From Motion Blur to Motion Flow: a Deep Learning Solution for Removing Heterogeneous Motion Blur

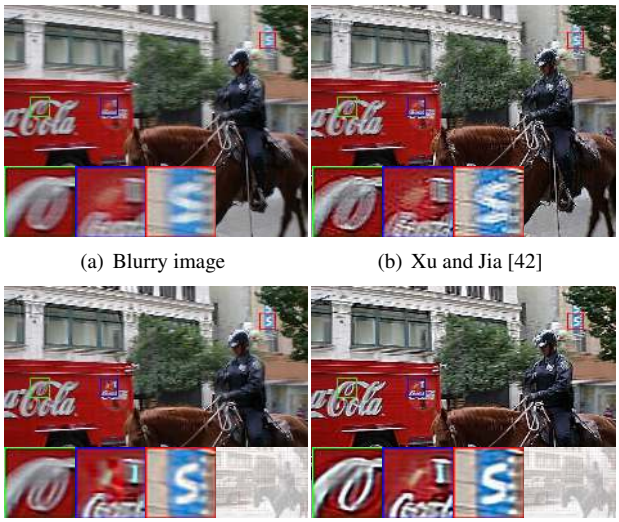
Dong Gong[†‡], Jie Yang[‡], Lingqiao Liu[‡§], Yanning Zhang[†], Ian Reid[‡§],
Chunhua Shen[‡§], Anton van den Hengel[‡§], Qinfeng Shi[‡*]
[†]School of Computer Science and Engineering, Northwestern Polytechnical University, China
[‡]The University of Adelaide, [§]Australian Centre for Robotic Vision
https://donggong1.github.io/blur2mflow

Abstract

Removing pixel-wise heterogeneous motion blur is challenging due to the ill-posed nature of the problem. The predominant solution is to estimate the blur kernel by adding a prior, but extensive literature on the subject indicates the difficulty in identifying a prior which is suitably informative, and general. Rather than imposing a prior based on theory, we propose instead to learn one from the data. Learning a prior over the latent image would require modeling all possible image content. The critical observation underpinning our approach, however, is that learning the motion flow instead allows the model to focus on the cause of the blur, irrespective of the image content. This is a much easier learning task, but it also avoids the iterative process through which latent image priors are typically applied. Our approach directly estimates the motion flow from the blurred image through a fully-convolutional deep neural network (FCN) and recovers the unblurred image from the estimated motion flow. Our FCN is the first universal end-to-end mapping from the blurred image to the dense motion flow. To train the FCN, we simulate motion flows to generate synthetic blurred-image-motion-flow pairs thus avoiding the need for human labeling. Extensive experiments on challenging realistic blurred images demonstrate that the proposed method outperforms the state-of-the-art.

1. Introduction

Motion blur is ubiquitous in photography, especially when using light-weight mobile devices, such as cellphones and on-board cameras. While there has been significant progress in image deblurring [9, 6, 42, 27, 28, 10], most work focuses on *spatially-uniform* blur. Some recent methods [40, 12, 14, 18, 26, 32] have been proposed to remove *spatially-varying* blur caused by camera panning, and/or object movement, with some restrictive assumptions on the types of blur, image prior, or both. In this work, we focus on recovering a blur-free latent image from a single observation degraded by *heterogeneous motion blur*, *i.e.* the blur kernels may independently vary from pixel to pixel.



(a) Blurry image (b) Xu and Jia [42]
(c) Sun *et al.* [33] (d) Ours

Figure 1. A blurry image with heterogeneous motion blur from a widely used dataset Microsoft COCO [23]. Estimated motion flows are shown in the bottom right corner of each image.

Motion blur in real images has a variety of causes, including camera [40, 47] and object motion [15, 26], leading to blur patterns with complex variations (See Figure 1 (a)). In practice, uniform deblurring methods [9, 6, 42] usually fail to remove the non-uniform blur (See Figure 1 (b)). Most existing non-uniform deblurring methods rely on a specific motion model, such as 3D camera motion modeling

*This work was supported by NSFC (61231016, 61572405), China 863 (2015AA016402), ARC (DP160100703), ARC Centre for Robotic Vision CE140100016; an ARC Laureate Fellowship FL130100102 to I. Reid, and an ARC DECRA Fellowship DE170101259 to L. Liu. D. Gong was supported by a scholarship from CSC.

[11, 40] and segment-wise motion [20, 26]. Although a recent method [18] uses a flexible *motion flow* map to handle heterogeneous motion blur, it requires a time-consuming iterative estimator. In addition to the assumptions about the cause of blur, most existing deblurring methods also rely on predefined priors or manually designed image features. Most conventional methods [9, 22, 44] need to iteratively update the intermediate image and the blur kernel with using these predefined image priors to reduce the ill-posedness. Solving these non-convex problems is non-trivial, and many real images do not conform to the assumptions behind a particular model. Recently, learning-based discriminative methods [4, 7] have been proposed to learn blur image patterns and avoid the heavy computational cost of blur estimation. However, their representation and prediction abilities are limited by their manually designed features and simple mapping functions. Although a deep learning based method [33] aimed to overcome these problems, it restrictively conducts the learning process at the patch-level and thus cannot take full advantage of the context information from larger image regions.

In summary, there are three main problems with existing approaches: 1) the range of applicable motion types is limited, 2) manually defined priors and image features may not reflect the nature of the data and 3) complicated and time-consuming optimization and/or post-processing is required. Generally, these problems limit the practical applicability of blur removal methods to real images, as they tend to cause worse artifacts than they cure.

To handle general heterogeneous motion blur, based on the motion flow model, we propose a deep neural network based method able to directly estimate a pixel-wise motion flow map from a single blurred image by learning from tens of thousands of examples. To summarize, the main contributions of this paper are:

- We propose an approach to estimate and remove pixel-wise heterogeneous motion blur by training on simulated examples. Our method uses a flexible blur model and makes almost no assumptions about the underlying images, resulting in effectiveness on diverse data.
- We introduce a universal FCN for end-to-end estimation of dense heterogeneous motion flow from a single blurry image. Beyond the previous patch-level learning [33], we directly perform training and testing on the whole image, which utilizes the spatial context over a wider area and estimates a dense motion flow map accurately. Moreover, our method does not require any post-processing.

