# Learning to Deblur

Christian J. Schuler, Michael Hirsch, Stefan Harmeling, and Bernhard Schölkopf

**Abstract**—We describe a learning-based approach to blind image deconvolution. It uses a deep layered architecture, parts of which are borrowed from recent work on neural network learning, and parts of which incorporate computations that are specific to image deconvolution. The system is trained end-to-end on a set of artificially generated training examples, enabling competitive performance in blind deconvolution, both with respect to quality and runtime.

**Index Terms**—Sharpening and deblurring, neural networks, machine learning

## 1 Introduction

Blind image deconvolution is the task of recovering the underlying sharp image from a recorded image corrupted by blur and noise. Examples of such distortions are manifold: In photography, long exposure times might result in camera shake, often in combination with image noise caused by low light conditions. Lens aberrations can create blur across images in particular for wide apertures. In astronomy and remote sensing, atmospheric turbulence blurs images. All these image reconstruction tasks are examples of inverse problems, which are particularly hard to solve since not only the underlying image is unknown, but also the blur. We assume that the blurred image $\mathbf{y}$ is generated by linearly transforming the underlying image $\mathbf{x}$ (sometimes called the "true image" or "latent image") by a convolution (denoted by $*$), and additive noise $\mathbf{n}$,

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

The task of blind image deconvolution is to recover $\mathbf{x}$ given only the blurry image $\mathbf{y}$, without knowing $\mathbf{k}$. A number of approaches have recently been proposed to recover the true image from blurred photographs, e.g., [1], [2], [3]. Usually these methods assume some sparsity inducing image prior for $\mathbf{x}$, and follow an iterative, multi-scale estimation scheme, alternating between blur and latent image estimation.

The idea of the proposed method is to "unroll" this reconstruction procedure and pose it as a nonlinear regression problem, where the optimal parameters are learned from artificially generated data. As a function approximator, we use a layered architecture akin to a deep neural network (NN) or multilayer perceptron. Some of the layers are convolutional, as popular in many recent approaches, while others are non-standard and specific to blind deconvolution. Overall, the system is inspired by [4] who formulated the idea of neural networks as general processing schemes with large numbers of parameters.

Using extensive training on a large image dataset in combination with simulated camera shakes, we train the blind deconvolution NN to automatically obtain an efficient procedure to approximate the true underlying image, given only the blurry measurement.

A common problem with NNs, and in particular with large custom built architectures, is that the devil is in the details and it can be nontrivial to get systems to function as desired. We thus put particular emphasis on describing the implementation details, and we make the code publicly available.

*Main contributions.* We show how a trainable model can be designed in order to be able to *learn* blind deconvolution. Results are comparable to the state of the art of hand-crafted approaches and go beyond when the model is allowed to specialize to a particular image category.

## 2 Related Work

Some recent approaches to blind deconvolution in photography have already been mentioned above. We refer the interested reader to [5] for an overview of the subject, and focus only on how NNs have been previously used for deconvolution.

Neural networks have been used extensively in image processing. Comprehensive reviews are [6], [7], both of which present broad overviews of applying NNs to all sorts of image processing operations, including segmentation, edge detection, pattern recognition, and nonlinear filtering.

Image deconvolution, often also called image reconstruction, has been approached with NNs for a long time. However, these approaches are quite different from our proposed work:

- *NN to identify the type of blur:* [8] and similarly [9] apply NNs to identify the type of blur from a restricted set of parametrized blurs, e.g., linear-motion, out-of-focus and Gaussian blur, and possibly their parameters.
- *NN to model blurry images:* [10] models the blurry image as the result of a neural network where at the different layers the blur kernel and the true image appear.
- *NN to inversely filter blurry images:* [11] learns an inverse filter represented by a NN to deblur text.

- *C.J. Schuler, M. Hirsch, and B. Schölkopf are with the MPI for Intelligent Systems, Spemannstr. 38, Tuebingen 72076, Germany. E-mail: {cschuler, mhirsch, bs}@tuebingen.mpg.de.*
- *S. Harmeling is with the MPI for Intelligent Systems, Spemannstr. 38, Tuebingen 72076, Germany and the University of Düsseldorf. E-mail: harmeling@tuebingen.mpg.de.*

*Manuscript received 29 July 2014; revised 3 Apr. 2015; accepted 20 July 2015. Date of publication 22 Sept. 2015; date of current version 10 June 2016. Recommended for acceptance by T. Brox. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TPAMI.2015.2481418*

- *NN to optimize a regularized least squares objective:* [12] proposes also the common two-stage procedure to first estimate the blur kernel and then to recover the image. For both tasks Hopfield networks are employed to solve the optimization problems.
- *NN to remove a known blur:* [13] and [14] both present methods for non-blind deblurring based on deep neural networks. The first approach starts with a straight-forward division in Fourier space as a preprocessing step, while the second method includes this idea in its learning pipeline.
- *NN to remove a parametrized blur:* the method of [15] comes closest to ours: it also uses a deep neural network to infer a blur. However, it is limited to inferring linear blurs parametrized by length and angle.

Other learning-based approaches for blind deconvolution try to learn the deconvolution solution for a single image, in particular they attempt to learn an appropriate sparse representation for a given blurry image, e.g., [16]. In our work, we follow a different strategy: instead of learning the solution or part of a solution for a single fixed image, we use a neural network to learn a general procedure that is directly applicable to other images and to different blurs. Also close to our approach is the work of [17], who train a deblurring procedure based on regression tree fields. However, even though their approach is not limited to a specific blur or a specific image, they consider only the problem of *non-blind* deblurring, i.e. their method assumes that the blur kernel is known. This is in contrast to our contribution, which demonstrates how to train a NN to solve the blind deconvolution problem. This is a much harder problem, since not only do we have to solve an underdetermined linear problem (originating from the least-squares formulation of non-blind deconvolution), but an underdetermined bilinear problem, which appears since the entries of the unknown blur kernel $\mathbf{k}$ and the pixel values of the unknown image $\mathbf{x}$ are multiplied by the convolution, see Eq. (1).

Finally we note that deconvolutional networks, introduced in [18] and further extended in [19], are not architectures for image deconvolution. Instead, they use convolutions to link sparse features to given images. Their goals are good image representations for tasks such as object recognition and image denoising.

## 3 Blind Deconvolution as a Layered Network

Existing fast blind deconvolution methods work by alternating between the following three steps [2], [3], [20]:

1) *Feature extraction* computes image representations that are useful for kernel estimation. These representations may simply be the gradient of the blurry image and the gradient of the current estimate, possibly thresholded in flat regions [3]; they may also be preliminary estimates of the sharp image, for instance computed by heuristic nonlinear filtering [2].
2) *Kernel estimation* uses the extracted features to estimate the convolution kernel.
3) *Image estimation* attempts to compute an approximation of the sharp image using the current kernel estimate.

We will represent these steps by a trainable deep neural network, thus adding more flexibility to them and allowing them to optimally adapt to the problem. The layers of the network alternate between (1) a local convolutional estimation to extract good features, (2) the estimation of the blur kernel, globally combining the extracted features, and finally (3) the estimation of the sharp image. Parts (2) and (3) are fixed (having only one hyper-parameter for regularization). The free parameters of the network appear in part (1), the feature extraction module. Thus, instead of having to learn a model on the full dimensionality of the input image, which would not be doable using realistic training set sizes, the learning problem is naturally reduced to learning filters with a limited receptive field.

### 3.1 Architecture Layout

Below, we describe the different parts of the network, which are also illustrated in Fig. 1.

#### 3.1.1 Feature Extraction Module

The first module is intended to only extract features of the image, and does not directly predict the original sharp image. To make an information-optimal removal of the blur kernel from the image, knowledge about the kernel has to be combined from all parts of the image, which in turn would require connectivity with all pixels of the input layer, and in consequence a huge parameter space that would impede convergence.

Instead, we introduce the feature extraction module to extract the relevant information locally, which will be combined globally to a kernel prediction by the kernel estimation module. Therefore, connectivity of the feature extraction module can be limited, which greatly reduces the parameters that have to be learned.

What makes a good feature for kernel estimation can reasonably be assumed to be a translation invariant problem, i.e., independent from the position within the image. We therefore model the feature extractors using shared weights applying across all image locations, i.e., as a convolutional NN layer,[1] creating several feature representations of the image.

This is followed by two layers that introduce nonlinearity into the feature generation. First, every value is transformed by a tanh-unit, then the feature representations are pixel-wise linearly combined to new hidden images, formally speaking

$$\begin{aligned} \tilde{\mathbf{y}}_i &= \sum_j \alpha_{ij} \tanh(\mathbf{f}_j * \mathbf{y}) \text{ and} \\ \tilde{\mathbf{x}}_i &= \sum_j \beta_{ij} \tanh(\mathbf{f}_j * \mathbf{y}), \end{aligned} \tag{2}$$

where $\mathbf{f}_j$ are the filters of the convolution layer (shared for $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{y}}_i$), the function $\tanh$ operates coordinate-wise, and $\alpha_{ij}$ and $\beta_{ij}$ are the coefficients to linearly combine the hidden representations. Note that we usually extract several gradient-like images $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{y}}_i$. Depending on the desired

1. Note that this convolution has nothing to do with the convolution appearing in our image formation model (Eq. (1))—causally, it goes in the opposite direction, representing one step in the inverse process turning the blurry image into an estimate of the underlying image.

