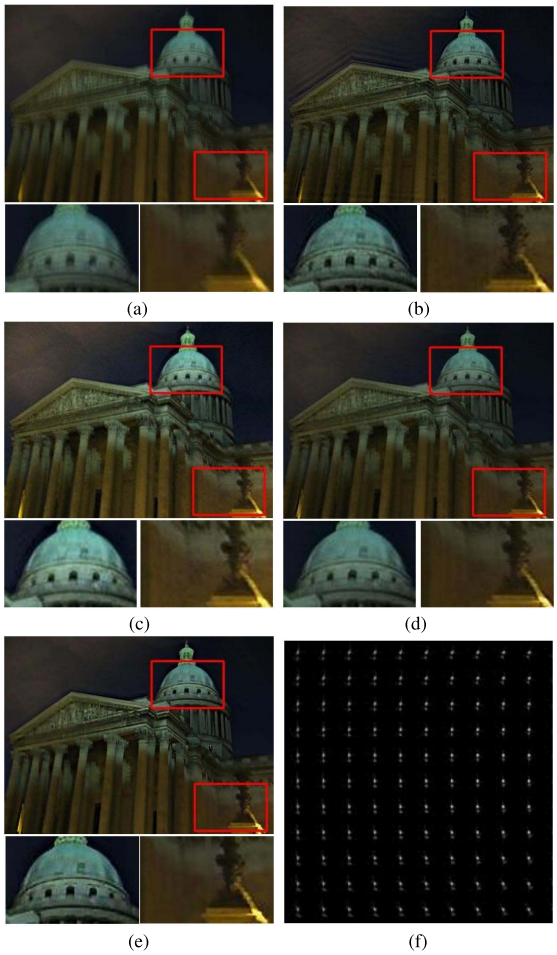


Fig. 16. Visual comparison with state-of-the-art non-uniform deblurring methods. (a) Blurred image. (b) Whyte *et al.* [34]. (c) Hirsch *et al.* [38]. (d) Xu *et al.* [11]. (e) Ours. (f) Our estimated kernels.

