Learning to Push the Limits of Efficient FFT-based Image Deconvolution

Jakob Kruse[1] Carsten Rother[1] Uwe Schmidt[2]
[1] TU Dresden [2] MPI of Molecular Cell Biology and Genetics, Dresden

Abstract

This work addresses the task of non-blind image deconvolution. Motivated to keep up with the constant increase in image size, with megapixel images becoming the norm, we aim at pushing the limits of efficient FFT-based techniques. Based on an analysis of traditional and more recent learning-based methods, we generalize existing discriminative approaches by using more powerful regularization, based on convolutional neural networks. Additionally, we propose a simple, yet effective, boundary adjustment method that alleviates the problematic circular convolution assumption, which is necessary for FFT-based deconvolution. We evaluate our approach on two common non-blind deconvolution benchmarks and achieve state-of-the-art results even when including methods which are computationally considerably more expensive.

1. Introduction

Image deblurring is a classic image restoration problem with a vast body of work in computer vision, signal processing and related fields (see [25] for a fairly recent survey). In this work, we focus on the case of uniform blur, where the observed blurred image $\mathbf{y} = \mathbf{k} \otimes \mathbf{x} + \boldsymbol{\eta}$ is obtained via convolution of the true image $\mathbf{x}$ with known blur kernel (point spread function) $\mathbf{k}$ and additive Gaussian noise $\boldsymbol{\eta}$. The task of recovering $\mathbf{x}$ is then called (non-blind) image *deconvolution*. Note that although the assumption of uniform blur is often not accurate [12, 15], such image deconvolution techniques can in fact outperform methods which assume a more realistic non-uniform blur model [*cf.* 12]. Furthermore, image deconvolution can be used as a building block to address the removal of non-uniform blur [*cf.* 28].

When it comes to image deconvolution methods, it is useful to broadly separate them into two classes: *(1)* those where the most costly computational operations are a fixed number of Fourier transforms or convolutions, and *(2)* those which require more expensive computation, often due to (iterative) solvers for large linear systems of equations. While the first class of methods can scale to large megapixel-sized images, the latter class generally falls short in this regard. These computationally demanding methods often exhibit high restoration quality [*e.g.*, 20, 31], but typically need several minutes, or more, to deconvolve images of 1 megapixel (see current deblurring benchmarks [*e.g.*, 12, 24]). Of course, this runtime issue is even more severe for images that are multiple times larger, which are common nowadays. While the power of computers increases each year, so does the size of images taken by typical cameras. For the reasons outlined above, in this paper we focus on a class of deconvolution methods where the fast Fourier transform (FFT) is the most expensive operation, with computational complexity $\mathcal{O}(n \log n)$ for input size, *i.e.* pixel count, n. Note that, while our proposed method is very efficient, we even slightly outperform state-of-the-art techniques which are considerably more computationally expensive.

Since image deconvolution is mathematically ill-posed in the presence of noise, some form of regularization has to be used to recover a restored image. A classic fast FFT-based deconvolution method is the *Wiener filter* [27], which uses quadratic regularization of the expected image spectrum to obtain the restored image in closed form. However, it is well-known that quadratic regularization is not ideal for natural images, which we assume here. Hence, better methods [*e.g.*, 13, 26] employ sparse regularization and iterative optimization, where each iteration is similar to a Wiener filter. More recently, the advent of discriminative deblurring [22] has generalized these methods to yield even higher quality results without increasing the computational demands [9, 21, 28]. In Section 2, we study these traditional FFT-based deconvolution methods and their more recent learning-based extensions. Based on this analysis, we propose a new, generalized learning-based approach utilizing the power of convolutional neural networks in Section 3.

In order to improve the quality of the restored image even further, we address the often neglected topic of image boundary handling. FFT-based deconvolution hinges on a blur model which assumes a convolution with periodic (circular) boundary conditions. Unfortunately, this assumption is almost never satisfied for blurred natural images, such as typical photographs degraded by camera shake or motion blur. To alleviate this problem, an observed blurred image is typically padded and pre-processed by an "edgeta-

To appear in Proceedings of the *IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, October 2017.

per" operation[1] [*cf.* 19], which applies additional circular convolution to only the boundary region of the padded image. However, we want to go beyond this dated boundary processing approach. Towards this end, we take inspiration from recent work [2, 16] and devise a simple, yet effective, boundary adjustment strategy that can easily be applied to any FFT-based deconvolution method, without introducing additional parameters or computational cost.

We show the efficacy of our proposed model and boundary adjustment method in various non-blind deconvolution experiments in Section 4, before we conclude in Section 5.

In this work, we solely focus on non-blind deconvolution, while recent research in the field has arguably shifted its focus towards blind deconvolution, which aims to estimate both the blur kernel $\mathbf{k}$ and the restored image $\mathbf{x}$. However, most of these approaches make use of non-blind deconvolution steps [*e.g.*, 4, 29]. Recent discriminative methods [23, 28] alternate between updating the blur kernel and employing non-blind deconvolution to update the restored image. Hence, it remains important to develop better non-blind techniques.

In summary, our main contributions are threefold:

- We generalize discriminative FFT-based deconvolution approaches by using more powerful regularization based on convolutional neural networks.
- We propose a simple and effective boundary adjustment method that alleviates the problematic circular convolution assumption, which is necessary for FFT-based deconvolution.
- We obtain state-of-the-art results on non-blind deconvolution benchmarks, even when including methods that are computationally considerably more expensive.

2. Review of FFT-based deconvolution

We consider the common blur model

$$\mathbf{y} = \mathbf{k} \otimes \mathbf{x} + \boldsymbol{\eta}, \tag{1}$$

where the observed corrupted image $\mathbf{y}$ is the result of circular convolution of the image $\mathbf{x}$ with blur kernel $\mathbf{k}$ plus Gaussian noise with variance σ^2, *i.e.* $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}/\lambda)$ with precision $\lambda = 1/\sigma^2$ and $\mathbf{I}$ being the identity matrix. For notational convenience, we assume all variables in bold to be vectors *(lower case)* or matrices *(upper case)*.

