MACHINE LEARNING APPROACHES TO IMAGE DECONVOLUTION

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Christian Johannes Schuler
aus Nürnberg

Tübingen
2014

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	30. 9. 2015
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dr. Hendrik Lensch
2. Berichterstatter:	Prof. Dr. Bernhard Schölkopf
3. Berichterstatter:	Prof. Dr. Yair Weiss

Summary

Image blur is a fundamental problem in both photography and scientific imaging. Even the most well-engineered optics are imperfect, and finite exposure times cause motion blur. To reconstruct the original sharp image, the field of *image deconvolution* tries to recover recorded photographs algorithmically. When the blur is known, this problem is called *non-blind* deconvolution. When the blur is unknown and has to be inferred from the observed image, it is called *blind* deconvolution. The key to reconstructing information lost due to blur and noise is to use prior knowledge. To this end, this thesis develops approaches inspired by machine learning that include more available information and advance the current state of the art for both non-blind and blind image deconvolution.

Optical aberrations of a lens are encoded in an initial calibration step as a spatially-varying point spread function. With prior information about the distribution of gradients in natural images, the original image is reconstructed in a maximum a posteriori (MAP) estimation, with results comparing favorably to previous methods. By including the camera's color filter array in the forward model, the estimation procedure can perform demosaicing and deconvolution jointly and thereby surpass the quality of the results yielded by a separate demosaicing step.

The applicability of removing optical aberrations is broadened further by estimating the point spread function from the image itself. We extend an existing MAP-based blind deconvolution approach to the first algorithm that is able to remove spatially-varying lens blur *blindly*, including chromatic aberrations. The properties of lenses restrict the class of possible point spread functions and reduce the space of parameters to be inferred, enabling results on par with the best non-blind approaches for the lenses tested in our experiments.

To capture more information about the distribution of natural images and capitalize on the abundance of training data, neural networks prove to be a useful tool. As other successful non-blind deconvolution methods, a regularized inversion of the blur is performed in the Fourier domain as an initial step. Next, a large neural network learns the mapping from the preprocessed image back to the uncorrupted original. The trained network surpasses results of state-of-the-art algorithms on both artificial and real-world examples.

For the first time, a learning approach also succeeds in blind image deconvolution. A deep neural network "unrolls" the estimation procedure of existing methods for this task. After training end-to-end on artificially generated example images, the network achieves performance competitive with state-of-the-art methods in the generic case, and even goes beyond when trained for a specific image category.

Zusammenfassung

Unscharfe Bilder sind ein häufiges Problem, sowohl in der Fotografie als auch in der wissenschaftlichen Bildgebung. Auch die leistungsfähigsten optischen Systeme sind nicht perfekt, und endliche Belichtungszeiten verursachen Bewegungsunschärfe. *Dekonvolution* hat das Ziel das ursprünglich scharfe Bild aus der Aufnahme mit Hilfe von *algorithmischen* Verfahren wiederherzustellen. Kennt man die exakte Form der Unschärfe, so wird dieses Rekonstruktions-Problem als *nicht-blinde* Dekonvolution bezeichnet. Wenn die Unschärfe aus dem Bild selbst inferiert werden muss, so spricht man von *blinder* Dekonvolution. Der Schlüssel zum Wiederherstellen von verlorengegangener Bildinformation liegt im Verwenden von verfügbarem Vorwissen über Bilder und die Entstehung der Unschärfe. Hierzu entwickelt diese Arbeit verschiedene Ansätze um dieses Vorwissen besser verwenden zu können, basierend auf Methoden des maschinellen Lernens, und verbessert damit den Stand der Technik, sowohl für nicht-blinde als auch für blinde Dekonvolution.

Optische Abbildungsfehler lassen sich in einem einmal ausgeführten Kalibrierungsschritt vermessen und als eine ortsabhängige Punktverteilungsfunktion des einfallenden Lichtes beschreiben. Mit dem Vorwissen über die Verteilung von Gradienten in Bildern kann das ursprüngliche Bild durch eine Maximum-a-posteriori (MAP) Schätzung wiederhergestellt werden, wobei die resultierenden Ergebnisse vergleichbare Methoden übertreffen. Wenn man des Weiteren im Vorwärtsmodell die Farbfilter des Sensors berücksichtigt, so kann das Schätzverfahren Demosaicking und Dekonvolution simultan ausführen, in einer Qualität die den Ergebnissen durch Demosaicking in einem separaten Schritt überlegen ist.

Die Korrektur von Linsenfehlern wird breiter anwendbar indem man die Punktverteilungsfunktion vom Bild selbst inferiert. Wir erweitern einen existierenden MAP-basierenden Ansatz für blinde Dekonvolution zum ersten Algorithmus, der in der Lage ist auch ortsabhängige optische Unschärfen *blind* zu entfernen, einschließlich chromatischer Aberration. Die spezifischen Eigenschaften von Kamera-Objektiven schränken den Raum der zu schätzenden Punktverteilungsfunktionen weit genug ein, so dass für die in unseren Experimenten untersuchten Objektive die erreichte Bildrekonstruktion ähnlich erfolgreich ist wie bei nicht-blinden Verfahren.

Es zeigt sich, dass neuronale Netze von im Überfluss vorhandenen Bilddatenbanken profitieren können um mehr über die Bildern zugrundeliegende Wahrscheinlichkeitsverteilung zu lernen. Ähnlich wie in anderen erfolgreichen nicht-blinden Dekonvolutions-Ansätzen wird die Unschärfe zuerst durch eine regularisierte Inversion im Fourier-Raum vermindert. Danach ist es einem neuronalen Netz mit großer Kapazität möglich zu lernen, wie aus einem derart vorver-

arbeiteten Bild das fehlerfreie Original geschätzt werden kann. Das trainierte Netz produziert anderen Methoden überlegene Ergebnisse, sowohl auf künstlich generierten Beispielen als auch auf tatsächlichen unscharfen Fotos.

Zum ersten Mal ist ein lernendes Verfahren auch hinsichtlich der blinden Bild-Dekonvolution erfolgreich. Ein tiefes neuronales Netz modelliert die Herangehensweise von bisherigen Schätzverfahren und wird auf künstlich generierten Beispielen trainiert die Unschärfe vorherzusagen. Nach Abschluss des Trainings ist es in der Lage, mit anderen aktuellen Methoden vergleichbare Ergebnisse zu erzielen, und geht über deren Ergebnisse hinaus, wenn man speziell für eine bestimmten Subtyp von Bildern trainiert.

Papers Included in this Thesis

Papers included in this thesis:

[Sch+11] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. "Non-stationary correction of optical aberrations". In: *IEEE Int. Conf. Computer Vision*. 2011. DOI: 10.1109/iccv.2011.6126301

[Sch+12] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. "Blind Correction of Optical Aberrations". In: *Computer Vision – ECCV 2012*. Lecture Notes in Computer Science. Springer, 2012, pp. 187–200. DOI: 10.1007/978-3-642-33712-3_14

[Sch+13b] C. J. Schuler, H. C. Burger, S. Harmeling, and B. Schölkopf. "A Machine Learning Approach for Non-blind Image Deconvolution". In: *IEEE Conf. Computer Vision and Pattern Recognition*. 2013, pp. 1067–1074. DOI: 10.1109/cvpr.2013.142

[Sch+14] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. "Learning to Deblur". In: *ArXiv e-prints* (2014). arXiv: 1406.7444 [cs.CV]. Submitted to a journal.

Papers related to, but not included in this thesis:

[Hir+11] M. Hirsch, C. J. Schuler, S. Harmeling, and B. Schölkopf. "Fast removal of non-uniform camera shake". In: *IEEE Int. Conf. Computer Vision*. 2011, pp. 463–470. DOI: 10.1109/iccv.2011.6126276

[BSH12c] H. C. Burger, C. J. Schuler, and S. Harmeling. "Image denoising: Can plain Neural Networks compete with BM3D?". In: *IEEE Conf. Computer Vision and Pattern Recognition*. 2012, pp. 2392–2399. DOI: 10.1109/cvpr.2012.6247952

[BSH12a] H. C. Burger, C. J. Schuler, and S. Harmeling. "Image denoising with multi-layer perceptrons, part 1: comparison with existing algorithms and with bounds". In: *ArXiv e-prints* (2012). arXiv: 1211.1544 [cs.CV]

[BSH12b] H. C. Burger, C. J. Schuler, and S. Harmeling. "Image denoising with multi-layer perceptrons, part 2: training trade-offs and analysis of their mechanisms". In: *ArXiv e-prints* (2012). arXiv: 1211.1552 [cs.CV]

I would like to thank my co-authors for their permission to base chapters of my thesis on joint publications.

Contents

List of Figures

Introduction

Blur is a fundamental problem of physical imaging. Camera optics are never perfect, and typically spread the light emanating from a single point in the scene over several pixels in the sensor plane. It is not always possible to prevent the camera or objects in the scene from moving during image capture, which is especially problematic for long exposure times. In both examples, the result is a blurred image. Blur poses a problem both for everyday photography and scientific imaging, e.g., microscopy or astronomy across all spectral regions from radio waves to gamma rays.

One remedy is to improve the imaging device: using optics with less aberrations, or increasing the sensor's sensitivity to allow smaller exposure times. However, this increases production costs and usually is a trade-off between several target variables. For example, it is easier to minimize aberrations for lenses with small apertures, but small apertures collect less light and increase exposure time. Some limits are even fundamental and independent of current technology (cf. Abbe diffraction limit [BW99]).

Another solution is to recover the original signal *computationally*. This approach belongs to the class of inverse problems and is called deconvolution. When the blurring process is known a priori (e.g., a fixed optical system that has been measured precisely) the problem is called *non-blind* deconvolution. Even though the blur is known, the reconstruction is challenging, because high frequencies of the signal lose intensity for most blurs, often going towards zero. The noise however, which is an issue without blur in itself, is unaffected in strength and, consequently, can drown the signal. To recover the lost information, prior knowledge has to be incorporated, e.g., in the form of the distribution of the expected image content or the noise.

The reconstruction becomes even more challenging when the blur is not known beforehand. For example, in the case of camera shake, although it is possible to measure the movement of the camera, it is usually impractical. The problem of inferring both the uncorrupted image and the blur is known as *blind* deconvolution. Even without noise, the problem is underdetermined since many combinations of blur and reconstructed image could explain the observation. Again, prior knowledge has to be taken into account for the blind case, in addition to knowledge about the image also about the specifics of the convolution, e.g., camera shake stemming from physically plausible translations and rotations of the camera.

Deconvolution stands as a mature field with many successful applications. Since the 1950s, deconvolution has been used in seismology to infer the Earth's structure from recorded seismograms [Rob54]. The launch of the Hubble space telescope in 1990 motivated the use of

deconvolution methods in the field of astronomy, after the telescope's primary mirror was found to be flawed [Wal90]. Measurements done in the twelve year period until it was repaired necessitated the use of computational corrections. In the field of computational photography, cameras are even designed such that the recorded image has to be post-processed in software, optimizing the full imaging pipeline for a desired property of the final photograph. For example, a coded aperture captures depth information of the scene by adding a depth-dependent blur [Lev+07].

Blur and noise corrupt the information contained within an image. The key idea of reconstructing a signal suffering from loss of information is to use available prior knowledge. Ideally, the prior information contains the full probability distribution of the unknown quantities. Practically, the distribution of all images is computationally intractable, and with respect to its dimensionality only sparsely sampled by even the largest collection of images. *Machine Learning* provides tools and methodologies to approximate the distribution explicitly or implicitly, and thus obtain a prediction for the desired unknown quantity. This could be the most likely solution or the solution otherwise optimal in some sense, e.g., minimizing the expected square error. Using this approach, it is possible to counter the errors introduced by inferior hardware, or for high-quality devices push the boundaries of what is possible.

While image deconvolution is a long-standing problem, and numerous methods exist for both the blind and non-blind case, the work presented in this thesis was able to improve on the state of the art in both cases. Later chapters will demonstrate that we are successful both with maximum a posteriori (MAP) based approaches and with using neural networks to learn to automatically solve the problem of deconvolution.

The outline of this thesis is as follows: after introducing the necessary fundamentals in Chapter 1, we treat the problem of *non-blind correction of optical aberrations* in Chapter 2. As explained above, optical aberrations cause an unwanted reduction of image quality. A calibration step encodes the spatially-varying aberrations of the optical system. The original image is then reconstructed by solving a MAP problem using a prior on the distribution of gradients in natural images. The results compare favorably to existing approaches for correcting optical aberrations. Additionally, the reconstruction procedure can include demosaicing, and it turns out that treating demosaicing and deconvolution jointly is superior to performing both tasks separately.

Next, the approach in Chapter 3 dispenses with the calibration step and performs *blind correction of optical aberrations*. Instead of measuring the lens error for every combination of lens, camera, and setting of the lens, the aberration can be inferred directly from the image. This is the first time blur induced by optical aberrations is corrected *blindly*. The proposed approach modifies an existing blind deconvolution method based on MAP and includes domain-specific information to reduce the number of unknown parameters. The class of spatially-varying blurs to be inferred is restricted to a basis of physically plausible lens blurs. Furthermore, blur that is different across multiple color channels can be estimated.

Chapter 4 returns to the basic deconvolution problem with known spatially invariant blur and pushes the state of the art by *learning non-blind deconvolution*. An initial preprocessing step reduces the blur by performing a regularized inversion in Fourier space, a step which is also common in previously existing deconvolution methods. A large neural network is trained on artificially generated training examples to learn the mapping from the preprocessed image back

to the uncorrupted original. The trained network achieves state-of-the-art deconvolution results, however, it is specific to a single blur kernel. It is straightforward to include other steps from the imaging pipeline in the reconstruction process, for example demosaicing, as demonstrated on a real-world example.

After the success of a learning method in the non-blind case, the next logical step is to use neural networks for *learning blind deconvolution*, which is presented in Chapter 5. Inspired by existing hand-crafted blind deconvolution methods, a deep neural network "unrolls" the blur estimation by using both conventional layers and layers specific to the problem. The neural network toolbox designed for this procedure is available as open-source software and is described in Appendix B. The proposed method is applicable to both spatially invariant and spatially-varying blurs. After training, the algorithm shows competitive performance on generic blurry images and is superior when trained on specific image categories, where the learning approach can play to its strengths.

CHAPTER 1

Fundamentals

Photography is the process of collecting light which has been emitted or reflected by objects in a scene. As is well-known from physics, light is electro-magnetic radiation that is described by Maxwell's equations.

The electric and magnetic field strengths present in photography are small enough such that the polarization response of media (the density of induced electric and magnetic dipole moments) can be assumed to be linear. This ignores effects of nonlinear optics, for example the frequency doubling used in green laser pointers to create green light from an infrared laser diode. However, it enables us to model our optical system as a *linear* system, which means the superposition principle holds for all electric and magnetic fields. Assuming incoherent light, i.e., electromagnetic waves with random phase differences, this also holds true for the light intensities as measured by the camera.

Ideally, in imaging, light from a single point in the scene should be mapped to a single position on the sensor, excluding cases where for artistic purposes the opposite may be desired. Because of imperfections of the optical system, or movement of the camera during exposure, this is often not the case. Therefore, because of the stated linearity of light, we model the general image formation process as

$$\mathbf{y} = K\mathbf{x} + \mathbf{n}. \tag{1.1}$$

where $\mathbf{x}$ is the vector of the intensities of pixels in the image we would measure in the absence of any errors (a.k.a. ground truth), either for a single color or for an averaged grayscale value. For simplicity, we represent two-dimensional images (or three-dimensional if we consider color) reshaped as vectors. Also, this is a discrete approximation to the continuous signal of the scene. The recorded image $\mathbf{y}$ is obtained after transformation by the matrix K and addition of the measurement noise $\mathbf{n}$, which could optionally also be dependent on the values of the true signal $\mathbf{x}$.

In this work, we will address corruptions K that can be described as a convolution *locally*, we distinguish either *stationary* (Section 1.1.1) or *spatially-varying* (see Section 1.1.2) convo-

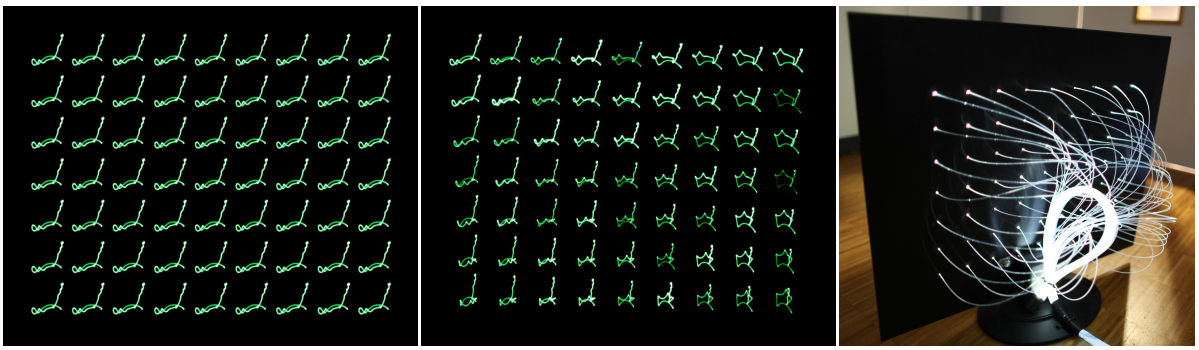


Figure 1.1.: *Left:* Image of a grid of light points with stationary motion blur (artificially created). *Middle:* Photo of the same grid with spatially-varying motion blur (non-artifical). *Right:* Grid of light points for illustration of camera shake (rear view).

lutions. The task is to recover $\mathbf{x}$ from a given observation $\mathbf{y}$. If K is known, we call the problem *non-blind deconvolution*, otherwise *blind deconvolution*.

In the following, we introduce the fundamentals necessary for removing blur from images. First, in Section 1.1, we state the mathematical properties of blur in photos. Next, we introduce methods to reconstruct $\mathbf{x}$ when the aberration is *known* (Section 1.2) and when it is *unknown* (Section 1.3). Lastly, in Section 1.4, we explain the machine learning tools we use to go beyond the established image reconstruction methods.

1.1. Mathematical problem description

To correct for aberrations in images, it is important to be able to describe these aberrations mathematically. In the context of this work, we always work in the domain of discrete pixels in an image (e.g., taken by the sensor of a digital camera), which is an approximation to the continuous domain of the electromagnetic waves which produced this signal.

1.1.1. Stationary convolutions

If the blur is independent of the position within the image, it can be described by a stationary convolution. An example would be a pure translation of the camera parallel to a scene that lies within a single plane, as illustrated in Fig. 1.1 on the left. Mathematically, a convolution is defined as

$$y[n] = (k * x)[n] = \sum_{m=-\infty}^{\infty} k[m]x[n-m] = \sum_{m=-\infty}^{\infty} k[n-m]x[m] \tag{1.2}$$

with functions y, k and x, defined on the set of integers $\mathbb{Z}$, where $y[n]$ denotes the value of y at n. For simplicity, we only show the math for a one-dimensional convolution. Intuitively, if k is only non-zero near 0, then $y[n]$ is a weighted average of the values of x near n, weighted by k. Figure 1.2 shows examples of convolutions in two dimensions, applied to an image.

Sharp image | Convolved with Gaussian blur | Convolved with motion blur

Figure 1.2.: Example of an image corrupted by a stationary convolution.

When working on finite domains, there are two noteworthy cases of performing the convolution:

Valid: The finite input vector $\mathbf{x}$ of size n_x is blurred by a kernel $\mathbf{k}$ of size n_k, resulting in an output $\mathbf{y}$ with size $n_x - n_k + 1$. This is the typical setting in photography, which is physically *valid*. A pixel on the sensor may receive light from an object outside of the field of view because of a blur with non-zero extent. Denoting the i-th element of a vector $\mathbf{v}$ as v_i, we define

$$\mathbf{y} = \mathbf{k} \underset{\text{valid}}{*} \mathbf{x} \quad \text{where} \quad y_n = \sum_{m=1}^{n_k} k_m x_{n+n_k-m}. \tag{1.3}$$

Circular: The discrete input signal $\mathbf{x}$ is assumed to be periodic with period n_x, this means

$$\mathbf{y} = \mathbf{k} \underset{\text{circ}}{*} \mathbf{x} \quad \text{where} \quad y_n = \sum_{m=1}^{n_k} k_m x_{((n-m) \bmod n_x)+1}, \tag{1.4}$$

or creating a circulant matrix K from the entries of $\mathbf{k}$,

$$\mathbf{y} = K\mathbf{x}. \tag{1.5}$$

If we perfom the computation as described in this equation, it requires $O(n_x n_k)$ multiplications and additions, which can be expensive especially for two-dimensional convolutions. We note that the discrete Fourier matrix F diagonalizes a circulant matrix $C = F^{\mathsf{H}} \operatorname{Diag}(F\mathbf{c})F$, where "Diag" creates a diagonal matrix from a vector, $\mathbf{c}$ is the first column of C, and F^{H} is the Hermitian transpose of F. Equation (1.5) can be rewritten as the discrete version of the convolution theorem,

$$\mathbf{y} = \mathbf{k} * \mathbf{x} = F^{\mathsf{H}} \left(F\mathbf{k} \odot F\mathbf{x} \right), \tag{1.6}$$

since $\operatorname{Diag}(\mathbf{a})\mathbf{b} = \mathbf{a} \odot \mathbf{b}$ and assuming that $\mathbf{k}$ is zero-padded to the length of $\mathbf{x}$. The symbol $\odot$ denotes the Hadamard product.

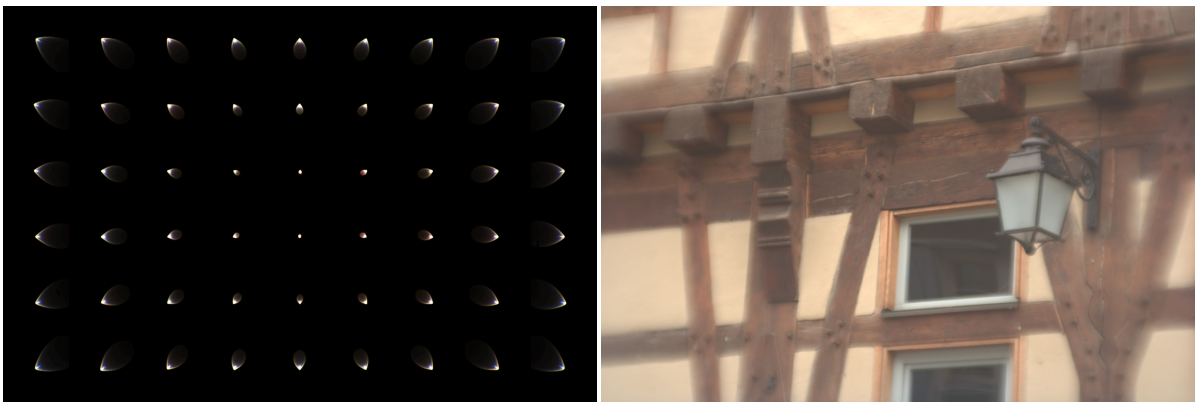


Figure 1.3.: *Left:* Spatially-varying point spread function (PSF) of a self-built 120 mm lens. The measurement was performed as described in Section 2.5. *Right:* Photo taken with this lens.

Using fast Fourier transforms (FFTs), the discrete Fourier matrix can be multiplied with a vector of length n in time $O(n \log n)$, which means that the FFT applied to the convolution theorem enables us to calculate $\mathbf{y}$ in time $O(n_x \log n_x)$. For large blurs the dependence $\log n_x$ of this approach versus n_k of Eq. (1.4) results in lower computation times. Additionally, as we will see in Section 1.2, the Fourier representation is especially advantageous when trying to *invert* convolutions, i.e., reconstructing the original sharp image.

Revisiting the different convolution types, and introducing an operator C_y which appropriately crops a vector by the size of the kernel minus 1 in every dimension, and furthermore an operator C_k^T which pads to the size of $\mathbf{x}$, we obtain:

Valid:

$$\mathbf{y} = C_y \left(\mathbf{k} * \mathbf{x}\right) = C_y F^\mathsf{H} \left(F C_k^\mathsf{T} \mathbf{k} \odot F\mathbf{x}\right). \tag{1.7}$$

Circular:

$$\mathbf{y} = \mathbf{k} * \mathbf{x} = F^\mathsf{H} \left(F C_k^\mathsf{T} \mathbf{k} \odot F\mathbf{x}\right). \tag{1.8}$$

Note that the transpose of a cropping matrix C is a zero-padding matrix C^T.

1.1.2. Spatially-varying blur

Looking at the middle part of Fig. 1.1, a photo of a point grid corrupted by camera-shake, we see that the assumption of stationarity does not always hold for camera shake. This is clear if we consider a rotation of the camera along the axis perpendicular to the image plane: a point in the upper right corner of the image will be shifted in the opposite direction of a point in the lower right corner. However, two points in spatial proximity within the scene will be transformed similarly, this means then the blur varies *smoothly* also across the image plane.

A further example, lens aberrations, exhibits similar behavior, see Fig. 1.3. Again, the corruption is *locally* a convolution, but varies smoothly across the image plane. In Section 2.3

we explain the aberrations that cause the incoming light rays to deviate from a single focal point.

We have seen that, in the case of linear optics, Eq. (1.1) is sufficient to describe the image formation process. However, the computational complexity in the general case is $O(n^2)$, which is infeasible for a large number of pixels n in the ground truth image. Noting that corruptions like camera shake or lens aberrations are still convolutions *locally*, but vary *smoothly*, it is possible to approximate them with the efficient filter flow (EFF) framework [Hir+10]. Its basic idea is to cover the image with overlapping patches, to each of which a blur kernel is assigned.

In this framework, the forward operation is modeled as

$$\mathbf{y} = \sum_{r=1}^{R} \mathbf{k}^{(r)} * \left(\mathbf{w}^{(r)} \odot \mathbf{x}\right), \tag{1.9}$$

where $\mathbf{x}$ denotes the ideal image and $\mathbf{y}$ is the image degraded by optical aberrations. Here $\mathbf{x}$ and $\mathbf{y}$ are discretely sampled images, i.e., $\mathbf{x}$ and $\mathbf{y}$ are finite-sized vectors whose entries correspond to pixel intensities, $\mathbf{w}^{(r)}$ are weighting vectors that mask out all of the image $\mathbf{x}$ except for a local patch by Hadamard multiplication. The r-th patch is convolved appropriately with a local blur kernel $\mathbf{k}^{(r)}$ in two dimensions. All blurred patches are summed up to form the degraded image. The more patches are considered (R is the total number of patches), the better the approximation to the true non-uniform PSF [Hir12]. Note that the patches defined by the weighting vectors $\mathbf{w}^{(r)}$ usually overlap to yield *smoothly* varying blurs. The weights are chosen such that they sum up to one for each pixel, i.e., $\sum_{r=1}^{R} \mathbf{w}^{(r)} = \mathbf{1}$, where $\mathbf{1}$ denotes a vector of ones. This means that, depending on the weighting vectors, every pixel can have a different blur, a linear combination of the given blur kernels in its neighborhood. Hirsch et al. [Hir+10] show that this forward model can be computed efficiently by making use of the short-time Fourier transform, writing it either in terms of $\mathbf{x}$ or $\mathbf{k}$:

$$\mathbf{y} = K\mathbf{x} = C_y \sum_{r=1}^{R} C_r^{\mathsf{T}} F^{\mathsf{H}} \operatorname{Diag}\left(F C_k^{\mathsf{T}} \mathbf{k}^{(r)}\right) F C_r \operatorname{Diag}\left(\mathbf{w}^{(r)}\right) \mathbf{x}, \tag{1.10}$$

$$\mathbf{y} = X\mathbf{k} = C_y \sum_{r=1}^{R} C_r^{\mathsf{T}} F^{\mathsf{H}} \operatorname{Diag}\left(F C_r \operatorname{Diag}\left(\mathbf{w}^{(r)}\right) \mathbf{x}\right) F C_k^{\mathsf{T}} S_r \, \mathbf{k}. \tag{1.11}$$

Adopting the notation of [Hir+10], the matrices C_k and C_r are appropriately chosen cropping matrices, and S_r selects $\mathbf{k}^{(r)}$ from the kernel stack $\mathbf{k}$. The most expensive operation in this formulation is the FFT ($O(n \log n)$), whereas the other operations, including cropping and zero-padding, are possible to perform in linear time.

From the operator notations in Eqs. (1.10) and (1.11) it is directly clear how to obtain the transpose operators K^{T} and X^{T}. These are required for gradient calculations, e.g. when solving deconvolution as an optimization problem. For example, the gradient of the log-likelihood function $E(\mathbf{x}) = \|K\mathbf{x} - \mathbf{y}\|^2$, appearing in the next section, would be

$$\frac{\partial E}{\partial \mathbf{x}} = 2K^{\mathsf{T}} \left(K\mathbf{x} - \mathbf{y}\right). \tag{1.12}$$

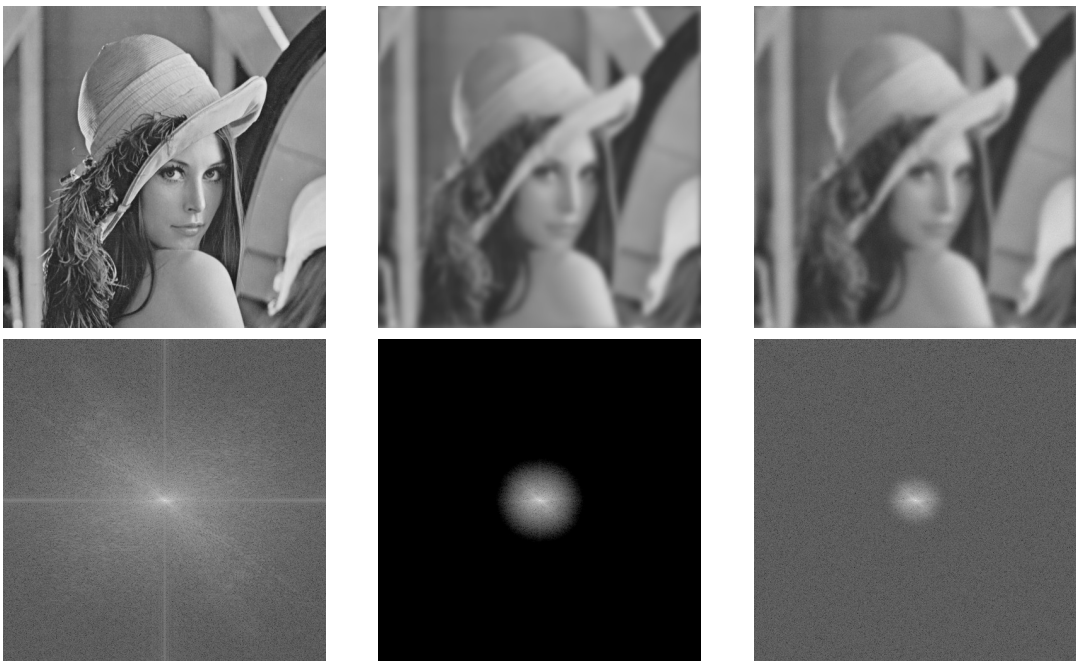


Figure 1.4.: Effect of a blur on the power spectrum. Top row is the image in real space, bottom row the power spectrum on a logarithmic scale. *Left:* Original image. *Center:* Blurred with a Gaussian blur. *Right:* Blurred and with 1% Gaussian noise. Note how the blur and the noise corrupt high frequencies.

For some cases of spatially-varying blur, alternative representations could be used. For example for camera shake, the blurry image could be described by a superposition of homographies of the true scene [Why+10]. For lens aberrations, the first few Zernike polynomials approximate the phase of the incoming light well, which is nonlinearly related to the blur throughout the sensor plane [BW99].

1.2. Non-blind deconvolution methods

It is often possible to obtain knowledge about the blur of an image a priori. For example, the aberrations of a lens are fixed, and even motion blur could be measured with accelerometers and gyroscopes [Jos+10].

Given the blur, trying to invert convolution is still an underdetermined problem: we see the effect of a convolution in Fig. 1.4, where the blur lowers the high frequencies. After adding a small amount of noise, the signal-to-noise ratio (SNR) becomes close to 0 in a large part of the spectrum. This is why prior knowledge has to be included to reconstruct the original image from the remaining signal.

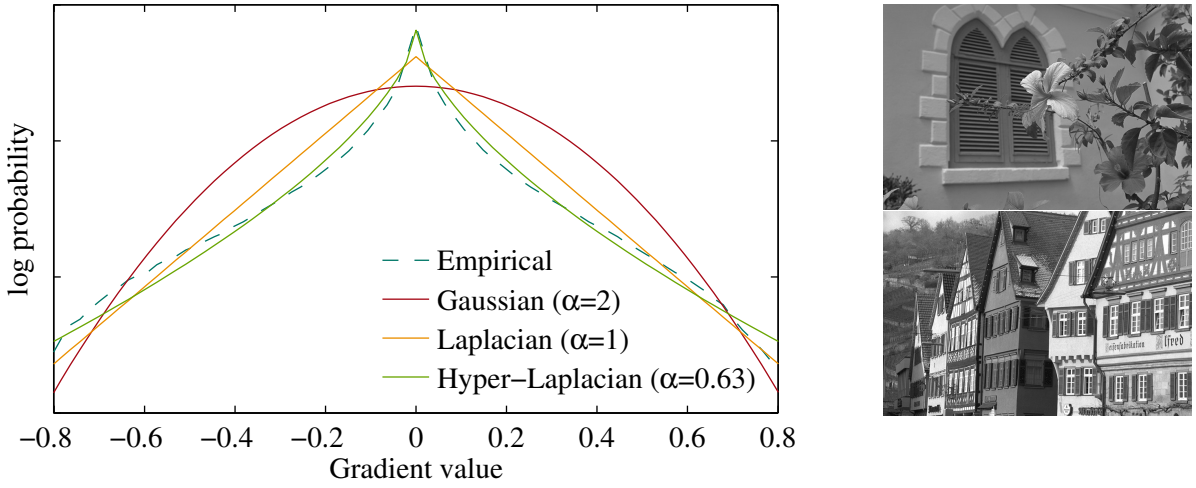


Figure 1.5.: *Left:* Distribution of horizontal and vertical gradients in natural images (grayscale). The empirical distribution generated from the Kodak image data set[1]and fits of three analytic distributions $p(x) \propto \exp(-c|x|^{\alpha})$ are shown. *Right:* Two examples from the Kodak image data set.

1.2.1. Wiener deconvolution

In the simplest case, both the signal and the noise, which are additive, are assumed to be *stationary linear* stochastic processes, i.e., they have shift-invariant linear joint probability distribution. This also means that the processes' spectral properties are sufficient to describe their properties. It can then be shown that the minimum mean square error (MMSE) estimator $\tilde{\mathbf{x}}$ of the original signal is [GW02]

$$\tilde{\mathbf{x}} = F^{\mathsf{H}} \frac{\overline{F\mathbf{k}} \odot F\mathbf{y}}{|F\mathbf{k}|^2 + \mathbf{P}_n/\mathbf{P}_x}, \tag{1.13}$$

assuming circular convolution and that $\mathbf{k}$ is appropriately zero-padded to the size of $\mathbf{x}$. Here the division, the absolute value and the conjugation (indicated by the bar above a symbol) are pointwise operations. Additionally, we introduced the frequency dependent power spectral density of the noise $\mathbf{P}_n$ and the signal $\mathbf{P}_x$, i.e., the Fourier transform of their respective auto-correlations. Since this is the MMSE estimator, given our assumptions, there cannot be a better estimator in the square error sense. Comparing Eq. (1.13) with Eq. (1.8), we see that $\overline{F\mathbf{k}}/(|F\mathbf{k}|^2 + \mathbf{P}_n/\mathbf{P}_x)$ acts as a convolution filter on the blurry observation $\mathbf{y}$. This filter is also known as the Wiener filter.

1.2.2. Tikhonov regularization

Contrary to the previous assumptions, images are typically *not* generated by a linear stationary process, which means that the Wiener deconvolution of Eq. (1.13) can only be an approximation.

[1] `http://r0k.us/graphics/kodak/`

Bayes' rule tells us that the posterior probability density function (PDF)

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})} \tag{1.14}$$

of $\mathbf{x}$ given an observation $\mathbf{y}$ is proportional to the likelihood $p(\mathbf{y}|\mathbf{x})$ and the prior $p(\mathbf{x})$, normalized by $p(\mathbf{y})$. From Fig. 1.5 we see that a Gaussian distribution is a rough approximation to the distribution of gradients in natural images (while a hyper-Laplacian would be better suited, it is also less tractable [KF09]). This provides a prior $p(\mathbf{x}) \propto \exp\left(-(\|G\mathbf{x}\|^2)/(2\sigma_{Gx}^2)\right)$ with standard deviation σ_{Gx} and the gradient matrix G. Assuming Gaussian noise, the likelihood is $p(\mathbf{y}|\mathbf{x}) \propto \exp\left(-(\|\mathbf{k} * \mathbf{x} - \mathbf{y}\|^2)/(2\sigma_n^2)\right)$ with standard deviation σ_n. Then the PDF of $\mathbf{x}$ conditioned on $\mathbf{y}$ is

$$p(\mathbf{x}|\mathbf{y}) \propto \exp\left(-\frac{\|\mathbf{k} * \mathbf{x} - \mathbf{y}\|^2}{2\sigma_n^2}\right) \cdot \exp\left(-\frac{\|G\mathbf{x}\|^2}{2\sigma_{Gx}^2}\right). \tag{1.15}$$

In the case of a Gaussian distribution the mean and the maximum (a.k.a. mode) are at the same point, the MMSE estimator consequently is the MAP solution. Since the logarithm is a strictly increasing function that does not affect the position of extrema, we can instead determine the minimum of the negative logarithm of the posterior,

$$\|\mathbf{y} - \mathbf{k} * \mathbf{x}\|^2 + \frac{\sigma_n^2}{\sigma_{Gx}^2}\|G\mathbf{x}\|^2, \tag{1.16}$$

Because only L2 norms appear in this objective, in the case of circular convolution there exists a closed-form solution [CL09]

$$\mathbf{x} = F^{\mathsf{H}} \frac{\overline{F\mathbf{k}} \odot F\mathbf{y}}{|F\mathbf{k}|^2 + \sum_i \frac{\sigma_n^2}{\sigma_{Gx}^2}|F\mathbf{g}_i|^2}, \tag{1.17}$$

assuming that $\|G\mathbf{x}\|^2$ can be expressed as the sum $\sum_i \|\mathbf{g}_i * \mathbf{x}\|^2$ of convolutions with gradient filters $\mathbf{g}_i$ zero-padded to the size of $\mathbf{x}$. Note that this result is a special case of Eq. (1.13), where $\mathbf{P}_n = \sigma_n^2\mathbf{1}$ and $\mathbf{P}_x = \sum_i \sigma_{Gx}^2/|F\mathbf{g}_i|^2$.