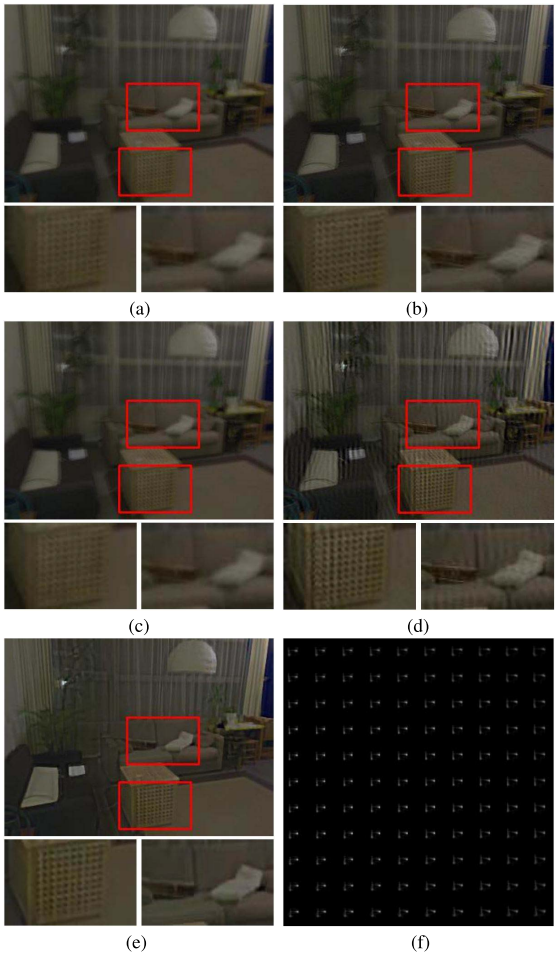


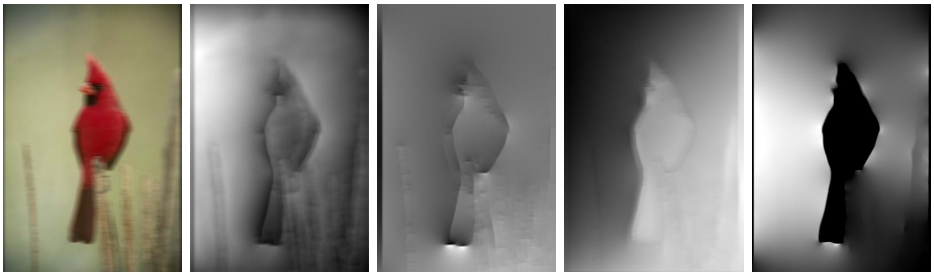
Fig. 17. Visual comparison with state-of-the-art non-uniform deblurring methods. (a) Blurred image. (b) Ren *et al.* [39]. (c) Xu *et al.* [11]. (d) Pan *et al.* [40]. (e) Ours. (f) Our estimated kernels.

estimation process is computationally more efficient than the state-of-the-art methods. The run-time for kernel estimation reported in Table I is obtained on the same machine with an Intel i7 CPU, a GTX Titan GPU and 64GB memory. Overall, the proposed algorithm performs favorably against other state-of-the-art methods in terms of run-time. In addition, we replace the coarse-to-fine strategy in the deblurring methods [7], [18] with the proposed edge prediction model. As shown in Table II, the run-time for kernel estimation is also reduced by removing the coarse-to-fine edge selection process. We do not conduct this experiment on the other baseline methods as the source code is not available.

IV. Analysis and Discussion

A. Effect of Proposed Network

As the three-layer CNN architecture has been shown to be effective to learn image filters [27], we analyze the proposed network as two parts. The first three-layer sub-network is used to remove extraneous details from blurred images and the following three-layer sub-network is used to restore sharp edges. The edges restoration process is in spirit similar to the sharp signal recovery step as discussed in Section II-A. To demonstrate how the proposed network performs, we present some intermediate feature maps from the third layer in Figure 18. We note that these feature maps contain main structures with few details, which indicates that



(a) (b) (c) (d) (e)

Fig. 18. Visualization of feature maps from the third hidden layer. (a) is the input blurry image; (b)-(d) are the feature maps of the third layer visualized by Poisson reconstruction; (e) is the restored sharp edges from the output layer.

TABLE I

RUN-TIME (IN SECONDS) OF DIFFERENT METHODS IN THREE IMAGE RESOLUTION

Method	200×200	400×400	800×800
Krishnan *et al.* [7] (Matlab)	12.74	52.61	125.93
Levin *et al.* [18] (Matlab)	134.61	225.46	706.27
Cai *et al.* [33] (Matlab)	203.12	978.82	1450.22
Cho and Lee [3] (C++)	0.44	1.66	7.41
Xu *et al.* [11] (C++)	0.94	1.73	5.95
Ours (Matlab)	0.36	1.80	8.24

TABLE II

RUN-TIME OF REPLACING THE COARSE-TO-FINE PROCESS IN EXISTED DEBLURRING METHODS WITH THE PROPOSED EDGE PREDICTION MODEL (BEFORE REPLACING / AFTER REPLACING)

Method	200×200	400×400	800×800
Krishnan *et al.* [7]	12.74/7.94	52.61/33.65	125.93/64.58
Levin *et al.* [18]	134.61/92.25	225.46/153.86	706.27/508.56

the first three-layer CNN part is able to remove extraneous details. We also note that the restored edge map from the output layer becomes sharper through the second three-layer CNN model. This process is similar to the edge prediction illustrated in Section II-A, which shows the effectiveness of the proposed algorithm.