2. Related Work

Conventional blind image deblurring To constrain the solution space for blind deblurring, a common assumption is that image blur is spatially uniform [5, 6, 9, 22, 28, 10]. Numerous image priors or regularizers have been studied to overcome the ill-posed nature of the problem, such as the total variational regularizer [5, 29], Gaussian scale mixture priors [9] and ℓ_1/ℓ_2-norms [19], ℓ_0-norms [44, 27], and dark channel [28] based regularizers. Various estimators have been proposed for more robust kernel estimation, such as edge-extraction-based maximum-a-posteriori (MAP) [6, 34], gradient activation based MAP [10], variational Bayesian methods [21, 22, 46], *etc* . Although these powerful priors and estimators work well on many benchmark datasets, they are often characterised by restrictive assumptions that limit their practical applicability.

Spatially-varying blur removal To handle spatially-varying blur, more flexible blur models are proposed. In [35], a projective motion path model formulates a blurry image as the weighted sum of a set of transformed sharp images, an approach which is which is simplified and extended in [40] and [45]. Gupta *et al.* [11] model the camera motion as a motion density function for non-uniform deblurring. Several locally uniform overlapping-patch-based models [13, 12] are proposed to reduce the computational burden. Zheng *et al.* [47] specifically modelled the blur caused by forward camera motion. To handle blur caused by object motion, some methods [20, 8, 15, 26] segment images into areas with different types of blur, and are thus heavily dependent on an accruate segmentation of a blurred image. Recently, a pixel-wise linear motion model [18] is proposed to handle heterogeneous motion blur. Although the motion is assumed to be locally linear, there is no assumption on the latent motion, making it flexible enough to handle an extensive range of possible motion.

Learning based motion blur removing Recently, learning based methods have been used to achieve more flexible and efficient blur removal. Some discriminative methods are proposed for non-blind deconvolution based on Gaussian CRF [30], multi-layer perceptron (MLP) [31], and deep convolution neural network (CNN) [43], etc, which all require the known blur kernels. Some end-to-end methods [17, 25] are proposed to reconstruct blur-free images, however, they can only handle mild Gaussian blur. Recently, Wieschollek *et al.* [41] introduce an MLP based blind deblurring method by using information in multiple images with small variations. Chakrabarti [3] trains a patch-based neural network to estimate the frequency information for uniform motion blur removal. The most relevant work is a method based on CNN and patch-level blur type classification [33], which also focuses on estimating the motion flow from single blurry image. The authors train a CNN on small patch examples with *uniform* motion blur, where each patch is assigned a single motion label, violating the real data nature and ignoring the correspondence in larger areas. Many post-processing such as MRF are required for the final dense motion flow.

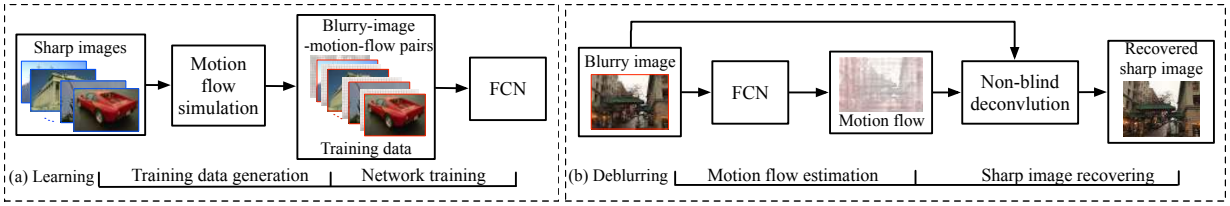


Figure 2. Overview of our scheme for heterogeneous motion blur removal. (a) We train an FCN using examples based on simulated motion flow maps. (b) Given a blurry image, we perform end-to-end motion flow estimation using the trained FCN, and then recover the sharp image via non-blind deconvolution.

3. Estimating Motion Flow for Blur Removal

3.1. A Heterogeneous Motion Blur Model

Letting $*$ denote a general convolution operator, a $P \times Q$ blurred image $\mathbf{Y}$ can be modeled as

$$\mathbf{Y} = \mathcal{K} * \mathbf{X} + \mathbf{N}, \tag{1}$$

where $\mathbf{X}$ denotes the latent sharp image, $\mathbf{N}$ refers to additive noise, and $\mathcal{K}$ denotes a heterogeneous motion blur kernel map with different blur kernels for each pixel in $\mathbf{X}$. Let $\mathcal{K}_{(i,j)}$ represent the kernel from $\mathcal{K}$ that operates on a region of the image centered at pixel (i, j). Thus, at each pixel of $\mathbf{Y}$, we have

$$\mathbf{Y}(i,j) = \sum_{i',j'} \mathcal{K}_{(i,j)}(i',j')\mathbf{X}(i+i', j+j'). \tag{2}$$

If we define an operator $\text{vec}(\cdot)$ which vectorises a matrix and let $\mathbf{y} = \text{vec}(\mathbf{Y})$, $\mathbf{x} = \text{vec}(\mathbf{X})$ and $\mathbf{n} = \text{vec}(\mathbf{N})$ then (1) can also be represented as

$$\mathbf{y} = \mathbf{H}(\mathcal{K})\mathbf{x} + \mathbf{n}, \tag{3}$$

where $\mathbf{H}(\mathcal{K}) \in \mathbb{R}^{PQ \times PQ}$[1] and each row corresponds to a blur kernel located at each pixel (*i.e.* $\mathcal{K}_{(i,j)}$).

3.2. Blur Removal via Motion Flow Estimation

Given a blurry image $\mathbf{Y}$, our goal is to estimate the blur kernel $\mathcal{K}$ and recover a blur-free latent image $\mathbf{X}$ through non-blind deconvolution that can be performed by solving a convex problem (Figure 2 (b)). As mentioned above, kernel estimation is the most difficult and crucial part.