Unfortunately, the operator C_y that would be necessary for valid convolutions is not diagonal in Fourier space, which breaks the fast inversion by Fourier division. However, this is the convolution type appearing in photography applications. To make it applicable, the blurry image $\mathbf{y}$ is tapered in the area of the edges such that there is a smooth transition between opposite edges, e.g., by multiplying the border regions with a window function smoothly going from 1 to 0. [KF09]

1.2.3. Recent deconvolution methods

The methods mentioned above use an explicit image prior (e.g., the Wiener filter incorporates the power spectra of signal and noise) to reconstruct the original signal, and many other methods with more sophisticated explicit image priors are also available. A different class of methods instead tries to first apply the simple deconvolution from Eq. (1.17) and, in a second step, remove the artifacts created from this imperfect reconstruction. We will discuss these two classes of algorithms and their relation to our proposed method in more detail in Section 4.2.

1.3. Blind deconvolution methods

Lenses have many different settings, including aperture, zoom and focus, and inertia sensors for motion blur add additional hardware constraints, making it difficult to measure the exact lens or motion blur. Thus, it is advantageous to infer both the true image $\mathbf{x}$ and the blur kernel from the recorded image, a problem known as *blind deconvolution*.

In this case, the problem becomes underdetermined even without noise, because many combinations of blur kernel and reconstructed image could explain the observation. From a probabilistic perspective, every pair of kernel $\mathbf{k}$ and true image $\mathbf{x}$ is associated with a posterior probability

$$p(\mathbf{x},\mathbf{k}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x},\mathbf{k})p(\mathbf{x})p(\mathbf{k})}{p(\mathbf{y})} \propto p(\mathbf{y}|\mathbf{x},\mathbf{k})p(\mathbf{x})p(\mathbf{k}), \tag{1.18}$$

analogous to Eq. (1.14). When solving blind deconvolution, we are interested in an estimated solution $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{k}}$ optimal under a certain loss, i.e., the solution that minimizes the expected error $\int E(\tilde{\mathbf{x}},\tilde{\mathbf{k}},\mathbf{x},\mathbf{k})p(\mathbf{x},\mathbf{k}|\mathbf{y})dxdk$ with the $\mathbf{x}$- and $\mathbf{k}$-space volume elements dx and dk, and a given loss $E(\tilde{\mathbf{x}},\tilde{\mathbf{k}},\mathbf{x},\mathbf{k})$.

For a 0–1 loss, which means that the cost is 0 for the estimated solution equal to the ground truth, and 1 for all wrong solutions, the optimal estimator is the MAP estimator [Mur12] introduced in Section 1.2.2, $\operatorname{argmax}_{\mathbf{x},\mathbf{k}} p(\mathbf{x},\mathbf{k}|\mathbf{y})$. It is one of the two common approaches for blind deconvolution. The other approach also considers a 0–1 loss, but only with respect to $\mathbf{k}$, and marginalized over $\mathbf{x}$, i.e., $\operatorname{argmax}_{\mathbf{k}} p(\mathbf{k}|\mathbf{y}) = \operatorname{argmax}_{\mathbf{k}} \int p(\mathbf{x},\mathbf{k}|\mathbf{y})dx$. In the following we will introduce both approaches.

1.3.1. MAP approaches

The maximum a posteriori approach finds the maximum of Eq. (1.18). As above, this is equivalent to the minimum of its negative logarithm. Therefore, we need to minimize

$$\|\mathbf{k} * \mathbf{x} - \mathbf{y}\|^2 + g_x(\mathbf{x}) + g_k(\mathbf{k}) \tag{1.19}$$

where $\|\mathbf{k} * \mathbf{x} - \mathbf{y}\|^2$ is the log-likelihood term under the assumption of Gaussian noise, and $g_x(\mathbf{x})$ and $g_k(\mathbf{k})$ are derived from $p(\mathbf{x})$ and $p(\mathbf{k})$, acting as regularizers in the new objective.

The regularizer $g_x(\mathbf{x})$ penalizes blurry images, for example by using a sparse prior on the gradients [SJA08], which prefers having a single large gradient instead of a smooth transition with many small gradients. While Shan et al. [SJA08] fit to the distribution of gradients of natural images, Xu et al. [XZJ13] are more successful with a sparsity-promoting L0 prior, suggesting that it is more important to be discriminative between sharp and blurry images than to use a generative distribution for sharp images. Similarly, the normalized sparsity measure from [KTF11] is an improper prior since the integral of all prior values is not finite.

Another class of methods only uses implicit regularization for the image, without explicitly defining a regularization term $g_x(\mathbf{x})$. For example, this can be achieved by nonlinear filtering of the image such that edges are emphasized and artifacts are suppressed. Cho and Lee [CL09] apply a shock-filter — the inversion of a diffusion process — and a bilateral filter to achieve this, as can be seen in Figs. 1.7 and 3.5. Another method [XJ10] defines a heuristic procedure to

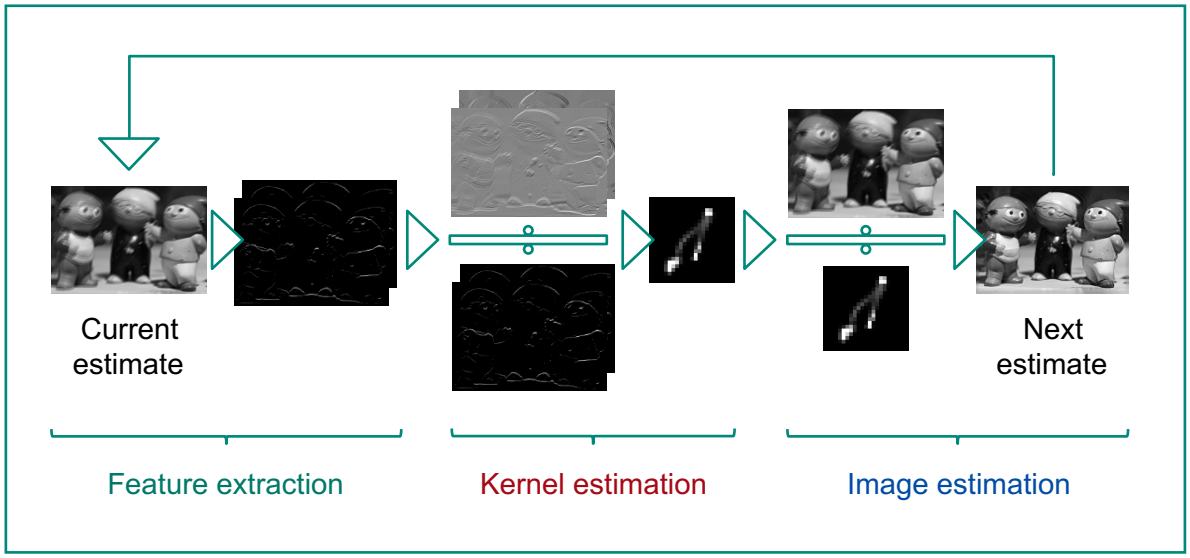


Figure 1.6.: Blind deconvolution procedure for MAP approaches. PSF and image estimates are refined by repeating the three steps *feature extraction*, *kernel estimation* and *image estimation*. The feature extraction shown here creates two images in gradient space. The kernel and image estimation are often performed by division in Fourier space, hence the division symbol ÷.

select gradients that are believed be most informative to estimate the blur kernel. Our proposed method in Chapter 5 also falls into the category of implicit regularization, but *learns* the optimal regularization operation.

Having chosen a regularization procedure, MAP approaches iterate between the following three steps, also shown in Fig. 1.6:

Feature extraction: The feature extraction creates image representations that are useful for kernel estimation, using one of the approaches mentioned above. This intermediate feature image is sometimes also called unnatural image representation [XZJ13]. Compared to the sharp image, it is dominated by step-edges and contains less details. The feature extraction is often only performed in gradient space. Figure 1.7 shows a comparison of these intermediate representations, including a direct regularization method [CL09], a marginalization approach [Fer+06] explained below, and a method with implicit regularization [XZJ13] (the latter two reconstructed from gradients).

Kernel estimation: The next step takes the extracted features to estimate the convolution kernel. The prior on the kernel is often an analytically tractable Gaussian prior, with $g_k(\mathbf{k}) \propto \|\mathbf{k}\|^2$ [CL09], or a sparsity-inducing L1 prior $g_k(\mathbf{k}) \propto \|\mathbf{k}\|_1$ [SJA08]. It is also possible to combine a prior with a heuristic approach, e.g., support detection of the kernel [XJ10], to detect the regions where the kernel is non-zero. It is recommended to work on images in gradient space, since in this case the optimization problem converges faster [CL09].

Cho and Lee [CL09] Fergus et al. [Fer+06] Xu et al. [XZJ13]

Figure 1.7.: Intermediate image representations for kernel estimation. These images are intermediate results of blind deconvolution methods analogous to Fig. 1 in [XZJ13]. Fergus' result has been reconstructed from gradients.

Image estimation: Lastly, the sharp image is estimated using the current kernel estimate, which will be the starting point for the next feature extraction step.

The three steps are repeated several times on different image scales, starting on a coarse scale with a downsampled version of the blurry image. On the coarser scale the blur is also smaller and easier to infer. After upsampling, the previous solution gives a better initialization for the next scale.

An important consideration of blind deconvolution is to make it *fast*, i.e., it should ideally not take more than a few seconds to deblur a standard-sized photo. Usually this requires to solve Eq. (1.19) in a few steps, see Eq. (1.17), instead of performing many iterations with an optimizer. While Eq. (1.17) works only for some regularization terms, e.g., $g_x(\mathbf{x}) \propto \|\mathbf{x}\|^2$, procedures for non-L2 terms exist [KF09; XZJ13].

1.3.2. Marginalization approaches

A toy example for Eq. (1.18) is shown in Fig. 1.8 on the right. While, in this case, the distribution is still simple with both the kernel and image being one-dimensional and only Gaussian priors, in practice local minima pose problems. Additionally, depending on the prior, the trivial solution where the blur is a delta peak can become the global optimum [WZ13], even though regularization schemes as mentioned above can alleviate the problem.

Levin et al. [Lev+09] proposed to instead marginalize over $\mathbf{x}$ to obtain $p(\mathbf{k}|\mathbf{y}) = \int p(\mathbf{x},\mathbf{k}|\mathbf{y})dx$ and find the $\mathbf{k}$ that maximizes

$$\max_{\mathbf{k}} p(\mathbf{k}|\mathbf{y}) \equiv \min_{\mathbf{k}} -2 \log p(\mathbf{y}|\mathbf{k})p(\mathbf{k}). \tag{1.20}$$

Afterwards $\mathbf{x}$ can be obtained by non-blind deconvolution. In the toy example, the marginal is plotted on the left. The advantage of this approach is the reduced dimensionality of the optimization problem. However, marginalizing over $\mathbf{x}$ is difficult and can only be done approximately, e.g., with variational Bayes (VB) [Lev+11]. As noted by Wipf and Zhang [WZ13],

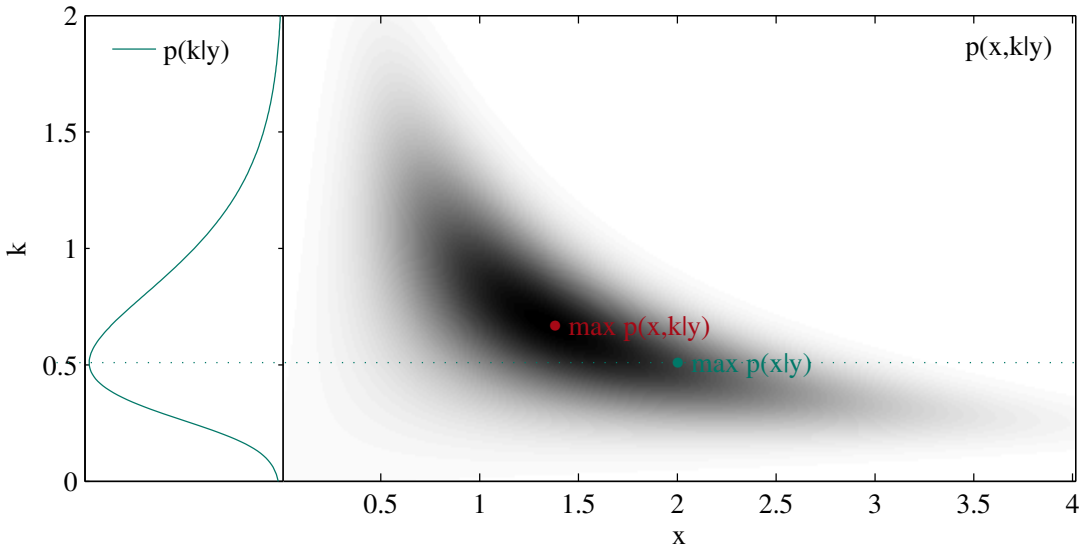


Figure 1.8.: Illustration of a toy example with $p(x,k|y) \propto \exp\left(-\frac{(k\cdot x-1)^2}{2\cdot 0.1^2} - \frac{x^2}{2\cdot 2^2} - \frac{k^2}{2\cdot 0.5^2}\right)$, and its marginals. The mode of the full distribution is different from the mode of its marginal $p(k|y)$.

the assumptions behind VB are actually not ideal for blind deblurring (e.g., factorization of the distribution). The reported superiority over MAP [Lev+11] seems to be mainly due to avoiding local minima, and the priors should be such that they discriminate well between blurry and sharp images, rather than being generative for sharp images. For a more in-depth review, see [WZ13].

1.4. Neural networks

Another approach does not employ an explicit image prior $p(\mathbf{x})$ but *learns* an estimator from samples of the ground truth $\mathbf{x}$ and the corresponding observation $\mathbf{y}$ drawn from $p(\mathbf{x})$ and $p(\mathbf{y}|\mathbf{x})$, respectively. To draw samples from $p(\mathbf{x})$, a large collection of photos can be used. We obtain samples from $p(\mathbf{y}|\mathbf{x})$ by applying the corruption process artificially. A learning algorithm can then obtain information about the intractable distribution $p(\mathbf{x})$ of images implicitly. Artificial neural networks (NNs) have demonstrated to be successful for this task in the context of image processing [BSH12c], and we will also employ them in this thesis.

While the origins of NNs go back to the 1940s [MP43], they are currently seeing a resurge in popularity, thanks to advances in neural network architectures and faster hardware. NNs are broadly applicable to many different learning tasks and come in many different variants. In the context of this work, NNs mean *feed-forward* neural networks applied to *supervised* learning tasks. For a broader picture of NNs, we refer the interested reader to [Sch14].

In our setting, training a NN on a data set is a special case of nonlinear regression, where the function to be learned is a concatenation of elementary functions

$$f(\mathbf{i}) = f_n\left(\ldots f_2\left(f_1\left(\mathbf{i},\mathbf{p}_1\right),\mathbf{p}_2\right)\ldots,\mathbf{p}_n\right). \tag{1.21}$$

In the context of neural networks, the building blocks $f_i(\mathbf{i}, \mathbf{p}_i)$ of vector-valued functions with input $\mathbf{i}$ and parameters $\mathbf{p}_i$ are called *layers*. Given a loss function $E(f(\mathbf{i}), \mathbf{t})$, e.g. a square loss $\|f(\mathbf{i}) - \mathbf{t}\|^2$ with known input and target pairs $\mathbf{i}$ and $\mathbf{t}$, parameters $\mathbf{p}_i$ can be learned such that they minimize the expected loss. An architecture often used is the multilayer perceptron (MLP), where the concatenated functions alternate between linear transformations and nonlinearities, e.g.,

$$f(\mathbf{i}) = \mathbf{b}_3 + W_3 \tanh(\mathbf{b}_2 + W_2 \tanh(\mathbf{b}_1 + W_1 \mathbf{i})), \tag{1.22}$$

where $f_i(\mathbf{i}, \mathbf{p}_i) = \tanh(\mathbf{b}_i + W_i \mathbf{i})$ with weight matrix W_i and bias vector $\mathbf{b}_i$ is commonly called a hidden layer.

Recently, it has been shown that even this simple architecture can achieve state-of-the-art results for denoising [BSH12c] or digit recognition [Cir+10]. The key difference to older architectures is the use of *deep* architectures, meaning four or more of the mentioned layers. This is also the architecture we use for our non-blind deconvolution algorithm in Chapter 4. For other challenging computer vision problems like object recognition, the NN is a more general layered computation consisting of convolutional layers, pooling layers, normalization layers, or rectifying linear units and other layer types [Le+12]. The learning method for blind deconvolution proposed in Chapter 5 works in the same spirit.

1.4.1. Training

Commonly, NNs are trained with *back-propagation* [Wer74]. Given a training example with input and expected output, we can calculate the gradient $\mathbf{g}_i$ of the loss with respect to a certain parameter vector p_i by applying the chain-rule:

$$\mathbf{g}_i = \frac{\partial E}{\partial \mathbf{p}_i} = \frac{\partial E}{\partial f_n(\mathbf{i})} \frac{\partial f_n(\mathbf{i})}{\partial f_{n-1}(\mathbf{i})} \frac{\partial f_{n-1}(\mathbf{i})}{\partial f_{n-2}(\mathbf{i})} \cdots \frac{\partial f_i(\mathbf{i})}{\partial \mathbf{p}_i}. \tag{1.23}$$

We see that the derivative of a layer i includes all derivatives $\partial f_j(\mathbf{i}) / \partial f_{j-1}(\mathbf{i})$ of subsequent layers. The intermediate results can be reused from layer to layer, which means the derivative is *back-propagated*. Next, every layer is updated by changing the parameters as

$$\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} + \Delta \mathbf{p}_i^{(t)}. \tag{1.24}$$

One possible approach to adapt the network's parameters in order to minimize the cost on a training set is the stochastic gradient descent (SGD) algorithm [Bot91]. Every update is the scaled gradient

$$\Delta \mathbf{p}_i^{(t)} = -\eta \mathbf{g}_i^{(t)} \tag{1.25}$$

with learning rate η. Note that "stochastic" means that the update is calculated on a random selection of training examples. The convergence rate can be improved by adapting the learning rate layer-wise (e.g., dividing by a factor depending on the input dimension of each layer [LeC+98b]).

The momentum update rule [RHW86] includes not only the current gradient, but the past gradients with exponentially decaying importance, $\Delta \mathbf{p}_i^{(t)} = \sum_{\tau=0}^{t-1} \rho^\tau \mathbf{g}_i^{(t-\tau)}$. This smoothens random variations orthogonal to the dominant gradient direction.

Fast convergence rates and less tuning for different architectures are achieved by the ADAGRAD [DHS11] method. It normalizes the update with the root mean square (RMS) of all previous gradients, effectively setting a different learning rate for every parameter:

$$\Delta \mathbf{p}_i^{(t)} = -\eta / \sqrt{\sum_{\tau=1}^{t} \left(\mathbf{g}_i^{(\tau)}\right)^2} \cdot \mathbf{g}_i^{(t)}, \tag{1.26}$$

where all operations are performed element-wise. The diverging RMS of the past gradients causes the updates to go to zero over time, which may be desired to reduce overfitting on limited training data.

When near infinite training data is available, for example when sampling from a generative model, this behavior may be undesired. The ADADELTA [Zei12] method proposes to include past gradients in the calculation of the RMS with decaying importance,

$$\begin{aligned} \Delta \mathbf{p}_i^{(t)} = -\eta \frac{\mathrm{RMS}[\Delta \mathbf{p}_i]^{(t-1)}}{\mathrm{RMS}[\mathbf{g}_i]^{(t)}} \mathbf{g}_i^{(t)} \quad & \text{with } \mathrm{RMS}[\mathbf{g}]^{(t)} = \sqrt{E[\mathbf{g}^2]^{(t)} + \epsilon} \\ & \text{and } E[\mathbf{g}^2]^{(t)} = \rho E[\mathbf{g}^2]^{(t-1)} + (1-\rho)\mathbf{g}^{(t)2}. \end{aligned} \tag{1.27}$$

Additionally, it adds a speed-up term in the numerator, which becomes large when the previous updates are large.

The methods mentioned here are often performed on several training examples simultaneously, called mini-batches, where multiple gradients are combined to a single, more meaningful gradient. This can be advantageous because some operations in the neural network need less instructions if performed over multiple inputs, as compared to multiple executions on a single input. A good example is a single matrix-matrix multiplication instead of multiple matrix-vector multiplications.

To successfully train NNs, there are many other important design choices, for example number of hidden units, and "tricks", like the initialization of the parameters p_i, data normalization or layer-wise hyper-parameter settings discussed in [Ben12].

1.4.2. Toy example

Neural networks have proven to be successful for classification and regression tasks. How do they achieve this? To give some insights, we trained[2] a small MLP to solve a two dimensional classification task with two classes which are not linearly separable (see Fig. 1.9 on the left). The network has two input dimensions, one hidden layer with two dimensions, and two output classes, allowing us to plot the intermediary and final outputs in their full dimensionality. While commonly one hidden layer in an MLP means both a nonlinearity and an affine transformation, we here view the affine transformation as separate layers.

The network maps an input $\mathbf{i}$ to an output $\mathbf{o}$ where o_k is the probability of $\mathbf{i}$ belonging to a class k of K total classes. When providing labels t_k for the training data (1 if its true class is k, otherwise 0), we can train the network on the cross-entropy loss $-\sum_k^K t_k \ln(o_k)$ using

[2] We employed ConvNetJS (`http://github.com/karpathy/convnetjs`) for both training and visualization.

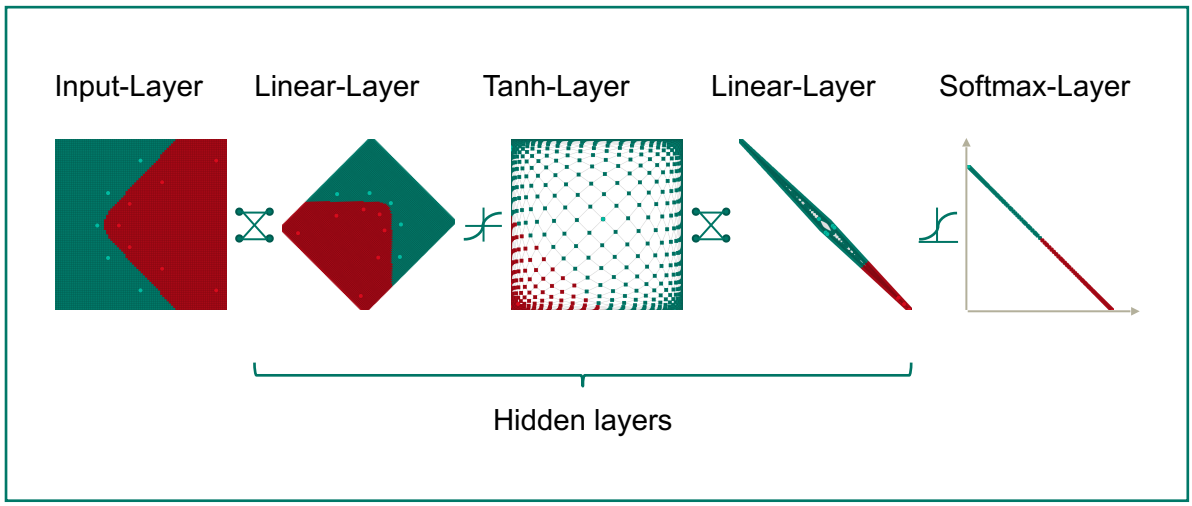


Figure 1.9.: Classification with a neural network. This toy example shows an MLP with two input dimensions, "one" hidden layer with two neurons (the linear layers are commonly discounted in the number of hidden layers) and two output classes. See Section 1.4.2 for details.

the training procedure of our choice, here SGD. While the network may converge to different parameter configurations, one successful outcome after training is a network that performs the following operations on the input:

1. *Linear layer:* The first hidden layer applies an affine transformation $\mathbf{o}_1 = \mathbf{b}_1 + W_1\mathbf{i}$. We see from Fig. 1.9 that the input plane is rotated such that the two directions which could separate the classes are aligned to the coordinate system of the feature space.

2. *Tanh layer:* Next, the data is nonlinearly transformed as $\mathbf{o}_2 = \tanh(\mathbf{o}_1)$, "straightening out" the region between the different classes, and making the two classes linearly separable.

3. *Linear layer:* A second affine transformation $\mathbf{o}_3 = \mathbf{b}_3 + W_3\mathbf{i}$ projects out the dimension along the line separating the two classes in the nonlinear feature space, since the network decided this dimension is not relevant for the classification decision. Also, the output of the linear layer is rotated relative to its input to be suitable for the subsequent softmax layer.

4. *Softmax layer:* The final output layer converts its input into the normalized probability of a certain point belonging to a particular class, i.e., $\mathbf{o} = \exp(\mathbf{o}_3)/\sum_{k=1}^{K}\exp(o_{3,k})$. The colored areas in Fig. 1.9 depict in red or green where the probability for class 1 or 2 is larger than 0.5, respectively. At the red end of the line the NN is certain that the corresponding point in the input layer belongs to class 1, at the green end that it belongs to class 2.

We see that the MLP transformed the nonlinearly separable input to a nonlinear feature space, where it is *linearly* separable, allowing a classification decision.

CHAPTER 2

Non-Blind Correction of Optical Aberrations

Taking a sharp photo at several megapixel resolution traditionally relies on high grade lenses. In this chapter, we present an approach to alleviate image degradations caused by imperfect optics. We rely on a calibration step to encode the optical aberrations in a space-variant point spread function and obtain a corrected image by non-stationary deconvolution. By including the Bayer array in our image formation model, we can perform demosaicing as part of the deconvolution.

The material of this chapter is based on the following publication:

[Sch+11] *C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. "Non-stationary correction of optical aberrations". In:* IEEE Int. Conf. Computer Vision. *2011.* DOI: 10.1109/iccv.2011.6126301

2.1. Introduction

In an ideal optical system as described theoretically by paraxial optics, all light rays emitted by a point source converge to a single point in the focal plane, forming a clear and sharp image. Departures of an optical system from this behavior are called aberrations, causing unwanted blurring of the image.

Manufacturers of photographic lenses attempt to minimize optical aberrations by combining several lenses. The design and complexity of a compound lens depends on various factors, e.g., aperture size, focal length, and constraints on distortions. Optical aberrations are inevitable and the design of a lens is always a trade-off between various parameters, including price. To correct these errors in software is still an unresolved problem.

Rather than proposing new designs for complicated compound lenses, we show that almost all optical aberrations can be corrected by digital image processing. For this, we note that optical aberrations of a linear optical system are fully described by their PSF. We will show

Figure 2.1.: Self-made photographic lens with one glass element only, mounted on a remote controlled platform to take photos in different angles.

Figure 2.2.: Image taken through self-made lens without and with lens correction.

how various optical aberrations encountered in real photographic lenses can be represented and approximated as PSFs in non-stationary convolutions, as mentioned in Section 1.1.2. For a given lens/camera combination, the parameters of the non-stationary convolution are estimated via an automated calibration procedure that measures the PSF at a grid covering the image. We also include demosaicing into our image reconstruction, because it fits naturally into our forward model. Our results surpass the previous state of the art.

Main contributions: We show how to reconstruct a full-color image, i.e., all three color channels at full resolution, given a raw image that is corrupted by various monochromatic and chromatic aberrations, and Bayer filtered by a color filter array (CFA) of our off-the-shelf camera. This image reconstruction is even possible for heavily degraded images, taken with a self-constructed lens consisting of a *single lens element* attached to a standard camera, see Fig. 2.1.

2.2. Related work

There exist many different methods solely for demosaicing, for reviews see [Ram+02; Gun+05; AL08; LGZ08]. However, none of them model and exploit the spatially-varying aberration of the lens to facilitate demosaicing as our method does. Most closely related is a MAP approach that treats deconvolution and demosaicing jointly [ST09], but for blur constant across the image and all color channels.

Chromatic aberrations arise because the refractive index of glass, and thus focal length and image scale, is dependent on the wave length. A common approach to correct for lateral chromatic aberrations is a non-rigid registration of the different color channels [BW92; KL05; MW07]. Such methods correspond to restricting our model to delta-peaked PSFs, and generally ignore other optical aberrations. The method of [CKS09] measures chromatic aberration at edges through color differences and compensates locally, however without using a PSF model of the lens. The approach in [JSK08] also relies on the estimation of sharp step edges and can be used in a non-blind fashion. Even though full PSFs are estimated, they are only used to remove chromatic aberrations, where a rough knowledge of the PSF is sufficient. None of these approaches consider demosaicing.

A method that focuses on correcting coma has been proposed in [Gif08], showing how to reduce coma by locally applying blind deconvolution methods to image patches. This method is designed for grayscale images and thus does neither consider chromatic aberration nor demosaicing.

Algorithmically related to our work is [FEM06], considering sparsity regularization in the luminance channel, and Tikhonov regularization in the two chromaticity channels. However, [FEM06] combines the image information from several images, while our method works with a single image. Also, [FEM06] combines demosaicing with super-resolution, while we combine it with correction for chromatic aberrations.

The image reconstruction problem we are addressing can also be dealt with using the proprietary software "DxO Optics Pro" (DXO), which tries to correct for image aberrations. DXO is considered state-of-the-art among professional photographers and presumably uses the same kind of information as our approach (it contains a custom database of lens/camera combinations). It has been developed over a number of years and is highly optimized. DXO states that it can correct for "lens softness", which their website[1] defines as image blur that varies across the image and between color channels in strength and direction. It is not known to us whether DXO models the blur as space-variant defocus blur of different shapes or with more flexible PSFs as we do; neither do we know whether DXO demosaics and deblurs simultaneously as we do. In the experimental section we show that our results compare favorably with results obtained by DXO.

Using deconvolution to correct for lens aberrations is also discussed in [Kee+11]. This work focuses on removing lens blur across multiple aperture and zoom settings of a given lens. A calibration method similar to [JSK08] is used. While this method can correct aberrations across many different settings of a lens, the blur shape is modelled as a Gaussian. As can be seen in

[1] `https://web.archive.org/web/20110519014422/http://dxo.com/us/photo/dxo_optics_pro/optics_geometry_corrections/lens_softness`

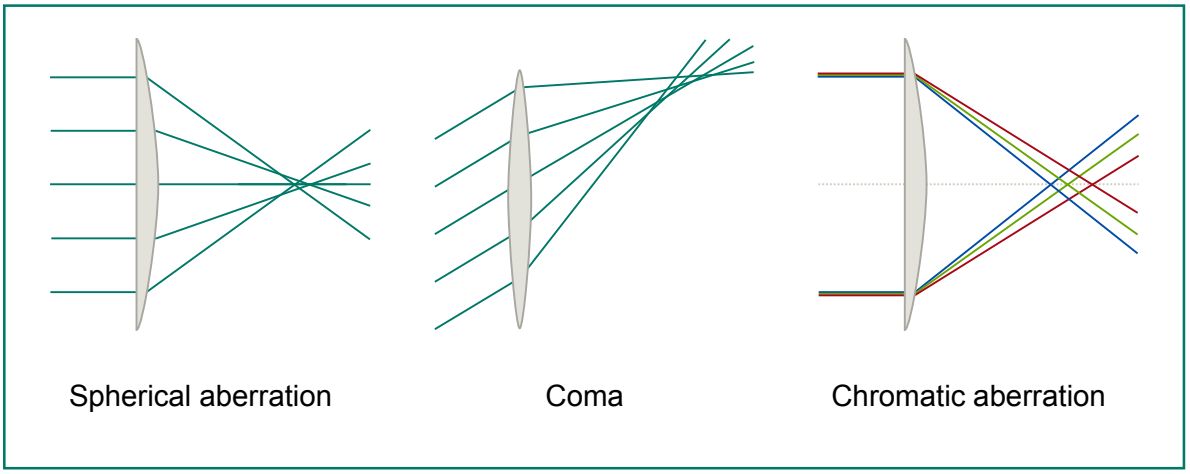


Figure 2.3.: Examples of optical aberrations: light rays are not focused into a single point.

Fig. 2.9 this is not appropriate for strong aberrations considered in this paper. Also, the problem of demosaicing is not treated.

There exist several papers which suggest calibration procedures to measure the lens, e.g. [SA94; Ste02; JSK08]. However, they mainly focus on correcting geometric distortion or do not address monochromatic aberrations.

A closely related method published after the material of this chapter also infers the spatially-varying PSF in a measurement step and removes the blur algorithmically [Hei+13]. It obtains the lens aberration from a calibration pattern and then performs non-blind deconvolution with a cross-channel term designed to minimize color fringing.

2.3. Aberrations as a non-stationary convolution

While the aberrations of an imaging system can be described as a simple matrix operator, we have noted in Chapter 1 that the required matrix-vector multiplication would be computationally expensive. As can be seen in Fig. 2.7 on the left, the aberration can also be represented as PSFs that vary in size, shape, orientation and intensity, depending on the position in the image. Since the PSFs also vary smoothly across the image plane, using EFF (Section 1.1.2) is a valid approximation in this case.

Lenses are affected by a variety of different aberrations. Each of them have in common that light rays are not focused into a single point anymore. In the following, we explain the categories of aberrations that we can include in our model and remove in the reconstruction step:

Monochromatic aberrations. This class of aberrations includes *spherical aberration* (in spherical lenses, the focal length is a function of the distance from the axis, see Fig. 2.3 on the left) as well as a number of off-axis aberrations: *coma* occurs in an oblique light bundle when the intersection of the rays is shifted with respect to its axis (Fig. 2.3 in the middle); *field curvature* occurs when the focal surface is non-planar; *astigmatism* denotes the case when

the sagittal and tangential focal surfaces do not coincide (i.e., the system is not rotationally symmetric for off axis light bundles); *distortion*, which is the only aberration we do not address, is related to a spatially-varying image scale. All these monochromatic aberrations lead to blur that varies across the image. Any such blur can be expressed in the EFF framework by appropriately choosing the local blur kernels $\mathbf{k}^{(1)}, \ldots, \mathbf{k}^{(R)}$ in Eq. (1.9).

Chromatic aberration. The refraction index of most materials including glass is dependent on the wavelength of the transmitted light. Axially, this results in the focus of a lens being a function of the wavelength (*longitudinal chromatic aberration*, Fig. 2.3 on the right); off-axis, we observe *lateral chromatic aberration* caused by the fact that the different focal lengths for different wavelengths directly imply that the image scale slightly varies with wavelength. By modeling the three color channels with separate space-variant PSFs, we are able to describe such chromatic aberration. This means on the red, green and blue color channels $\mathbf{x}_\mathrm{R}$, $\mathbf{x}_\mathrm{G}$, and $\mathbf{x}_\mathrm{B}$ each acts a blur K_R, K_G and K_B, which we can also write as a blur K acting on the full color image $\mathbf{x}$.

Vignetting. Because oblique light bundles do not reach the focal plane in their entirety, the intensity of the image falls off towards the image corners. This can be corrected by photographing a *flat field* frame, i.e., an image of a homogeneous background, and dividing the image by it. While this is straightforward, the EFF framework can also include vignetting into our model by omitting the energy conservation constraint, in that case the filters $\mathbf{k}^{(r)}$ in Eq. (1.9) do not have to sum up to one, i.e., we only require $\sum_j k_j^{(r)} \leq 1$ and $k_j^{(r)} \geq 0$ for all j and r. By allowing dimmer filters we automatically correct for vignetting using our procedure. Note that summation constraint of the windows $w^{(r)}$ is unaffected by relaxing the energy conservation constraint.

2.4. Forward model including mosaicing

Both demosaicing and deblurring are ill-posed linear inverse problems. Demosaicing aims at recovering a full-color image from the spatially undersampled color samples yielded by an image sensor overlaid with a CFA as used in most modern digital cameras.

The image blurred by the blur K is the image that will enter the CFA, just before being mosaiced (cf. Fig. 2.4). The operation of the CFA can be described as a linear map represented by some matrix D, whose result will be the image that hits the photo-sensitive sensor *behind* the CFA. Note that D is a rectangular matrix with three times as many columns as rows. The entries of D will linearly combine pixels in $\mathbf{x}$, e.g., the first pixels of $\mathbf{x}_\mathrm{R}$, $\mathbf{x}_\mathrm{G}$, and $\mathbf{x}_\mathrm{B}$ will be linearly combined to form the first pixel of $\mathbf{y}$ with weights dependent on whether the corresponding spot on the CFA has a red, blue, or green color filter.

The forward model combines the lens aberration and Bayer filtering into a single matrix A and adds noise $\mathbf{n}$, i.e.,

$$\mathbf{y} = DK\mathbf{x} + \mathbf{n} = A\mathbf{x} + \mathbf{n}. \tag{2.1}$$

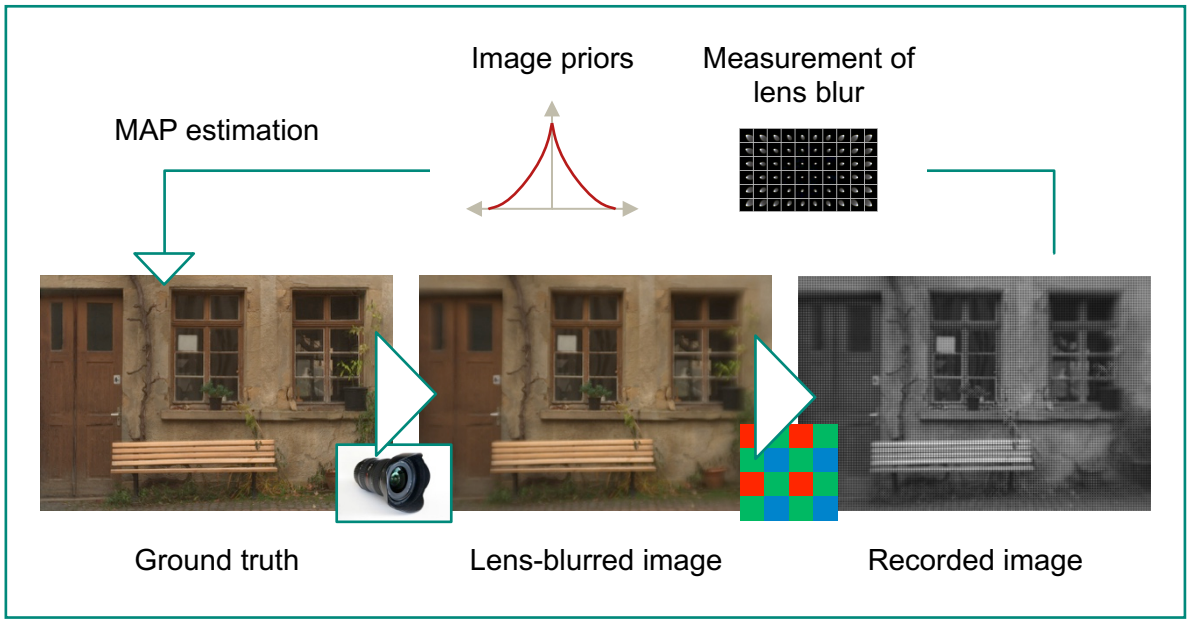


Figure 2.4.: Overview of our approach for non-blind correction of optical aberrations. The ground truth image $\mathbf{x}$ is blurred by the lens blur K and converted to grayscale by the Bayer matrix D. The recorded image $\mathbf{y}$ is the input to our MAP estimation, which employs a spatially-varying measurement of the lens blur and suitable image priors to reconstruct the original image.

