Erich Christoph Kobler, BSc

Learning variational models for blind image deconvolution

MASTER'S THESIS

to achieve the university degree of
Diplom-Ingenieur

Master's degree programme
Information and Computer Engineering

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr.techn. Thomas Pock
Institute for Computer Graphics and Vision

Graz, Austria, Sep. 2014

Abstract

Along with noise, image blur is probably the most widespread reason for image degradation. It originates from a vast variety of sources, including atmospheric turbulences, defocus and motion. Nowadays fast and accurate deblurring algorithms become more and more important due to the ubiquitous smartphones. The majority of recent deblurring algorithms first estimate the point spread function, also known as blur kernel, and then perform a non-blind image deblurring. In this work we introduce a novel approach for both non-blind and blind image deblurring, which is motivated by variational models. We follow the idea of Chen et al. 2015 and derive a network structure which is related to minimizing an iteratively adapted energy functional. Moreover, we present a differentiable projection onto the unit simplex based on the Bregman divergence to constrain the blur kernels. The non-blind as well as blind deblurring networks are trained in a discriminative fashion to enhance properties of natural sharp images because recent discriminative reconstruction approaches demonstrated their superiority in terms of quality and runtime. Both deblurring networks are qualitatively evaluated and numerous experiments demonstrate the clear quality boost of the resulting image and blur kernel estimates. Furthermore, in contrast do neural networks, all individual parameters of the proposed networks can be easily interpreted due to the close relation to energy minimization.

Kurzfassung

Verzerrungen und Unschärfen sind neben Rauschen wahrscheinlich die weitverbreitesten Gründe für degradierte Bilder. Unter anderem können Bewegungen einer Kamera oder von Objekten während der Bildaufnahme, aber auch Atmosphärische Turbulenzen und falsche Fokussierung zu verwischten und verzerrten Bildern führen. Die Anzahl der so degenerierten Bilder wird heutzutage zunehmend größer aufgrund der allgegenwärtigen Smartphones und der darin enthaltenen Kameras. Deshalb werden schnelle und qualitativ hochwertige Algorithmen zum Entzerren von Bildern zunehmend bedeutender. Der Großteil der kürzlich veröffentlichten Methoden schätzt zuerst die Punktverteilungsfunktion, auch bekannt als Verzerrungskernel, und entzerrt dann das verwischte Bild mit dem geschätzten Kernel. In dieser Arbeit stellen wir einen neuen Ansatz zum Entzerren von Bildern vor, der sowohl für bekannte Verzerrungskernel als auch unbekannte angewendet werden kann. Unser Entzerrungs-Algorithmus ist angelehnt an Variationsmethoden und motiviert durch die Ideen von Chen et al. 2015. Die so abgeleitete Netzwerkstruktur kann als Minimierung eines iterativ adaptierten Energiefunktionales interpretiert werden. Zusätzlich präsentieren wir eine differenzierbare Projektion auf das Einheitssimplex basierend auf Bregman Distanzen, um die möglichen Verzerrungskernel zu beschränken. Die beiden vorgestellten Netzwerke werden diskriminativ trainiert, um der Methode Eigenschaften von natürlichen scharfen Bildern beizubringen und aufgrund der herausragenden Qualität und Laufzeit neuster diskriminativer Bild-Rekonstruktionsmethoden. Die Netzwerke werden betreffend der resultierenden Bildqualität evaluiert und zahlreiche Experimente zeigen die herausragende Qualität der Bild- und Verzerrungskernel-Schätzungen. Des Weiteren hilft unsere Methode ein besseres Verständnis für gute Entzerrungs-Algorithmen aufzubauen. Denn im Gegensatz zu Neuronalen Netzwerken kann jeder einzelne Parameter eines Netzwerkes einfach interpretiert werden aufgrund der engen Verbindung zur Energieminimierung.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

The text document uploaded to TUGRAZonline is identical to the presented master's thesis dissertation.

Place	Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Ort	Datum	Unterschrift

Acknowledgments

First, I would like to express my sincere gratitude and appreciation to my supervisor Prof. Dr. Thomas Pock. He maneuvered me kindly around turbulences when I got stuck somewhere, had always an open ear to my questions and shared a lot of time with me developing this approach. The profound discussions helped me to gain a lot of knowledge in the field of optimization problems, image regularization and beyond. For all this and the comprehensive answers to all my questions a special thank you.

At this point I would also like to express by gratitude to my colleagues at the Institute for computer vision and graphics, who supported me. In particular I would like to thank Yunjin Chen for sharing his ideas and Patrick Knöbelreiter, Teresa Klatzer and Kerstin Hammernik for the substantive discussions.

Moreover, I would like to give a special thanks to all my brothers and sisters and foremost my father, who all supported me unconditionally when ever I needed someone during my entire life. Also my study colleagues and friends deserve an extraordinary thank you for supporting me and sweetening my student life.

Finally, I would like to thank Silke for her patience, support and love.

Contents

List of Figures

List of Tables

1 Introduction

Contents

Recording images has never been easier than today. An image can be acquired in a fraction of a second almost everywhere on earth and easily shared using todays smartphones. Due to this freedom the number of images taken around the world exploded. A major part of these images suffer from bad quality due to noise and motion degradation. While the noise is inherently generated by the acquisition sensor, image blurring denotes the degradation of images by virtue of the image formation process itself. Possible sources for blurring are e.g. object motion, camera shake and so forth.

One strategy to overcome this problem is to avoid the degradation in the first place. To do so, one should either take pictures of quasi static scenes just under suitable lightning conditions, or use high-end cameras which generate low noise, have a sensor stabilization to reduce camera shake and very sensitive sensors for fast imaging. However, in this thesis we do not propose any hardware modifications but rather concentrate on the restoration of degraded images.

1.1 Image restoration

The field of image restoration deals with reconstructing a sharp and clean image of an observed scene by solely considering a degraded image. The most fundamental restoration problems are image denoising and deblurring, since both sources of degradation frequently occur in today's imaging systems. In this work we focus on deconvolution, motivated by the recent progress in image denoising [13].

1.2 Image deblurring

When taking an image of a scene the resulting observation is often blurry. This blur either originates from camera intrinsic elements such as out of focus blur, or extrinsic factors such as relative movement between camera and scene. These different blur sources can be further categorized as shown in the next section.

1.2.1 Blur types

To better understand image deblurring, a deeper knowledge of the different reasons for image blurring is necessary.

1.2.1.1 Motion blur

Generally speaking, motion blur is induced by a relative movement between a camera and a scene during exposure. It is usually caused by either long exposure time or fast motion. Depending on the moving component we can further distinguish between object motion blur and camera shake blur. Object motion blur occurs when taking a picture of a dynamic scene with a fixed camera, e.g. see Figure 1.1a, whereas, camera shake blur is generated by a movement of the camera, as depicted in Figure 1.1b. When looking at the images in Figure 1.1 one can see that the blurring is not necessarily constant throughout an image. For instance, in Figure 1.1a only the moving tram is smeared, also in Figure 1.1c just the background is blurred. Furthermore, the blurring in the background in Figure 1.1c is spatially variant due to the rotational movement. Parts of the image which are closer to the rotation axis are not as much blurred as those further away. This circumstance is especially visible when comparing the region where the hammock is mounted with the upper border. The smearing of the tree leaves is much larger than those of the string mounted around the tree. Consequently, the blurring in images of typical scenes is spatially variant.

Also the combination of object motion and camera shake blur appears quite often in modern photography. Figure 1.1d provides an example. Clearly, the camera was moved during exposure time, however, it also seems as if the women moved as well. In this case finding the blur and the non-blurred image at all positions in the image is rather hard since the motion can be arbitrary complex.

1.2.1.2 Out of focus blur

Nowadays imaging systems are built up on lenses. By adjusting these lenses photographers can focus exactly on one distance for each shot. However, the sharpness of objects decreases steadily before and behind this focal depth. In photography the distance between the nearest and farthest object appearing acceptable sharp in an image is called the Depth of Field (DoF). In other words, within the *DoF* the unsharpness of objects is not visible. It depends amongst others on the object-camera distance, focal length of the lens and iris aperture.

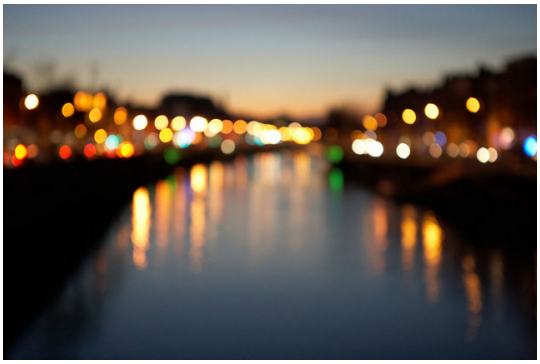
(a)

(b)

(c)

(d)

Figure 1.1: Motion blur examples: (a) static scene with a moving object; (b) scene where the camera follows the linear motion of a cyclist; (c) rotational movement of a camera looking at a static scene: (d) diffuse motion under bad lightning conditions. All images are taken from Flickr[1].



(a)

(b)

Figure 1.2: Out of focus blur examples: (a) incorrect focus setting; (b) artistic usage of limited *DoF*. Both images are taken from Flickr[1].

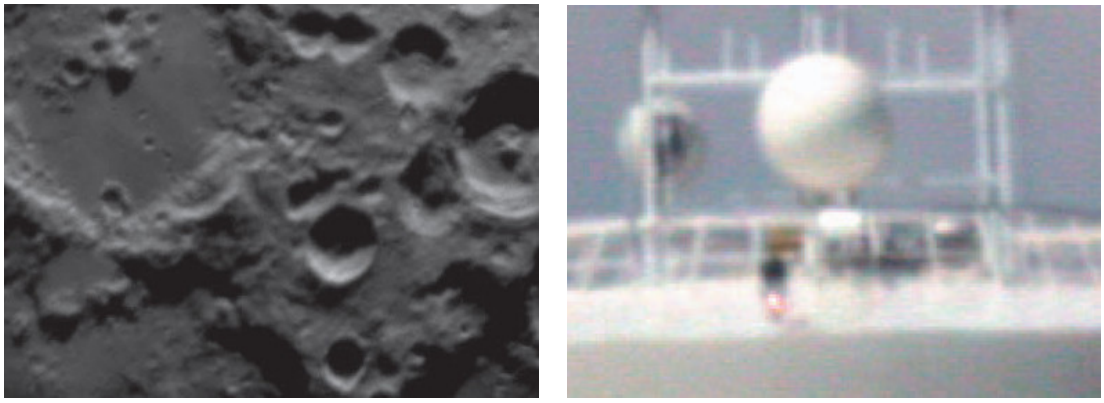
[1] `http://www.flickr.com/`, Accessed: 2015-07-14

Figure 1.2a shows one picture taken with a wrong focus setting. In this image the whole scene is outside the *DoF*, hence the image is entirely blurred. However, this effect is often used in art to highlight the foreground, as Figure 1.2b demonstrates. The jars are apparently sharp, whereas, the background is badly blurred. Even the edge of the table is blurry.

The human visual system behaves in the same manner. When looking at an object close to ones eyes, the background appears blurry. Furthermore, if one holds his/her hand close in front of his/her eyes and focuses on the background, the edges of the fingers are blurred as well. This also holds if just one eye is used. This behavior evolves naturally, since todays imaging systems operate in a similar way as the human eye.

1.2.1.3 Atmospheric turbulence blur

Blurring due to atmospheric turbulences is caused by the nonuniform index of refraction when light waves emit through our atmosphere. As pointed out by Roggemann [35] the atmosphere is built up of a multitude of randomly distributed regions of uniform index of refraction, called turbulent eddies. At each eddy a light wave traveling through the atmosphere is refracted differently, which manifests in a blurry observation of the light source. This phenomenon is always present when taking images via long distances, for example at sea, areal/satellite photography and astronomy. Both images in Figure 1.3



(a) **(b)**

Figure 1.3: Atmospheric turbulence blur examples: (a) blurred observation of the moon surface; (b) atmospheric blurring due to naval long distance imaging. Images are taken from [48].

have blurry edges due to the random refraction of the light while traveling through the atmosphere. The blurring caused by this effect is likely to be nonuniform by virtue of the random distribution of the turbulent eddy's index of refraction.

1.2.2 Uniform vs nonuniform blur

The above blur examples pointed out some important details. Not only can the shape of a blur be arbitrary, but also it is likely that it changes within an image, which we call a nonuniform blur. However, a uniform blur is defined such that it does not change across the entire observed image. Small camera shake when imaging a distant scene can be well approximated by this assumption. Nevertheless, large objection motions or camera shake strongly violet it.

Despite these limitations, we consider uniform blur in the rest of this work, since we first have to solve this problem properly before we can try to tackle the nonuniform case.

1.2.3 Blurring as convolution operations

A typical mathematical formulation to describe the uniform blurring process is by means of a convolution operation. In the continuous setting the two dimensional continuous convolution is defined by

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u(x-\chi, y-\xi)a(\chi,\xi)d\chi d\xi + n(x,y), \tag{1.1}$$

where f is the blurred observed image, u the true sharp image, a the blur function (often called Point Spread Function (PSF)) and n the noise generated by the image acquisition process. Clearly, this formulation smears the intensity values of a hypothetic scene picture, denoted by the sharp image u, according to a in oder to generate the blurry observation f. For further considerations we assume that the noise n is zero-mean Gaussian white noise. Note, also other noise types may occur, e.g. , Poisson noise or impulsive noise.

Since we are dealing with digital, discrete images, we have to formulate the blur process also in the discrete setting. The discrete formulation of Equation (1.1) is

$$f(i,j) = \sum_{h=-H/2}^{H/2}\sum_{w=-W/2}^{W/2} u(i-h, j-w)a(h,w) + n(i,j). \tag{1.2}$$

Again, f is the blurry observation, u the sharp image and $a \in \mathbb{R}^{H\times W}$ the blur kernel, being the discrete version of the *PSF*. Within this formulation the finitely sized image f is constructed from the also not infinitely large image u. Therefore, assumptions about image pixels outside of u have to be made due to the indexing $u(i-h, j-w)$, which may result in negative indexes or ones being larger than the actual size of u. The treatment of the indexes outside the domain of a digital image is called border handling. Many different approaches exists and the most popular are valid, zero-padded, symmetric or circular border handling. In this work we consider the valid and zero-padded boundary handling, see Appendix B. However, quite often also circular boundaries are used due to their theoretical properties. The associated discrete, circular convolution formulation is

defined by

$$f(i,j) = \sum_{h=-H/2}^{H/2} \sum_{w=-W/2}^{W/2} u(i-h \bmod M, j-w \bmod N)a(h,w) + n(i,j), \tag{1.3}$$

where 'mod' defines the modulo operation and the image u is of size $M \times N$. For the sake of simplicity, we use the short vector notation

$$f = a * u + n, \tag{1.4}$$

to express the valid convolution defined in Appendix B and

$$f = u \overset{M\times N}{*} a + n, \tag{1.5}$$

if the circular convolution is used. To sum up, convolution operations can be used to mathematically describe the uniform blurring process in the continuous and discrete setting elegantly. Thus, we can treat the deblurring problem as a deconvolution problem.

1.2.4 Blind and non-blind image deconvolution

Deconvolution problems can be divided into two fundamentally different problems:

non-blind: Given a noisy observation f *and* the (probably estimated) *PSF* a, find the unknown sharp image u.

blind: Given just a noisy observation f, find the unknown *PSF* a *and* the sharp image u.

At first glance the non-blind image deconvolution problem seems to be simpler than the blind one. Indeed, it is easier to solve, however in practice both inverse problems are hard to solve due to their ill-posedness. Well-posed problems in the sense of Hadamard satisfy the existence and uniqueness of their solution and furthermore have solutions that continuously depend on the data. In contrast, an ill-posed problem has non-unique solutions or non-existing ones for every possible data or does not continuously depend on the data. In the following sections we outline the ill-posed nature of the deconvolution problems.

1.2.4.1 Non-blind image deconvolution

In the continuous setting Bertero and Boccacci [4] provide a detailed analysis why the image deconvolution problem is ill-posed. To sum up, despite the uniqueness of the solution, the existence and continuous dependency on the data is not satisfied. Fortunately, Bertero and Boccacci [4] also pointed out that ill-posed problems can be turned into well-posed ones by discretization. However, the solutions of these problems are usually unacceptable from a physical point of view due to their corruption by noise. The following experiment illustrates this circumstance.

Let us denote by

$$\mathcal{F}(u)(k,l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} u(m,n) \exp\left(-2\pi i \left(\frac{mk}{M} + \frac{nl}{N}\right)\right) \tag{1.6}$$

the circular Discrete Fourier Transform (DFT) of an image u of size $M \times N$ and its inverse by

$$u(m,n) = \mathcal{F}^{-1}(\mathcal{F}(u))(m,n) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} \mathcal{F}(u)(k,l) \exp\left(2\pi i \left(\frac{mk}{M} + \frac{nl}{N}\right)\right). \tag{1.7}$$

A well known property of the Fourier transform is the convolution property, defined as

$$\mathcal{F}(a \overset{M\times N}{*} u) = \mathcal{F}(a)\mathcal{F}(u), \tag{1.8}$$

where periodic boundary conditions of the discrete convolution are required. Imagine, for some reason we have acquired a noise free image called $\hat{f}$, however it is still blurred. Due to the uniform blur constraint, the formation process of $\hat{f}$ can be specified by

$$\hat{f} = a \overset{M\times N}{*} u. \tag{1.9}$$

If we apply the *DFT* on this equation and use its convolution property we end up with

$$\mathcal{F}(\hat{f}) = \mathcal{F}(a)\mathcal{F}(u). \tag{1.10}$$

After rearranging this equation and computing the inverse *DFT*, we get a closed form solution for the non-blind deconvolution problem

$$u(\hat{f}, a) = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(\hat{f})}{\mathcal{F}(a)}\right), \tag{1.11}$$

which is also known as inverse filtering. Figure 1.4a depicts the noiseless observation $\hat{f}$ and Figure 1.4b the reconstructed image using Equation (1.11). As we can see the reconstruction is perfect, yielding the desired true sharp image. However, if we perform the same reconstruction with the noisy observation f instead of $\hat{f}$, the resulting image, shown in Figure 1.4d is definitely not physically plausible. Although the difference between f (Fig. 1.4c) and $\hat{f}$ (Fig. 1.4a) is minimal, the two reconstructed images are far apart, which violates the continuous dependence of the solution of a well-posed problem on the input data. This issue originates from the division by the Fourier transform of a in Equation (1.11). If the value at a certain frequency is very small and inaccurate, it can severely influence the result of the inverse filtering output due to its global influence. To overcome this problem, the popular Wiener deconvolution added a constant to the