Based on the model in (1) and (2), heterogeneous motion blur can be modeled by a set of blur kernels, one associated with each pixel and its motion. By using a linear motion model to indicate each pixel's motion during imaging process [18], and letting $\mathbf{p} = (i, j)$ denote a pixel location, the motion at pixel $\mathbf{p}$, can be represented by a 2-dimensional *motion vector* $(u_\mathbf{p}, v_\mathbf{p})$, where $u_\mathbf{p}$ and $v_\mathbf{p}$ represent the movement in the horizontal and vertical directions, respectively (See Figure 3 (a)). By a slight abuse of notation we express this as $\mathcal{M}_\mathbf{p} = (u_\mathbf{p}, v_\mathbf{p})$, which characterizes the movement at pixel $\mathbf{p}$ over the exposure time. If we have the feasible domain $u_\mathbf{p} \in \mathbb{D}_u$ and $v_\mathbf{p} \in \mathbb{D}_v$, then $\mathcal{M}_\mathbf{p} \in \mathbb{D}_u \times \mathbb{D}_v$, but will be introduced in detail later. As shown in Figure 3, the blur kernel on each pixel appears as a line trace with nonzero components only along the motion trace. As a result, the motion blur $\mathcal{K}_\mathbf{p}$ in (2) can be expressed as [2]:

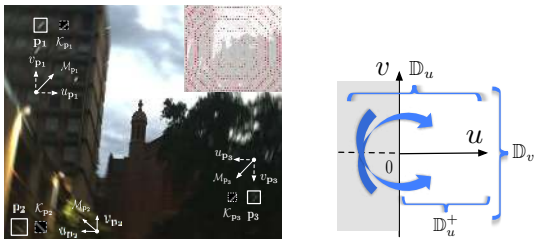
$$\mathcal{K}_\mathbf{p}(i',j') = \begin{cases} 0, & \text{if } \|(i',j')\|_2 \geq \frac{\|\mathcal{M}_\mathbf{p}\|_2}{2}, \\ \frac{1}{\|\mathcal{M}_\mathbf{p}\|_2}\delta(v_\mathbf{p} i' - u_\mathbf{p} j'), & \text{otherwise}, \end{cases} \tag{4}$$

where $\delta(\cdot)$ denotes the Dirac delta function. We thus can achieve heterogeneous motion blur estimation by estimating the motion vectors on all pixels, the result of which is $\mathcal{M}$, which is referred as *motion flow*. For convenience of expression, we let $\mathcal{M} = (\mathbf{U}, \mathbf{V})$, where $\mathbf{U}$ and $\mathbf{V}$ denote the motion maps in the horizontal and vertical directions, respectively. For any pixel $\mathbf{p} = (i, j)$, we define $\mathcal{M}_\mathbf{p} = (\mathbf{U}(i,j), \mathbf{V}(i,j))$ with $\mathbf{U}(i,j) = u_\mathbf{p}$ and $\mathbf{V}(i,j) = v_\mathbf{p}$.

As shown in Figure 2 (b), given a blurred image and the estimated motion flow, we can recover the sharp image by solving an non-blind deconvolution problem

$$\min_\mathbf{x} \|\mathbf{y} - \mathbf{H}(\mathcal{K})\mathbf{x}\|_2^2 + \Omega(\mathbf{x})$$

with regularizer $\Omega(\mathbf{x})$ on the unknown sharp image. In practice, we use a Gaussian mixture model based regularizer as $\Omega(\mathbf{x})$ [48, 33].



(a) Motion blur and motion flow (b) Domain of motion

Figure 3. Motion blur and motion vector. (a) An example with blur cause by clock-wise rotation. Three examples of the blur pattern, linear blur kernel and motion vector are shown. The blur kernels on $\mathbf{p}_1$ and $\mathbf{p}_3$ caused by motions with opposite directions and have the same appearance. (b) Illustrations of the feasible domain of motion flow.

[1]For simplicity, we assume $\mathbf{X}$ and $\mathbf{Y}$ have the same size.

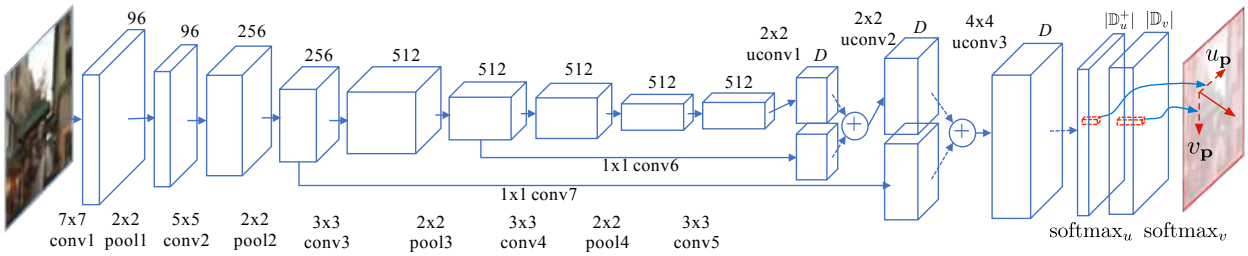


Figure 4. Our network structure. A blurred image goes through layers and produces a pixel-wise dense motion flow map. conv means a convolutional layer and uconv means a fractionally-strided convolutional (deconvolutional) layer, where $n \times n$ for each uconv layer denotes that the up-sampling size is n. Skip connections on top of pool2 and pool3 are used to combine features with different resolutions.

3.3. Learning for Motion Flow Estimation

The key contribution of our work is to show how to obtain the motion flow field that results in the pixel-wise motion blur. To do so we train a FCN to directly estimate the motion flow field from the blurry image.