2.1. Traditional approaches

A classic solution to obtain an estimate of the restored image is given by the Wiener filter [27] as

$$\hat{\mathbf{x}} = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(\mathbf{k} \circledast \mathbf{y})}{|\mathcal{F}(\mathbf{k})|^2 + \mathbf{n}/\mathbf{s}}\right), \tag{2}$$

where $\mathcal{F}$ corresponds to the two-dimensional discrete Fourier transform and $\mathbf{n} = \mathbf{1}/\lambda$ and $\mathbf{s}$ are the expected power spectra of the noise and image, respectively. Note that $\mathbf{k} \circledast \mathbf{y} = \mathcal{F}^{-1}(\overline{\mathcal{F}(\mathbf{k})} \odot \mathcal{F}(\mathbf{y}))$ denotes correlation of $\mathbf{y}$ and $\mathbf{k}$, where $\odot$ is the entrywise (Hadamard) product and $\overline{\mathbf{v}}$ is the complex conjugate of $\mathbf{v}$. All other operations in Eq. (2), such as division, are applied entrywise.

The Wiener filter is very efficient due to FFT-based inference, but not state-of-the-art anymore. Many modern methods are based on minimizing an energy function

$$E(\mathbf{x}) = \frac{\lambda}{2}\|\mathbf{k} \otimes \mathbf{x} - \mathbf{y}\|^2 + \sum\nolimits_i \mathbf{1}^\mathsf{T} \rho_i(\mathbf{f}_i \otimes \mathbf{x}), \tag{3}$$

where the data term stems from the blur model of Eq. (1), and the regularization term with $i = 1, \ldots, N$ is based on penalty functions ρ_i, applied entrywise to responses of linear filters $\mathbf{f}_i$, which most commonly are simple image derivative filters [*e.g.*, 13]. If quadratic penalty functions are used, *i.e.* $\rho_i(u) = \frac{\beta}{2}u^2$, then $\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} E(\mathbf{x})$ yields the same form as the Wiener filter of Eq. (2), except that the spectrum $\mathbf{s}$ is replaced by $\beta \sum_i |\mathcal{F}(\mathbf{f}_i)|^2$:

$$\hat{\mathbf{x}} = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(\mathbf{k} \circledast \mathbf{y})}{|\mathcal{F}(\mathbf{k})|^2 + \frac{\beta}{\lambda}\sum_i |\mathcal{F}(\mathbf{f}_i)|^2}\right). \tag{4}$$

It is well-known that quadratic regularization leads to inferior restoration results for natural images. High-quality results can be achieved by using sparse (non-quadratic) regularization terms, *e.g.* with hyper-Laplacian penalty functions $\rho_i(u) = |u|^\alpha$ and $0 < \alpha \leq 1$ [13]. Unfortunately, this makes energy minimization more complicated. To address this issue, it has been shown helpful to use *half-quadratic splitting* [10, 26], in this context also known as *quadratic penalty method* [*cf.* 17, § 17.1]. To that end, the energy is augmented with latent variables $\mathbf{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ as

$$E_\beta(\mathbf{x}, \mathbf{z}) = \frac{\lambda}{2}\|\mathbf{k} \otimes \mathbf{x} - \mathbf{y}\|^2 + \sum\nolimits_i \mathbf{1}^\mathsf{T} \rho_i(\mathbf{z}_i) + \frac{\beta}{2}\|\mathbf{f}_i \otimes \mathbf{x} - \mathbf{z}_i\|^2, \tag{5}$$

such that $E(\mathbf{x}) = \lim_{\beta \to \infty} E_\beta(\mathbf{x}, \mathbf{z})$. Energy minimization is now carried out in an iterative manner, where the latent variables and the restored image are updated at each step t:

$$\mathbf{z}_i^{t+1} = \arg\min\nolimits_{\mathbf{z}_i} E_\beta(\mathbf{x}^t, \mathbf{z}) = \psi_i(\mathbf{f}_i \otimes \mathbf{x}^t) \tag{6}$$

$$\mathbf{x}^{t+1} = \arg\min\nolimits_{\mathbf{x}} E_\beta(\mathbf{x}, \mathbf{z}^{t+1}). \tag{7}$$

In Eq. (6), $\psi_i = \mathrm{prox}_{\rho_i/\beta}$ is a 1D shrinkage function obtained as the proximal operator of penalty ρ_i with parameter β^{-1} [*cf.* 18]. Note that β needs to be increased during optimization such that the result of the optimization closely resembles a solution to the original energy of Eq. (3). By combining Eqs. (6) and (7), we obtain the following update

[1] *cf.* MATLAB's `edgetaper` function.

equation for the restored image at step t:

$$\mathbf{x}^{t+1} = \mathcal{F}^{-1}\left(\frac{\mathcal{F}\big(\mathbf{k} \circledast \mathbf{y} + \frac{\beta}{\lambda}\phi(\mathbf{x}^t)\big)}{|\mathcal{F}(\mathbf{k})|^2 + \frac{\beta}{\lambda}\sum_i |\mathcal{F}(\mathbf{f}_i)|^2}\right) \tag{8}$$

$$\text{with} \quad \phi(\mathbf{x}^t) = \sum\nolimits_i \mathbf{f}_i \circledast \psi_i(\mathbf{f}_i \otimes \mathbf{x}^t). \tag{9}$$

Note that Eq. (8) has the same form as Eq. (4) when using a quadratic penalty, with the only difference that the term $\frac{\beta}{\lambda}\phi(\mathbf{x}^t)$, based on the current image estimate $\mathbf{x}^t$, appears in the numerator. While this change may seem insignificant, it does lead to deconvolution results of much higher quality when Eq. (8) is applied iteratively [13, 26].

Note that there are many different variants of splitting methods [*cf.* 8] besides the one that we presented above, such as the popular *alternating direction method of multipliers* (ADMM) [*cf.* 17, § 17.4]. Applied in our context, ADMM is actually an extension of the splitting approach of Eqs. (5) to (9) with the benefit of converging more quickly to a minimum of Eq. (3). However, such improved convergence behavior is not relevant for us, since we will use a discriminative generalization of Eqs. (8) and (9) that does not aim to minimize Eq. (3) anymore.

2.2. Discriminative learning-based approaches

Discriminative non-blind deblurring has been proposed by Schmidt *et al.* [20] through "unrolling" iterative half-quadratic optimization for a fixed small number of steps. Good results can be achieved by learning specialized model parameters for each of the few optimization steps. Schmidt and Roth [21] applied this idea to the iterative FFT-based deconvolution updates of Eq. (8) by learning step-specific weights β_t, shrinkage functions ψ_{it}, and linear filters $\mathbf{f}_{it}$ to directly optimize the quality of the restored image, instead of minimizing Eq. (3). Crucial to their approach is that they directly modeled ψ_{it} as differentiable functions with closed-form expressions, which allowed them to use standard loss-based training with gradient-based methods. They called the resulting model a *cascade of shrinkage fields* (CSF).

