Neural Embedding of an Iterative Deconvolution Algorithm for Motion Blur Estimation and Removal

Thomas Eboli, Jian Sun, Jean Ponce

Neural Embedding of an Iterative Deconvolution Algorithm for Motion Blur Estimation and Removal

Thomas Eboli* **Jian Sun†** **Jean Ponce***

Abstract

This paper introduces a new learning-based approach to motion blur removal. A local linear motion model is first estimated at each pixel using a convolutional neural network (CNN) in a regression setting. These estimates are then used to drive an algorithm that casts non-blind, non-uniform image deblurring as a least-squares problem regularized by natural image priors in the form of sparsity constraints. This problem is solved by combining the alternative direction method of multipliers with an iterative residual compensation algorithm, with a finite number of iterations embedded into a second CNN whose trainable parameters are deconvolution filters. The second network outputs the sharp image, and the two CNNs can be trained together in an end-to-end manner. Our experiments demonstrate that the proposed method is significantly faster than existing ones, and provides competitive results with the state of the art on several synthetic and real datasets.

1 Introduction

The goal of blind image deconvolution is to recover a sharp image from a picture degraded by camera shake, object motion or defocus given only the blurry picture of the scene as input. It has wide applications in computational photography, astronomy, microscopy and medical image enhancement for example. Classical methods such as Wiener [32] and Richardson-Lucy [24] filtering apply inverse filters to the corrupted image but they are known to generate artifacts in the restored images. Variational methods rely on a set of constraints to recover sharp pictures by solving an inverse problem with, for instance, sparsity priors [5, 30, 22, 25, 19, 33]. These priors are handcrafted and may fail to capture all the constraints that lead to visually appealing images. Recently, convolutional neural networks (CNNs) have also been applied to blind deconvolution problems [31, 3, 20, 21, 16], case as supervised regression. These methods essentially ignore the underlying "physics" of the image corruption process, but exploit various tricks such as skip connections [11], batch normalization [13] or generative adversarial networks [9] to directly output a sharp image.

The aim of this paper is to combine the classical image processing approach with the modern neural one, embedding an iterative deconvolution algorithm in a CNN, thus obtaining a relatively shallow architecture that can be trained end to end, and yields competitive accuracy and significantly improved speed. We focus on motion blur removal and decompose this task into motion estimation and non-blind deconvolution. Like others, we assume that each image patch is corrupted by a local linear motion [14]. In [26], the authors design a CNN to predict a linear motion kernel per patch in the blurry image for every overlapping patch and they enforce smoothness of the global blur kernel by applying a Markov Random Field algorithm. In [8], the authors propose a fully convolutional network (FCN) to predict the non-uniform blur kernel directly for the whole image. The motion estimates can then be fed to a non-blind deblurring algorithm (e.g., [36]), but this pipeline is not trainable end to end, and it is computationally expensive.

*Inria, École normale supérieure, CNRS, PSL Research University, 75005 Paris, France.
†School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, 710049, China.

Preprint. Work in progress.

Algorithm 2.1: Iterative residual compensation algorithm for updating x.

data: $z_i,\ d_i,\ \epsilon,\ T_{max},\ x_0$ (current estimate of x).
result: x.
initialization: $t \leftarrow 0$
while $\sum_{i=0}^{n} ||r_t^i||_2^2 > \epsilon$ *and* $t < T_{max}$ **do**

1. for $i = 0, \ldots, n$ do $r_{t+1}^i \leftarrow z_i - k_i \star x$;
2. $x_{t+1} \leftarrow x_t + \sum_{i=0}^{n} d_i \star r_{t+1}^i$;
3. $t \leftarrow t + 1$.

return *Estimated clean image* x_{t+1}.

The main contributions of this paper can be summarized as follows. **(1)** We propose an iterative algorithm that casts non-blind, non-uniform image deblurring as a least-squares problem regularized by natural image priors in the form of sparsity constraints. This problem is solved by combining the alternative direction method of multipliers with an iterative residual compensation algorithm with theoretical convergence guarantees. A finite number of iterations of this algorithm are embedded in a CNN whose trainable parameters are deconvolution filters (see [35, 10] for similar ideas in the context of *uniform* image deblurring or sparse coding). This approach combines high accuracy with faster convergence than other solvers for the same task. **(2)** We propose a CNN-based regressor for local linear motion estimation and demonstrate that it is both more accurate and faster than [8, 26]. By stacking the two networks, we obtain a simple and relatively shallow architecture for blind motion blur removal that can be trained end to end, and is competitive in terms of accuracy with the state of the art [1, 20] on the dataset of [1], as well as significantly faster.