In the next two sections we will describe the steps necessary to obtain an estimate of $\mathbf{x}$ from the observation $\mathbf{y}$. First we measure the spatially-varying blur K, then we perform a MAP estimation incorporating prior knowledge about the ground truth image.

2.5. Estimating the non-stationary convolution

In our non-blind approach for correction of optical aberrations, we require a measurement of the local blur kernels $\mathbf{k}^{(r)}$ that constitute K.

Leaving aside diffraction effects (e.g., by ensuring the pixel size to be larger than the Airy disk), a point light source should influence just a single pixel on the imaging sensor of a digital camera. However, this would only happen if a digital camera was a perfect optical system. In practice, the various lens aberrations discussed above will spread out the point light source over a larger region of the imaging sensor. This local pattern characterizes the PSF, so by recording these patterns across the image plane we can set the filters of the non-stationary convolution described above.

To automate the measurements, we mounted a camera on a motor-driven platform with two rotational degrees of freedom. A lens measurement process is conducted in a completely dark room by remotely changing the angles of the camera towards a point light source (a gas lamp emitting light through an aperture of 100 µm in 12 meters distance) such that in subsequent exposures the light point is captured at equidistant locations on the sensor.

In our experiments we use a 18 times 27 grid of supporting points for the EFF framework. The blur kernels were recorded by averaging three dark frame subtracted images of the point light source and thresholding noise. This simple setup gives sufficiently good measurements for the PSF, as can be seen in the final deconvolution results in Section 2.7.2.

2.6. Recovering the corrected, full-color image

In this section we describe how we construct and optimize a MAP objective function to estimate the ground truth image from the image recorded on the sensor of the camera and a measurement of the spatially-varying blur.

When we assume the weights in the Bayer matrix D are fixed and known (we use a trivial Bayer matrix disregarding cross-talk between color channels), the linear transformation A, i.e., the PSF, is parameterized by the set of filters that determine the EFF matrices K_{R}, K_{G}, and K_{B} for the three color channels. These filters depend on the lens and the camera used and can be measured according to the procedure of the previous section.

Assuming the noise in Eq. (2.1) to be Gaussian, we could recover the unknown full-color image $\mathbf{x}$ from a measured raw image $\mathbf{y}$ by solving a least-squares problem, i.e., by minimizing $\|\mathbf{y} - A\mathbf{x}\|^2$ wrt. $\mathbf{x}$. However, the PSF parameterized by the EFF framework is only an approximation to the true PSF and is subject to errors. Using stochastic robust matrix approximation [BV04], we add a regularization term $\|(\mathrm{E}[U^{\mathsf{T}}U])^{1/2}\mathbf{x}\|^2$, where U is an additive random variable that corrupts A, and $\mathrm{E}[\cdot]$ denotes the expectation value. Assuming that each of the n elements of the PSF exhibits a standard deviation of σ with zero mean, we add $n\sigma^2\|\mathbf{x}\|^2$ just for the EFF matrices. Including the Bayer matrix with twice as many green pixels as blue and red pixels, the regularization can be approximated as $n\sigma^2(\|\mathbf{x}_R\|^2/4 + \|\mathbf{x}_G\|^2/2 + \|\mathbf{x}_B\|^2/4)$.

One challenge of processing real photos is that pixels might be saturated, their true values may be clipped due to limited dynamic range. Thus the measured values of clipped pixels are not in agreement with the physical model of the blur. We exclude saturated pixels in the data-fidelity term $\|\mathbf{y} - A\mathbf{x}\|^2$ by summing only over non-saturated pixels.

This term corresponds to the likelihood term (or data fit) of the implicitly underlying probabilistic model. However, because we are trying to estimate three color channels from a single raw image, which means there are three times as many unknowns as observations, our deblurring problem is ill-posed. To regularize it we include prior knowledge about natural images: it has been shown that the image gradients approximately follow a hyper-Laplacian distribution [KF09; SA96], as also illustrated in Fig. 1.5. This can be incorporated into the optimization problem by adding a regularization term of the form $|G\mathbf{x}|^\gamma$ to the objective function. The effect of this regularization is to penalize strong gradients and therefore to smooth the image. We follow Farsiu et al. [FEM06] who transformed the RGB image to a luminance/chrominance color space (here we use YUV) before applying the regularization. This allows us to regularize more strongly in the chrominance channels, and less in luminance. Note that the human eye is more sensitive to differences in luminance than in chrominance, i.e., a visually pleasing result has to be sharp in the luminance channel. The transformation from RGB to YUV is simply a matrix vector multiplication $[\mathbf{x}_{\mathrm{Y}}^{\mathsf{T}}, \mathbf{x}_{\mathrm{U}}^{\mathsf{T}}, \mathbf{x}_{\mathrm{V}}^{\mathsf{T}}]^{\mathsf{T}} = C[\mathbf{x}_{\mathrm{R}}^{\mathsf{T}}, \mathbf{x}_{\mathrm{G}}^{\mathsf{T}}, \mathbf{x}_{\mathrm{B}}^{\mathsf{T}}]^{\mathsf{T}}$ with appropriately chosen matrix

C. With $\mathbf{x}_\mathrm{Y}$, $\mathbf{x}_\mathrm{U}$, and $\mathbf{x}_\mathrm{V}$ we we can write our combined objective function as

$$\|\mathbf{y} - A\mathbf{x}\|^2 + \alpha|G\mathbf{x}_\mathrm{Y}|^\gamma + \beta|G\mathbf{x}_\mathrm{U}|^\gamma + \beta|G\mathbf{x}_\mathrm{V}|^\gamma + n\sigma^2(\|\mathbf{x}_\mathrm{R}\|^2/4 + \|\mathbf{x}_\mathrm{G}\|^2/2 + \|\mathbf{x}_\mathrm{B}\|^2/4). \quad (2.2)$$

We obtained good results by setting $\alpha = 10^{-4}$, $\beta = 10^{-3}$, $\gamma = 0.65$ and $\sigma = 10^{-3}$ in our simulated experiments. On real images, the optimal values for α and β were smaller by a factor of ten.

We minimize the non-convex objective function with respect to $\mathbf{x}$ by adapting Krishnan and Fergus' [KF09] approach to our setup. The minimization problem

$$\min_{\mathbf{x},\mathbf{w}_\mathrm{Y},\mathbf{w}_\mathrm{U},\mathbf{w}_\mathrm{V}} \|\mathbf{y} - A\mathbf{x}\|^2 + n\sigma^2(\|\mathbf{x}_\mathrm{R}\|^2/4 + \|\mathbf{x}_\mathrm{G}\|^2/2 + \|\mathbf{x}_\mathrm{B}\|^2/4) +$$
$$\frac{c}{2}\left(\|G\mathbf{x}_\mathrm{Y} - \mathbf{w}_\mathrm{Y}\|^2 + \|G\mathbf{x}_\mathrm{U} - \mathbf{w}_\mathrm{U}\|^2 + \|G\mathbf{x}_\mathrm{V} - \mathbf{w}_\mathrm{V}\|^2\right) + \alpha|\mathbf{w}_\mathrm{Y}|^\gamma + \beta|\mathbf{w}_\mathrm{U}|^\gamma + \beta|\mathbf{w}_\mathrm{V}|^\gamma, \quad (2.3)$$

which introduces the auxiliary variables $\mathbf{w}_\mathrm{Y}$, $\mathbf{w}_\mathrm{U}$ and $\mathbf{w}_\mathrm{V}$, is equivalent to minimizing Eq. (2.2) when $c \to \infty$. This new objective is convex in $\mathbf{x}$ and non-convex in $\mathbf{w}$, for a fixed c it can be solved by iterating between the $\mathbf{x}$- and $\mathbf{w}$-subproblem:

- $\mathbf{x}$-*subproblem:* The convex phase minimizes with respect to $\mathbf{x}$ by employing the iterative quasi-Newton method L-BFGS-B[2], where it is necessary to provide the gradient of the objective. The non-trivial part of this gradient is the expression $A^\mathsf{T} = D^\mathsf{T}K^\mathsf{T}$, which requires K^T from Eq. (1.10).

- $\mathbf{w}$-*subproblem:* The non-convex phase solves

$$\min_{\mathbf{w}_\mathrm{Y}} \frac{c}{2}\|G\mathbf{x}_\mathrm{Y} - \mathbf{w}_\mathrm{Y}\|^2 + \alpha|\mathbf{w}_\mathrm{Y}|^\gamma, \quad (2.4)$$

  which can be done component-wise for all entries of $\mathbf{w}_\mathrm{Y}$ (and analogously for $\mathbf{w}_\mathrm{U}$ and $\mathbf{w}_\mathrm{V}$). Furthermore, since this process has to be repeated many times for every iteration (number of pixels times color channels), precomputing a lookup table of solutions gives a large speed-up [KF09].

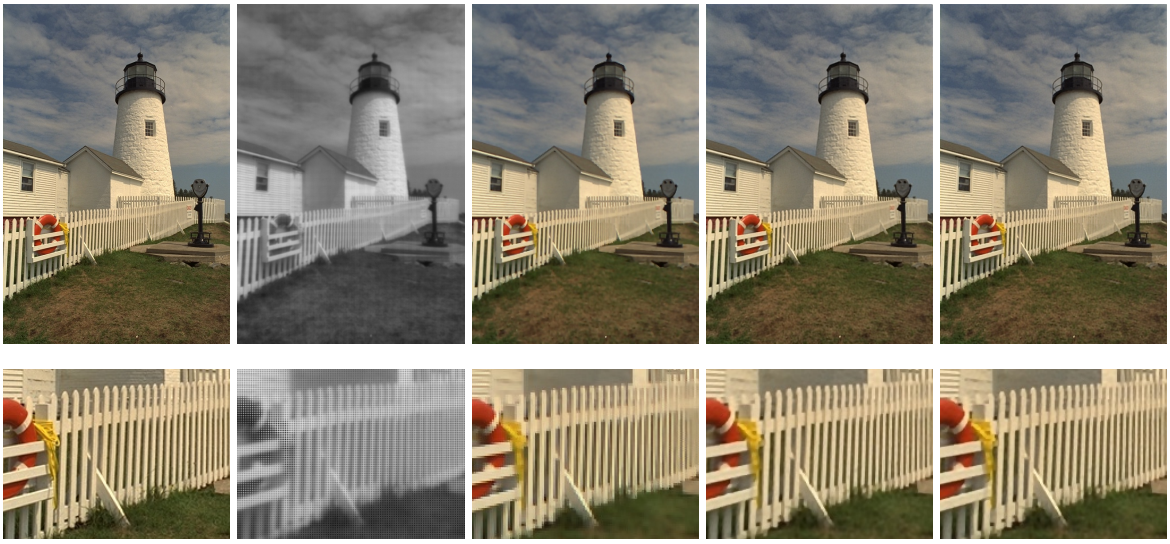
For a fixed c we alternate two times between convex and non-convex phase. Afterwards, c is increased by a factor of $2\sqrt{2}$. Our schedule for c starts with $c = 1$ and increases its value five times.

2.7. Results

2.7.1. Simulated images

To test our method under controlled conditions, we artificially blurred test images usually used for evaluating demosaicing algorithms from the Kodak PhotoCD. To simulate the lens aberrations, we created a 4×6 filter array containing measured blur kernels of a Canon 50mm

[2] `https://github.com/pcmoritz/traffic-project/tree/master/lbfgsb-matlab`



Ground truth | Blurred and mosaiced input image | (a) Deconv. then demosaic. PSNR 24.77 dB | (b) Demosaic. then deconv. PSNR 27.87 dB | (c) Joint approach PSNR **28.46** dB

Figure 2.5.: Comparison of our joint approach vs. sequential demosaicing and deconvolution procedures. The PSF used for the simulations are shown in Fig. 2.6. Gaussian noise with a PSNR of 50 dB has been added.

f/1.4 lens at maximum aperture on a Canon 5D Mk II. This filter array contains the parameters of a non-stationary convolution that represent our estimated model of the artificial lens aberrations. To account for the fact that the true PSF is not exactly known, we modify these filters with a low pass filter before convolving the ground truth images. In the image reconstruction process, the non-modified blur filters were used. We then added white noise with a peak signal-to-noise ratio (PSNR) of 50 dB and mosaiced the result with a Bayer filter array.

With the simulated experiments we want to investigate whether (a) we should apply the aberration correction separately on each color channel and subsequently demosaic with a state-of-the-art demosaicing algorithm [Pal+07], whether (b) our aberration correction should be better applied to images that have been already demosaiced by a standard demosaicing procedure, or whether (c) it is best to apply the forward model that includes the mosaicing (as described in Section 2.4), i.e., to jointly correct the aberrations and the demosaicing.

Table 2.1 compares the PSNRs of the reconstructed images for the approaches (a), (b), and (c) on the image data set. For all 24 images the joint approach (c) leads to the best results, approach (b) being a close runner-up. This finding is also visually confirmed in Fig. 2.5 where approach (c) leads to the best reconstruction. Note that to suppress influence of the border region, a 15 pixel border on all edges has been excluded in the calculation of the PSNR.

We believe that our approach is able to compete with state-of-the-art demosaicing algorithms because separating demosaicing and deblurring has the disadvantage that it does not require the result to be consistent with the image formation model. Because of the blur, we gain knowledge about possible values for missing color information. For example, if we measure no light at

Image	(a) Deconv. then demosaic.	(b) Demosaic. then deconv.	(c) Joint approach
1	23.09	25.92	**26.35**
2	30.11	31.92	**32.23**
3	30.67	33.47	**33.68**
4	29.12	32.23	**32.49**
5	22.58	26.08	**26.62**
6	24.84	27.09	**27.47**
7	27.87	33.07	**33.47**
8	20.32	23.77	**24.28**
9	28.02	32.11	**32.51**
10	28.54	31.53	**31.96**
11	25.92	28.77	**29.11**
12	29.51	32.67	**33.04**
13	21.32	23.32	**23.81**
14	25.34	28.32	**28.79**
15	28.90	32.14	**32.52**
16	28.41	30.40	**30.68**
17	28.22	31.33	**31.68**
18	25.06	27.75	**28.20**
19	24.77	27.87	**28.46**
20	27.66	31.40	**31.78**
21	25.27	28.17	**28.63**
22	26.86	29.61	**29.95**
23	30.00	34.08	**34.59**
24	23.74	26.06	**26.34**
Average	26.51	29.54	**29.94**

Table 2.1.: Comparison of PSNR in dB for Kodak image data set. Consistently, the joint approach outperforms the sequential demosaicing and deconvolution procedures (higher number means better reconstruction).

a certain pixel, we can infer that in the deblurred image the surrounding region given by the size of the PSF also has to be dark. Furthermore, typical demosaicing algorithms do not take chromatic aberration into account, which leads to a spatial separation of edge information across different color channels.

In fact, lens aberrations can even assist in debayering. Imagine a perfect lens mapping each point to a point on the sensor, imaging a single red dot. If this red dot happens to fall on a green or blue filter of the Bayer pattern, it is lost. If, however, the red dot leads to a slightly larger PSF, then in principle there is enough information to recover the dot even if its center did not fall on a pixel with red filter. This observation is the reason why DSLR manufacturers generally put a spatial smoothing (low pass) filter in front of the Bayer pattern.

2.7.2. Real images

Using the automated procedure from Section 2.5, we approximate the PSFs of three different lenses: (i) Canon 50mm f/1.4, (ii) Canon 24mm f/1.4 L, and (iii) a self-built lens consisting of a single glass element, see Fig. 2.7. For the Canon lenses, we took several pictures with a

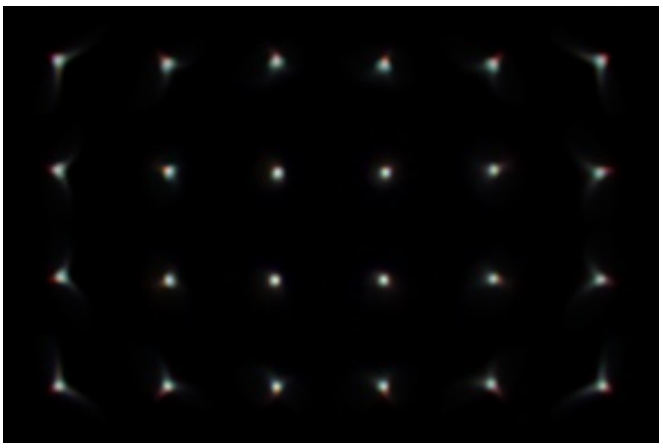


Figure 2.6.: Point spread function used for simulations on the Kodak image data set.

Canon 5D Mk II digital camera, for the self-built lens we used a Canon 5D Mk I. We applied our image reconstruction procedure described in Section 2.6 to these images and next describe the results.

In our PSF measurement we only obtain mosaiced versions. However, as can be seen in Fig. 2.9, the blur is sufficiently well behaved such that bilinear interpolation gives a good approximation to the true PSF.

Canon 50mm f/1.4. First, we use a Canon 50mm f/1.4 prime lens on a Canon 5D Mark II at maximum aperture. The comparison between original photo and the image corrected for lens errors is in Fig. 2.7. In Fig. 2.8, it is compared with the result of DXO version 6 (see Section 2.2), a software that is also able to correct for lens aberrations. Similar to our approach, it relies on previously recorded information about the error of a certain lens/camera combination. In the comparison, all image improvements except the correction for "lens unsharpness", chromatic aberration and vignetting were deactivated. While in the DXO result the edges are sharpened, the objects have a halo, e.g., around the wooden bars, which is not present in the original scene. This means the blur introduced by the lens is not completely removed.

Canon 24mm f/1.4. Furthermore, we correct the errors of a Canon EF 24mm f/1.4 at maximum aperture, which exhibits considerably visible errors in the border regions of the image at fully open aperture. The original and the corrected image can be seen in Fig. 2.7. In the recorded image strong chromatic aberration is visible as green and red lines near edges, which are reduced in the deconvolved result. This lens is not available in the DXO database for the Canon 5D Mk II, so DXO cannot be applied.

Self-built lens with a single lens element. The two lenses used above are high-end lenses with a complicated system of compound lenses that are built to minimize optical errors. Trying to make our algorithm fail, we constructed a simple photographic lens from a single convex-concave lens with focal length 120mm. Amazingly, the image can be well reconstructed as can be seen in Figures 2.2 and 2.7. In Figure 2.7, nearly no detail is recognizable in the grain

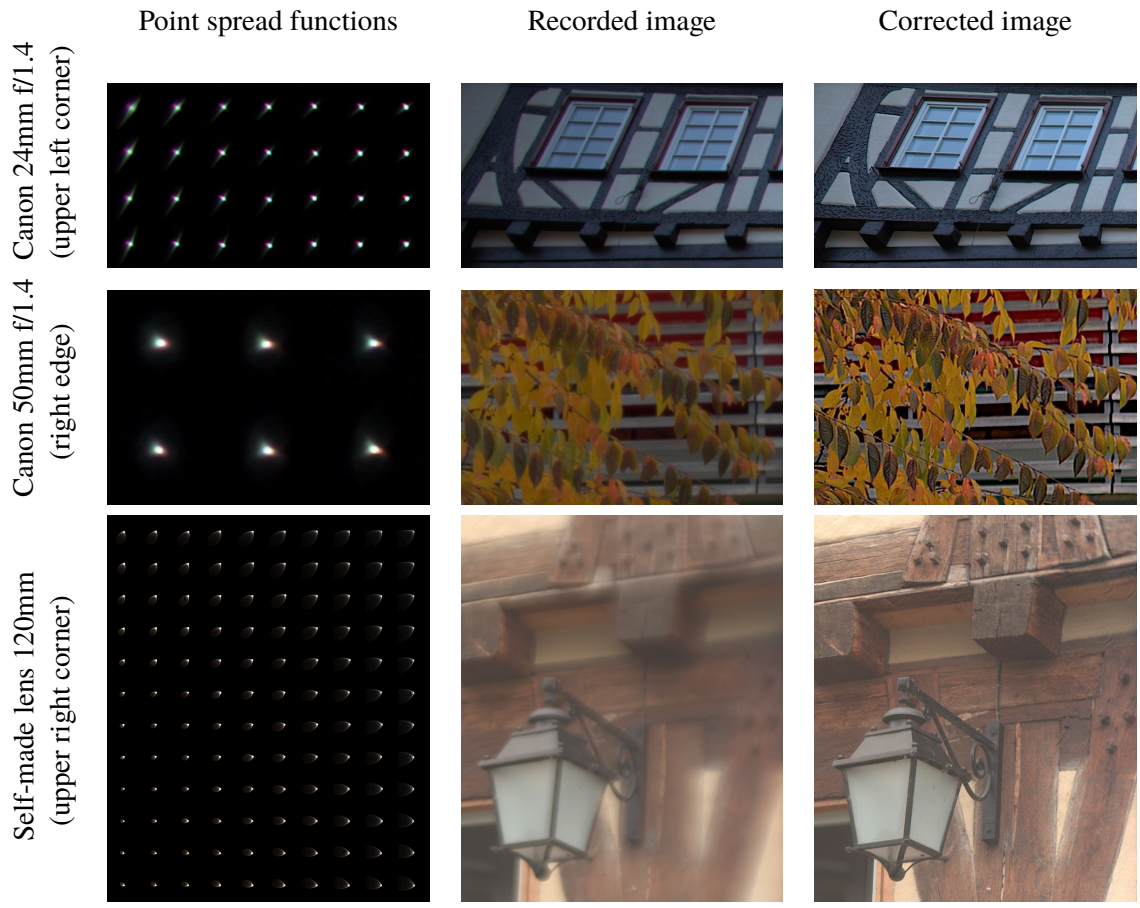


Figure 2.7.: Comparison between original and corrected image and the respective PSFs.

Figure 2.8.: Comparison with DXO for images taken with a Canon EF 50mm f/1.4 lens.

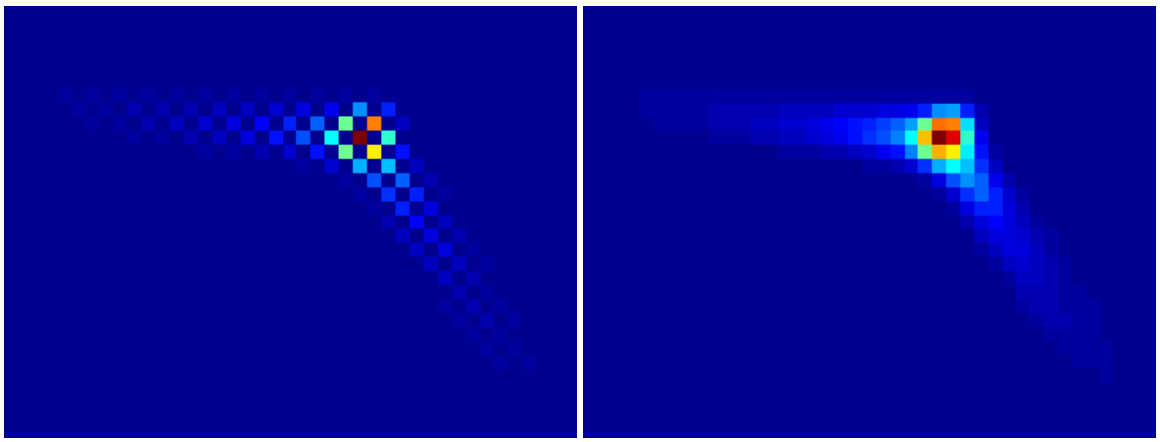


Figure 2.9.: Interpolation of a mosaiced PSF at the example of a green PSF from the Canon 50mm f/1.4 lens.

Recorded image | Corrected with [Hei+13] | Our result

Figure 2.10.: Comparison with the newer method from [Hei+13]. Shown is a patch from the upper right corner of Fig. 2.2.

of the wood in the original image. Also, the pegs on the right and upper edge of the image are hardly visible. The corrected image does not suffer from these problems.

A method published after the material of this chapter slightly improves on our results. As can be seen from Fig. 2.10, it exhibits less color artifacts at sharp edges, for example at the right border of the window.

Running time. For the 21.1 megapixel photos taken with the Canon lenses, the full-color non-convex optimization problem has more than 60 M unknowns. It needs about 5 hours running time on a quad-core computer. For the self-built lens, we used a camera which produces 12.8 megapixel images and a blur size of 200×200. In the EFF framework with 27×18 supporting points, the processing takes about 7 hours using a MATLAB implementation of the algorithm.

This running time is impractical. However, we show how the EFF framework can be used to do *Direct Deconvolution* in Fourier space with a slightly modified version of our objective function. Since the demosaicing operator is not diagonal in Fourier space, we work on each

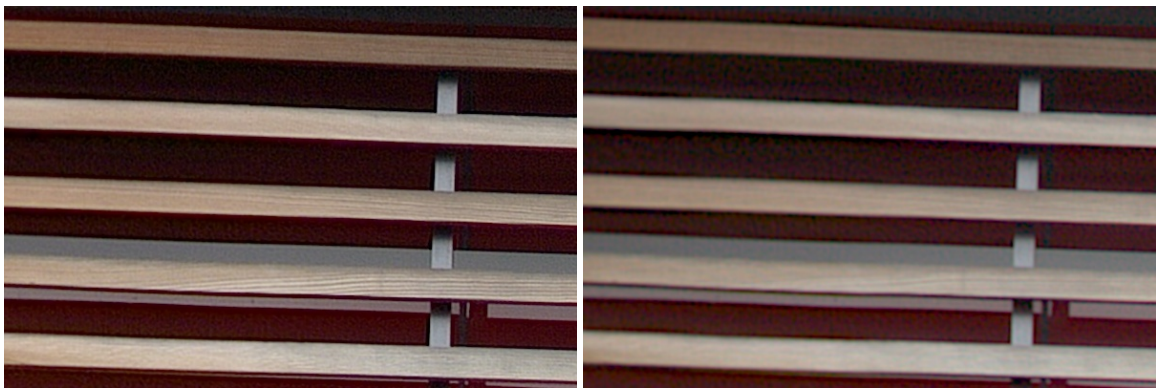


Figure 2.11.: Comparison of deconvolution with optimization (left) and direct method (right).

already demosaiced color channel separately and solve the problem

$$\|\mathbf{y} - K\mathbf{x}\|^2 + \alpha|G\mathbf{x}|^\gamma + n\sigma^2\|\mathbf{x}\|^2. \tag{2.5}$$

This can be done with the approach of [KF09], which describes a fast solution for the non-convex problem of reconstructing an image with invariant blur assuming a hyper-Laplacian gradient distribution. It iterates between a convex and non-convex phase, where the convex sub-problem is similar to Eq. (1.16), enabling a closed-form solution Eq. (1.17) in Fourier space.

This is also the approach we adapted in Section 2.6 to solve Eq. (2.2). However, we relied on an iterative quasi-Newton solver to minimize the convex phase, since the matrix D does not diagonalize in Fourier space, as would be necessary for a one-step solution.

Without mosaicing, this is possible. Using the expression from Eq. (1.10), K can be approximately inverted as

$$\mathbf{x} \approx \mathbf{v} \odot \sum_r \operatorname{Diag}(\mathbf{w}^{(r)})^{1/2} C_r^\mathsf{T} F^\mathsf{H} \frac{\overline{FC_k^\mathsf{T}\mathbf{k}^{(r)}} \odot (FC_r \operatorname{Diag}(\mathbf{w}^{(r)})^{1/2}\mathbf{y})}{|FC_k^\mathsf{T}\mathbf{k}^{(r)}|^2 + \alpha \sum_i |FC_{g_i}^\mathsf{T}\mathbf{g}_i|^2 + n\sigma^2}, \tag{2.6}$$

similar to Eq. (8) in [Hir+11]. The filters $\mathbf{g}_i$ have a regularizing effect and assume that the gradient operator G can be expressed convolutionally such that $G\mathbf{x} \equiv \sum_i \mathbf{g}_i * \mathbf{x}$. The transposes of the cropping matrices C_{g_i} appropriately zero-pad $\mathbf{g}_i$ to the correct size. The weighting $\mathbf{v}$ is obtained by applying the inversion to a constant image and is necessary to remove artifacts stemming from inverting the windows. In Fig. 2.11 the results obtained by optimizing the more sophisticated objective function (Eq. (2.2)) are compared to the direct method. While losing a small amount of image quality, the running time is only 2 minutes for a 21.1 megapixel image.

2.8. Conclusion

In this chapter, we have proposed a method to correct the aberrations in optical imaging systems. A spatially-varying PSF is obtained in a calibration step, encoding the errors of the imaging

system. These are then removed by non-stationary deconvolution. Furthermore, by requiring the corrected image to be consistent with the image formation model, we are able to recover missing image information. We have shown this using the example of reconstructing color data lost in a mosaicing process.

Using controlled experiments on images artificially convolved with a non-stationary PSF, we have seen that our linear image formation model leads to better results than separately deblurring and demosaicing Bayer-filtered photos. More importantly, we were able to show that in a real imaging setup, we can correct the optical aberrations rather well both for commercial camera lenses and optically poor single element lenses. The results compare favorably to DXO, a commercially available software package considered state-of-the-art in lens error correction among professional photographers.

2.8.1. Limitations

For the image taken with a one-element lens, we have seen that although a drastic improvement can be achieved, a perfect reconstruction was not possible. Moreover, our measurement procedure suffers from the fact that the PSFs obtained are already subject to mosaicing, therefore the PSFs used in the joint demosaicing/deblurring are only an approximation. A better PSF could, e.g., be obtained with a monochromatic camera and color filters. The general quality of the PSF could for example be improved with wavefront measurement.

Also, the lens aberrations depend to a certain extent on the settings of the lens (aperture, focus, zoom), which cannot be trivially modeled. In the case of lens blurs that can be approximated as a Gaussian with spatially-varying parameters, it has been demonstrated how the change of the lens aberrations can be modeled [Kee+11]. In the case of non-Gaussian blurs, as treated in this work, this problem has still to be solved. This would make our method feasible for lenses with a large number of possible settings, e.g., zoom lenses.

2.8.2. Future work

Imperfections in image deblurring due to the varying distance could be handled by building a database of PSFs at different distances. Combining this with a semi-blind deconvolution approach, a correct deconvolution might yield partial information about the depth-map of the photograph.

Furthermore, the current sampling of the PSF on a regular grid does not exploit the fact that for most lenses the PSF is nearly constant (and often rather small) over a large region in the center of the image. There is thus potential to get away with fewer supporting points in the EFF.

A further common error of imaging systems, distortions, can in principle also be encoded in a spatially-varying PSF. However, in the case of strong distortions this would require PSFs as large as 500×500 pixels, say, and a large computational load. It would, however, be an elegant method for correcting all optical aberrations in one framework.

We believe that the results of this can have significant implications for the design of lenses, which today are probably the most expensive components of high-end camera systems.

CHAPTER 3

Blind Correction of Optical Aberrations

Camera lenses are a critical component of optical imaging systems, and lens imperfections compromise image quality. While, traditionally, sophisticated lens design and quality control aim at limiting optical aberrations, recent works [JSK08; Kee+11] promote the correction of optical flaws by computational means. These approaches rely on elaborate measurement procedures to characterize an optical system, and perform image correction by non-blind deconvolution.

In this chapter, we present a method that utilizes physically plausible assumptions to estimate non-stationary lens aberrations — as opposed to the previous chapter — blindly*, and thus can correct images without knowledge of specifics of camera and lens. The blur estimation features a novel preconditioning step that enables fast deconvolution. We obtain results that are competitive with state-of-the-art non-blind approaches.*

The material of this chapter is based on the following publication:

[Sch+12] *C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. "Blind Correction of Optical Aberrations". In:* Computer Vision – ECCV 2012*. Lecture Notes in Computer Science. Springer, 2012, pp. 187–200.* DOI*: 10.1007/978-3-642-33712-3_14*

3.1. Introduction

In the previous chapter we have seen that lens aberrations can be modeled as a non-stationary convolution. With this knowledge, we were able to perform deconvolution and correct the shortcomings of the optical system *in software*. This enabled us to improve the imaging quality of both low-quality and high-end lenses. A major disadvantage, however, was that we had to measure each camera/lens combination in a time-consuming calibration procedure. This is especially an issue for lenses with a multitude of settings: zoom, aperture, focus. While some of these parameters have only a minor influence on the aberrations (e.g., focus), others are quite

significant (e.g., aperture). Sometimes, the camera is not available anymore, for example for old photos recorded over a century ago, which may still be of sentimental or historical value.

To overcome these issues, we propose to use *non-stationary blind deconvolution*. Deconvolution is a hard inverse problem, which implies that in practice, even non-blind uniform deconvolution requires assumptions to work robustly, as we have seen in the previous chapter. Blind deconvolution is harder still, since we additionally have to estimate the blur kernel, and non-uniform deconvolution means that we have to estimate the blur kernels as a function of image position. The art of making this work consists of finding the right assumptions, sufficiently constraining the solution space while being at least approximately true in practice, and designing an efficient method to solve the inverse problem under these assumptions. Our approach is based on a *forward model* for the image formation process that incorporates two assumptions:

1. The image contains certain elements typical of natural images, in particular, there are sharp edges.

2. Even though the blur due to optical aberrations is non-uniform (spatially-varying across the image), there are circular symmetries that we can exploit.

Inverting a forward model has the benefit that if the assumptions are correct, it will lead to a plausible *explanation* of the image, making it more credible than an image obtained by sharpening the blurry image using, say, an algorithm that filters the image to increase high frequencies.

Furthermore, we emphasize that the approach of this chapter is *blind*, i.e., it requires as an input only the blurry image, and not a point spread function that we may have obtained using other means such as a calibration step. This is a substantial advantage, because of the mentioned dependence of the blur not only on the particular photographic lens but also on settings such as focus, aperture and zoom. Also, this approach scales easily to the countless, already existing lenses and cameras, which is especially important since the aberration on a pixel level also depends on the exact combination between camera and lens.

Main contributions: We design a class of PSF families containing realistic optical aberrations, via a set of suitable symmetry properties. Furthermore, we represent the PSF basis using an orthonormal basis to improve conditioning, and allow for direct PSF estimation. With this, we show how to avoid calibration for specific camera/lens combinations by proposing a *blind* approach for inferring the PSFs, widening the applicability to any photographs (e.g., with missing lens information such as historical images). We also extend blur estimation to multiple color channels to remove chromatic aberrations as well, and finally we present experimental results showing that our approach is competitive with non-blind approaches.

3.2. Related work

The existing deconvolution methods to reduce blur due to optical aberrations are *non-blind* methods, i.e., they require a time-consuming calibration step to measure the PSF of the given

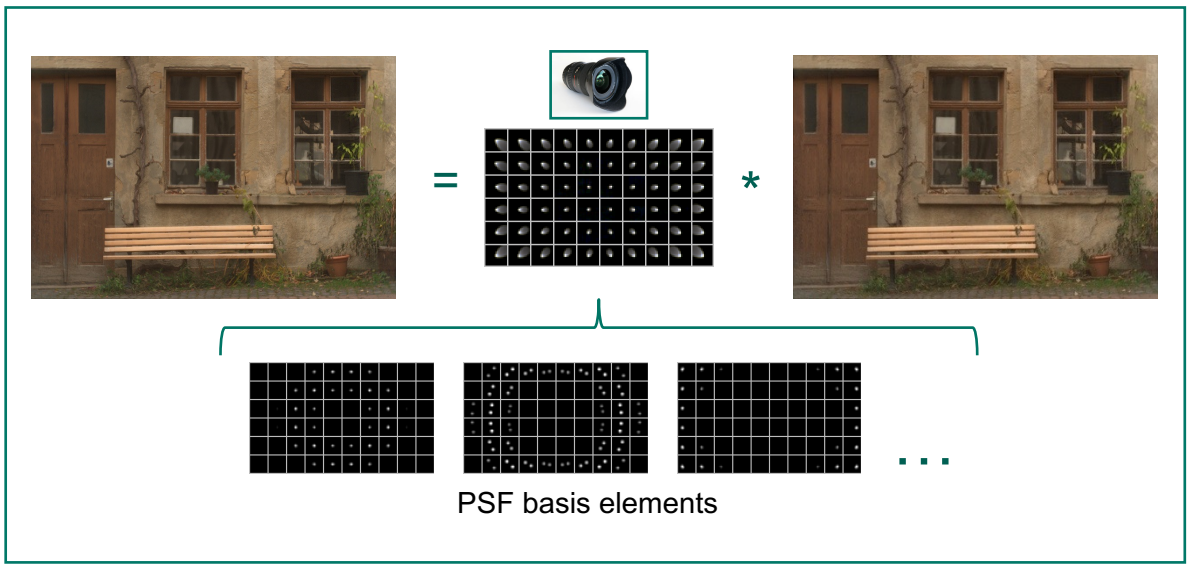


Figure 3.1.: Optical aberration as a forward model.

camera-lens combination, and, in principle, they require this for all parameter settings. The previous chapter introduced an approach in this spirit, and we discussed comparable methods in Section 2.2.

The background for techniques to deconvolve optical aberration *blindly* is recent progress in the area of removing camera shake. For a detailed discussion, we refer to Sections 1.3 and 5.2. As opposed to Chapter 5, instead of creating a new blind deconvolution method from scratch, we here start with an existing blind method and adapt it to our specific problem domain, mainly that our spatially-varying blur has specific properties, and that we have to deal with color-dependent PSFs. Methodically, our general procedure is related to the stationary method Cho and Lee and a variant for non-uniform camera shake [Hir+11]. These were chosen for their fast runtime and their modifiability towards handling full-color blurs. However, this is the first method to correct for lens aberrations *blindly*.