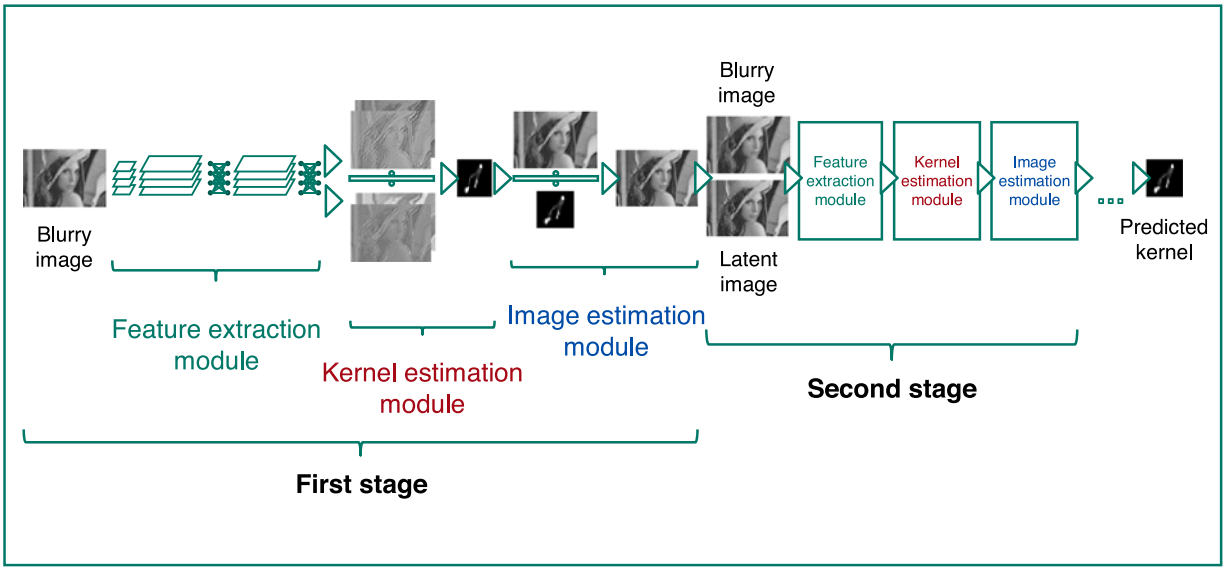



Fig. 1. Architecture of our proposed blind deblurring network. First the feature extraction module transforms the image to a learned gradient-like representation suitable for kernel estimation. Next, the kernel is estimated by division in Fourier space, then similarly the latent image. The next stages, each consisting of these three operations, operate on both the blurry image and the latent image.

nonlinearity, these two layers can be stacked multiple times, leading to the final image representations based on features useful for kernel estimation. The feature extraction module is the most crucial ingredient in our NN. Therefore the entire following Section 4 is devoted to provide a thorough analysis and understanding of its inherent mechanism and functioning.

#### 3.1.2 Kernel Estimation Module

The next module collects the local estimation to a single global estimation of the convolution kernel.

Given $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{y}}_i$ which contain features tuned for optimal kernel estimation, the kernel $\tilde{\mathbf{k}}$ can be estimated by minimizing

$$\sum_i \|\tilde{\mathbf{k}} * \tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_i\|^2 + \beta_k \|\tilde{\mathbf{k}}\|^2 \tag{3}$$

for $\tilde{\mathbf{k}}$ given the results from the previous step $\tilde{\mathbf{x}}_i$ and their blurry counterparts $\tilde{\mathbf{y}}_i$. Assuming no noise and a kernel without zeros in its power spectrum, the true gradients of the sharp image $\mathbf{x}$ and its blurred version $\mathbf{y}$ would return the true kernel for $\beta_k = 0$. Typically, in existing methods $\tilde{\mathbf{y}}_i$ are just the gradients of the blurry image, while here these can also be learned representations predicted from the previous layer. The minimization problem can be solved in one step in Fourier space if we assume circular boundary conditions [2]:

$$\tilde{\mathbf{k}} = F^{\mathsf{H}} \frac{\sum_i \overline{F\tilde{\mathbf{x}}_i} \odot F\tilde{\mathbf{y}}_i}{\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k}. \tag{4}$$

Here $F$ is the discrete Fourier transform matrix, $\odot$ the Hadamard product, $\overline{\mathbf{v}}$ the complex conjugate of a vector $\mathbf{v}$, and the division and absolute value are performed element-wise. The one step solution is only possible because we use a simple Gaussian prior on the kernel. We call this step the *quotient layer*, which is an uncommon computation in NNs that usually only combine linear layers and nonlinear thresholding units. The final kernel is returned by cropping to the particular kernel size and thresholding negative values. To reduce artifacts from the incorrect assumption of circular boundary conditions, the borders of the image representations are weighted with a Barthann window such that they smoothly fade to zero.

Due to varying size of the input image, we set $\beta_k = 10^{-4}$ for numerical stability only. This forces the network to not rely on the prior, which would lose importance for a larger input image relative to the likelihood term (since the kernel size is fixed).

#### 3.1.3 Image Estimation Module

Before adding another feature extraction module, the estimated kernel is used to obtain an update of the sharp latent image: analogously to Eq. (3), we solve

$$\|\tilde{\mathbf{k}} * \tilde{\mathbf{x}} - \mathbf{y}\|^2 + \beta_x \|\tilde{\mathbf{x}}\|^2 \tag{5}$$

for $\tilde{\mathbf{x}}$, which can also be performed with a quotient layer. This can be done in one step (which would not be possible when using a sparse prior on $\tilde{\mathbf{x}}$). The following convolution layer then has access to both the latent image and the blurry image, which are stacked along the third dimension (meaning that the learned filters are also three-dimensional). The hyper-parameter $\beta_x$ is also learned during training.

### 3.2 Iterations as Stacked Networks

As illustrated in Fig. 1, the feature extraction module, kernel estimation module and the image estimation module can be stacked several times, corresponding to multiple iterations in non-learned blind deconvolution methods. This leads to a single NN that can be trained end-to-end with back-propagation by taking the derivatives of all steps (see supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2015.2481418, for the derivatives of the solutions to Eq. (3) and the analogous Eq. (5)), increasing

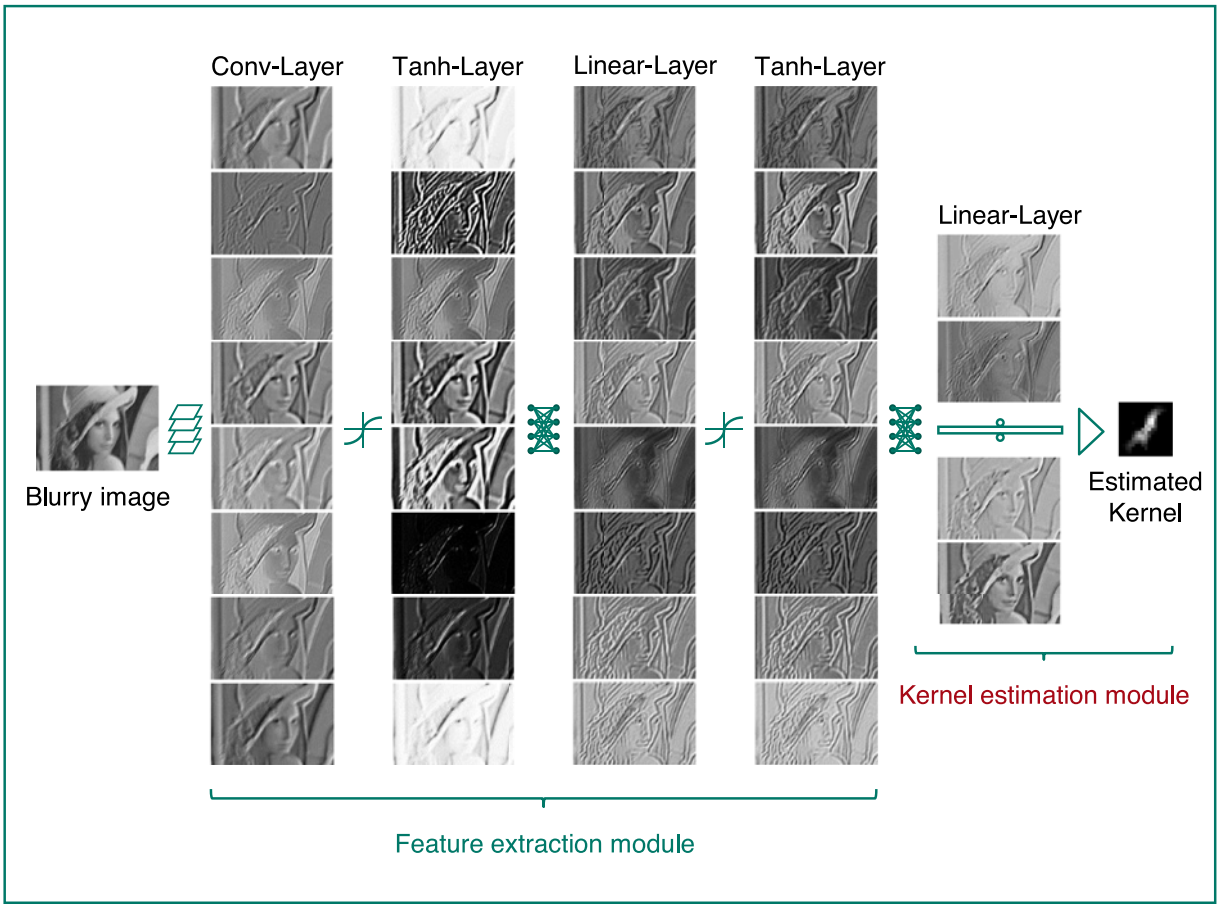



Fig. 2. Intermediary outputs of the first stage (see Fig. 1) of a single-stage NN with the following architecture: eight convolution filters; tanh nonlinearity; linear recombination to eight images; tanh nonlinearity; linear recombination to two sharp and two blurry feature images; kernel estimation.

the performance as shown in Fig. 3 (but at the same time increasing runtime).