2 Iterative Non-Blind Deconvolution

2.1 Uniform Blur

Let us first consider the case of uniform blur. We use the classical model $y = k \star x + \epsilon$, where x is the (unknown) sharp image, k is the (known) uniform blur kernel, y is the (observed) blurry image, and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is an additive noise term. Regularized non-blind deconvolution amounts to solving $\min_x ||k \star x - y||_2^2 + \lambda R(x)$, where R is some regularizer associated with natural images priors. A common prior is to impose sparsity constraints on filtered versions of x. More specifically, we consider here n filters k_i, and solve the problem:

$$\min_x \tfrac{1}{2}||k \star x - y||_2^2 + \lambda \textstyle\sum_{i=1}^{n} ||k_i \star x||_l \qquad \text{with} \quad 0 \leq l \leq 1. \tag{1}$$

We use the alternating direction method of multipliers (ADMM)[2] algorithm to solve Eq. (1): We introduce n auxiliary variables $z_1, \ldots, z_n$ and expand Eq. (1) as follows:

$$\min_{x,z} \tfrac{1}{2}||k \star x - y||_2^2 + \lambda \textstyle\sum_{i=1}^{n} ||z_i||_l \qquad \text{s.t} \quad z_i = k_i \star x, \quad \forall i \in [1, 2, \ldots, n]\,. \tag{2}$$

The corresponding augmented Lagrangian is:

$$L(x, z, \alpha) = \tfrac{1}{2}||k \star x - y||_2^2 + \lambda \textstyle\sum_{i=1}^{n} ||z_i||_l - \sum_{i=1}^{n} \langle \alpha_i,\ k_i \star x - z_i \rangle + \tfrac{\rho}{2} \sum_{i=1}^{n} ||k_i \star x - z_i||_2^2, \tag{3}$$

where the coefficients α_i are Lagrange multipliers and ρ is a penalty parameter. We solve Eq. (2) by alternatively updating the variables z_i, x, and $\beta_i = \alpha_i / \rho$ (scaled Lagrange multipliers).

• z_i **update :** Each one of the n independent optimization problems

$$\min_{z_i} \lambda ||z_i||_l + \tfrac{\rho}{2}||k_i \star x + \beta_i - z_i||_2^2 \tag{4}$$

can be solved in closed-form when $l = 0$ or $l = 1$ using hard- or soft-thresholding [6].

• x **update :** The related optimization problem is, with $z_0 = \frac{1}{\sqrt{\rho}} y$ and $k_0 = \frac{1}{\sqrt{\rho}} k$:

$$\min_x \textstyle\sum_{i=0}^{n} ||k_i \star x + \beta_i - z_i||_2^2. \tag{5}$$

This sum of least-squares problems is itself a linear least-squares problem which can be solved by conjugate gradient descent for example, but its special form affords the much more efficient *iterated residual compensation* (or *IRC* method presented now). Imagine for the moment that we have a single least-squares problem of the form $\min_x ||k \star x - y||^2$. We propose to repeatedly apply an inverse deconvolution filter d (obtained by Wiener filtering for example, more details on this below) to blurry residuals until convergence. We initialize the algorithm with $x_0 = d \star y$ and repeat the following two steps T times.

1. We compute the blurry residual: $r_{t+1} = y - k \star x_t$;
2. We remove the *deblurred* residual from x_t: $x_{t+1} = x_t + d \star r_{t+1}$.

Proposition : *Let δ denote the unit impulse, and H denote the square matrix version of the filter $\delta - d \star k$. When the spectral radius ρ of H is smaller than 1, the sequence x_t converges linearly to x with convergence rate ρ.*

Proof. By denoting by $\bar{x}$ the vectorized version of x, we have, by definition, for $t \geq 1$,

$$x_{t+1} = x_t + d \star k \star (x - x_t) = h \star x_t + d \star k \star x. \tag{6}$$

Thus,

$$x_{t+1} - x = h \star (x_t - x) \Longleftrightarrow \bar{x}_{t+1} - \bar{x} = H(\bar{x}_t - \bar{x}), \tag{7}$$

and

$$||\bar{x}_{t+1} - \bar{x}||_F^2 = ||H(\bar{x}_t - \bar{x})||_F^2 \leq \rho^2 ||\bar{x} - \bar{x}||_F^2. \tag{8}$$

Finally,

$$||x_{t+1} - x||_F^2 \leq \rho^2 ||x_t - x||_F^2. \tag{9}$$

□

Empirically, as also shown there, this algorithm limits the artifacts obtained by common deconvolution methods such as Wiener filtering, with accuracy comparable to FFT and conjugate-gradient methods for a fraction of the computational cost of the latter. It is also easily adapted to the sum of least-squares problems of Eq (5), resulting in Algorithm 2.1. We use finite-support deconvolution filters d_i instead of a classical FFT-based approach because they have proven fast and accurate. They are also easily adapted to the non-uniform case, and can be efficiently implemented by local operations. One detail not discussed yet is how we explicitly compute the inverse filters d_i. Let us assume that we can compute local deconvolution kernels $d_0, d_1, \cdots, d_n$ in $\mathbb{R}^{w_f \times w_f}$ for the k_i in $\mathbb{R}^{w_k \times w_k}$ such that

$$\sum_{i=0}^{n} d_i \star k_i = \delta. \tag{10}$$

We have in this case

$$\begin{aligned} ||x - \sum_{i=0}^{n} d_i \star z_i||_F^2 &= ||\sum_{i=0}^{n} d_i \star (k_i \star x - z_i)||_F^2 \\ &\leq \sum_{i=0}^{n} ||d_i||_F^2 ||z_i - k_i \star x||_F^2. \end{aligned} \tag{11}$$

As $||z_i - k_i \star x||_F^2$ is supposed to be close to 0, it is therefore reasonable to take $x = \sum_{i=0}^{n} d_i \star z_i$ as an *approximate* solution to (5). This approximation will of course need to be refined iteratively.