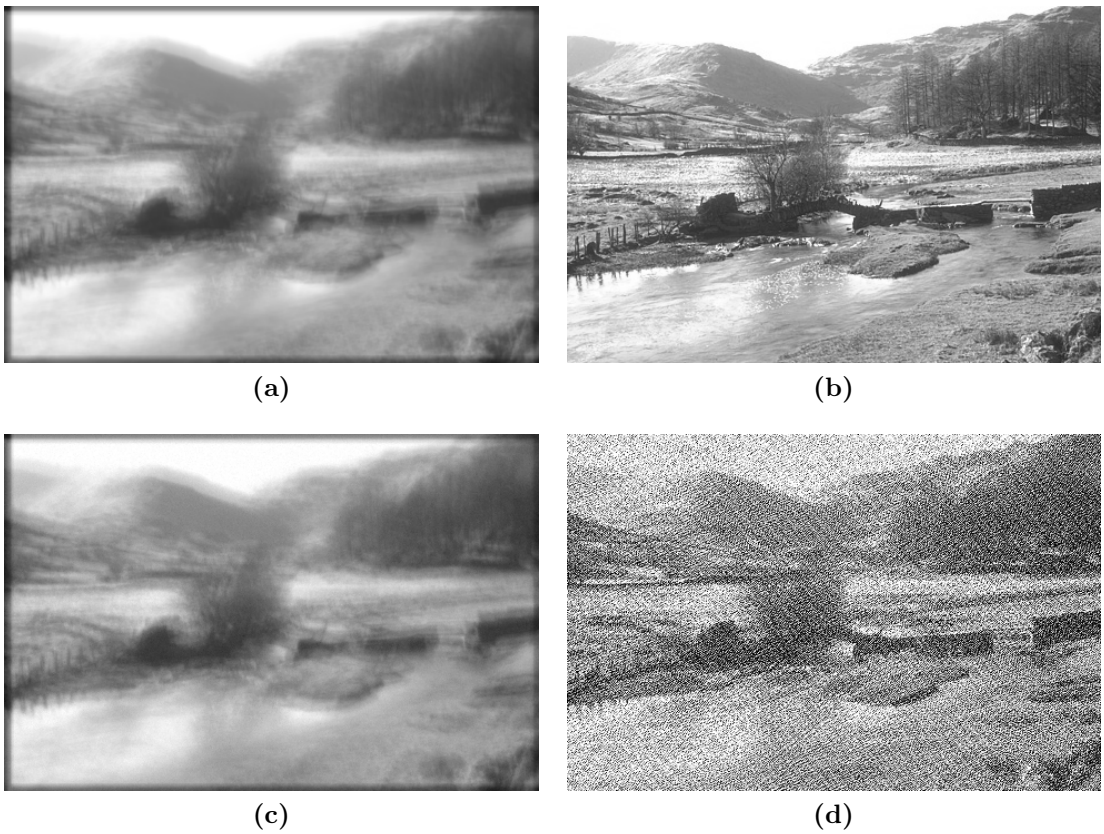


Figure 1.4: Illustration of the ill-posed nature of non-blind image deconvolution: (a) blurry but noise free observation of a scene; (b) reconstructed sharp image using Equation (1.11); (c) same image as in (a) with 1% Gaussian white noise added; (d) reconstruction using Equation (1.11) and the noisy observation.

denominator and is defined as

$$\hat{u}(\hat{f}, a) = \mathcal{F}^{-1}\left(\frac{\overline{\mathcal{F}(a)}\mathcal{F}(\hat{f})}{|\mathcal{F}(a)|^2 + \frac{1}{\mathrm{SNR(u)}}}\right),$$

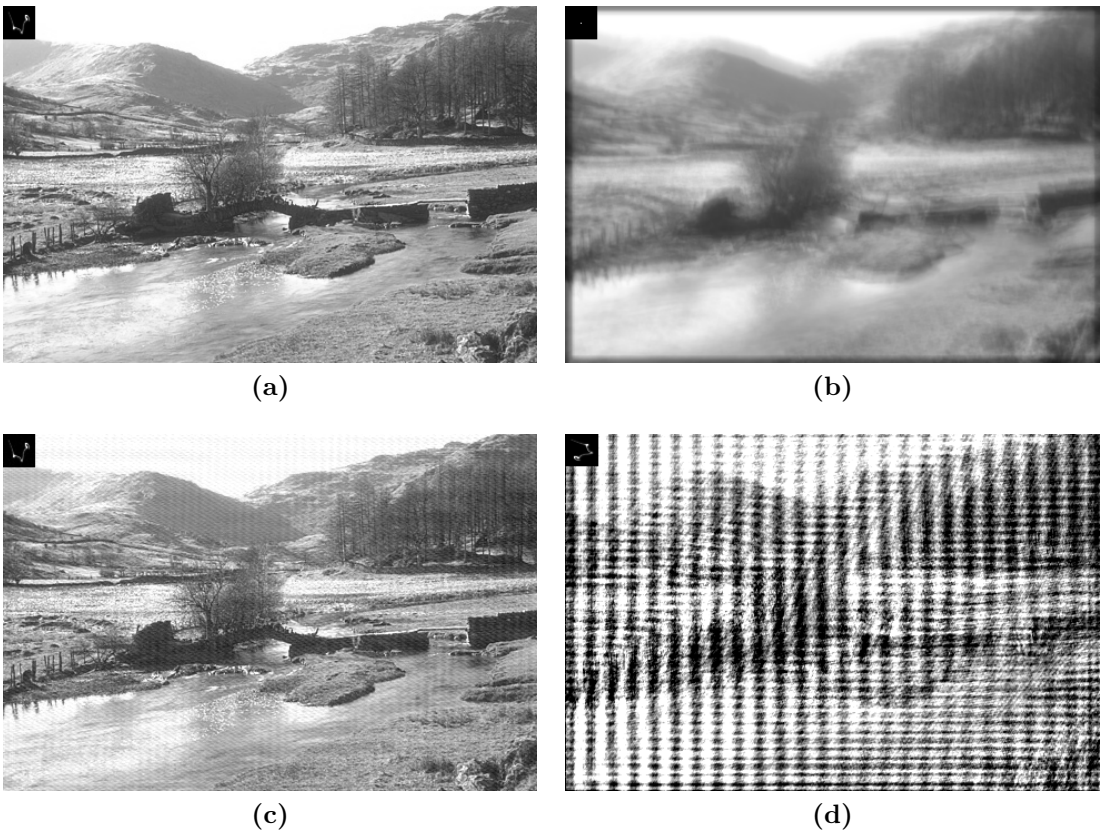
where $\bar{\cdot}$ denotes the complex conjugate and $\mathrm{SNR}(u)$ the signal noise ratio of u. Moreover, the multiplication and division on the frequency domain are point-wise. Consequently, the non-blind image deconvolution problem can be solved, however, the image quality of this simple method is rather poor.

1.2.4.2 Blind image deconvolution

Obviously the blind image deconvolution problem is also ill-posed as it has basically the same structure as its non-blind pendant. Yet, it is by far more difficult to solve since it does not even fulfill the uniqueness constrained of well-posed problems, as Chaudhuri et

al. [10] showed. The bottom line is because of the low-pass filtering nature of blurring, parts of the information are lost during acquisition. As a consequence, there exists a huge manifold of approximate solutions for the sharp image u and the blur kernel a yielding almost the same observation f within some noise level.

This fact is demonstrated by a second experiment. Once more, we consider a noise free but blurry image $\hat{f}$. If we reinterpret Equation (1.11), we can choose any blur kernel a and compute an image u such that we get exactly the same observation $\hat{f}$, when convolving a with u. Of course, this procedure is only successful if all elements of the *DFT* are different from zero, because the division by zero is not defined. A selection of kernel and image pairs all computing $\hat{f}$ when convolved, are depicted in Figure 1.5. Even though the difference of these four image and kernel pairs is huge, they all point to the same point in the solution space. Note that the same experiment still works if the noisy observation f instead of the



(a) **(b)** **(c)** **(d)**

Figure 1.5: Illustration of non uniqueness nature of blind image deconvolution: (a) true sharp image and correct blur kernel; (b) trivial solution consisting of the blurry observation and the delta kernel; (c) correct blur kernel corrupted with 1% Gaussian white noise and its associated image; (d) transposed true blur kernel and the computed image using Equation (1.11).

noise free $\hat{f}$ is used. The resulting images u would then be much more noisy, similar to Figure 1.4d.

As a result, neither the non-blind deconvolution problem nor the blind one can be suitably solved using simple inverse filtering approaches. However, a typical way to turn such problems into well-conditioned ones, is by exploiting all *a priori* knowledge about the result, thus reducing the space of possible solutions.

1.2.5 Bayesian inference

Without knowing the true sharp image u (and the blur kernel a in the blind case) it is difficult to formulate any explicit prior knowledge. Nevertheless, incorporating constraints is still possible if we consider the sharp images u (and the blur kernels a) to be samples of a probability distribution describing for example naturally plausible images (or kernels). We further denote this distribution as $p(u|, a, f)$ for non-blind image deconvolution and $p(u, a|f)$ in the blind case. For each hypothesis (u or (u, a)) a probability expressing how well it suits the observation (f) under the prior knowledge can be computed using the so called Bayesian framework.

The selection of the best hypothesis is usually done by minimizing the expected loss. For the non-blind deconvolution problem this minimization problem is defined by

$$u^* = \arg\min_u \int_{\mathcal{U}} \ell(u - \bar{u}) p(\bar{u}|a, f) d\bar{u},$$

whereby the loss function ℓ is typically chosen as a ℓ_p-norm and $\mathcal{U}$ denotes the set of all hypothesis. A very popular choice is the squared error loss $\ell = 1/2\,||\cdot||_2^2$, which yields

$$u^* = \arg\min_u \int_{\mathcal{U}} \frac{1}{2}\,||u - \bar{u}||_2^2\, p(\bar{u}|a, f) d\bar{u}.$$

The solution of this problem can be computed by setting its derivative to zero.

$$\int_{\mathcal{U}} (u^* - \bar{u}) p(\bar{u}|a, f) d\bar{u} = u^* \underbrace{\int_{\mathcal{U}} p(\bar{u}|a, f) d\bar{u}}_{1} - \int_{\mathcal{U}} \bar{u} p(\bar{u}|a, f) d\bar{u} \stackrel{!}{=} 0$$

$$u^* = \int_{\mathcal{U}} \bar{u} p(\bar{u}|a, f) d\bar{u} \tag{1.12}$$

Thus, this loss is minimized when computing the expected value of the distribution $p(u|a, f)$. The associated solution is called the mean squared estimate. Another important loss function is the zero-one loss. It assigns zero loss to the true image and one to all the others. Its norm based description in the infinite function space is tricky. Thus, we characterize it as $\ell(u - \bar{u}) = 1 - \delta(u - \bar{u})$, where $\delta(u - \bar{u})$ is defined such that it satisfies

$$\int_{\mathcal{U}} f(u)\delta(u - \bar{u}) du = f(\bar{u}).$$

Hence, it is a variant of the Dirac delta function. The minimizer of the associated expected loss can be computed by solving

$$u^* = \arg\min_u \int_{\mathcal{U}} (1 - \delta(u - \bar{u}))p(\bar{u}|a, f)d\bar{u}.$$

This problem can be reformulates as

$$\begin{aligned} u^* &= \arg\min_u \underbrace{\int_{\mathcal{U}} p(\bar{u}|a, f)d\bar{u}}_{1} - \int_{\mathcal{U}} \delta(u - \bar{u})p(\bar{u}|a, f)d\bar{u} \\ &= \arg\min_u 1 - p(u|a, f) \\ &= \arg\max_u p(u|a, f), \end{aligned} \tag{1.13}$$

because the integral of the Dirac delta function over the entire domain is 1. So, this loss is minimized by choosing the hypothesis which maximizes the conditional a posteriori probability. Therefore, this loss motivated approach is call Maximum A Posteriori (MAP) estimation within Bayesian inference theory. In general this problem is easier to solve, since it avoids the computation of the high dimensional integral in Equation (1.12). The same derivations can also be performed for blind image deconvolution.

Consequently, we are interested in finding the image u (and kernel a) maximizing the a posteriori distribution given the observation f.

$$\text{non-blind:} \qquad u^* = \arg\max_u p(u|a, f) \tag{1.14}$$

$$\text{blind:} \qquad (u^*, a^*) = \arg\max_{u,a} p(u, a|f). \tag{1.15}$$

By using Bayes rule we can further refine it to

$$\text{non-blind:} \quad p(u|a, f) = \frac{p(f|u, a)p(u)}{p(f)} \tag{1.16}$$

$$\text{blind:} \qquad p(u, a|f) = p(u|a, f)p(a) = \frac{p(f|u, a)p(u)p(a)}{p(f)}, \tag{1.17}$$

where $p(f|u, a)$ is the likelihood of the observation f given the hypothesis of the sharp image u and the blur kernel a, $p(u)$ specifies the prior distribution of sharp, natural images u and $p(a)$ the one of the kernel. The probability distribution $p(f)$ can be computed by marginalization over the likelihood $p(f|u, a)$. It is used as a normalization factor to ensure proper posterior probabilities. Hence, it can be neglected from the *MAP* formulation.

A still open question in research is how to model these probability distributions properly. In case of the likelihood term and when dealing with independent and identically

distributed (iid) additive white Gaussian noise, an often made choice is

$$p(f|u,a) = \prod_{i=1}^{MN} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{((a*u)_i - f_i)^2}{2\sigma^2}\right)$$
$$= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} ||a*u-f||_2^2\right), \tag{1.18}$$

where MN is the number of pixels of $f \in \mathbb{R}^{M\times N}$, σ the standard deviation of the Gaussian noise and $||\cdot||_2$ the ℓ_2-norm.

A classical but still broadly used prior model for natural images is

$$p(u) = \frac{1}{C_u} \exp\left(-\rho(Ku)\right), \tag{1.19}$$

where ρ is called the potential or penalty function, K a linear operator applied on an image u and C_u a normalization constant, which ensures that marginalization over $p(u)$ yields 1. The task of this prior is to sort out unnatural images and favor sharp naturally appealing ones. This simple yet powerful prior permits modeling a vast variety of statistical image properties in order to express the statistics of natural images.

For blind image deconvolution also a model of natural blur kernels is needed. From the physical process of blurring we know the follow two facts. First, it is impossible that a *PSF* removes any light, since it aggregates the light of its neighborhood to form the output intensity. Hence, the *PSF* must be positive on its entire domain. Second, blurring does not change the average light intensity of a scene because it does not remove or add light. Consequently, the integral over the whole domain of the *PSF* must be 1. For a general blur we cannot make any further assumptions like sparsity or anything else due to the vast diversity of blur kernels. These requirements in the continuous setting can be directly translated into the discrete one. Thus, all the elements of a discrete blur kernel a must be positive and sum up to 1. In other words a must life on the unit simplex, being defined as

$$\Delta = \left\{ a \in \mathbb{R}^{H\times W} : \sum_{i=1}^{HW} a_i = 1,\ a_i \geq 0\ i = 1 \dots HW \right\}. \tag{1.20}$$

Based on this definition we can define the indicator function of the unit simplex.

$$\delta_\Delta(a) = \begin{cases} 0 & \text{if } a \in \Delta \\ \infty & \text{else} \end{cases} \tag{1.21}$$

Using this knowledge we can state the kernel prior $p(a)$ formulated as

$$p(a) = \frac{1}{C_a} \exp(-\delta_\Delta(a)), \tag{1.22}$$

where C_a is a normalization constant, which ensures that this distribution integrates to 1. So, $p(a)$ is either one if a lies on the unit simplex or zero if a is outside of it.

All this prior information can be plugged into Equation (1.16) and (1.17) and if we omit the constants we end up with

$$\text{non-blind:} \quad p(u|a,f) \propto \exp\left(-\rho(Ku) - \frac{1}{2\sigma^2}\,||a*u-f||_2^2\right) \tag{1.23}$$

$$\text{blind:} \quad p(u,a|f) \propto \exp\left(-\rho(Ku) - \delta_\Delta(a) - \frac{1}{2\sigma^2}\,||a*u-f||_2^2\right). \tag{1.24}$$

If we compute the *MAP* solution based on these equations and take the negative logarithm, we get

$$\text{non-blind:} \quad u^* = \arg\min_u \rho(Ku) + \frac{1}{2\sigma^2}\,||a*u-f||_2^2 \tag{1.25}$$

$$\text{blind:} \quad (u^*,a^*) = \arg\min_{u,a} \rho(Ku) + \delta_\Delta(a) + \frac{1}{2\sigma^2}\,||a*u-f||_2^2\,. \tag{1.26}$$

Both arguments of the minimization problems can be interpreted as energy functionals. This yields the variational formulation of the image deconvolution problems, which are defined as

$$\text{non-blind:} \quad u^* = \arg\min_u E(u) = \rho(Ku) + \frac{\lambda}{2}\,||a*u-f||_2^2 \tag{1.27}$$

$$\text{blind:} \quad (u^*,a^*) = \arg\min_{u,a} E(u,a) = \rho(Ku) + \delta_\Delta(a) + \frac{\lambda}{2}\,||a*u-f||_2^2\,. \tag{1.28}$$

In variational models the first terms depending solely on the solution is called the regularization, while the second term, which originates from the likelihood, is usually known as the data term. Instead of the noise variance σ^2 a parameter λ is used to tune the trade-off between regularization and data fidelity. Thus, the proper choice of λ is closely related to the noise level within an image.

A similar derivation of this energy minimization approach is possible by using a Markov Random field (MRF) formulation. A *MRF* is a set of random variables defined over a graph $G = \mathcal{G}(\mathcal{V},\mathcal{E})$, which consists of vertexes $\mathcal{V}$ and their connections are defined by the edge set $\mathcal{E}$, see [1]. Typically a *MRF* assigns a random variable to each vertex, which models the data likelihood, and another one to each clique within the graph. A clique is a subset of a graph whose vertexes are fully connected. The statistical properties within the graph are modeled based on the cliques. This theory can be applied to images by considering the regular pixel grid as a graph, whose nodes are given by the pixels and the edges are defined by the incorporated neighborhood. Provided that the distribution is strictly positive, an associated Gibbs measure can be defined, which is based on a

Boltzmann distribution

$$p(x) = \frac{\exp(-E(x))}{\int_{x \in \mathcal{X}} \exp(-E(x))}.$$

In this formulation the energy $E(x)$ is set up such that it satisfy the properties of the graphical model. Hence, the energy of the associated Boltzmann distribution builds a bridge between *MRF* and variational models. However, the normalization of the probability distribution is in practice infeasible due to the large set of possible hypothesis $\mathcal{X}$. Therefore, the variational approach is easier to solve, as it avoid to evaluate high dimensional integrals.

In the following chapters we derive our image deconvolution methods based on these variational formulations.

2

Learning variational models for image deconvolution

Contents

The task of non-blind image deconvolution is rather theoretical because it uses a blur kernel as input. Therefore, it cannot directly be used for blind image deblurring. However, many recent blind deblurring methods [23, 24, 47] first estimate the unknown blur kernel and then deconvolve the blurred observation. This restoration process is frequently used as it circumvents the problem of naive Maximum A Posteriori (MAP) methods when jointly estimating the unknown blur kernel and the true image, see Chapter 3. Moreover, the Point Spread Function (PSF) and its associated blur kernel can be accurately measured in different imaging systems, e.g. Magnetic Resonance Imaging (MRI) or astronomy. For instance, the *PSF* of a telescope can be determined by pointing it to a bright isolated star. Thus non-blind image deconvolution is still an important problem.

2.1 Image deconvolution in energy minimization

As we have seen in the previous chapter, the task of non-blind image deconvolution is defined as:

> Given a blurry and noisy observation f of a scene and the according blur kernel a, estimate the sharp image u.

Moreover, we pointed out that in the continuous setting this problem is ill-posed, whereas its discrete version is well-posed with unacceptable noise driven solutions though. To handle this problem we incorporated prior knowledge about the true image u and ended up

with a variational approach. It is a widely adopted field of mathematical analysis dealing with maximization or minimization of functionals. If we interpret these functionals as energies, we wind up with the field of energy minimization. Consequently, *MAP* estimation, calculus of variations and energy minimization are closely related.

