

Learning a convolutional neural network for non-uniform motion blur removal

Jian Sun, Wenfei Cao, Zongben Xu, Jean Ponce

▶ **To cite this version:**

Jian Sun, Wenfei Cao, Zongben Xu, Jean Ponce. Learning a convolutional neural network for non-uniform motion blur removal. CVPR 2015 - IEEE Conference on Computer Vision and Pattern Recognition 2015, Jun 2015, Boston, United States. 10.1109/CVPR.2015.7298677 . hal-01250478

HAL Id: hal-01250478
https://inria.hal.science/hal-01250478

Submitted on 4 Jan 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Learning a Convolutional Neural Network for Non-uniform Motion Blur Removal

Jian Sun[1], Wenfei Cao[1], Zongben Xu[1], Jean Ponce[2, *]
[1]Xi'an Jiaotong University, [2]École Normale Supérieure / PSL Research University

Abstract

In this paper, we address the problem of estimating and removing non-uniform motion blur from a single blurry image. We propose a deep learning approach to predicting the probabilistic distribution of motion blur at the patch level using a convolutional neural network (CNN). We further extend the candidate set of motion kernels predicted by the CNN using carefully designed image rotations. A Markov random field model is then used to infer a dense non-uniform motion blur field enforcing motion smoothness. Finally, motion blur is removed by a non-uniform deblurring model using patch-level image prior. Experimental evaluations show that our approach can effectively estimate and remove complex non-uniform motion blur that is not handled well by previous approaches.

1. Introduction

Image deblurring [2, 6, 9, 11, 14, 19, 23, 30] aims at recovering sharp image from a blurry image due to camera shake, object motion or out-of-focus. In this paper, we focus on estimating and removing spatially varying motion blur.

Non-uniform deblurring [10, 13, 20] has attracted much attention in recent years. Methods in [7, 8, 26, 31] work on non-uniform blur caused by camera rotations, in-plane translations or forward out-of-plane translations. They are effective for removing non-uniform blur consistent with these motion assumptions. Another category of approaches works on non-uniform motion blur caused by object motion. They estimate blur kernels by analyzing image statistics [17], blur spectrum [1], or with a learning approach using hand-crafted features [3]. Other approaches [13, 29] jointly estimate the sharp image and blur kernels using a sparsity prior. It is still challenging today to remove strongly non-uniform motion blur captured in complex scenes.

In this work, we propose a novel deep learning-based approach to estimating non-uniform motion blur, followed by a patch statistics-based deblurring model adapted to non-uniform motion blur. We estimate the probabilities of motion kernels at the patch level using a convolutional neural network (CNN) [5, 12, 15, 16], then fuse the patch-based estimations into a dense field of motion kernels using a Markov random field (MRF) model. To fully utilize the CNN, we propose to extend the candidate motion kernel set predicted by CNN using an image rotation technique, which significantly boost its performance for motion kernel estimation. Taking advantage of the strong feature learning power of CNNs, we can well predict the challenging non-uniform motion blur that can hardly be well estimated by the state-of-the-art approaches.

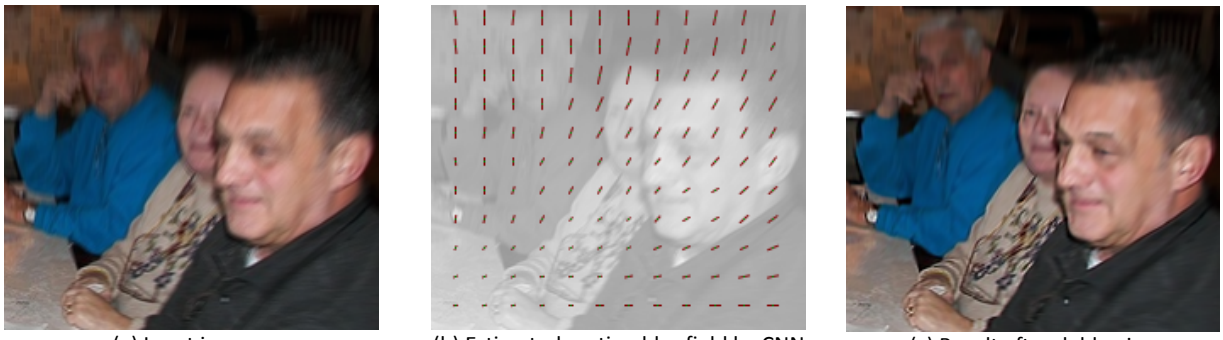
Figure 1 illustrates our approach. Given a blurry image, we first estimate non-uniform motion blur field by a CNN model, then we deconvolve the blurry image. Our approach can effectively estimate the spatially varying motion kernels, which enable us to well remove the motion blur.

1.1. Related Work

Estimating accurate motion blur kernels is essential to non-uniform image deblurring. In [7, 8, 25, 26, 31], non-uniform motion blur is modeled as a global camera motion, which basically estimates an uniform kernel in the camera motion space. Methods in [10, 13, 29] jointly estimate the motion kernels and sharp image. They rely on a sparsity prior to infer the latent sharp image for better motion kernel estimation. Different to them, we estimate motion blur kernels directly using the local patches, which does not require the estimation of camera motion or a latent sharp image.

Another category of approaches [1, 4] estimates spatially varying motion blur based on local image features. The method in [1] estimates motion blur based on blur spectrum analysis of image patch in Fourier transform space. [17] predicts motion blur kernel using natural image statistics. [4] estimates motion blur by analyzing the alpha maps of image edges. [3] learns a regression function to predict motion blur kernel based on some hand-crafted features. Different to them, we estimate motion blur kernels using a convolutional neural network, followed by a carefully designed motion kernel extension method and MRF model to predict

*WILLOW project-team, Département d'Informatique de l'Ecole Normale Supérieure, ENS/Inria/CNRS UMR 8548.