The local deconvolution kernels $d_0, \cdots, d_n$ can be computed by solving the linear equations:

$$\sum_{i=0}^{n} K_i^T \bar{d}_i = \left[K_0^T, \cdots, K_n^T\right] \begin{bmatrix} \bar{d}_0 \\ \vdots \\ \bar{d}_n \end{bmatrix} = \bar{\delta}, \tag{12}$$

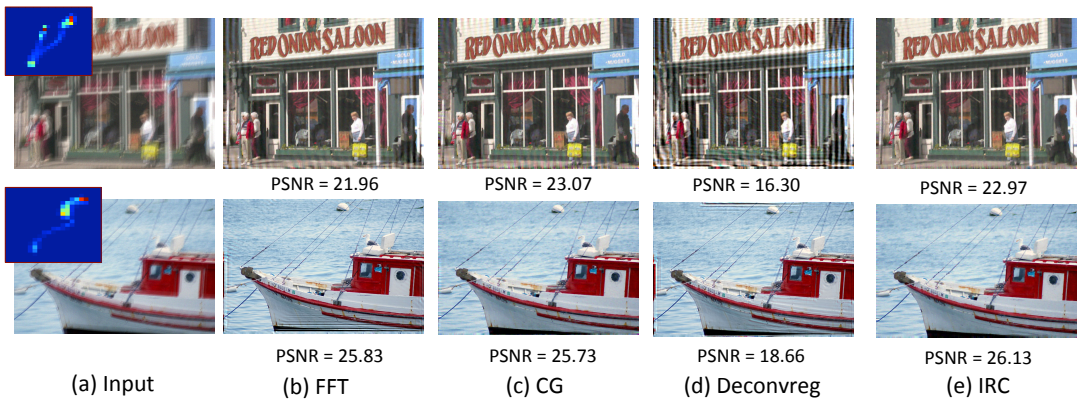


Figure 1: Examples of regularized deconvolution results. "Deconvreg" is Matlab's deconvolution method using Laplacian regularization.

where matrix K_i is the linear operator associated with k_i, and $\bar{d}_i$ is the vectorized version of d_i. This is a under-determined linear system. Writing $Q = \left[K_0^T, \cdots, K_n^T\right], \bar{d} = [\bar{d}_0^T, \cdots, \bar{d}_n^T]^T$, we can find its minimal-norm solution as

$$\bar{d} = Q^\dagger \bar{\delta} = Q^T (QQ^T)^{-1} \bar{\delta}, \tag{13}$$

where $Q^\dagger$ is the right pseudo-inverse of Q. Therefore each $\bar{d}_i$ is given by

$$\hat{d}_i = K_i (QQ^T)^{-1} \hat{\delta}, \quad \text{for } i = 0, \cdots, n. \tag{14}$$

Given the local *approximate* deconvolution filters, we run the iterative residual compensation algorithm (Alg. 2.1) to minimize the *correct* residual $\sum_{i=1}^{n} z_i - k_i \star x_i$.

• β_i **update :** The closed-form solution to

$$\min_{\beta_i} \langle \beta_i,\ k_i \star x - z_i \rangle, \quad \forall i \in [1, 2, \ldots, n]\,. \tag{15}$$

with update rate η_i is $\beta_i^* = \beta_i + \eta_i\,(k_i \star x - z_i)$.

2.1.1 Experimental Evaluation for the Regularized Model

We compare the extension of the IRC algorithm, i.e. Alg. 2.1, with the case of a penalized least-squares problem. We build a test set made of 20 images taken PASCAL VOC dataset blurred with the 8 kernels from [17]. Overall, we get 160 test images. We compare the IRC algorithm with the fast Fourier transform (FFT)-based algorithm (boundaries are carefully treated), FFT-nopad (no care for the boundaries) and a conjugate gradient descent-based method by solving:

$$\min_x ||k \star x - y||_2^2 + \lambda ||k_1 \star x||_1 + \lambda ||k_2 \star x||_1, \tag{16}$$

where $k_1 = [-1, 1]$ and $k_2 = [-1, 1]^T$). We set $\lambda = 0.003$, 10 iterations of the ADMM algorithm and $\rho = 2.43 \times 10^{-5}$, $\forall i \in [1, \ldots, 10]$ to balance the different terms of Eq. (5).

Table 1 gathers the evaluation results. It shows that our method competes with the traditional conjugate gradient descent method but is significantly faster. Indeed, over the 8 kinds of blur we consider, the proposed IRC algorithm achieves 4 times the best average PSNR score and 3 times, it is better than conjugate gradient descent, i.e. +0.23 dB for kernel 6, +0.41 dB for kernel 7 and +0.21 dB for kernel 8. Over the 160 images of the proposed benchmark, we achieve a better average PSNR score over CG (+0.06 dB) and the FFT-based method taking into account boundary effects (+0.49 dB). We are a slower than the FFT-based method (1.3 seconds for FFT against 8.6 seconds for IRC) because we need first to estimate the inverse kernel of k (with Wiener filtering for example) and then we apply the proposed iterative scheme. However, we are 8 times faster than the most serious competitors in terms of accuracy. This validates the choice of Alg. 2.1 to address non-blind deconvolution in a penalized setting. Figure 1 illustrates this comparison.