The CNN model used in [27] is not able to restore sharp edges from blurry images. We retrain this model with exhaustive parameter tuning including the size and number of filters in each layer. The restored result still contains blurry details and few sharp edges as shown in Figure 15(b), while our method generates better sharp edges (Figure 15(c)) which facilitate kernel estimation. In addition, we quantitatively evaluate the model from [27] and the proposed network on the synthetic dataset developed in this work. The average PSNR of the results by the three-layer CNN model is 26.59dB, which is significantly lower than 30.23dB achieved by the proposed 6-layer network. These results demonstrate that the CNN model with three-layer is not able to extract sharp edges from blurry images for kernel estimation.

B. Network Parameters

There are several essential parameters in the proposed algorithm including the filter number and size of the CNN model, and the L_0-smoothing weight used for generating the training data. In this section, we show how these parameters affect the deblurring results by training different models. For fair comparisons, these networks are trained using the same process except the parameter being analyzed. We evaluate the results on the proposed synthetic dataset.

1) Filter Number: Compared with other CNN structures [14], [27], we use a relatively small number of convolution filters to restore sharp edge for image deblurring. In general, the proposed algorithm performs better if we enlarge the network scale by adding more filters. Here, we evaluate the models using different numbers of filters. Based on our default network setting with $c_n = 128$, we conduct two additional experiments: one larger network with the filter number $c_n = 256$, and the other smaller network with $c_n = 100$. As shown in Table III, better performance can be achieved by using more filters but the difference is not significant, which demonstrates that the proposed model is robust to the filter numbers within a reasonable range.

2) Filter Size: As demonstrated in [41], the filter size in the first layer of the network is important to the performance in low-level problems. We analyze the effect of filter size in the first layer for blur kernel estimation. In the above-mentioned experimental results, we set the filter size of the first layer as $s_1 = 9$. Here we enlarge the filter size to $s_1 = 11$ for experiments. The average PSNR of the deblurring results on the proposed dataset is 30.22dB, which is almost the same as 30.23dB reported in Table III. This suggests that a reasonably smaller filter size is sufficient for sharp edge restoration in the proposed algorithm without increasing computational cost.

3) L_0-Smoothing Weight: When generating the ground truth sharp edges for training, we use a relatively small L_0-smoothing weight of 0.02. We find that larger L_0-smoothing weight values decrease the accuracy of kernel estimation as most sharp edges are smoothed. Our experimental results show that the PSNR of the deblurring results is 30.23dB when the L_0-smoothing weight is 0.02, whereas the PSNR is 30.08dB when the L_0-smoothing weight is 0.08.

C. Comparisons With Simple Filters

We evaluate the proposed algorithm with the exact same restoration process except replacing the network with the bilateral and shock filters with similar parameters as those used for training. The average PSNR of the filtering based