(a) Input image (b) Estimated motion blur field by CNN (c) Result after deblurring

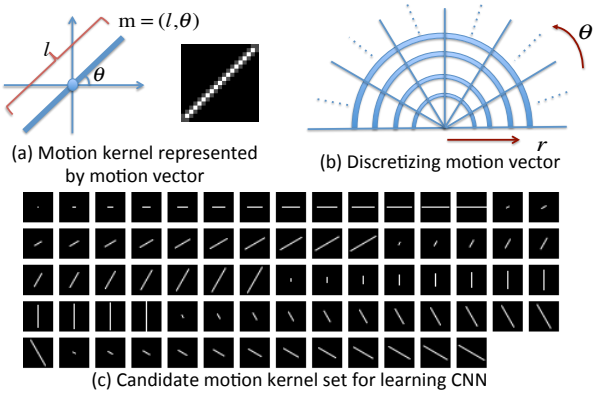
Figure 1. An example illustrating our approach. Given an image with non-uniform motion blur (left). We first estimate the field of non-uniform motion blur kernels by a convolutional neural network (middle), then deconvolve the blurred image (right).

a dense field of motion kernels. Our approach can well estimate complex and strong motion blur, which can hardly be well estimated by the previous approaches.

Recently, there has been some related work on learning-based deblurring approaches. [21] proposes a discriminative deblurring approach using cascade of Gaussian CRF models for uniform blur removal. [22] proposes a neural network approach for learning a denoiser to suppress noises during deconvolution. [28] designs an image deconvolution neural network for non-blind deconvolution. These approaches above focus on designing better learning-based model for uniform blur removal. Our approach works on a more challenging task of non-uniform motion blur estimation and removal. Our CNN-based approach provides an effective method for solving this problem.

2. Learning a CNN for Motion Blur Estimation

We propose to estimate spatially-varying motion blur kernels using a convolutional neural network. The basic idea is that we first predict the probabilities of different motion kernels for each image patch. Then we estimate dense motion blur kernels for the whole image using a Markov random field model enforcing motion smoothness.



(a) Motion kernel represented by motion vector (b) Discretizing motion vector

(c) Candidate motion kernel set for learning CNN

Figure 2. Representation of motion blur kernel by motion vector and generation of motion kernel candidates.

Before giving the details of our approach, let us first introduce our general formulation for non-uniform motion blur. We consider non-uniform image blur caused by object or camera motion. Given a blurry image I, we represent the local motion blur kernel at an image pixel $p \in \Omega$ (Ω is the image region) by a ***motion vector*** $\mathbf{m_p} = (l_p, o_p)$, which characterizes the length and orientation of the motion field in p when the camera shutter is open. As shown in Fig. 2(a), each motion vector determines a motion kernel with non-zero values only along the motion trace. The blurry image can then be represented by $I = k(M) * I_0$, i.e., the convolution of a latent sharp image I_0 with the non-uniform motion blur kernels $k(M)$ determined by the motion field $M = \{\mathbf{m_p}\}_{p \in \Omega}$.

In the following paragraph, we also represent the motion vector $\mathbf{m_p}$ as (u_p, v_p) in Cartesian coordinate system based on the transform:

$$u_p = l_p \cos(o_p), v_p = l_p \sin(o_p). \tag{1}$$

The estimation of spatially-varying motion blur kernels is equivalent to estimating the motion field[1] from a single blurry image. In our approach, we do not make any global parametric assumptions (e.g., homography) on the motion, therefore the motion kernel estimation is challenging, and we only use local image regions for predicting these kernels.

2.1. Patch-level Motion Kernel Estimation by CNN

We now present our approach to predicting motion blur kernels (or equivalently, the motion vector) at the patch level. We decompose the image into overlapping patches of size 30×30. Given a blurry patch Ψ_p centered at pixel p, we aim to predict the probabilistic distribution of motion kernels:

$$P(\mathbf{m} = (l, o)|\Psi_p) \tag{2}$$

for all $l \in S^l$ and $o \in S^o$, S^l and S^o are the sets of motion lengths and orientations respectively. In the followings, we

[1]Note that motions $\mathbf{m} = (l, o)$ and $\mathbf{m}' = (l, o + 180^\circ)$ generate the same motion blur kernel. We therefore only need to estimate the motions with $o \in [0, 180^\circ)$.

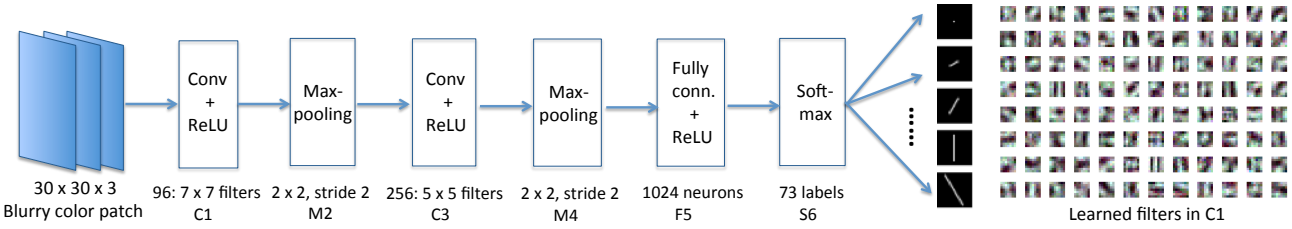


Figure 3. Structure of CNN for motion kernels prediction. It is composed of 6 layers of convolutional layers and fully connected layers. It outputs the probability of each candidate motion kernel using soft-max layer. The right sub-figure shows the learned filters in C1.

call this distribution as ***motion distribution***.

Taking the problem of motion kernel estimation as a learning problem, we utilize convolutional neural network to learn the effective features for predicting motion distributions in Eqn. (2). We generate a set of candidate motion kernels by discretizing the motion space, i.e., the ranges of length and orientation of the motion vectors. In our implementation, we discretize the range of motion length into 13 samples from $l = 1$ to 25 with interval of two, and discretize the range of motion orientation $[0, 180^\circ)$ into 6 samples from 0° to 150° with interval of 30°. Note that when the motion length $l = 1$, all motion vectors correspond to the same blur kernel (i.e., identity kernel) on image grid regardless of the motion orientation. We therefore generate 73 candidate motion vectors (shown in Fig. 2(c)) in different combinations of motion lengths and orientations. We denote the above set of motion kernel candidates as S and the sets of motion lengths and motion orientations as S^l and S^o respectively. Obviously, these candidate motion vectors are far from dense in the continuous motion space. In Section 2.2 we will show how to extend the motion kernels of CNN to predict motion kernels outside the set S.