Similar to other blind deconvolution approaches, we also use a multi-scale procedure. The first (and coarsest) scale is just a network as described above, trained for a particular blur kernel size. For the second scale, the previous scale network is applied to a downsampled version of the blurry image, and its estimated latent image is upsampled again to the second scale. The second scale network then takes both the blurry image and the result of the previous scale as input. We repeat these steps until the final scale can treat the desired blur kernel size on the full resolution image.

### 3.3 Training

To train the network, we generate pairs of sharp and blurry images. The sharp image is sampled from a subset of about

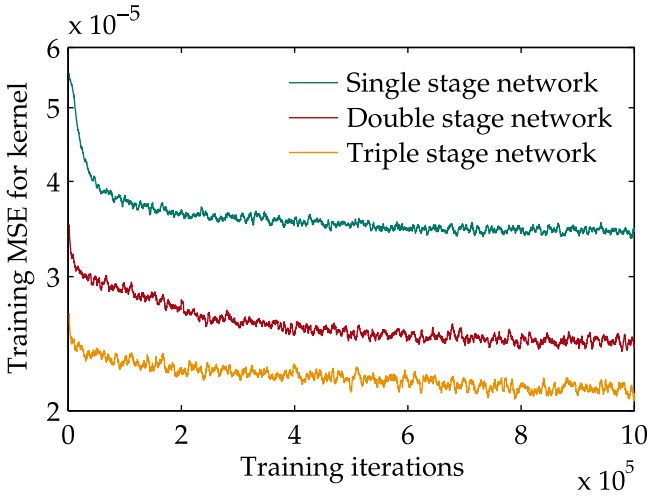



Fig. 3. Deeper networks are better at kernel prediction. One stage for kernel prediction consists of a convolutional layer, two hidden layers and a kernel estimation module.

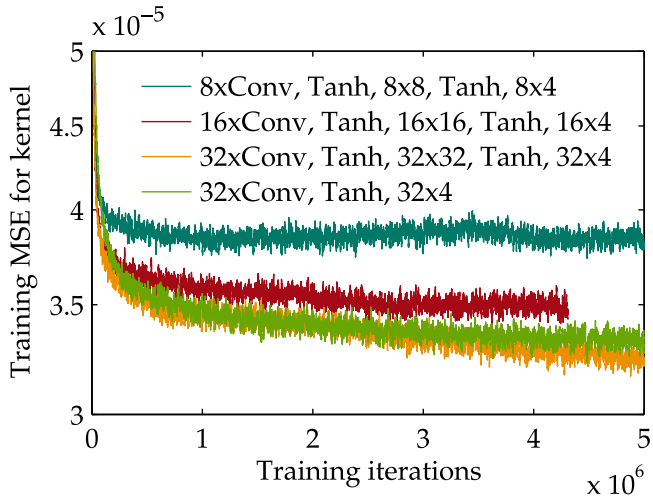



Fig. 4. The performance of the network for kernel estimation depends on the architecture. More filters in the convolutional layer and more hidden layers are better.

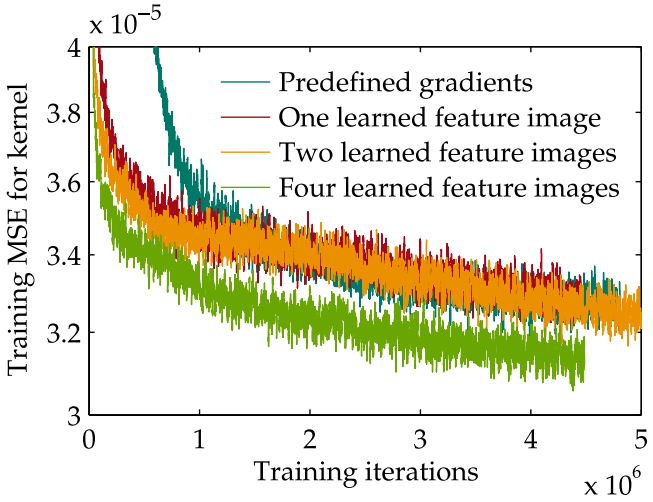



Fig. 5. The performance of the network depends on the number of the predicted gradient-like images used in the kernel estimation module. Predefining $\tilde{\mathbf{y}}_i$ to x- and y-gradients and not learning these representations slows down training.

1.6 million images of the ImageNet [21] dataset, randomly cropped to a size of $256 \times 256$. Next, a blur trajectory is generated by sampling both its x- and y-coordinates separately from a Gaussian Process $f_x(t), f_y(t) \sim \mathcal{GP}\big(0, k(t,t')\big)$ with

$$k(t,t') = \sigma_f^2 \left(1 + \frac{\sqrt{5}|t-t'|}{l} + \frac{5(t-t')^2}{3l^2}\right) \cdot \exp\left(-\frac{\sqrt{5}|t-t'|}{l}\right), \tag{6}$$

which is a Matérn covariance function with $\nu = 3/2$ [22]. The length scale $l$ is set to $0.3$, the signal standard deviation $\sigma_f$ to $1/4$. The coordinates are then scaled to a fixed blur kernel size and the final kernel is shifted to have its center of mass in the middle. This simple procedure generates realistic looking blur kernels, examples for both small and large kernel sizes are illustrated in Fig. 6. For every setting, we generate 1 million noise-free training examples, and add Gaussian noise during training (by default, $\sigma = 0.01$).

To avoid that the training process is disturbed by examples where the blur kernel cannot be estimated, e.g., a homogeneous area of sky without any gradient information, we reject examples where less than 6 percent pixels have gradients in x- and y-direction with an absolute value 0.05 or above.

As described in the previous subsection, a network for a certain blur kernel, i.e., a particular scale, consists of several stages, each iterating between the feature extraction, kernel estimation and the image estimation module.

We use pre-training for our network: we start by training only one stage minimizing the L2 error between the estimated and the ground truth kernel, then add a second stage after about 1 million training steps. For the next 1,000 steps, the parameters of the first stage stay fixed and only the parameters of the second stage are updated. After that, the network is trained end-to-end, until a potential next stage is added.

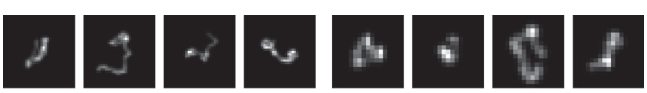


Fig. 6. Examples of blurs sampled from a Gaussian process (left: 33 px, right: 17 px).

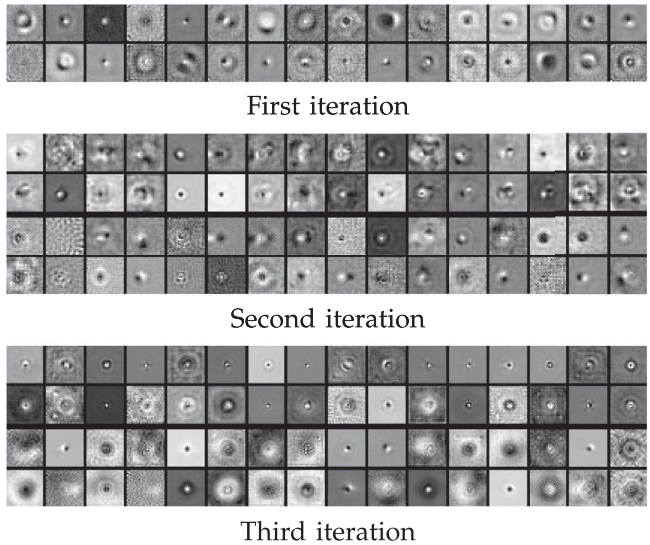



Fig. 7. Learned filters of the convolution layer for each of the three iterations within a single scale of a trained NN for kernel size $17 \times 17$ pixels. See text for details.