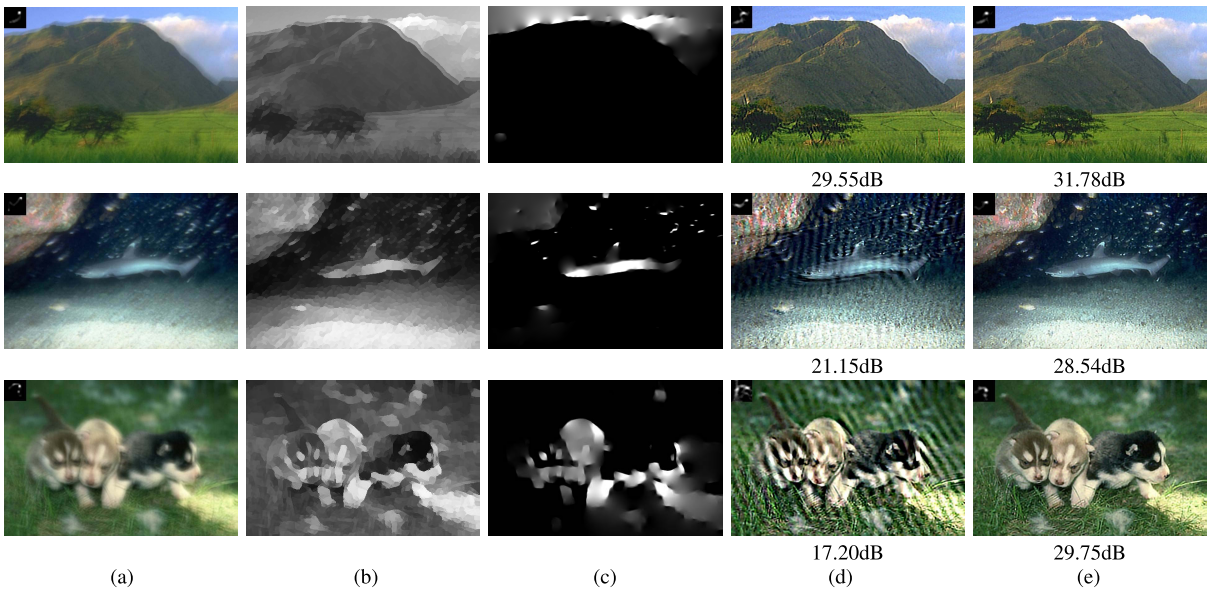


Fig. 19. Comparisons with bilateral and shock filters. The proposed CNN model generates sharper edges which facilitate image deblurring. (a) Blurred image. (b) Edges by simple filters. (c) Our learned edges. (d) Result using (b). (e) Result using (c).

TABLE III
DEBLURRING RESULTS USING DIFFERENT FILTER NUMBERS ON THE PROPOSED SYNTHETIC DATASET

Filter number	$c_n = 100$	$c_n = 128$	$c_n = 256$
Average PSNR (dB)	30.21	30.23	30.29

Fig. 20. A failure example by the proposed algorithm on the blurred image with numerous outliers. (a) Blurred input. (b) Deblurred output.

method is 28.99dB on the proposed dataset which is lower than 30.23dB of using the proposed model. We show three examples in Figure 19. The method with simple filters generates numerous ambiguous edges (Figure 19(b)) while the proposed model generates much sharper edges (Figure 19(c)).

D. Limitations

Although the proposed algorithm is able to extract sharp edges from blurry images for kernel estimation, the formulation is based on the conventional models. Thus, it has the same limitations as the state-of-the-art deblurring methods and is likely to fail when the blurred images contain a significant amount of outliers such as saturated regions. As shown in Figure 20, the proposed method is not able to deblur images well as the assumption of the blur model does not hold for images with outliers [42]. Our future work will focus on developing a network which considers outliers in the kernel estimation process.

V. CONCLUSION

In this paper, we propose a deep learning algorithm for image deblurring. We thoroughly analyze the proposed deep network and show that it is able to learn to extract sharp edges from blurry images for kernel estimation. The proposed method does not require heuristic edge selection steps or coarse-to-fine strategies which are widely used in image deblurring. It significantly simplifies the kernel estimation process and reduces the computation cost. Extensive experimental evaluations show that the proposed algorithm is effective for generic image deblurring.

ACKNOWLEDGEMENT

This work is supported in part by the NSF CAREER Grant 1149783, NSF of China (No. 61673234 and U1636124), 973 Program of China (No. 2014CB347600), NSF of China (No. 61732007), NSF of Jiangsu Province (No. BK20140058), the National Key R&D Program of China (No. 2016YFB1001001), and gifts from Adobe and Nvidia. X. Xu is supported in part by a scholarship from China Scholarship Council.