2.1.1 Formulation

A more general formulation of the non-blind image deconvolution problem than (1.27) is

$$u^* = \arg\min_u E(u) = R(u) + \frac{\lambda}{2} ||a * u - f||_2^2 , \tag{2.1}$$

where $R(u)$ is a regularization term expressing the prior knowledge about the true image u, λ is still a tuning parameter for weighting the data fidelity and $||\cdot||_2$ is the standard ℓ_2-norm. This formulation is basically the same for all *MAP* or energy minimization related methods. The diversity within these methods originates from the choice of the regularizer $R(u)$ and the optimization method. Most methods use the ℓ_2-norm around the image formation process due to the Gaussian noise assumption. However, there are some algorithms such as Xu and Jia [47] which use a ℓ_1-norm instead, to better cope with outliers. In the next section we look at these methods in more detail.

2.1.2 Popular regularization

An early approach to circumvent the ill-posedness of problems is called the Tikhonov regularization, which was introduced by Tikhonov and Arsenin [45]. It uses a quadratic penalty function and if applied to the deconvolution problem we get the following formulation

$$u^* = \arg\min_u E(u) = ||\nabla u||_2^2 + \frac{\lambda}{2} ||a * u - f||_2^2 , \tag{2.2}$$

where ∇ is the first-order finite difference operator. If we have a closer look at this energy functional we see that it is quadratic and convex and the solution u^* can be computed in closed form. It has to fulfill the first order optimality condition for unconstrained optimization problems defined as

$$\left.\frac{\partial E}{\partial u}\right|_{u^*} \stackrel{!}{=} 0.$$

To ease computation, we can rewrite the convolution operation using matrix-vector notation $a * u \Leftrightarrow Au$ without changing the energy functional, see Appendix B. The derivative is then given by

$$\frac{\partial E}{\partial u} = \nabla^\mathsf{T} \nabla u + \lambda A^\mathsf{T} (Au - f).$$

Thus, the optimal solution can be computed as

$$u^* = \lambda \left(\nabla^\mathsf{T} \nabla + \lambda A^\mathsf{T} A \right)^{-1} A^\mathsf{T} f.$$

The advantage of the Tikhonov regularization in this formulation is that the solution can be efficiently computed in closed form. However, the actual result is not satisfactory, see Figure 2.1b, although it is by far better than the output of the simple inverse filtering (Figure 1.4d). The difference to the blurry observation is rather small though. The edges in the output image of the Tikhonov model are over smooth due to the quadratic penalty function, which assigns high cost to steep edges.

A very popular regularization for image denoising that overcomes this problem was introduced by Rudin, Osher and Fatemi [38]. They proposed the so called Total Variation (TV), which in its isotropic discrete form is defined as

$$\mathrm{TV}(u) = ||\nabla u||_{2,1} = \sum_{i=1}^{MN} \sqrt{(\nabla_x u)_i^2 + (\nabla_y u)_i^2},$$

where ∇_x is the first-order difference operator in x-direction, ∇_y in y-direction respectively and the image u is of size $M \times N$. Soon after its introduction this model was also applied to non-blind image deconvolution [37] and its discrete formulation is given by

$$u^* = \arg\min_u E(u) = ||\nabla u||_{2,1} + \frac{\lambda}{2} ||a * u - f||_2^2 . \tag{2.3}$$

Compared to the Tikhonov model, see Equation (2.2), the ROF-model uses the ℓ_1-norm over the image gradients. This slight change lead to a dramatic improvement in image regularization. In Figure 2.1 the results of both models are depicted. While the output of the Tikhonov model suffers from over smooth edges, the ROF-model result posses sharp ones. This demonstrates the huge benefit of the *TV* regularization since it disfavors small

(a) **(b)** **(c)**

Figure 2.1: Image deconvolution using a Tikhonov and *TV* prior. (a) corrupted observation (blur kernel in upper left corner and 1% Gaussian noise); (b) deconvolution result with $\lambda = 5$; (c) resulting image when using a *TV* prior and $\lambda = 500$.

fluctuations like noise but allows steep discontinuities such as sharp edges. A drawback

Algorithm 1: Primal-dual algorithm of Chambolle and Pock [8].

Data: f, Choose: $\tau, \sigma > 0$, $\tau\sigma L^2 = 1$, $\theta \in [0, 1]$ and x_0, y_0
Result: x
$\bar{x}_0 \leftarrow x_0$;
while *not converged* **do**
  $y_{n+1} \leftarrow \text{prox}_{\sigma F^*}(y_n + \sigma K \bar{x}_n)$;
  $x_{n+1} \leftarrow \text{prox}_{\tau G}(x_n - \tau K^* y_{n+1})$;
  $\bar{x}_{n+1} \leftarrow x_{n+1} + \theta(x_{n+1} - n_n)$;
  $t \leftarrow t + 1$;
$x \leftarrow x_{t+1}$;

might be that it cannot be solved in closed form, however there exist efficient and accurate algorithms to solve it [8]. The *TV* is very popular within image regularization and is still widely used, even in todays State-of-the-Art (SotA) algorithms [47] for image deconvolution.

Let us have a closer look at the primal-dual algorithm of Chambolle and Pock [8]. It can solve saddle-point problems of the form

$$\min_x \max_y \langle Kx, y\rangle + G(x) - F^*(y),$$

where $G(x)$ and $F^*(x)$ are proper, convex, lower semi-continuous functions. Furthermore, F^* is the convex conjugate of a convex lower semi-continuous functions. Their approach is stated in Algorithm 1. In order to apply this algorithm, we need to map the *TV* regularized ROF model, see Equation (2.3), to a saddle-point problem of the above form. Therefore, we need the convex conjugate of a function, which is defined as

$$f^*(y) = \sup_x \langle y, x\rangle - f(x).$$

The biconjugate of a function f is then defined by

$$f^{**}(x) = \sup_y \langle x, y\rangle - f^*(y).$$

Basically it is the convex conjugate of the convex conjugate of a function. For a convex and lower semi-continuous function it follows that $f(x) = f^{**}(x)$. Hence, the convex function itself is given by its biconjugate. The *TV* can also be expressed using the ℓ_1-norm. Thus, we can set $f(x) = ||x||_1$ and since it is a convex and lower semi-continuous function, it can be substituted by its biconjugate. If we plug this idea into Equation (2.3), we get

$$u^* = \arg\min_u \max_p \ \langle \nabla u, p\rangle + \frac{\lambda}{2} ||Au - f||_2^2 - \delta_{||.||_\infty \leq 1},$$

Algorithm 2: Primal-dual algorithm for non-blind image deconvolution.

Data: f, Choose: $\tau, \sigma > 0$, $\tau\sigma L^2 = 1$, $\theta \in [0, 1]$ and u_0, p_0
Result: x
$\bar{u}_0 \leftarrow u_0$;
while *not converged* **do**
  $p_{n+1} \leftarrow \text{proj}_{||.||_\infty \leq 1}(p_n + \sigma \nabla \bar{u}_n)$;
  $u_{n+1} \leftarrow (I + \tau\lambda A^\mathsf{T} A)^{-1}((u_n - \tau \nabla^\mathsf{T} p_{n+1}) + \tau\lambda A^\mathsf{T} f)$;
  $\bar{x}_{n+1} \leftarrow x_{n+1} + \theta(x_{n+1} - n_n)$;
  $t \leftarrow t + 1$;
$x \leftarrow x_{t+1}$;

whereby A is the matrix associated to the convolution operation and $\delta_{||.||_\infty \leq 1}$ is the dual function of $||.||_1$, which is given by the indicator function of the dual norm ball. In order to apply Algorithm 1, we need to compute two proximal maps. The proximal map of the dual variable p can be easily computed because it is related to an indicator function. Thus, it is given by the projection onto the ℓ_∞-norm ball.

$$p = \text{prox}_{\tau\delta_{||.||_\infty \leq 1}}(\tilde{p}) = \text{proj}_{||.||_\infty \leq 1}(\tilde{p}) \Leftrightarrow p_i = \frac{\tilde{p}_i}{\max(1, ||\tilde{p}_i||_2)}$$

Moreover, the proximal map of the primal variable u can also be solved in closed form

$$u = \text{prox}_{\tau\frac{\lambda}{2}||Au-f||_2^2}(\tilde{u}) = \arg\min_u \frac{1}{2\tau}||u - \tilde{u}||_2^2 + \frac{\lambda}{2}||Au - f||_2^2 .$$

The solution of this quadratic, convex optimization problem is given by

$$u = (I + \tau\lambda A^\mathsf{T} A)^{-1}(\tilde{u} + \tau\lambda A^\mathsf{T} f).$$

Based on this results, we can state the primal-dual algorithm for non-blind image deconvolution, see Algorithm 2. It can be used to efficiently and accurately solve the ROF model. Later we will use this algorithm as a reference method.

If we take a closer look at Figure 2.1c, we see the major drawback of the ROF-model as it generates so-called 'stair-case' artifacts. These develop because the minimizer of the ROF-model favors piecewise-constant regions, which can be especially seen at the river or mountains in Figure 2.1c. This side effect can be avoided when incorporating higher order derivatives, as Bredies et al. [6] showed with the total generalized variation regularization. Due to additional information of higher order derivatives it correctly reconstructs steep edges and smooth regions.

All of the so far discussed regularizer are convex. Hence, the associated models have a global minimizer since also the data term is convex. However, if we consider the statistics of image gradients, depicted in Figure 2.2, we have to conclude that they follow a heavy

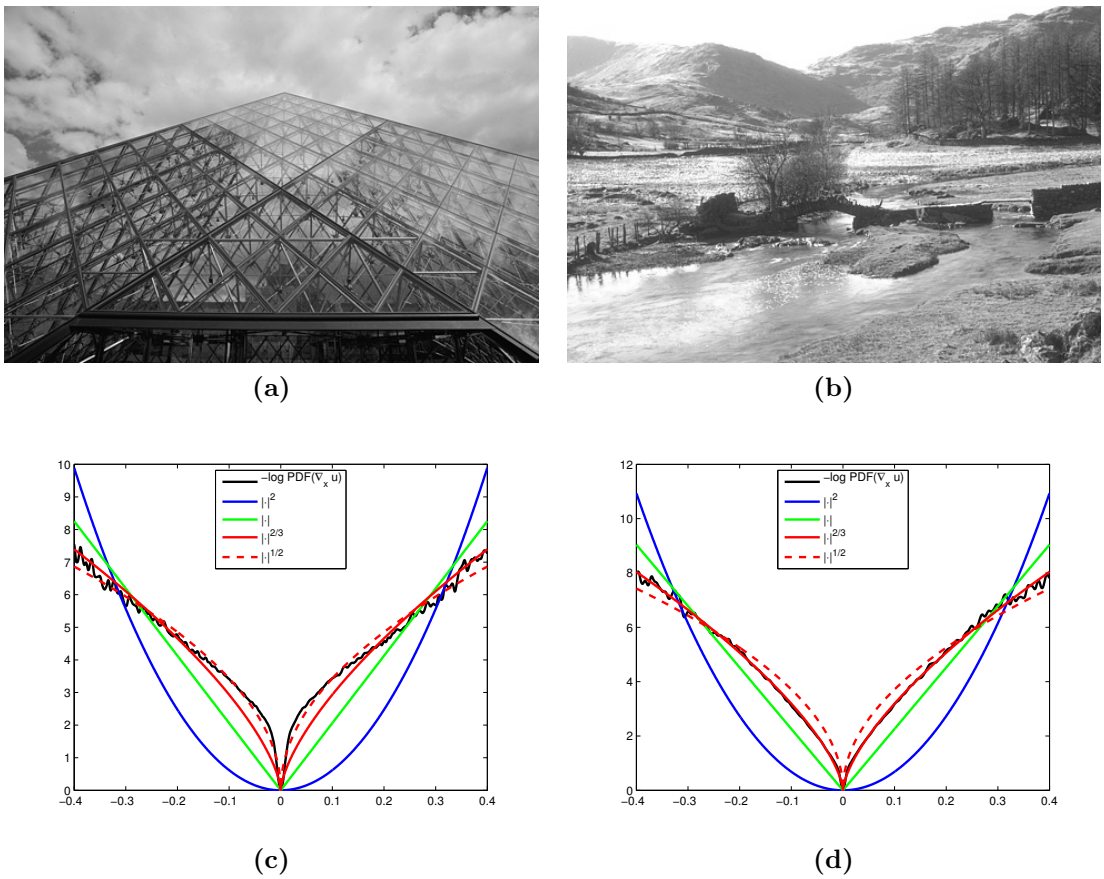


Figure 2.2: Negative logarithmic Probability Density Function (PDF) of the first order image gradients ($\nabla_x u$). In the top row are the images, the statistics are computed from. The bottom row depicts the associated *PDF* with fitted approximations. Note that for image (a) $|\cdot|^{\frac{1}{2}}$ fits best, whereas, the statistics of image (b) are better described using $|\cdot|^{\frac{2}{3}}$.

tailed distribution. Consequently, the quadratic penalty function of the Tikhonov model is a rather rough choice. The ℓ_1-norm of the *TV*, being the closet convex approximation, is better suited, which validates its success. It is a rather poor fit though.

From a statistical point of view, better regularizer embed the underlying statistics, which leads us to non-convex regularization. In 2007 Levin et al. [25] introduced a sparse non-convex prior on image gradients. Their model is formulated as

$$u^* = \arg\min_u E(u) = \sum_{i=1}^{MN} |(\nabla_x u)_i|^\alpha + |(\nabla_y u)_i|^\alpha + \lambda \, ||a * u - f||_1 \,, \tag{2.4}$$

where the image u is of size $M \times N$ and ∇_x and ∇_y are the directional first order image gradient operators. It employs a ℓ_1-norm data term, which is invariant to large outliers as shown in [8]. Moreover, it approximates the distribution of natural image gradients

by a hyper Laplacian distribution with $\alpha = 0.8$. As they pointed out, in contrast to a Gaussian quadratic prior, a sparse gradient prior concentrates the gradients to a small amount of pixel, while leaving the majority constant. This effect helps to reduce noise, generate sharper edges and avoid image artifacts such as ringing.

The downside of this model is that it is not convex anymore, in spite of the convex data term. Therefore, finding a global minimizer is a very hard problem, however, stationary points, i.e. plateaus, global and local minimas, can be still computed. Levin and colleagues used an iterative reweighted least squares process that optimizes a sequence of least squares problems whose derivative weights are updated based on the result of the previous iterations. The individual least squares problems are solved using a variant of the conjugate gradient algorithm.

Motivated by the long runtime of the iterative reweighted least squares algorithm, Krishnan and Fergus [22] proposed their own model, based on hyper-Laplacian distributions of image gradients. It is closely related to the previous model of Levin et al. and defined as

$$u^* = \arg\min_u E(u) = \sum_{i=1}^{MN} \sum_{k=1}^{N_k} |(f_k * u)_i|^\alpha + \frac{\lambda}{2} ||a * u - f||_2^2 , \tag{2.5}$$

where $N_k = 2$ typically, f_1 and f_2 are the first order x/y-directional derivative filters and $\alpha \in (0, 1)$ is the hyper-Lapalcian parameter. In their paper they experimented with different values for α, including $1/2$ and $2/3$. If these values are used, the associated Laplacian distribution fits the underlying data well, see Figure 2.2. This model would be equivalent to Levin's model if a ℓ_1-norm data term and $\alpha = 0.8$ was used. To optimize this non-convex energy functional, Krishnan and Fergus adopted a half-quadratic penalty method [16, 17], which uses auxiliary variables for each filter f_k. The two associated subproblems can then be efficiently solved using the Fourier transform for optimizing u and a lookup table or an analytic solution for computing the auxiliary variables (details see [22]) because they become independent over the pixels due to splitting. By using this approach, the runtime is up to 350 times faster than the one proposed by Levin et al.

2.1.3 Learning image regularization

So far all the proposed methods use regularization techniques constructed by humans to incorporate information about the true solution. As we have seen in the previous section, finding a suitable regularizer is hard, even for the rather basic first order directional derivative filters. Consequently, the field of learning image priors evolved, motivated by the success of machine learning and the urge for rich local image priors with larger domain to better describe more complex properties of natural images.

A still handcrafted first attempt to infer this kind of regularization was done by the pioneering work of Geman and Geman [18]. They interpreted images as graphs $G = \mathcal{G}(V, E)$, where the nodes V of the graph represent the image pixels and the edges E the

relations within the nodes based on some defined neighborhood. Their prior model is set up by a manifold of potential functions, which are defined on cliques that are inferred by the neighborhood. Each of these potential functions assigns a certain value (constant) to the energy, based on its clique.

If we transfer this idea to variational models and allow the potential functions to be proper functions, we end up with the Field of Experts (FoE) model [36], formulated as

$$R(u) = \sum_{p=1}^{MN} \sum_{i=1}^{N_k} \alpha_i \rho_i \left((k_i * u)_p \right), \tag{2.6}$$

where ρ_i model the potential functions, also known as experts, evaluated at a local clique, k_i is a convolution kernel determining the clique, α_i is the expert parameter fixing the influence of expert i and the image u is of size $M \times N$. Due to the flexibility of the experts and the formulation of the cliques as filters, this model is able to describe a huge variety of complex image statistics. In their introductory work [36], Roth and Black applied this regularizer to image denoising and inpainting. They fixed the expert functions to be either the heavy-tailed Student-t distribution or a smooth approximation of the ℓ_1-norm [9] and trained the associated filters k_i and expert parameters α_i. Despite the simplicity of this model, they achieved results close to *SotA* methods, even though they did not train their model for a specific task. The *FoE* prior was also applied to non-blind image deconvolution yielding also good results [11] due to its powerful statistical basis.

Chen et al. [13] took the idea a bit further. Based on the *FoE* model they proposed a reaction diffusion process for image restoration such as denoising and JPEG-deblocking. The success of this method lies in two points. First, they tremendously increase the expressive power of the *FoE* model by also learning the expert functions. Second, the fixed amount of iterations and the optimization of all model parameters lead to a fast and high quality image restoration method at *SotA* level. The related variational formulation of this model is defined as

$$u^* = \arg\min_u E(u) = \sum_{p=1}^{MN} \sum_{i=1}^{N_k} \rho_i \left((k_i * u)_p \right) + \frac{\lambda}{2} \left| |a * u - f| \right|_2^2, \tag{2.7}$$

where ρ_i is the i-th penalty function, k_i the associated linear filter and the image is of size $M \times N$. In this formulation we omitted the time-dependence of the parameters ϕ_i, k_i and λ to point out the relation to energy minimization. One interpretation of their algorithm is that they minimize the energy (Equation (2.7)) by just performing a certain amount of pure gradient descent steps. The parameters of each of those steps are then learned in a supervised manner such that the output best matches the true image. The success of this model in image restoration highly motivates our approach to learn specific models for image deconvolution.

A completely different approach for learning image priors, being patch based, was

introduced by Zoran and Weiss [49]. They maximize the so called Expected Patch Log Likelihood (EPLL) under a prior p, which is defined as

$$EPLL_p(x) = \sum_i \log p(P_i u), \tag{2.8}$$

where P_i is a matrix extracting the i-th patch of all overlapping patches from an image u. To incorporate the statistics of natural images, they model the prior p by means of Gaussian Mixture Models (GMM)

$$\log p(x) = \log \left(\sum_{j=1}^{J} \pi_j \mathcal{N}(x|\mu_j, \Sigma_j) \right),$$

where J defines the number of Gaussian mixtures, π_j are the mixing coefficients, which sum up to 1, and μ_j and Σ_j are the mean and covariance of the j-th Gaussian mixture. The parameters of this prior model are trained using the Expectation Maximization (EM) algorithm. Despite the fact that their prior is patch-based, the results of non-blind image deconvolution with this prior is on par with *SotA*.