Methods	ker 1	ker 2	ker 3	ker 4	ker 5	ker 6	ker 7	ker 8	Aver.	Runtime (s)
FFT	25.95	24.99	26.01	22.44	**27.24**	26.80	24.58	23.20	25.16	1.3
FFT nopad	20.81	20.18	22.44	17.68	23.34	21.06	19.68	18.29	20.54	1.2
CG	**25.93**	**25.52**	**25.99**	**24.02**	27.11	26.61	24.96	24.56	25.59	60
IRC	25.91	25.33	**25.99**	23.74	27.23	**26.84**	**25.37**	**24.73**	**25.65**	$(6, 2.6)$

Table 1: Comparison of regularized deconvolution results. The numbers are average PSNRs over 160 test images. The computational time for IRC is split to the computation of all local deconvolution kernels (first number) and ADMM iterations with these kernels (second number).

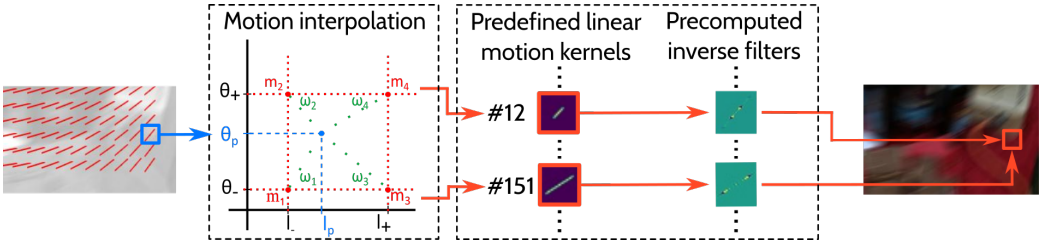


Figure 2: For a given patch taken from y, we assume that the corresponding blur kernel can be interpolated by four of the linear motions we have selected beforehand. We apply the corresponding inverse filters, computed ahead of time, to that patch and average the results. Repeating this operation to every overlapping patch of y enables fast non-uniform deconvolution.

2.2 Non-Uniform Blur

The motion blur is globally non-uniform but *locally* uniform. We can thus restore individual, overlapping patches of y with Alg. 2.1. However, this potentially requires computing as many inverse filters as there are pixels in y, which would dramatically slow down our approach in practice. To skirt this issue, we follow [14] and [26], and assume that a wide range of non-uniform motion blurs, including those due to camera rotation, can be *locally* approximated by linear motions. We select a relatively small set of linear motions to represent a wide range of *globally* non-uniform blurs. Following [26], we select the linear motions of magnitude $l = 1, 3, ..., 35$ pixels and orientations $\theta = 0, 6, \ldots, 174$ degrees. Overall, we consider 511 different filters ($l = 1$ is counted only once as it is the same filter for each possible value of θ). We compute offline an inverse kernel for each of those 511 filters.

During inference, a (continuous) linear motion parametrized by a couple (l, θ) in $\mathbb{R}_+ \times [0, 180)$ is associated with each patch from y. We propose to use interpolation to switch from this representation to our discrete predefined filters while limiting quantization errors. For a patch p with predicted motion $m = (l_p, \theta_p)$, we fetch from the 511 predefined linear motions the *closest* couples $m_1 = (l_-, \theta_-)$, $m_2 = (l_-, \theta+)$, $m_3 = (l_+, \theta_-)$ and $m_4 = (l_+, \theta+)$ such that $l_- \leq l_p \leq l_+$ and $\theta_- \leq \theta_p \leq \theta_+$. We then compute the Euclidean distances between m_p and the m_i and we make then sum to 1 by applying the softmax function S. The m_i correspond to four kernels linear motions kernels. If the d^i are the corresponding inverse filters, we have:

$$\hat{x} = \sum_{i=1}^{4} w_i(m_p) d^i \star p \quad \text{with} \quad w_i(m_p) = S(||m_p - m_i||_2^2). \tag{17}$$

This method is easily adaptable to an entire image. That is, we can remove any linear blur parametrized by continuous coordinates with a finite set of linear motions in a non-uniform setting. Figure 2 illustrates this non-uniform deconvolution step.

3 Non-Uniform Motion Blur Estimation and Removal

In this section, we design two models. The first one is a CNN taking as input y and outputing $\hat{k}$, an estimate of the locally linear non-uniform blur kernel k. The second one is a CNN formulation of the IRC algorithm to tackle non-blind deblurring when provided y and k or $\hat{k}$, and outputing $\hat{x}$, an

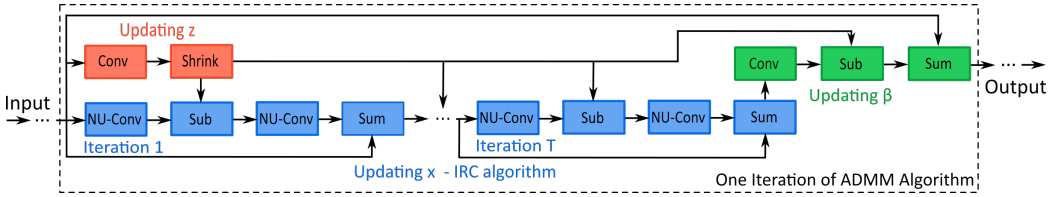


Figure 3: One stage of the ADMM algorithm with IRC for updating x. Concatenating N stages forms a trainable structure implementing the ADMM algorithm. In practice, $N = 5$ and $T = 2$ is enough the get good results. The gradient with respect to the loss can flow in the graph to update every trainable parameter like in a usual CNN.

estimate of the sharp image. Finally, we show how to combine both architectures to get an end-to-end learning framework that directly outputs $\hat{x}$ given only y.