Zhang *et al.* [30] adopted a similar approach to shrinkage fields, but used fixed horizontal and vertical image gradient filters $\mathbf{f}_{it}$ and replaced univariate shrinkage functions ψ_{it} with standard convolutional neural networks (CNNs). However, they did not train the CNNs to directly optimize image quality. In the context of combining low-level and high-level image processing, Diamond *et al.* [9] extended shrinkage fields specifically for color images by replacing shrinkage functions with CNNs that operate independently in the spatial domain but exploit correlations between color channels. Schuler *et al.* [23] addressed discriminative blind deconvolution by making use of CNNs and alternating updates of the restored image and blur kernel. However, their image updates are based on a simple Wiener filter, where they used a flat spectrum $\mathbf{n}/\mathbf{s} = \beta\mathbf{1}$ with learned scalar β.

3. Our approach

Given the insights from the previous section, we now introduce our own approach which further generalizes the formulation of discriminative methods. After that, we describe our second contribution, which is a simple, yet effective boundary adjustment technique. An overview of our full approach is illustrated in Fig. 1.

Although in a limited manner, previous work [9, 30] has already attempted to replace the 1D shrinkage functions ψ_i in Eq. (9) with CNNs that go beyond pixel-independent processing. However, we want to go further than just replacing ψ_i and instead propose to replace ϕ (Eq. 9) altogether with a CNN, thereby generalizing Eq. (8), since numerator and denominator are no longer coupled through shared filters $\mathbf{f}_i$. As a result, we alter the update step to

$$\mathbf{x}^{t+1} = \mathcal{F}^{-1}\left(\frac{\mathcal{F}\big(\mathbf{k} \circledast \mathbf{y} + \frac{1}{\omega_t(\lambda)}\phi_t^{\text{CNN}}(\mathbf{x}^t)\big)}{|\mathcal{F}(\mathbf{k})|^2 + \frac{1}{\omega_t(\lambda)}\sum_i |\mathcal{F}(\mathbf{f}_{it})|^2}\right), \tag{10}$$

where we make explicit that we learn a specialized CNN-based term ϕ_t^{CNN} for every step t besides the linear filters $\mathbf{f}_{it}$. Furthermore, we replace λ with a learned scalar function $\omega_t(\lambda)$ that acts as a *noise-specific* regularization weight; this is necessary, because simply using $\lambda = 1/\sigma^2$ based on noise level σ empirically leads to sub-par results. Most previous work [*e.g.*, 13, 20, 21, 23] addressed this issue by learning a *fixed* regularization weight ω_t, hence they need to train a separate model for each noise level σ. In contrast, Eq. (10) generalizes well to a range of noise levels if exposed to them during training (*cf.* Section 4). Note that we also remove the scalar weight β, since it can be absorbed into $\omega_t(\lambda)$, which we parameterize as a multilayer perceptron.

In general, our motivation is to push the limits of a flexible and powerful regularization, without breaking the efficient FFT-based optimization, which is made possible by the assumptions underlying the common blur formation model of Eq. (1). To improve the quality of the restored image even further, we should also make sure that these assumptions are satisfied, which specifically are: *(1)* convolution is carried out with circular (periodic) boundary conditions and *(2)* that noise is additive and drawn pixel-independently from a Gaussian distribution.

While the Gaussian noise assumption can be problematic, especially in low-light conditions, we are not going to address this here. Instead, we focus on the issue that the circular convolution assumption is especially troubling for typical blurred photographs, since it can lead to strong restoration artifacts [*cf.* 19]. A more realistic blur model is that the convolution $\mathbf{y} = \mathbf{k} \otimes_{\text{V}} \mathbf{x}$ does not go beyond the image boundary[2] of $\mathbf{x}$. As a result, the observed blurred image $\mathbf{y} \in \mathbb{R}^m$ is actually smaller than the true image

[2] Often called convolution with "valid" or "inner" boundary conditions.

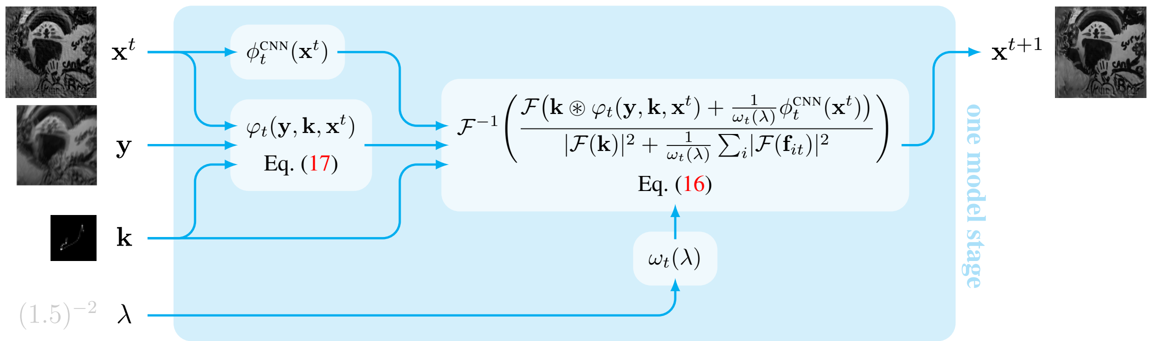


Figure 1: **Overview for one model stage.** We propose an extension of iterative FFT-based deconvolution methods, which update the restored image $\mathbf{x}^t \to \mathbf{x}^{t+1}$ at each step t. Specifically, we generalize *shrinkage fields* [21] by removing unnecessary parameter sharing and replacing pixel-wise applied shrinkage functions with CNNs (ϕ_t^{CNN}) that operate on the whole image, *i.e.* can take advantage of spatial dependencies between pixels (see Fig. 6 for an example). Additionally, we take inspiration from [2, 16] and propose a simple, yet effective, strategy to better adhere to the circular blur assumption that underlies all FFT-based deconvolution methods; to that end, we replace the observed blurred image $\mathbf{y}$ with $\varphi_t(\mathbf{y}, \mathbf{k}, \mathbf{x}^t)$.