Let $\{(\mathbf{Y}^t, \mathcal{M}^t)\}_{t=1}^T$ be a set of blurred-image and motion-flow-map pairs, which we take as our training set. Our task is to learn an end-to-end mapping function $\mathcal{M} = f(\mathbf{Y})$ from any observed blurry image $\mathbf{Y}$ to the underlying motion flow $\mathcal{M}$. In practice, the challenge is that obtaining the training ground-truth dense motion flow for sufficiently many and varied real blurry images is infeasible. Human labeling is impossible, and training from automated methods for image deblurring would defeat the purpose. To overcome this problem, we generate the training set by simulating motion flows maps. (See section 4.2). Specifically, we collect a set of sharp images $\{\mathbf{X}^n\}$, simulate T motion flows $\{\mathcal{M}^t\}$ in total for all images in $\{\mathbf{X}^n\}$, and then generate T blurred images $\{\mathbf{Y}^t\}$ based on the models in (1) and (4) (See Figure 2 (a)).

Feasible domain of motion flow To simplify the training process, we train the FCN over a discrete output domain. Interestingly, classification on discrete output space has achieved some impressive results for some similar applications, *e.g.* optical flow estimation [36] and surface normal prediction [37]. In our work, we adopt an integer domain for both $\mathbf{U}$ and $\mathbf{V}$, and treat the mapping $\mathcal{M} = f(\mathbf{Y})$ as a multi-class classification problem. Specifically, we uniformly discretize the motion values as integers with a 1 (pixel) interval, which provides a high-precision approximation to the latent continuous space. As a result, by assuming the maximum movements in the horizontal and vertical directions to be u_{max} and v_{max}, respectively, we have $\mathbb{D}_u = \{u | u \in \mathbb{Z}, |u| \leq u_{max}\}$ and $\mathbb{D}_v = \{v | v \in \mathbb{Z}, |v| \leq v_{max}\}$, where $\mathbb{Z}$ denotes the integer domain.

As shown in Figure 3 (a), any linear blur kernel is symmetric. Any two motion vectors with same length and opposite directions, *e.g.* $(u_\mathbf{p}, v_\mathbf{p})$ and $(-u_\mathbf{p}, -v_\mathbf{p})$, generate the same blur pattern, which may confuse the learning process. We thus further restrict the motion in the horizontal direction to be nonnegative as shown in Figure 3 (b), *i.e.* $u_\mathbf{p} \in \mathbb{D}_u^+ = \{u | u \in \mathbb{Z}_0^+, |u| \leq u_{max}\}$, by letting $(u_\mathbf{p}, v_\mathbf{p}) = \phi(u_\mathbf{p}, v_\mathbf{p})$ where

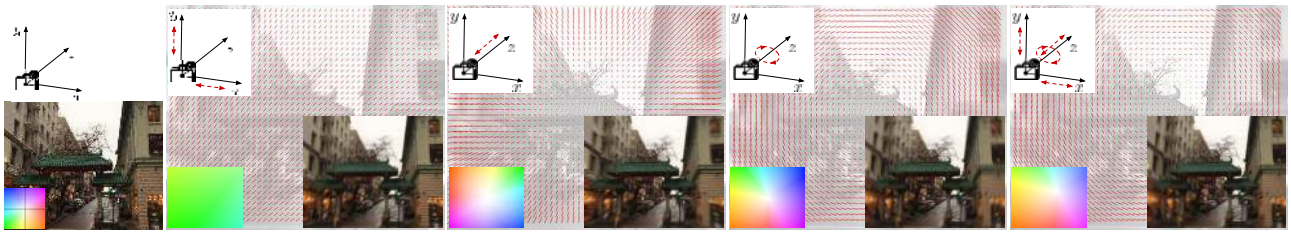
$$\phi(u_\mathbf{p}, v_\mathbf{p}) = \begin{cases} (-u_\mathbf{p}, -v_\mathbf{p}), & \text{if } u_\mathbf{p} < 0, \\ (u_\mathbf{p}, v_\mathbf{p}), & \text{otherwise.} \end{cases} \tag{5}$$

4. Dense Motion Flow Estimation

4.1. Network Design

The goal of this FCN network is to achieve the end-to-end mapping from a blurry image to its corresponding motion flow map. Given any RGB image with the arbitrary size $P \times Q$, the FCN is used to estimate a motion flow map $\mathcal{M} = (\mathbf{U}, \mathbf{V})$ with the same size to the input image, where $\mathbf{U}(i,j) \in \mathbb{D}_u^+$ and $\mathbf{V}(i,j) \in \mathbb{D}_v$, $\forall i, j$. For convenience, we let $D = |\mathbb{D}_u^+| + |\mathbb{D}_v|$ denote the total number of labels for both $\mathbf{U}$ and $\mathbf{V}$. Our network structure is similar to the FCN in [24]. As shown in Figure 4, we use 7 convolutional (conv) layers and 4 max-pooling (pool) layers as well as 3 uconv layers to up-sample the prediction maps. Following [38], uconv denotes the fractionally-strided convolution, a.k.a. deconvolution. We use a small stride of 1 pixel for all convolutional layers. The uconv layers are initialized with bilinear interpolation and used to up-sample the activations. We also add skip connections which combine the information from different layers as shown in Figure 4.

The feature map of the last uconv layer (conv7 + uconv2) is a $P \times Q \times D$ tensor with the top $|\mathbb{D}_u^+|$ slices of feature maps ($P \times Q \times |\mathbb{D}_u^+|$) corresponding to the estimation of $\mathbf{U}$, and the remaining $|\mathbb{D}_v|$ slices of feature maps ($P \times Q \times |\mathbb{D}_v|$) corresponding to the estimation of $\mathbf{V}$. Two separate soft-max layers are applied to those two parts respectively to obtain the posterior probability estimation of both channels. Let $F_{u,i,j}(\mathbf{Y})$ represent the probability that the pixel at (i,j) having a movement u along the horizontal direction, and $F_{v,i,j}(\mathbf{Y})$ represent the probability that the pixel at (i,j) having a movement v along the vertical direction, we then use the sum of the cross entropy loss from



(a) Sharp Image (b) x and y-axis translation (c) z-axis translation (d) z-axis rotation (e) Arbitrary sampled motion