For the update of the parameters, convergence proved to be best with ADADELTA [23], a heuristic weighting scheme of parameter updates using gradients and updates from previous training steps. For our experiments, we choose a learning rate of 0.01 and a decay rate of 0.95. Moreover, it makes training more robust to outliers with strong gradients since it divides by the weighted root-mean-square of the seen gradients, including the current one. Responsible for the mentioned strong gradients are typically images dominated by abrupt step-edges, which create ringing artifacts in the deconvolution Eq. (5). In ImageNet these often are photos of objects with trimmed background. To make the training even more robust, we don't back-propagate examples with an error above 10 times the current average error.

## 4 Learning Features for Blur Estimation

The task of the Feature Extraction Module is to emphasize and enhance those image features that contain information about the unknown blur kernel. Fig. 7 shows the learned filters of the convolution layer for each of the three stages within a single scale of a trained NN for kernel size $17 \times 17$ pixels. While the learned filters of the first stage are reminiscent of gradient filters of various extent and orientations including Gabor and Laplace-like filters, the filters of the subsequent stages are much more intricate and more difficult to interpret.

The first stage takes a single (possibly down-sampled) version of a blurry image as input, the subsequent stages take both the restored latent image (obtained by non-blind deconvolution with the current estimate of the kernel) and the blurry image as input. The outputs of each stage are nonlinearly filtered versions of the input images. In Fig. 8 we demonstrate the effect of the first stage of a NN with two predicted output images on both the Lena image and a toy example image consisting of four disks blurred with

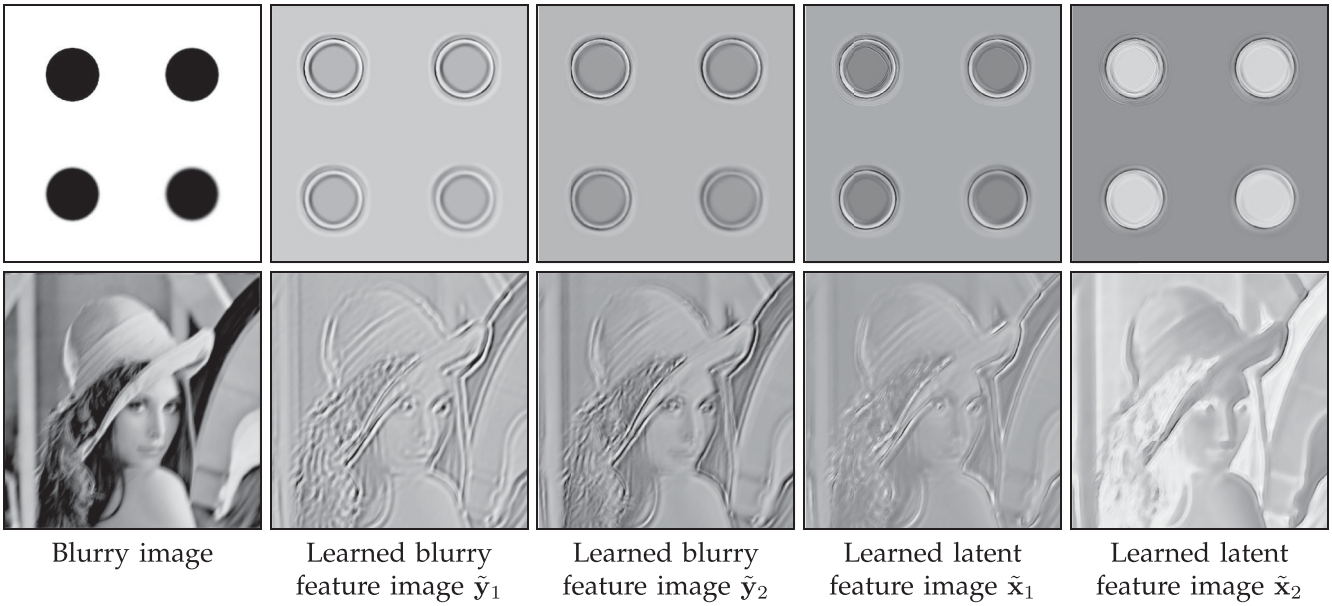



Fig. 8. Visualization of the effect of the first stage of a network with two predicted output images on toy example with disks blurred with Gaussians of varying size (top row) and motion blurred Lena image (bottom row). While for the circles in $\tilde{y}_i$ the different sizes of the Gaussian blurs are clearly visible, the NN replaces them in $\tilde{x}_i$ with shapes of comparable sharpness.

Gaussians of varying size: all latent feature images exhibit sharp edges.

Once these feature images have passed the subsequent tanh and recombination layer, they serve as input to the quotient layer, which computes an estimate of the unknown blur kernel. Fig. 2 shows the intermediary results directly after the convolution layer and how they progressively change after passing through tanh and linear recombination layer two times, which seem to emphasize strong edges.

From Fig. 9 we can understand how this is achieved. The intensity values of the intermediate output of the convolution layer, which applies only a linear transformation to its input images, exhibit a sparse distribution. After the nonlinear tanh transformation, the extreme values dominate, and sharp edges are created (see the example in Fig. 10). The subsequent layers combine these saturated images back to an image with a sparse distribution, but containing the emphasized edges.

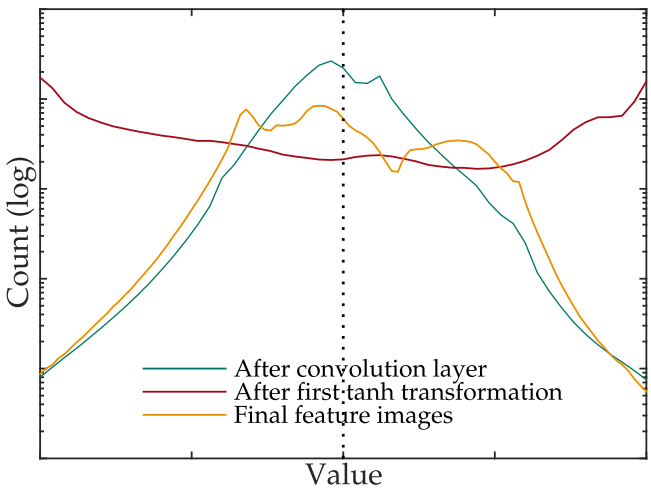



Fig. 9. Distribution of the intensity values of the feature images at different steps during their creation (averaged over 200 different images): (1) after the convolution layer (2) after the first tanh nonlinearity, where it is dominated by extreme values (3) the final blurry and sharp feature images, after recombination and a second nonlinearity.

Note that our feature extraction module outputs nonlinearly filtered images for both the blurry and the latent sharp image, both of which serve as input to the subsequent quotient layer, which in turn computes an estimate of the blur kernel. This is in contrast to other existing approaches [2], [3], which apply a *nonlinear* filter to the current estimate of the latent image, but use only a *linearly* filtered version of the blurry input image for kernel estimation. The advantage of our approach is evident from Fig. 5 where we compare the performance of trained NNs with a varying number of feature images also with the common procedure where predefined linear gradient filters are applied to the blurry input image.

It is instructive to investigate which features contribute most to the prediction performance of the network. Since it is possible to calculate the gradient of the kernel prediction error of our NNs with respect to the input, we can optimize for the ideal input (similar to "activation maximization" [24]). This process is comparable to the training of the network, but with optimization of the input instead of the

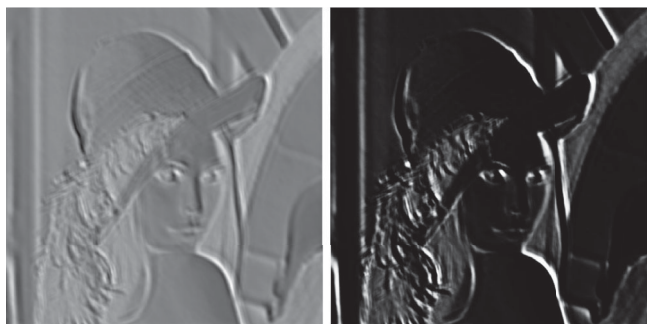


Fig. 10. Intermediate result of a feature image before (left) and after (right) the first tanh-layer. The saturation effect creates sharp edges.

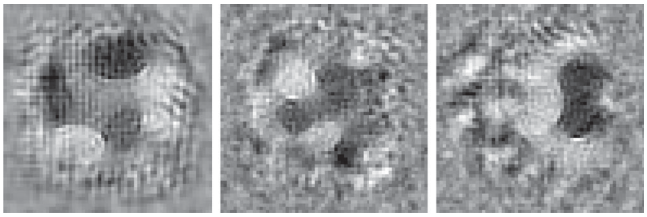


Fig. 11. Optimal input to the NN for kernel prediction found by activation maximization. All examples contain sharp edges with all orientations.

network parameters: we start from a random initialization of the input image with a size of 64 × 64 pixels, then repeatedly blur the input with a sampled kernel, add noise, calculate the prediction error, and update the input via gradient descent. These steps are repeated until convergence is achieved.

Three examples obtained according to this procedure from a trained single-stage NN for kernel size 17 × 17 are shown in Fig. 11. All examples exhibit strong edges with all possible orientations, whereas the edges encompass mostly flat areas (apart from the border regions of the input images). This is in agreement with the observations from [25] and [3] on which type of structures make good features for image deblurring.