$\mathbf{x} \in \mathbb{R}^n$, *i.e.* $m < n$. Hence, we would ideally like to use unknown boundary conditions, *i.e.* disable the blur model at the boundary and only use the regularization term. Unfortunately, only determinate boundary conditions may lead to structured optimization problems that admit fast inference [*cf.* 2, 16]. Of those, circular boundary conditions are arguably the most appealing, since they lead to equation systems with matrices that can be diagonalized in Fourier space, hence admit fast and closed-form image updates as presented throughout this section. Given this, we seek to modify the observed blurred image $\mathbf{y}$ to better adhere to the circular blur model, which we discuss next.

3.1. Boundary adjustment

A common boundary pre-processing step is to first pad the observed blurred image $\mathbf{y} \in \mathbb{R}^m$ by replicating its edge pixels[3] with linear operator $\mathbf{P}_r \in \mathbb{R}^{n \times m}$ such that $\mathbf{P}_r\mathbf{y} \in \mathbb{R}^n$ has the same size as the true image $\mathbf{x} \in \mathbb{R}^n$. This is followed by the classic edgetaper operation [*cf.* 19] to arrive at the modified blurred image

$$\tilde{\mathbf{y}} = \text{edgetaper}(\mathbf{P}_r\mathbf{y}, \mathbf{k}), \tag{11}$$

which better adheres to the circular blur model. While this pre-processing approach goes a long way to reduce restoration artifacts, it is several decades old and does not solve the problem completely.

In order to come up with a better approach, Matakos *et al.* [16] and Almeida and Figueiredo [2] proposed to change the blur model from Eq. (1) to

$$\mathbf{y} = \mathbf{C}(\mathbf{k} \otimes \mathbf{x}) + \boldsymbol{\eta}, \tag{12}$$

where convolution is still carried out with periodic boundary conditions, but the result is cropped via multiplication with matrix $\mathbf{C} \in \mathbb{R}^{m \times n}$, such that only the inner part is retained, which is of the same size as the observed blurred image $\mathbf{y}$. Note that $\mathbf{C}(\mathbf{k} \otimes \mathbf{x}) = \mathbf{k} \otimes_{\text{V}} \mathbf{x}$ corresponds to the more realistic blur assumption, as described above. Using common regularizers, both [2, 16] then develop an efficient deconvolution algorithm based on the ADMM framework.

Since at the core of their approach is also a quadratic splitting technique, we can adopt it to extend Eq. (5). To that end, we replace $\mathbf{k} \otimes \mathbf{x}$ by the latent vector $\mathbf{u}$ and impose a soft constraint based on weight γ that favors both terms to be equal. With such a modification, we arrive at

$$E_{\beta,\gamma}(\mathbf{x}, \mathbf{z}, \mathbf{u}) = \frac{\lambda}{2}\|\mathbf{C}\mathbf{u} - \mathbf{y}\|^2 + \frac{\gamma}{2}\|\mathbf{k} \otimes \mathbf{x} - \mathbf{u}\|^2 + \sum_i \mathbf{1}^\mathsf{T}\rho_i(\mathbf{z}_i) + \frac{\beta}{2}\|\mathbf{f}_i \otimes \mathbf{x} - \mathbf{z}_i\|^2, \tag{13}$$

where again $E(\mathbf{x}) = \lim_{\beta\to\infty,\gamma\to\infty} E_{\beta,\gamma}(\mathbf{x}, \mathbf{z}, \mathbf{u})$, but based on the new blur model of Eq. (12).

We now employ the same alternating optimization as we have used in Eqs. (5) to (7), but based on Eq. (13) and with the additional update step $\mathbf{u}^{t+1} = \arg\min_{\mathbf{u}} E_{\beta,\gamma}(\mathbf{x}^t, \mathbf{z}^t, \mathbf{u})$. Note that the update steps for $\mathbf{x}$ and $\mathbf{z}$ actually do not change, the difference is only that $\mathbf{u}$ has taken the place of $\mathbf{y}$ (and γ that of λ). Again, we combine all equations to obtain the update of the restored image at step t as

$$\mathbf{x}^{t+1} = \mathcal{F}^{-1}\left(\frac{\mathcal{F}\left(\mathbf{k} \circledast \varphi(\mathbf{y}, \mathbf{k}, \mathbf{x}^t) + \frac{\beta}{\gamma}\phi(\mathbf{x}^t)\right)}{|\mathcal{F}(\mathbf{k})|^2 + \frac{\beta}{\gamma}\sum_i |\mathcal{F}(\mathbf{f}_i)|^2}\right), \tag{14}$$

where ϕ is defined as in Eq. (9) and

$$\begin{aligned}\varphi(\mathbf{y}, \mathbf{k}, \mathbf{x}^t) &= \left(\tfrac{\lambda}{\gamma}\mathbf{C}^\mathsf{T}\mathbf{C} + \mathbf{I}\right)^{-1}\left(\tfrac{\lambda}{\gamma}\mathbf{C}^\mathsf{T}\mathbf{y} + (\mathbf{k} \otimes \mathbf{x}^t)\right) \\ &= \mathbf{M}_\text{I}\left(\alpha\mathbf{P}_0\mathbf{y} + (1-\alpha)(\mathbf{k} \otimes \mathbf{x}^t)\right) + \mathbf{M}_\text{E}\left(\mathbf{k} \otimes \mathbf{x}^t\right)\end{aligned} \tag{15}$$

with $\alpha = \lambda/(\lambda + \gamma)$, $\mathbf{P}_0 = \mathbf{C}^\mathsf{T}$, and "masking" matrices $\mathbf{M}_\text{I}$ and $\mathbf{M}_\text{E}$ for interior and exterior (*i.e.* boundary) pixels,

[3] Often called padding with "replicate" or "edge" mode.

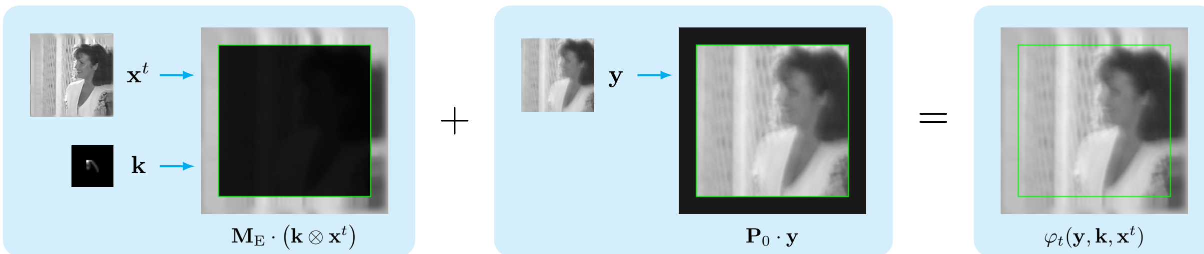


Figure 2: **Boundary adjustment approach** of Eq. (17) for $t > 0$. The current best estimate $\mathbf{x}^t$, the blurred image $\mathbf{y}$ and the blur kernel $\mathbf{k}$ are used to construct two images *(middle)*, which are combined to give the boundary-adjusted observation *(right)*. The circular blur model behind FFT-based deconvolution methods assumes knowledge of the circularly blurred boundary regions of the true image $\mathbf{x}$. Since these are not available, we instead employ our current best estimate $\mathbf{x}^t$ as a proxy for $\mathbf{x}$ and artificially blur its boundaries to be used instead. This allows us to better adhere to the circular blur model and as a consequence obtain image deconvolution results of higher quality.