3.1 Motion-Net for Non-Uniform Motion Estimation

We want to compute an estimate of the motion corresponding to y. Previous approaches select a small set of linear motions, as in the previous section, and assign to each pixel of y one of those motions. For example, Gong *et al.* [8] adapt a fully connected network (FCN) initially designed for semantic segmentation [18] to this classification problem. We take instead Deeplab [4], a CNN-based approach that has been shown to be faster and more accurate for semantic segmentation than [4], and adapt it to the *regression* problem of assigning a continuous linear motion to every pixel of y by replacing the final classification layer by a 2-channel (magnitude and direction) convolutional layer.

3.2 IRC-Net for Non-Uniform and Non-Blind Blur Removal

Figure 3 illustrates how we embed a few iterations of the ADMM algorithm within a CNN, dubbed IRC-Net. At each one of the N stages of the ADMM algorithm, we first proceed to the z_i update by mapping them onto the convolution ("Conv") and shrinkage ("Shrink") layers in the red part. Second, we sequentially solve Eq. (5) by mapping it onto a first non-uniform convolution layer ("NU-Conv") applying to each patch of y the corresponding local k_i kernels, a subtraction layer ("Sub"), a second non-uniform convolution layer ("NU-Conv") implementing the operations discussed in Section 2.2 and an addition layer ("Sum") in the blue branch. We repeat T times those four operations to execute Alg. 2.1. We finally update the scaled Lagrangian multipliers β in the green part.

The **NU-Conv layer** performs non-uniform convolution. The trainable parameters are filters initialized with the predefined inverse filters we compute ahead of time in the case of the second "NU-Conv" block of Fig. 3. Thus, we use the backpropagation algorithm to fine-tune the filters just like for a regular convolution layer. The **Shrink layer** performs the shrinkage operation involved in the z_i updates. We turn the soft-shrinkage function into a layer with learnable parameters following [34] in order to fit more flexible shrinkage operations.

3.3 Embedding Blur Estimation and Removal in a Neural Architecture

We can tackle blind deblurring problems by concatenating Motion-Net and IRC-Net in a single pipeline and fine-tune them in an end-to-end framework. To connect both models while ensuring the gradient can flow from IRC-Net to Motion-Net, we use the interpolation technique introduced in Section 2.2. Let us consider a patch p of corresponding linear motion $\hat{m}_p$ predicted by Motion-Net. If Motion-Net of parameters Θ_M is denoted by f and IRC-Net of parameters Θ_D, e.g., the precomputed inverse filters d_i, is denoted by g, we have:

$$\hat{x} = g(p, \{k_i\}_{i=1}^4, \{w_i\}_{i=1}^4 \; ; \; \Theta_D) \quad \text{with} \quad w_i = S(||f(p \; ; \; \Theta_M) - m_i||_2^2). \tag{18}$$

This method is easily adaptable to an entire image and is illustrated in Fig. 4. Thus, we can update Θ_D and Θ_M without any supervision on $\hat{k}$.

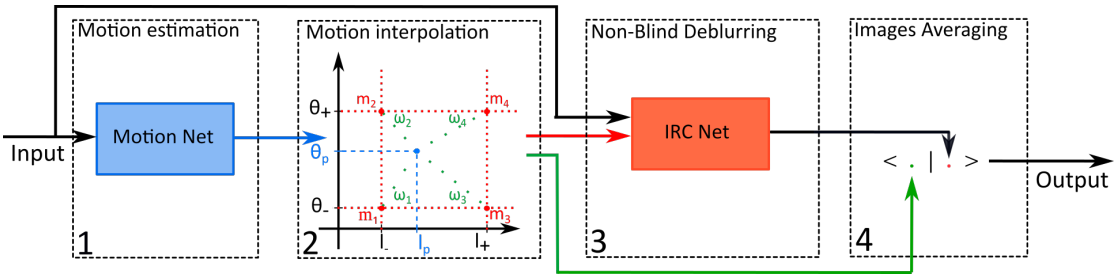


Figure 4: The proposed trainable pipeline to address blind deblurring problems. First, we predict the local linear motions responsible for the blur (1). We then look for the reference motions on a discrete grid corresponding to the predicted motion to build the estimate of the kernel $\hat{k}$ and compute matching weights kept in memory for the averaging step (2). We then use IRC-Net for each of the inverse filters selected (3) and finally we average all the restored versions of the blurry image y to build the estimate $\hat{x}$ (4).

4 Experiments

In this section, we experimentally validate the proposed models for motion estimation, non-blind and finally blind deblurring on synthetic and real-world images.

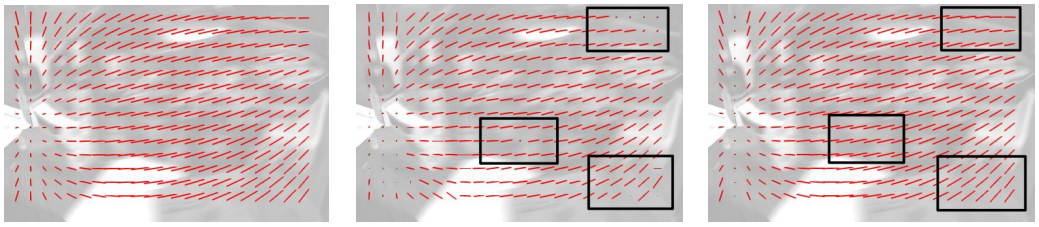
4.1 Motion Estimation

Real-world non-uniform blur kernels synthesis: A way to get a large dataset with ground-truth kernels is to synthesize camera motions as proposed in [8] but real-world motions captured by cameras are not just rotations and uniform translations. We create a dataset containing more realistic blurs with *pseudo* ground-truth kernels by averaging around 10 consecutive frames randomly selected from one of the GOPRO tracks provided by [20]. That is, we get a realistic blurry image. The corresponding kernel is obtained by estimating the optical flow between each pair of consecutive frames using [12]. Then, we sum all the predicted flows to get an approximate linear motion of each pixel in the image which can be used as a target information to learn to predict realistic non-uniform locally linear kernels.