REFERENCES

[1] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman, "Removing camera shake from a single photograph," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 787–794, 2006.
[2] Q. Shan, J. Jia, and A. Agarwala, "High-quality motion deblurring from a single image," *ACM Trans. Graph.*, vol. 27, no. 3, p. 73, 2008.
[3] S. Cho and S. Lee, "Fast motion deblurring," *ACM Trans. Graph.*, vol. 28, no. 5, p. 145, 2009.
[4] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Understanding and evaluating blind deconvolution algorithms," in *Proc. CVPR*, Jun. 2009, pp. 1964–1971.
[5] N. Joshi, R. Szeliski, and D. J. Kriegman, "PSF estimation using sharp edge prediction," in *Proc. CVPR*, Jun. 2008, pp. 1–8.
[6] D. Krishnan and R. Fergus, "Fast image deconvolution using hyper-Laplacian priors," in *Proc. NIPS*, 2009, pp. 1033–1041.
[7] D. Krishnan, T. Tay, and R. Fergus, "Blind deconvolution using a normalized sparsity measure," in *Proc. CVPR*, 2011, pp. 233–240.

[8] J. Pan, Z. Hu, Z. Su, and M.-H. Yang, "Deblurring text images via L_0-regularized intensity and gradient prior," in *Proc. CVPR*, 2014, pp. 2901–2908.
[9] L. Sun, S. Cho, J. Wang, and J. Hays, "Edge-based blur kernel estimation using patch priors," in *Proc. ICCP*, 2013, pp. 1–8.
[10] L. Xu and J. Jia, "Two-phase kernel estimation for robust motion deblurring," in *Proc. ECCV*, 2010, pp. 157–170.
[11] L. Xu, S. Zheng, and J. Jia, "Unnatural L_0 sparse representation for natural image deblurring," in *Proc. CVPR*, 2013, pp. 1107–1114.
[12] T. S. Cho, S. Paris, B. K. Horn, and W. T. Freeman, "Blur kernel estimation using the radon transform," in *Proc. CVPR*, 2011, pp. 241–248.
[13] C. Schuler, H. Burger, S. Harmeling, and B. Scholkopf, "A machine learning approach for non-blind image deconvolution," in *Proc. CVPR*, 2013, pp. 1067–1074.
[14] M. Hradiš, J. Kotera, P. Zemcík, and F. Šroubek, "Convolutional neural networks for direct text deblurring," in *Proc. BMVC*, 2015, p. 10.
[15] P. Svoboda, M. Hradis, L. Marsik, and P. Zemcik, "CNN for license plate motion deblurring," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2016, pp. 3832–3836.
[16] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf, "Learning to deblur," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 7, pp. 1439–1451, 2016.
[17] L. Xu, J. S. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution," in *Proc. NIPS*, 2014, pp. 1790–1798.
[18] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Efficient marginal likelihood optimization in blind deconvolution," in *Proc. CVPR*, 2011, pp. 2657–2664.
[19] Y. Weiss and W. T. Freeman, "What makes a good model of natural images?" in *Proc. CVPR*, Jun. 2007, pp. 1–8.
[20] A. Levin, R. Fergus, F. Durand, and W. T. Freeman, "Image and depth from a conventional camera with a coded aperture," *ACM Trans. Graph.*, vol. 26, no. 3, p. 70, 2007.
[21] T. Michaeli and M. Irani, "Blind deblurring using internal patch recurrence," in *Proc. ECCV*, 2014, pp. 783–798.
[22] H. Cho, J. Wang, and S. Lee, "Text image deblurring using text-specific properties," in *Proc. ECCV*, 2012, pp. 524–537.
[23] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. ICCV*, 1998, pp. 839–846.
[24] S. Osher and L. I. Rudin, "Feature-oriented image enhancement using shock filters," *SIAM J. Numer. Anal.*, vol. 27, no. 4, pp. 919–940, 1990.
[25] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. ECCV*, 2014, pp. 184–199.
[26] D. Eigen, D. Krishnan, and R. Fergus, "Restoring an image taken through a window covered with dirt or rain," in *Proc. ICCV*, 2013, pp. 633–640.
[27] L. Xu, J. Ren, Q. Yan, R. Liao, and J. Jia, "Deep edge-aware filters," in *Proc. ICML*, 2015, pp. 1669–1678.
[28] A. Chakrabarti, "A neural approach to blind motion deblurring," in *Proc. ECCV*, 2016, pp. 221–235.
[29] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. ICML*, 2010, pp. 807–814.
[30] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image smoothing via L_0 gradient minimization," *ACM Trans. Graph. (SIGGRAPH)*, vol. 30, no. 6, p. 174, 2011.
[31] U. Schmidt, C. Rother, S. Nowozin, J. Jancsary, and S. Roth, "Discriminative non-blind deblurring," in *Proc. CVPR*, 2013, pp. 604–611.
[32] J. Pan, Z. Hu, Z. Su, and M.-H. Yang, "Deblurring face images with exemplars," in *Proc. ECCV*, 2014, pp. 47–62.
[33] 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.
[34] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce, "Non-uniform deblurring for shaken images," *Int. J. Comput. Vis.*, vol. 98, no. 2, pp. 168–186, 2012.
[35] P. Arbeláez, 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.
[36] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. AISTATS*, 2010, pp. 249–256.
[37] D. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980
[38] M. Hirsch, C. J. Schuler, S. Harmeling, and B. Schölkopf, "Fast removal of non-uniform camera shake," in *Proc. ICCV*, 2011, pp. 463–470.
[39] W. Ren, X. Cao, J. Pan, X. Guo, W. Zuo, and M.-H. Yang, "Image deblurring via enhanced low-rank prior," *IEEE Trans. Image Process.*, vol. 25, no. 7, pp. 3426–3437, Jul. 2016.
[40] J. Pan, Z. Hu, Z. Su, and M.-H. Yang, "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.
[41] R. Liao, X. Tao, R. Li, Z. Ma, and J. Jia, "Video super-resolution via deep draft-ensemble learning," in *Proc. ICCV*, 2015, pp. 531–539.
[42] J. Pan, Z. Lin, Z. Su, and M.-H. Yang, "Robust kernel estimation with outliers handling for image deblurring," in *Proc. CVPR*, 2016, pp. 2800–2808.