respectively. The diagonal matrix $\mathbf{M}_\mathrm{I} = \mathbf{P}_0\mathbf{C} \in \mathbb{R}^{n \times n}$ has entries of 1 for all interior pixels and zeros otherwise; multiplication with $\mathbf{M}_\mathrm{E} = \mathbf{I} - \mathbf{M}_\mathrm{I}$ conversely retains all exterior pixels, while setting all interior pixels to zero (*cf.* Fig. 2). Note that multiplication with $\mathbf{P}_0$ performs 0-padding, *i.e.* $\mathbf{M}_\mathrm{I}\mathbf{v}$ can be understood as first cropping the boundary of $\mathbf{v}$ with subsequent padding of zeros.

While Eq. (15) may look complicated at first glance, its role can be understood quite easily. First, if we compare Eq. (8), which does not use any boundary adjustment, to Eq. (14), we find that $\mathbf{y}$ has been replaced by $\varphi(\mathbf{y}, \mathbf{k}, \mathbf{x}^t)$. Hence, we can interpret Eq. (15) as padding $\mathbf{y}$ with the boundary of $\mathbf{k} \otimes \mathbf{x}^t$ (a circularly blurred copy of the current estimate of the deconvolved image); furthermore, the interior of $\mathbf{y}$ is replaced by a convex combination of $\mathbf{y}$ and $\mathbf{k} \otimes \mathbf{x}^t$ based on weight $\alpha \in [0, 1]$.

Intuitively, it is very sensible to pad $\mathbf{y}$ with the boundary of $\mathbf{k} \otimes \mathbf{x}^t$, but only if $\mathbf{x}^t$ is already similar to the true image $\mathbf{x}$. By doing this, we essentially modify $\mathbf{y}$ to better adhere to the circular blur assumption. However, replacing the boundary in such a way does not seem to be a good idea for $t = 0$, since $\mathbf{x}^0$ is usually initialized as the edgetapered blurred image $\widetilde{\mathbf{y}}$ (Eq. 11) and thus is typically very dissimilar to the true image $\mathbf{x}$. Consequently, we adopt this boundary modification approach only for $t > 0$.

For $t = 0$, we use the standard edgetapered image $\widetilde{\mathbf{y}}$. Furthermore, we choose not to use a convex combination of $\mathbf{k} \otimes \mathbf{x}^t$ and $\mathbf{y}$ for the interior pixels; since the blur model of Eq. (1) is actually valid for the interior pixels, we simply use $\mathbf{y}$ as is. Overall, this leads us to modify our previously chosen update equation (Eq. 10) to arrive at our final model

$$\mathbf{x}^{t+1} = \mathcal{F}^{-1}\left(\frac{\mathcal{F}\big(\mathbf{k} \circledast \varphi_t(\mathbf{y}, \mathbf{k}, \mathbf{x}^t) + \frac{1}{\omega_t(\lambda)}\phi_t^{\mathrm{CNN}}(\mathbf{x}^t)\big)}{|\mathcal{F}(\mathbf{k})|^2 + \frac{1}{\omega_t(\lambda)}\sum_i |\mathcal{F}(\mathbf{f}_{it})|^2}\right) \quad (16)$$

with the boundary adjustment function

$$\varphi_t(\mathbf{y}, \mathbf{k}, \mathbf{x}^t) = \begin{cases} \widetilde{\mathbf{y}} & \text{if } t = 0 \\ \mathbf{P}_0\mathbf{y} + \mathbf{M}_\mathrm{E}\big(\mathbf{k} \otimes \mathbf{x}^t\big) & \text{if } t > 0, \end{cases} \quad (17)$$

which we visualize in Fig. 2. Our boundary strategy of Eq. (17) is very simple, yet effective (*cf.* Section 4.3), does not add any parameters or increase the computational burden, and can easily be applied to existing FFT-based deconvolution methods.

4. Experiments

In the following we conduct experiments with our proposed model for the task of non-blind image deconvolution. We compare our results to the state-of-the-art on two popular datasets, both in terms of average *peak signal-to-noise ratio* (PSNR) and test runtime. Additional experiments show the effectiveness of our boundary strategy (Section 4.3) as compared to the common edgetapering (Eq. 11).

Our implementation[4] is based on Keras [6] and TensorFlow [1], allowing us to make use of built-in GPU acceleration and automatic differentiation for all model parameters.

4.1. Model configuration and training

For all experiments, we parameterize ϕ_t^{CNN} (*cf.* Fig. 1) with a common CNN architecture of six sequential convolutional layers with $3{\times}3$ kernels. While the first five layers each have 32 feature channels followed by ELU [7] activations, the final layer outputs a single channel and does not use a non-linear activation function. We choose a small multilayer perceptron to specify $\omega_t(\lambda)$, using 3 hidden layers of 16 neurons each (with ELU activations); the final output neuron goes through a softplus activation function to ensure positivity. Finally, at each step t we use 24 linear filters $\mathbf{f}_{it}$ of size $5{\times}5$ pixels for the denominator of Eq. (16).

Our full model consists of several identically structured *stages* as defined in Eq. (16), each taking as input the prediction made by its predecessor. Since each stage hinges on the (fast) Fourier transform and all stages together form a single deep network, we call our model *Fourier Deconvolution Network* (FDN). Following [5, 21], who report their best

[4] Code is available on our webpages.