Figure 5. Demonstration of the motion flow simulation. (a) A sharp example image and the coordinate system of camera. (b)-(c) The sampled motion flow and the corresponding blurred image by simulating the translation along x and y-axes ($\mathcal{M}_{T_x} + \mathcal{M}_{T_y}$), translation along z-axis ($\mathcal{M}_{T_z}$) and rotation around z-axis ($\mathcal{M}_{R_z}$), respectively. (d) A sample based on the model considering all components in (6).

both channels as the final loss function:

$$L(\mathbf{Y}, \mathcal{M}) = -\sum_{i=1}^{P}\sum_{j=1}^{Q}\sum_{u\in\mathbb{D}_u^+} \mathbb{1}(\mathbf{U}(i,j)=u)\log(F_{u,i,j}(\mathbf{Y})) - \sum_{i=1}^{P}\sum_{j=1}^{Q}\sum_{v\in\mathbb{D}_v} \mathbb{1}(\mathbf{V}(i,j)=v)\log(F_{v,i,j}(\mathbf{Y})),$$

where $\mathbb{1}$ is an indicator function.

4.2. Simulate Motion Flow for Data Generation

The gist of this section is generating a dataset that contains realistic blur patterns on diverse images for training. Although an i.i.d. random sampling may generate very diverse training samples, since the realistic motion flow preserves some properties such as piece-wise smoothness, we aim to design a simulation method to generate motion flows reflecting the natural properties of the movement in imaging process. Although the object motion [15] can lead to heterogeneous motion blur in real images, our method only simulates the motion flow caused by camera motion for learning. Even so, as shown in Section 5.5, data generated by our method can also give the model certain ability to handle object motion.

For simplicity, we generate a 3D coordinate system where the origin at the camera's optical center, the xy-plane is aligned with the camera sensors, and the z-axis is perpendicular to the xy-plane, as shown in Figure 5. Since our objective is the motion flow on an image grid, we directly simulate the motion flow projected on 2D image instead of the 3D motion trajectory [40]. Considering the ambiguities caused by rotations around x and y axis [11], we simulate a motion flow $\mathcal{M}$ by sampling four additive components:

$$\mathcal{M} = \mathcal{M}_{T_x} + \mathcal{M}_{T_y} + \mathcal{M}_{T_z} + \mathcal{M}_{R_z}, \tag{6}$$

where $\mathcal{M}_{T_x}$, $\mathcal{M}_{T_y}$ and $\mathcal{M}_{T_z}$ denote the motion flows associated with the translations along x, y and z axis, receptively, and $\mathcal{M}_{R_z}$ represents the motion from the rotation around z axis. We generate each element as the following.

Translation along x or y axis We describe the generation of $\mathcal{M}_{T_x}$ as an example. We first sample a central pixel $\mathbf{p}_{T_x} = (i_{T_x}, j_{T_x})$ on image plane, a basic motion value t_{T_x} and a acceleration coefficient r_{T_x}. Then $\mathcal{M}_{T_x} = (\mathbf{U}_{T_x}, \mathbf{V}_{T_x})$ can be generated as the following $\mathbf{U}_{T_x}(i,j) = (i - i_{T_x})r_{T_x} + t_{T_x}, \mathbf{V}_{T_x}(i,j) = 0$. $\mathcal{M}_{T_y}$ can be generated in a similar way.

Translation along z axis The translation along z axis usually causes radial motion blur pattern towards the vanishing point [47]. By ignoring the semantic context and assuming a simple radial pattern, $\mathcal{M}_{T_z}$ can be generated by $\mathbf{U}_{T_z}(i,j) = t_{T_z} d(i,j)^{\zeta}(i - i_{T_z}), \mathbf{V}_{T_z}(i,j) = t_{T_z} d(i,j)^{\zeta}(j - j_{T_z})$ where $\mathbf{p}_{T_z}$ denotes a sampled vanishing point, $d(i,j) = \|(i,j) - \mathbf{p}_{T_z}\|_2$ is the distance from any pixel (i,j) to the vanishing point, ζ and t_{T_z} are used to control the shape of radial patterns, which reflects the moving speed.

Rotation around z axis We first sample a rotation center $\mathbf{p}_{R_z}$ and an angular velocity ω, where $\omega > 0$ denotes the clockwise rotation. Let $d(i,j) = \|(i,j) - \mathbf{p}_{R_z}\|_2$. The motion magnitude at each pixel is $s(i,j) = 2d(i,j)\tan(\omega/2)$. By letting $\theta(i,j) = \text{atan}[(i - i_{R_z})/(j - j_{R_z})] \in [-\pi, \pi]$, motion vector at pixel (i,j) can be generated as $\mathbf{U}_{R_z}(i,j) = s(i,j)\cos(\theta(i,j) - \pi/2), \mathbf{V}_{R_z}(i,j) = s(i,j)\sin(\theta(i,j) - \pi/2)$.

We place uniform priors over all the parameters corresponding to the motion flow simulation as Uniform(α, β). More details can be found in supplementary materials. Note that the four components in (6) are simulated in continuous domain and are then discretized as integers.

Training dataset generation We use 200 training images with sizes around 300×460 from the dataset BSD500 [1] as our sharp image set $\{\mathbf{X}^n\}$. We then independently simulate 10,000 motion flow maps $\{\mathcal{M}^t\}$ with ranges $u_{max} = v_{max} = 36$ and assign each $\mathbf{X}^n$ 50 motion flow maps without duplication. The non-blurred images $\{\mathbf{X}^n\}$ with $\mathbf{U}(i,j) = 0$ and $\mathbf{V}(i,j) = 0$, $\forall i,j$ are used for training. As a result we have a dataset with 10,200 blurred-image-motion-flow pairs $\{\mathbf{Y}^t, \mathcal{M}^t\}$ for training.

Table 1. Evaluation on motion blur estimation. Comparison on PSNR and SSIM of the recovered images with the estimated blur kernel.