3.3. An efficient filter flow basis for optical aberrations

Since optical aberrations lead to image degradations that can be locally modeled as convolutions, the EFF framework from Section 1.1.2 is a valid model. However, not all blurs expressible in the EFF framework do correspond to blurs that could be caused by optical aberrations. We thus define a PSF basis that constrains the EFF framework to physically plausible PSFs only. This step is important to reduce the dimensionality of the problem significantly and make a *blind* solution feasible.

To define the basis we introduce a few notions. As usual in the EFF framework, the image $\mathbf{y}$ is split into overlapping patches. For each patch, the symbol $l^{(r)}$ denotes the line from the patch center to the image center, and $d^{(r)}$ the length of line $l^{(r)}$, i.e., the distance between patch center and image center. We assume that local blur kernels $\mathbf{k}^{(r)}$ originating from optical aberrations

have the following properties:

1. **Local reflection symmetry:** a local blur kernel $\mathbf{k}^{(r)}$ is reflection symmetric with respect to the line $l^{(r)}$.

2. **Global rotation symmetry:** two local blur kernels $\mathbf{k}^{(r)}$ and $\mathbf{k}^{(s)}$ at the same distance to the image center (i.e., $d^{(r)} = d^{(s)}$) are related to each other by a rotation around the image center.

3. **Radial behavior:** along a line through the image center, the local blur kernels change smoothly. Furthermore, the maximum size of a blur kernel is assumed to scale linearly with its distance to the image center.

Note that these properties are compromises that lead to good approximations of real-world lens aberrations. Due to issues such as decentering, real world lenses may not be absolutely rotationally symmetric. Our exemplar of the Canon 24mm f/1.4 (see Fig. 3.10) exhibits PSFs that deviate slightly from the local reflection symmetry. The assumption, however, still turns out to be useful in that case.

For two dimensional blur kernels, we represent the basis by M basis elements $\mathbf{b}_m$ each consisting of R local blur kernels $\mathbf{b}_m^{(1)}, \ldots, \mathbf{b}_m^{(R)}$. Then the actual blur kernel $\mathbf{k}^{(r)}$ can be represented as linear combinations of basis elements,

$$\mathbf{k}^{(r)} = \sum_{m=1}^{M} \tau_m \mathbf{b}_m^{(r)} \tag{3.1}$$

with weights τ_m. To define the basis elements we group the patches into overlapping groups, such that each group contains all patches inside a certain ring around the image center, i.e., the center distance $d^{(r)}$ determines whether a patch belongs to a particular group. Basis elements for three example groups are shown Fig. 3.2. All patches inside a group will be assigned similar kernels. The width and the overlap of the rings determine the amount of smoothness between groups (see property (c) above).

For a single group we define a series of basis elements as follows. For each patch in the group we generate matching blur kernels by placing a single delta peak inside the blur kernel and then mirror the kernel with respect to the line $l^{(r)}$ (see Fig. 3.3). For patches not in the current group (i.e., in the current ring), the corresponding local blur kernels are zero. This generation process creates basis elements that fulfill the symmetry properties listed above. To increase smoothness of the basis and avoid effects due to pixelization, we place little Gaussian blurs (standard deviation 0.5 pixels) instead of delta peaks.

3.4. An orthonormal efficient filter flow basis

The basis elements constrain possible blur kernels to fulfill the above symmetry and smoothness properties. However, the basis is overcomplete and direct projection on the basis is not possible.

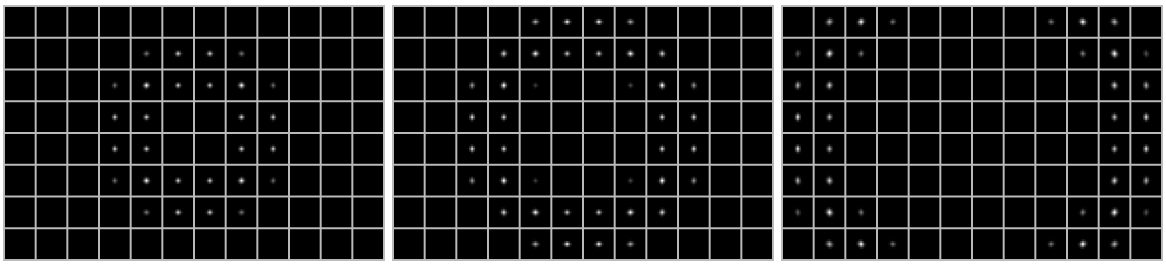
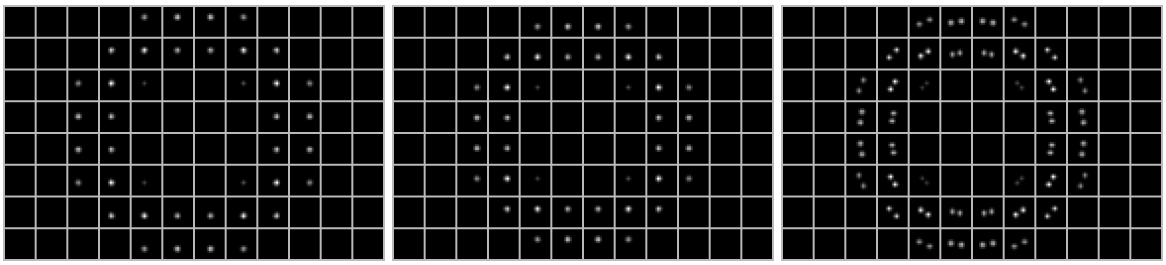


Figure 3.2.: Three example groups of patches, each forming a ring.



(a) outside parallel to $l^{(r)}$ (a) inside parallel to $l^{(r)}$ (c) perpendicular to $l^{(r)}$

Figure 3.3.: Shifts to generate basis elements for the middle group of Fig. 3.2.

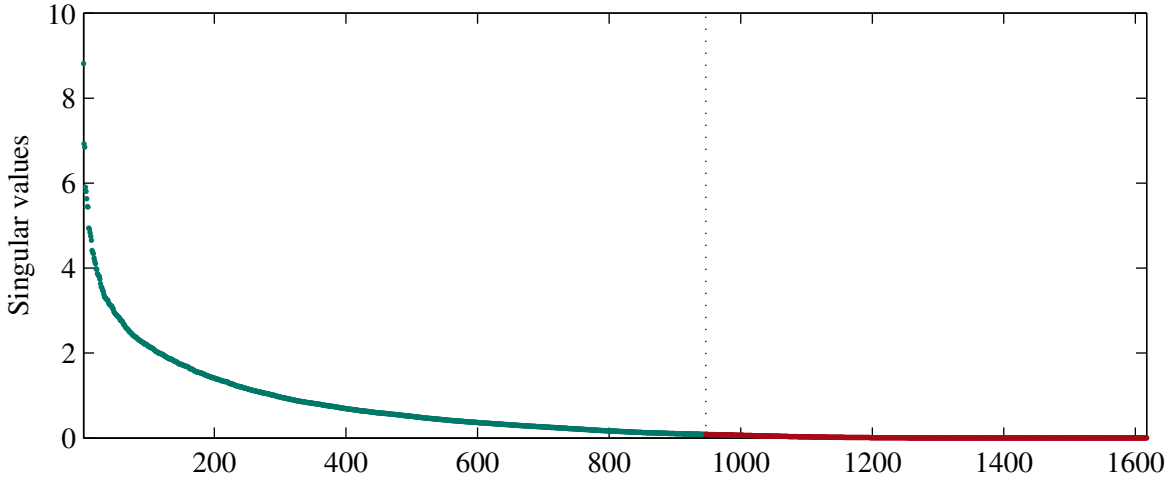
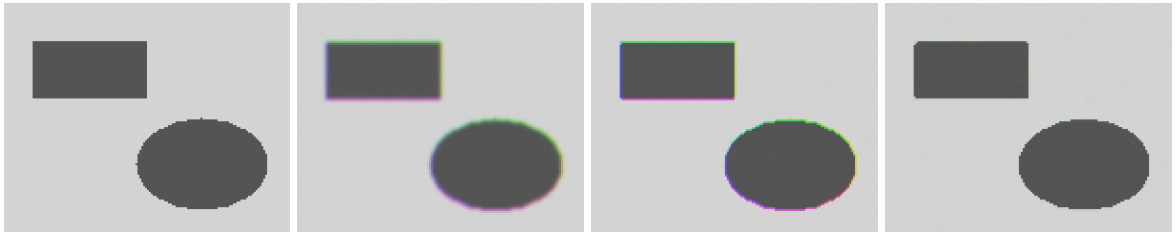


Figure 3.4.: SVD spectrum of a typical basis matrix B with cut-off.



Original | Blurry | Shock filter [OR90] | Chromatic shock filter

Figure 3.5.: Chromatic shock filter removes color fringing (adapted from [OR90]).

Therefore, we approximate it with an orthonormal one. To explain this step with matrices, we stack each local blur kernel $\mathbf{b}_m^{(r)}$ for all patches r:

$$\mathbf{b}_m = \left[\mathbf{b}_m^{(1)\mathsf{T}} \dots \mathbf{b}_m^{(R)\mathsf{T}} \right]^\mathsf{T}. \tag{3.2}$$

Let B be the matrix containing the basis vectors $b_1, \dots, b_M$ as columns. Then we can calculate the singular value decomposition (SVD) of B,

$$B = USV^\mathsf{T}. \tag{3.3}$$

with S being a diagonal matrix containing the singular values of B. Figure 3.4 shows the SVD spectrum and the chosen cut-off of some typical basis matrix B, with approximately half of the eigenvalues being below numerical precision.

We define an orthonormal EFF basis Ξ that is the matrix that consists of the column vectors of U that correspond to large singular values, i.e., that contains the relevant left singular vectors of B. Properly chopping the column vectors of Ξ into shorter vectors, one per patch, we obtain orthonormal basis vectors $\boldsymbol{\xi}_m^{(r)}$ for the EFF framework that are tailored to optical aberrations. This representation can be plugged into the EFF forward model in Eq. (1.9),

$$\mathbf{y} = \boldsymbol{\mu} \diamond \mathbf{x} := \sum_{r=1}^{R} \left(\sum_{m=1}^{M} \mu_m \boldsymbol{\xi}_m^{(r)} \right) * \left(\mathbf{w}^{(r)} \odot \mathbf{x} \right). \tag{3.4}$$

Note that the resulting forward model is linear in the parameters $\boldsymbol{\mu}$.

3.5. Blind deconvolution with chromatic shock filtering

Having defined a PSF basis, we perform blind deconvolution by extending [CL09] to our non-uniform blur model Eq. (3.4) (similar to [HHS10; Hir+11]). However, instead of considering only a grayscale image during PSF estimation, we are processing the full-color image. This allows us to better address chromatic aberrations by an improved *shock filtering* procedure that

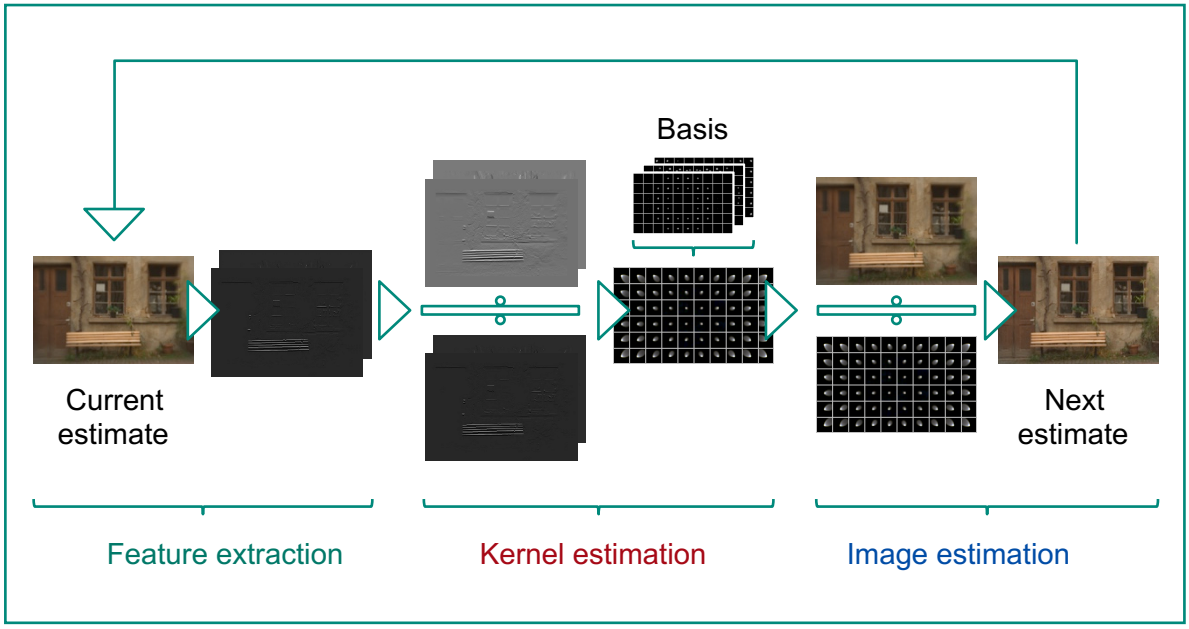


Figure 3.6.: Overview of our approach for blind correction of optical aberrations. Similar to Fig. 1.6, our method iterates between *feature extraction*, *kernel estimation* and *image estimation*, but incorporates a spatially-varying lens blur basis, and operates on full-color images.

is tailored to color images. The color channels $\mathbf{x}_R$, $\mathbf{x}_G$ and $\mathbf{x}_B$ are shock filtered separately but share the same sign expression depending only on the grayscale image $\mathbf{z}$:

$$\begin{aligned} \mathbf{x}_R^{t+1} &= \mathbf{x}_R^t - \Delta t \cdot \operatorname{sign}(z_{\eta\eta}^t)|\nabla \mathbf{x}_R^t| \\ \mathbf{x}_G^{t+1} &= \mathbf{x}_G^t - \Delta t \cdot \operatorname{sign}(z_{\eta\eta}^t)|\nabla \mathbf{x}_G^t| \qquad \text{with } \mathbf{z}^t = (\mathbf{x}_R^t + \mathbf{x}_G^t + \mathbf{x}_B^t)/3 \\ \mathbf{x}_B^{t+1} &= \mathbf{x}_B^t - \Delta t \cdot \operatorname{sign}(z_{\eta\eta}^t)|\nabla \mathbf{x}_B^t| \end{aligned} \tag{3.5}$$

where $z_{\eta\eta}$ denotes the second derivative in the direction of the gradient. We call this extension *chromatic shock filtering* since it takes all three color channels simultaneously into account. Figure 3.5 shows the reduction of color fringing on the example of Osher and Rudin [OR90] adapted to three color channels.

Combining the forward model $\mathbf{y} = \boldsymbol{\mu} \diamond \mathbf{x}$ defined above and the chromatic shock filtering, the PSF parameters $\boldsymbol{\mu}$ and the image $\mathbf{x}$ (initialized with $\mathbf{y}$) are estimated by iterating over three steps, as illustrated in Fig. 3.6:

1. **Feature extraction:** the current estimate $\mathbf{x}$ is first denoised with a bilateral filter, then edges are emphasized with chromatic shock filtering and by zeroing flat gradient regions in the image (see [CL09] for further details). The gradient selection is modified such that for every radius ring the strongest gradients are selected.

2. **Kernel estimation:** if we work with the overcomplete basis B, we would like to find

coefficients $\boldsymbol{\tau}$ that minimize the regularized fit of the gradient images $G\mathbf{y}$ and $G\mathbf{x}$,

$$\|G\mathbf{y} - \sum_{r=1}^{R}(B^{(r)}\boldsymbol{\tau}) * (\mathbf{w}^{(r)} \odot G\mathbf{x})\|^2 + \alpha \sum_{r=1}^{R}\|GB^{(r)}\boldsymbol{\tau}\|^2 + \beta \sum_{r=1}^{R}\|B^{(r)}\boldsymbol{\tau}\|^2 \tag{3.6}$$

where $B^{(r)}$ is the matrix containing the basis elements for the r-th patch. Note that $\boldsymbol{\tau}$ is the same for all patches. This optimization can be performed iteratively. The regularization parameters α and β are set to 0.1 and 0.01, respectively.

However, the iterations are costly, and we can speed up things by using the orthonormal basis Ξ. The blur is initially estimated *unconstrained* and then projected onto the orthonormal basis. In particular, we first minimize the fit of the general EFF forward model (without the basis) with an additional regularization term on the local blur kernels, i.e., we minimize

$$\|G\mathbf{y} - \sum_{r=1}^{R}\mathbf{k}^{(r)} * (\mathbf{w}^{(r)} \odot G\mathbf{x})\|^2 + \alpha \sum_{r=1}^{R}\|G\mathbf{k}^{(r)}\|^2 + \beta \sum_{r=1}^{R}\|\mathbf{k}^{(r)}\|^2 \tag{3.7}$$

This optimization problem is approximately minimized using a single step of direct deconvolution in Fourier space, i.e.,

$$\mathbf{k}^{(r)} \approx F^{\mathsf{H}} \frac{\overline{FC_rG\mathbf{x}} \odot (FC_r \operatorname{Diag}(\mathbf{w}^{(r)})\, C_y^{\mathsf{T}}\mathbf{y})}{|FC_rG\mathbf{x}|^2 + \alpha \sum_i |FC_{g_i}^{\mathsf{T}}\mathbf{g}_i|^2 + \beta} \qquad \text{for all } r. \tag{3.8}$$

This approximation is inspired by Eq. (8) in [Hir+11]. The matrices C and C^{T} crop and zero-pad image and kernel appropriately, and G is assumed to be expressible with convolution filters $\mathbf{g}_i$ as in Eq. (2.6).

Finally, the resulting unconstrained blur kernels $\mathbf{k}^{(r)}$ are projected onto the orthonormal basis Ξ, leading to the estimate of the blur parameters $\boldsymbol{\mu}$.

3. **Image estimation:** For image estimation given the blurry image $\mathbf{y}$ and blur parameters $\boldsymbol{\mu}$, we apply Tikhonov regularization with $\gamma = 0.01$ on the gradients of the latent image $\mathbf{x}$, i.e.,

$$\|\mathbf{y} - \boldsymbol{\mu} \diamond \mathbf{x}\|^2 + \gamma\|G\mathbf{x}\|^2. \tag{3.9}$$

As in [Hir+11] Eq. (8), this expression can be approximately minimized with respect to $\mathbf{x}$ using a single step of the following direct deconvolution:

$$\mathbf{x} \approx \mathbf{v} \odot \sum_r C_r^{\mathsf{T}} F^{\mathsf{H}} \frac{\overline{FC_b^{\mathsf{T}}\Xi\boldsymbol{\mu}} \odot (FC_r \operatorname{Diag}(\mathbf{w}^{(r)})\, C_y^{\mathsf{T}}\mathbf{y})}{|FC_b^{\mathsf{T}}\Xi\boldsymbol{\mu}|^2 + \gamma \sum_i |FC_{g_i}^{\mathsf{T}}\mathbf{g}_i|^2}. \tag{3.10}$$

with the appropriate zero-padding and cropping matrices C_b^{T}, C_y^{T}, $C_{g_i}^{\mathsf{T}}$ and C_r. The normalization factor $\mathbf{v}$ accounts for artifacts at patch boundaries which originate from windowing (see [Hir+11]) and is obtained by applying the inversion to a constant image.

image	image dims	local blur	patches	using B	using Ξ	NBD
bridge	2601 × 1733	19×19	10×8	127 sec	16 sec	1.4 sec
bench	1097 × 730	81×81	10×6	85 sec	14 sec	0.7 sec
historical	2191 × 1464	29×29	10×6	103 sec	13 sec	1.0 sec
facade	2817 × 1877	29×29	12×8	166 sec	21 sec	1.7 sec
	(a)	(b)	(c)	(d)	(e)	(f)

Table 3.1.: Runtime comparison for blind removal of optical aberrations: (a) image sizes, (b) size of the local blur kernels, (c) number of patches horizontally and vertically, (d) runtime of PSF estimation using the overcomplete basis B (see Eq. (3.6)), (e) runtime of PSF estimation using the orthonormal basis Ξ (see Eq. (3.7)) as used in our approach, (f) runtime of the final non-blind deconvolution.

Similar to [CL09] and [Hir+11] the algorithm follows a coarse-to-fine approach, solving first a downsampled version of the problem, and then repeatedly initializing the next, larger scale with the kernel estimate from the smaller scale. Having estimated the final blur parameters $\boldsymbol{\mu}$ we use a non-uniform version of Krishnan and Fergus' approach [KF09; Hir+11] for the non-blind deconvolution to recover a high-quality estimate of the true image, similar to Eq. (2.5). The solution of the $\mathbf{x}$-subproblem is based on the direct deconvolution formula Eq. (3.10) instead of Eq. (2.6).

3.6. Implementation and running times

The algorithm is implemented on a GPU in Python using PyCUDA[1]. All experiments were performed on a 3.0 GHz Intel Xeon with an NVIDIA Tesla C2070 GPU with 6 GB of memory. The basis elements generated as detailed in Section 3.3 are orthogonalized using the SVDLIBC library[2]. Calculating the SVD for the occurring large sparse matrices can require a few minutes of running time. However, the basis is independent of the image content, so we can compute the orthonormal basis once and reuse it. Table 3.1 reports the running times of our experiments for both PSF and final non-blind deconvolution along with the EFF parameters and image dimensions. In particular, it shows that using the orthonormal basis instead of the overcomplete one improves the running times by a factor of about six to eight.

3.7. Results

In the following, we show results on real photos and do a comprehensive comparison with other approaches for removing optical aberrations. Image sizes and blur parameters are shown in Table 3.1.

[1] `http://mathema.tician.de/software/pycuda`
[2] `http://tedlab.mit.edu/~dr/SVDLIBC/`

3.7.1. Self-built lens with a single lens element

Chapter 2 showed deblurring results on images taken with a lens that consists only of a single element, thus exhibiting strong optical aberrations, in particular coma. The previous approach is non-blind and measures the non-uniform PSF with a point source and applies non-blind deconvolution. In contrast, the current chapter's approach is blind and is directly applied to the blurry image.

To better approximate the large blur of that lens, we additionally assume that the local blurs scale linearly with radial position, which can be easily incorporated into our basis generation scheme. For comparison, we apply Photoshop's "Smart Sharpening" function for removing lens blur. It depends on the blur size and the amount of blur, which are manually controlled by the user. Thus we call this method *semi-blind* since it assumes a parametric form. Even though we choose its parameters carefully, we are not able to obtain comparable results.

Comparing our blind method against the non-blind approach of the previous chapter, we observe that our estimated PSF matches the measured PSFs rather well (see Fig. 3.9). However, surprisingly we do get an image that may be considered sharper. The reason could be over-sharpening or a less conservative regularization in the final deconvolution; it is also conceivable that the calibration procedure used previously is not sufficiently accurate. Note that neither DXO nor Kee et al.'s approach can be applied, lacking calibration data for this lens.

3.7.2. Canon 24mm f/1.4

The PSF constraints we are considering assume local axial symmetry of the PSF with respect to the radial axis. For the Canon 24mm f/1.4 lens also used in the previous chapter, this is not exactly fulfilled, which can be seen in the inset in Fig. 3.10. The wings of the green blur do not have the same length. Nonetheless, our blind estimation with enforced symmetry still approximates the PSF shape well and yields a comparable quality of image correction. We stress the fact that this was obtained blindly in contrast to Chapter 2.

3.7.3. Kee et al.'s image

Figure 3.8 shows results on an image taken from Kee et al. [Kee+11]. The close-ups reveal that Kee's *non-blind* approach is slightly superior in terms of sharpness and noise-robustness. However, our *blind* approach removes chromatic aberration to a greater degree. A general problem of methods relying on a prior calibration is that optical aberrations depend on the wavelength of the transmitting light continuously: an approximation with only a few (generally three) color channels therefore depends on the lighting of the scene and could change if there is a discrepancy between the calibration setup and a photo's lighting conditions. This is avoided with a blind approach.

We also apply "DxO Optics Pro 7.2" to the blurry image. DXO uses a database for combinations of cameras/lenses. While it uses calibration data, it is not clear whether it additionally infers elements of the optical aberration from the image. For comparison, we process the photo with the options "chromatic aberrations" and "DxO lens softness" set to their default values.

Blurred image | Our approach (blind)

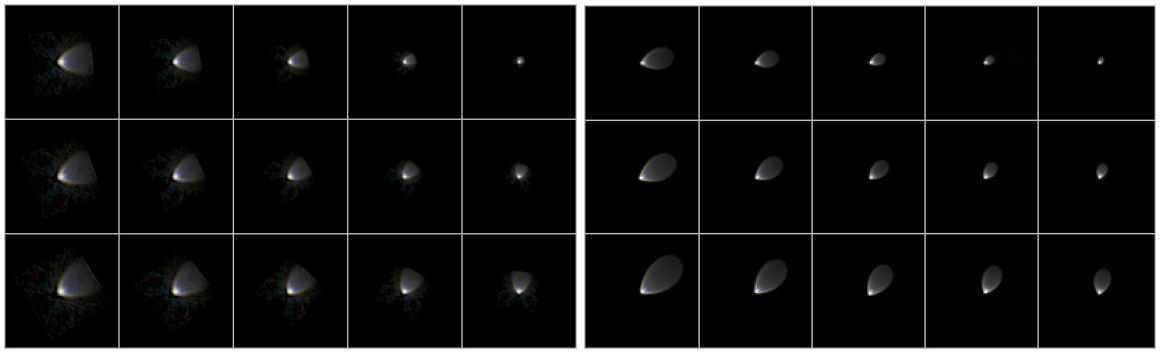
Adobe's "Smart Sharpen" (semi-blind) | Approch of Chapter 2 (non-blind)

Figure 3.7.: Comparison with non-blind approach, self-built lens. Full image and lower left corner.

Blurred image | Our approach (blind)

DXO | Kee et al. [Kee+11] (non-blind)

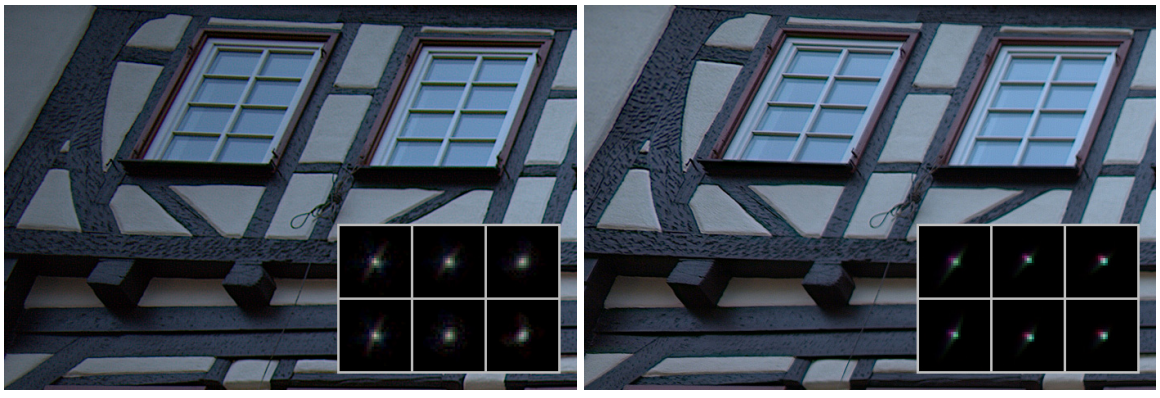
Figure 3.8.: Comparison between blind approach and two non-blind approaches of Kee et al. [Kee+11] and DXO. Full image and upper right corner.



Blindly estimated by our approach | Measured

Figure 3.9.: PSF comparison with non-blind approach, self-built lens. Lower left corner of the PSF.

Blurred image



Our approach (blind) | Approch of Chapter 2 (non-blind)

Figure 3.10.: Comparison with non-blind approach, Canon 24mm f1/4 lens. Shown is the upper left corner of the image. PSF inset is three times the original size.

Blurred image | Our approach (blind) | Adobe's "Smart Sharpen" (semi-blind)

Figure 3.11.: Comparison on historical image from 1940.

The result is good and exhibits less noise than the other two approaches (see Fig. 3.8), however, it is not clear if an additional denoising step is employed by the software.

3.7.4. Historical images

A *blind* approach to removing optical aberrations can also be applied to historical photos, where information about the lens is not available. The left column of Fig. 3.11 shows a photo (and some detail) from the Library of Congress archive that was taken around 1940[3]. Assuming that the analog film used has a sufficiently linear light response, we applied our blind lens correction method and obtained a sharper image. However, the blur appeared to be small, so algorithms like Adobe's "Smart Sharpen" also give reasonable results. Note that neither DXO nor Kee et al.'s approach can be applied here since lens data is not available.

3.8. Conclusion

We have proposed a method to blindly remove spatially-varying blur caused by imperfections in lens designs, including chromatic aberrations. Without relying on elaborate calibration procedures, results comparable to non-blind methods can be achieved. By creating a suitable orthonormal basis, the PSF is constrained to a class that exhibits the generic symmetry properties of lens blurs, while fast PSF estimation is possible.

[3] `http://www.loc.gov/pictures/item/fsa1992000018/PP/`

3.8.1. Limitations

Our assumptions about the lens blur are only an approximation for lenses with poor rotation symmetry.

The image prior used in this work is only suitable for natural images, and is hence content specific. For images containing only text or patterns, in particular camera test charts used for quantifying the image quality, this would not be ideal.

3.8.2. Future work

While it is useful to be able to infer the image blur from a single image, the PSF does not change for photos taken with the same lens settings. On the one hand, this implies that we can transfer the PSFs estimated for these settings for instance to images where our image prior assumptions are violated. On the other hand, it suggests the possibility of improving the quality of the PSF estimates by utilizing a substantial database of images.

Finally, while optical aberrations are a major source of image degradation, a picture may also suffer from motion blur. By choosing a suitable basis, these two effects could be combined. It would also be interesting to see if non-uniform motion deblurring could profit from a direct PSF estimation step as introduced in the present work.

CHAPTER 4

Learning Non-Blind Deconvolution

Image deconvolution is the ill-posed problem of recovering a sharp image, given a blurry one generated by a convolution. In this chapter, we deal with space-invariant, non-blind deconvolution.

Currently, the most successful methods involve a regularized inversion of the blur in Fourier domain as a first step. This step amplifies and colors the noise, and corrupts the image information. In a second (and arguably more difficult) step, one then needs to remove the colored noise, typically using a cleverly engineered algorithm. However, the methods based on this two-step approach do not properly address the fact that the image information has been corrupted.

In this chapter, we also rely on a two-step procedure, but learn the second step on a large dataset of natural images, using a neural network. We will show that this approach outperforms the previous state of the art on a large dataset of artificially blurred images. We demonstrate the practical applicability of our method in a real-world example with photographic out-of-focus blur.

The material of this chapter is based on the following publication:

[Sch+13b] *C. J. Schuler, H. C. Burger, S. Harmeling, and B. Schölkopf. "A Machine Learning Approach for Non-blind Image Deconvolution". In:* IEEE Conf. Computer Vision and Pattern Recognition. *2013, pp. 1067–1074.* DOI: *10.1109/cvpr.2013.142*

4.1. Introduction

In the previous chapters we discussed how to remove image blur caused by imperfections in the optical system. Specifically, in Chapter 2 we assumed that the blur is known, and the main challenge we were confronted with was that lens blur was non-stationary.

However, as we discussed in Section 1.2, even *stationary* convolutions with known kernels are hard to invert, especially in the case of noise, where the blurry image $\mathbf{y}$ is given by $\mathbf{y} = \mathbf{x} * \mathbf{k} + \mathbf{n}$. Solving this ill-posed problem as best as possible is the foundation for also improving performance in the spatially-varying applications of the previous chapters.

Therefore, we now address space-invariant non-blind deconvolution, i.e., we want to recover $\mathbf{x}$ given $\mathbf{y}$ and $\mathbf{k}$ and assume $\mathbf{k}$ to be constant (space-invariant) over the image. Even though this is a long-standing problem, it turns out that there is room for improvement over the best existing methods. While most methods are well-engineered algorithms, we ask the question: is it possible to automatically learn an image deconvolution procedure? We will show that this is indeed possible.

Main contributions: We present an image deconvolution procedure that is *learned* on a large dataset of natural images with a neural network. We compare our approach to other methods on a large dataset of synthetically blurred images, and obtain state-of-the-art results for all tested blur kernels. Our method also achieves excellent results on a real photograph corrupted by out-of-focus blur. The execution time of our approach is reasonable (once trained for a specific blur) and scales linearly with the size of the image. We provide a toolbox on our website to test our method.[1]

4.2. Related work

Image deconvolution methods can be broadly separated into two classes. The first class of methods is based on probabilistic image priors, whereas the second class of methods relies on a preprocessing step followed by denoising.

Levin et al. [Lev+07], Krishnan and Fergus [KF09], FoE [SSR11], and Zoran and Weiss [ZW11] belong to the first category. Levin et al., Krishnan and Fergus, and EPLL seek a MAP estimate of the clean image $\mathbf{x}$, given a blurry (and noisy) version $\mathbf{y}$ and the PSF $\mathbf{k}$. In other words, one seeks to find the $\mathbf{x}$ maximizing $p(\mathbf{x}|\mathbf{y},\mathbf{k}) \propto p(\mathbf{y}|\mathbf{x},\mathbf{k})p(\mathbf{x})$. The first term is a Gaussian likelihood, but modeling the marginal distribution of images $p(\mathbf{x})$ is a long-standing research problem and can be handled in a number of ways. Levin et al. as well as Krishnan and Fergus assume that the image gradients follow a hyper-Laplacian distribution (this is a common and well-founded assumption, see Fig. 1.5 or [SA96]). EPLL [ZW11] models $p(\mathbf{x})$ using a Gaussian mixture model. FoE [SSR11] uses a Bayesian MMSE instead of a MAP estimate and uses the Fields of Experts [RB09] framework to model $p(\mathbf{x})$.

The second category of methods applies a regularized inversion of the blur, followed by a denoising procedure. In Fourier domain, the inversion of the blur can be seen as a pointwise division by the blur kernel. This makes the image sharper, but also has the effect of amplifying the noise, as well as creating correlations in the noise, see Fig. 4.1. Hence, these methods address deconvolution as a *denoising* problem. Unfortunately, most denoising methods are designed to remove additive white Gaussian (AWG) noise [Por+03; EA06; Dab+07]. Deconvolution via denoising requires the denoising algorithm to be able to remove colored noise (non-flat power

[1] `http://webdav.is.mpg.de/pixel/neural_deconvolution/`

spectrum of the noise, not to be confused with color noise of RGB images). Methods that are able to remove colored noise, such as DEB-BM3D [Dab+08], IDD-BM3D [DKE12] and others (e.g. [GMP08]) have been shown to achieve good deconvolution results. A method published at the same time as the material of this chapter [Sch+13a] does not completely fit this second category. While also a discriminative approach without generative prior, it does not employ an explicit preprocessing step, but trains a model consisting of a cascade of Gaussian conditional random fields.

Image denoising is itself a well-studied problem, with methods too numerous to list here. Some approaches to denoising rely on learning, where learning can involve learning a probabilistic model of natural images [RB09], or of smaller natural image patches [ZW11]. In that case, denoising can be achieved using a maximum a posteriori method. In other cases, learning involves learning a discriminative model for denoising, for example using convolutional neural networks [LeC+98a]. In [JS08], it is shown that convolutional neural networks can achieve good image denoising results for AWG noise.

More recently, it was shown that a type of neural network based on stacked denoising auto-encoders [Vin+10] can achieve good results in image denoising for AWG noise as well as for "blind" image inpainting [XXC12] (when the positions of the pixels to be inpainted are unknown).

Also recently, plain neural networks achieved state-of-the-art results in image denoising for AWG noise, provided the neural nets have enough capacity and that sufficient training data is provided [BSH12c; BSH12a]. It was also shown that plain neural networks can achieve good results on other types of noise, such as noise resembling stripes, salt-and-pepper noise, JPEG-artifacts and mixed Poisson-Gaussian noise.

Differences and similarities to our work: We address the deconvolution problem as a denoising problem and therefore take an approach that is in line with [Dab+08; DKE12; GMP08], but different from [KF09]. However, as opposed to *engineered* algorithms [Dab+08; DKE12; GMP08], ours is *learned*. In that respect, we are similar to [RB09; ZW11; Sch+13a]. However, our method is a discriminative method, and therefore more in line with [JS08; XXC12; BSH12c]. We make no effort to use specialized learning architectures [JS08; XXC12] but use multilayer perceptrons, similar to [BSH12c].

4.3. Method

The most direct way to deconvolve images with neural networks is to train them directly on blurry/clean patch pairs. However, as we will see in Section 4.4, this does not lead to good results. Instead, our method relies on two steps, as illustrated in Fig. 4.2: (i) a regularized inversion of the blur in Fourier domain and (ii) a denoising step using a neural network. In this section, we describe these two steps in detail.