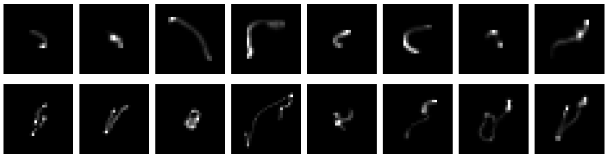


Figure 3: **Top row.** Examples of simulated blur kernels from [20], which we use for model training. **Bottom row.** The 8 blur kernels used in the benchmark datasets [15, 24].

results with a *greedy* training scheme, we first train each successive stage individually. Since each stage is differentiable, we also investigate to jointly *finetune* the parameters of all stages in an end-to-end fashion. We apply Adam [11] to minimize negative PSNR as our objective function.

We use grayscale images from the Berkeley segmentation dataset [3] to train our model, specifically by extracting random patches that we then synthetically blur with a randomly selected blur kernel. To that end, we make use of simulated kernels taken from [20] (see top row of Fig. 3 for some examples). We add Gaussian noise to the blurred image and subsequently use 8-bit quantization for all pixels.

For all experiments, we train on 3000 random image patches $\mathbf{x}$, which are blurred with kernels $\mathbf{k}$ of sizes up to 37×37 to yield blurred images $\mathbf{y}$ of 284×284 pixels each.

4.2. Evaluation

We evaluate our model on two well-known benchmark datasets. The one compiled by Levin *et al.* [15] consists of four 255×255 grayscale images, each *optically* blurred with a set of eight real-world blur kernels to yield 32 images in total. The eight kernels are shown in the bottom row of Fig. 3. The standard deviation σ of Gaussian noise on these blurred images is commonly stated as 1% of the dynamic range [*e.g.*, 23, 24], *i.e.* $\sigma = 2.55$ for typical images with pixel values $0 \ldots 255$. However, we found this to be inaccurate and empirically determined σ to be closer to 1.5.

Sun *et al.* [24] use the same eight blur kernels as Levin *et al.*, but apply each of them to *synthetically* blur 80 higher resolution images (long side scaled to 1024 pixels), yielding a benchmark dataset of 640 images in total. Finally, 1% Gaussian noise (*i.e.*, $\sigma = 2.55$) is added to each image before 8-bit quantization.

Instead of following the common practice [*e.g.*, 20, 21, 23] of training a specialized model for each noise level σ (here, 1.5 and 2.55), we instead learn a more versatile model from training data with various amounts of noise. Specifically, we create our training images by adding noise with σ uniformly chosen at random from the interval $[1.0, 3.0]$, which allows us to train a single model that yields excellent results on both benchmark datasets. Please see the supplemental material for results that were obtained from models trained for either a single σ or wider range of noise levels.

Method	σ_{train}	Levin [15]	Sun [24]
$\text{FDN}^{10}_{\text{G}}$ (ours)	$[1.0, 3.0]$	34.98 (1.5)	32.62 (2.55)
$\text{FDN}^{10}_{\text{T}}$ (ours)	$[1.0, 3.0]$	**35.09** (1.5)	**32.67** (2.55)
$\text{CSF}^{3}_{\text{pw.}}$ [21]	0.5	33.48[1] (0.5)	
$\text{CSF}^{5}_{5\times5}$	1.5	34.06 (1.5)	
(trained by us)	2.55		32.21 (2.55)
EPLL [31]	–	34.75 (1.5)	32.46[2] (2.55)
RTF [20]	0.5	33.97[3] (0.5)	
	2.55		32.49 (2.55)
Levin [14]	–	33.82[2] (?)	

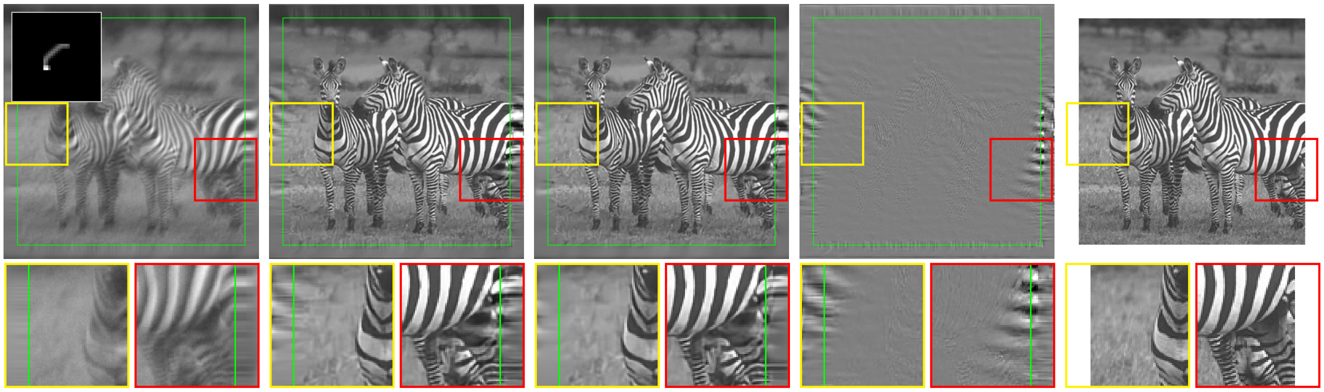
Table 1: **Results for non-blind deblurring benchmarks.** Average PSNR for two well-known deblurring benchmarks [15, 24], where each method uses the ground truth blur kernels. The second column denotes the noise level that the respective method was trained for, whereas the small numbers in parentheses in columns 3 and 4 denote the noise level assumed or given as input at test time. The upper part of the table shows efficient FFT-based methods, while methods in the lower part have higher computational cost. Scores marked with [1], [2] and [3] quoted from [21], [24] and [20], respectively; others computed with publicly available code.

As mentioned above, we consider two training variants: First, we greedily train our FDN model with 10 stages, which we abbreviate as $\text{FDN}^{10}_{\text{G}}$. Second, we use the parameters from $\text{FDN}^{10}_{\text{G}}$ as initialization to jointly finetune all stages and denote the resulting model as $\text{FDN}^{10}_{\text{T}}$. We apply our two models to both benchmark datasets. Note that we strictly adhere to the evaluation protocol of the respective dataset to ensure a fair comparison, which includes discarding regions close to the border of each image.

Table 1 shows the results of our models compared to other state-of-the-art methods on both datasets. We outperform our strongest competitors (EPLL [31] and RTF [20], respectively) by around $0.2 - 0.3\,\text{dB}$. Please see Fig. 7 for a qualitative comparison. While this performance improvement may not seem very large, it is important to note that our approach is orders of magnitude faster than both EPLL and RTF (Section 4.4), neither of which can use efficient FFT-based inference.