Given the candidate motion kernel set S, we next construct and learn CNN for predicting the motion distribution over S given a blurry patch. The convolutional neural network is constructed as follows. As shown in Fig. 3, the network has six layers: $C1 - M2 - C3 - M4 - F5 - S6$. $C1$ is a convolutional layer using filters ($7 \times 7 \times 3$) followed by ReLU (i.e., $f(x) = max(x, 0)$ [15]) non-linear transform; $M2$ is a max-pooling layer over 2×2 cells with stride 2; $C3$ is a convolutional layer using 256 filters ($5 \times 5 \times 96$); $M4$ is a max-pooling layer same as $M2$; $F5$ is a fully connected layer with 1024 neurons; $S6$ is a soft-max layer with 73 labels, and each label corresponds to a candidate motion blur kernel in S as shown in Fig. 2(c).

To train the CNN model, we generate a large set of training data $T = \{\Psi_k, \mathbf{m}_k\}_{k=1}^K$, which are composed of blurry patch / motion kernel pairs. We synthetically generate blurry images by convolving clean natural images with the 73 possible motion kernels, then randomly crop $30 \times 30 \times 3$ color patches from the blurry images as the training patches $\{\Psi_k\}_{k=1}^K$, and take the labels of corresponding ground-truth motion kernels as the training labels $\{\mathbf{m}_k\}_{k=1}^K$. We generate training data using 1000 images randomly sampled from PASCAL VOC 2010 database and finally construct a training set of around 1.4 million pairs of blurry patches and their ground-truth motion kernels. Using Caffe [5][2], we train the CNN model in one million iterations by stochastic gradient descent algorithm with batches of 64 patches in each iteration.

Because the final layer of the CNN is a soft-max layer, we can predict the probabilities of motion kernels given an observed blurry patch Ψ as

$$P(\mathbf{m} = (l, o)|\Psi) = \frac{\exp((w_c^{S6})^T \phi_{F5}(\Psi))}{\sum_n \exp((w_n^{S6})^T \phi_{F5}(\Psi))}, \qquad (3)$$

where w_c^{S6} is the vector of weights on neuron connections from F5 layer to the neuron in $S6$ layer representing the motion kernel (l, o), c is the index of (l, o) in S. $\phi_{F5}(\Psi)$ is the output features of $F5$ layer of a blurry patch Ψ, which is a 1024-dimensional feature vector.

In our implementation, we also tried to learn more complex CNN structures (e.g., with one more convolutional layer or more filters in convolutional layers), but the learning speed is significantly slower while the final prediction results are not significantly improved. Figure 3 (right) shows examples of automatically learned filters by our CNN model for motion kernel prediction. These filters reflect diverse local structures in sharp or blurry patch instances.

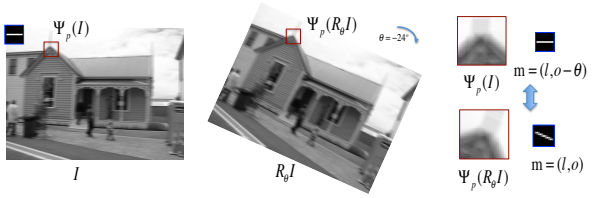


Figure 4. Motion kernel estimation on a rotated patch. I is a blurry image, $R_\theta I$ is the rotated image with θ ($\theta = -24^o$ in this case).

2.2. Extending the Motion Kernel Set of CNN

Our learned CNN model can predict the probabilities of 73 candidate motion kernels in S. Obviously, they are not sufficiently dense in the motion space. We next extend the motion kernel set predicted by the CNN to enable the prediction for motion kernels outside S.

[2]http://caffe.berkeleyvision.org

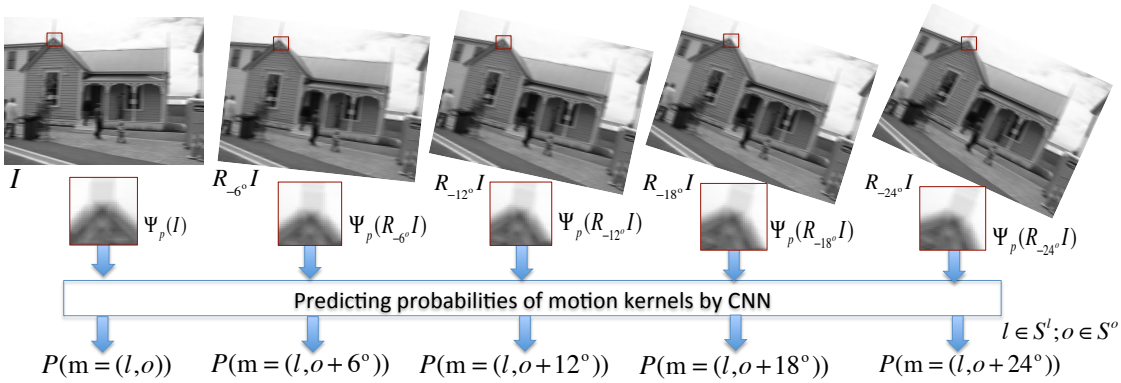


Figure 5. Extension of motion kernel set predicted by CNN using rotated images. For an image I, we generate its rotated images $R_{-6^\circ}I, R_{-12^\circ}I, R_{-18^\circ}I, R_{-24^\circ}I$, then feed each patch and its rotated versions into the CNN to predict motion distributions. By concatenating all the motion distribution estimations, we can estimate the probabilities of more densely sampled motion kernels.

We make the extension based on the following observation. As shown in Fig. 4, given a blurry image I, we rotate it by θ degrees (denoted as $R_\theta I$, R_θ is a rotation operator). For a pair of patches $\Psi_p(I)$ and its rotated version $\Psi_p(R_\theta I)$ cropped from I and $R_\theta I$ centered at pixel p respectively, if we can predict that the motion kernel of $\Psi_p(R_\theta I)$ is $\mathbf{m} = (l, o)$, then we can deduce directly that the motion kernel of corresponding patch $\Psi_p(I)$ in I is $\mathbf{m} = (l, o-\theta)$.

Based on the above observation, we can estimate the probabilities of motion kernels for patch $\Psi_p(I)$ using its rotated patch $\Psi_p(R_\theta I)$. By feeding the rotated patch into CNN, we can estimate the probabilities of motion kernels for the rotated patch: $P(\mathbf{m} = (l, o)|\Psi_p(R_\theta I)), \mathbf{m} \in S$, then we can deduce that the motion distribution of the original patch $\Psi_p(I)$ before rotation is:

$$P(\mathbf{m} = (l, o-\theta)|\Psi_p(I)) = P(\mathbf{m} = (l, o)|\Psi_p(R_\theta I)). \quad (4)$$

Note that motion $\mathbf{m} = (l, o-\theta)$ may not belong to the motion kernel set of CNN (i.e., S).