4.3.1. Direct deconvolution

The goal of this step is to make the blurry image sharper. This has the positive effect of localizing information, but it has the negative side-effect of introducing new artifacts. In our

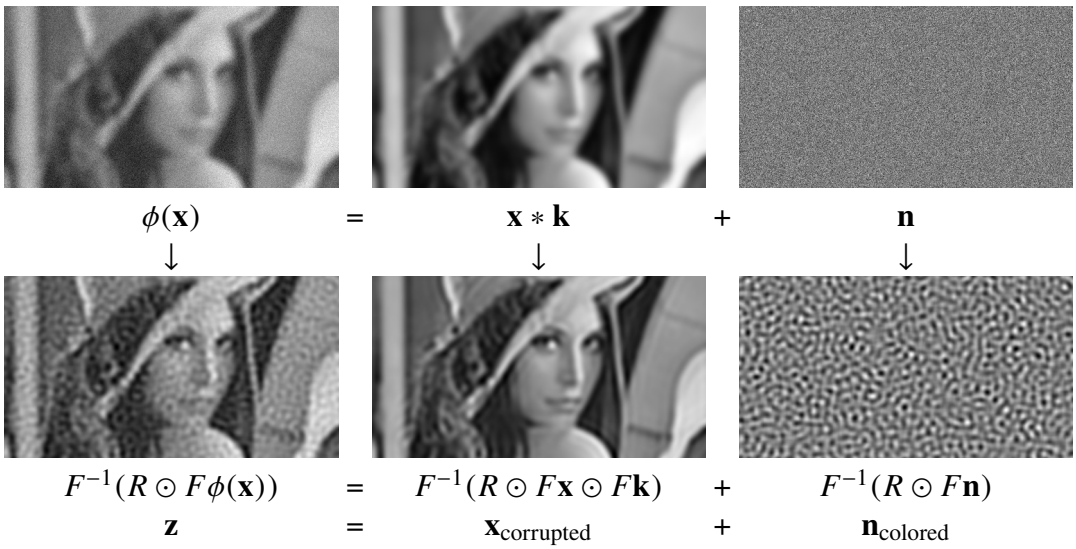


Figure 4.1.: Illustration of the effect of the regularized blur inversion. The goal of image deconvolution is to deblur $\mathbf{y}$. The result $\mathbf{z}$ of the regularized inversion is the sum of a corrupted image $\mathbf{x}_{\text{corrupted}}$ and colored noise $\mathbf{n}_{\text{colored}}$. Other methods [Dab+08; DKE12; GMP08] attempt to remove $\mathbf{n}_{\text{colored}}$ but ignore the noise in $\mathbf{x}_{\text{corrupted}}$, whereas our method learns to denoise $\mathbf{z}$ and therefore addresses both problems.

model, the underlying true (sharp) image $\mathbf{x}$ is blurred with a PSF $\mathbf{k}$ and corrupted with AWG noise $\mathbf{n}$ with standard deviation σ:

$$\mathbf{y} = \mathbf{k} * \mathbf{x} + \mathbf{n}. \tag{4.1}$$

Assuming a Gaussian prior on the gradients of $\mathbf{x}$, and that our measurement of the blur kernel is corrupted by AWG, we know from Section 1.2.2 that the MMSE estimator is found by minimizing

$$\|\mathbf{y} - \mathbf{k} * \mathbf{x}\|^2 + \alpha\sigma^2\|G\mathbf{x}\|^2 + \beta\|\mathbf{x}\|^2, \tag{4.2}$$

where similar to Section 2.6 we assumed that our measurement of the blur kernel could be corrupted by AWG, adding a further term $\beta\|\mathbf{x}\|^2$ (see Sec. 6.4.1 in [BV04]). As noted previously, this can be solved in Fourier domain in a single step in the case of circular convolution (cf. Eq. (1.17)). We introduce the regularized inverse of the blurring transformation as

$$R = \frac{\overline{F\mathbf{k}}}{|F\mathbf{k}|^2 + \alpha\sigma^2 \sum_i |F\mathbf{g}_i|^2 + \beta}, \tag{4.3}$$

assuming for notational simplicity that $\mathbf{k}$ and $\mathbf{g}_i$ are zero-padded to the size of $\mathbf{x}$ and $\mathbf{y}$. The hyper-parameters α and β are responsible for the regularization: If both $\alpha = 0$ and $\beta = 0$, there is no regularization. Using the regularized inverse R, we can estimate the true image by the so-called *direct deconvolution* (following [Hir+11])

$$\begin{aligned}\mathbf{z} = F^{-1}(R \odot F\mathbf{k}) &= F^{-1}\left(R \odot (F\mathbf{x} \odot F\mathbf{k} + F\mathbf{n})\right) \\ &= F^{-1}(R \odot F\mathbf{x} \odot F\mathbf{k}) + F^{-1}(R \odot F\mathbf{n}),\end{aligned} \tag{4.4}$$

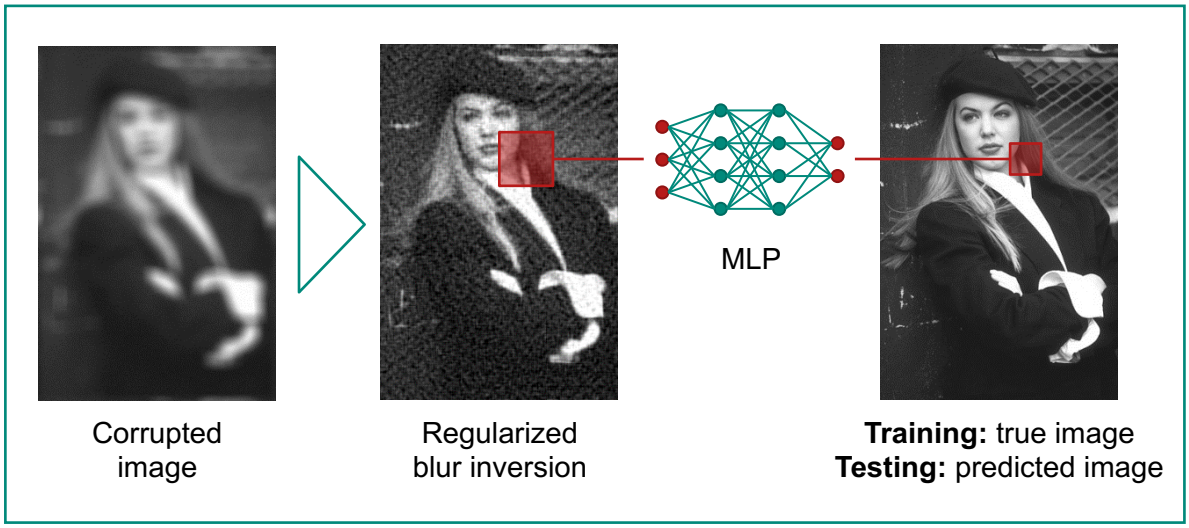


Figure 4.2.: Overview of our non-blind deconvolution approach. The method consists of two steps: first, the image is preprocessed by performing a regularized blur inversion, localizing the image information, but also introducing new artifacts. Next, a MLP predicts the ground truth image patch-wise from the the preprocessed image. The MLP has to be trained on a large dataset of patches.

where $\odot$ as previously defined is element-wise multiplication. Hence, the image recovered through the regularized inverse is given by the sum of the colored noise image $F^{-1}(R \odot F\mathbf{n})$ and an image $F^{-1}(R \odot F\mathbf{x} \odot F\mathbf{k})$ (as illustrated in Fig. 4.1). The latter image is exactly equivalent to $\mathbf{x}$ if $\alpha = \beta = 0$ and the blur kernel doesn't exhibit zeroes in its frequency spectrum, but otherwise generally not. We therefore see that methods trying to remove the colored noise component $F^{-1}(R \odot F\mathbf{n})$ ignore the fact that the image itself is corrupted. We propose as step (ii) a procedure that removes the colored noise and additional image artifacts.

4.3.2. Artifact removal by multilayer perceptrons

For our proposed method we use a simple NN, an MLP as introduced in Section 1.4, which is of the form

$$f(\mathbf{i}) = \mathbf{b}_3 + W_3 \tanh(\mathbf{b}_2 + W_2 \tanh(\mathbf{b}_1 + W_1\mathbf{i})), \tag{4.5}$$

with weight matrices W_i and vector-valued biases $\mathbf{b}_i$. We denote the *architecture* of an MLP by a tuple of integers, e.g. $(39^2, 2047, 2047, 2047, 2047, 13^2)$ describes an MLP with four hidden layers (each having 2047 nodes) and patches of size 39×39 as input, and of size 13×13 as output. Such an MLP has approximately 1.6×10^7 parameters to learn, which is similar in scale to other large networks reported in literature [Cir+10; SL11]. All our MLPs are *fully connected*, meaning that the weight matrices W_i are dense. One could also imagine MLPs which are not fully connected, using sparse weight matrices. Sparsely connected MLPs have the advantage of being potentially computationally easier to train and evaluate.

Training procedure: Our goal is to learn an MLP that maps corrupted input patches to clean output patches. How do we generate training examples? Starting with a clean image $\mathbf{x}$ from an image database, we transform it by a function ϕ that implements our knowledge of the image formation process. For instance, in the simulated experiment in Section 4.4.2, the clean image $\mathbf{x}$ is blurred by the PSF $\mathbf{k}$ and additionally corrupted by noise $\mathbf{n}$. In this case ϕ is equivalent to the linear blur model in Eq. (4.1).

The real-world photograph deblurred in Section 4.4.4 requires a more complicated ϕ as described in that section. We apply the direct deconvolution to $\phi(\mathbf{x})$ to obtain the image

$$\mathbf{z} = F^{-1}(R \odot F\phi(\mathbf{x})), \tag{4.6}$$

which is an image containing artifacts introduced by the direct deconvolution. Input-output pairs for training of the MLP are obtained by chopping $\mathbf{z}$ and $\mathbf{x}$ into patches. Using a large image database we can generate an abundance of training pairs.

The free parameters of the MLP are learned on such pairs of corrupted and clean image patches from $\mathbf{z}$ and $\mathbf{x}$, using SGD [LeC+98a]. The parameters of the MLP are then updated using the *back-propagation* algorithm [RHW86] explained in Section 1.4.1, minimizing the pixel-wise squared error between the prediction of the MLP and the clean patch. The use of the squared error is motivated by the fact that we are interested in optimizing the PSNR, which is monotonically related to the square error. We follow the setup described in [BSH12a] for the training procedure.

Specifically, this means:

- *Data normalization:* the mean and variance of the input data should be close to zero and one, respectively. For natural images with a range between 0 and 1, this is achieved by subtracting 0.5 and dividing by 0.2.
- *Weight initialization:* to use both the linear and the non-linear part of the tanh activation functions, the entries of the weight matrices W_i should be drawn from a uniform distribution between $-\sqrt{6/(n_i + n_{i+1})}$ and $+\sqrt{6/(n_i + n_{i+1})}$, where n_i is the number of input dimensions to the layer i. This procedure assumes that the input to the MLP is normalized as described above.
- *Choice of learning rate:* the global learning rate is adapted to the different number of parameters of each layer by dividing by the number of its input dimensions n_i. We use a global learning rate of 0.05 for all experiments.

We perform the training procedure on a modern GPU, resulting in a speedup factor of approximately an order of magnitude compared to a CPU implementation.

Application to images: To deblur an image, we first apply the direct deconvolution. The resulting image (showing characteristic artifacts) is then chopped into overlapping patches and each patch is processed separately by the trained MLP. The resulting reconstructed patches are placed at the locations over their corrupted counterparts and averaged in regions where they overlap. As described in [BSH12c], instead of choosing every sliding-window patch, we

use a stride size of 3 (we pick every third patch) to achieve a speed-up factor of 9, while still achieving excellent results. This way, we can remove artifacts from an image of size 512×512 in approximately one minute on a modern computer (on CPU in Matlab).

4.4. Results

4.4.1. Choice of parameter values

Which experimental setups lead to good results? To answer this question, we monitor the results achieved with different setups at different times during the training procedure. Figure 4.4 shows that the results tend to improve with longer training times, but that the choice of the MLP's architecture as well as of the regularization strength α during direct deconvolution is important. Using four hidden layers instead of one leads to better results, given the same setting for direct deconvolution. If four hidden layers are used, better results are achieved with $\alpha = 20$ than with $\alpha = 10$. This is explained by the fact that too weak a regularization amplifies the noise too much, making its removal more difficult and training potentially unstable. In our experiments, we use $\alpha = 20$ for the direct deconvolution and $(39^2, 4 \times 2047, 13^2)$ for the architecture.

As mentioned above, it is also conceivable to train directly on blurry/clean patch pairs (i.e., on pairs $\phi(\mathbf{x})$ and $\mathbf{x}$, instead of on pairs $\mathbf{z}$ and $\mathbf{x}$), but this leads to results that are approximately 1.5 dB worse after convergence (given the same architecture).

4.4.2. Comparison to other methods

To compare our approach to existing methods (described in Section 4.2), we first perform controlled experiments on a large set of images, where both the underlying true image and the PSF are known. Since the PSF is known exactly, we set β to zero. We train five MLPs, one for each of the following scenarios.

(a) Gaussian blur with standard deviation 1.6 (size 25×25) and AWG noise with $\sigma = 0.04$.

(b) Gaussian blur with standard deviation 1.6 (size 25×25) and AWG noise with $\sigma = 2/255$ (≈ 0.008).

(c) Gaussian blur with standard deviation 3.0 (size 25×25) and AWG noise with $\sigma = 0.04$.

(d) Square blur (box blur) with size 19×19 and AWG noise with $\sigma = 0.01$.

(e) Motion blur from [Lev+09] and AWG noise with $\sigma = 0.01$.

Scenarios (a) and (b) use a small PSF and (c) and (d) use a large PSF, whereas (b) and (d) use weak noise and (a) and (c) use strong noise. Scenarios (a), (b) and (c) have been used elsewhere, e.g., [DKE12]. All of these blurs are particularly destructive to high frequencies and therefore especially challenging to deblur. Scenario (e) uses a motion blur recorded in [Lev+09], which is easier to deblur. Each MLP is trained on randomly selected patches from about $1.8 \cdot 10^8$ photos from the ImageNet dataset [Den+09]. Results seem to converge after approximately

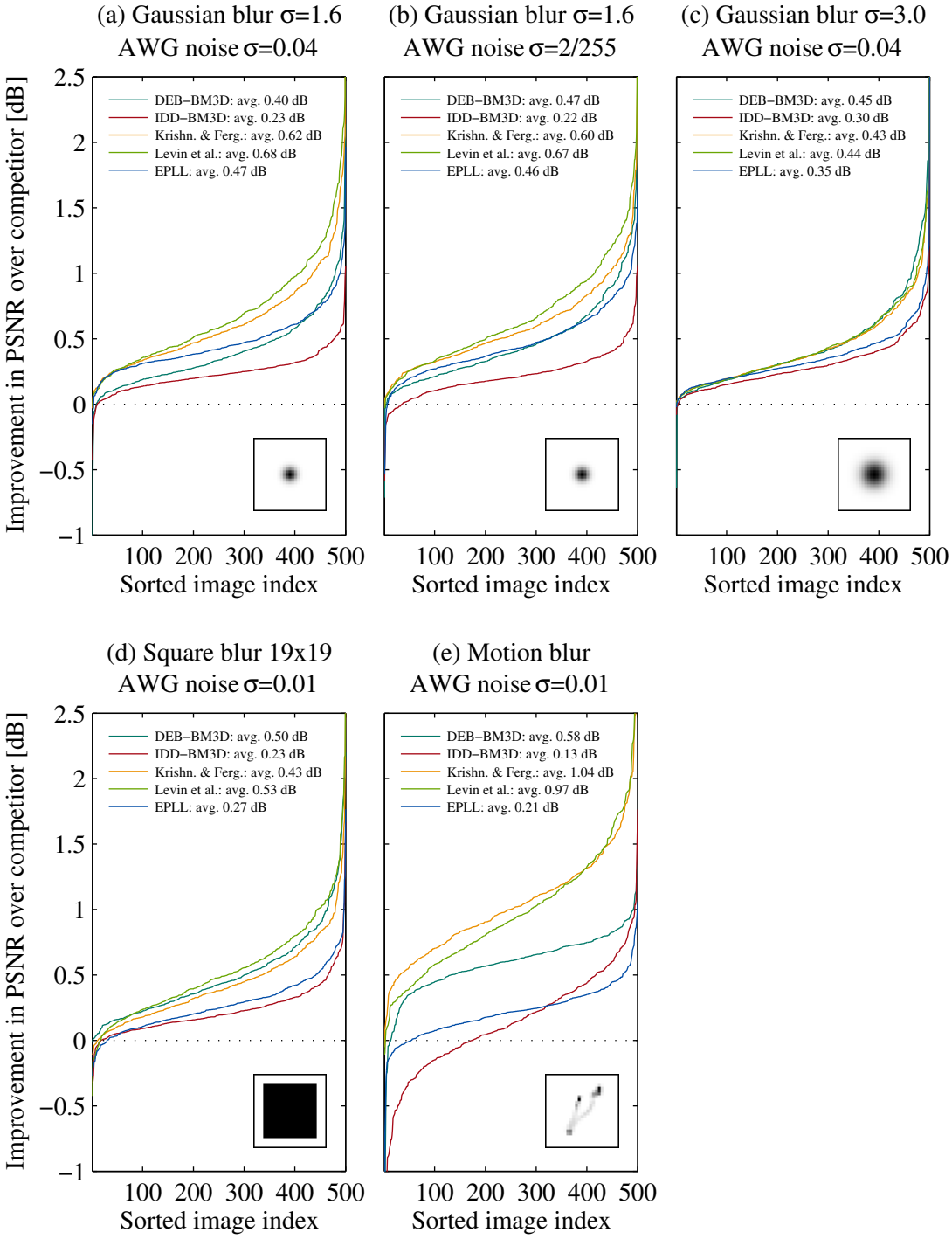


Figure 4.3.: Comparison of performance over competitors. Values above zero indicate that our method outperforms the competitor.

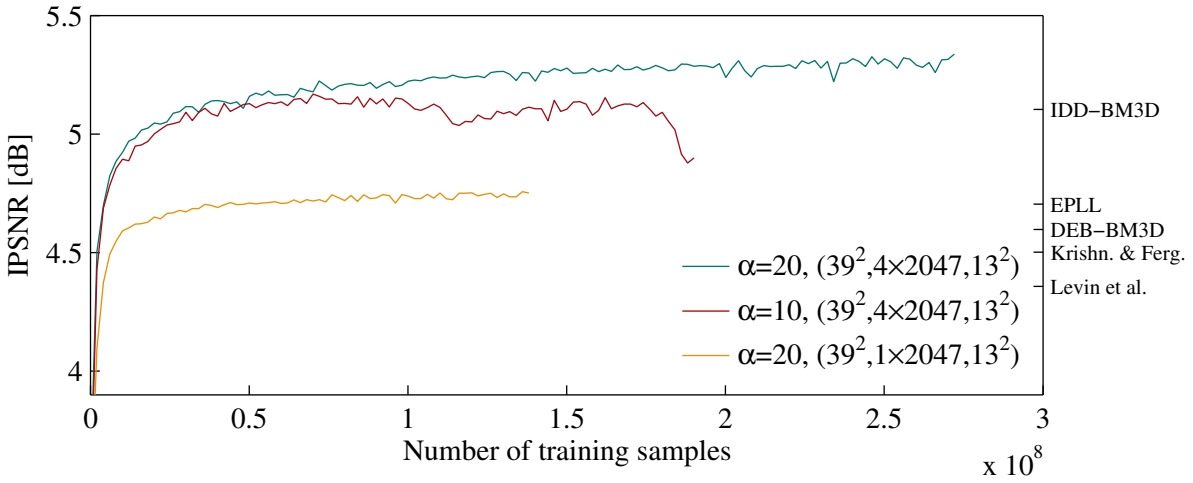


Figure 4.4.: MLPs with more capacity lead to better results. If the regularization in the direct deconvolution is weak, strong artifacts are created, leading to bad results. IPSNR refers to the mean improvement in PSNR over 11 test images over their blurry counterparts. A square blur was used to produce this figure. The labels on the right indicate the results achieved with competing methods.

$2 \cdot 10^8$ training samples, corresponding to two weeks of GPU time. However, most competing methods are surpassed within the first day of training.

We evaluate our method as well as all competitors on black-and-white versions of the 500 images of the Berkeley segmentation dataset. The exponent of the sparseness prior in Krishnan and Fergus [KF09] was set to 0.8. Levin et al. [Lev+07], EPLL [ZW11] and Krishnan and Fergus each require one regularization parameter, IDD-BM3D [DKE12] has two hyper-parameters. We optimized unknown values of these parameters on 20 randomly chosen images from ImageNet. Since only the methods using an image prior would be able to treat the boundary conditions correctly, we use circular convolution in all methods but exclude the borders of the images in the evaluation (we cropped by half the size of the blur kernel).

A performance profile of our method against all others on the full dataset is shown in Fig. 4.3 and two example images are shown in Figs. 4.6 and 4.7. Our method outperforms all competitors on most images, sometimes by a large margin (several dB). The average improvement over all competitors is significant. In Figs. 4.6 and 4.7 we see that in smooth areas, IDD-BM3D [DKE12] and DEB-BM3D [Dab+08] produce artifacts resembling the PSF (square blur), whereas our method does not. The results achieved by Levin et al. and Krishnan and Fergus look "grainy" and the results achieved by EPLL [ZW11] look more blurry than those achieved by our method. However, IDD-BM3D yields better results than our method in areas with repeating structures.

A comparison with the Fields of Experts based method [SSR11] was infeasible on the Berkeley dataset, due to long running times. Table 4.1 summarizes the results achieved on 11 standard test images for denoising [Dab+07], downsampled to 128×128 pixels.

For our scenarios IDD-BM3D is consistently the runner-up to our method. The other methods rank differently depending on noise and blur strength. For example, DEB-BM3D performs

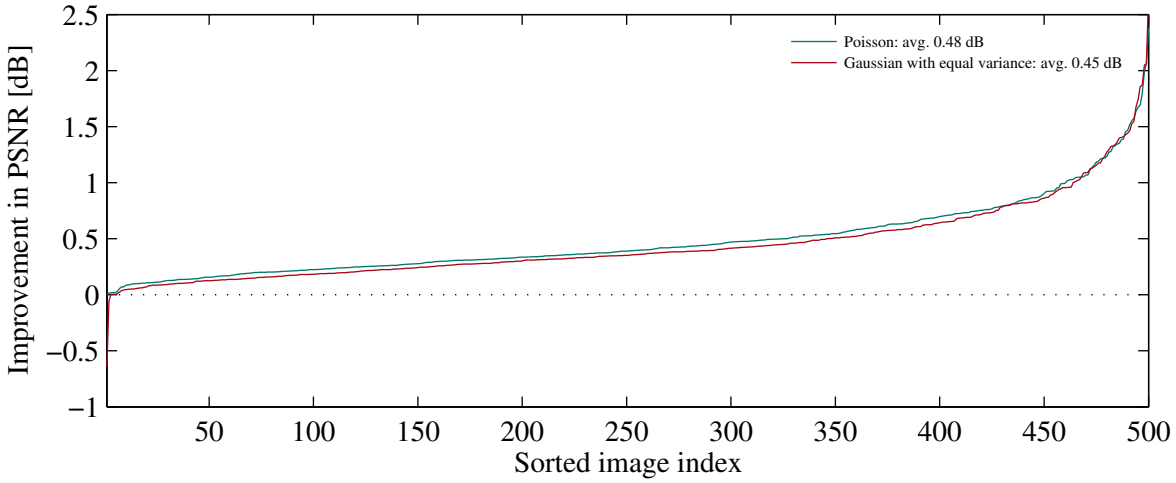


Figure 4.5.: Comparison of performance over BM3D for Poisson noise. The experiment is equivalent to scenario (c), but with Poisson noise of equal average variance instead of the Gaussian noise. Values above zero indicate that our method outperforms the competitor.

	(a)	(b)	(c)	(d)	(e)
EPLL [ZW11]	24.36	26.75	21.82	22.30	29.32
Levin et al. [Lev+07]	24.09	26.51	21.72	21.91	28.33
Krishnan and Fergus [KF09]	24.17	26.60	21.73	22.07	28.17
DEB-BM3D [Dab+08]	24.19	26.30	21.48	22.20	28.26
IDD-BM3D [DKE12]	24.68	27.13	21.99	22.69	**29.41**
FoE [SSR11]	24.07	26.56	21.61	22.04	28.83
MLP	**24.76**	**27.23**	**22.20**	**22.75**	**29.42**

Table 4.1.: Comparison on 11 standard test images. Values in dB.

well for the small PSFs.

Poisson noise: For scenario (c) we also consider Poisson noise with equivalent average variance. Poisson noise is approximately equivalent to additive Gaussian noise, where the variance of the noise depends linearly on the intensity of the underlying pixel. We compare against DEB-BM3D, for which we set the input parameter (the estimated variance of the noise) in such a way as to achieve the best results. Averaged over the 500 images in the Berkeley dataset, the results achieved with an MLP trained on this type of noise are slightly better (0.015 dB) than with equivalent AWG noise, whereas the results achieved with DEB-BM3D are slightly worse (0.022 dB) than on AWG noise (see also Fig. 4.5). The fact that our results become somewhat better is consistent with the finding that equivalent Poisson noise is slightly easier to remove [MF11]. We note that even though the improvement is slight, this result shows that MLPs are able to automatically adapt to a new noise type, whereas methods that are not based on learning would ideally have to be engineered to cope with a new noise type (e.g. [MF11]

describes adaptations to BM3D [Dab+07] for mixed Poisson-Gaussian noise, [CWL11] handles outliers in the imaging process).

4.4.3. Noise dependence

A disadvantage of our method is that it is only optimal for the noise level used during training. Other methods, e.g. DEB-BM3D, can incorporate this as a parameter during application. However, as we see in Fig. 4.8 the performance of the MLP degrades gradually and is still acceptable if the noise strength differs by a factor of 2.

Recently, Wang and Morel [WM14] showed that normalizing the input patches can help to remove the noise dependence of a denoising-type neural network. This is achieved by scaling input patches by a factor of $\sigma_{\text{trained}}/\sigma$, such that the standard deviation σ of the noise of the observed image is equal to the standard deviation σ_{trained} of the noise the network was trained on. However, this scaling may produce patches that are unlike the patches the MLP was trained on. To reduce this effect, the mean (of the image content, not of the noise) of every scaled patch is shifted to be equal to the mean where the training distribution of the MLP was densest, and where the MLP is most likely to have learned about patches similar to the rescaled patches. If we apply this procedure to a deconvolution-type neural network (scaling the observed image before the regularized blur inversion), we see from Fig. 4.8 that this extends the applicability of the MLP to noise strengths larger than it was trained on. The effect is reversed for small noise levels, presumably since in this case the scaling factor $\sigma_{\text{trained}}/\sigma > 1$ also increases artifacts created by the blur inversion.

4.4.4. Qualitative results on a real photograph

To test the performance of our method in a real-world setting, we remove defocus blur from a photograph. We use a Canon 5D Mark II with a Canon EF 85mm f/1.2 L II USM lens to take an out-of-focus image of a poster, see Fig. 4.9. In order to make the defocus blur approximately constant over the image plane, the lens is stopped down to f/5.6, which minimizes lens aberrations.

The function ϕ mimicking the image formation for this setup performs the following steps. First, an image from the training dataset is gamma-decompressed and transformed to the color-space of the camera (coefficients can be obtained from DCRAW). Then the image is blurred with a pillbox PSF (a circular averaging filter) with radius randomly chosen between 18.2 and 18.6. The radius of the actual PSF can be estimated by looking at the position of the first zero-frequency in Fourier domain. The randomness in the size of the pillbox PSF expresses that we do not know the exact blur and a pillbox is only an approximation. This is especially true for our lens stopped down by eight shutter blades. Then the color image is converted to four half-size, grayscale images to model the Bayer pattern. Next, noise is added to the image. The variance of readout noise is independent of the expected illumination, but photon shot noise scales linearly with the mean, and pixel non-uniformity causes a quadratic increase in variance [Mar]. Our noise measurements on light frames are in agreement with this and can therefore be modeled by a second-order polynomial. We have shown in Section 4.4.2 that our method is able to handle intensity-dependent noise.

Ground Truth

20.36 dB
Corrupted

26.49 dB
EPLL [ZW11]

25.81 dB
Krishn. & Ferg. [KF09]

25.76 dB
Levin et al. [Lev+07]

25.39 dB
DEB-BM3D [Dab+08]

26.44 dB
IDD-BM3D [DKE12]

27.02 dB
MLP

Figure 4.6.: Images from the best 5% results of scenario (d) as compared to IDD-BM3D [DKE12].

Ground Truth

19.34 dB
Corrupted

24.34 dB
EPLL [ZW11]

24.05 dB
Krishn. & Ferg. [KF09]

23.78 dB
Levin et al. [Lev+07]

24.25 dB
DEB-BM3D [Dab+08]

24.80 dB
IDD-BM3D [DKE12]

24.81 dB
MLP

Figure 4.7.: Images from the worst 5% results of scenario (d) as compared to IDD-BM3D [DKE12].

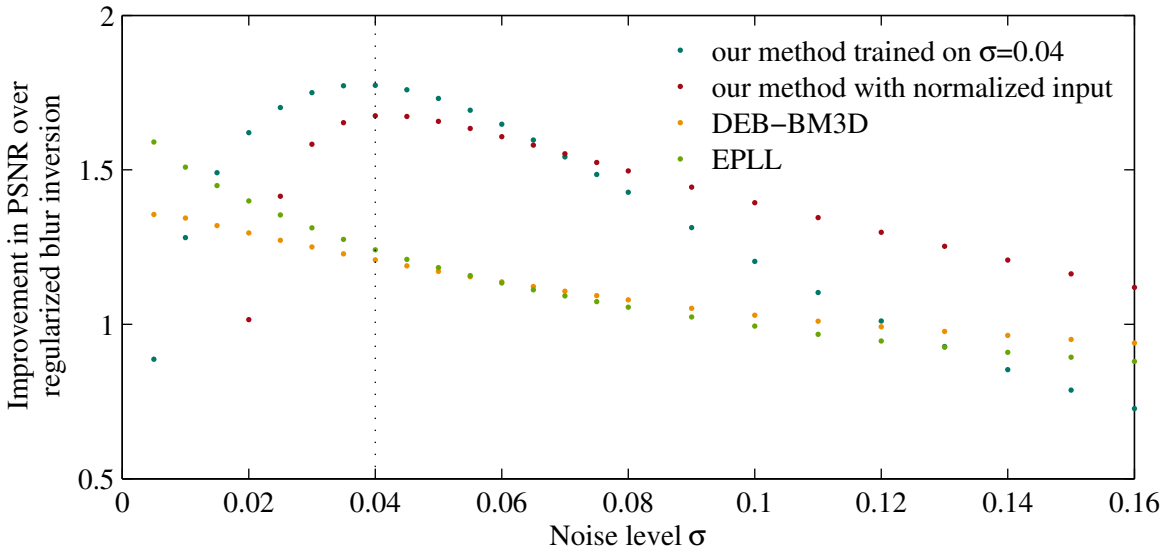


Figure 4.8.: Behavior of the MLP from scenario (c) at different noise levels, evaluated on 11 standard test images for denoising. The plot shows the improvement over the regularized blur inversion. DEB-BM3D and EPLL take only the noise level as a parameter and should be optimal at every noise level. The MLP was trained only on $\sigma = 0.04$, where its performance peaks, for other values it degrades slowly. With input normalized according to [WM14] the MLP can adapt to high noise levels.

To generate the input to the MLP we preprocess each of the four channels generated by the Bayer pattern via direct deconvolution using a pillbox of the corresponding size at this resolution (radius 9.2). Because of the uncertainty of the true kernel we set $\beta = 10^{-3}$. With this input, we learn the mapping to the original full resolution images with three color channels. The problem is higher-dimensional than in previous experiments, which is why we also increase the number of units in the hidden layers to 3071 (the architecture is therefore $(4 \times 39^2, 4 \times 3071, 3 \times 9^2)$). In Fig. 4.9 we compare to the best visual results we could achieve with DEB-BM3D, the top algorithm with only one tunable parameter. The results were obtained by first demosaicing and then deconvolving every color channel separately, as shown in Fig. 4.9.

In summary, we achieve a visually pleasing result by simply modeling the image formation process. By training on the full pipeline, we even avoid the need for a separate demosaicing step. It is not clear how this can be optimally incorporated in an engineered approach.

4.5. Understanding

Our MLPs achieve state-of-the-art results in image deblurring. But how do they work? In this section, we provide some answers to this question.

Following [BSH12b], we call weights connecting the input to the first hidden layer *feature de-*

Defocused Image

Direct Deconvolution

Krishnan and Fergus [KF09]
(deconv., then demosaic.)

DEB-BM3D [Dab+08]
(deconv., then demosaic.)

DEB-BM3D [Dab+08]
(demosaic., then deconv.)

MLP

Figure 4.9.: Removal of defocus blur in a photograph.

tectors and weights connecting the last layer to the output *feature generators*, both of which can be represented as patches. Assigning an input to an MLP and performing a forward pass assigns values to the hidden units, called *activations*. Finding an *input pattern* $\mathbf{i}^*$ that maximizes the activation of a specific hidden unit $a(\mathbf{i})$ can be performed using *activation maximization* [ECB10],

$$\mathbf{i}^* = \underset{\mathbf{i} \text{ s.t. } \|\mathbf{i}\|<\rho}{\operatorname{argmax}} \; a(\mathbf{i}). \tag{4.7}$$

The solution is constrained to have a norm smaller than ρ to prevent input patterns with values not observed during training. A local maximum to this non-convex optimization problem is found via gradient ascent.

We will analyze two MLPs trained on the square PSF from scenario (d), both with the architecture $(39^2, 4 \times 2047, 13^2)$. The first MLP is trained on patches that are preprocessed with direct deconvolution, whereas the second MLP is trained on the blurry image patches themselves (i.e., no preprocessing is performed).

Analysis of the feature detectors: We start with the feature detectors of the MLP trained with preprocessed patches, see Fig. 4.10. The feature detectors are of size 39×39 pixels. The area covered by the output patch lies in the middle of the patches and is of size 13×13 pixels. Some feature detectors seem to focus on small features resembling a cross. Others detect larger features in the area covered by the output patch (the middle 13×13 pixels). Still other feature detectors are more difficult to describe. Finally, some feature detectors detect edges that are completely outside the area covered by the output patch. A potential explanation for this surprising observation is that these feature detectors focus on artifacts created by the regularized inversion of the blur.

We perform the same analysis on the MLP trained on blurry patches, see Fig. 4.11. The shape of the blur is evident in most feature detectors: they resemble squares. In some feature detectors, the shape of the blur is not evident (the three rightmost). We also observe that all features are large compared to the size of the output patch (the output patches are three times smaller than the input patches). This was not the case for the MLP trained with preprocessing (Fig. 4.10) and is explained by the fact that in the blurry inputs, information is very spread out. We clearly see that the direct deconvolution has the effect of making the information more local.

Analysis of the feature generators: We now analyze the feature generators learned by the MLPs. We will compare the feature generators to the input patterns maximizing the activation of their corresponding unit. We want to answer the question: what input feature causes the generation of a specific feature in the output?

We start with the MLP trained on preprocessed patches. Figure 4.12 shows eight feature generators (bottom row) along with their corresponding input features (top row), maximizing the activation of the same hidden unit. The input patterns were found using activation maximization [ECB10]. Surprisingly, the input patterns look similar to the feature generators. We can interpret the behavior of this MLP as follows: if the MLP detects a certain feature in the corrupted input, it copies the same feature into the output.

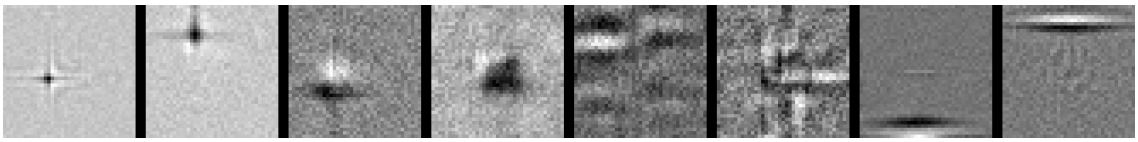


Figure 4.10.: Eight feature detectors of an MLP trained to remove a square blur. The MLP was trained on patches preprocessed with direct deconvolution. The two rightmost features detect edges that are outside the area covered by the output patch, presumably detecting artifacts.

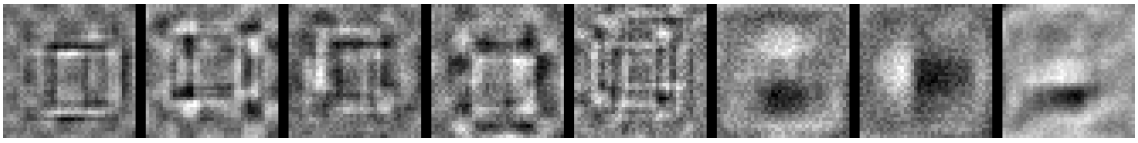


Figure 4.11.: Eight feature detectors of an MLP trained to remove a square blur. The MLP was trained on the blurry patches themselves (i.e., no preprocessing). The features are large compared to the output patches because the information in the input is very spread out, due to the blur.

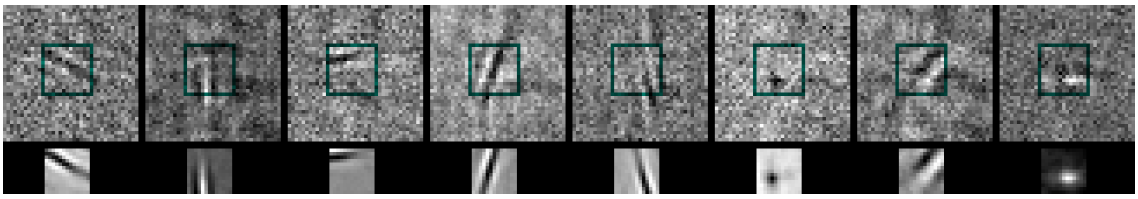


Figure 4.12.: Input patterns found via activation maximization [ECB10] (top row) vs. feature generators (bottom row) in an MLP trained on preprocessed patches. We see a clear correspondence between the input patterns and the feature generators. The MLP works by generating the same features it detects.

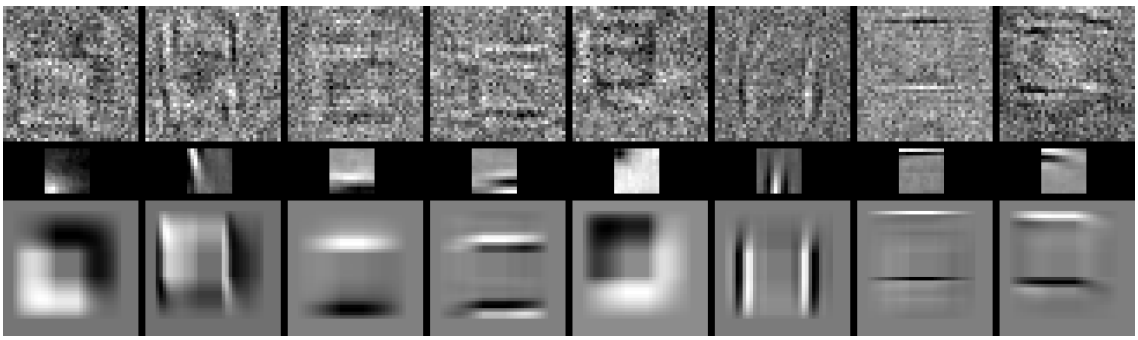


Figure 4.13.: Input patterns found via activation maximization [ECB10] (top row) vs. feature generators (middle row) in an MLP trained on blurry patches (i.e., no preprocessing). The input patterns look like the feature generators convolved with the PSF (bottom row). The MLP works by detecting blurry features and generating sharp ones.