Training: We build a dataset made of 7,000 images taken from VOC 2012 [7] and synthetically blurred with camera rotations and translations. We add 5,000 images built with the method explained in the previous paragraph. Overall, we gather a pool of 12,000 images containing various types of blurs. We select 10,000 images from that pool to be the train set and 2,000 to be the validation set. The maximum magnitude of the linear motions is 35 pixels. Finally, we build a test set of 600 images: 300 images are corrupted with motions up to 21 pixels and 300 other images up to 35 pixels. We randomly flip and rotate by 90 degrees the samples and add a Gaussian noise of variance 2. The model is trained during 150 epochs with Adam optimizer and starting learning rate of 10^{-4}, multiplied by 0.1 every 50 epochs. We use full images with batch size of 1 and replace batch normalization by instance normalization [28] as we use only one image per forward pass during training. Table 2 and Fig. 5 show that our approach achieves slightly better results than the ones of FCN [8] with respect to the metric proposed by [26]. Motion-Net produces smoother results while being nearly 50 times faster.

Methods	MSE (Moderate)	MSE (Severe)	Runtime (s)
CNN-Patch [26]	20.08	54.02	79
FCN [8]	7.09	10.34	7
Ours	**5.91**	**8.36**	0.2

Table 2: Results for motion estimation over a test set made of 600 blurry images. Mean runtime is for RGB images of size 500 $\times$ 375 pixels.



(a) Ground-truth motion blur (b) [8], MSE = 23.83 (c) Ours, MSE = 8.59

Figure 5: Example of motion field estimation for the severe dataset.

4.2 Non-Blind Deblurring

Initialization: The number of stages of the network is set to 5 with 2 iterations for the inner network responsible for IRC. In the formulation of Eq. (1), we set $l = 1$ and we initialize k_1 and k_2 with vertical and horizontal gradient filters.

Training: We train IRC-Net using stochastic gradient descent and learning rate set to 10^{-7} on a synthetic dataset made of 5,400 synthetically blurred patch pairs of size 121×121 with motions corresponding to the selected linear motions presented in Section 2.2 to update a single precomputed inverse filters with the exact corresponding linear motion. We do not use the *pseudo* ground-truth kernels used for training Motion-Net as predictions may be incorrect, leading to inexact ground-truth data and thus, wrong updates of the parameters of IRC-Net. At test time, we use continuous linear motions and take advantage of the interpolation technique explained in Section 2.2. We quantitatively evaluate our approach on a synthetic test set with 60 pairs of clean and blurry images from VOC 2012 with synthetic spatially-varying motion blur. The training loss is the l_2 distance. Table 3 shows that our approach, before fine-tuning, gives significantly better results than a widely used FFT-based non-uniform deconvolution method (NU-FFT) [22, 33] and Gaussian mixture method (GMM) prior-based approach (GMM-Patch) but falls behind the conjugate gradient-based algorithm (NU-CG) with a gap of 0.23dB. When fine-tuned, it achieves slightly better results than NU-CG, i.e. +0.14dB, and is more than 300 times faster.

Methods	PSNR (dB)	SSIM	Runtime (s)
NU-FFT	22.97	0.776	47
NU-CG [29]	26,77	**0.864**	335
GMM-Patch [26]	26.10	0.843	415
Ours	26.54	0.855	0.8
Ours (fine-tuned)	**26.91**	0.861	0.8

Table 3: Quantitative comparison to other non-blind methods. Mean runtime is for RGB images of size 375×500 pixels.

(a) NU-CG (b) Ours (c) Ours (fine-tuned)

Figure 6: Example of non-blind deblurring results. NU-CG achieves a PSNR score of 25.76 dB. Our approach achieves 25.56 dB and after fine-tuning reaches 25.75 dB.

4.3 Blind Deblurring

We select the IRC-Net with two inner iterations and we combine it with the proposed Motion Net to build a blind deblurring approach following the method described in Paragraph 3.3. We fine-tune the pipeline on a dataset made of 3,000 synthetic images and 3,000 images from the dataset provided

by [20] with stochastic gradient descent with learning rate set to 10^{-9}, momentum set to 0.9, and batch size set to 1. The training loss is the l_2 distance. Fine-tuning Motion-Net and IRC-Net together improves the deblurring results.

Methods	PSNR (dB)	SSIM	PSNR (dB)	SSIM	Runtime (s)
GMM-Patch [26]	24.14	0.714	20.84	0.56	497
GMM-FCN [8]	24.81	0.742	22.38	0.622	422
Reblurring [1]	**24.87**	0.743	22.39	**0.646**	n/a
DMCNN [20]	24.68	**0.760**	22.28	**0.646**	3.4
Ours	24.71	0.734	22.50	0.630	0.9
Ours (fine-tuned)	24.80	0.737	**22.54**	0.632	0.9

Table 4: Quantitative comparison for blind deblurring methods on the datasets of [1]. The mean runtime is for images of size 375×500 pixels. Left column is for moderate blur, right one for severe.