While the FFT-based deconvolution method CSF [21] has similar computational cost as our approach, we do outperform it on the benchmarks of Sun *et al.* and Levin *et al.* by large margins of around 0.5 and $1\,\text{dB}$, respectively. Note that our improvements are already compared to more powerful CSF models[5] that we trained on datasets of the same size as ours. One major reason for the inferior perfor-

[5]We use 24 filters of 5×5 pixels, since this most closely resembles our FDN model. A simpler pairwise CSF as used in [21] performed much worse in our tests. CSF results are already saturated after 5 stages.



(a) Edgetapered input (b) Traditional edgetapering (c) Our method (d) Difference of (b) and (c) (e) Ground truth

Figure 4: **Example result of the proposed boundary adjustment method.** Output of our greedy ten-stage model with our proposed boundary adjustment method *(c)* compared to just using traditional edgetapering *(b)*. The boundary region outside the green inner square is discarded for the final output. While most of the changes are close to the image border, the difference image *(d)* shows that our boundary approach also has an effect on details within the image.

mance of CSF is due to its use of edgetapering after each stage, which is clearly inferior to our boundary adjustment strategy (Section 4.3). In particular, we find that the performance of our $\text{FDN}_{\text{G}}^{10}$ model would deteriorate by a substantial $0.74\,\text{dB}$ on the benchmark of Levin *et al*. if we used the same boundary approach as CSF does.

4.3. Boundary adjustment comparison

We compare our proposed boundary adjustment (BA) strategy (*Our BA*, *cf.* Eq. 17 and Fig. 2) to the traditional edgetapering method (*ET once*, *cf.* Eq. 11); Fig. 4 provides an illustration for an example image. Since the CSF model [21] additionally crops its current estimate of the restored image after each stage and re-applies edgetapering to it (*ET each*), we also compare against this BA method.

Furthermore, we not only compare these BA strategies within our FDN model, but also apply them to the CSF model and a standard Wiener filter. To that end, we train separate variants of each model that only differ in their BA strategy, but are otherwise trained in exactly the same way.

The results of our evaluation on the benchmark of Levin *et al*. [15] are shown in Fig. 5; more details can be found in the supplemental material. First, we find that our BA strategy is always superior to using edgetapering, which also demonstrates the applicability to other FFT-based deconvolution methods. Especially remarkable is that we can boost the performance of a Wiener filter by over $1\,\text{dB}$ when we apply it iteratively with our BA method. Second, the results allow us to better analyze the respective contributions from the CNN-based regularization on one hand, and our BA strategy on the other hand. Using *ET each*, after 5 stages we only see a modest improvement of $0.16\,\text{dB}$ with our FDN model over CSF. However, we see a boost twice as large ($0.32\,\text{dB}$) when using *Our BA*, which suggests that

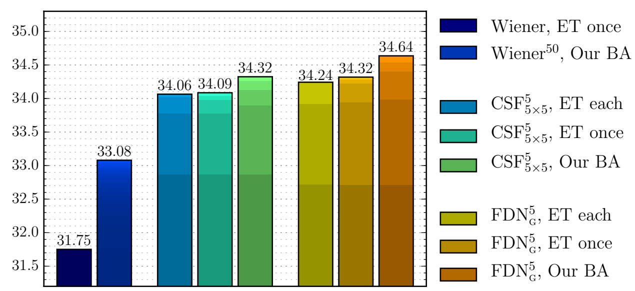


Figure 5: **Comparison of boundary adjustment methods.** Average PSNR (in dB) on the benchmark of Levin *et al*. with different boundary adjustment (BA) strategies. Vertical partitions of bars correspond to respective model stages. Our BA method shows a clear improvement over edgetapering (ET). See text for details.

our BA approach is actually important to exploit our more flexible CNN-based regularization. Third, we find that the performance of FDN does not improve further after stage 3 with *ET each*; this does not apply to our BA, which enables FDN to increase the PSNR by $0.58\,\text{dB}$ within stages $4-10$[6].

4.4. Runtime

As mentioned before, when it comes to runtime, we find it useful to distinguish between deconvolution methods that admit efficient FFT-based inference on one hand, and much more computationally demanding methods, such as EPLL [31] and RTF [20], on the other hand. Furthermore, FFT-based methods employ closed-form update steps and thus offer predictable runtime. In contrast, slower methods typically need to use iterative equation system solvers, whose runtime may vary significantly based on the given image and blur kernel.

[6] Stages $6-10$ not shown in Fig. 5 for fair comparison to 5-stage CSF.

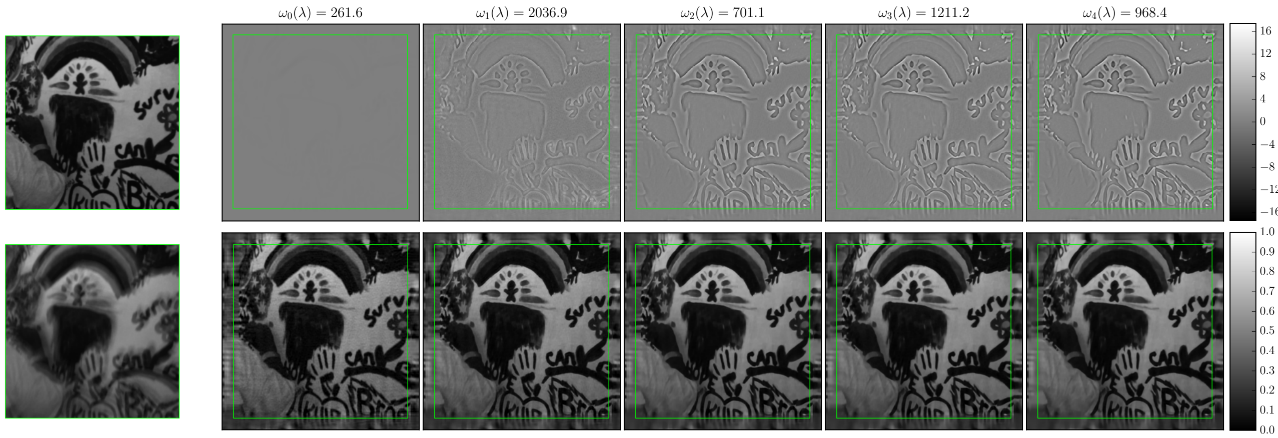


Figure 6: **CNN outputs with associated weights (top) and model predictions (bottom) for first 5 stages of FDN_G^{10}.** From left to right, we show the CNN outputs $\phi_t^{\text{CNN}}(\mathbf{x}^t)$ with their associated noise-specific weights $\omega_t(\lambda)$ and the predictions $\mathbf{x}^{t+1}$ with $t \in \{0 \ldots 4\}$ for an example image (bottom far left, associated ground truth shown above; $\lambda = 1/1.5^2$). The green lines delineate the padded boundary region.