2.1.4 Choice of weighting parameter λ

In the variational formulation of the non-blind image deconvolution, see Equation (2.1), basically two parameters can be tuned. First and probably most important the regularization, as we have seen in the previous section. Second, the weighting parameter λ. It balances the trade-off between data fidelity and regularization and is mainly defined by the assumed noise variance σ^2, which was pointed out in Chapter 1. Thus, one assumes that if the noise level (σ) stays constant and an image is convolved with different blur kernels, the best deconvolution results should be obtained with the same λ. However, in practice this is not the case.

To demonstrate this effect, the following experiment was set up. We took two images of the dataset of Levin et al. [28] and convolved each one with all eight blur kernels within this dataset. After the convolution we added Gaussian white noise with a standard deviation of $\sigma = 0.01$, which is equivalent to a 1% noise level, to all blurry images. Then these noisy and blurry observations were deconvolved using a *TV* prior and different values for λ by applying the primal-dual Algorithm 2. The λ values which resulted in the best PSNR values for the deconvolved images are depicted in Figure 2.3. Unfortunately, the maximum PSNR value for each image and kernel cannot be obtained for a certain λ. Not only depends the best estimated λ on the blur kernel but also on the image, as we can see if we compare both plots in Figure 2.3. Let us for instance consider kernel 4. While for image 1 $\lambda = 100$ is best, image 2 requires a three times larger one. Thus, this data dependency makes an automated choice for λ very hard. A typical approach to circumvent this problem is to start with low λ values and gradually increase them [28, 34].

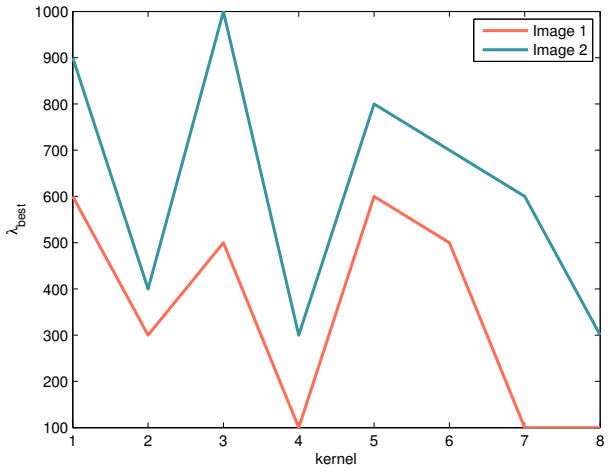


Figure 2.3: Demonstration of the influence of the weighting parameter λ. Both plots depict the λ_{best} value, which resulted in the best Peak Signal Noise Ratio (PSNR) values of the deconvolved image for varying blur kernels. The plots were generated using the first two blurry samples of the Levin et al. dataset [28].

2.2 Image deconvolution using iteratively adapted energy minimization

We formulate our non-blind image deconvolution approach based on the image denoising framework proposed by Chen et al. [13]. Although Chen's framework was originally designed as a reaction diffusion process, we motivate our approach by minimizing its associated energy. For more details about the relation between energy minimization and diffusion processes we refer to [12]. By applying the associated regularizer of Chen's framework into our general non-blind deconvolution variational formulation, we end up with

$$u^* = \arg\min_u E(u) = \frac{1}{N_k} \sum_{p=1}^{MN} \sum_{i=1}^{N_k} \rho_i \left((k_i * u)_p \right) + \frac{\lambda}{2} ||a * u - f||_2^2 , \tag{2.9}$$

where ρ_i is an arbitrary smooth penalty function, k_i a linear filter and λ a weighting parameter. In contrast to Chen's reaction diffusion process, we added a normalization factor $1/N_k$ for implementation issues. Since the convolution operation can be expressed as matrix-vector product $a*u \Leftrightarrow Au \Leftrightarrow Ua$ with suitably defined matrices, see Appendix B, our formulation changes to

$$u^* = \arg\min_u E(u) = \frac{1}{N_k} \sum_{p=1}^{MN} \sum_{i=1}^{N_k} \rho_i \left((K_i u)_p \right) + \frac{\lambda}{2} ||Au - f||_2^2 . \tag{2.10}$$

Unfortunately, this energy cannot be solved in closed form because the smooth penalty functions ρ_i can be arbitrarily shaped. However, the first and easiest algorithm to minimize this kind of energy is the first-order steepest gradient descent method. If one wants to minimize a function $f(x)$, it is defined as

$$x_t = x_{t-1} - h_t \nabla f(x_{t-1}), \tag{2.11}$$

where h_t is a proper step size and $\nabla f(x)$ the gradient of $f(x)$. This method generates a sequence $\{x_t\}_1^T$ that converges to a stationary point, which can either be a local/global minimum/maximum or a saddle point. In order to apply this method, we have to compute the gradient of $E(u)$, which is given by

$$\nabla E(u) = \frac{1}{N_k} \sum_{i=1}^{N_k} K_i^\mathsf{T} \phi_i(K_i u) + \lambda A^\mathsf{T}(Au - f), \tag{2.12}$$

whereby $\phi(x) = \partial\rho(x)/\partial x$ is the derivative of the scalar function $\rho(x)$. Hence, we can apply the steepest gradient descent method to our problem and get the following iterative update rule

$$u_t = u_{(t-1)} - h_t \left\{ \frac{1}{N_k} \sum_{i=1}^{N_k} K_i^\mathsf{T} \phi_i(K_i u_{(t-1)}) + \lambda A^\mathsf{T}(A u_{(t-1)} - f) \right\}, \tag{2.13}$$

where we have to select a suitable step size h_t in each iteration t.

Usually, this update scheme is repeated in hundreds or thousands of iterations due to the poor convergence rate of the first-order steepest gradient descent method. Analogue to Chen, we fix the amount of iterations to T and try to find the best update scheme (2.13), given some objective. To do so, we break up the chains of this static update strategy by allowing free changes of the regularizer parameters ϕ_i and k_i and the weighting parameter λ. As a result, we transfer our approach from an iterative scheme to a network structure, whose stages are motivated by the iterative updates (2.13). If we cast this idea into a mathematical formulation, we can define the update of a certain stage t within our deconvolution process as

$$u_t = \begin{cases} u_{(t-1)} - h_t \left\{ \frac{1}{N_k} \sum_{i=1}^{N_k} K_{ti}^\mathsf{T} \phi_{ti}(K_{ti} u_{(t-1)}) + \lambda_t A^\mathsf{T}(A u_{(t-1)} - f) \right\} & \text{if } t > 0 \\ u_0 & \text{if } t = 0. \end{cases}$$

Due to this change, the scale of the influence functions $\phi_{ti}(x)$ and λ_t can be arbitrary selected within each stage. Therefore, we can define a new set of functions $\phi_{ti}(x) = h_t \tilde{\phi}_{ti}(x)$ and weights $\lambda_t = h_t \tilde{\lambda}_t$ without loss of generality. As a consequence, the update scheme

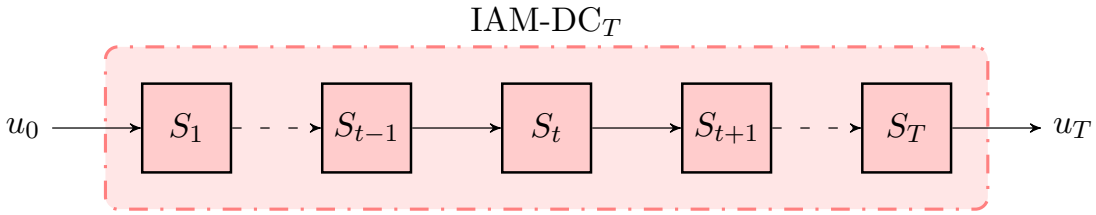


Figure 2.4: Demonstration of the Iteratively Adapted Minimization for non-blind Deconvolution (IAM-DC) network structure. The input of the procedure is the blurry observation u_0, which is processed by a series of stages S_t, implementing the update rule (2.14). The output of the net after performing T updates is the deconvolved image u_T.

changes to

$$u_t = \begin{cases} u_{(t-1)} - \left\{ \frac{1}{N_k} \sum_{i=1}^{N_k} K_{ti}^{\mathsf{T}} \phi_{ti}(K_{ti} u_{(t-1)}) + \lambda_t A^{\mathsf{T}}(A u_{(t-1)} - f) \right\} & \text{if } t > 0 \\ u_0 & \text{if } t = 0, \end{cases} \tag{2.14}$$

which is identical to the denoising formulation of Chen et al. [13] despite the presence of the convolution operator A in the reaction term. Figure 2.4 depicts the structure of the proposed *IAM-DC* network. The input u_0 is the blurry observation, which is initially the best guess for the true image, since we cannot predict the sharp one. The initial guess is then processed by a sequence of successive stages S_t until the final estimate u_T is restored after T updates.

Due to the relaxation of the regularization parameters and λ, this approach does not minimize the energy in Equation (2.10) exactly. However, an interpretation is that in each stage a gradient descent step of a related energy functional, which depends on the network parameters (ϕ_{ti}, k_{ti} and λ_t), is performed. Thus the network attempts to minimize in every stage an iteratively adapted energy of the form

$$E_t(u) = \frac{1}{N_k} \sum_{p=1}^{MN} \sum_{i=1}^{N_k} \frac{1}{h_t} \rho_{ti}\left((K_{ti} u)_p\right) + \frac{\lambda_t}{2h_t} \|Au - f\|_2^2 ,$$

where h_t represents the latent step size at stage t.

2.2.1 A straight forward IAM-DC network

Motivated by the simple update rule of Equation (2.14), we introduce a first *IAM-DC* based network, whose successive stages are defined as

$$u_t = T\left\{ u_{p(t-1)} - \left(\frac{1}{N_k} \sum_{i=1}^{N_k} K_{ti}^{\mathsf{T}} \phi_{ti}(\eta K_{ti} u_{p(t-1)}) + \lambda_t A^{\mathsf{T}}\left(A u_{p(t-1)} - f\right) \right) \right\}, \tag{2.15}$$

where $u_{p(t-1)} = Pu_{(t-1)}$ is a padded version of the input to this stage. This padding is required to limit the influence of inconsistent border handling. The operator T is a matrix representation of a truncation operation, which is necessary to ensure that the image size of $u_{(t-1)}$ and u_t is the same. In contrast to the previous model, we added a constant factor η, which is used to scale the argument of the influence functions ϕ_{ti} for implementation issues. The basic idea of this model is to use the expressive power of the regularization, defined by the kernels k_{ti} and the associated penalty functions $\rho_{ti} = \int \phi_{ti}(x)dx$, to learn better priors respective regularizer. This is achieved by training the network.

2.2.1.1 Training

Due to the success in image denoising of Chen's framework [13] and other discriminative learning approaches in the filed of image deconvolution [39–41], we also train the networks in a discriminative fashion. Therefore, we define an objective function

$$L(\Theta) = \sum_{s=1}^{S} \ell(u_T(\Theta, u_0^s), u_{gt}^s), \tag{2.16}$$

which is minimized by a training algorithm. In this formulation S defines the amount of training samples and each of those consists of a blurry and noisy observation u_0^s and its associated ground truth, which is the true sharp image u_{gt}^s. Furthermore, we see that the output of the *IAM-DC* network u_T is a function of the input u_0^s and the network parameters Θ, which are defined as

$$\begin{aligned} \Theta &= \{\theta_t,\ t = 1 \dots T\} \\ \theta_t &= \{\lambda_t, (\phi_{ti}, k_{ti})\ i = 1 \dots N_k\}. \end{aligned}$$

The usual choice of the function $\ell(u_T, u_{gt}^s)$ is based on a certain image measure. In our case we want to maximize the *PSNR*, which is larger the smaller the squared error of the image differences is

$$\ell(u_T, u_{gt}^s) = \frac{1}{2} \left|\left| M(u_T - u_{gt}^s) \right|\right|_2^2, \tag{2.17}$$

where the diagonal matrix M is used to suppress errors at border regions. This elimination of an error at the border is necessary because all typical border-handling methods (zero-padding, cyclic or symmetric repetition of the image values) induce errors. Moreover, if the cropping was not done, also image inpainting at border regions would be part of the deconvolution problem. For the sake of simplicity we further on denote the output of the network stage t for training sample s as u_t^s.

Along with [13, 39], which solve similar problems, we use the famous L-BFGS algorithm [14] to minimize the objective function (2.16). It is a memory efficient quasi-Newton gradient related optimization algorithm which was successfully applied to many large scale

problems. In order to use it for minimizing an objective function, one just needs to compute the function value and its derivative with respect to its parameters for a given point in the parameter space. Both computations are necessary since it employs sophisticated line-search schemes.

Due to the complexity of non-blind image deconvolution, we cannot assume that the network will deliver good results for just a few (one or two) stages, thus some kind of pre-training is necessary. The task of a pre-training phase is to guide the network parameters to an already good solution, which can be further improved by a final joint training. Furthermore, the depth of a network drastically increases its expressive power. Along with its expressiveness also the complexity of the network increases, which hardens joint training. Therefore, we split the training of the networks into two phases. First, the network is greedily pre-trained until a depth of 5 is reached, then these stages are trained jointly. Afterwards the next 5 stages are added in a greedy manner and trained jointly, while keeping the parameters of the first 5 stages constant. This block-training procedure is repeated until the final depth is reached.

Greedy pre-training: To get good initial parameters for the joint training, we employ a divide and conquer strategy. For that reason, we use a greedy scheme which allows to train the parameters θ_t of each stage independently. We start with a single stage S_1, train its parameters using L-BFGS and then fix its output u_1. Afterwards, another stage, which uses the constant output of the previous stage as input, is added and its parameters θ_2 are trained. In the same way, also the output of this stage u_2 is fixed. By following these scheme we successively add stages and train them independently until the next joint training phase is executed. The training of the individual stages is done by minimizing the following objective function

$$L(\theta_t) = \sum_{s=1}^{S} \ell(u_t^s, u_{gt}^s). \tag{2.18}$$

Hence, one computes the parameter set θ_t such that the output of stage t resembles the true image as close as possible. This greed of each stage for the best fit gives the applied approach its name.

However, the important advantage compared to the joint training is that u_t^s just depends on the parameters of the current stage θ_t, given the constant output of the previous stage $u_{(t-1)}^s$. After the pre-training of each stage, the parameters θ_t are stacked together to form the initial estimate.

Joint training: The aim of this training phase is to refine the parameters of the current network part, in order to minimize the objective function (2.16). It is again optimized using the L-BFGS algorithm. In contrast to the pre-training, the flexibility while minimizing the objective function is much larger because here the intermediate latent images u_t can

be arbitrary. Consequently, this training stage is the most important, since it can exploit the entire complexity of the network.

In the next section the computation of the derivatives required for training are outlined. Furthermore, it refines the parametrization of the influence functions ϕ_{ti} and its associated kernels k_{ti}.

2.2.1.2 Gradient derivation

Most of the derivations are the same as in [12] due to the similarity of the approaches. Analogue to the training phases, also the derivation of the gradients is partitioned, starting with those necessary for pre-training.

In order to minimize (2.18) using L-BFGS, we need to compute its derivative with respect to the parameters θ_t of a single stage.

$$\frac{\partial L}{\partial \theta_t} = \sum_{s=1}^{S} \frac{\partial \ell(u_t^s, u_{gt}^s)}{\partial \theta_t}. \tag{2.19}$$

For the sake of simplicity we consider only one training example in the following derivations, since the gradient of more training samples can be easily computed by simply summing those of the individual ones. Consequently, Equation (2.19) can be expressed by summing up all the following derivatives of the individual training samples.

$$\frac{\partial \ell(u_t, u_{gt})}{\partial \theta_t} = \frac{\partial \ell}{\partial \theta_t} = \frac{\partial u_t}{\partial \theta_t} M^\mathsf{T} M(u_t - u_{gt}) = \frac{\partial u_t}{\partial \theta_t} \underbrace{M(u_t - u_{gt})}_{=:e_t} = \frac{\partial u_t}{\partial \theta_t} e_t.$$

$M^\mathsf{T} M = M$, since M is a diagonal matrix holding just zeros and ones. In this notation we introduced a new variable e_t, which defines the error at each pixel given the current output u_t. The derivative of the stage output with respect to its parameters can be computed by splitting up the parameter vector θ_t into its individual components.

Weighting parameter λ_t: This derivative can be easily computed using the matrix calculus and is defined as

$$\frac{\partial u_t}{\partial \lambda_t} = -(A u_{p(t-1)} - f)^\mathsf{T} A^\mathsf{T} T^\mathsf{T}, \tag{2.20}$$

where T^T is the adjoint operator of the truncation which is basically a padding with zeros. Thus $\partial \ell / \partial \lambda_t$ is given by

$$\frac{\partial \ell}{\partial \lambda_t} = -(A u_{p(t-1)} - f)^\mathsf{T} A^\mathsf{T} T^\mathsf{T} e_t.$$

Regularizer filters k_{ti}: The computation of this derivative requires exact knowledge about the problem. If we start with the convolutional formulation of the stage update rule (2.15) and omit all parts which do not depend on k_{ti}, we get

$$\frac{\partial u_t}{\partial k_{ti}} = -\frac{1}{N_k}\frac{\partial}{\partial k_{ti}}\left(g(k_{ti}) * \underbrace{\phi_{ti}(\eta u_{(t-1)} * k_{ti})}_{h(k_{ti})}\right) T^\mathsf{T}, \tag{2.21}$$

where $g(k_{ti})$ is a function miming the behavior of the transpose operation on a convolution matrix K_{ti}^T. This function can be further expressed as

$$g(k_{ti}) = R_{180}k_{ti}, \tag{2.22}$$

whereby the matrix R_{180} computes the adjunct of the kernel k_{ti} by simply rotating it by 180 degree. Note, that this operation is not valid for all boundary-handling methods of the convolution operation. Based on this formulation we can refine the above derivative by using the chain rule

$$\frac{\partial u_t}{\partial k_{ti}} = -\frac{1}{N_k}\left(\frac{\partial g}{\partial k_{ti}}\frac{\partial u_t}{\partial g} + \frac{\partial h}{\partial k_{ti}}\frac{\partial u_t}{\partial h}\right) T^\mathsf{T}$$

The outer convolution $g * h$ in Equation (2.21) can be expressed by its associated matrix-vector representation $Gh \Leftrightarrow Hg$. Consequently, the individual derivatives are

$$\begin{aligned}
\frac{\partial u_t}{\partial g} &= H^\mathsf{T} \\
\frac{\partial g}{\partial k_{ti}} &= R_{180}^\mathsf{T} \\
\frac{\partial u_t}{\partial h} &= G^\mathsf{T} = (K_{ti}^\mathsf{T})^\mathsf{T} = K_{ti} \\
\frac{\partial h}{\partial k_{ti}} &= \eta U_{p(t-1)}^\mathsf{T}\mathrm{diag}\left(\phi_{ti}'(\eta U_{p(t-1)}k_{ti})\right) = \eta U_{p(t-1)}^\mathsf{T}\Lambda_{ti},
\end{aligned}$$

where $U_{p(t-1)}$ is a convolution matrix for that $k_{ti} * u_{p(t-1)} \Leftrightarrow U_{p(t-1)}k_{ti} \Leftrightarrow K_{ti}u_{p(t-1)}$ is equivalent and $\phi_{ti}'(\eta U_{p(t-1)}k_{ti})$ denotes the derivative of the function $\phi_{ti}(x)$ evaluated at every pixel of the image $\eta u_{p(t-1)} * k_{ti}$. Based on this results we can state the desired gradient

$$\frac{\partial u_t}{\partial k_{ti}} = -\frac{1}{N_k}\left(R_{180}^\mathsf{T}H^\mathsf{T} + \eta U_{p(t-1)}^\mathsf{T}\Lambda_{ti}K_{ti}\right) T^\mathsf{T}. \tag{2.23}$$

As pointed out by Chen [12] the convolution $K_{ti}^\mathsf{T}\phi_{ti}(K_{ti}u_{p(t-1)})$ suffers from a scaling problem. To overcome it, we fix the kernel norm and let the influence functions be arbitrary. An easy method to fix their norm to one, is to construct them from normed basis functions.