TABLE 1
The Method Is Very Fast: Runtime in Seconds for Kernel Estimation with Varying Image Size on an Intel i5 in Matlab

| Blur size | 255 × 255 | 800 × 800 | 1,024 × 1,024 | 2,048 × 2,048 |
|---|---|---|---|---|
| 17 × 17 | 1.1 | 5.2 | 9.5 | 53.1 |
| 25 × 25 | 1.2 | 14.3 | 18.5 | 89.1 |
| 33 × 33 | 1.6 | 10.7 | 22.6 | 91.9 |

## 5 Implementation

We make the code for both training and testing our method available for download. For training, we use our

Fig. 12. Typical example images of *valley* (top row) and *blackboard* (bottom row) categories from ImageNet used for content specific training.

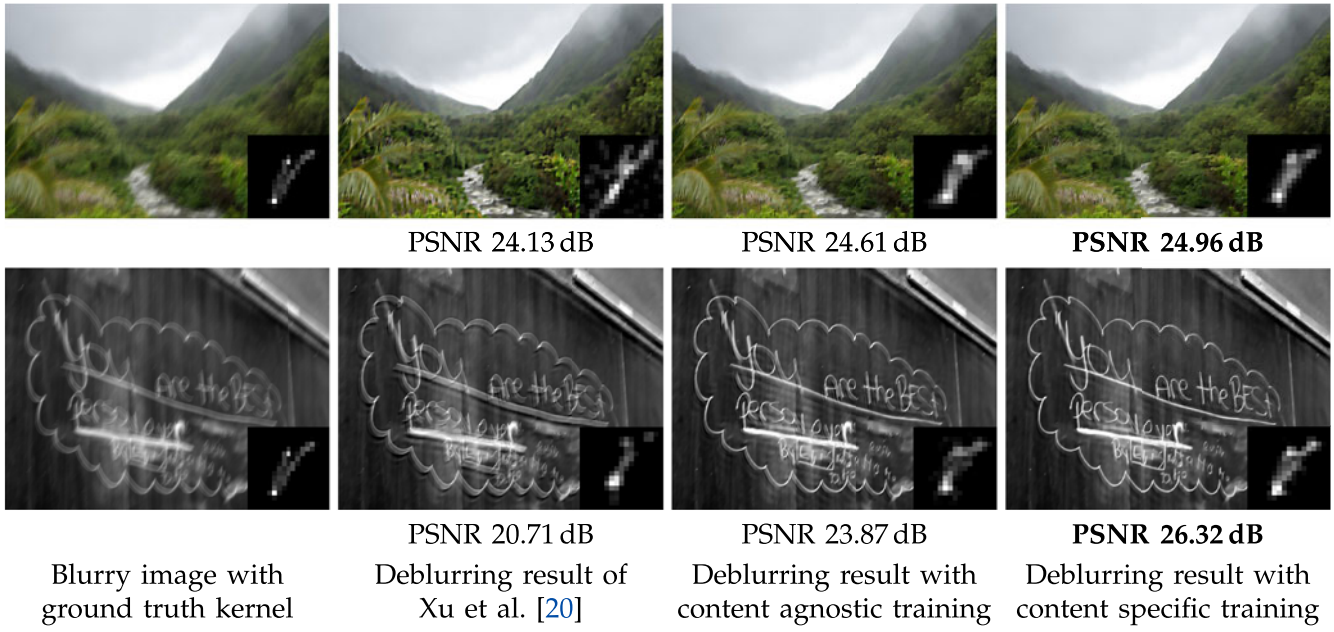



Fig. 13. Comparison of deblurring results for NNs that have been trained with image examples from the entire ImageNet dataset (content *agnostic*) and from particular subsets (content *specific*), i.e., image categories *valley* (top row) and *blackboard* (bottom row). We also show the results of the state-of-the-art method [20].

TABLE 2
PSNRs of Content Agnostic versus Content Specific Training on Images of the Category *Valley* (Top) and *Blackboard* (Bottom)

| Valley image (Flickr ID) | Xu et al. [20] (dB) | Cont. agnostic training (dB) | Cont. specific training (dB) |
|---|---|---|---|
| fCetj6 | 24.13 | 24.61 | **24.96** |
| hR7YPb | **28.59** | 27.96 | 28.32 |
| i1GWi8 | 20.70 | 20.17 | **20.91** |
| nz5BxB | **22.97** | 22.28 | 22.80 |
| nRuiRC | 23.32 | 23.20 | **23.44** |
| Average | 23.94 | 23.64 | **24.09** |

| Blackboard image (Flickr ID) | Xu et al. [20] (dB) | Cont. agnostic training (dB) | Cont. specific training (dB) |
|---|---|---|---|
| bkkdz3 | 26.37 | 23.63 | **27.53** |
| btKo6w | 20.71 | 23.87 | **26.32** |
| dbQBYX | 20.88 | 22.36 | **24.25** |
| f9jSxK | 23.63 | 21.60 | **24.41** |
| fK3V16 | **25.32** | 23.77 | 25.22 |
| Average | 23.38 | 23.04 | **25.55** |

*All images were downloaded in resolution "medium" and have been blurred with blur kernel number 2 from [26].*

own C++/CUDA-based neural network toolbox. After training once for a certain blur class (e.g., camera shake), which takes about two days per stage, applying the network is very fast and can be done in Matlab without additional dependencies.

The runtimes on an Intel i5 using only Matlab are shown in Table 1. The most expensive calculation is the creation of the multiple hidden representations in the convolutional layer of the NN (in this example: 32).

## 6 Experiments

If not otherwise stated, all experiments were performed with a multi-scale, triple stage architecture. We use up to three scales for kernels of size $17 \times 17$, $25 \times 25$, $33 \times 33$. On each scale each feature extraction module consists of a convolution layer with 32 filters, a tanh-layer, a linear recombination to 32 new hidden images, a further tanh-layer, and a recombination to four gradient-like images, two for $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{y}}_i$ each (in the third stage: eight gradient-like images). In the case of the network with blur kernels of size $33 \times 33$, we deconvolve the estimated kernels with a small Gaussian with $\sigma = 0.5$ to counter the over-smoothing effect of the L2 norm used during training. For the specific choice of the architecture, we refer to the influence of model parameters on the kernel estimation performance in Figs. 3 to 4.

### 6.1 Image Content Specific Training

A number of recent works [25], [27], [28] have pointed out the shortcoming of state-of-the-art algorithms [2], [3] to depend on the presence of strong salient edges and their diminished performance in the case of images that contain textured scenes such as natural landscape images. The reason for this is the deficiency of the so-called image prediction step, which applies a combination of bilateral and shock filtering to restore latent edges that are used for subsequent kernel estimation.

In this context, learning the latent image prediction step offers a great advantage: by training our network with a particular class of images, it is able to focus on those features that are informative for the particular type of image. In other words, the network learns content-specific nonlinear filters, which yield improved performance.

To demonstrate this, we used the same training procedure as described above, however, we reduced the training set to images from a specific image category within the ImageNet dataset. In particular, we used the image category *valley*[2] containing a total of 1,395 pictures. In a second experiment, we trained a network on the image category *blackboard*[3] with a total of 1,376 pictures. Fig. 12 shows typical example images from these two classes. Fig. 13 compares deblurring results of the state-of-the-art algorithm described in [20] with our approach trained on images sampled from the entire ImageNet dataset, and trained on the aforementioned image categories only. Table 2 compares the two networks on a random selection of photos taken from Flickr. We see that content specific training outperforms content agnostic training for all examples, and demonstrates on par performance with [20] for the category *valley*. For category *blackboard*, which is more distinct from generic natural images, it surpasses its competitor by a large margin.

### 6.2 Noise Specific Training

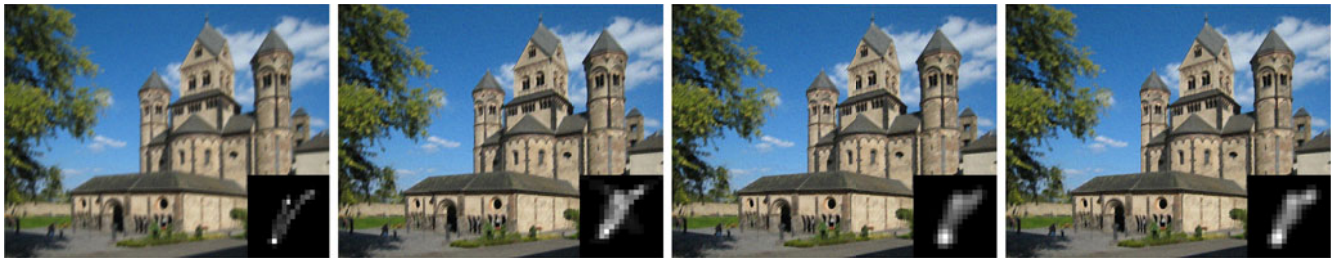
Typically, image noise impedes kernel estimation. To counter noise in blurry images, current state-of-the-art deblurring algorithms apply a denoising step during latent image prediction such as bilateral filtering [2] or Gaussian filtering [3]. However, in a recent work [29], the authors show that for increased levels of noise current methods fail to yield satisfactory results and propose a novel robust deblurring algorithm. Again, if we include image noise in our training phase, our network is able to adapt and learn filters that perform better in the presence of noise. In particular, we trained a network on images with Gaussian noise of 5 percent added during the training phase. Fig. 14 compares the results for an image taken from [29] with 5 percent Gaussian noise for a network trained with 1 and 5 percent of added Gaussian noise during training, respectively. We also show the result of [29] and compare peak signal-to-noise ratio (PSNR) for objective evaluation. All results use the same non-blind deconvolution of [29]. The noise specific training is most successful, but even the noise agnostic NN outperforms the non-learned method on this example.