By carefully designing the image rotations, we can extend the motion kernel set of CNN as follows. Remember that the original CNN can predict probabilities of 73 motion kernels in S with orientations in $S^o = \{0^\circ, 30^\circ, 60^\circ, 90^\circ, 120^\circ, 150^\circ\}$ with interval of 30°. As shown in Fig. 5, given a blurry image I, we first generate its rotated images $R_{-6^\circ}I, R_{-12^\circ}I, R_{-18^\circ}I, R_{-24^\circ}I$ with rotation angles within $[0, 30^\circ)$ and interval of 6°. For each patch $\Psi_p(I)$ centered at pixel p, we extract its rotated versions $\Psi_p(R_\theta I)$ ($\theta \in \{-6^\circ, -12^\circ, -18^\circ, -24^\circ\}$) from the rotated images. By feeding these patches into CNN, we can predict the probabilities of motion kernels for patch $\Psi_p(I)$ using each patch based on Eqn. (4):

$$\begin{aligned}
P(\mathbf{m} = (l, o)|\Psi_p(I)) &= P(\mathbf{m} = (l, o)|\Psi_p(I)),\\
P(\mathbf{m} = (l, o+6^\circ)|\Psi_p(I)) &= P(\mathbf{m} = (l, o)|\Psi_p(R_{-6^\circ}I)),\\
P(\mathbf{m} = (l, o+12^\circ)|\Psi_p(I)) &= P(\mathbf{m} = (l, o)|\Psi_p(R_{-12^\circ}I)),\\
P(\mathbf{m} = (l, o+18^\circ)|\Psi_p(I)) &= P(\mathbf{m} = (l, o)|\Psi_p(R_{-18^\circ}I)),\\
P(\mathbf{m} = (l, o+24^\circ)|\Psi_p(I)) &= P(\mathbf{m} = (l, o)|\Psi_p(R_{-24^\circ}I)),
\end{aligned}$$

where $l \in S^l, o \in S^o$. By concatenating all the above estimations from one patch and its rotated versions, we can therefore predict the motion distribution in an extended motion kernel set of CNN: $P(\mathbf{m} = (l, o)|\Psi_p(I)), o \in S^o_{ext} = \{0^\circ, 6^\circ, 12^\circ, \cdots, 174^\circ\}, l \in S^l$. After motion kernels extension, we can totally predict probabilities of 361 candidate motion kernels[3] for an image patch by CNN, which is almost 5 times of the number of candidate motion kernels in S predicted by CNN. Note that this process does not require the CNN retraining, but just feed this image and its rotated versions to our learned CNN.

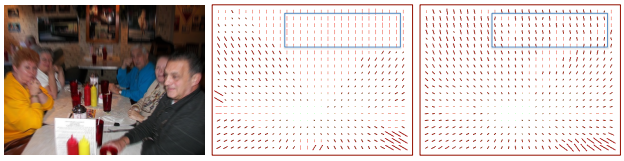


Figure 6. Effect of CNN motion kernel set extension.

Figure 6 shows an example of motion kernel estimation without and with CNN motion kernel set extension. In this example, we synthetically generate the motion blur using a camera motion. As shown in Fig. 6(b), the estimated motion kernels suffer from blocky artifacts in the blue rectangle because all pixels in it are predicted to have the same orientation due to the large quantization interval of motion orientations. By extending the motion kernel set of CNN, we can predict more accurate motion kernels shown in Fig. 6(c). The mean squared error (MSE) w.r.t. ground-truth motion kernels is reduced from 10.3 to 8.4.

3. Dense Motion Field Estimation by MRF

The CNN predicts distribution of motion kernels in an image at the patch level. We now discuss how to fuse these

[3]There are totally 390 possible kernels by combining the motion lengths in S^l ($|S^l| = 13$) and motion orientations in S^o_{ext} ($|S^o_{ext}| = 30$). But the motion kernels $\{\mathbf{m} = (l, o)\}_{o \in S^o_{ext}}$ when $l = 1$ are all the same, we only retain one of them.

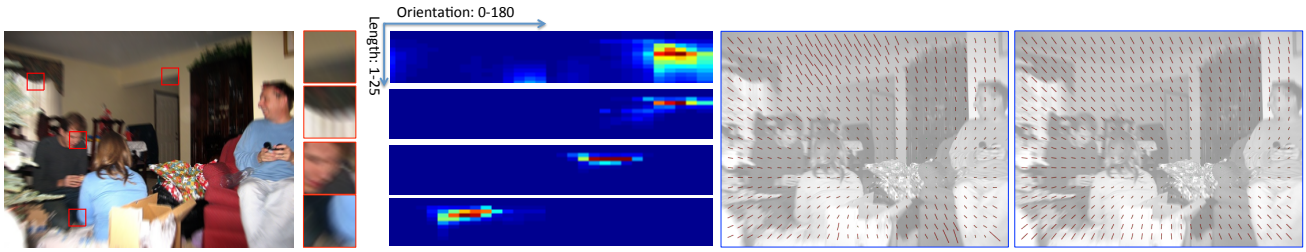


Figure 7. Examples of motion kernel probabilities. The left of (b) show four blurry patches cropped from (a). Each color map on the right of (b) shows the probabilities of motion kernels in different motion lengths and orientations estimated for each blurry patch by CNN. Note that the high probability regions are local in each map. (c) shows our final motion kernel estimation.

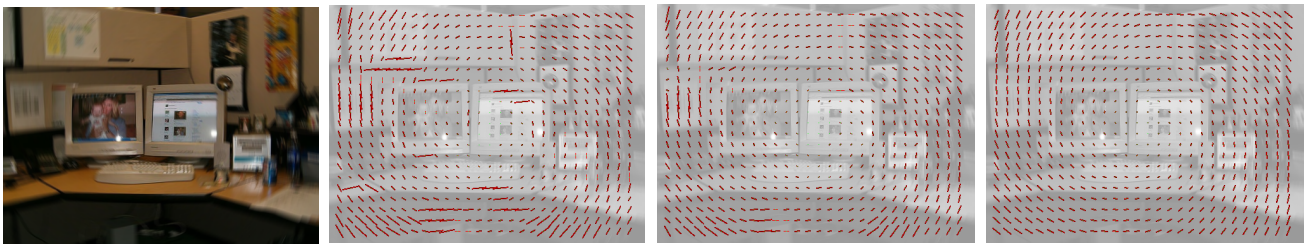


Figure 8. Example of non-uniform motion kernel estimation. (b) Estimation using the unary term of Eqn.(6), i.e., choosing the motion kernel with highest confidence for each pixel. (c) Estimation using the full model of Eqn.(6) with motion smoothness constraint. (d) Ground-truth motion blur. MSE_motion is an accuracy measurement of motion blur defined in Section 5.

patch-level motion kernel estimations into a dense field of motion kernels for the image.