We report in Table 4 comparison results for the datasets of [1]. We are significantly above the classical baseline [26] in terms of accuracy and running time. In terms of PSNR, we also compete with recent approaches such as [20, 8, 1]. We are slightly above the CNN of [20] by 0.12 dB on the moderate dataset and by 0.26 dB on the severe dataset while being 5 times faster. We are more than 400 times faster than [8] while achieving similar results on the moderate dataset and better results on the severe benchmark (+0.16 dB). We are behind [1] by only 0.07 dB. We achieve poorer results in terms of SSIM than the best competitors on both benchmarks but not by far, i.e. -0.06 on the moderate dataset and -0.14 on the severe dataset in comparison with [1]. Blur estimation and removal on real-world images are illustrated in Fig. 7. More examples are presented in the supplementary material.

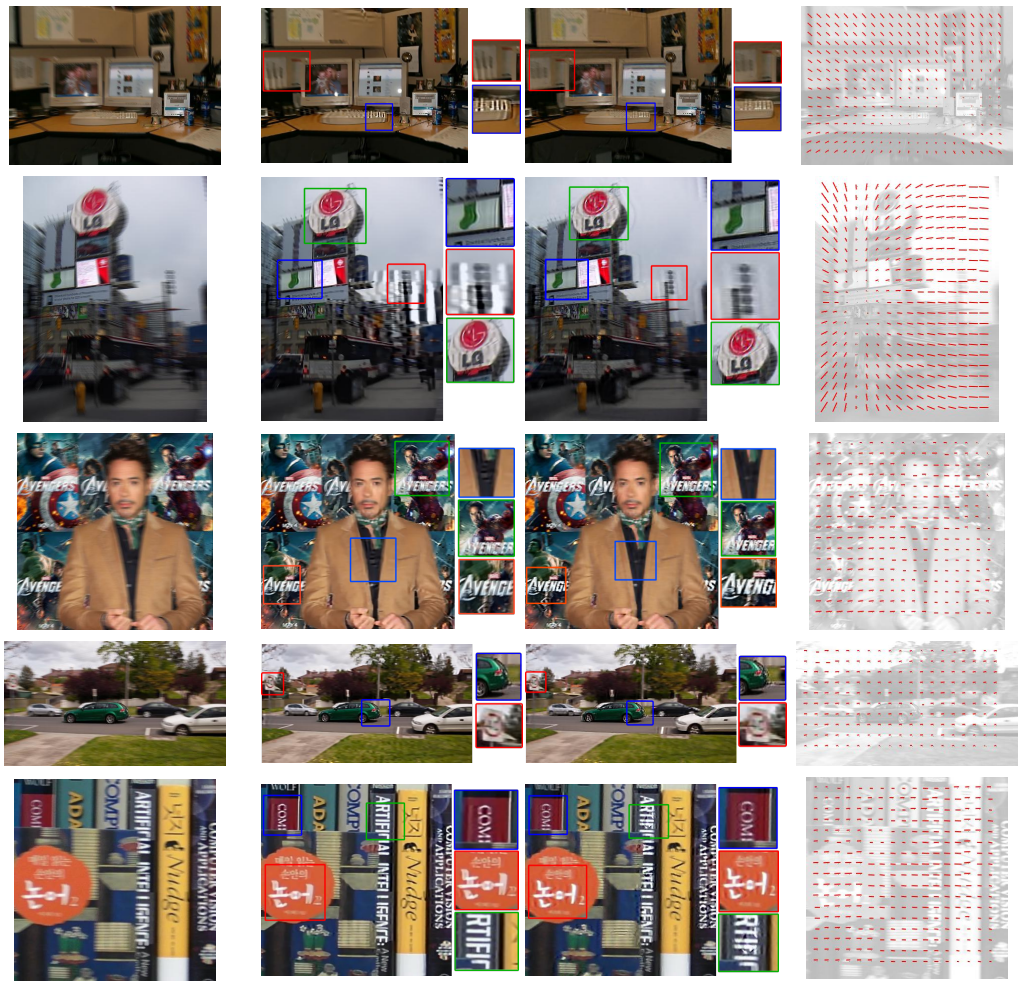
Methods	PSNR (dB)	SSIM	Runtime (s)
GMM-Patch [26]	25.22	0.774	20 minutes
GMM-FCN [8]	24.92	0.784	20 minutes
Reblurring [1]	26.33	n/a	n/a
DMCNN [20]	**26.48**	**0.808**	4.7
Ours	25.01	0.779	2.1
Ours (fine-tuned)	25.42	0.795	2.1

Table 5: Quantitative comparison for blind deconvolution methods on the dataset of [15]. Mean runtime is for images of size 800×800 pixels.

Table 5 summarizes comparison results on the classical benchmark for blind image deblurring proposed by Köhler *et al.* [15]. We achieve a PSNR score of 25.49 dB, which is better than similar approaches proposing to estimate and then remove blur such as [8] and [26]. They respectively score 24.92 dB and 25.22 dB. It is well behind the 26.33 dB claimed by [1] or the 26.48 dB claimed by [20]. This shows a limitation of our method based on approximating non-uniform blurs with local linear motions. Very complex blurs that cannot be approximated by linear motions, e.g., curve motions, require better models to be estimated and then removed in a two-steps fashion.