Dataset	Metric	GT $\mathcal{K}$	Xu and Jia [42]	Whyte *et al.* [40]	Xu *et al.* [44]	noMRF [33]	patchCNN [33]	Ours
BSD-S	PSNR	23.022	17.773	17.360	18.351	20.483	20.534	**21.947**
	SSIM	0.6609	0.4431	0.3910	0.4766	0.5272	0.5296	**0.6309**
BSD-M	PSNR	24.655	19.673	18.451	20.057	22.789	22.9683	**23.978**
	SSIM	0.7481	0.5661	0.5010	0.5973	0.6666	0.6735	**0.7249**

5. Experiments

We implement our model based on Caffe [16] and train it by stochastic gradient descent with momentum and batch size 1. In the training on the dataset simulated on BSD, we use a learning rate of 10^{-9} and a step size of 2×10^5. The training converges after 65 epochs. The code can be found at `https://donggong1.github.io/blur2mflow.html`.

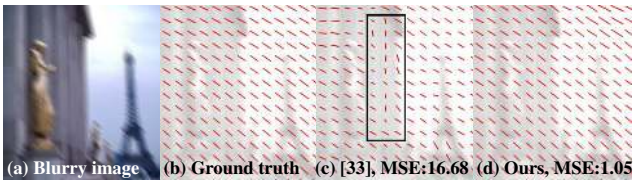


Figure 6. A motion flow estimation example on a synthetic image in BSD-M. The method of Sun *et al.* [33] is more sensitive to the image content (See the black box in (c)).

5.1. Datasets and Evaluation Metrics

Datasets We conduct the experiments on both *synthetic datasets* and *real-world images*. Since ground truth motion flow and sharp image for real blurry image are difficult to obtain, to perform general quantitative evaluation, we first generate two synthetic datasets, which both contain 300 blurred images, with 100 sharp images randomly picked from BSD500 [1][2], and 3 different motion flow maps for each. Note that no two motion flow maps are the same. We simulate the motion flow with $u_{max} = v_{max} = 36$, which is same as in the training set. For fairness to the method [33] with a smaller output space, we also generate relative mild motion flows for the second dataset with $u_{max} = v_{max} = 17$. These two are referred as **BSD-S** and **BSD-M**, respectively. In addition, we evaluate the generalization ability of the proposed method using two synthetic datasets (**MC-S** and **MC-M**) with 60 blurry images generated from 20 sharp images from Microsoft COCO [23] and above motion flow generation setting.

Evaluation Metrics For evaluating the accuracy of the motion flow, we measure the mean-squared-error (**MSE**) of the motion flow map. Specifically, given an estimated motion flow $\widehat{\mathcal{M}}$ and the ground truth $\mathcal{M}$, the MSE is defined as $\frac{1}{2|M|}\sum_{i,j}((\mathbf{U}(i,j)-\widehat{\mathbf{U}}(i,j))^2+((\mathbf{V}(i,j)-\widehat{\mathbf{V}}(i,j))^2$, where $|\mathcal{M}|$ denotes the number of motion vectors in $\mathcal{M}$. For evaluating the image quality, we adopt peak signal-to-noise-ratio (**PSNR**) and structural similarity index (**SSIM**) [39].

5.2. Evaluation of Motion Flow Estimation

We first compare with the method of Sun *et al.* ("patchCNN") [33], the only method with available code for estimating motion flow from blurry images[3]. This method performs training and testing on small image patches, and uses MRF to improve the accuracy on the entire image. Its version without MRF post-processing ("noMRF") is also compared, where the soft-max output is directly used to get the motion flow as in our method. Table 2 shows the average MSE of the estimated motion flow maps on all images in BSD-S and BSD-M. It is noteworthy that, even without any post-processing such as MRF or CRF, the comparison manifests the high quality of our estimated motion flow maps. Furthermore, our method can still produce accurate motion flow even on the more challenging BSD-S dataset, on which the accuracies of the patch based method [33] decrease significantly. We also show an example of the the estimated motion flows in Figure 6, which shows that our method achieves motion flow very similar to the ground truth, and the method of Sun *et al.* [33] is more sensitive to the image contents. From this example, we can see that the method of Sun *et al.* [33] generally underestimates the motion values and produces errors near the strong edges, maybe because its patch-level processing is confused by the strong edges and ignores the blur pattern context in a larger area.

Table 2. Evaluation on motion flow estimation (MSE).

Dataset	patchCNN [33]	noMRF [33]	Ours
BSD-S	50.1168	54.4863	**6.6198**
BSD-M	15.6389	20.7761	**5.2051**

To compare with other blind deblurring methods of Xu and Jia [42], Xu *et al.* [44] and Whyte *et al.* [40], which do not estimate the motion flow, we directly evaluate the quality of the image recovered using their estimated blur kernel. For fairness, we use the same non-blind deconvolution method with least square loss function and a Gaussian mixture model prior [48] to recover the sharp image. As the non-blind deconvolution method may limit the recovering quality, we evaluate the images recovered using the ground truth motion flow as reference. As shown in 1, our method produces significantly better results than the others.

[2]No overlapping with the training dataset.

[3]The code of the other motion flow based method [18] is unavailable.

Figure 7. Examples of motion flow estimation on real-world blurry images. From top to bottom: Blurry image $\mathbf{Y}$, motion flow estimated by the patchCNN [33], and by our motion flow $\mathcal{M}$. Our results are more smooth and more accurate on moving objects.

5.3. Evaluation of Generalization Ability

To evaluate the generalization ability of our approach on different images, we use the datasets based on the Microsoft COCO [23] (*i.e.* MC-S and MC-M) to evaluate our model trained on the dataset based on BSD500 [1]. Table 3 shows the evaluation and comparison with the "patchCNN" [33]. The results demonstrate that our method stably produces high accuracies on both datasets. This experiment suggests that the generalization ability of our approach is strong.

Table 3. Evaluation of the generalization ability on datasets MC-S and MC-M. The best results are bold-faced.