A suitable choice is the discrete cosine transform basis, as it forms a basis for the whole space the filters live in. The constant basis function of it can be omitted to enforce the derivative nature of the filters. For filters of size $K_s \times K_s$ the corresponding discrete cosine basis B is of size $K_s^2 \times (K_s^2 - 1)$, where the constant basis function is already neglected. Thus, the filter kernels can be constructed by

$$k_{ti} = B \frac{c_{ti}}{||c_{ti}||_2},$$

which ensures that the norm of the kernels is always one. However, one has to compute the gradient of ℓ with respect to c_{ti}, since it is the actual training parameters.

$$\begin{aligned} \frac{\partial \ell}{\partial c_{ti}} &= \frac{\partial k_{ti}}{\partial c_{ti}} \frac{\partial \ell}{\partial k_{ti}} \\ \frac{\partial k_{ti}}{\partial c_{ti}} &= \frac{1}{||c_{ti}||_2} \left(I - \frac{c_{ti} c_{ti}^\mathsf{T}}{c_{ti}^\mathsf{T} c_{ti}} \right) B^\mathsf{T} \end{aligned}$$

Combining the above derivatives yields

$$\frac{\partial \ell}{\partial c_{ti}} = -\frac{1}{N_k \, ||c_{ti}||_2} \left(I - \frac{c_{ti} c_{ti}^\mathsf{T}}{c_{ti}^\mathsf{T} c_{ti}} \right) B^\mathsf{T} \left(R_{180}^\mathsf{T} H^\mathsf{T} + \eta U_{p(t-1)}^\mathsf{T} \Lambda_{ti} K_{ti} \right) T^\mathsf{T} e_t.$$

Influence functions ϕ_{ti}: If we derive the stage update Equation (2.15) with respect to the influence functions, we get

$$\frac{\partial u_t}{\partial \phi_{ti}} = -\frac{1}{N_k} \frac{\partial}{\partial \phi_{ti}} \left(K_{ti}^\mathsf{T} \phi_{ti}(K_{ti} u_{p(t-1)}) \right) T^\mathsf{T}.$$

In oder to make the influence functions trainable, some kind of parametrization is necessary. In our work we use Gaussian radial basis functions because they have been successfully used in many regression tasks. Hence, a single influence function can be expressed as

$$\phi_{ti}(x) = \sum_{j=1}^{N_w} w_{ti}^j \varphi \left(\frac{x - \mu_j}{\gamma} \right),$$

whereby μ_j is the mean of the j-th basis function and γ its standard deviation and the individual basis functions are Gaussian $\varphi(x) = \exp(-x^2/2)$. This parametrization can also be expressed in terms of a matrix vector product, where the matrix $\Phi(x)$ is constructed by all basis functions and the weights w_{ti}^j form a vector. If ϕ_{ti} is applied to a vector $x \in \mathbb{R}^O$,

we get

$$\underbrace{\begin{pmatrix} \varphi\left(\frac{x_1-\mu_1}{\gamma}\right) & \cdots & \varphi\left(\frac{x_O-\mu_1}{\gamma}\right) \\ \vdots & \ddots & \vdots \\ \varphi\left(\frac{x_1-\mu_{N_w}}{\gamma}\right) & \cdots & \varphi\left(\frac{x_O-\mu_{N_w}}{\gamma}\right) \end{pmatrix}}_{=:\Phi(x)} \begin{pmatrix} w_{ti}^1 \\ w_{ti}^2 \\ \vdots \\ w_{ti}^{N_w} \end{pmatrix} = \begin{pmatrix} \phi_{ti}(x_1) \\ \phi_{ti}(x_2) \\ \vdots \\ \phi_{ti}(x_O) \end{pmatrix}.$$

As a result, we have to compute the derivative with respect to the actual parameter of the influence functions, which is w_{ti}.

$$\begin{aligned} \frac{\partial u_t}{\partial w_{ti}} &= -\frac{1}{N_k}\frac{\partial}{\partial w_{ti}}\left(K_{ti}^\mathsf{T}\Phi_{ti}(K_{it}u_{p(t-1)})w_{ti}\right)T^\mathsf{T} \\ &= -\frac{1}{N_k}\Phi^\mathsf{T}(K_{it}u_{p(t-1)})K_{ti}T^\mathsf{T} \end{aligned}$$

Finally, we can state the derivative of the loss function ℓ with respect to the influence function weights w_{ti}

$$\frac{\partial \ell}{\partial w_{ti}} = -\frac{1}{N_k}\Phi^\mathsf{T}(K_{it}u_{p(t-1)})K_{ti}T^\mathsf{T}e_t. \tag{2.24}$$

Finally, we computed all derivatives to express $\partial\ell/\partial\theta_t$ and consequently the pre-training can be performed. However, the parameter set of a single stage changed to

$$\theta_t = \{\lambda_t, (w_{ti}, c_{ti})\ i = 1 \dots N_k\},$$

due to the re-parametrization of the influence functions and their kernels.

Joint training: Analogue to the greedy pre-training phase we need to compute the gradient of the objective function with respect to the parameters. In this case, these are the parameters of the current network block Θ, though. Furthermore, just the block output u_T is processed by L, see (2.16). Hence, the derivative of L for the joint traning of a single block is given by

$$\frac{\partial L}{\partial \Theta} = \sum_{s=1}^{S} \frac{\partial \ell(u_T^s, u_{gt})}{\partial \Theta}.$$

For the same reasons as above, considering a single training example for the computation of this derivative is sufficient. Thus, we just compute the gradient $\partial\ell(u_T, u_{gt})/\partial\Theta$, which

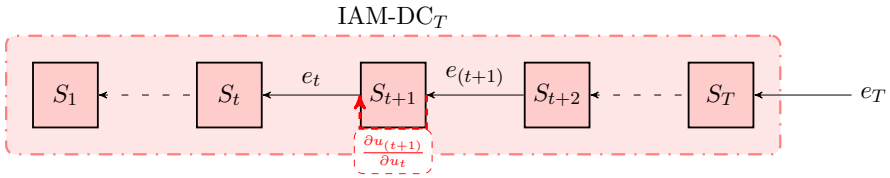


Figure 2.5: Illustration of the error back propagation within a *IAM-DC* net. The error at the final stage e_T is back propagated through the network by following the reverse path within the stage updates. This scheme enables an efficient training.

is given by

$$\frac{\partial \ell(u_T, u_{gt})}{\partial \Theta} = \frac{\partial \ell}{\partial \Theta} = \frac{\partial u_T}{\partial \Theta} \underbrace{M(u_T - u_{gt})}_{=:e_T} = \frac{\partial u_T}{\partial \Theta} e_T.$$

Computing $\partial u_T / \partial \Theta$ directly is rather difficult, however, the network structure can be exploited. Therefore, we group the parameters Θ according to their stages S_t and compute $\partial u_T / \partial \theta_t$ for each stage of the current network block. In the context of Neural Network (NN) this divide and conquer approach is well known as the back-propagation algorithm. So, we need to compute $\partial \ell / \partial \theta_t$, which can be done by the successive use of the chain rule

$$\begin{aligned}
\frac{\partial \ell}{\partial \theta_t} &= \frac{\partial u_{(T-1)}}{\partial \theta_t} \underbrace{\frac{\partial u_T}{\partial u_{(T-1)}} e_T}_{e_{(T-1)}} &&= \frac{\partial u_{(T-1)}}{\partial \theta_t} e_{(T-1)} \\
&= \frac{\partial u_{(T-2)}}{\partial \theta_t} \underbrace{\frac{\partial u_{(T-1)}}{\partial u_{(T-2)}} e_{(T-1)}}_{e_{(T-2)}} &&= \frac{\partial u_{(T-2)}}{\partial \theta_t} e_{(T-2)} \\
&\vdots && \vdots \\
&= \frac{\partial u_t}{\partial \theta_t} \underbrace{\frac{\partial u_{(t+1)}}{\partial u_t} e_{(t+1)}}_{e_t} &&= \frac{\partial u_t}{\partial \theta_t} e_t.
\end{aligned}$$

This scheme can be stopped as soon as stage t is reached, because the latent images u_i, $i < t$ are not influenced by the parameters of stage t. Moreover, we can extract a recursive definition for the error e_t, which is generated by the output of stage t, and is defined as

$$e_t = \frac{\partial u_{(t+1)}}{\partial u_t} e_{(t+1)}.$$

The recursive manner of this back propagation scheme is depicted in Figure 2.5. Because we already computed $\partial u_t / \partial \theta_t$ for the pre-training, we just need to compute the gradient of the output of a stage with respect to its input. Due to the padding, this gradient can

be constructed by two parts.

$$\frac{\partial u_{(t+1)}}{\partial u_t} = \frac{\partial u_{tp}}{\partial u_t} \frac{\partial u_{(t+1)}}{\partial u_{tp}}$$

Based on the stage update rule, the first gradient can be easily obtained by using the chain rule of the matrix calculus and is given by

$$\frac{\partial u_{(t+1)}}{\partial u_{tp}} = \left\{ I - \left(\frac{1}{N_k} \sum_{i=1}^{N_k} \eta K_{(t+1)i}^{\mathsf{T}} \Lambda_{(t+1)i} K_{(t+1)i} + \lambda_{(t+1)} A^{\mathsf{T}} A \right) \right\} T^{\mathsf{T}},$$

where $\Lambda_{(t+1)i}$ again is a diagonal matrix holding the first oder derivatives of the influence function $\phi'_{(t+1)i}$ evaluated at the image $\eta K_{(t+1)i} u_{pt}$. The second gradient is defined as

$$\frac{\partial u_{tp}}{\partial u_t} = P^{\mathsf{T}},$$

which is basically the adjoint operator of the padding operation. By putting both parts together we get

$$\frac{\partial u_{(t+1)}}{\partial u_t} = P^{\mathsf{T}} \left\{ I - \left(\frac{1}{N_k} \sum_{i=1}^{N_k} \eta K_{(t+1)i}^{\mathsf{T}} \Lambda_{(t+1)i} K_{(t+1)i} + \lambda_{(t+1)} A^{\mathsf{T}} A \right) \right\} T^{\mathsf{T}}.$$

This recursive back-propagation of the error, enables an efficient training scheme, where first the error information is spread through the network and then the individual gradient $\partial \ell / \partial \theta_t$ are evaluated. The final gradient $\partial \ell / \partial \Theta$ is then given by stacking the ones of the stages.

2.2.1.3 Evaluation

In order to evaluate the proposed network for different hyper-parameters such as net depth, kernel size and so forth, we trained each variant on the same training dataset. It consists of 200 samples of the form $(u_0^s, f^s, u_{gt}^s, a^s)$, where u_0^s is the initial image estimate, f^s the blurry observation, u_{gt}^s the true sharp image and a^s the true blur kernel. 100 of these samples use the blurry observation as initial image estimate $u_0^s = f^s$, whereas, the remaining 100 samples are initialized with the true image $u_0^s = u_{gt}^s$. This split helps the regularization to better distinguish between sharp and blurry images. All the training images are cropped patches extracted from images in the BSD500 dataset [30] and the blur kernels are randomly sampled from the dataset proposed by Schelten et al. [39].

Throughout the different tested *IAM-DC* networks, we use $N_w = 63$ Gaussian radial basis functions to parametrize the influence functions ϕ_{ti} and an argument scaling parameter $\eta = 1/1.25$. The Gaussian basis functions are uniformly distributed in the interval $[-1, 1]$ and have a standard deviation $\gamma = 2/N_w$. The influence functions were initialized

	TV		IAM-DC$_{10}^{3\times3}$		IAM-DC$_{15}^{3\times3}$		IAM-DC$_{20}^{3\times3}$	
Dataset	PSNR	σ	PSNR	σ	PSNR	σ	PSNR	σ
Levin	29.361	3.584	30.820	2.065	31.334	2.113	31.548	2.123
BSD500	27.285	3.489	28.267	3.037	28.666	3.018	28.892	2.999

Table 2.1: Comparison of the experimental results for the proposed *IAM-DC*$^{3\times3}$ nets. Depicted are the average PSNR(dB) and the corresponding standard deviation σ for the data set of Levin et al. [28] and 32 image samples from the BSD500 dataset [30] convolved with the kernels of Levin. The TV results are computed by using the primal-dual algorithm of Chambolle and Pock [8], which is outlined in Algorithm 2. While all the network filters k_{ti} are of size 3×3, the depth of the nets varies. For instance the IAM-DC$_{10}^{3\times3}$ consists of 10 stages.

	IAM-DC$_{10}^{5\times5}$		IAM-DC$_{15}^{5\times5}$		IAM-DC$_{20}^{5\times5}$	
Dataset	PSNR	σ	PSNR	σ	PSNR	σ
Levin	31.495	2.064	32.073	2.233	32.132	2.236
BSD500	28.721	3.021	29.074	3.048	29.138	3.053

Table 2.2: Comparison of the experimental results for the proposed *IAM-DC*$^{5\times5}$ nets. Depicted are the average PSNR(dB) and the corresponding standard deviation σ for the data set of Levin et al. [28] and 32 image samples from the BSD500 dataset [30] convolved with the kernels of Levin. All the network filters k_{ti} are of size 5×5, however the depth of the nets varies.

to follow the gradient of the heavy tailed distribution $\log(1 + \alpha x^2)$ and the filters k_{ti} were set up by the i-th basis of the discrete cosine transform. The padding operation P is performed by replicating the image borders with $\lfloor H/2 \rfloor$ in y- and $\lfloor W/2 \rfloor$ in x-direction pixels if the blur kernel a^s is of size $H \times W$. Consequently, the truncation operator T removes these padded regions by cropping the image accordingly. Furthermore, we ensured that the intermediate images have values in the interval $[0, 1]$ by truncating those outside.

Based on this configuration the networks were trained as described in Section 2.2.1.1. The pre-training of the stage parameters θ_t involved 100 iterations of L-BFGS and the blocks, consisting of 5 stages, were trained jointly by executing another 100 L-BFGS iterates. This procedure was repeated until the final net depth was reached.

The so trained network was evaluated on the dataset of Levin et al. [28], which consists of 4 images, depicted in Figure 2.6, and 8 blur kernels, hence 32 combinations are possible. Moreover, we used 32 randomly extracted images from the validation set of the BSD500 dataset and computed blurry observations using the blur kernels of Levin and colleagues. This test set is called BSD500. Note that the images and kernels in both test sets are different than those in the training data. Each test data was constructed using the ground truth images within this dataset and the true blur kernels. All convolved images were corrupted by 1% additive Gaussian white noise to form the noisy and blurry observation f^s. The same corruption was also performed with the training data. The input to the network was given by the blurry observation $u_0^s = f^s$. Table 2.1 and 2.2 depict the average *PSNR* values and the standard deviation across both datasets.

The first column of Table 2.1 states the results of the primal-dual algorithm [8] applied

(a) **(b)**

(c) **(d)**

Figure 2.6: Ground truth sharp images of the dataset of Levin et al. [28]. (a) - (d) are images 1 to 4.

to non-blind image deconvolution with TV regularization. Despite the problem of the data dependent choice of λ, see Section 2.1.4, we used $\lambda = 500$ as it is a good compromise and perform 1000 iterations. This simple model already generates good results if we look at Figures 2.7b to 2.9b. However, the results suffer from the TV favor of piece-wise constant regions, see for instance the chin of the child in Figure 2.8b, the grass in Figure 2.7b or the shirt of the children in Figure 2.9b. The last image is also corrupted by artifacts due to inconsistencies at the border, which cannot be avoided despite the large amount of

Figure 2.7: Deconvolution results of a sample image in the BSD500 test set. (a) blurry observation with true blur kernel in upper left corner. Deconvolution results (b) primal-dual *TV* algorithm, (c) IAM-DC$_{20}^{3\times3}$ and (d) IAM-DC$_{20}^{5\times5}$.

iterations. This example is challenging, though, as kernel 4 is the largest in the dataset and the majority of its weights is concentrated at the corners.

In contrast, the proposed *IAM-DC* networks are much better suited for image deconvolution. Not only are all networks by a margin of almost 1dB better than the *TV* method in terms of *PSNR*, but they can also be computed much more efficiently and faster due to the small amount of stages. The bottom rows in Figures 2.7 to 2.9 show the outputs of the IAM-DC$_{20}^{3\times3}$ and IAM-DC$_{20}^{5\times5}$ net. Compared to the *TV* results, these methods can restore details within the images much better, see for instance the hair of the child in Figure 2.8 or the wrinkle below the left eye. However, very fine texture is considered as noise and filtered out. This effect is visible at the sand in the bottom right corner in Figure 2.9. In the same image the pattern of the children's shirt is better restored, despite the large blur, because the leopard pattern has a large enough support and contrast to be not considered as noise. Furthermore, the structure of the grass in Figure 2.7 or the barrel are better reconstructed. However, if we have a look at the top right corner in this images, we see both network outputs suffer from ringing artifacts at strong edges. Nevertheless, the outputs of the IAM-DC$_{20}^{5\times5}$ posses better *PSNR* results, which can be explained by the

Figure 2.8: Deconvolution results for image 4 and kernel 7 of Levin's dataset. (a) blurry observation with true blur kernel in upper left corner. Deconvolution results (b) primal-dual *TV* algorithm, (c) IAM-DC$_{20}^{3\times3}$ and (d) IAM-DC$_{20}^{5\times5}$.

better suppression of noise and ringing artifacts. If we take a closer look at the children's chin and compare Figure 2.8c and 2.8d, we can observe that the result of the IAM-DC$_{20}^{5\times5}$ net is much smoother. That is because the increased filter size and the larger amount of filter and influence function pairs allow the network to better describe image statistics and consequently improve its performance to suppress artifacts and noise.