Given an image I, we sample $30 \times 30 \times 3$ overlapping color patches with a spatial interval of 6 pixels over image I. Each patch $\Psi_p(I)$ produces motion kernel probabilities: $P(\mathbf{m} = (l, o)|\Psi_p(I))$ $(l \in S^l, o \in S^o_{ext})$ by applying the CNN. Figure 7(b) shows examples of motion distribution maps for four blurry patches, and each of them is sparse and composed of one local high probability region. We assume that the pixels in patch $\Psi_p(I)$ share the same motion distribution. Then each pixel has multiple estimates for each motion kernel probability from all patches containing it. For a pixel p, we perform weighted average over the multiple estimates of each motion kernel probability, and define the confidence of motion kernel $\mathbf{m} = (l, o)$ at pixel p as

$$C(\mathbf{m}_p = (l, o)) = \frac{1}{Z} \sum_{q:p\in\Psi_q} G_\sigma(||x_p - x_q||^2) P(\mathbf{m} = (l, o)|\Psi_q), \quad (5)$$

for all $l \in S^l, o \in S^o_{ext}$. x_p is the coordinate of pixel p. As a Gaussian function, $G_\sigma(||x_p - x_q||^2)$ imposes higher weights on the patch Ψ_q in the summation if its center pixel q is closer to pixel p. σ is set to 10 in our implementation. This means that we trust more the motion prediction from the patch containing pixel p closer to its patch center. $Z = \sum_{q:p\in\Psi_q} G_\sigma(||x_p - x_q||)$ is a normalization constant.

We further assume that the motion kernels are spatially smooth. This is reasonable because the moving objects or camera are moving smoothly during capturing image, and nearby pixels should have similar motions. Then we estimate the dense motion field $M = \{\mathbf{m}_p = (l_p, o_p)\}_{p\in\Omega}$ over image I by optimizing the following MRF model:

$$\min_M \sum_{p\in\Omega} [-C(\mathbf{m}_p = (l_p, o_p)) + \sum_{q\in N(p)} \lambda[(u_p - u_q)^2 + (v_p - v_q)^2], \quad (6)$$

where $l_p \in S^l, o_p \in S^o_{ext}$, (u_p, v_p) and (u_q, v_q) are motion vectors $\mathbf{m}_p$ and $\mathbf{m}_q$ in Cartesian coordinates that are related to (l_p, o_p) and (l_q, o_q) by Eqn. (1). $N(p)$ is the neighborhood of p. By minimizing the energy function, the first term encourages to choose the motion kernel for each pixel with higher confidence estimated by CNN, and the second term enforces the smoothness of nearby motion kernels.

For each pixel, there are 361 motion kernel candidates, it is inefficient to optimize the MRF problem with such a large number of candidate labels. We therefore generate candidate motion kernels for each pixel by selecting the top 20 motion kernels with highest confidence values, together with 30 sampled motion candidates from the remaining candidates to make the motion kernel candidate set for each pixel both prominent and diverse. Since the candidate label sets are spatially varying, we cannot use the off-the-shelf graph cut toolbox [24], we therefore optimize the energy by max-product belief propagation algorithm. Predicting dense motion blur for an image of size 300×400 takes

around 80 seconds using CPU including computing patch-level motion distributions by CNN.

Figure 8 shows an example of motion blur estimation. As shown in Fig. 8(b, c), the full MRF model can effectively remove the noisy estimates in Fig. 8(b) using smoothness term, and quantitative results are significantly improved.

4. Non-Uniform Motion Deblurring

With the dense non-uniform motion kernels estimated by CNN, we now deconvolve the blurry image to estimate the sharp image. It is challenging to deconvolve the image blurred by non-uniform motion blur. We adapt the uniform deconvolution approach in [32] to the non-uniform deconvolution problem. The non-uniform deconvolution is modeled as optimizing:

$$\min_I \frac{\lambda}{2}||k(M) * I - O||_2^2 - \sum_{i\in\Omega} \log(P(R_i I)) \quad (7)$$

where O is the observed blurry image, R_i is an operator to extract the patch located at i from an image. $P(\cdot)$ is the prior distribution of natural image patches, which is modeled as a Gaussian mixture model learned from natural image patches [32].

Different to uniform deblur in [32], the first term in Eqn.(7) is modeled for non-uniform motion blur. We optimize the above problem by half-quadratic splitting algorithm, i.e., optimizing: $\min_{I,\{z_i\}} \frac{\lambda}{2}||k(M) * I - O||_2^2 + \sum_{i\in\Omega}(\frac{\beta}{2}||R_i I - z_i||_2^2 - \log(P(z_i)))$, where auxiliary variables $\{z_i\}$ are introduced. We iteratively optimize I and $\{z_i\}$ by increasing β. In the iterations, we need to optimize the following two sub-problems. (1) By fixing $\{z_i\}$, we optimize sharp image: $\min_I \frac{\lambda}{2}||k(M) * I - O||_2^2 + \sum_{i\in\Omega}(\frac{\beta}{2}||R_i I - z_i||_2^2)$. (2) By fixing I, we optimize $\{z_i\}$: $\min_{z_i} \frac{\beta}{2}||R_i I - z_i||_2^2 - \log(P(z_i)), i \in \Omega$.

For sub-problem (1), the blur kernels $k(M)$ are non-uniform and determined by spatially varying motion vectors M. By re-writing the non-uniform convolution as matrix-vector multiplication (i.e., $k(M) * I = K_M I$), we optimize sharp image by solving the linear equations deduced by setting the gradients of cost in sub-problem (1) to zeros:

$$[\lambda K_M^T K_M + \beta \sum_{i\in\Omega}(R_i^T R_i)]I = \lambda K_M^T O + \beta(\sum_{i\in\Omega} R_i^T z_i). \quad (8)$$

We solve these linear equations using a conjugate gradient algorithm. In the implementation, all the involved matrix-vector multiplications can be efficiently implemented by convolutions or local operations around each pixel. $R_i^T z$ is an operation to put the patch z back to the region where it was extracted. The sub-problem (2) can be optimized following [32]. In implementation, we set the patch size to 8×8, $\lambda = 2 \times 10^5$, and β is increased from 50 to 3200 with a ratio of 2 in 7 iterations of alternative optimizations.