Dataset	Metric	GT $\mathcal{K}$	patchCNN	noMRF [33]	Ours
MC-S	MSE	–	52.1234	60.9397	**7.8038**
	PSNR	22.620	20.172	20.217	**21.954**
	SSIM	0.6953	0.5764	0.5772	**0.6641**
MC-M	MSE	–	22.4383	31.2754	**7.3405**
	PSNR	23.827	22.186	22.028	**23.227**
	SSIM	0.7620	0.6924	0.6839	**0.7402**

5.4. Running-time Evaluation

We conduct a running-time comparison with the relevant motion flow estimation methods [33, 18] by estimating motion flow for 60 blurred images with sizes around 640×480 on a PC with an NVIDIA GeForce 980 Ti graphics card and Intel Core i7 CPU. For the method in [18], we quote its running-time from the paper. Note that both the method of Sun *et al.* and ours use the GPU to accelerate the computation. As shown in Table 4, the method in [18] takes very long time due to its iterative optimization scheme. Our method takes less than 10 seconds, which is more efficient than others. The patchCNN method [33] takes more time because many post-processing steps are required.

Table 4. Running-time comparison.

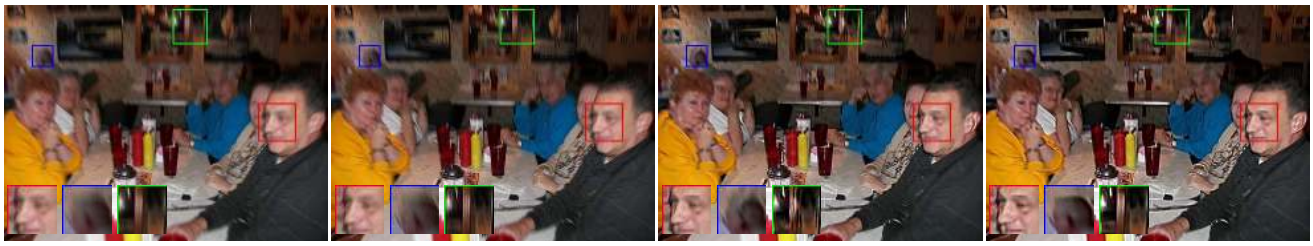
Method	[18]	patchCNN [33]	noMRF [33]	Ours
Time (s)	1500	45.2	18.5	**8.4**

5.5. Evaluation on Real-world Images

As the ground truth images of real-world blurry images are unavailable, we only present the visual evaluation and comparison against several state-of-the-art methods for spatially-varying blur removing.

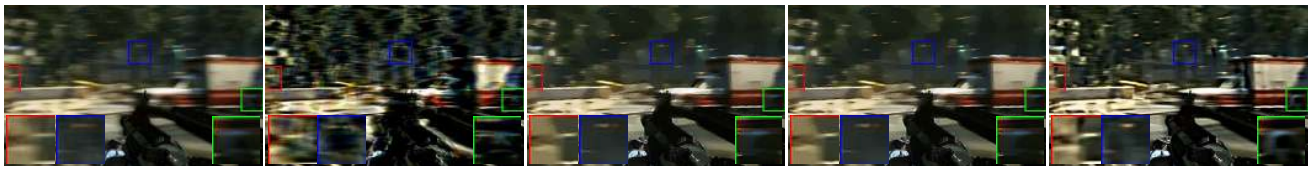
Results of motion flow estimation We first compare the proposed method with the method of Sun *et al.* [33] on motion flow estimation. Four examples are shown in Figure 7. Since the method of Sun *et al.* performs on local patches, their motion flow components are often misestimated, especially when the blur pattern in a small local area is subtle or confusing, such as the areas with low illumination or textures. Thanks to the universal end-to-end mapping, our method generates natural results with smooth flow and less clutters. Although we train our model with only smoothly varying motion flow, compared with [33], our method can obtain better results on images with moving object.

Comparison with the method in [18] Kim *et al.* [18] use the similar heterogeneous motion blur model as ours and also estimate motion flow for deblurring. As their code is unavailable, we directly perform a comparison on their real-world data. Figure 11 shows the results on an example. Compared with the results of Kim and Lee [18], our motion flow more accurately reflects the complex blur pattern, and



(a) Blurry image (b) Whyte *et al.* [40] (c) Sun *et al.* [33] (d) Ours

Figure 8. Deblurring results on an image with camera motion blur.



(a) Blurry image (b) Whyte *et al.* [40] (c) Kim and Lee [18] (d) Sun *et al.* [33] (e) Ours

Figure 9. Deblurring results on an non-uniform blur image with strong blur on background.

(a) Blurry image (b) Pan *et al.* [26] (c) Sun *et al.* [33] (d) Ours

Figure 10. Deblurring results on an image with large scale motion blur caused by moving object.

Figure 11. Comparison with the method of Kim and Lee [18].

our recovered image contains more details and less artifacts.

Images with camera motion blur Figure 8 shows an example containing blur mainly caused by the camera motion. The result generated by the non-uniform camera shake deblurring method [40] suffers from heavy blur because its model ignores the blur caused by large forward motion. Compared with the result of Sun *et al.* [33], our result is sharper and contains more details and less artifacts.

Images with object motion blur We evaluate our method on the images containing object motion blur. In Figure 9, the result of Whyte *et al.* [40] contains heavy ringing artifacts due to the object motion. Our method can handle the strong blur in the background and generate a more natural image. We further compare with the segmentation-based deblurring method of Pan *et al.* [26] on an image with large scale blur caused by moving object on static background. As shown in Figure 10, the result of Sun *et al.* [33] is over-smooth due to the underestimate of motion flow. In the result of Pan *et al.* [26], some details are lost due to the segmentation error. Our proposed method can recover the details on blurred moving foreground and keep the sharp background as original.

6. Conclusion

In this paper, we proposed a flexible and efficient deep learning based method for estimating and removing the heterogeneous motion blur. By representing the heterogeneous motion blur as pixel-wise linear motion blur, the proposed method uses a FCN to estimate the a dense motion flow map for blur removal. Moreover, we automatically generate training data with simulated motion flow maps for training the FCN. Experimental results on both synthetic and real-world data show the excellence of the proposed method.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(5):898–916, 2011.