We repeat the analysis for the MLP trained on blurry patches (i.e., without preprocessing). Figure 4.13 shows eight feature generators (middle row) along with their corresponding input features (top row). This time, the features found with activation maximization look different from their corresponding feature generators. However, the feature detectors look remarkably similar to the feature generators convolved with the PSF (bottom row). We interpret this observation as follows: if the MLP detects a blurry version of a certain feature in the input, it copies the (non-blurry) feature into the output.

Summary: Our MLPs are nonlinear functions with millions of parameters. Nonetheless, we were able to make a number of observations regarding how the MLPs achieve their results. This was possible by looking at the weights connecting the input to the first hidden layer and the weights connecting the last hidden layer to the output, as well as through the use of activation maximization [ECB10].

We have seen that the MLP trained on blurry patches has to learn large feature detectors, because the information in the input is very spread-out. The MLP trained on preprocessed patches is able to learn finer feature detectors. For both MLPs, the feature generators look similar: many resemble Gabor filters or blobs. Similar features are learned by a variety of methods and seem to be useful for a number of tasks [EA06; Vin+10]. We were also able to answer the question: which inputs cause the individual feature generators to activate? Roughly speaking, in the case of the MLP trained on preprocessed patches, the inputs have to look like the feature generators themselves, whereas in the case of the MLP trained on blurry patches, the inputs have to look like the feature generators convolved with the PSF. Additionally, some feature detectors seem to focus on typical preprocessing artifacts.

4.6. Convolutional training

4.6.1. Differences to patch-wise approach

Additionally, as an alternative to the training approach described in Section 4.3.2 and published in [Sch+13b], where the MLP was trained on patches from the images $\mathbf{z}$ and $\mathbf{x}$, it is also possible to train a network *convolutionally* on uncropped pairs of images $\mathbf{z}$ and $\mathbf{x}$. A convolutional network could be more suited to encode the translation invariance of the distribution of natural images.

Training is performed on whole images, not on cropped patches. The first layer creates several image-like like feature representations from the input image by performing convolutions with different filters. These filters will be learned during training. Next, every pixel in every feature representation is nonlinearly transformed with a tanh. The subsequent step then linearly recombines these transformed feature representations *pixel-wise* to a number of new feature representations, with weights learned during training. The last two steps, the tanh and the linear recombination, can be repeated several times, until finally the feature representations are combined to a single output, an estimate of the sharp image. The main difference to an MLP of the form of Eq. (4.5) therefore is the first layer, which creates a matrix where each row i is $\mathbf{z} * \mathbf{f}_i$, the input image $\mathbf{z}$ convolved with a learned filter $\mathbf{f}_i$.

Architecture	1×2048	4×128	4×256	4×512	4×1024
Avg. PSNR (dB)	24.24	24.67	24.76	24.78	24.59
PSNR after "fine-tuning"			24.83	24.85	

Table 4.2.: Results of convolutional neural networks for scenario (d). The architecture is described as number of hidden layers × number of hidden representations. The patch-wise MLP achieved a PSNR of 24.84 dB. For the two most successful architectures we continued training with a learning rate reduced by a factor of 10 (also called "fine-tuning" [BSH12b]).

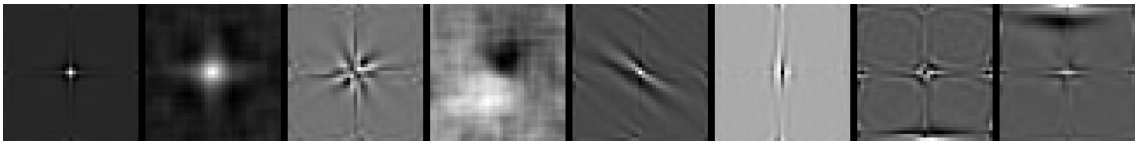


Figure 4.14.: Eight feature detectors of a convolutional NN trained to remove a square blur. The NN was trained on patches preprocessed with direct deconvolution. Note that all learned filters are centered, as opposed to Fig. 4.10.

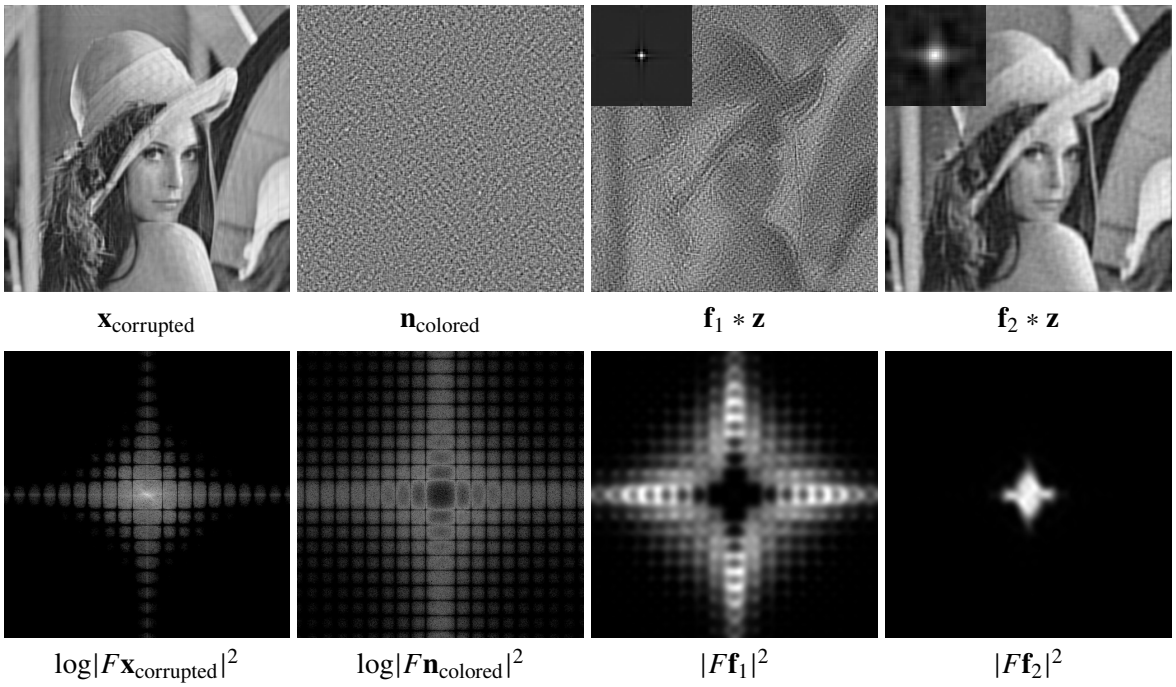


Figure 4.15.: Effect of two filters of a convolutional NN trained to remove a square blur. *Left two columns:* Corrupted image $\mathbf{x}_{\text{corrupted}}$ and colored noise $\mathbf{n}_{\text{colored}}$, and their respective power spectra in logarithmic scale. *Right two columns:* Two filters $\mathbf{f}_1$ and $\mathbf{f}_1$ applied to the preprocessed input (filters as inset), and their respective power spectra. The filter $\mathbf{f}_1$ concentrates mostly on the colored noise, $\mathbf{f}_2$ on the image information unaffected by noise.

This becomes more clear by comparing the convolutional procedure to a $(39^2, 256, 128, 1^2)$-MLP trained patch-wise: instead of multiplying a vectorized input patch with a 256×39^2 weight matrix, the first layer of the new network applies 256 convolutions with learned filters of size 39^2 to the image $\mathbf{z}$, creating 256 hidden image-like representations. After transforming every pixel in these hidden representations nonlinearly with a tanh, the next layer linearly recombines the 256 images to 128 new images. This is equivalent to applying a 128×256 weight matrix pixel-wise to every pixel across the 256 input images, as the MLP does for a single input vector of length 256. The last layer recombines the 128 hidden representations to a single prediction for $\mathbf{x}$, while the patch-wise MLP predicts only a single pixel. However, in both approaches, every pixel in the output is created from 39^2 pixels in the input, the size of the input patch or the size of the convolution filter.

We see that our proposed convolutional network is comparable to a patch-wise trained MLP with an output patch of size 1. For a patch-wise trained MLP smaller output patches reduce the dimensionality of the back-propagated gradient and lead to worse results [BSH12b]. The convolutional approach, on the other hand, back-propagates gradient information from the whole image, not from a single pixel, and consequently should not exhibit worse results.

To test the proposed approach, we trained several convolutional neural networks for scenario (d), each with a filter size of 39^2, but with different numbers of hidden representations and hidden layers. To avoid having to find a successful layer-wise adaption of the learning rate in SGD to the number of hidden representations, the filter size and the image size, we instead employed the adaptive training method ADADELTA (see Eq. (1.27)). A learning rate of 0.1 and a decay rate of 0.95 proved to work well. The results of different architectures are shown in Table 4.2. We see that larger architectures are more successful, but with diminishing returns. Deep models with too many hidden representations, however, seem to fail to converge to a good parameter setting.

Following [BSH12b], we also employ "fine-tuning". After training with a certain learning rate until convergence (which similar to the patch-wise MLP takes about two weeks on GPU), we continue training with a learning rate reduced by a factor of 10 (for about an additional two days). While this procedure did not significantly boost the patch-wise MLPs, it improves the results produced by the convolutional NNs. As can be seen from the bottom row of Table 4.2, the final convolutional performance is nearly identical to the patch-wise approach (within 0.01 dB for both the convolutional net with 256 and the one with 512 hidden representations). This means a convolutional net can achieve the same performance as patch-wise trained MLP with only 4% of the number of parameters. We will discuss a possible reason for this result in the next subsection. Even though the final performance and the training time are not improved as compared to patch-wise MLPs, the reduced number of parameters speeds up the application to blurry images and makes the convolutional model an interesting alternative to be further investigated, also with respect to denoising.

4.6.2. Understanding the learned filters

While the analysis of feature generators is not applicable to the convolutional approach, the learned filters are equivalent to feature detectors of the patch-wise MLPs.

From Fig. 4.14 we see that the filters after training have a structure comparable to the feature detectors we previously observed. Some feature detectors are cross-like, others focus on structures at the edge of the receptive field defined by the filter. However, unlike the feature detectors from Figs. 4.10 and 4.11, all filters possess a clear center. This could explain the observed competitive performance of a convolutional NN that possesses fewer parameters than a respective patch-wise MLP: the MLP uses capacity for learning to predict image structures shifted within the input and output patches, while the convolutional NN is translation invariant by design, since every pixel in its output image originates from a single region (as defined by the filter size) in the input image.

While the nonlinearities of the NN complicate interpreting the effect of a specific filter in the first layer, some insights into the cross-like structures can be gained — which to a certain extent occur in most of the convolutional filters, and in some of the feature detectors of the MLP — at the example of the first two filters from Fig. 4.14. In Fig. 4.15 on the left the power spectra of the corrupted image and the colored noise are shown. It is instructive to look at the power spectra of the convolution filters, which have a multiplicative effect on the power spectra of their input ($|F\mathbf{x}_{\text{output}}|^2 = |F\mathbf{f}|^2 |F\mathbf{x}_{\text{input}}|^2$). On the right, it can be seen that the power spectra of the two filters focus on different parts of the preprocessed blurry image: filter $\mathbf{f}_1$ is non-zero at frequencies where the colored noise is strong, filter $\mathbf{f}_2$ at frequencies of the corrupted image that are not affected by the noise.

4.7. Conclusion

We have shown that neural networks achieve a new state of the art in image deconvolution. This is true for all scenarios we tested. Our method presents a clear benefit in that it is based on learning: we do not need to design or select features or even decide on a useful transform domain, the neural network automatically takes care of these tasks. An additional benefit related to learning is that we can handle different types of noise, whereas it is not clear if this is always possible for other methods. Finally, by directly learning the mapping from a corrupted input to a clean output, we handle both types of artifacts introduced by the direct deconvolution, instead of being limited to removing colored noise. We were able to gain insight into how our networks operate: they detect features in the input and generate corresponding features in the output, either patch-wise or convolutionally. Our networks have to be trained on GPU to achieve good results in a reasonable amount of time, but once learned, deblurring on CPU is practically feasible. A limitation of our approach is that each network has to be trained on only one blur kernel: results achieved with networks trained on several blur kernels are inferior to those achieved with one trained on a single blur kernel. This makes our approach less useful for motion blurs, which are different for every image. However, in this case the deblurring quality is currently more limited by errors in the blur estimation than in the non-blind deconvolution step. Possibly our method could be further improved with a meta-procedure, such as [JNR12].

CHAPTER 5

Learning Blind Deconvolution

In this chapter, 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.

The material of this chapter is based on the following publication:

[Sch+14] *C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. "Learning to Deblur". In:* ArXiv e-prints *(2014). arXiv: 1406.7444 [cs.CV]. Submitted to a journal.*

5.1. Introduction

In Chapters 2 and 3 we have seen how to extend both non-blind and blind deconvolution to the problem of removing lens aberrations. In Chapter 4, we investigated the fundamental problem of non-blind deconvolution and demonstrated how to achieve state-of-the-art results with a pure learning based approach. The missing piece is extending the success of a learned method to the more challenging problem of *blind* deconvolution.

Similar to the previous chapter, we focus on *stationary* blurs, even though we also demonstrate how to extend this method to non-uniform corruptions. Hence, we again start with

$$\mathbf{y} = \mathbf{k} * \mathbf{x} + \mathbf{n}. \tag{5.1}$$

As explained in Section 1.3, the task of blind image deconvolution is to recover $\mathbf{x}$ given only the blurry image $\mathbf{y}$, without knowing $\mathbf{k}$ or the noise $\mathbf{n}$. A number of approaches [Fer+06; CL09; XJ10] introduced in Section 1.3.1 infer the true image implicitly or explicitly by regularizing towards a sharp solution. Usually, these methods are fast 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 (DNN) 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 [BG91] who formulated the idea of neural networks as general processing schemes with large numbers of parameters. This is in contrast to the previous chapter, which relied on a basic MLP.[1]

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.[2]

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.

5.2. Related work

Some recent approaches to blind deconvolution in photography have already been mentioned above and in Section 1.3. We refer the interested reader to [WZ13] 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 [ERH02; DeR+03], 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:* [Aiz+06] and similarly [KN11] 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:* [CD91] 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:* [TOM96] learns an inverse filter represented by a NN to deblur text.

[1] However, we could also view the regularized inversion step that preprocesses the MLP's input as its first layer, making the previous approach more similar to the idea of the current chapter.

[2] `http://webdav.is.mpg.de/pixel/neural_blind_deconvolution/`

- *NN to optimize a regularized least squares objective:* [SF94] 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 colored noise:* The previous chapter presented a method for non-blind deblurring that starts with a straight-forward division in Fourier space and then removes the resulting artifacts (mainly colored noise) with large neural networks.

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. [HHY10]. 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. Closest to our approach is the work of [Sch+13a], 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. (5.1).

Finally we note that *deconvolutional networks*, introduced in [Zei+10] and further extended in [ZTF11], 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.

5.3. Blind deconvolution as a layered network

As explained in Section 1.3.1, existing fast blind deconvolution methods work by alternating between the following steps [XZJ13; XJ10; CL09]:

1. *Feature extraction*

2. *Kernel estimation*

3. *Image estimation*

We will represent these steps by a trainable DNN, 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.

5.3.1. Architecture layout

Below, we describe the different parts of the network (cf. Fig. 5.1).

Feature extraction module

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,[3] creating several feature representations of the image. This is followed by two layers that introduce nonlinearity into the model. First, every value is transformed by a tanh-unit, then the feature representations are pixel-wise linearly combined to new hidden images, formally speaking

$$\tilde{\mathbf{y}}_i = \sum_j \alpha_{ij} \tanh(\mathbf{f}_j * \mathbf{y}) \qquad \text{and} \qquad \tilde{\mathbf{x}}_i = \sum_j \beta_{ij} \tanh(\mathbf{f}_j * \mathbf{y}) \tag{5.2}$$

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 α_{ij} and β_{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 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 similar to the convolutional network from Section 4.6, except that it can have multiple input and output images.

Kernel estimation module

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

$$\sum_i \|\tilde{\mathbf{k}} * \tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_i\|^2 + \beta_k \|\tilde{\mathbf{k}}\|^2 \tag{5.3}$$

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

[3] Note that this convolution has nothing to do with the convolution appearing in our image formation model (Eq. (5.1)) — 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.

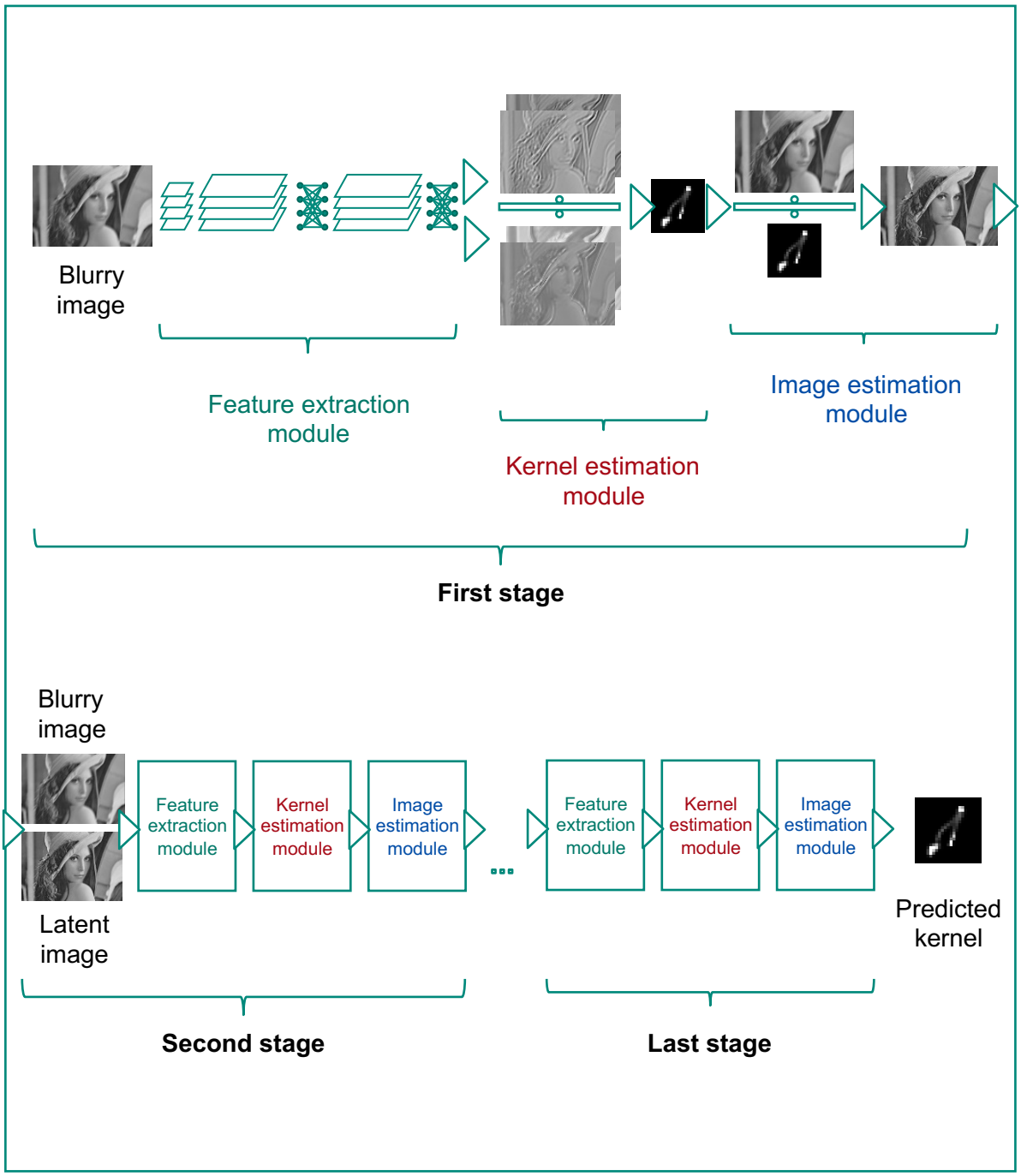


Figure 5.1.: Architecture of our proposed blind deblurring network. First the *feature extraction module* transforms the image to a learned gradient-like representation suitable for kernel estimation. Next, the kernel is estimated by division in Fourier space, then similarly the latent image. The next stages, each consisting of these three operations, operate on both the blurry image and the latent image.

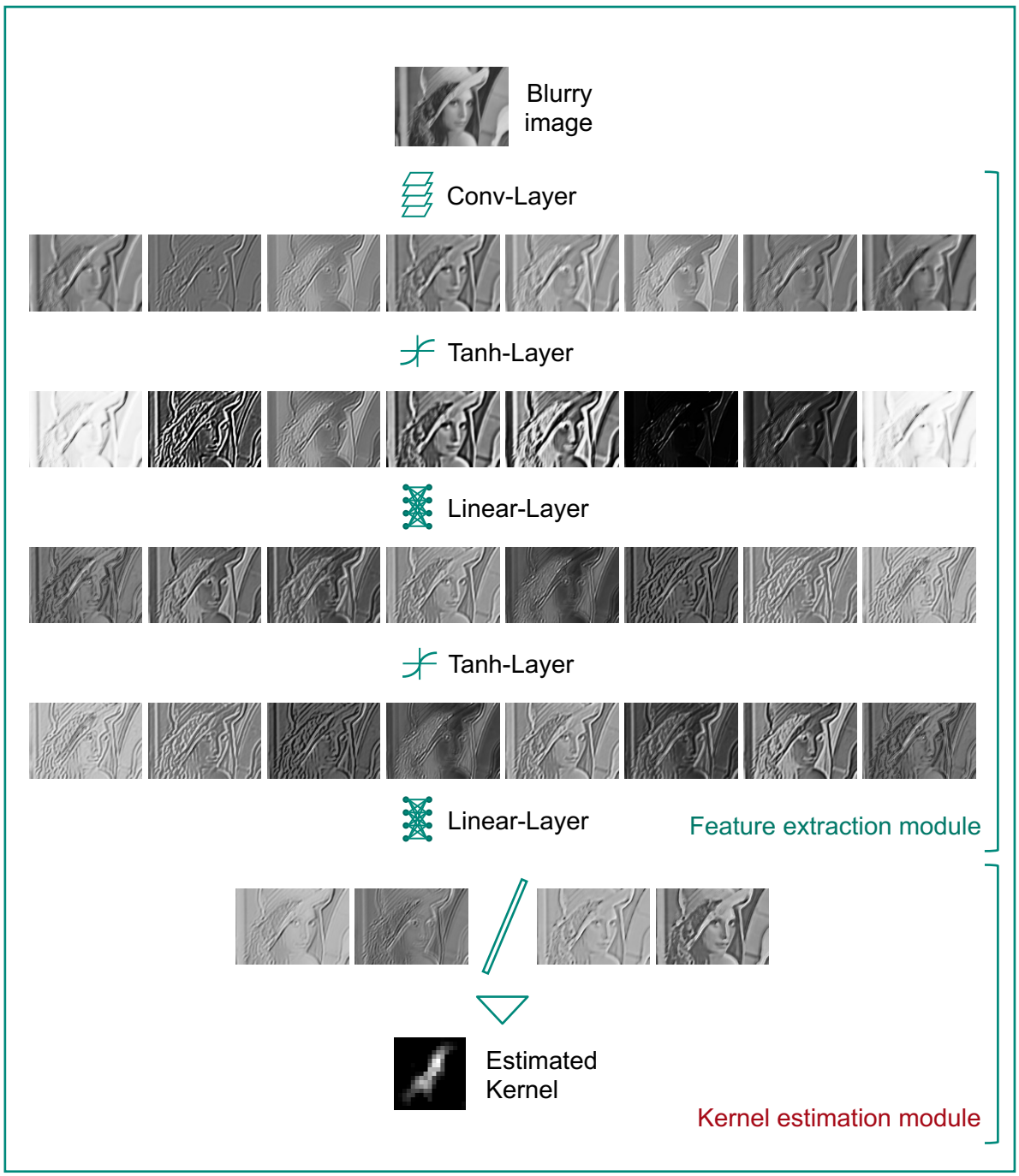


Figure 5.2.: Intermediary outputs of a single-stage NN with architecture $8 \times$ Conv, Tanh, 8×8, Tanh, 8×4.

representations predicted from the previous layer. The minimization problem can be solved in one step in Fourier space if we assume circular boundary conditions (cf. Eq. (1.17)):

$$\tilde{\mathbf{k}} = F^{\mathsf{H}} \frac{\sum_i \overline{F\tilde{\mathbf{x}}_i} \odot F\tilde{\mathbf{y}}_i}{\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k}. \tag{5.4}$$

This 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).

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. (5.3), we solve

$$\|\tilde{\mathbf{k}} * \tilde{\mathbf{x}} - \mathbf{y}\|^2 + \beta_x \|\tilde{\mathbf{x}}\|^2 \tag{5.5}$$

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 β_x is also learned during training.

5.3.2. Iterations as stacked networks

The *feature extraction module*, *kernel estimation module* and the *image estimation module* can be stacked several times, resulting in the network shown in Fig. 5.1, and 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 Appendix A for the derivatives of the solutions to Eq. (5.3) and the analogous Eq. (5.5)), increasing the performance as shown in Fig. 5.3 (but at the same time increasing runtime).

Similar to other blind deconvolution approaches and the approach of Chapter 3, 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.

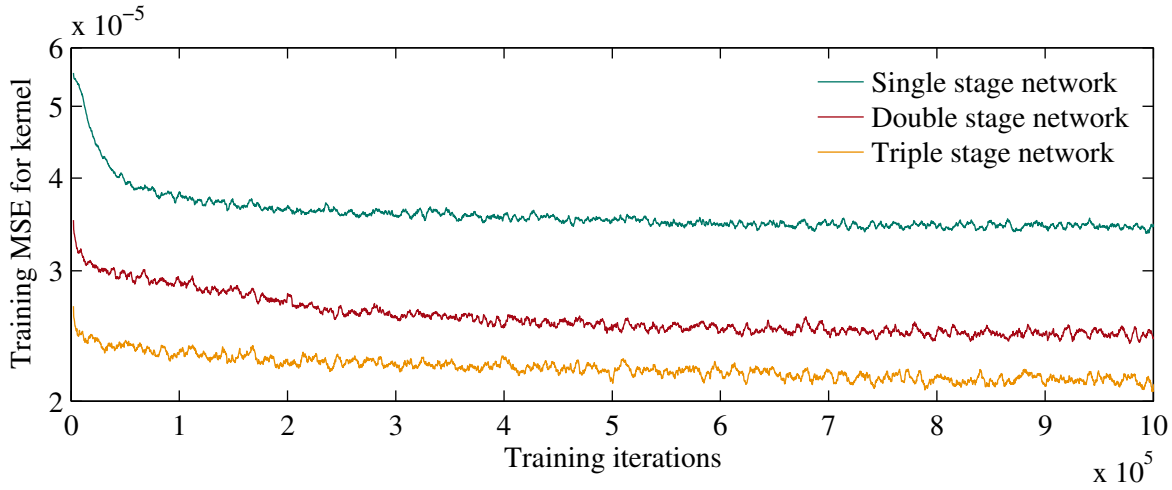


Figure 5.3.: Deeper networks are better at kernel prediction. One stage for kernel prediction consists of a convolutional layer, two hidden layers and a kernel estimation module.

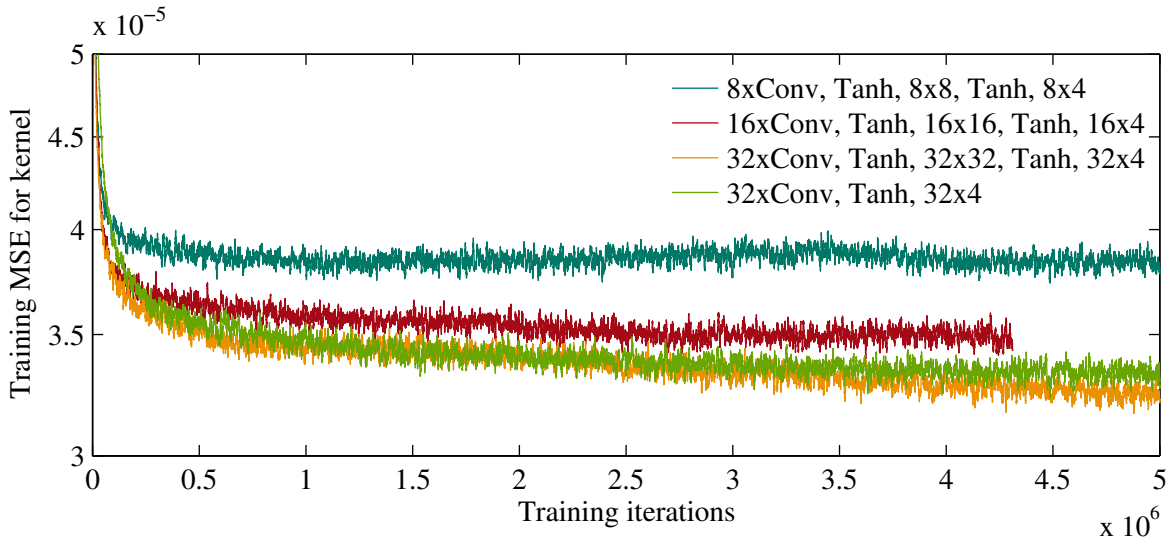


Figure 5.4.: The performance of the network for kernel estimation depends on the architecture. More filters in the convolutional layer and more hidden layers are better.

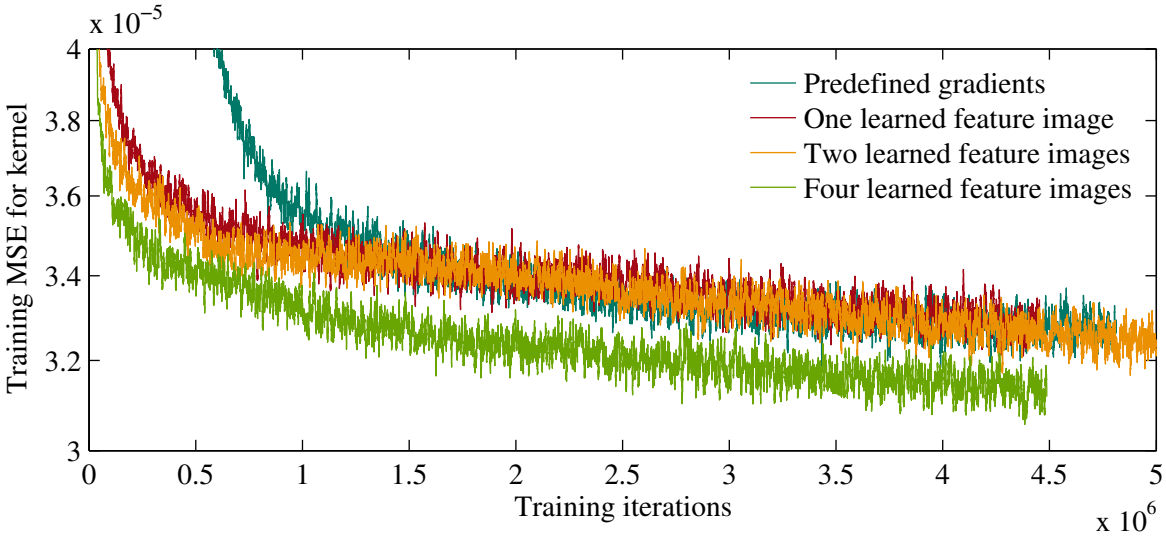


Figure 5.5.: The performance of the network depends on the number of the predicted gradient-like images used in the kernel estimation module. Predefining $\tilde{\mathbf{y}}_i$ to x- and y-gradients and not learning these representations slows down training.

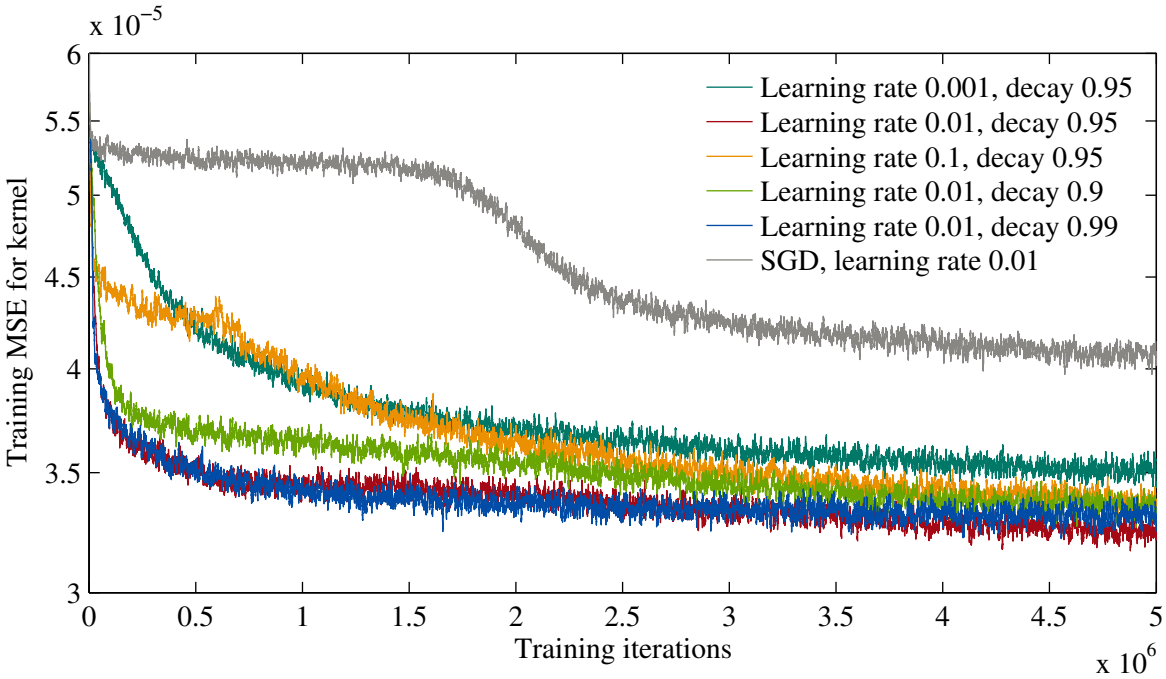


Figure 5.6.: Influence of the parameters of ADADELTA [Zei12] or SGD on the convergence of the network.

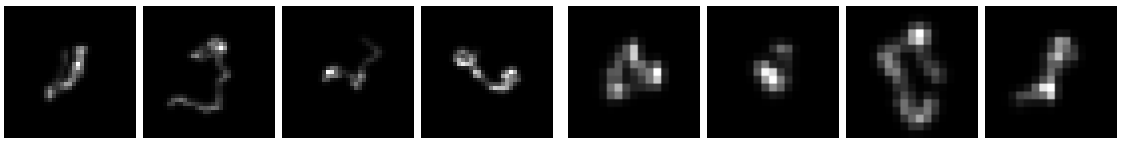


Figure 5.7.: Examples of blurs sampled from a Gaussian process (left: 33 px, right: 17 px).

5.3.3. Training

To train the network, we generate pairs of sharp and blurry images. The sharp image is sampled from a subset of about 1.6 million images of the ImageNet [Den+09] dataset, randomly cropped to a size of 256×256. Next, a blur trajectory is generated by sampling both its x- and y-coordinates separately from a Gaussian Process

$$f_x(t), f_y(t) \sim \mathcal{GP}(0, k(t,t')),\ k(t,t') = \sigma_f^2 \left(1 + \frac{\sqrt{5}|t-t'|}{l} + \frac{5(t-t')^2}{3l^2}\right) \exp\left(-\frac{\sqrt{5}|t-t'|}{l}\right), \tag{5.6}$$

where $k(t,t')$ is a Matérn covariance function with $\nu = 3/2$ [RW06] (samples from a Gaussian Process are distributed according to a multivariate normal distribution with its covariance matrix defined by the covariance function). The length scale l is set to 0.3, the signal standard deviation σ_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. 5.7. 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% 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 1000 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.

For the update of the parameters, convergence proved to be best with ADADELTA [Zei12], a heuristic weighting scheme of parameter updates using gradients and updates from previous training steps, see Eq. (1.27). For the influence of its parameters on the convergence speed see Fig. 5.6. 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. (5.5). In ImageNet these often are photos of objects

Blur size	255×255	800×800	1024×1024	2048×2048
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

Table 5.1.: The method is very fast: runtime in seconds for kernel estimation with varying image size on an Intel i5 in Matlab.

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.

5.4. Implementation

We make the code for both training and testing our method available for download.[4] For training, we use our own C++/CUDA-based neural network toolbox described in Appendix B. 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 5.1. The most expensive calculation is the creation of the multiple hidden representations in the convolutional layer of the NN (in this example: 32).

5.5. 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 17×17, 25×25, 33×33. 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 33×33, 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. 5.3 to 5.4.

5.5.1. Image content specific training

A number of recent works [HY12; Sun+13; Wan+13] have pointed out the shortcoming of state-of-the-art algorithms [CL09; XJ10] 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,

[4] `http://webdav.is.mpg.de/pixel/neural_blind_deconvolution/`

Figure 5.8.: Typical example images of *valley* (top row) and *blackboard* (bottom row) categories from ImageNet used for content specific training.