Another key advantage of the proposed network is that the individual parameters can

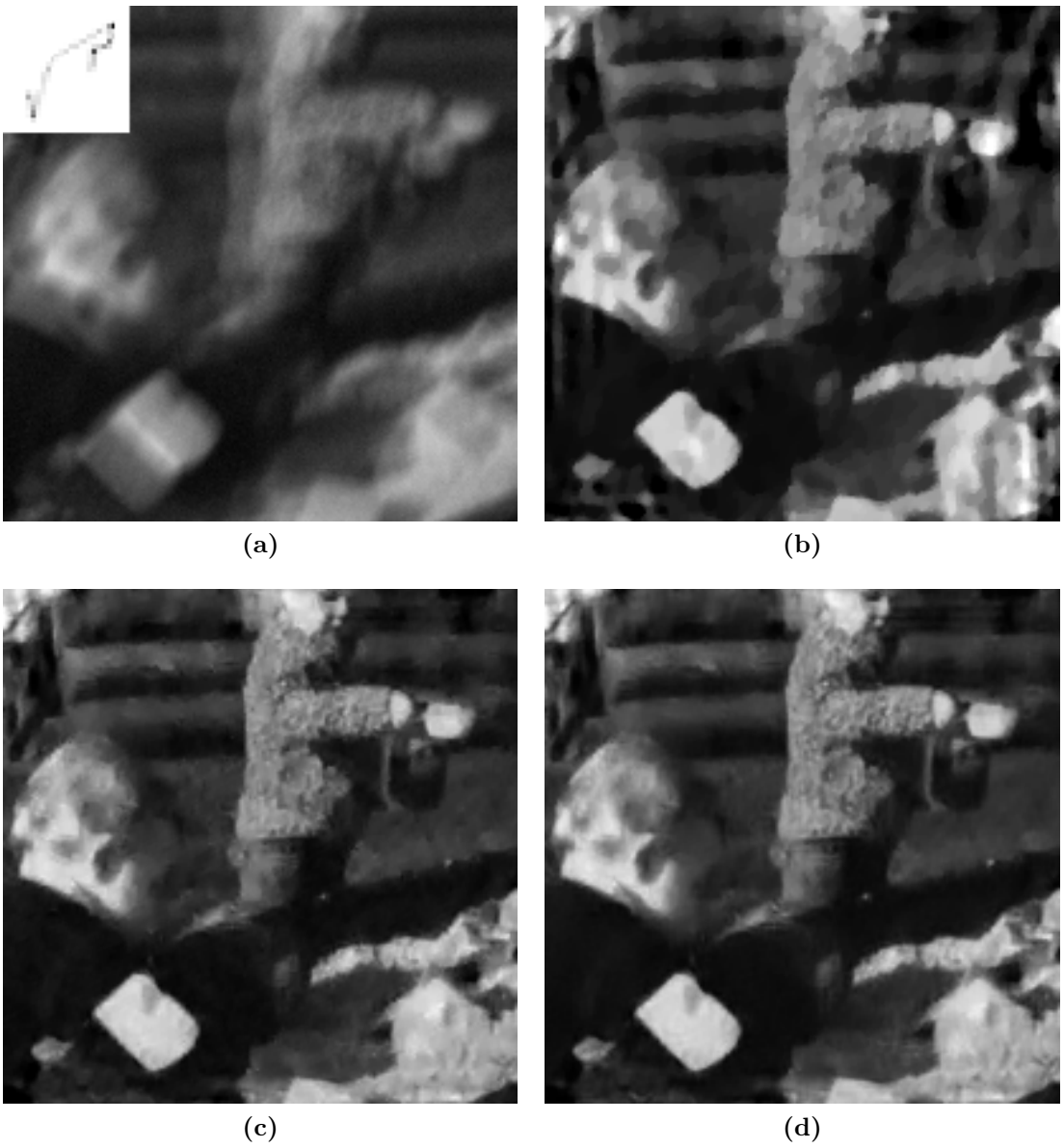


Figure 2.9: Deconvolution results for image 1 and kernel 4 of Levin's dataset. (a) blurry observation with true blur kernel in upper left corner. Deconvolution results (b) primal-dual *TV* algorithm, (c) IAM-DC$_{20}^{3\times3}$ and (d) IAM-DC$_{20}^{5\times5}$.

be simply interpreted despite the learning approach due to its close relation to energy minimization. Figure 2.10 illustrates the learned filters k_{ti} and their associated penalty functions $\rho_{ti} = \int \phi_{ti}(x)dx$ for selected stages of the IAM-DC$_{20}^{5\times5}$ net. The integration of the influence functions was approximated by its discrete counter part and performed by the MATLAB command `cumsum`$(\phi_{ti}(x))$. The penalty functions of the first stage are dominated by various forms of the Mexican hat function. Due to the close relation to

energy minimization we can interpret this circumstance such that the network assigns high costs to small and high filter responses, while, medium valued generate low costs and thus they are enhanced. Furthermore, this Mexican hat functions are mainly associated with complex kernels, hence they remove noise. However, the largest impact comes from the truncated 'convex' functions in the first and second row because of their scale. For example the first kernel-function pair in the first row penalizes large diagonal edges and the last one in that row its orthogonal pendant. Therefore, they model the sparse gradient constraint of natural images.

The behavior of the second stage is fundamentally different. As depicted in the center of Figure 2.10, much more 'concave' shaped functions are present and moreover their scale undermines their importance. The first and last kernel-function pair in the first row are related to vertical and horizontal edges due to the kernel shape. The concavity of the related influence functions enforces the regularizer to suppress small edges and enforce large ones. Thus, this pairs are responsible for reducing ringing artifacts. Also the first and last kernel-function pair in the second row support this effect. Additionally, the second pair in the third row and the last one in the fifth row repress small dark and bright spots. Within the whole network the effect of those two stages is successively repeated, as can be seen in Figure 2.11. While the odd stages remove noise and fine texture, the even stages try to restore the fine details within an image and push down ringing artifacts. Without the joint training of the network blocks this behavior would not be possible because in the greedy pre-training phase the input image cannot be altered, which limits the expressiveness.

If we take a large leap towards the end of the network, we see that the stage behavior changed once more. The kernel and penalty function pairs of stage 20 are illustrated at the bottom of Figure 2.10. This stage is dominated by truncated 'convex' functions, which assign low costs for small and medium filter responses. Therefore, small structures like fine lines and textures are reconstructed towards the end of the network. However, some functions are hard to interpret due to their irregular shape. The second last influence function in the fifth row is almost identical to the absolute function but its associated filter is not a first oder derivative, thus it does not correspond to the TV.

Over all, the proposed network performs very well if we compare the depicted images and the performance measures despite its simple structure. Moreover, its relation to energy minimization eases the interpretation which helps to understand the performed operations. Furthermore, its simple operations and structure allow an efficient implementation.

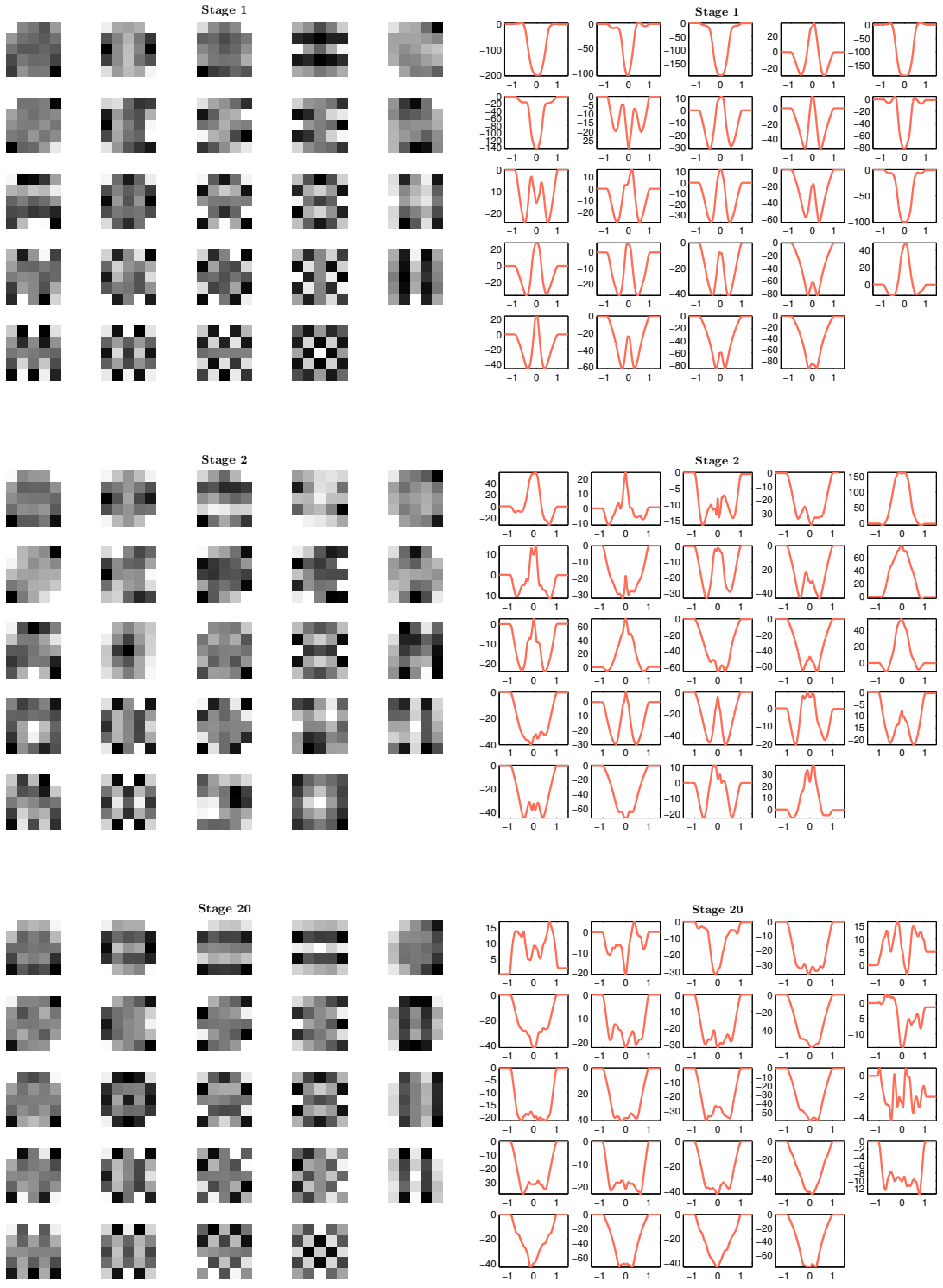


Figure 2.10: Filters k_{ti} and associated penalty functions ρ_{ti} of some stages of the IAM-DC$_{20}^{5\times5}$ net. Top row illustrates the parameters of the first stage, in the central row are those depicted of the second stage, while, the last row shows the filters and penalty functions of the final stage.

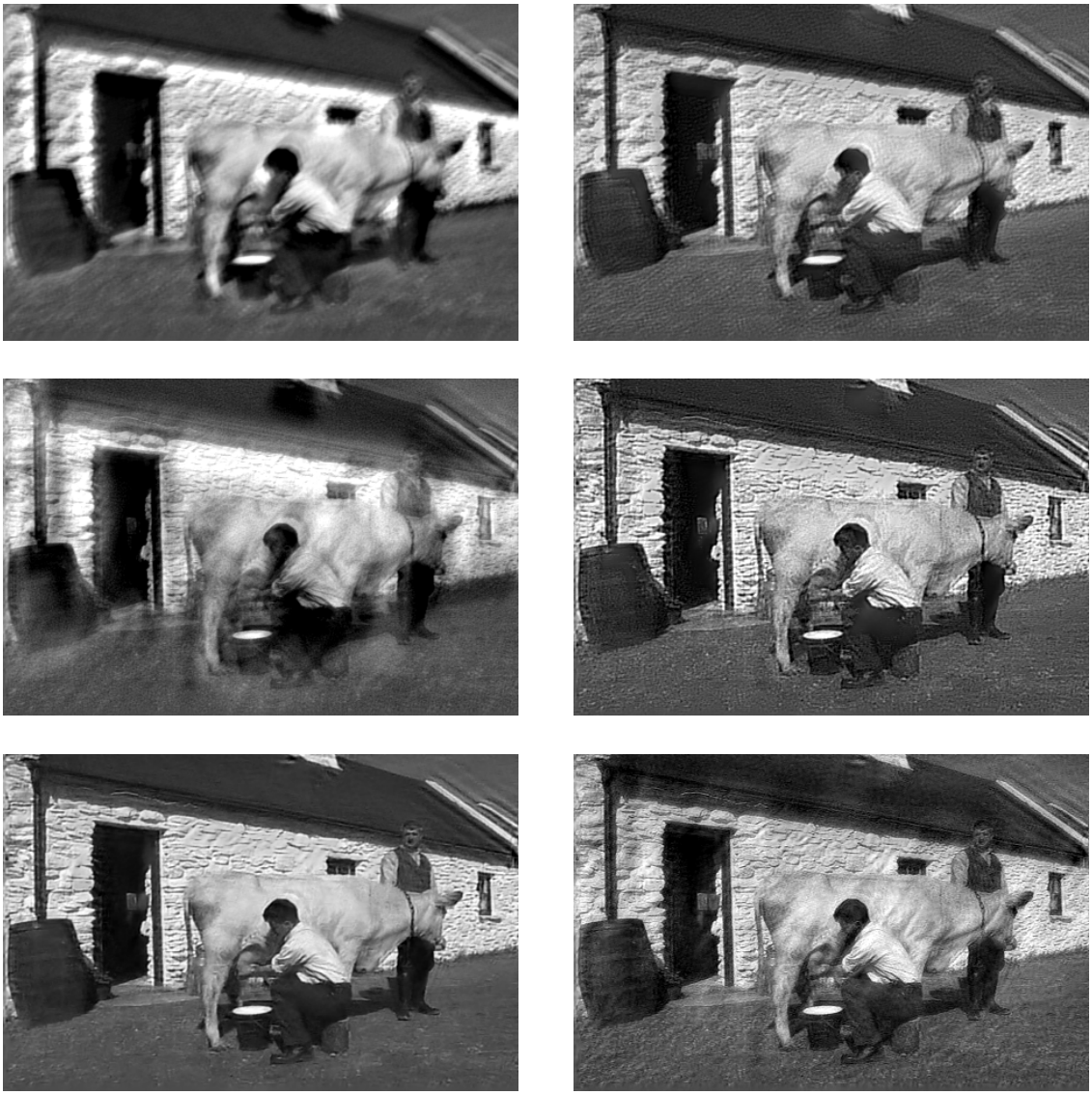


Figure 2.11: Intermediate results of the first 6 stages of the IAM-DC$_{20}^{5\times5}$ net for the image depicted in Figure 2.7. Top left image is the output of the first stage, to the left is the one of the second stage, while u_3 is below and so forth.

2.2.2 Improved IAM-DC network

The treatment of the data term within the various non-blind deconvolution methods discussed in Section 2.1.2 is almost identical. Most models use an ℓ_2-norm motivated by the Gaussian noise assumption, however, Levin et al. [25] use an ℓ_1-norm to suppress effects of large outliers. Both assumptions work, so a natural extension of the *IAM-DC* network defined in Equation (2.15) is to train this penalty function too. Moreover, we employ differently filtered versions of the data term to better recover the image at edge regions, motivated by the results of Shan et al. [42]. Furthermore, the noise of the blurry observation can be better handled. Consequently, the stage update rule changes to

$$u_t = T\left\{u_{p(t-1)} - \left(\frac{1}{N_k}\sum_{i=1}^{N_k} K_{ti}^{\mathsf{T}}\phi_{ti}(\eta K_{ti}u_{p(t-1)}) + \ldots \right.\right.$$
$$\left.\left.\frac{1}{N_d}\sum_{i=1}^{N_d} A^{\mathsf{T}}\bar{K}_{ti}^{\mathsf{T}}\bar{\phi}_{ti}\left(\bar{\eta}\bar{K}_{ti}\left(Au_{p(t-1)} - f\right)\right)\right)\right\}, \tag{2.25}$$

where $\bar{\phi}_{ti}(x)$ defines the influence function of the data term, which is associated with a filter $\bar{k}_{ti}$ being represented by its convolution matrix $\bar{K}_{ti}$. The scalars η and $\bar{\eta}$ are used to scale the argument of $\phi(x)$ and $\bar{\phi}(x)$ respectively such that they can be constructed from a common basis. By comparing both parts of this update rule, we see that they resemble each other. The sole difference in terms of structure is the argument of the influence functions. While, the regularization works just on the image u, the data term influence is based on the image formation process.

Due to the modification of the stage update rule, the parameters of a single stage have changed to

$$\theta_t = \{(c_{ti}, w_{ti})\, i = 1 \ldots N_k, (\bar{c}_{ti}, \bar{w}_{ti})\, i = 1 \ldots N_d\}.$$

The parametrization of the kernels and influence functions of the data term is done in the same way as those of the regularizer. Hence, the kernels $\bar{k}_{ti}$ are set up using a discrete cosine basis

$$\bar{k}_{ti} = B_d \frac{\bar{c}_{ti}}{||\bar{c}_{ti}||_2}, \tag{2.26}$$

whereby the basis B_d holds additionally the constant basis function. It is required to express the delta kernel, which allows the computation of the unfiltered data term $A^{\mathsf{T}}(Au_{p(t-1)} - f)$. The network structure was not effected by this modification, though. Hence, this *IAM-DC* model can be trained in the same manner as the previous one.

2.2.2.1 Gradient derivation

Since the stage parameters changed due to the modification of the stage update rule, we need to adapt the derivative $\partial u_t / \partial \theta_t$. The gradients with respect to c_{ti} and w_{ti} stay unchanged because the regularization was not changed compared to the previous model, see Equation (2.15). The parameter λ_t was neglected, as it can be included in the scale of the influence functions $\bar{\phi}_{ti}(x)$.