### 6.3 Spatially-Varying Blur

Since the prediction step of our trained NN is independent of the convolution model, we can also use it in

2. ImageNet 2011 Fall Release > Geological formation, formation > Natural depression, depression > Valley, vale (http://www.image-net.org/synset?wnid=n09468604)

3. ImageNet 2011 Fall Release > Artifact, artifact > Sheet, flat solid > Blackboard, chalkboard (http://www.image-net.org/synset?wnid=n02846511)



| | PSNR 23.17 dB | PSNR 23.28 dB | **PSNR 23.52 dB** |
|---|---|---|---|
| Blurry image with ground truth kernel | Deblurring result of of Zhong et al. [29] | Deblurring result w. noise agnostic training | Deblurring result w. noise specific training |

Fig. 14. Comparison of deblurring results for NNs that have been trained with different amounts of noise added to the sample images during training. The network that has been trained with the same amount of noise as the input blurry image (5 percent noise) performs best. We also show the results of a recently proposed deblurring method tailored for increased levels of noise [29].

conjunction with the recently proposed fast forward model of [31] to restore images with spatially-varying blur. To this end, we replace the objective function Eq. (5) with Eq. (8) of [31] in our kernel estimation module in a network trained for spatially invariant deconvolution. We solve for $k$ in a two-step procedure: first we compute local blur kernels using the efficient filter flow (EFF) model of [32]; in a second step we project the blur kernels onto a motion basis aka [31], as explained below. Fig. 15 shows a comparison between recent state-of-the-art algorithms for spatially-varying blur along with our deblurring result that features comparable quality.

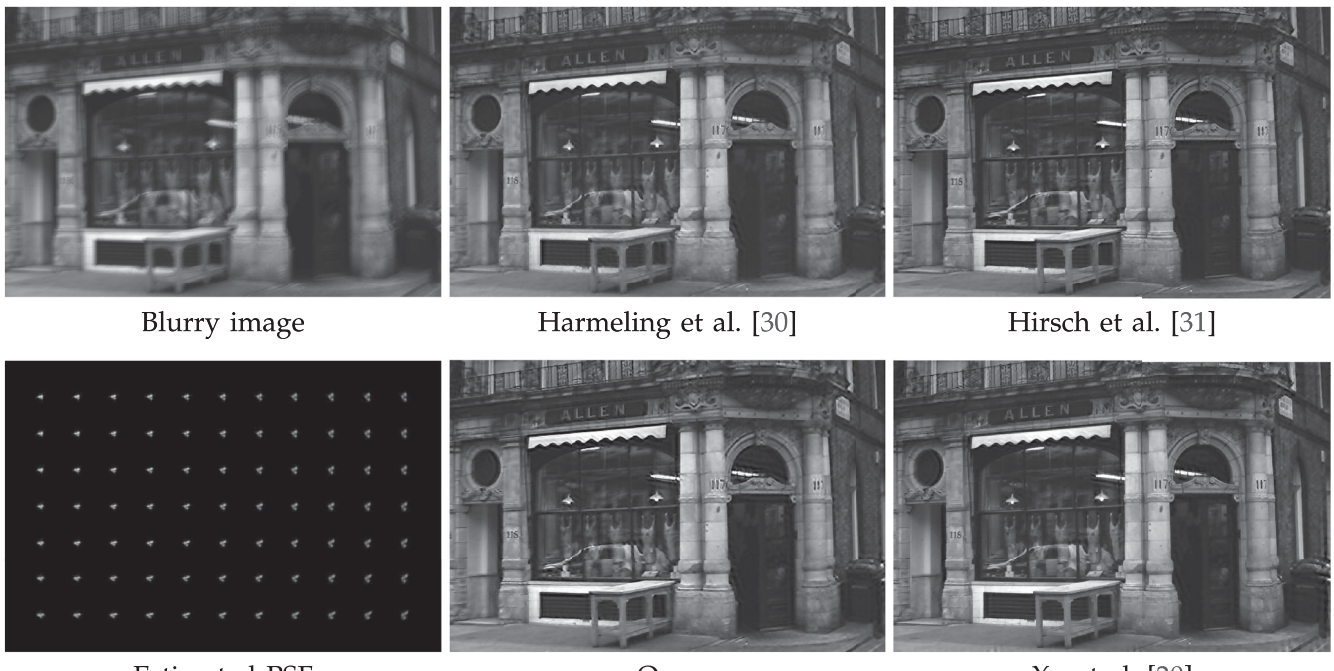
For the estimation of spatially-varying blur we solve the following objective

$$\sum_i \|\tilde{X}_i\,\tilde{\mathbf{k}} - \tilde{\mathbf{y}}_i\|^2 + \beta_k \|\tilde{\mathbf{k}}\|^2 \tag{7}$$

in our kernel estimation module. Here $\tilde{X}_i$ denotes the EFF matrix of $\tilde{\mathbf{x}}$ (cf. [32, Eq. (9)]) and $\tilde{\mathbf{k}}$ the stacked sequence of local kernels $\tilde{\mathbf{k}}^{(r)}$, one for each patch that are enumerated by index $r$. Since Eq. (7) is quadratic in $\tilde{\mathbf{k}}$, we can solve for a local blur $\tilde{\mathbf{k}}^{(r)}$ in a single step

$$\tilde{\mathbf{k}}^{(r)}_{direct} \approx F^{\mathsf{H}} \frac{\sum_i \overline{FC_r \operatorname{Diag}(\mathbf{w}^{(r)})\,\tilde{\mathbf{x}}_i} \odot (FC_r \operatorname{Diag}(\mathbf{w}^{(r)})\,\tilde{\mathbf{y}}_i)}{\sum_i |FC_r \operatorname{Diag}(\mathbf{w}^{(r)})\,\tilde{\mathbf{x}}_i|^2 + \beta_k}, \tag{8}$$

where $C_r$ are appropriately chosen cropping matrices, and $\mathbf{w}^{(r)}$ are window functions matching the size of $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{y}}_i$. Note that Eq. (8) is only approximately true and is motivated by Eq. (8) in [31]. Subsequently, we project the estimated kernel computed by Eq. (8) to a motion blur kernel basis. In our experiments we use the same basis as [31] comprising translations within the image plane and in-plane rotations only. This additional projection step constrains the estimated blur to physically plausible ones. Formally, we compute



| | | |
|---|---|---|
| Blurry image | Harmeling et al. [30] | Hirsch et al. [31] |
| Estimated PSF | Ours | Xu et al. [20] |

Fig. 15. Comparison on *Butcher Shop* example [30] of state-of-the-art deblurring methods for removing non-uniform blur together with our estimated PSF.

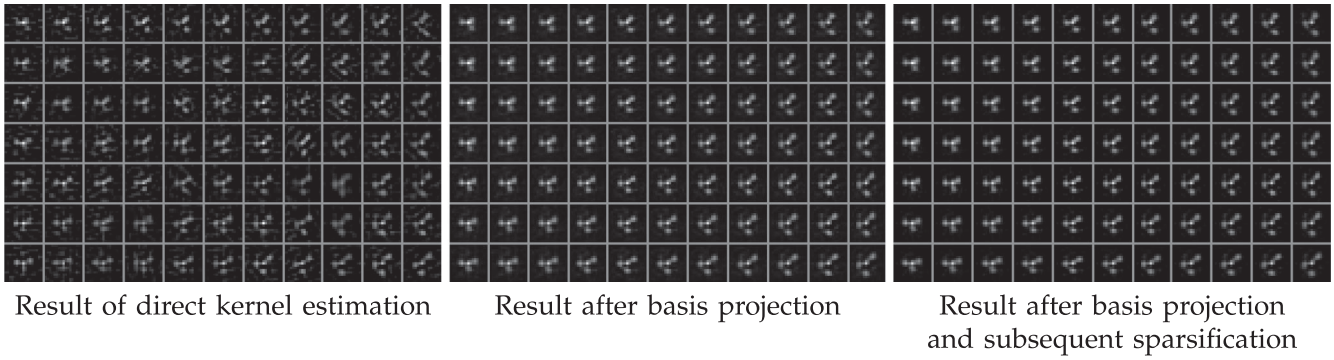



Fig. 16. Visualisation of our kernel estimation in the case of spatially-varying blur for the *Butcher Shop* example shown in Fig. 15. The left panel shows the kernel estimated with Eq. (8), the middle and right panels show the kernel after applying a subsequent projection step to our motion basis, i.e., the result of Eq. (9) with $\eta$ set to 1.0 and 0.0314 respectively.

$$\tilde{\mathbf{k}}_{est}^{(r)} \approx B^{(r)}\, T_\eta \underbrace{\sum_r (B^{(r)})^T \tilde{\mathbf{k}}_{direct}^{(r)}}_{\mu}, \tag{9}$$

where again we make use of the notation chosen in [31], i.e., $B^{(r)}$ denotes the motion blur kernel basis for patch $r$. Then $\mu$ are the coefficients in the basis of valid motion blurs. $T_\eta$ denotes a thresholding operator that sets all elements to zero below a certain threshold whereby the threshold is chosen such that only $\eta$ percent of entries remain non-zero. This thresholding step is motivated by [2], who also apply a hard thresholding step to the estimated kernels in order to get rid of spurious artefacts. Fig. 16 shows the intermediate results of our kernel estimation procedure in the case of spatially-varying blur.