Table 1. Comparison of motion kernel estimation on 15 test images with synthetic motion blur. "BlurSpect" is based on the approach in [1]. "SLayerRegr" is the extension of approach in [3].

Methods	DL_MRF	DL_noMRF	DL_noLE	BlurSpect	SLayerRegr
MSE_motion	**7.83**	16.35	9.01	44.56	65.10
PSNR_motion	**44.55**	37.14	43.17	26.58	22.70

5. Experiments

To evaluate the quantitative accuracy of our approach for non-uniform motion kernel estimation, we generate 15 synthetic blurred images with ground-truth non-uniform motion kernels caused by camera motions (rotation and translation). The examples shown in Figs. 5-8 are from this synthetic image set. Given the estimated motion blur kernels $M = \{u_p, v_p\}_{p\in\Omega}$ and ground-truth motion blur kernels $M^{gt} = \{u_p^{gt}, v_p^{gt}\}_{p\in\Omega}$ in the Cartesian coordinate system, we measure the accuracy of the estimated motion kernel by the mean-squared-error (MSE_motion): $\mathbf{MSE_motion}(M, M^{gt}) = \frac{1}{2|\Omega|}\sum_{p\in\Omega}[(u_p - u_p^{gt})^2 + (v_p - v_p^{gt})^2]$ and peak signal-to-noise ratio (PSNR_motion): $\mathbf{PSNR_motion}(M, M^{gt}) = -10\log\frac{\mathbf{MSE_motion}(M,M^{gt})}{d_{max}^2}$, $d_{max} = 25$ is the maximum motion length.

Figure 9 presents four examples with strongly non-uniform motion blur captured for scenes with complex depth layers. The first three examples are real-captured blurry images, and the final example is a synthetic blurry image. All these examples show that our CNN-based approach can effectively predict the spatially varying motion kernels.

In Table 1, we evaluate and compare our approach to the other approaches for non-uniform motion kernel estimation. "DL_noMRF" is our approach using only the unary term in Eqn. (6). "DL_noLE" is our MRF-based approach without using the motion kernel set extension. "DL_MRF" is our full estimation approach. "BlurSpect" is the approach proposed in [1]. It was originally designed for estimating horizontal or vertical motion blur, and we extend it to estimate motion kernels with orientations in S_{ext}^o by the technique in Section 2.2. "SLayerRegr" is an extended version of approach in [3]. The original approach learns a logistic regressor to estimate discrete motion kernels in horizontal direction using hand-crafted features. To predict motion kernels in other directions, we implement [3] using the same features and learn SVMs for predicting 73 motion kernels in S, then extend motion kernel set by the method in Section 2.2. "SLayerRegr" can be seen as a learning machine with a single layer of hand-crafted features. As shown in Table 1, both "BlurSpect" and "SLayerRegr" perform poorly on estimating the challenging non-uniform motion blur with diverse motion lengths and orientations. Our approach can effectively estimate the motion kernels with average MSE_motion 7.83 and PSNR_motion 44.55. More-

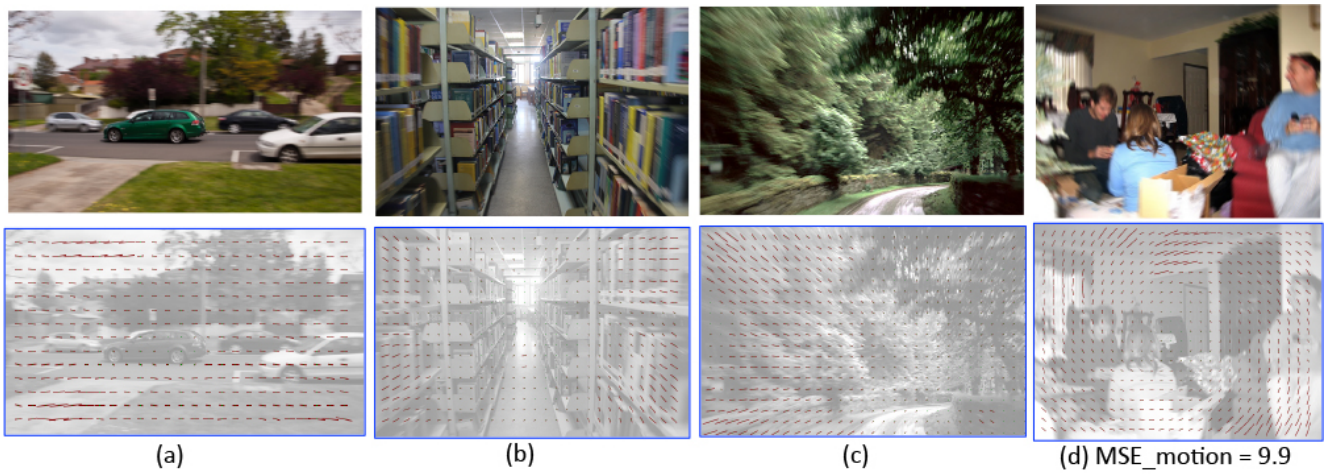


Figure 9. Examples on motion kernel estimation. The first three columns are real blurry images, the last column shows a synthetic picture with camera rotation (MSE_motion = 9.9).