Data-term filters $\bar{k}_{ti}$: The gradient $\partial u_t / \partial \bar{k}_{ti}$ can be computed in a similar way as the regularizer filters beforehand because the problem structure is the same. If we derive the stage update rule (2.25) by $\bar{k}_{ti}$, we end up with

$$\frac{\partial u_t}{\partial \bar{k}_{ti}} = -\frac{1}{N_d} \frac{\partial}{\partial \bar{k}_{ti}} \left(g(\bar{k}_{ti}) * \underbrace{\bar{\phi}_{ti}(\bar{\eta} \bar{k}_{ti} * \left(A u_{p(t-1)} - f\right))}_{h(\bar{k}_{ti})} \right) A T^\mathsf{T},$$

where the function $g(x)$ is defined in Equation (2.22). As before, we can employ the chain rule to express this derivative

$$\frac{\partial u_t}{\partial \bar{k}_{ti}} = -\frac{1}{N_d} \left(\frac{\partial g}{\partial \bar{k}_{ti}} \frac{\partial u_t}{\partial g} + \frac{\partial h}{\partial \bar{k}_{ti}} \frac{\partial u_t}{\partial h} \right) A T^\mathsf{T}.$$

Furthermore, the convolution $g * h$ can be expressed by its associated matrix-vector representation $Gh \Leftrightarrow Hg$. As a result, the individual derivatives are defined by

$$\begin{aligned}
\frac{\partial u_t}{\partial g} &= H^\mathsf{T} \\
\frac{\partial g}{\partial \bar{k}_{ti}} &= R_{180}^\mathsf{T} \\
\frac{\partial u_t}{\partial h} &= G^\mathsf{T} = (\bar{K}_{ti}^\mathsf{T})^\mathsf{T} = \bar{K}_{ti} \\
\frac{\partial h}{\partial \bar{k}_{ti}} &= \bar{\eta} \left(A u_{p(t-1)} - f\right)^\mathsf{T} \operatorname{diag}\left(\bar{\phi}'_{ti}(\bar{\eta} \bar{k}_{ti} * \left(A u_{p(t-1)} - f\right))\right) = \bar{\eta} \left(A u_{p(t-1)} - f\right)^\mathsf{T} \bar{\Lambda}_{ti},
\end{aligned}$$

where $\left(A u_{p(t-1)} - f\right)$ is a convolution matrix for that $k_{ti} * \left(A u_{p(t-1)} - f\right) \Leftrightarrow \left(A u_{p(t-1)} - f\right) k_{ti}$ is equivalent and $\bar{\phi}'_{ti}(x)$ denotes the derivative of the data term influence function $\bar{\phi}(x)$. Consequently, the gradient is given by

$$\frac{\partial u_t}{\partial \bar{k}_{ti}} = -\frac{1}{N_d} \left(R_{180}^\mathsf{T} H^\mathsf{T} + \bar{\eta} \left(A u_{p(t-1)} - f\right)^\mathsf{T} \bar{\Lambda}_{ti} \bar{K}_{ti} \right) A T^\mathsf{T}. \tag{2.27}$$

Due to the parametrization of the filters $\bar{k}_{ti}$, see Equation (2.26), this derivative needs to be further refined.

$$\frac{\partial \bar{k}_{ti}}{\partial \bar{c}_{ti}} = \frac{1}{||\bar{c}_{ti}||_2} \left(I - \frac{\bar{c}_{ti}\bar{c}_{ti}^\mathsf{T}}{\bar{c}_{ti}^\mathsf{T}\bar{c}_{ti}} \right) B_d^\mathsf{T}$$

Now we can compute the required derivative for training by putting all the pieces together, which yields

$$\frac{\partial \ell}{\partial \bar{c}_{ti}} = -\frac{1}{N_d \, ||\bar{c}_{ti}||_2} \left(I - \frac{\bar{c}_{ti}\bar{c}_{ti}^\mathsf{T}}{\bar{c}_{ti}^\mathsf{T}\bar{c}_{ti}} \right) B_d^\mathsf{T} \left(R_{180}^\mathsf{T} H^\mathsf{T} + \bar{\eta} \left(A u_{p(t-1)} - f \right)^\mathsf{T} \bar{\Lambda}_{ti} \bar{K}_{ti} \right) A T^\mathsf{T} e_t.$$

Data-term influence functions $\bar{\phi}_{ti}$: Also this influence functions are constructed using Gaussian radial basis functions and a weight vector $\bar{w}_{ti}$

$$\bar{\phi}_{ti}(x) = \Phi(x)\bar{w}_{ti}.$$

If we insert this circumstance into the stage update rule (2.25) and compute the derivative with respect to $\bar{w}_{ti}$, we get

$$\begin{aligned} \frac{\partial u_t}{\partial \bar{w}_{ti}} &= -\frac{1}{N_d} \frac{\partial}{\partial \bar{w}_{ti}} \left(A^\mathsf{T} \bar{K}_{ti}^\mathsf{T} \Phi \left(\bar{\eta} \bar{K}_{ti} (A u_{p(t-1)} - f) \right) \bar{w}_{ti} \right) T^\mathsf{T} \\ &= -\frac{1}{N_d} \Phi^\mathsf{T} \left(\bar{\eta} \bar{K}_{ti} (A u_{p(t-1)} - f) \right) \bar{K}_{ti} A T^\mathsf{T} \end{aligned}$$

Finally, we can state the gradient of the objective function in case of greedy pre-training

$$\frac{\partial \ell}{\partial \bar{w}_{ti}} = -\frac{1}{N_d} \Phi^\mathsf{T} \left(\bar{\eta} \bar{K}_{ti} (A u_{p(t-1)} - f) \right) \bar{K}_{ti} A T^\mathsf{T} e_t. \tag{2.28}$$

This concludes the derivation of the gradients necessary for the greedy pre-training phase. In addition, the modification of the stage update rule also effects the joint training of the network.

Joint training: The basic principle of the joint training phase stays the same, as the network structure was not effected by the modifications of the stage update rule. However, the error propagation within this framework relies on the gradient $\partial u_{(t+1)}/\partial u_t$, which is indeed effected by any change of the update rule. Thus, we need to recompute it. As its structure is almost identical to the regularization it can be derived using the chain rule and is given by

$$\frac{\partial u_{(t+1)}}{\partial u_t} = P^\mathsf{T} \left\{ I - \left(\frac{1}{N_k} \sum_{i=1}^{N_k} \eta K_{(t+1)i}^\mathsf{T} \Lambda_{(t+1)i} K_{(t+1)i} + \frac{1}{N_d} \sum_{i=1}^{N_d} \bar{\eta} A^\mathsf{T} \bar{K}_{(t+1)i}^\mathsf{T} \bar{\Lambda}_{(t+1)i} \bar{K}_{(t+1)i} A \right) \right\} T^\mathsf{T},$$

where $\Lambda_{(t+1)i}$ and $\bar{\Lambda}_{(t+1)i}$ are diagonal matrices which hold the derivatives of the influence functions $\phi'_{(t+1)i}(\eta u_{pt} * k_{(t+1)i})$ and $\bar{\phi}'_{(t+1)i}(\bar{\eta}(u_{pt} * a - f) * \bar{k}_{(t+1)i})$ respectively.

	$\text{I}^2\text{AM-DC}_{10}^{3\times3}$		$\text{I}^2\text{AM-DC}_{15}^{3\times3}$		$\text{I}^2\text{AM-DC}_{20}^{3\times3}$	
Dataset	PSNR	σ	PSNR	σ	PSNR	σ
Levin	31.498	1.765	31.844	1.738	31.921	1.754
BSD500	28.790	2.939	28.960	2.950	29.004	2.953

Table 2.3: Comparison of the experimental results for the proposed Improved Iteratively Adapted Minimization for non-blind Deconvolution $(\text{I}^2\text{AM-DC})^{3\times3}$ nets. Depicted are the average PSNR(dB) and the corresponding standard deviation σ for the data set of Levin et al. [28] and 32 image samples from the BSD500 dataset [30] convolved with the kernels of Levin. All the network filters k_{ti} and $\bar{k}_{ti}$ are of size 3×3, however the depth of the nets varies, e.g. the $\text{IAM-DC}_{10}^{3\times3}$ consists of 10 stages.

	$\text{I}^2\text{AM-DC}_{10}^{5\times5}$		$\text{I}^2\text{AM-DC}_{15}^{5\times5}$		$\text{I}^2\text{AM-DC}_{20}^{5\times5}$	
Dataset	PSNR	σ	PSNR	σ	PSNR	σ
Levin	31.917	1.767	32.402	1.715	32.594	1.680
BSD500	28.918	2.936	29.137	2.984	29.229	2.996

Table 2.4: Comparison of the experimental results for the proposed $\textit{I}^2\textit{AM-DC}^{5\times5}$ nets. Depicted are the average PSNR(dB) and the corresponding standard deviation σ for the data set of Levin et al. [28] and 32 image samples from the BSD500 dataset [30] convolved with the kernels of Levin. All the network filters k_{ti} and $\bar{k}_{ti}$ are of size 5×5, however the depth of the nets varies.

Eventually, the network defined by the stage update rule (2.25) can be trained using the scheme developed in Section 2.2.1.1 and the updated gradients $\partial u_t / \partial \theta_t$ and $\partial u_{(t+1)} / \partial u_t$.

2.2.2.2 Evaluation

The improved network models were trained with the same training set as the *IAM-DC* nets, consisting of 200 training samples of the form $(u_0^s, f^s, u_{gt}^s, a^s)$, in order to ease comparison. Moreover, the same training strategy was employed. First, each stage is greedily pre-trained by 100 L-BFGS iterations until all stages within a block are done (5 stages are one block). Then the entire block is trained jointly, afterwards all its parameters are fixed. The next block is trained in the same fashion using the output of the first one as constant input. This approach is repeated until the final network depth T is reached. Furthermore, the network parameters such as Gaussian radial basis functions, η and the initialization of the regularization parameters were unchanged. The kernels of the data term $\bar{k}_{ti}$ were also initialized using the i-th basis component of the according discrete cosine transform basis. However, the delta kernel was used instead of the constant basis to express the classic data term loss. In contrast to the regularization, the influence functions of the data term were initialized as $\bar{\phi}_{ti}(x) = x$, which is the derivative of $1/2x^2$. The corresponding argument parameter $\bar{\eta}$ was set to 1.

The various $\textit{I}^2\textit{AM-DC}$ networks were then evaluated using the 32 images of the Levin et al. [28] dataset and the same BSD500 test set as before. Their resulting average *PSNR* and the according standard deviation for both test sets are illustrated in Table 2.3 and 2.4.

Figure 2.12: Deconvolution results of a random image sample from the BSD500 test set. (a) blurry observation with true blur kernel in upper left corner. Deconvolution results (b) IAM-DC$_{20}^{5\times5}$, (c) I^{2}AM-DC$_{20}^{3\times3}$ and (d) I^{2}AM-DC$_{20}^{5\times5}$.

If we compare the results of the I^{2}AM-DC$_{10}^{3\times3}$ to those of IAM-DC$_{10}^{3\times3}$ for the test set of Levin, we see that the average *PSNR* is 0.67dB larger. However, the mean *PSNR* of the I^{2}AM-DC$_{15}^{3\times3}$ is just 0.51dB better than those of the IAM-DC$_{15}^{3\times3}$ and furthermore the difference after 20 stages is just 0.38dB. So, the deeper the network is, the less important is the training of the data term parameters. Interestingly though, we can observe the opposite effect when considering the I^{2}AM-DC$^{5\times5}$ nets. For 10 stages the margin is 0.42dB, after 15 stages it is a bit more than 0.33dB and for the 20 stage networks the *PSNR* of the improved net is 0.46dB larger. Hence, the training of the data term penalty functions does improve the results quite strongly. If we look at the resulting images in Figures 2.12 to 2.14, the visual difference is rather small due to the high *PSNR* values. The superior results of the improved networks manifest in a better recovering of details within images and suppression of ringing artifacts.

Again, the 5×5 kernels improve the denoising performance of the network, which can be seen when comparing the resulting images in Figures 2.12 to 2.14. Moreover, a difference of 0.67dB of the 20 stages improved nets validates this observation.

Also the parameters of the improved nets can be analyzed in the same fashion as in

Figure 2.13: Deconvolution results for image 4 and kernel 7 of Levin's dataset. (a) blurry observation with true blur kernel in upper left corner. Deconvolution results (b) IAM-DC$_{20}^{5\times5}$, (c) I^{2}AM-DC$_{20}^{3\times3}$ and (d) I^{2}AM-DC$_{20}^{5\times5}$.

the evaluation of the *IAM-DC* networks. Figure 2.15 illustrates the filters k_{ti} and penalty functions ρ_{ti} of the regularization for selected stages of the I^{2}AM-DC$_{20}^{5\times5}$ net. At the top the parameters of the first stage are depicted. Although the filters resemble those of the IAM-DC$_{20}^{5\times5}$, the penalty functions are quite different. The first stage functions ρ_{1i} of the IAM-DC$_{20}^{5\times5}$ net were dominated by Mexican hat variants, whereas, in the I^{2}AM-DC$_{20}^{5\times5}$ network this influence is rather limited. Most penalty functions are shaped like a tailed

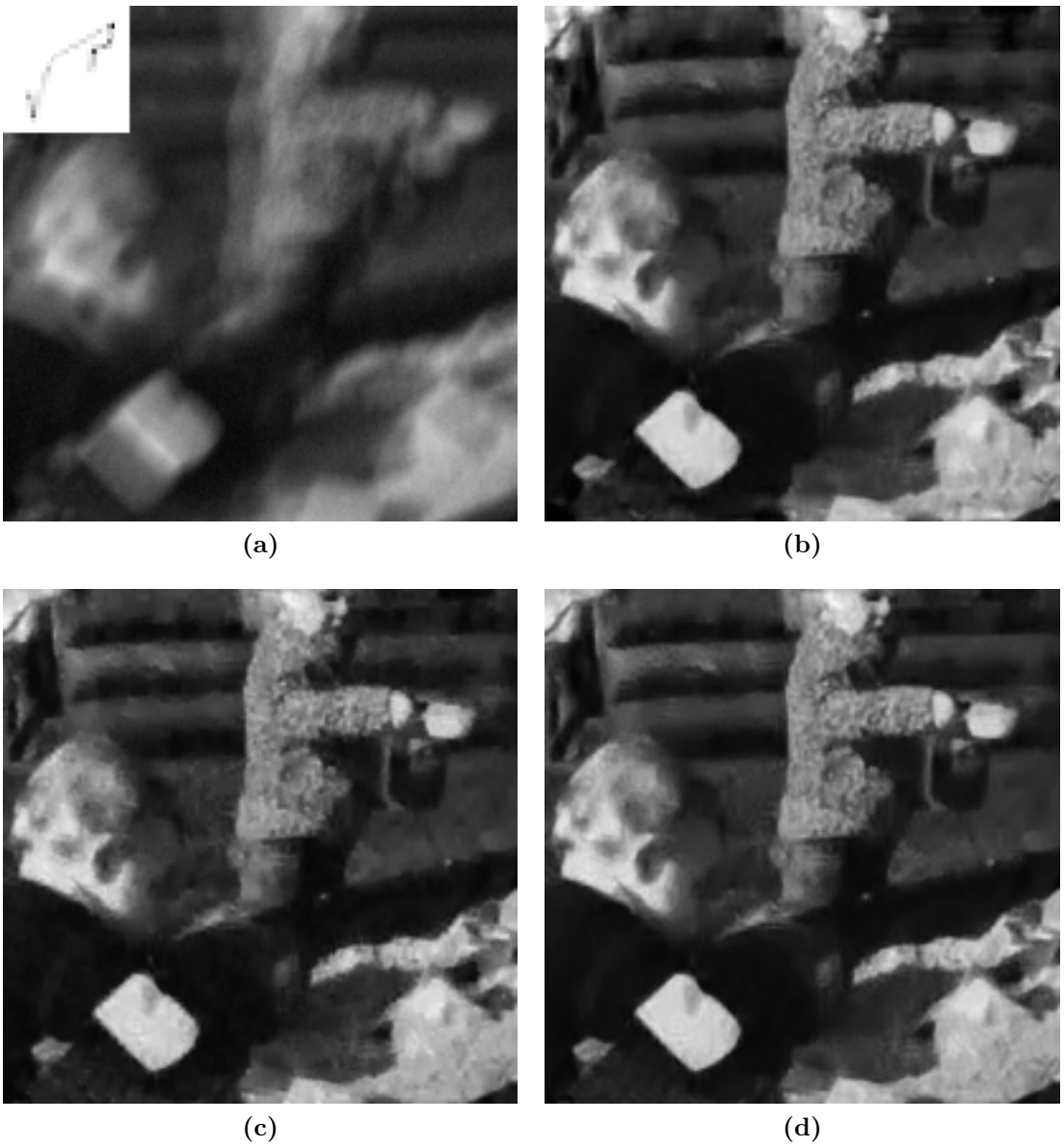


Figure 2.14: Deconvolution results for image 1 and kernel 4 of Levin's dataset. (a) blurry observation with true blur kernel in upper left corner. Deconvolution results (b) IAM-DC$_{20}^{5\times5}$, (c) I^{2}AM-DC$_{20}^{3\times3}$ and (d) I^{2}AM-DC$_{20}^{5\times5}$.

distribution with a little nibble in the center. The first and last kernel-function pair in the first row represent the sparse gradient constraint of natural images. Overall, this stage seems to have a smoothing character.

The second stage, however, seems to remove artifacts such as ringing because the first and last gradient related functions in the first row have 'concave' shapes. Therefore, they assign high cost to small edges while favoring large ones. In contrast, the first filter in the

fourth and the second in the last row together with their penalty functions express once more the sparse gradient constraint of natural images. It seems as if those two effects cancel each other out but the scale of the concave functions is much larger. Thus, the second stage mainly reduces artifacts. This behavior is also shown in Figure 2.17, which depicts the intermediate images of the first six stages of the $\text{I}^2\text{AM-DC}_{20}^{5\times5}$ net. While the result of the first stage suffers from severe ringing artifacts, the second stage reduced their intensity to a large extent. However, along with the increasing amount of details, more finer ringing artifacts appear.

If we compare the filters and functions of the final stage of the $\text{I}^2\text{AM-DC}_{20}^{5\times5}$ net to those of the $\text{IAM-DC}_{20}^{5\times5}$, we see that they are rather similar. Strong edges are enhanced and small details are reconstructed. The irregularly shaped functions have little influence due to their scale.

The data term filters $\bar{k}_{ti}$ and their associated penalty functions $\bar{\rho}_{ti}$ are depicted in Figure 2.16 for the first, second and last stage of the $\text{I}^2\text{AM-DC}_{20}^{5\times5}$ net, to highlight their influence. The penalty functions were initialized by the quadratic function $1/2x^2$. However, in the first stage most of the functions are either shaped like the absolute function or heavy tailed. So, the training did change them quite a lot. This circumstances undermine the application of the ℓ_1-norm for the data term. Furthermore, the filters have a more derivative based characteristic than those of the discrete cosine basis.

However, with increasing net depth the domination of the absolute and heavy tailed functions deceases. As the intermediate images of the stages get closer and closer to the true sharp image, see Figure 2.17, just the noise remains in the data term $u_t * a - f \approx n$. Due to the Gaussian nature of the noise, the optimal penalty function is the quadratic one. Consequently, most of the penalty functions of the deeper stages preserve their quadratic shape.

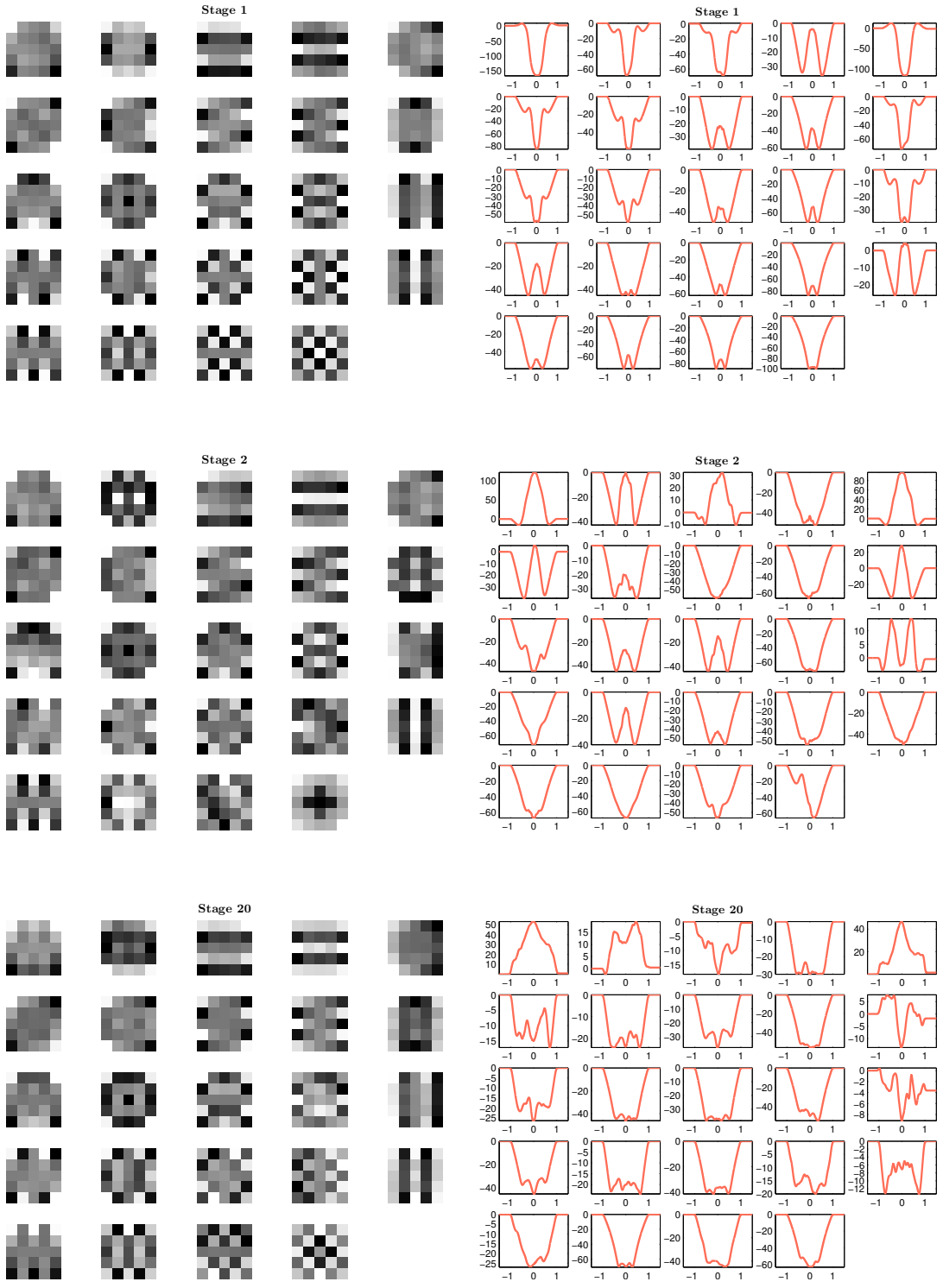


Figure 2.15: Filters k_{ti} and associated penalty functions ρ_{ti} of some stages of the $\text{I}^2\text{AM-DC}_{20}^{5\times5}$ net. Top row illustrates the parameters of the first stage, in the central row are those depicted of the second stage, while, the last row shows the filters and penalty functions of the final stage.