### 6.4 Quantitative Comparison

We evaluate our method on the benchmark dataset of [27], which is an extended version of the standard test set of [26]. It consists of 80 images with about one megapixel in size each. The images are artificially blurred each with eight different blur kernels and contain 1 percent additive Gaussian noise. The performance is illustrated in Fig. 17. Here we compare with Levin et al. [33], Cho and Lee [2], Krishnan et al. [34], Sun et al. [27], and Xu and Jia [3] and Michaeli and Irani [35], where however [27] and [35] have runtimes in the order of hours. For fair comparison we used the non-blind deconvolution method of [36] for all methods in our evaluation. While our method is competitive with the state of the art for small blur kernels, our method falls short in performance for blur kernel sizes above 25 × 25 pixels. We discuss reasons for this in Section 6.6.

For real-world examples, we refer the interested reader to the supplementary material, available online.

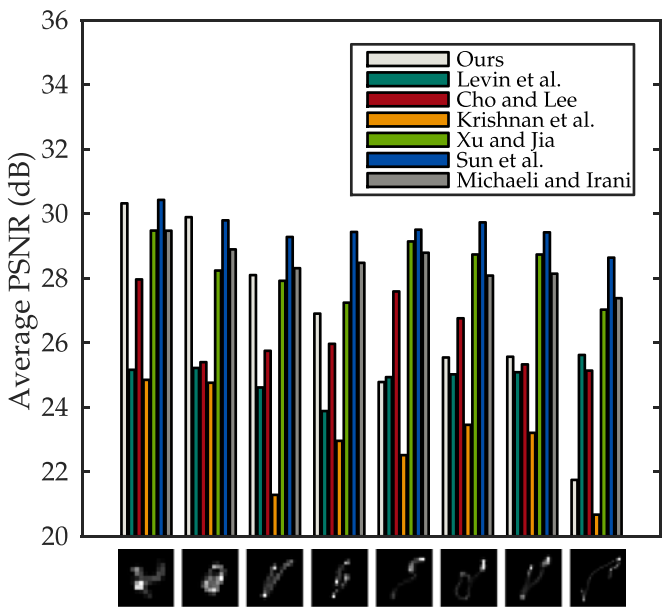



Fig. 17. Results of the benchmark dataset of Sun et al. [27]. The results are sorted according to blur kernel size. While for kernels up to a size of 25 × 25 pixels, our approach yields comparable results, it falls short for larger blur kernels. Reasons for the performance drop are discussed in Section 6.6.

### 6.5 Dependence on Size of the Observed Image

As noted by Hu and Yang [25], blind deblurring methods are most successful in predicting the kernel in regions of an image that exhibit strong salient edges. Other regions are less informative about the kernel, and have been shown to even hurt kernel estimation when included in the input to the blind deconvolution algorithm. Ideally, an estimation procedure should weight its input according to its information content. In the worst case, a larger input would not improve the results, but would not cause a deterioration either.

We study the behavior of our method with respect to the size of the observed image. It is possible that the NN learned to ignore image content detrimental to the

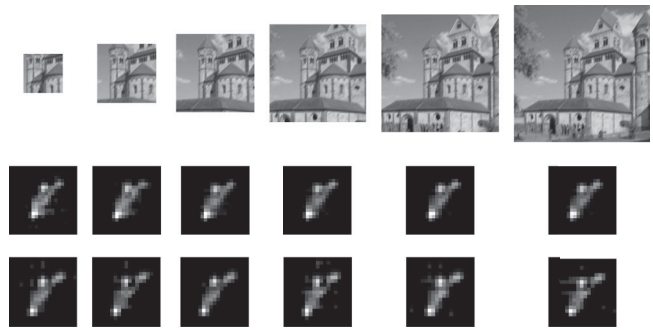


Fig. 18. Results for kernel estimation for different sizes of the observed image. *Top row:* Differently sized inputs to the blind deblurring algorithm. *Middle row:* Estimated kernels of our method. Larger inputs lead to better results. *Bottom Row:* Estimated kernels of [20]. No clear trend is visible.

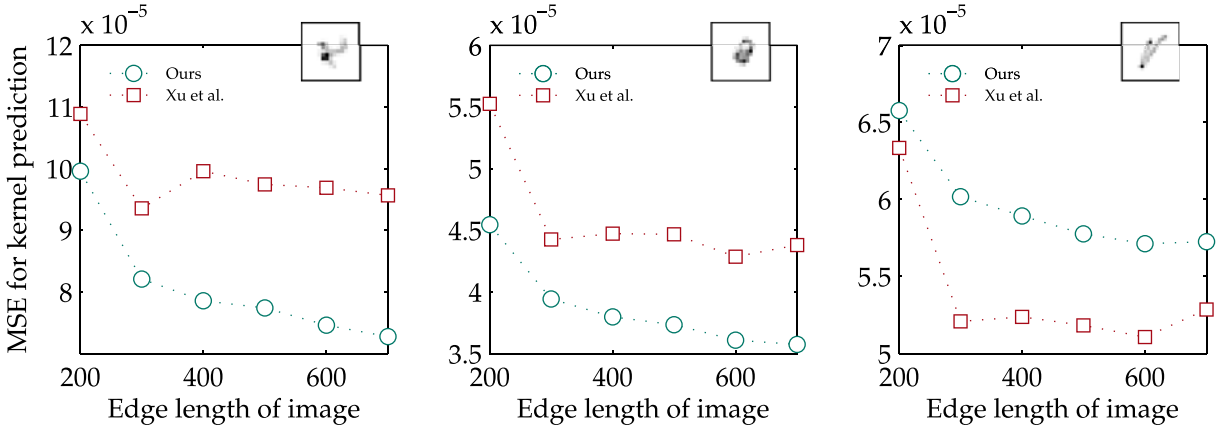



Fig. 19. Dependence of the estimated kernel on the size of the observed image. We show the MSE of predictions of three kernels for different sized inputs (cf. Fig. 18), averaged over the 52 largest images from [27]. The results of the NN decrease monotonously.

kernel estimation. In Fig. 18 the predicted kernels for different sized crops of a blurry image are shown. Indeed, this example suggests that for our learned algorithm the kernel converges with input images increasing in size, while a non-learned state-of-the-art algorithm [20] exhibits no such trend. The more thorough analysis in Fig. 19 confirms this behavior: when evaluating the MSE for three different kernels, averaged over the 52 largest images from [27], it monotonously decreases for larger crops of the blurry image. We also see that the NN outperforms the competing method for the two small blur kernels.

### 6.6 Limitations

A limitation of our current approach is the performance drop in the case of larger blur kernels. Fig. 20 shows an example failure case from the benchmark dataset of [26]. We believe the reason for this is the suboptimal architecture of our multi-scale approach at higher resolution scales. While a multi-scale approach exhibits better performance compared to a single scale network, the observed performance drop for larger blurs suggests that our approach to handling large blurs is not yet optimal: below a kernel size of 17 $\times$ 17 pixels we use a single-scale network with competitive performance, up to 25 $\times$ 25 a two-scale network with reasonable results, and for larger kernels a three-scale architecture that exhibits the observed drop in performance.

## 7 Conclusion

We have shown that it is possible to automatically learn blind deconvolution by reformulating the task as a single large nonlinear regression problem, mapping between blurry input and predicted kernel. The key idea is to incorporate the properties of the generative forward model into our algorithm, namely that the image is convolved with the same blur kernel everywhere. While features are extracted locally in the image, the kernel estimation module combines them globally. Next, the image estimation module propagates the information to the whole image, reducing the difficulty of the problem for the following iteration.

Our approach can adapt to different settings (e.g., blurry images with strong noise, or specific image classes), and it could be further extended to combine deblurring with other steps of the imaging pipeline, including over-saturation, Bayer filtering, HDR, or super-resolution.

The blur class also invites future research: instead of artificially sampling from a stochastic process, one could use recorded spatially-varying camera shakes, or a different source of unsharpness, like lens aberrations or atmospheric turbulences in astrophotography. Additionally, the insights gained from the trained system could be beneficial to existing hand-crafted methods. This includes using higher-order gradient representations and extended gradient filters.

Scalability of our method to large kernel sizes is still an issue, and this may benefit from future improvements in neural net architecture and training.