Xiangyu Xu received the B.E. degree from the Department of Electronic Engineering, Tsinghua University, China, in 2013. He is currently pursuing the joint-training Ph.D. degree with the Department of Electronic Engineering, Tsinghua University, and electrical engineering and computer science with the University of California at Merced, Merced, CA, USA. His research interest includes image deblurring and machine learning.

Jinshan Pan received the Ph.D. degree in computational mathematics from the Dalian University of Technology, China, in 2017. He was a joint-training Ph.D. student with the School of Mathematical Sciences and the School of Electrical Engineering and Computer Science, University of California at Merced, Merced, CA, USA, from 2014 to 2016. He is currently a Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology. His research interest includes image deblurring, image/video analysis and enhancement, and related vision problems.

Yu-Jin Zhang (SM'99) received the Ph.D. degree in applied science from the Montefiore Institute, State University of Liege, Liege, Belgium, in 1989. He was a Post-Doctoral Fellow and a Research Fellow with the Department of Applied Physics and the Department of Electrical Engineering, Delft University of Technology, Delft, The Netherlands, from 1989 to 1993. In 1993, he joined the Department of Electronic Engineering, Tsinghua University, Beijing, China, where he has been a Professor of image engineering since 1997. He has authored or coauthored over 40 books and over 500 papers in image processing, image analysis, and image understanding. He is a fellow of SPIE for achievements in image engineering. He is the Vice-President of the China Society of Image and Graphics.

Ming-Hsuan Yang (M'92–SM'06) received the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign, USA, in 2000. He is currently a Professor of electrical engineering and computer science with the University of California at Merced, Merced, CA, USA. He is a Senior Member of the ACM. He received the NSF CAREER Award in 2012 and the Google Faculty Award in 2009. He served as an Associate Editor of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE from 2007 to 2011. He is an Associate Editor of the *International Journal of Computer Vision, Image and Vision Computing* and the *Journal of Artificial Intelligence Research*.