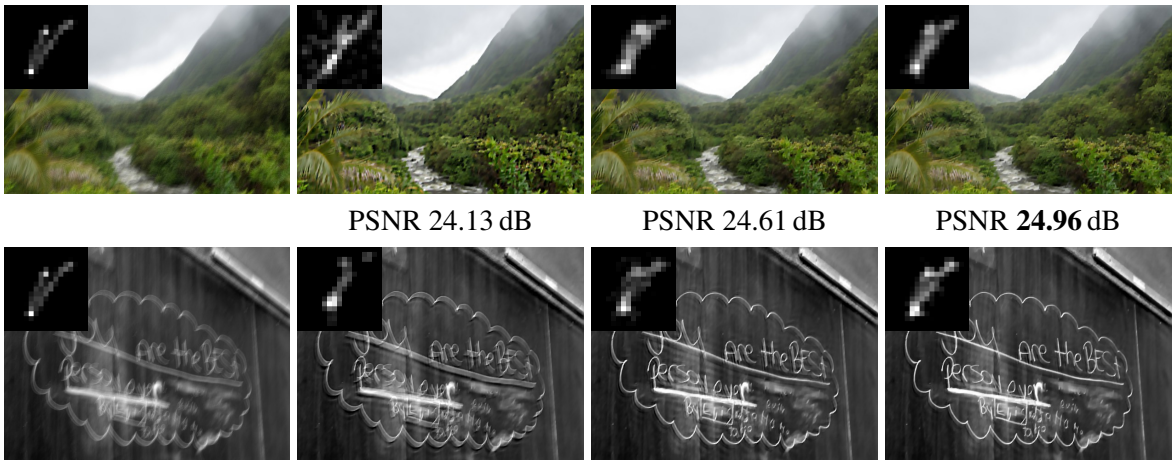
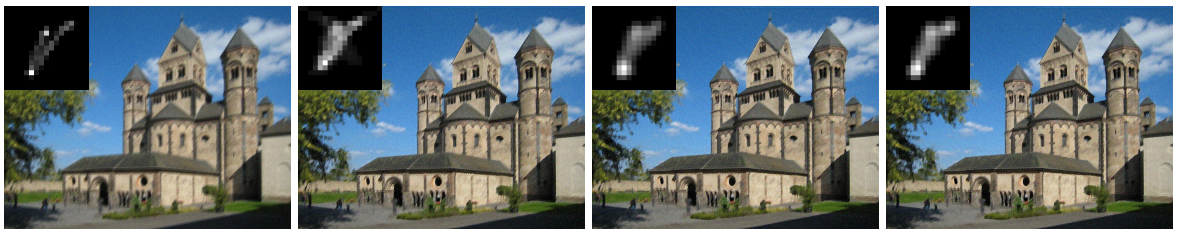


Figure 5.9.: Comparison of deblurring results for NNs that have been trained with image examples from the entire ImageNet dataset (content *agnostic*) and from particular subsets (content *specific*), i.e., image categories *valley* (top) and *blackboard* (bottom). We also show results of the state-of-the-art method [XZJ13].

Valley image (Flickr ID)	Xu et al. [XZJ13]	*Agnostic* training	*Specific* training	Blackboard img. (Flickr ID)	Xu et al. [XZJ13]	*Agnostic* training	*Specific* training
fCetj6	24.13	24.61	**24.96**	bkkdz3	26.37	23.63	**27.53**
hR7YPb	**28.59**	27.96	28.32	btKo6w	20.71	23.87	**26.32**
i1GWi8	20.70	20.17	**20.91**	dbQBYX	20.88	22.36	**24.25**
nz5BxB	**22.97**	22.28	22.80	f9jSxK	23.63	21.60	**24.41**
nRuiRC	23.32	23.20	**23.44**	fK3V16	**25.32**	23.77	25.22
Average	23.94	23.64	**24.09**	Average	23.38	23.04	**25.55**

Table 5.2.: PSNRs of content *agnostic* vs. content *specific* training. Values in dB. All images were downloaded from Flickr in resolution "medium" and have been blurred with blur kernel 2 from [Lev+09].



Blurry image	PSNR 23.17 dB Zhong et al. [Zho+13]	PSNR 23.28 dB Noise *agnostic* training	PSNR **23.52** dB Noise *specific* training

Figure 5.10.: 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% noise) performs best. We also show the results of a recently proposed deblurring method tailored for increased levels of noise [Zho+13].

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*[5] containing a total of 1395 pictures. In a second experiment, we trained a network on the image category *blackboard*[6] with a total of 1376 pictures. Figure 5.8 shows typical example images from these two classes. Figure 5.9 compares deblurring results of the state-of-the-art algorithm described in [XZJ13] with our approach trained on images sampled from the entire ImageNet dataset, and trained on the aforementioned image categories only. Table 5.2 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 [XZJ13] for the category *valley*. For category *blackboard*, which is more distinct from generic natural images, it surpasses its competitor by a large margin.

5.5.2. 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 [CL09] or Gaussian filtering [XJ10]. However, in a recent work [Zho+13], the authors show that for increased levels of noise current methods fail to yield satisfactory

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

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

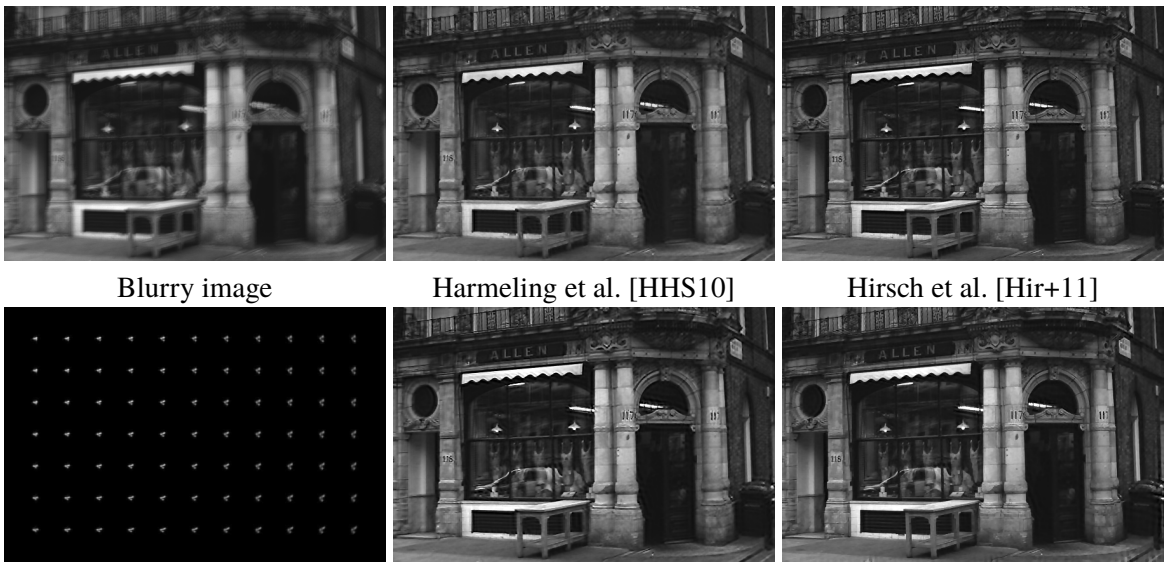
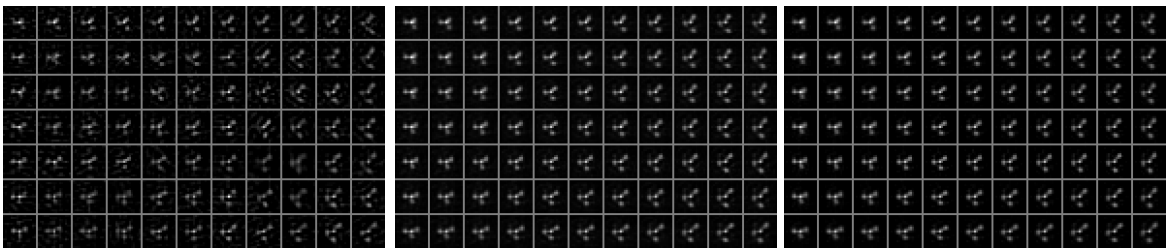


Figure 5.11.: Comparison on *Butcher Shop* example [HHS10] of state-of-the-art deblurring methods for removing non-uniform blur together with our estimated PSF.

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% added during the training phase. Figure 5.9 compares the results for an image taken from [Zho+13] with 5% Gaussian noise for a network trained with 1% and 5% of added Gaussian noise during training, respectively. We also show the result of [Zho+13] and compare PSNR for objective evaluation. All results use the same non-blind deconvolution of [Zho+13]. The noise specific training is most successful, but even the noise agnostic NN outperforms the non-learned method on this example.

5.5.3. Spatially-varying blur

Since the prediction step of our trained NN is independent of the convolution model, we can also use it in conjunction with the recently proposed fast forward model of [Hir+11] to restore images with spatially-varying blur. To this end, we replace the objective function Eq. (5.5) with Eq. (8) of [Hir+11] 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 EFF model of [Hir+10]; in a second step we project the blur kernels onto a motion basis aka [Hir+11], as explained below. Figure 5.11 shows a comparison between recent state-of-the-art algorithms for spatially-varying blur along with our deblurring result that features comparable quality.



Result of direct kernel estimation | Result after basis projection | Result after basis projection and subsequent sparsification

Figure 5.12.: Visualisation of our kernel estimation in the case of spatially-varying blur for the *Butcher Shop* example shown in Fig. 5.11. The left panel shows the kernel estimated with Eq. (5.8), the middle and right panels show the kernel after applying a subsequent projection step to our motion basis, i.e., the result of Eq. (5.9) with η set to 1.0 and 0.0314, respectively.

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

$$\sum_i \|\tilde{X}_i\,\tilde{\mathbf{k}} - \tilde{\mathbf{y}}_i\|^2 + \beta_k \|\tilde{\mathbf{k}}\|^2 \tag{5.7}$$

in our kernel estimation module. Here $\tilde{X}_i$ denotes the EFF matrix of $\tilde{\mathbf{x}}$ (cf. Eq. (1.11)) 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. (5.7) is quadratic in $\tilde{\mathbf{k}}$, we can solve for a local blur $\tilde{\mathbf{k}}^{(r)}$ in a single step

$$\tilde{\mathbf{k}}^{(r)}_{direct} \approx F^{\mathsf{H}} \frac{\sum_i \overline{FC_r\,\mathrm{Diag}(\mathbf{w}^{(r)})\,\tilde{\mathbf{x}}_i} \odot (FC_r\,\mathrm{Diag}(\mathbf{w}^{(r)})\,\tilde{\mathbf{y}}_i)}{\sum_i |FC_r\,\mathrm{Diag}(\mathbf{w}^{(r)})\,\tilde{\mathbf{x}}_i|^2 + \beta_k}. \tag{5.8}$$

Note that Eq. (5.8) is only approximately true and is motivated by Eq. (8) in [Hir+11]. Subsequently, we project the estimated kernel computed by Eq. (5.8) to a motion blur kernel basis. In our experiments we use the same basis as [Hir+11] 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

$$\tilde{\mathbf{k}}^{(r)}_{est} \approx B^{(r)}\; T_\eta \underbrace{\sum_r (B^{(r)})^T \tilde{\mathbf{k}}^{(r)}_{direct}}_{\boldsymbol{\mu}}, \tag{5.9}$$

where again we make use of the notation chosen in [Hir+11], i.e., $B^{(r)}$ denotes the motion blur kernel basis for patch r. Then $\boldsymbol{\mu}$ are the coefficients in the basis of valid motion blurs. T_η denotes a thresholding operator that sets all elements to zero below a certain threshold whereby the threshold is chosen such that only η percent of entries remain non-zero. This thresholding step is motivated by [CL09], who also apply a hard thresholding step to the estimated kernels in order to get rid of spurious artefacts. Figure 5.12 shows the intermediate results of our kernel estimation procedure in the case of spatially-varying blur.

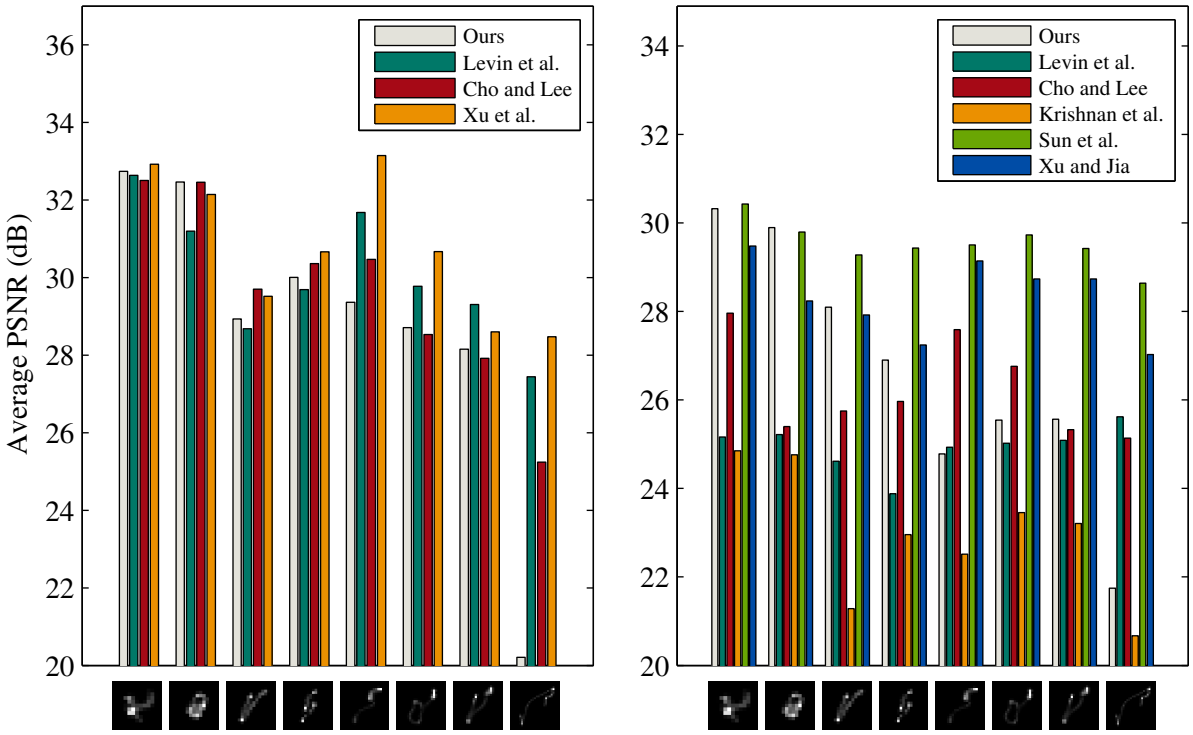


Figure 5.13.: Results of the benchmark dataset of Levin et al. [Lev+09] and the extended benchmark of Sun et al. [Sun+13]. The results are sorted according to blur kernel size. While for kernels up to a size of 25×25 pixels, our approach yields comparable results, it falls short for larger blur kernels. Reasons for the performance drop are discussed in Section 5.6.3.

5.5.4. Comparisons

Benchmark Datasets

We evaluate our method on the standard test sets from [Lev+09; Sun+13]. The four images of [Lev+09] are 255×255 pixels in size and are artificially blurred each with eight different blur kernels and contain 1% additive Gaussian noise. The performance is illustrated in Fig. 5.13 on the left, with blur kernels sorted according to increasing size. We compare with Levin et al. [Lev+11], Cho and Lee [CL09], and Xu et al. [XZJ13]. 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 5.6.3. The second benchmark from [Sun+13] extends this dataset to 80 new images with about one megapixel in size each, using the same blur kernels as in [Lev+09]. Results are shown in Fig. 5.13 on the right. Here we compare with Levin et al. [Lev+11], Cho and Lee [CL09], Krishnan et al. [KTF11], Sun et al. [Sun+13], and Xu and Jia [XJ10], where however [Sun+13] has runtime in the order of hours. Again, we see competitive performance for small blur kernels.

Real-World Images

In Figs. 5.14 and 5.15 we show results of our method on real-world images. Figure 5.14 shows examples for invariant blur, while Fig. 5.15 depicts images with spatially-varying camera shake. In both examples, our approach is able to recover images comparable in quality with the state of the art.

5.6. Discussion

5.6.1. Learned filters

The task of the *Feature Extraction Module* is to emphasize and enhance those image features that contain information about the unknown blur kernel. Figure 5.16 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 17×17 pixels. While 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. 5.18 we visualize 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 Gaussians of varying size. 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 [CL09; XJ10], 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.

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. Figure 5.2 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.

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.

As discussed in Section 5.5, the learned filters depend on the image set that the NN was trained with, i.e., the *feature extraction module* learns image content specific features that are informative about the unknown blur kernel. In Fig. 5.17, we show the learned filters of the experiments in Section 5.5.1 for the *valley* and *blackboard* image category of the ImageNet dataset; they indeed differ from the ones trained on all images. For example, most of the filters learned for valley images are mirror or rotational symmetric, unlike many filters of the generic NN.

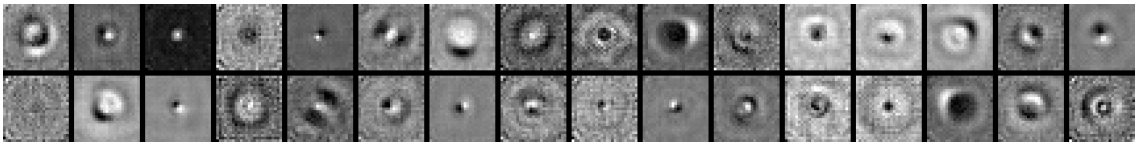
Blurry image | Cho et Lee [CL09] | Our results

Blurry image | Fergus et al. [Fer+06] | Our result

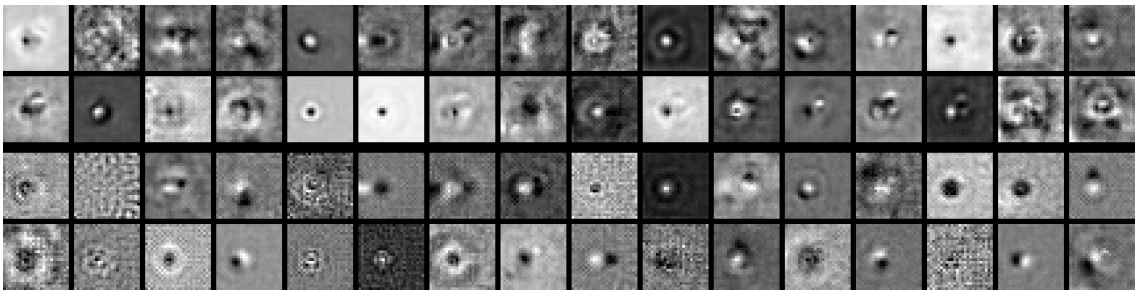
Blurry image | Shan et al. [SJA08] | Our result

Figure 5.14.: Comparison on real-world example images taken from the literature with spatially invariant blur.

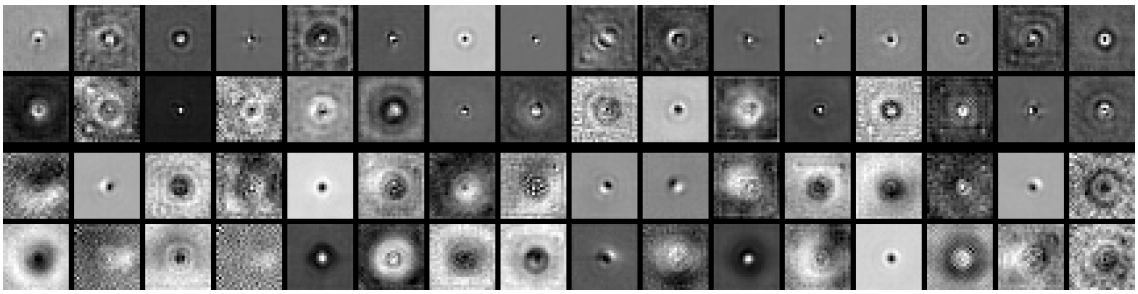
Figure 5.15.: Comparison on real-world example images taken from the literature with spatially-varying blur.



First iteration

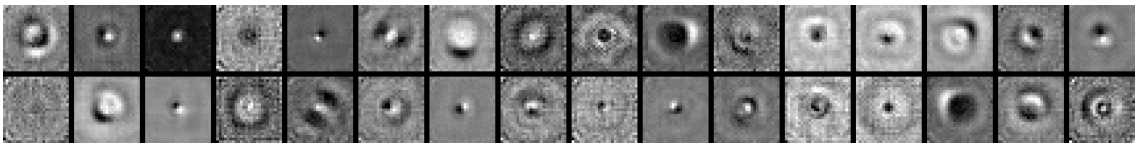


Second iteration

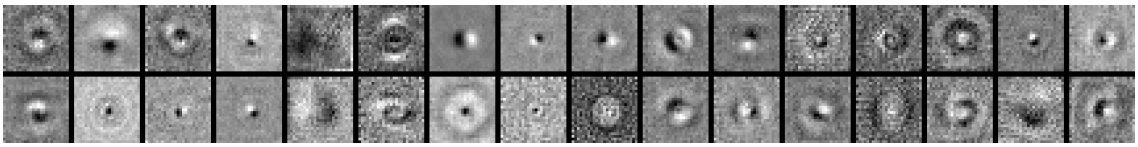


Third iteration

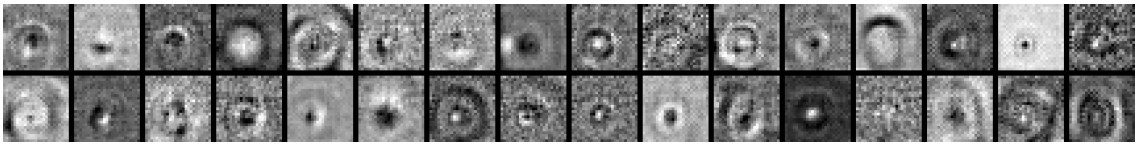
Figure 5.16.: Learned filters of the convolution layer for each of the three iterations within a single scale of a trained NN for kernel size 17×17 pixels. See text for details.



NN trained on all images

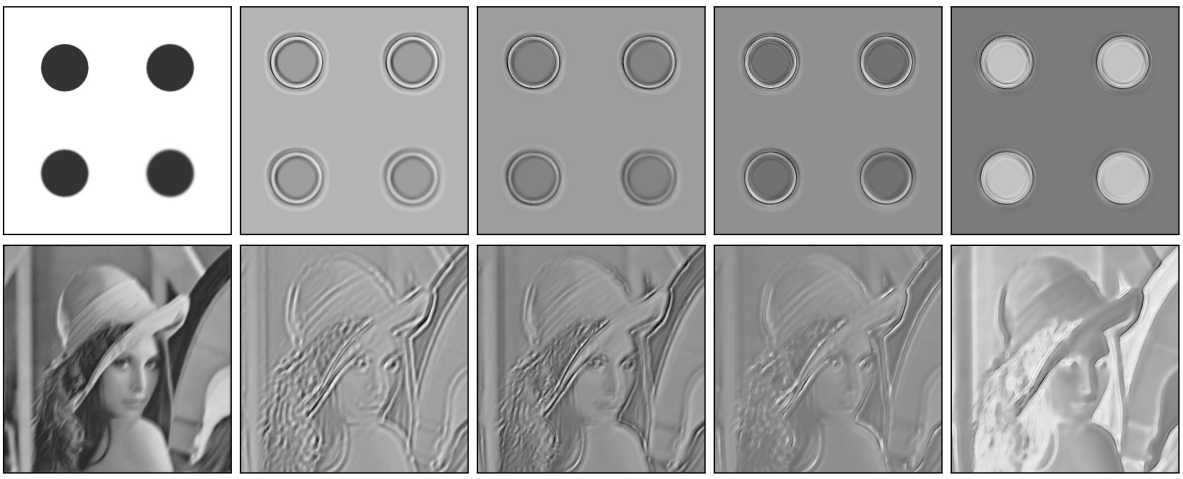


NN trained on valley images



NN trained on blackboard images

Figure 5.17.: Learned filters of the convolution layer for three different types of images.



Blurry image | Learned blurry feature image $\tilde{\mathbf{y}}_1$ | Learned blurry feature image $\tilde{\mathbf{y}}_2$ | Learned latent feature image $\tilde{\mathbf{x}}_1$ | Learned latent feature image $\tilde{\mathbf{x}}_2$

Figure 5.18.: 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 $\tilde{\mathbf{y}}_i$ the different sizes of the Gaussian blurs are clearly visible, the NN replaces them in $\tilde{\mathbf{x}}_i$ with shapes of comparable sharpness.

5.6.2. Dependence on the size of the observed image

As noted by Hu and Yang [HY12], 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 kernel estimation. In Fig. 5.19 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 [XZJ13] exhibits no such trend. The more thorough analysis in Fig. 5.20 confirms this behavior: when evaluating the MSE for three different kernels, averaged over the 52 largest images from [Sun+13], 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.

5.6.3. Limitations

A limitation of our current approach is the performance drop in the case of larger blur kernels. Figure 5.21 shows an example failure case from the benchmark dataset of [Lev+09]. We believe the reason for this is the suboptimal architecture of our multi-scale approach at higher

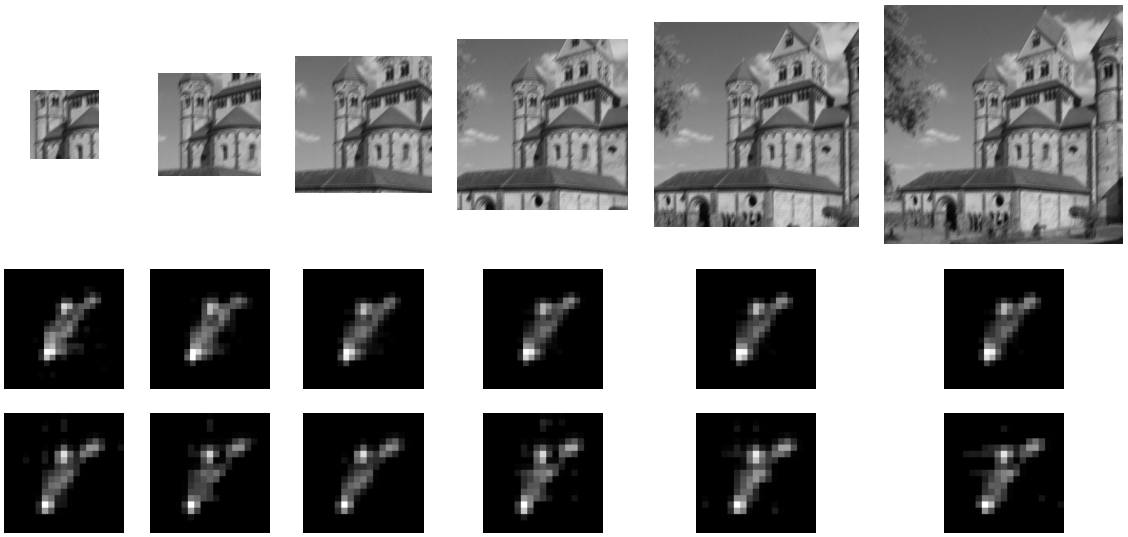


Figure 5.19.: Results for kernel estimation for different sizes of the observed image. *Top row:* Differently sized inputs to the blind deblurring algorithm. *Middle row:* Estimated kernels of our method. Larger inputs lead to better results. *Bottom Row:* Estimated kernels of [XZJ13]. No clear trend is visible.

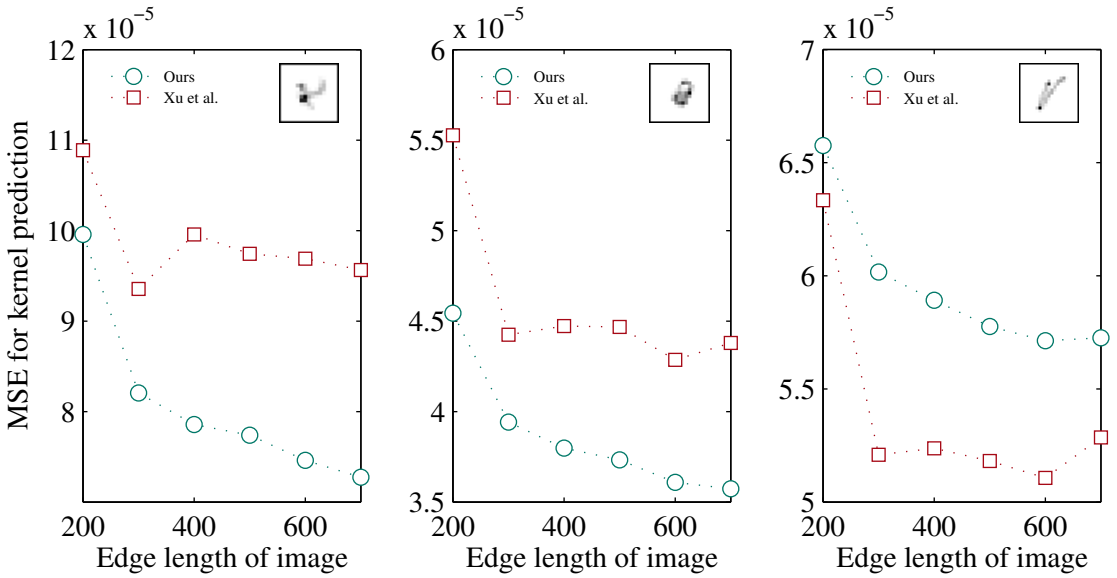
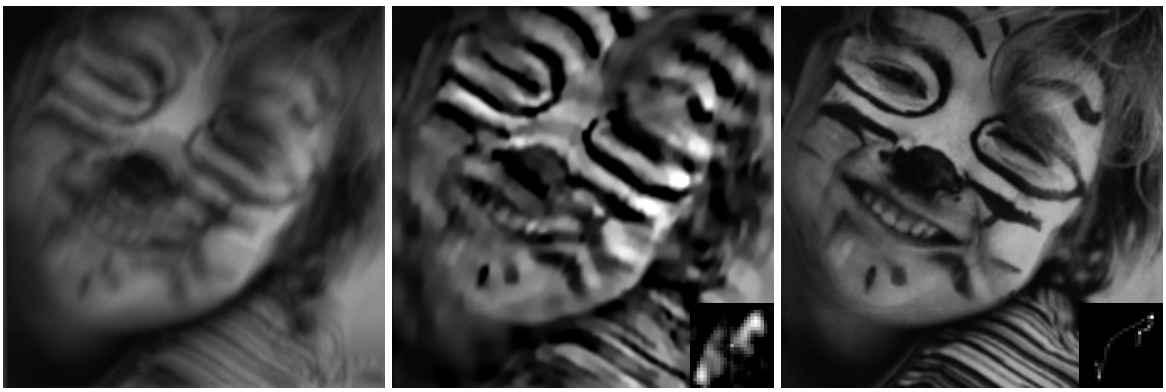


Figure 5.20.: Dependence of the estimated kernel on the size of the observed image. We show the MSE of predictions of three kernels for different sized inputs (cf. Fig. 5.19), averaged over the 52 largest images from [Sun+13]. The MSE of the NN decreases monotonously with image size.



Blurry image | Failure case of our approach | Ground truth

Figure 5.21.: For larger blur kernels our approach falls short in yielding acceptable deblurring results. Here, an example of the benchmark dataset of [Lev+09] with a blur kernel of size 27×27 pixels is shown.

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 using the same architecture at all scales is not optimal.

5.7. 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. To invite research in this direction, we make the code publicly available.

Conclusion and Outlook

Image deconvolution is a long-standing problem, and the key to recover images is to infer lost information by making use of prior knowledge. This thesis has gone beyond the previous state of the art both in non-blind and blind deconvolution, and has improved reconstruction results by including more prior knowledge.

We saw in Chapter 2 that even the simplest lenses consisting of a single glass element produce acceptable photos when corrected in software. The measured optical aberrations were encoded in a spatially-varying PSF and reduced in a MAP estimation with a prior on statistics of image gradients. This formulation also allows us to infer full-color information from a sensor sampling only one color channel per pixel. While most blur-inducing optical aberrations could be corrected, geometrical distortions were not included. It may be beneficial to add them to the forward model and remove them jointly in a single elegant formulation.

In Chapter 3, it turned out that an image affected by optical aberrations reveals the spatially-varying blur so well that additional calibration is not necessary. The physical properties of photographic lenses could be encoded in a linear basis for valid PSFs and thus the dimensionality of the solution space could be reduced. The approach was based on an existing MAP estimation procedure with an heuristic prior on sharp images, and extended to full-color blurs, which are responsible for chromatic aberrations. Since properties of lenses were included as hard constraints on symmetries of the resulting PSF, some manufacturing mistakes were excluded from correction. Instead, a prior distribution on deviations from the symmetries may be more suitable. Further, we did not capitalize on multiple images captured with the same lens. Ideally, our knowledge about the optical aberrations should increase with the size of the photographer's growing photo collection.

In Chapter 4 it was shown that the basic non-blind deconvolution problem with spatially invariant blur profits from better prior knowledge of natural images. Neural networks extracted the required information from a large training set of pairs of artificially generated blurry and sharp images and learned to correct both the noise and the image artifacts remaining after an initial preprocessing step. Larger models were more successful, and we also gained some insights into the learned image correction mechanism. The main downside — that a network must be trained for each blur kernel separately — could possibly be remedied by an approach akin to [Sch+13a].

Finally, Chapter 5 demonstrated that a learning approach is successful even for the more challenging case when the blur is not known a priori. The properties of the blind deconvolution

problem, in particular that the kernel has a limited size and is reproduced in the blur of the observation everywhere, guided the layout of a deep neural network. By iterating between extraction of information locally, and propagating the information globally, the relevant features are learned automatically. A learning algorithm has a competitive edge when allowed to specialize on single categories of image content or blur type, but there is still room for improvement for large kernels. Hopefully, future hand-engineered methods will also profit from insights into the problem as discovered by the neural network.

The proposed methods have been successful in advancing the state of the art of image deconvolution for photography applications. As mentioned above, there are still many possibilities for further improvements. However, it will be especially interesting to see how these approaches can extend to other problem domains. Blur is an issue important to resolve for medical and scientific imaging on all scales, from MRI diagnostics in radiology, to the nanoscopic analysis of new materials, to the search for exoplanets.

APPENDIX A

Mathematical Details

Quotient Layer

As introduced in Eq. (5.4), the quotient layer performs the operation

$$\tilde{\mathbf{k}} = F^{\mathsf{H}} \frac{\sum_i \overline{F\tilde{\mathbf{x}}_i} \odot F\tilde{\mathbf{y}}_i}{\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k} \tag{A.1}$$

to estimate the kernel $\tilde{\mathbf{k}}$ from images $\tilde{\mathbf{x}}_i$ and their blurry counterparts $\tilde{\mathbf{y}}_i$, both predicted by the previous layers of the NN. When we exchange the role of blur and kernel, the deconvolution is similar to Eq. (1.17), the solution of the optimization problem in Eq. (1.16).

The quotient layer also includes a learned regularization parameter β_k. As we have seen in Section 1.4, to train the NN, we need the gradients of the output with respect to its parameters (in this case $\tilde{\mathbf{x}}_i$, $\tilde{\mathbf{y}}_i$ and β_k) in every layer. From these we obtain the gradient steps $\Delta\tilde{\mathbf{x}}_j$, $\Delta\tilde{\mathbf{y}}_k$ and $\Delta\beta_k$ in terms of $\Delta\tilde{\mathbf{k}}$, where $\Delta\tilde{\mathbf{k}}$ is determined by the loss for the current training example, back-propagated through the layers subsequent to the quotient layer.

Derivative with respect to sharp images

To obtain $\Delta\tilde{\mathbf{x}}_j$, we first determine the differential form in numerator layout, which means for vectors $\mathbf{u}$, $\mathbf{v}$ and a matrix M that

$$d\mathbf{u} = \begin{pmatrix} du_1 \\ du_2 \\ \vdots \end{pmatrix}, \quad \frac{d\mathbf{u}}{d\mathbf{v}} = \begin{pmatrix} \frac{du_1}{dv_1} & \frac{du_1}{dv_2} & \cdots \\ \frac{du_2}{dv_1} & \frac{du_2}{dv_2} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \quad \text{and} \quad dM = \begin{pmatrix} dm_{11} & dm_{12} & \cdots \\ dm_{21} & dm_{22} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}. \tag{A.2}$$

Therefore, assuming that only $\tilde{\mathbf{x}}_j$ is variable, with the rules of matrix calculus [PP12] we arrive at

$$\begin{aligned} d\tilde{\mathbf{k}} &= F^{\mathsf{H}} \frac{\overline{F d\tilde{\mathbf{x}}_j} \odot F\tilde{\mathbf{y}}_j}{\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k} - F^{\mathsf{H}} \frac{(\sum_i \overline{F\tilde{\mathbf{x}}_i} \odot F\tilde{\mathbf{y}}_i) \odot d(F\tilde{\mathbf{x}}_j \odot \overline{F\tilde{\mathbf{x}}_j})}{(\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k)^2} = \\ &= \underbrace{F^{\mathsf{H}} \operatorname{Diag}\left(\frac{F\tilde{\mathbf{y}}_j}{\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k}\right) \overline{F}}_{A} d\tilde{\mathbf{x}}_j \\ &\quad - \underbrace{F^{\mathsf{H}} \operatorname{Diag}\left(\frac{(\sum_i \overline{F\tilde{\mathbf{x}}_i} \odot F\tilde{\mathbf{y}}_i) \odot \overline{F\tilde{\mathbf{x}}_j}}{(\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k)^2}\right) F}_{B} d\tilde{\mathbf{x}}_j - \underbrace{F^{\mathsf{H}} \operatorname{Diag}\left(\frac{(\sum_i \overline{F\tilde{\mathbf{x}}_i} \odot F\tilde{\mathbf{y}}_i) \odot F\tilde{\mathbf{x}}_j}{(\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k)^2}\right) \overline{F}}_{C} d\tilde{\mathbf{x}}_j, \end{aligned} \tag{A.3}$$

with the Hadamard product $\mathbf{a} \odot \mathbf{b} = \operatorname{Diag}(\mathbf{a})\mathbf{b}$ and the operation "Diag" that creates a diagonal matrix from a vector. Additionally, $\tilde{\mathbf{x}}_i$ is real, and therefore $\overline{d\tilde{\mathbf{x}}_j} = d\tilde{\mathbf{x}}_j$.

With the differential $d\tilde{\mathbf{k}} = M d\tilde{\mathbf{x}}_j$ (where $M = A + B + C$) in numerator layout, we note that the derivative $\frac{d\tilde{\mathbf{k}}}{d\tilde{\mathbf{x}}_j} = M$ and the gradient step $\Delta\tilde{\mathbf{x}}_j = M^{\mathsf{T}}\Delta\tilde{\mathbf{k}}$. Additionally, we use that $B^{\mathsf{T}}\Delta\tilde{\mathbf{k}}$ is real since it is the inverse Fourier transform of a point-wise product of vectors all with Hermitian symmetry (they are themselves purely real vectors that have been Fourier transformed). Hermitian symmetry for a vector $\mathbf{v}$ of length n means $\mathbf{v}_{n-i+1} = \overline{\mathbf{v}_i}$ for all its components i. Therefore,

$$\begin{aligned} \Delta\tilde{\mathbf{x}}_j &= A^{\mathsf{T}}\Delta\tilde{\mathbf{k}} - B^{\mathsf{T}}\Delta\tilde{\mathbf{k}} - C^{\mathsf{T}}\Delta\tilde{\mathbf{k}} = A^{\mathsf{T}}\Delta\tilde{\mathbf{k}} - \overline{B^{\mathsf{T}}\Delta\tilde{\mathbf{k}}} - C^{\mathsf{T}}\Delta\tilde{\mathbf{k}} = \\ &= F^{\mathsf{H}} \frac{\overline{F\Delta\tilde{\mathbf{k}}} \odot F\tilde{\mathbf{y}}_j}{\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k} \\ &\quad - F^{\mathsf{H}} \frac{\overline{\sum_i \overline{F\tilde{\mathbf{x}}_i} \odot F\tilde{\mathbf{y}}_i} \odot F\tilde{\mathbf{x}}_j \odot F\Delta\tilde{\mathbf{k}}}{(\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k)^2} - F^{\mathsf{H}} \frac{(\sum_i \overline{F\tilde{\mathbf{x}}_i} \odot F\tilde{\mathbf{y}}_i) \odot F\tilde{\mathbf{x}}_j \odot \overline{F\Delta\tilde{\mathbf{k}}}}{(\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k)^2} \\ &= F^{\mathsf{H}} \left(\frac{\overline{F\Delta\tilde{\mathbf{k}}} \odot F\tilde{\mathbf{y}}_j}{\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k} - \frac{2\Re\left(\overline{\sum_i \overline{F\tilde{\mathbf{x}}_i} \odot F\tilde{\mathbf{y}}_i} \odot F\Delta\tilde{\mathbf{k}}\right) \odot F\tilde{\mathbf{x}}_j}{(\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k)^2} \right), \end{aligned} \tag{A.4}$$

since the transpose has no effect on a diagonal matrix. The real part of a vector $\mathbf{v}$ is denoted as $\Re(\mathbf{v}) = \frac{1}{2}(\mathbf{v} + \overline{\mathbf{v}})$.