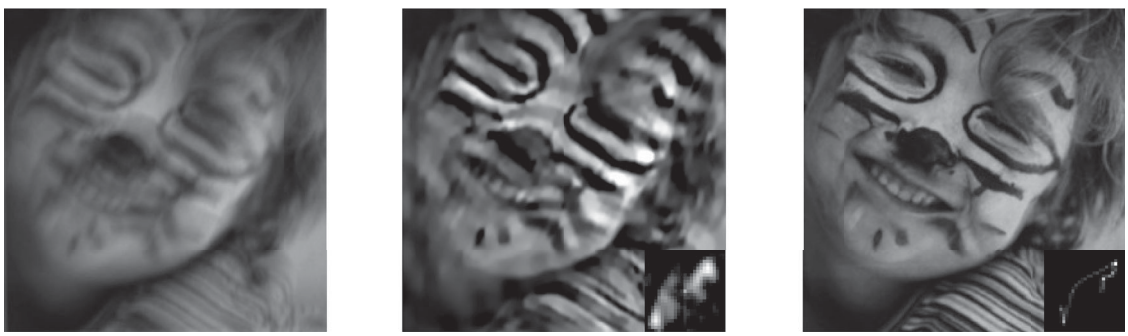



Fig. 20. For larger blur kernels our approach falls short in yielding acceptable deblurring results. Shown is an example from [26] with a kernel of size 27 $\times$ 27.

## REFERENCES


[1] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman, "Removing camera shake from a single photograph," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 787–794, 2006.

[2] S. Cho and S. Lee, "Fast motion deblurring," *ACM Trans. Graph.*, vol. 28, no. 5, p. 145, 2009.

[3] L. Xu and J. Jia, "Two-phase kernel estimation for robust motion deblurring," in *Proc. 11th Eur. Conf. Comput. Vis.*, 2010, pp. 157–170.

[4] L. Bottou and P. Gallinari, "A framework for the cooperation of learning algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 1991, pp. 781–788.

[5] D. Wipf and H. Zhang, "Revisiting Bayesian blind deconvolution," *ArXiv e-prints*, 2013.

[6] M. Egmont-Petersen, D. de Ridder, and H. Handels, "Image processing with neural networks a review," *Pattern Recog.*, vol. 35, no. 10, pp. 2279–2301, 2002.

[7] D. De Ridder, R. P. Duin, M. Egmont-Petersen, L. J. van Vliet, and P. W. Verbeek, "Nonlinear image processing using artificial neural networks," *Adv. Imag. Elect. Phys.*, vol. 126, pp. 352–450, 2003.

[8] I. Aizenberg, D. Paliy, C. Moraga, and J. Astola, "Blur identification using neural network for image restoration," in *Proc. 9th Int. Conf. Comput. Intell. Theory Appl.*, 2006, vol. 38, pp. 441–455.

[9] C. Khare and K. K. Nagwanshi, "Image restoration in neural network domain using back propagation network approach," *Int. J. Comput. Inf. Syst.*, vol. 2, no. 5, pp. 25–31, 2011.

[10] C. M. Cho and H. S. Don, "Blur identification and image restoration using a multilayer neural network," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 1991, pp. 2558–2563.

[11] J. E. Tansley, M. J. Oldfield, and D. J. MacKay, "Neural network image deconvolution," in *Maximum Entropy and Bayesian Methods*, ser. Fundamental Theories of Physics. New York, NY, USA: Springer, 1996, pp. 319–325.

[12] R. J. Steriti and M. A. Fiddy, "Blind deconvolution of images by use of neural networks," *Optics Lett.*, vol. 19, no. 8, pp. 575–577, 1994.

[13] C. J. Schuler, H. C. Burger, S. Harmeling, and B. Schlkopf, "A machine learning approach for non-blind image deconvolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 1067–1074.

[14] L. Xu, J. S. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1790–1798.

[15] J. Sun, W. Cao, Z. Xu, and J. Ponce, "Learning a convolutional neural network for non-uniform motion blur removal," *ArXiv e-prints*, 2015.

[16] Z. Hu, J.-B. Huang, and M.-H. Yang, "Single image deblurring with adaptive dictionary learning," in *Proc. IEEE Int. Conf. Image Process.*, 2010, pp. 1169–1172.

[17] U. Schmidt, C. Rother, S. Nowozin, J. Jancsary, and S. Roth, "Discriminative non-blind deblurring," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 604–611.

[18] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2010, pp. 2528–2535.

[19] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 2018–2025.

[20] L. Xu, S. Zheng, and J. Jia, "Unnatural l0 sparse representation for natural image deblurring," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 1107–1114.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2009, pp. 248–255.

[22] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. West Sussex, England: Univ. Press Group Limited, 2006.

[23] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," *ArXiv e-prints*, 2012.

[24] D. Erhan, A. Courville, and Y. Bengio, "Understanding representations learned in deep architectures," Univ de Montréal/DIRO, Montréal, QC, Canada, Tech. Rep. 1355, 2010.

[25] Z. Hu and M.-H. Yang, "Good regions to deblur," in *Proc. 12th Eur. Conf. Comput. Vis.* 2012, pp. 59–72.

[26] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Understanding and evaluating blind deconvolution algorithms," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2009, pp. 1964–1971.

[27] L. Sun, S. Cho, J. Wang, and J. Hays, "Edge-based blur kernel estimation using patch priors," in *Proc. IEEE Int. Conf. Comput. Photography*, 2013, pp. 1–8.

[28] C. Wang, Y. Yue, F. Dong, Y. Tao, X. Ma, G. Clapworthy, H. Lin, and X. Ye, "Nonedge-specific adaptive scheme for highly robust blind motion deblurring of natural images," *IEEE Trans. Image Process.*, vol. 22, no. 3, pp. 884–897, Mar. 2013.

[29] L. Zhong, S. Cho, D. Metaxas, S. Paris, and J. Wang, "Handling noise in single image deblurring using directional filters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 612–619.

[30] S. Harmeling, M. Hirsch, and B. Schlkopf, "Space-variant single-image blind deconvolution for removing camera shake," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 829–837.

[31] M. Hirsch, C. J. Schuler, S. Harmeling, and B. Schlkopf, "Fast removal of non-uniform camera shake," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 463–470.

[32] M. Hirsch, S. Sra, B. Schlkopf, and S. Harmeling, "Efficient filter flow for space-variant multiframe blind deconvolution." in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2010, pp. 607–614.

[33] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Efficient marginal likelihood optimization in blind deconvolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2011, pp. 2657–2664.

[34] D. Krishnan, T. Tay, and R. Fergus, "Blind deconvolution using a normalized sparsity measure," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2011, pp. 233–240.

[35] T. Michaeli and M. Irani, "Blind deblurring using internal patch recurrence," in *Proc. IEEE Eur. Conf. Comput. Vis.* 2014, pp. 783–798.

[36] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 479–486.


**Christian J. Schuler** studied physics at the Karlsruhe Institute of Technology (KIT) and received the degree with distinction in 2004. He received his PhD degree at the Max Planck Institute for Intelligent systems in Prof. Bernhard Schölkopf's Department of Empirical Inference in 2015. He is interested in applying machine learning to inverse problems, in particular, image deconvolution.

**Michael Hirsch** is a Senior Research Scientist at the Max Planck Institute for Intelligent Systems in Prof. Bernhard Schölkopf's department of Empirical Inference and is leading a research group on computational imaging. His research interests cover a wide range of signal and image processing problems in scientific imaging as well as computational photography. Dr. Hirsch studied physics and mathematics at the University of Erlangen and at Imperial College London. He received a Diploma in theoretical physics in 2007, before joining Prof. Dr. Bernhard Schölkopf's research group at the Max Planck Institute for Biological Cybernetics. After his doctoral studies he worked as a post-doctoral researcher at University College London from 2011 to 2014.

**Stefan Harmeling** is a Professor at the Heinrich-Heine-Universität in Düsseldorf, Germany. Before he was a senior Research Scientist at the Max Planck Institute for Intelligent Systems (formerly Biological Cybernetics) in Prof. Bernhard Schölkopf's department of Empirical Inference. His interests include machine learning, image processing, computational photography, probabilistic and causal inference. Dr. Harmeling studied mathematics and logic at the University of Münster (Dipl Math 1998) and computer science with an emphasis on artificial intelligence at Stanford University (MSc 2000). During his doctoral studies he was a member of Prof. Klaus-Robert Müller's research group at the Fraunhofer Institute FIRST (Dr rer nat 2004). Thereafter he was a Marie Curie Fellow at the University of Edinburgh from 2005 to 2007, before joining the Max Planck Institute of Biological Cybernetics/ Intelligent Systems. Prof. Harmeling is now at the Computer Science department at the Heinrich Heine University Düsseldorf.

**Bernhard Schölkopf** received the doctorate degree in computer science from the Technical University Berlin. His thesis on Support Vector Learning received the annual dissertation prize of the German Association for Computer Science (GI). He has done research at AT&T Bell Labs, GMD FIRST, Berlin, the Australian National University, Canberra, and Microsoft Research Cambridge, United Kingdom. In July 2001, he was appointed as a scientific member of the Max Planck Society. He received the J.K. Aggarwal Prize of the International Association for Pattern Recognition, and the Max Planck Research Award. The ISI lists him as a highly cited researcher, and he is a board member of the NIPS foundation and of the International Machine Learning Society.