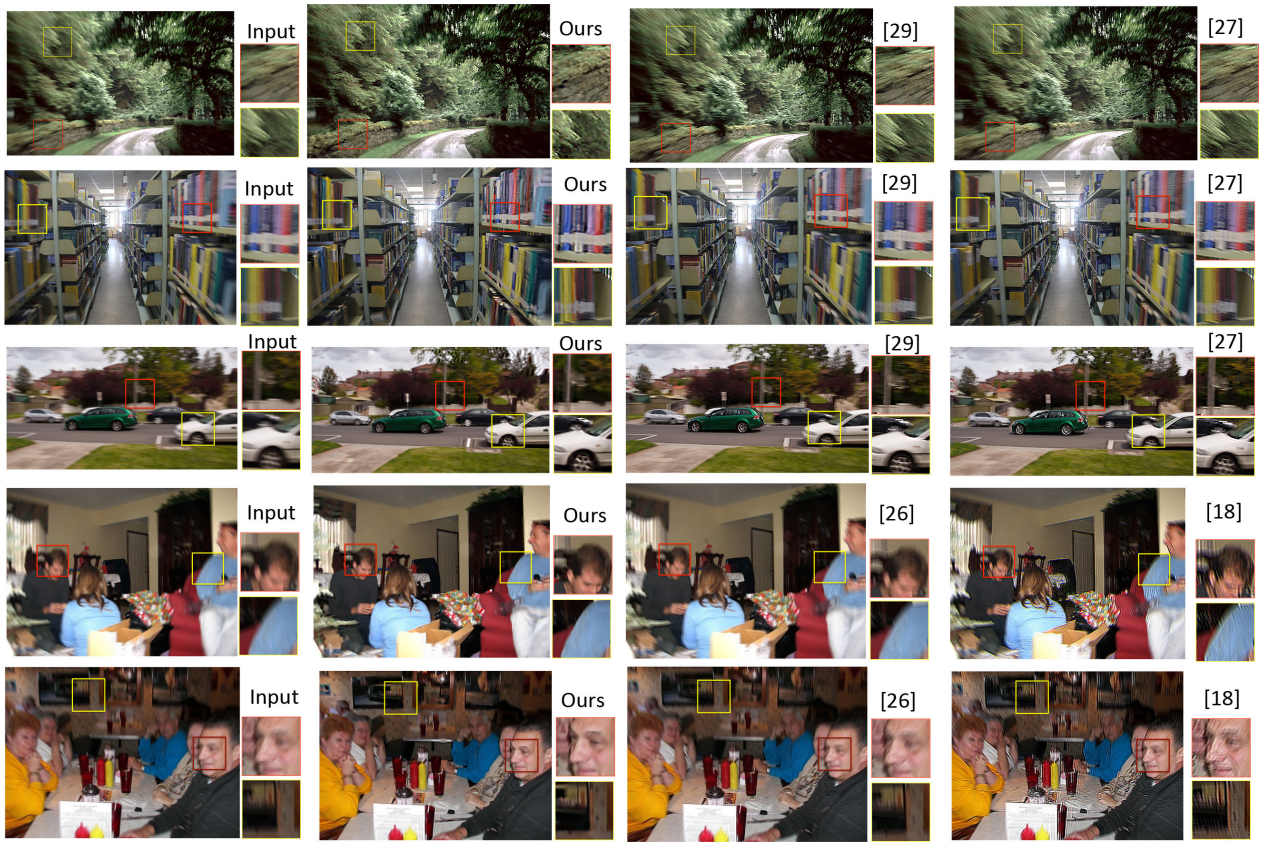


Figure 10. Examples of non-uniform motion deblurring. The first and second columns show the blurry images and our results. The third and fourth columns show the results of methods in [18, 26, 27, 29] using their source codes. These examples are challenging because the motion blur kernels are strongly non-uniform and the scenes are complex. Our estimated motion blur fields are shown in Figs. 9, 12 .

over, the motion kernel set extension and motion smoothness constraint significantly improve the accuracy of motion kernel estimation.

Figure 10 compares deblurring results of our approach, non-uniform deblurring approaches [26, 29] and uniform deblurring approaches [18, 27], for which the source codes are available. Except for ours, none of these methods handles the non-uniform blur in a satisfying manner for these examples. Our approach estimates more accurate motion blur kernels, which enables us to produce better final de-

Table 2. Accuracies of motion kernel estimation and blur removal on 15 test images with synthetic motion blur. "BlurSpect" is based on the approach in [1]. "SLayerRegr" is the extension of approach in [3]. "MSE_ker" is an error for non-uniform blur kernel estimation using the average MSE of blur kernels across image pixels. "PSNR_deblur" is the PSNR of the final deblurred results. The number in each table cell is the mean value over the image set.

	DL_MRF	DL_noMRF	DL_noLE	BlurSpect	SLayerRegr	TwoPhase [27]	MargLike [18]	NonUnif [26]	UnNatural [29]
MSE_ker	**0.024**	0.029	0.041	0.250	0.127	0.108	0.119	0.193	0.165
PSNR_deblur	**24.81**	24.66	24.61	21.72	19.04	21.26	18.49	20.65	21.33

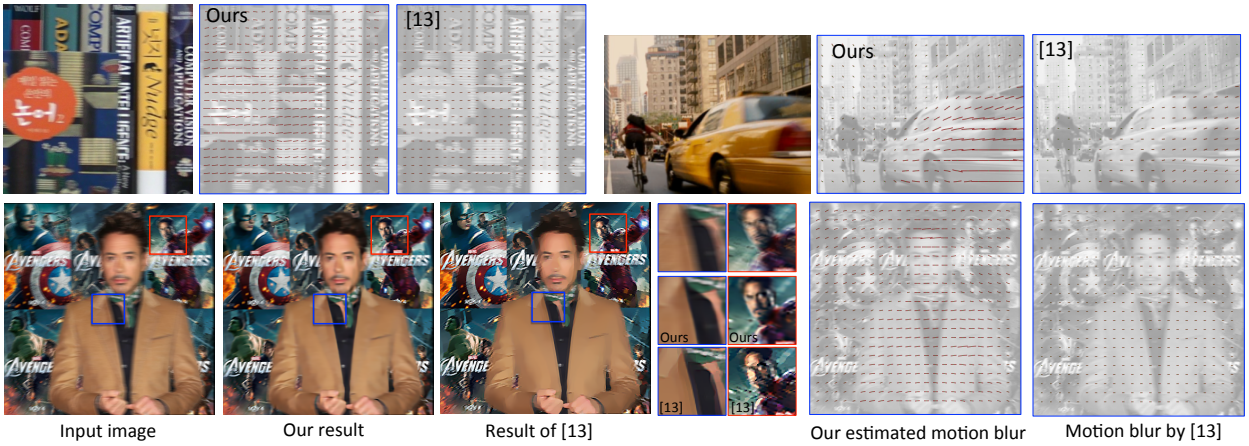


Figure 11. Comparison to [13]. Our CNN can better predict the different motion layers. The deblurring result of [13] is over-sharpened and image details are removed, while our result is visually more natural.