Derivative with respect to blurry images

The derivation for the gradient $\Delta\tilde{\mathbf{y}}_j$ is similar. We again start with the differential form

$$d\tilde{\mathbf{k}} = F^{\mathsf{H}} \frac{\overline{F\tilde{\mathbf{x}}_j} \odot F d\tilde{\mathbf{y}}_j}{\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k} = \underbrace{F^{\mathsf{H}} \operatorname{Diag}\left(\frac{\overline{F\tilde{\mathbf{x}}_j}}{\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k}\right) F}_{A} d\tilde{\mathbf{y}}_j, \tag{A.5}$$

this time assuming $d\tilde{\mathbf{x}}_j$ and $d\beta_k$ to be zero. Next, we use again that a real vector is unchanged by conjugation, and we obtain

$$\Delta\tilde{\mathbf{y}}_j = A^{\mathsf{T}}\Delta\tilde{\mathbf{k}} = \overline{A^{\mathsf{T}}\Delta\tilde{\mathbf{k}}} = F^{\mathsf{H}}\overline{\mathrm{Diag}\left(\frac{\overline{F\tilde{\mathbf{x}}_j}}{\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k}\right)F\Delta\tilde{\mathbf{k}}} = F^{\mathsf{H}}\left(\frac{F\tilde{\mathbf{x}}_j \odot F\Delta\tilde{\mathbf{k}}}{\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k}\right). \tag{A.6}$$

Derivative with respect to the regularization parameter

Following the previous procedure for the regularization parameter β_k we arrive at

$$d\tilde{\mathbf{k}} = F^{\mathsf{H}} \underbrace{\frac{\sum_i \overline{F\tilde{\mathbf{x}}_i} \odot F\tilde{\mathbf{y}}_i}{\left(\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k\right)^2}}_{A} d\beta_k, \tag{A.7}$$

assuming only β_k to be variable. Then, multiplying $\Delta\tilde{\mathbf{k}}$ by the transpose of $\frac{d\tilde{\mathbf{k}}}{d\tilde{\beta}_k} = A$ we get

$$\Delta\beta_k = A^{\mathsf{T}}\Delta\tilde{\mathbf{k}} = \left(F^{\mathsf{H}}\frac{\sum_i \overline{F\tilde{\mathbf{x}}_i} \odot F\tilde{\mathbf{y}}_i}{\left(\sum_i |F\tilde{\mathbf{x}}_i|^2 + \beta_k\right)^2}\right)^T \Delta\tilde{\mathbf{k}}. \tag{A.8}$$

APPENDIX B

Neural Network Toolbox

In Chapters 4 and 5 we learn to deconvolve with neural networks. To fit the specific needs for the applications in this thesis, we provide a specialized C++ neural network toolbox available for Linux, Windows and OS X as open source software[1] (also to be published on GitHub).

The toolbox is layer-based, and a neural network is represented as an acyclic graph on layers. Layers can either be GPU or CPU based, and data is only transferred between the video card's memory and the system's memory if necessary. While the toolbox is in C++, it provides wrappers both for Python and for Matlab (unlike, e.g., the Python-only Theano neural network software package[2]). Trained networks are saved in HDF5 format to be accessible also from external tools (for Chapter 4, however, an early toolbox version without HDF5 support was employed).

Because of the long training times involved, optimization for speed is crucial. To minimize development time and guarantee operations to be efficient also on the next generation of computing hardware, we "stand on the shoulders of giants" and express most computations such that they can be performed by heavily optimized linear algebra libraries. Specifically, these are libraries implementing the Basic Linear Algebra Subprograms (BLAS), for example OpenBLAS[3] on CPU or cuBLAS on GPU[4]. The C++ code for operations not included in BLAS, for example tanh, is written such that a suitable compiler can automatically make use of a CPU's single instruction, multiple data (SIMD) capabilities (e.g., auto-vectorization in GCC[5]). For the MLPs used in Chapter 4, our toolbox is within 3% of the performance of version 0.6 of the highly optimized Theano library, according to our benchmarks.

Layers currently implemented in the toolbox are shown in Fig. B.1. They include typical neural network layers, e.g., for linear transformations (*LinearLayer*) and nonlinearities (*TanhLayer*

[1] `http://webdav.is.mpg.de/pixel/files/nnet_toolbox.zip`
[2] `https://github.com/Theano/Theano`
[3] `http://xianyi.github.com/OpenBLAS`
[4] `https://developer.nvidia.com/cuBLAS`
[5] `http://gcc.gnu.org/projects/tree-ssa/vectorization.html`

or *RectLayer*). Some layers are specific to the application in Chapter 5, e.g., the *DeconvLayer* used in the *kernel estimation module*. For a detailed description, see the documentation included with the toolbox. A standard NN consists of an *InputLayer*, several transformation layers, and a *Criterion* layer which specifies the cost criterion to be minimized. The class *NeuralNet* facilitates creating such a network, as the following code example for learning the exclusive disjunction (XOR) problem demonstrates, using the Matlab wrapper of the toolbox:

```
% Creates neural network with two input neurons.
nnet = NeuralNet(2);
% Linear transformation from two input to two hidden neurons.
nnet.add_layer(LinearLayerGpu(2, 2));
% Adds nonlinear tanh transformation.
nnet.add_layer(TanhLayerGpu());
% Linear transformation from two hidden neurons to one output neuron.
nnet.add_layer(LinearLayerGpu(2, 1));
% Uses mean square error criterion as cost function.
nnet.add_layer(MseCriterionGpu());

% Sets input and target to the four states of the XOR problem.
nnet.set_input([-1 1 -1 1; -1 1 1 -1]);
nnet.set_target([-1 -1 1 1]);
% Sets learning rate of SGD training.
nnet.set_update(0.1);
% Performs 250 training iterations on permutations of the training data.
nnet.train_permuted(250);

% Applies trained network to a single example.
nnet.set_input([-1; -1]);
nnet.forward();
output = nnet.get_output();
```

After training, this simple MLP with two neurons in its hidden layer should have learned the XOR function, i.e., the variable `output` in the last line should be near -1.

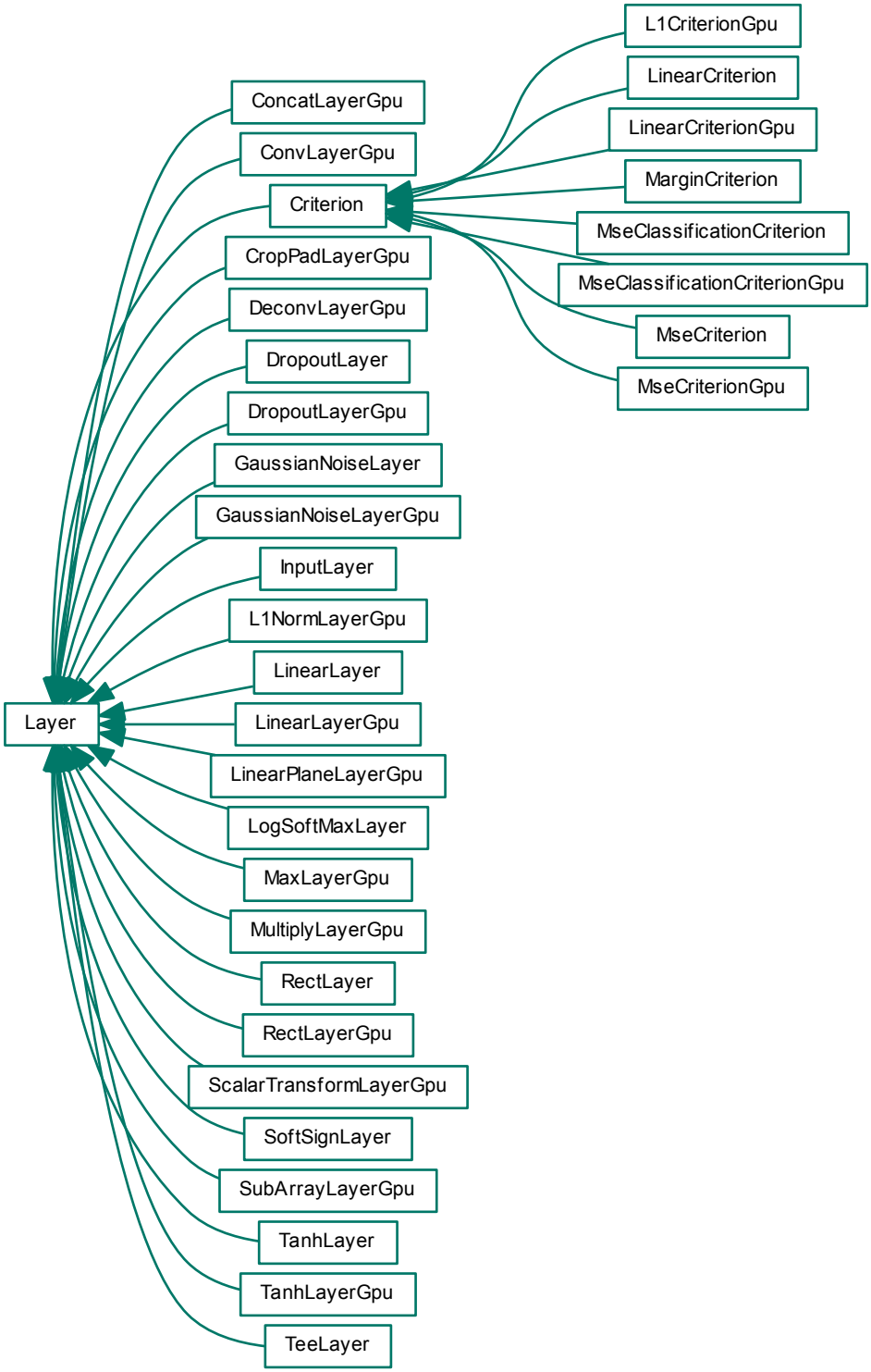


Figure B.1.: Class hierarchy of layers of the toolbox. A neural network is an acyclic graph on different layers, typically starting with a single *InputLayer* and ending with an implementation of a *Criterion* layer.

Acronyms

AWG	additive white Gaussian	54
BLAS	Basic Linear Algebra Subprograms	107
CFA	color filter array	22
DNN	deep neural network	76
EFF	efficient filter flow	9
FFT	fast Fourier transform	8
MAP	maximum a posteriori	iii, 2
MLP	multilayer perceptron	17
MMSE	minimum mean square error	11
NN	artificial neural network	16
PDF	probability density function	12
PSF	point spread function	8
PSNR	peak signal-to-noise ratio	29, 30
RMS	root mean square	18
SGD	stochastic gradient descent	17
SIMD	single instruction, multiple data	107
SNR	signal-to-noise ratio	10
VB	variational Bayes	15
XOR	exclusive disjunction	108

Nomenclature

Matrices and vectors

α, σ, t	scalar values
$\mathbf{v}$, $\mathbf{x}$, $\mathbf{k}$	column vectors
M, X, K	matrices
v_i	the i-th element of a vector $\mathbf{v}$
M_{ij}	the ij-th element of a matrix M
$\overline{\mathbf{v}}$, $\overline{M}$	complex conjugate of a vector or matrix
$\mathbf{v}^\mathsf{T}$, M^T	vector or matrix transpose
$\mathbf{v}^\mathsf{H}$, M^H	conjugate transpose of a vector or matrix
M^{-1}	matrix inverse
Diag $\mathbf{v}$	diagonal matrix with vector $\mathbf{v}$ on diagonal
F	discrete Fourier matrix
$\mathbf{1}$	vector of all ones

Other symbols

$\mathbb{Z}$	set of integers
$*$	convolution operator
$\diamond$	fast forward model operator
$\odot$	element-wise multiplication
$\sum$	summation operator
$\int$	integral operator
$f[n]$	value of a function f at n, where f is defined on $\mathbb{Z}$

Bibliography

[Aiz+06] I. Aizenberg, D. Paliy, C. Moraga, and J. Astola. "Blur identification using neural network for image restoration". In: *Computational Intelligence, Theory and Applications*. Vol. 38. Springer, 2006, pp. 441–455. DOI: 10.1007/3-540-34783-6_45.

[AL08] D. Alleysson and B. C. de Lavarène. "Frequency selection demosaicking: A review and a look ahead". In: *Proc. SPIE Visual Communications and Image Processing*. Vol. 6822. 2008, p. 68221M. DOI: 10.1117/12.771656.

[Ben12] Y. Bengio. "Practical recommendations for gradient-based training of deep architectures". In: *Neural Networks: Tricks of the Trade*. Lecture Notes in Computer Science. Springer, 2012, pp. 437–478. DOI: 10.1007/978-3-642-35289-8_26.

[BG91] L. Bottou and P. Gallinari. "A Framework for the Cooperation of Learning Algorithms". In: *Advances Neural Information Processing Systems*. 1991, pp. 781–788.

[Bot91] L. Bottou. "Stochastic Gradient Learning in Neural Networks". In: *Proceedings Neuro-Nîmes 91*. Vol. 91. 8. 1991.

[BSH12a] H. C. Burger, C. J. Schuler, and S. Harmeling. "Image denoising with multi-layer perceptrons, part 1: comparison with existing algorithms and with bounds". In: *ArXiv e-prints* (2012). arXiv: 1211.1544 [cs.CV].

[BSH12b] H. C. Burger, C. J. Schuler, and S. Harmeling. "Image denoising with multi-layer perceptrons, part 2: training trade-offs and analysis of their mechanisms". In: *ArXiv e-prints* (2012). arXiv: 1211.1552 [cs.CV].

[BSH12c] H. C. Burger, C. J. Schuler, and S. Harmeling. "Image denoising: Can plain Neural Networks compete with BM3D?" In: *IEEE Conf. Computer Vision and Pattern Recognition*. 2012, pp. 2392–2399. DOI: 10.1109/cvpr.2012.6247952.

[BV04] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004. DOI: 10.1017/cbo9780511804441.

[BW92] T. E. Boult and G. Wolberg. "Correcting chromatic aberrations using image warping". In: *Proc. IEEE Conf. Computer Vision and Pattern Recognition*. 1992, pp. 684–687. DOI: 10.1109/cvpr.1992.223201.

[BW99] M. Born and E. Wolf. *Principles of optics: electromagnetic theory of propagation, interference and diffraction of light*. Cambridge University Press, 1999. ISBN: 978-0521642224.

[CD91] C. M. Cho and H. S. Don. "Blur identification and image restoration using a multilayer neural network". In: *IEEE Int. Joint Conf. Neural Networks*. 1991, pp. 2558–2563. DOI: 10.1109/ijcnn.1991.170774.

[Cir+10] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber. "Deep, big, simple neural nets for handwritten digit recognition". In: *Neural Computation* 22.12 (2010), pp. 3207–3220. DOI: 10.1162/neco_a_00052.

[CKS09] S. W. Chung, B. K. Kim, and W. J. Song. "Detecting and eliminating chromatic aberration in digital images". In: *IEEE Int. Conf. Image Processing*. 2009, pp. 3905–3908. DOI: 10.1109/icip.2009.5413971.

[CL09] S. Cho and S. Lee. "Fast motion deblurring". In: *ACM Trans. Graphics* 28.5 (2009), p. 145. DOI: 10.1145/1618452.1618491.

[CWL11] S. Cho, J. Wang, and S. Lee. "Handling Outliers in Non-blind Image Deconvolution". In: *IEEE Int. Conf. Computer Vision*. 2011. DOI: 10.1109/iccv.2011.6126280.

[Dab+07] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. "Image denoising by sparse 3-D transform-domain collaborative filtering". In: *IEEE Trans. Image Processing* 16.8 (2007), pp. 2080–2095. DOI: 10.1109/tip.2007.901238.

[Dab+08] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. "Image restoration by sparse 3D transform-domain collaborative filtering". In: *Proc. SPIE Image Processing: Algorithms and Systems VI*. Vol. 6812. 2008, p. 681207. DOI: 10.1117/12.766355.

[Den+09] J. Deng et al. "ImageNet: A large-scale hierarchical image database". In: *IEEE Conf. Computer Vision and Pattern Recognition*. 2009. DOI: 10.1109/cvpr.2009.5206848.

[DeR+03] D. De Ridder et al. "Nonlinear image processing using artificial neural networks". In: *Advances Imaging and Electron Physics* 126 (2003), pp. 352–450.

[DHS11] J. Duchi, E. Hazan, and Y. Singer. "Adaptive subgradient methods for online learning and stochastic optimization". In: *Journal Machine Learning Research* 12 (2011), pp. 2121–2159.

[DKE12] A. Danielyan, V. Katkovnik, and K. Egiazarian. "BM3D frames and variational image deblurring". In: *IEEE Trans. Image Processing* 21.4 (2012), pp. 1715–1728. DOI: 10.1109/tip.2011.2176954.

[EA06] M. Elad and M. Aharon. "Image denoising via sparse and redundant representations over learned dictionaries". In: *IEEE Trans. on Image Processing* 15.12 (2006), pp. 3736–3745. DOI: 10.1109/tip.2006.881969.

[ECB10] D. Erhan, A. Courville, and Y. Bengio. *Understanding Representations Learned in Deep Architectures*. Tech. rep. 1355. Université de Montréal/DIRO, 2010.

[ERH02] M. Egmont-Petersen, D. de Ridder, and H. Handels. "Image processing with neural networks—a review". In: *Pattern recognition* 35.10 (2002), pp. 2279–2301. DOI: 10.1016/S0031-3203(01)00178-9.

[FEM06] S. Farsiu, M. Elad, and P. Milanfar. "Multiframe demosaicing and super-resolution of color images". In: *IEEE Trans. Image Processing* 15.1 (2006), pp. 141–159. DOI: 10.1109/tip.2005.860336.

[Fer+06] R. Fergus et al. "Removing camera shake from a single photograph". In: *ACM Trans. Graphics* 25.3 (2006), pp. 787–794. DOI: 10.1145/1141911.1141956.

[Gif08] S. Gifford. "Astronomical Coma Image Restoration Through The Use of Localized Deconvolution". In: *Soc. Astronomical Sciences Annu. Symp.* Vol. 27. 2008, p. 141.

[GMP08] J. A. Guerrero-Colón, L. Mancera, and J. Portilla. "Image restoration using space-variant Gaussian scale mixtures in overcomplete pyramids". In: *IEEE Trans. Image Processing* 17.1 (2008), pp. 27–41. DOI: 10.1109/tip.2007.911473.

[Gun+05] B. K. Gunturk et al. "Demosaicking: color filter array interpolation". In: *IEEE Signal Processing Mag.* 22.1 (2005), pp. 44–54. DOI: 10.1109/MSP.2005.1407714.

[Gup+10] A. Gupta et al. "Single Image Deblurring Using Motion Density Functions". In: *IEEE Europ. Conf. Computer Vision*. 2010.

[GW02] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice Hall, 2002. ISBN: 0-201-18075-8.

[Hei+13] F. Heide et al. "High-quality computational imaging through simple lenses". In: *ACM Trans. Graphics* 32.5 (2013), p. 149. DOI: 10.1145/2516971.2516974.

[HHS10] S. Harmeling, M. Hirsch, and B. Schölkopf. "Space-Variant Single-Image Blind Deconvolution for Removing Camera Shake." In: *Advances Neural Information Processing Systems*. 2010, pp. 829–837.

[HHY10] Z. Hu, J.-B. Huang, and M.-H. Yang. "Single image deblurring with adaptive dictionary learning". In: *IEEE Int. Conf. Image Processing*. 2010, pp. 1169–1172. DOI: 10.1109/icip.2010.5651892.

[Hir+10] M. Hirsch, S. Sra, B. Schölkopf, and S. Harmeling. "Efficient filter flow for space-variant multiframe blind deconvolution." In: *IEEE Conf. Computer Vision and Pattern Recognition*. 2010, pp. 607–614. DOI: 10.1109/cvpr.2010.5540158.

[Hir+11] M. Hirsch, C. J. Schuler, S. Harmeling, and B. Schölkopf. "Fast removal of non-uniform camera shake". In: *IEEE Int. Conf. Computer Vision*. 2011, pp. 463–470. DOI: 10.1109/iccv.2011.6126276.

[Hir12] M. Hirsch. "Blind Deconvolution in Scientific Imaging & Computational Photography". PhD thesis. Universität Tübingen, 2012.

[HY12] Z. Hu and M.-H. Yang. "Good Regions to Deblur". In: *Computer Vision – ECCV 2012*. Lecture Notes in Computer Science. Springer, 2012, pp. 59–72. DOI: 10.1007/978-3-642-33715-4_5.

[JNR12] J. Jancsary, S. Nowozin, and C. Rother. "Loss-Specific Training of Non-Parametric Image Restoration Models: A New State of the Art". In: *Computer Vision – ECCV 2012*. Lecture Notes in Computer Science. Springer, 2012, pp. 112–125. DOI: 10.1007/978-3-642-33786-4_9.

[Jos+10] N. Joshi, S. B. Kang, C. L. Zitnick, and R. Szeliski. "Image deblurring using inertial measurement sensors". In: *ACM Trans. Graphics* 29.4 (2010), p. 30. DOI: 10.1145/1833351.1778767.

[JS08] V. Jain and H. S. Seung. "Natural image denoising with convolutional networks". In: *Advances Neural Information Processing Systems* (2008), pp. 769–776.

[JSK08] N. Joshi, R. Szeliski, and D. J. Kriegman. "PSF estimation using sharp edge prediction". In: *IEEE Conf. Computer Vision and Pattern Recognition*. 2008. DOI: 10.1109/CVPR.2008.4587834.

[Kee+11] E. Kee, S. Paris, S. Chen, and J. Wang. "Modeling and Removing Spatially-Varying Optical Blur". In: *IEEE Int. Conf. Computational Photography*. 2011. DOI: 10.1109/iccphot.2011.5753120.

[KF09] D. Krishnan and R. Fergus. "Fast Image Deconvolution using Hyper-Laplacian Priors". In: *Advances Neural Information Processing Systems*. 2009, pp. 1033–1041.

[KL05] V. Kaufmann and R. Ladstädter. "Elimination of color fringes in digital photographs caused by lateral chromatic aberration". In: *Proc. XXth Int. CIPA Symp.* Vol. 26. 2005, pp. 403–408.

[KN11] C. Khare and K. K. Nagwanshi. "Image Restoration in Neural Network Domain using Back Propagation Network Approach". In: *Int. J. Computer Information Systems* 2.5 (2011), pp. 25–31.

[KTF11] D. Krishnan, T. Tay, and R. Fergus. "Blind deconvolution using a normalized sparsity measure". In: *IEEE Conf. Computer Vision and Pattern Recognition*. 2011. DOI: 10.1109/cvpr.2011.5995521.

[Le+12] Q. Le et al. "Building high-level features using large scale unsupervised learning". In: *Int. Conf. Machine Learning*. 2012.

[LeC+98a] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proc. IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[LeC+98b] Y. LeCun, L. Bottou, G. Orr, and K. Müller. "Efficient BackProp". In: *Neural Networks: Tricks of the Trade*. Lecture Notes in Computer Science. Springer, 1998. DOI: 10.1007/3-540-49430-8_2.

[Lev+07] A. Levin, R. Fergus, F. Durand, and W. T. Freeman. "Image and depth from a conventional camera with a coded aperture". In: *ACM Trans. Graphics* 26.3 (2007), p. 70. DOI: 10.1145/1276377.1276464.

[Lev+09] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. "Understanding and evaluating blind deconvolution algorithms". In: *IEEE Conf. Computer Vision and Pattern Recognition*. 2009, pp. 1964–1971. DOI: 10.1109/cvpr.2009.5206815.

[Lev+11] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. "Efficient marginal likelihood optimization in blind deconvolution". In: *IEEE Conf. Computer Vision and Pattern Recognition*. 2011. DOI: 10.1109/cvpr.2011.5995308.

[LGZ08] X. Lia, B. Gunturkb, and L. Zhangc. "Image demosaicing: A systematic survey". In: *Proc. SPIE Conf. Visual Communications and Image Processing*. 2008. DOI: 10.1117/12.766768.

[Mar] E. Martinec. *Noise, Dynamic Range and Bit Depth in Digital SLRs*. Visited on 05/22/14. URL: http://theory.uchicago.edu/~ejm/pix/20d/tests/noise/.

[MF11] M. Mäkitalo and A. Foi. "Optimal inversion of the Anscombe transformation in low-count Poisson image denoising". In: *IEEE Trans. Image Processing* 20.1 (2011), pp. 99–109. DOI: 10.1109/tip.2010.2056693.

[MP43] W. S. McCulloch and W. Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *Bulletin Mathematical Biophysics* 5.4 (1943), pp. 115–133. DOI: 10.1007/bf02478259.

[Mur12] K. P. Murphy. *Machine Learning: a Probabilistic Perspective*. MIT Press, 2012. ISBN: 978-0262018029.

[MW07] J. Mallon and P. F. Whelan. "Calibration and removal of lateral chromatic aberration in images". In: *Pattern Recognition Lett.* 28.1 (2007), pp. 125–135. DOI: 10.1016/j.patrec.2006.06.013.

[OR90] S. Osher and L. I. Rudin. "Feature-Oriented Image Enhancement Using Shock Filters". In: *SIAM J. Numerical Analysis* 27.4 (1990), pp. 919–940. DOI: 10.1137/0727053.

[Pal+07] D. Paliy et al. "Spatially adaptive color filter array interpolation for noiseless and noisy data". In: *Int. J. Imaging Systems and Technology* 17.3 (2007), pp. 105–122. DOI: 10.1002/ima.20109.

[Por+03] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli. "Image denoising using scale mixtures of Gaussians in the wavelet domain". In: *IEEE Trans. Image Processing* 12.11 (2003), pp. 1338–1351. DOI: 10.1109/tip.2003.818640.

[PP12] K. B. Petersen and M. S. Pedersen. *The matrix cookbook*. 2012.

[Ram+02] R. Ramanath, W. E. Snyder, G. L. Bilbro, and W. A. Sander III. "Demosaicking methods for Bayer color arrays". In: *J. Electronic Imaging* 11.3 (2002), pp. 306–315. DOI: 10.1117/1.1484495.

[RB09] S. Roth and M. J. Black. "Fields of experts". In: *Int. J. Computer Vision* 82.2 (2009), pp. 205–229. DOI: 10.1007/s11263-008-0197-6.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536. DOI: 10.1038/323533a0.

[Rob54] E. A. Robinson. "Predictive decomposition of time series with applications to seismic exploration". PhD thesis. Massachusetts Institute of Technology, 1954.

[RW06] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. 2006. ISBN: 978-0262182539.

[SA94] S. Shah and J. K. Aggarwal. "A simple calibration procedure for fish-eye (high distortion) lens camera". In: *Proc. IEEE Conf. Robotics and Automation*. 1994, pp. 3422–3427. DOI: 10.1109/robot.1994.351044.

[SA96] E. P. Simoncelli and E. H. Adelson. "Noise removal via Bayesian wavelet coring". In: *Proc. IEEE Int. Conf. Image Processing*. Vol. 1. 1996, pp. 379–382. DOI: 10.1109/icip.1996.559512.

[Sch+11] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. "Non-stationary correction of optical aberrations". In: *IEEE Int. Conf. Computer Vision*. 2011. DOI: 10.1109/iccv.2011.6126301.

[Sch+12] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. "Blind Correction of Optical Aberrations". In: *Computer Vision – ECCV 2012*. Lecture Notes in Computer Science. Springer, 2012, pp. 187–200. DOI: 10.1007/978-3-642-33712-3_14.

[Sch+13a] U. Schmidt et al. "Discriminative Non-blind Deblurring". In: *IEEE Conf. Computer Vision and Pattern Recognition*. 2013, pp. 604–611. DOI: 10.1109/cvpr.2013.84.

[Sch+13b] C. J. Schuler, H. C. Burger, S. Harmeling, and B. Schölkopf. "A Machine Learning Approach for Non-blind Image Deconvolution". In: *IEEE Conf. Computer Vision and Pattern Recognition*. 2013, pp. 1067–1074. DOI: 10.1109/cvpr.2013.142.

[Sch+14] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. "Learning to Deblur". In: *ArXiv e-prints* (2014). arXiv: 1406.7444 [cs.CV].

[Sch14] J. Schmidhuber. "Deep Learning in Neural Networks: An Overview". In: *ArXiv e-prints* (2014). arXiv: 1404.7828.

[SF94] R. J. Steriti and M. A. Fiddy. "Blind deconvolution of images by use of neural networks". In: *Optics letters* 19.8 (1994), pp. 575–577. DOI: 10.1364/ol.19.000575.

[SJA08] Q. Shan, J. Jia, and A. Agarwala. "High-quality motion deblurring from a single image". In: *ACM Trans. Graphics* 27.3 (2008), p. 73. DOI: 10.1145/1360612.1360672.

[SL11] P. Sermanet and Y. LeCun. "Traffic sign recognition with multi-scale convolutional networks". In: *IEEE Int. Joint Conf. Neural Networks*. 2011, pp. 2809–2813. DOI: 10.1109/ijcnn.2011.6033589.

[SSR11] U. Schmidt, K. Schelten, and S. Roth. "Bayesian deblurring with integrated noise estimation". In: *IEEE Conf. Computer Vision and Pattern Recognition*. 2011, pp. 2625–2632. DOI: 10.1109/cvpr.2011.5995653.

[ST09] F. Soulez and E. Thiébaut. "Joint deconvolution and demosaicing". In: *IEEE Int. Conf. Image Processing*. 2009, pp. 145–148. DOI: 10.1109/ICIP.2009.5414151.

[Ste02] G. Stein. "Lens distortion calibration using point correspondences". In: *Proc. IEEE Conf. Computer Vision and Pattern Recognition*. 2002, pp. 602–608. DOI: 10.1109/cvpr.1997.609387.

[Sun+13] L. Sun, S. Cho, J. Wang, and J. Hays. "Edge-based blur kernel estimation using patch priors". In: *IEEE Int. Conf. Computational Photography*. 2013. DOI: 10.1109/iccphot.2013.6528301.

[TOM96] J. E. Tansley, M. J. Oldfield, and D. J. MacKay. "Neural network image deconvolution". In: *Maximum Entropy and Bayesian Methods*. Fundamental Theories of Physics. Springer, 1996, pp. 319–325. DOI: 10.1007/978-94-015-8729-7_25.

[Vin+10] P. Vincent et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". In: *J. Machine Learning Research* 11 (2010), pp. 3371–3408.

[Wal90] M. M. Waldrop. "Hubble: The Case of the Single-Point Failure: The technical glitch in Hubble's mirror has been found; what remains to be identified are the managerial problems that led to it". In: *Science* 249.4970 (1990), pp. 735–736. DOI: 10.1126/science.249.4970.735.

[Wan+13] C. Wang et al. "Nonedge-specific adaptive scheme for highly robust blind motion deblurring of natural images". In: *IEEE Trans. Image Processing* 22.3 (2013), pp. 884–897. DOI: 10.1109/tip.2012.2219548.

[Wer74] P. Werbos. "Beyond regression: New tools for prediction and analysis in the behavioral sciences". PhD thesis. Harvard University, 1974.

[Why+10] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce. "Non-uniform Deblurring for Shaken Images". In: *IEEE Conf. Computer Vision and Pattern Recognition*. 2010, pp. 491–498. DOI: 10.1109/cvpr.2010.5540175.

[WM14] Y. Wang and J. Morel. "Can a single image denoising neural network handle all levels of Gaussian noise?" In: *IEEE Signal Processing Letters* 21.9 (2014), pp. 1150–1153. DOI: 10.1109/LSP.2014.2314613.

[WZ13] D. Wipf and H. Zhang. "Revisiting Bayesian Blind Deconvolution". In: *ArXiv e-prints* (2013). arXiv: 1305.2362 [cs.CV].

[XJ10] L. Xu and J. Jia. "Two-Phase Kernel Estimation for Robust Motion Deblurring". In: *Computer Vision – ECCV 2010*. Lecture Notes in Computer Science. Springer, 2010, pp. 157–170. DOI: 10.1007/978-3-642-15549-9_12.

[XXC12] J. Xie, L. Xu, and E. Chen. “Image Denoising and Inpainting with Deep Neural Networks”. In: *Advances Neural Information Processing Systems* (2012), pp. 350–358.

[XZJ13] L. Xu, S. Zheng, and J. Jia. “Unnatural l0 sparse representation for natural image deblurring”. In: *IEEE Conf. Computer Vision and Pattern Recognition*. 2013, pp. 1107–1114. DOI: 10.1109/cvpr.2013.147.

[Zei+10] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. “Deconvolutional networks”. In: *IEEE Conf. Computer Vision and Pattern Recognition*. 2010, pp. 2528–2535. DOI: 10.1109/cvpr.2010.5539957.

[Zei12] M. D. Zeiler. “ADADELTA: An Adaptive Learning Rate Method”. In: *ArXiv e-prints* (2012). arXiv: 1212.5701 [cs.LG].

[Zho+13] L. Zhong et al. “Handling Noise in Single Image Deblurring Using Directional Filters”. In: *IEEE Conf. Computer Vision and Pattern Recognition*. 2013, pp. 612–619. DOI: 10.1109/cvpr.2013.85.

[ZTF11] M. D. Zeiler, G. W. Taylor, and R. Fergus. “Adaptive deconvolutional networks for mid and high level feature learning”. In: *IEEE Int. Conf. Computer Vision*. 2011, pp. 2018–2025. DOI: 10.1109/iccv.2011.6126474.

[ZW11] D. Zoran and Y. Weiss. “From learning models of natural image patches to whole image restoration”. In: *IEEE Int. Conf. Computer Vision*. 2011, pp. 479–486. DOI: 10.1109/iccv.2011.6126278.

Contributions

Except otherwise explicitly stated, all mathematical derivations, algorithmic implementations and experimental evaluations were performed by the author of this thesis. Experimental evaluations of cited methods have either been produced by the authors of the respective method or with implementations of the method provided by the authors.

The images shown in Chapter 1 in Fig. 1.1 were created by Michael Hirsch and Stefan Harmeling. Michael Hirsch also provided his implementations of [CL09; XZJ13] that were used in generating the results shown in Fig. 1.7 and he helped with the experiments for Fig. 1.3.

The content of Chapter 2 is based on a manuscript created in cooperation with Michael Hirsch, Stefan Harmeling and Bernhard Schölkopf [Sch+11], who also provided the idea for this chapter. Michael Hirsch assisted with the experimental setup used in the measurement of the spatially-varying PSF and helped to build the 120 mm lens and take the photos shown as examples. Furthermore, he provided a Matlab implementation of the EFF framework and helped to derive Eq. (2.6).

The content of Chapter 3 is based on a manuscript created in cooperation with Michael Hirsch, Stefan Harmeling and Bernhard Schölkopf [Sch+12]. Michael Hirsch also helped with the Python implementation of the EFF framework and the derivations of Eqs. (3.8) and (3.10).

The content of Chapter 4 is based on a manuscript created in cooperation with Christopher Burger, Stefan Harmeling and Bernhard Schölkopf [Sch+13b]. Christopher Burger also assisted with the creation of Figs. 4.10 to 4.13 and provided valuable advice on the training of neural networks.

The content of Chapter 5 is based on a manuscript created in cooperation with Michael Hirsch, Stefan Harmeling and Bernhard Schölkopf [Sch+14]. Philipp Hennig assisted with the GP-based sampling of the blur trajectory. Michael Hirsch created Fig. 5.8 and Fig. 5.12 and helped with the derivation of Eq. (5.8).

Raffi Enficiaud contributed to the development of the neural network toolbox in Appendix B by improving the CMake build scripts and ensuring compatibility with OS X and Windows-based systems.

Acknowledgements

First and foremost, I would like to thank Prof. Dr. Schölkopf for enabling me to do this work and supporting me in every way. I would also like to thank Dr. Stefan Harmeling for his supervision and his constant help with all aspects of my research. I am indebted to Prof. Dr. Hendrik Lensch for agreeing to become my "Doktorvater" and supporting this dissertation.

Special thanks go to Dr. Christopher Burger and Dr. Michael Hirsch for incredibly valuable discussions that defined the direction of my research. Likewise, great thanks to my co-authors Prof. Dr. Schölkopf, Dr. Stefan Harmeling, Dr. Christopher Burger and Dr. Michael Hirsch for working together with me and for letting me reuse material from joint publications for this thesis. Thanks also to Martin Kiefel and Dr. Philipp Hennig for working with me on topics not directly related to this thesis, but contributing to my development as a scientist. Thanks to everyone who helped proof-read this document, including the people above, and, additionally Matthias Moeferdt and Michael Schober. I would also like to thank my office mates Alexander Loktyushin and Rolf Köhler for interesting discussions and making my days in the office more enjoyable.

It is also important to mention Prof. Dr. Kurt Busch here who supervised my diploma thesis and helped to lay the foundations for my scientific career. Additionally, thanks go to everyone at Google who contributed to further my software engineering skills and my knowledge about machine learning.

My gratitude goes to everyone who worked in the background to support my thesis, especially Sebastian Stark, Sabrina Rehbaum and Andrea Odermatt, who assisted me with all of my numerous requests. Sebastian Stark never got tired of making yet another modification to our computing resources, allowing me to carry out my countless experiments.

Finally, thanks to my family and my parents who have supported me in all of my decisions throughout my life.

Thank you!