5 Conclusion

We have proposed an iterative deconvolution approach and its embedding in a fine-tuned CNN where each feature map corresponds to a step of an ADMM algorithm, demonstrating its effectiveness on non-uniform deblurring problems. This learning-based method relies on image processing arguments, leading to fewer parameters and less intensive computations than deep CNNs. To extend it to the blind setting, we have designed a new motion regressor and a technique to combine both architectures within a single learning framework that competes with recent approaches while being faster. For



(a) Input (b) Bahat and Irani [1] (c) Ours - $\hat{x}$ (d) Ours - $\hat{k}$

Figure 7: Examples of blind deblurring results. The two first rows present synthetically blurred images. The other rows show real-world images.

future work, we plan to adapt Motion-Net to more complex local kernels than linear motions and enhance IRC-Net with richer priors such as [23] or [27].

Acknowledgments This work was supported in part by ERC grant VIDEOWORLD, the MSR-Inria joint lab and the Louis Vuitton ENS Chair on Artificial Intelligence.

References

[1] Yuval Bahat, Netalee Efrat, and Michal Irani. Non-uniform blind deblurring by reblurring. In *2017 IEEE International Conference on Computer Vision*, 2017.

[2] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *bite*, 2010.

[3] Ayan Chakrabarti. A neural approach to blind motion deblurring. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III*, pages 221–235, 2016.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848, 2018.

[5] Sunghyun Cho and Seungyong Lee. Fast motion deblurring. In *ACM SIGGRAPH Asia 2009 Papers*, SIGGRAPH Asia '09, pages 145:1–145:8, New York, NY, USA, 2009. ACM.

[6] Michael Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[8] Dong Gong, Jie Yang, Lingqiao Liu, Yanning Zhang, Ian Reid, Chunhua Shen, Anton van den Hengel, and Qinfeng Shi. From motion blur to motion flow: A deep learning solution for removing heterogeneous motion blur. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[10] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 399–406, 2010.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[12] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[14] Tae Hyun Kim and Kyoung Mu Lee. Segmentation-free dynamic scene deblurring. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 2766–2773, 2014.

[15] Rolf Köhler, Michael Hirsch, Betty Mohler, Bernhard Schölkopf, and Stefan Harmeling. Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 27–40, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[16] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiri Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. *CoRR*, abs/1711.07064, 2017.

[17] Anat Levin, Yair Weiss, Frédo Durand, and William T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 1964–1971, 2009.

[18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3431–3440, 2015.

[19] Tomer Michaeli and Michal Irani. Blind deblurring using internal patch recurrence. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III*, pages 783–798, 2014.

[20] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 257–265, 2017.

[21] Mehdi Noroozi, Paramanand Chandramouli, and Paolo Favaro. Motion deblurring in the wild. *CoRR*, abs/1701.01486, 2017.

[22] Michael oHirsch, Christian J. Schuler, Stefan Harmeling, and Bernhard Schölkopf. Fast removal of non-uniform camera shake. In Dimitris N. Metaxas, Long Quan, Alberto Sanfeliu, and Luc J. Van Gool, editors, *ICCV*, pages 463–470. IEEE Computer Society, 2011.

[23] Michael oHirscoHirsch, Christian J. Schuler, Stefan Harmeling, and Bernhard Schölkopf. Fast removal of non-uniform camera shake. In *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, pages 463–470, 2011.

[24] William Hadley Richardson. Bayesian-based iterative method of image restoration. *JOSA*, pages 55–59, 1972.

[25] Uwe Schmidt, Kevin Schelten, and Stefan Roth. Bayesian deblurring with integrated noise estimation. In *CVPR*, pages 2625–2632. IEEE Computer Society, 2011.

[26] Jian Sun, Wenfei Cao, Zongben Xu, and Jean Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 769–777, 2015.

[27] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. *arXiv:1711.10925*, 2017.

[28] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. *CoRR*, abs/1701.02096, 2017.

[29] Yilun Wang, Junfeng Yang, Wotao Yin, and Yin Zhang. A new alternating minimization algorithm for total variation image reconstruction. *SIAM J. Imaging Sciences*, 1(3):248–272, 2008.

[30] Oliver Whyte, Josef Sivic, Andrew Zisserman, and Jean Ponce. Non-uniform deblurring for shaken images. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 491–498, 2010.

[31] Li Xu, Jimmy S. J. Ren, Ce Liu, and Jiaya Jia. Deep convolutional neural network for image deconvolution. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, pages 1790–1798, Cambridge, MA, USA, 2014. MIT Press.

[32] Li Xu, Xin Tao, and Jiaya Jia. Inverse kernels for fast spatial deconvolution. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, pages 33–48, 2014.

[33] Li Xu, Shicheng Zheng, and Jiaya Jia. Unnatural L0 sparse representation for natural image deblurring. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 1107–1114, 2013.

[34] yan yang, Jian Sun, Huibin Li, and Zongben Xu. Deep admm-net for compressive sensing mri. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 10–18. Curran Associates, Inc., 2016.

[35] Jiawei Zhang, Jin-shan Pan, Wei-Sheng Lai, Rynson W. H. Lau, and Ming-Hsuan Yang. Learning fully convolutional networks for iterative non-blind deconvolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6969–6977, 2017.

[36] Daniel Zoran and Yair Weiss. From learning models of natural image patches to whole image restoration. In *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, pages 479–486, 2011.