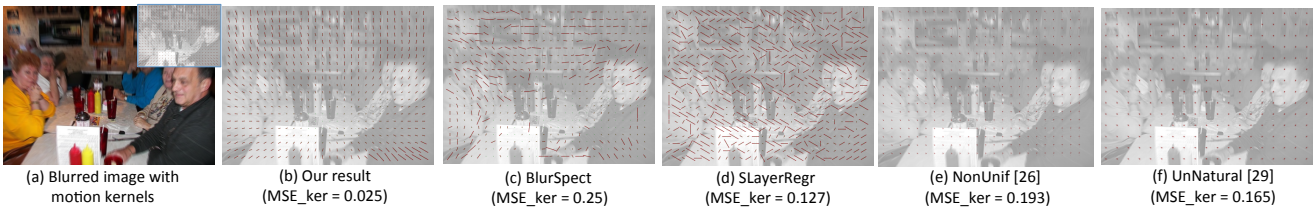


Figure 12. Comparison of motion kernel estimation. "BlurSpect" and "SLayerRegr" are based on the methods in [1] and [3] respectively.

blurring results. The method in [13] is an effective approach for motion deblurring. Because its source code is not available, we directly compare it on examples of [13] in Fig. 11. Our approach can better "recognize" the complex motions. The deblurring result of [13] is commonly over-sharpened, but our deblurring result is visually more natural.

In Table 2, we qualitatively compare our method to the state-of-the-art non-blind debluring approaches for both the motion blur kernel estimation and the final deblurred results. We define an error of "MSE_ker" for non-uniform motion blur estimation using average MSE of blur kernels across pixels in an image, and the MSE of each pixel is defined by the mean per-element squared difference between the estimated and ground-truth kernels after aligning kernels by centers. Contrary to the "MSE_motion" that measures the kernel error in the linear motion space, this error term directly measures the kernel differences in the spatial domain. We also evaluate the deblurring result by the PSNR of the deblurred image (denoted as "PSNR_deblur") w.r.t. the ground-truth clean image. All the values in Table 2 are the mean values over the image set. We can not qualitatively compare to the approach in [13] because the source codes are not available. These results clearly show that our approach can produce significantly better results both in the motion blur kernel estimation and the motion blur removal than the compared state-of-the-art approaches.

Figure 12 shows an example of motion blur estimation by different non-uniform blur estimation approaches. Our approach can produce significantly better non-uniform motion blur field than the compared approaches.

6. Conclusion

In this paper, we have proposed a novel CNN-based non-uniform motion deblurring approach. We learn an effective CNN for estimating motion kernels from local patches. Using an MRF model, we are able to well predict the non-uniform motion blur field. This leads to state-of-the-art motion deblurring results. In the future, we are interested in designing a CNN for estimating the general non-uniform blur kernels. We are also interested in designing an CNN system that can estimate and remove general non-uniform blurs in a single framework.

Acknowledgement

This work was partially supported by NSFC projects (61472313, 11131006), the 973 program (2013CB329404) and NCET-12-0442. Ponce was supported in part by the Institut Universitaire de France and European Research Council (VideoWorld project).

References

[1] A. Chakrabarti, T. Zickler, and W. Freeman. Analyzing spatially-varying blur. In *CVPR*, pages 2512–2519, 2010.

[2] S. Cho, Y. Matsushita, and S. Lee. Removing non-uniform motion blur from images. In *ICCV*, pages 1–8, 2007.

[3] F. Couzinié-Devy, J. Sun, K. Alahari, and J. Ponce. Learning to estimate and remove non-uniform image blur. In *CVPR*, 2013.

[4] S. Dai and Y. Wu. Motion from blur. In *CVPR*, 2008.

[5] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv:1310.1531*, 2013.

[6] R. Fergus, B. Singh, A. Hertzmann, S. Roweis, and W. Freeman. Removing camera shake from a single photograph. *ACM Transactions on Graphics*, 25(3):787–794, 2006.

[7] A. Gupta, N. Joshi, C. Lawrence Zitnick, M. Cohen, and B. Curless. Single image deblurring using motion density functions. In *ECCV*, 2010.

[8] M. Hirsch, C. Schuler, S. Harmeling, and B. Scholkopf. Fast removal of non-uniform camera shake. In *ICCV*, 2011.

[9] Z. Hu and M. Yang. Good regions to deblur. In *ECCV*, 2012.

[10] H. Ji and K. Wang. A two-stage approach to blind spatially-varying motion deblurring. In *CVPR*, 2012.

[11] N. Joshi, R. Szeliski, and D. Kriegman. Psf estimation using sharp edge prediction. In *CVPR*, 2008.

[12] L. Kang, P. Ye, Y. Li, and D. Doermann. Convolutional neural networks for no-reference image quality assessment. In *CVPR*, 2014.

[13] T. H. Kim and K. M. Lee. Segmentation-free dynamic scene deblurring. In *CVPR*, 2014.

[14] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. In *NIPS*, 2009.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 1998.

[17] A. Levin. Blind motion deblurring using image statistics. In *NIPS*, 2007.

[18] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Efficient marginal likelihood optimization in blind deconvolution. In *CVPR*, 2011.

[19] T. Michaeli and M. Irani. Blind deblurring using internal patch recurrence. In *ECCV*, 2014.

[20] C. Paramanand and A. N. Rajagopalan. Non-uniform motion deblurring for bilayer scenes. In *CVPR*, 2013.

[21] U. Schmidt, C. Rother, S. Nowozin, J. Jancsary, and S. Roth. Discriminative non-blind deblurring. In *CVPR*, 2013.

[22] C. J. Schuler, H. C. Burger, S. Harmeling, and B. Scholkopf. A machine learning approach for non-blind image deconvolution. In *CVPR*, 2013.

[23] L. Sun, S. Cho, J. Wang, and J. Hays. Good image priors for non-blind deconvolution. In *ECCV*. 2014.

[24] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, Kolmogorov, and et.al. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE T. PAMI*, 30(6):1068–1080, 2008.

[25] Y. Tai, P. Tan, and M. Brown. Richardson-lucy deblurring for scenes under a projective motion path. *IEEE T. PAMI*, 33(8):1603–1618, 2011.

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

[27] L. Xu and J. Jia. Two-phase kernel estimation for robust motion deblurring. In *ECCV*, 2010.

[28] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In *NIPS*, 2014.

[29] L. Xu, S. Zheng, and J. Jia. Unnatural l0 sparse representation for natural image deblurring. In *CVPR*, 2013.

[30] L. Yuan, J. Sun, L. Quan, and H. Shum. Image deblurring with blurred/noisy image pairs. *ACM Transactions on Graphics*, 26(3):1, 2007.

[31] S. Zheng, L. Xu, and J. Jia. Forward motion deblurring. In *ICCV*, 2013.

[32] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *ICCV*, 2011.