(a) EPLL (26.75 dB) (b) RTF (26.91 dB) (c) FDN_T^{10} (27.21 dB) (d) Ground truth

Figure 7: **Comparison of deconvolution results.** Result of our finetuned ten-stage model *(c)* compared to two state-of-the-art methods, EPLL *(a)* and RTF *(b)*, for a challenging image from the Sun *et al*. benchmark. Our FDN model is able to restore fine details even in highly textured image regions. Images are best viewed magnified on a screen.

While the specific runtime of a method depends on software implementation and computing hardware, we find it instructive to give ballpark numbers for some of the methods shown in Table 1. Our ten-stage model takes around 0.15 seconds for the small images from the dataset of Levin *et al*. (255×255 pixels), and roughly 0.75 seconds for the somewhat larger images from Sun *et al*. (less than 1 megapixel). These numbers are based on our TensorFlow implementation with an NVIDIA Titan X GPU.

Whereas CSF should have similar or slightly lower runtime compared to our method, EPLL and RTF are orders of magnitude slower. Based on their public CPU-based implementations, we find that they take around 1 minute for the small images from Levin *et al*., and in the order of $5-10$ minutes for the bigger Sun *et al*. images. While it is not entirely fair to compare such numbers to our GPU-based runtimes, it is evident that these slower methods are not practical for large images of potentially many megapixels.

5. Conclusion

We generalized efficient FFT-based deconvolution methods, specifically *shrinkage fields*, by introducing a CNN at each stage to provide more powerful regularization. Our model keeps all the benefits of fast FFT-based inference and discriminative end-to-end training, yet is shown to outperform much less efficient state-of-the-art methods on two non-blind deblurring benchmarks. We also proposed a simple, yet effective, scheme to better cope with the effects of the circular boundary assumption imposed by FFT-based inference. This method is generic, free of parameters, and shown to improve restoration results at essentially no extra cost. There are various avenues for future work, including the extension to blind deconvolution and non-uniform blur.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensor-

Flow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation. Savannah, Georgia, USA*, 2016.

[2] M. S. C. Almeida and M. A. T. Figueiredo. Deconvolving images with unknown boundaries using the alternating direction method of multipliers. *IEEE Transactions on Image Processing*, 22(8):3074–3086, 2013.

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

[4] A. Chakrabarti. A neural approach to blind motion deblurring. In *Proceedings of 14th European Conference on Computer Vision*, Lecture Notes in Computer Science, pages 221–235. Springer, 2016.

[5] Y. Chen, W. Yu, and T. Pock. On learning optimized reaction diffusion processes for effective image restoration. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Boston, Massachusetts, June 2015.

[6] F. Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[7] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv:1511.07289*, 2015.

[8] P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, editors, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, Springer Optimization and Its Applications, pages 185–212. Springer, 2011.

[9] S. Diamond, V. Sitzmann, S. P. Boyd, G. Wetzstein, and F. Heide. Dirty pixels: Optimizing image classification architectures for raw sensor data. *arXiv:1701.06487*, 2017.

[10] D. Geman and C. Yang. Nonlinear image recovery with half-quadratic regularization. *IEEE Transactions on Image Processing*, 4(7):932–946, July 1995.

[11] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[12] R. Köhler, M. Hirsch, B. Mohler, B. Schölkopf, and S. Harmeling. Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *Proceedings of the 12th European Conference on Computer Vision*, volume 7578 of *Lecture Notes in Computer Science*. Springer, 2012.

[13] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-Laplacian priors. In *Advances in Neural Information Processing Systems*, 2009.

[14] A. Levin, R. Fergus, F. Durand, and W. T. Freeman. Image and depth from a conventional camera with a coded aperture. *ACM Transactions on Graphics*, 26(3):70:1–70:9, July 2007.

[15] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Miami, Florida, June 2009.

[16] A. Matakos, S. Ramani, and J. A. Fessler. Accelerated edge-preserving image restoration without boundary artifacts. *IEEE Transactions on Image Processing*, 22(5):2019–2029, 2013.

[17] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

[18] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):123–231, 2013.

[19] S. J. Reeves. Fast image restoration without boundary artifacts. *IEEE Transactions on Image Processing*, 14(10):1448–1453, 2005.

[20] U. Schmidt, J. Jancsary, S. Nowozin, S. Roth, and C. Rother. Cascades of regression tree fields for image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):677–689, Apr. 2016.

[21] U. Schmidt and S. Roth. Shrinkage fields for effective image restoration. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Columbus, Ohio, June 2014.

[22] U. Schmidt, C. Rother, S. Nowozin, J. Jancsary, and S. Roth. Discriminative non-blind deblurring. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Portland, Oregon, June 2013.

[23] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. Learning to deblur. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7):1439–1451, 2016.

[24] L. Sun, S. Cho, J. Wang, and J. Hays. Edge-based blur kernel estimation using patch priors. In *Proceedings of IEEE International Conference on Computational Photography*, 2013.

[25] R. Wang and D. Tao. Recent progress in image deblurring. *arXiv:1409.6838*, 2014.

[26] Y. Wang, J. Yang, W. Yin, and Y. Zhang. A new alternating minimization algorithm for Total Variation image reconstruction. *SIAM Journal on Imaging Sciences*, 1(3):248–272, 2008.

[27] N. Wiener. *The Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. John Wiley & Sons, Inc., New York, N. Y., 1949.

[28] L. Xiao, J. Wang, W. Heidrich, and M. Hirsch. Learning high-order filters for efficient blind deconvolution of document photographs. In *Proceedings of 14th European Conference on Computer Vision*, Lecture Notes in Computer Science, pages 734–749. Springer, 2016.

[29] L. Xu and J. Jia. Two-phase kernel estimation for robust motion deblurring. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Proceedings of the 11th European Conference on Computer Vision*, volume 6311 of *Lecture Notes in Computer Science*. Springer, 2010.

[30] J. Zhang, J. Pan, W. Lai, R. Lau, and M. Yang. Learning fully convolutional networks for iterative non-blind deconvolution. *arXiv:1611.06495*, 2016.

[31] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *Proceedings of the Thirteenth IEEE International Conference on Computer Vision*, 2011.