[2] F. Brusius, U. Schwanecke, and P. Barth. Blind image deconvolution of linear motion blur. In *International Conference on Computer Vision, Imaging and Computer Graphics*, pages 105–119. Springer, 2011.

[3] A. Chakrabarti. A neural approach to blind motion deblurring. *European Conference on Computer Vision (ECCV)*, 2016.

[4] A. Chakrabarti, T. Zickler, and W. T. Freeman. Analyzing spatially-varying blur. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2512–2519, 2010.

[5] T. F. Chan and C.-K. Wong. Total variation blind deconvolution. *IEEE Transactions on Image Processing*, 7(3):370–375, 1998.

[6] S. Cho and S. Lee. Fast motion deblurring. *SIGGRAPH ASIA*, 2009.

[7] F. Couzinie-Devy, J. Sun, K. Alahari, and J. Ponce. Learning to estimate and remove non-uniform image blur. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1075–1082, 2013.

[8] S. Dai and Y. Wu. Removing partial blur in a single image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2544–2551. IEEE, 2009.

[9] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. In *ACM Transactions on Graphics*, 2006.

[10] D. Gong, M. Tan, Y. Zhang, A. van den Hengel, and Q. Shi. Blind image deconvolution by automatic gradient activation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] A. Gupta, N. Joshi, C. L. Zitnick, M. Cohen, and B. Curless. Single image deblurring using motion density functions. In *European Conference on Computer Vision (ECCV)*, pages 171–184, 2010.

[12] M. Hirsch, C. J. Schuler, S. Harmeling, and B. Schölkopf. Fast removal of non-uniform camera shake. In *The IEEE International Conference on Computer Vision (ICCV)*, 2011.

[13] M. Hirsch, S. Sra, B. Schölkopf, and S. Harmeling. Efficient filter flow for space-variant multiframe blind deconvolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 2, 2010.

[14] Z. Hu, L. Xu, and M.-H. Yang. Joint depth estimation and camera shake removal from single blurry image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2893–2900, 2014.

[15] T. Hyun Kim, B. Ahn, and K. Mu Lee. Dynamic scene deblurring. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3160–3167, 2013.

[16] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv*, 2014.

[17] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[18] T. H. Kim and K. M. Lee. Segmentation-free dynamic scene deblurring. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[19] D. Krishnan, T. Tay, and R. Fergus. Blind deconvolution using a normalized sparsity measure. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 233–240, 2011.

[20] A. Levin. Blind motion deblurring using image statistics. In *Advances in Neural Information Processing Systems (NIPS)*, pages 841–848, 2006.

[21] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1964–1971, 2009.

[22] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Efficient marginal likelihood optimization in blind deconvolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2657–2664, 2011.

[23] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.

[24] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[25] X.-J. Mao, C. Shen, and Y.-B. Yang. Image restoration using convolutional auto-encoders with symmetric skip connections. *arXiv*, 2016.

[26] J. Pan, Z. Hu, Z. Su, H.-Y. Lee, and M.-H. Yang. Soft-segmentation guided object motion deblurring. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[27] J. Pan, Z. Hu, Z. Su, and M.-H. Yang. Deblurring text images via l_0-regularized intensity and gradient prior. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2901–2908, 2014.

[28] J. Pan, D. Sun, H. Pfister, and M.-H. Yang. Blind image deblurring using dark channel prior. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[29] D. Perrone and P. Favaro. Total variation blind deconvolution: The devil is in the details. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2909–2916, 2014.

[30] U. Schmidt, C. Rother, S. Nowozin, J. Jancsary, and S. Roth. Discriminative non-blind deblurring. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 604–611, 2013.

[31] C. J. Schuler, H. Christopher Burger, S. Harmeling, and B. Scholkopf. A machine learning approach for non-blind image deconvolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1067–1074, 2013.

[32] A. Sellent, C. Rother, and S. Roth. Stereo video deblurring. In *European Conference on Computer Vision (ECCV)*, pages 558–575, 2016.

[33] J. Sun, W. Cao, Z. Xu, and J. Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 769–777. IEEE, 2015.

[34] L. Sun, S. Cho, J. Wang, and J. Hays. Edge-based blur kernel estimation using patch priors. In *International Conference on Computational Photography (ICCP)*, pages 1–8, 2013.

[35] Y.-W. Tai, P. Tan, and M. S. Brown. Richardson-lucy deblurring for scenes under a projective motion path. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(8):1603–1618, 2011.

[36] J. Walker, A. Gupta, and M. Hebert. Dense optical flow prediction from a static image. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 2443–2451, 2015.

[37] X. Wang, D. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 539–547, 2015.

[38] X. Wang and A. Gupta. Generative image modeling using style and structure adversarial networks. In *European Conference on Computer Vision (ECCV)*, 2016.

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

[40] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce. Non-uniform deblurring for shaken images. *International Journal of Computer Vision (IJCV)*, 98(2):168–186, 2012.

[41] P. Wieschollek, B. Schölkopf, H. P. A. Lensch, and M. Hirsch. End-to-end learning for image burst deblurring. In *Asian Conference on Computer Vision (ACCV)*, 2016.

[42] L. Xu and J. Jia. Two-phase kernel estimation for robust motion deblurring. In *European Conference on Computer Vision (ECCV)*, pages 157–170, 2010.

[43] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1790–1798, 2014.

[44] L. Xu, S. Zheng, and J. Jia. Unnatural l0 sparse representation for natural image deblurring. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1107–1114, 2013.

[45] H. Zhang and D. Wipf. Non-uniform camera shake removal using a spatially-adaptive sparse penalty. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1556–1564, 2013.

[46] H. Zhang, D. Wipf, and Y. Zhang. Multi-image blind deblurring using a coupled adaptive sparse prior. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[47] S. Zheng, L. Xu, and J. Jia. Forward motion deblurring. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1465–1472, 2013.

[48] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 479–486, 2011.