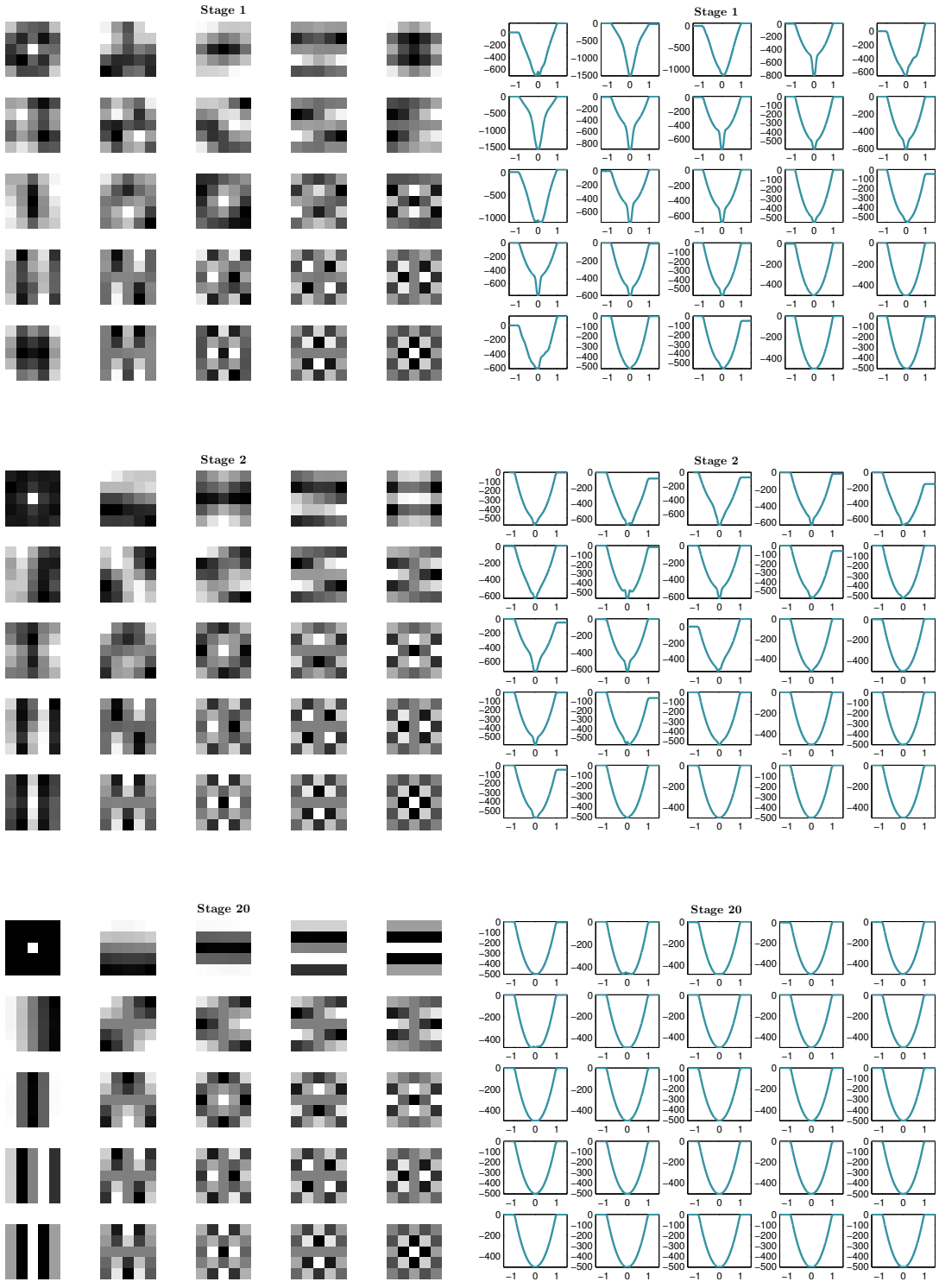


Figure 2.16: Filters $\bar{k}_{ti}$ and associated penalty functions $\bar{\rho}_{ti}$ of some stages of the $\text{I}^2\text{AM-DC}_{20}^{5\times5}$ net. Top row illustrates the parameters of the first stage, in the central row are those depicted of the second stage, while, the last row shows the filters and penalty functions of the final stage.

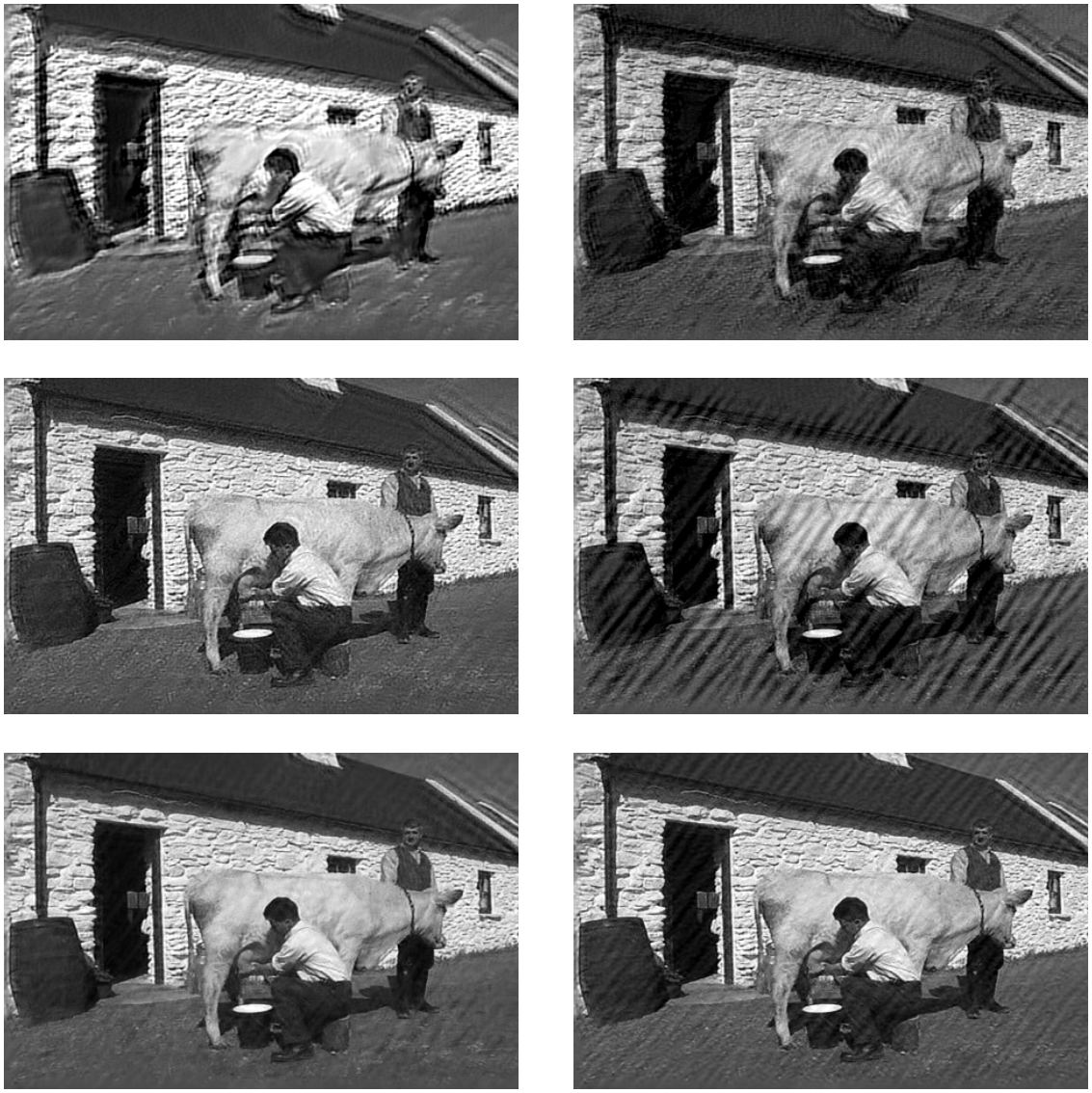


Figure 2.17: Intermediate results of the first 6 stages of the $\text{I}^2\text{AM-DC}_{20}^{5\times5}$ net for the image depicted in Figure 2.12. Top left image is the output of the first stage, to the left is the one of the second stage, while u_3 is below and so forth.

2.3 Conclusion

We proposed a novel iteratively adapted minimization approach for non-blind image deconvolution motivated by the denoising method of Chen et al. [13]. The models are trained by a greedy pre-training followed by a joint refinement of the parameters, which enables the full expressive power of the network blocks. Moreover, we demonstrated the flexibility of the method and evaluated the resulting models using different images *and* kernels. The obtained *PSNR* scores are up tp 3.23dB better than those of the simple *TV* regularized model on the test set of Levin and almost 2dB larger on the BSD500 test set. The resulting deconvolved images posses sharp edges and less noise, undermining the power of the proposed networks. The evaluation of the introduced networks pointed out that ringing artifacts can be better suppressed by using larger filters for the regularizer k_{ti} and the data term $\bar{k}_{ti}$. Furthermore, the improved data term also helps to avoid those artifacts.

Due to their close relation to the field of energy minimization, a profound analysis of the individual parameters is possible, as demonstrated in the sections above. Also all the intermediate images can be interpreted, since they all live in the same space as the input and output images. Moreover, the networks develop compelling regularizer, which can be employed in other problems. Additionally, the gained knowledge helps to better understand good deconvolution strategies and improves the insight into the problem.

The demonstrated experiments are just a small subset of the already performed ones and many still have to be done. For instance, training a network that shares the parameters across all stages and so forth. Moreover, the influence of even larger filters than 5×5 has to be investigated.

3

Learning variational models for blind image deconvolution

Contents

Blind image deconvolution is currently a hot topic in computer vision and image processing communities due to its fundamental application in image deblurring. The overall goal is to find an efficient and accurate algorithm that overcomes the strong ill-posedness and the non-uniqueness of the solution (u, a), as we have demonstrated in Chapter 1.

In the following sections we highlight the problem of blind image deconvolution in the field of energy minimization in more detail, discuss recently proposed algorithms that overcome those and derive our method.

3.1 Blind image deconvolution in energy minimization

In general, the blind image deconvolution problem is characterized as:

> Given a blurry and noisy observation f of a scene, estimate the sharp image u and the associated unknown blur kernel a.

A variational formulation of this problem was derived in Chapter 1, see Equation (1.28). If we consider the argument of this minimization problem as an energy, we can interpret the Maximum A Posteriori (MAP) motivated derivation of the blind deconvolution problem as energy minimization.

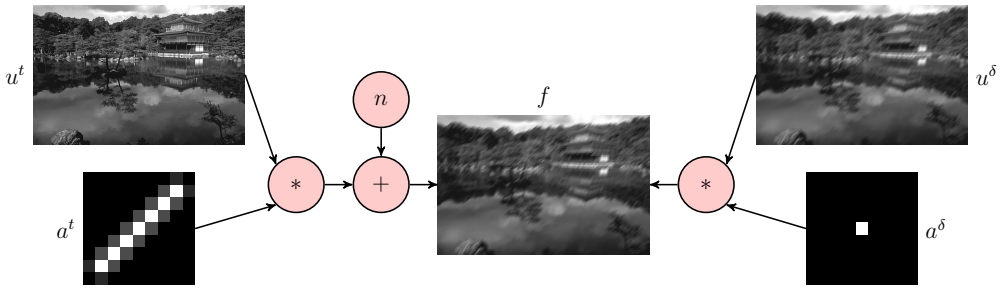


Figure 3.1: Demonstration of the underlying problem of the blind deconvolution energy minimization. Both image-kernel pairs (u^t, a^t) and (u^δ, a^δ) generate the same blurry observation f.

3.1.1 Energy formulation

By replacing the prior of (1.28) with a general regularizer on the image $R(u)$, we get

$$(u^*, a^*) = \arg\min_{u,a} E(u, a) = R(u) + \frac{\lambda}{2} ||a * u - f||_2^2 + \delta_\Delta(a), \tag{3.1}$$

where λ is a weighting parameter balancing between data-fitting and regularization and $\delta_\Delta(a)$ is the indicator function of the unit simplex (1.21). This is the basic formulation of the blind image deconvolution problem which we will further build on.

3.1.2 Energy minimization issue

An intuitive interpretation of Equation (3.1) is that one searches for the image-kernel pair (u, a) which has minimal energy $E(u, a)$. Of course, the obvious choice for the minimizing pair is (u^t, a^t), which is the true image and the true blur kernel. However, it turns out the actual minimizer of Equation (3.1) strongly depends on the applied regularizer, implying that (u^t, a^t) is *not* always the correct solution of problem (3.1).

To demonstrate this behavior let us consider two hypotheses for the minimizer of (3.1), which are the true solution (u^t, a^t) and the delta pair (u^δ, a^δ), both are depicted in Figure 3.1. The true solution (u^t, a^t) consists of the true unblurry noise-free image and the actual blur kernel that was used to generate the noise-free blurry image, while, the delta solution (u^δ, a^δ) contains the blurry and noisy observation and the delta kernel. If one convolves an image u with the delta kernel a^δ, the resulting image is exactly the same as the input to the convolution operation, thus it satisfies

$$u * a^\delta = u.$$

As we demonstrated in Chapter 1, both pairs generate the same result f, assuming that the noise n added to the true solution is the *same* as the one used during the image

formation process. If we plug the first pair into the energy $E(u, a)$, we get

$$E(u^t, a^t) = R(u^t) + \frac{\lambda}{2} \left|\left|u^t * a^t - f\right|\right|_2^2 + \delta_\Delta(a^t).$$

If we replace f with the mathematical formulation of the image formation process (1.4), we end up with

$$\begin{aligned} E(u^t, a^t) &= R(u^t) + \frac{\lambda}{2} \left|\left|u^t * a^t - (u^t * a^t + n)\right|\right|_2^2 + \underbrace{\delta_\Delta(a^t)}_{0} \\ &= R(u^t) + \frac{\lambda}{2} ||n||_2^2 . \end{aligned}$$

Thus, the energy is solely determined by the ℓ_2-norm of the noise and the regularization. The same derivations performed for the delta pair (u^δ, a^δ) yields

$$\begin{aligned} E(u^\delta, a^\delta) &= R(u^\delta) + \frac{\lambda}{2} \left|\left|u^\delta * a^\delta - f\right|\right|_2^2 + \underbrace{\delta_\Delta(a^\delta)}_{0} \\ &= R(u^\delta). \end{aligned}$$

Note that both kernels a^t and a^δ live on the unit simplex, hence the indicator function is zero. As we can see, both energy levels are mainly defined by the regularization. Since we are interested in computing the true solution (u^t, a^t) based on the observed image f, we have to ensure that

$$\begin{aligned} E(u^t, a^t) &< E(u^\delta, a^\delta) \\ R(u^t) + \frac{\lambda}{2} ||n||_2^2 &< R(u^\delta), \end{aligned}$$

otherwise we can never expect to end up with the true solution when minimizing the energy $E(u, a)$. It we further assume that the noise is small compared to the regularization energies, we end up with

$$\frac{R(u^t)}{R(u^\delta)} < 1. \tag{3.2}$$

Consequently, a suitable regularization has a lower energy for the true image u^t than for the blurry observation u^δ.

Despite the simplicity of inequality (3.2), not a single regularizer, which was introduced in Chapter 2, fulfills it. To demonstrate this circumstance, 100 pictures u^t were randomly selected from the BSD500 dataset [30]. These images were then convolved with a kernel of the Schelten et al. [39] database in order to compute u^δ, the blurry observation. For each sharp image u^t and its blurry pendant u^δ the regularization energy for a variety of typical image priors was computed, see Table 3.1. Additionally, the ratio of both regularizer

	$\|\|\nabla u\|\|_2^2$	$\|\|\nabla u\|\|_{2,1}$	$R_L(\nabla u)$	$R_{\log}(u)$	$R_{\text{IAM-DC}_{20}5\times5}(u)$
$R(u^t)$	1.37	7.14	16.94	1635	−14948
$R(u^\delta)$	0.068	1.87	6.08	1457	−15186
$\frac{R(u^t)}{R(u^\delta)}$	18.36	3.69	2.71	1.12	1.02

Table 3.1: Mean regularization energies divided by 10^3 for 100 randomly sampled images of the BSD500 dataset [30]. The regularizer are from left to right: Tikhonov, Total Variation (TV), spare gradient regularization of Levin et al. [25], $R_{\log} = \log(1 + \alpha(|\nabla_x u|^2 + |\nabla_y u|^2))$ with $\alpha = 10^9$ and the regularizer related to the first stage of the IAM-DC$_{20}^{5\times5}$ network. Note that the associated energies of the IAM-DC regularization are negative since the penalty functions ρ_{ti} are defined by $\phi_{ti}(x) = \int \phi_{ti}(x)dx$. Hence, an arbitrary constant can be added to each penalty function.

energies according to inequality (3.2) was computed. If a prior was suitable for blind-image deconvolution in energy minimization, the ratio should be less than 1. However, all these ratios are larger than 1, see the last row in Table 3.1. But why do all these regularizers fail? Most of them are designed to exploit the fact that natural images have sparse gradients. This physical property is very important in case of image denoising. Since true sparsity (ℓ_0-norm) is rather hard to optimize, approximations such as ℓ_1-norm or heavy-tailed distributions are used. The drawback of this penalty functions is that they prefer smaller gradients, see Figure 2.2. In other words, the energy assigned to smaller gradient values is lower than the one for larger gradients. Moreover, the gradients of the blurry image u^δ are always smaller than those of the associated true sharp one u^t due to the low-pass filtering nature of blurring. As a result, the regularization energy of a blurry image u^δ is smaller than the one of the sharp true image u^t for simple gradient based regularizer, thus they favor the delta solution. However, the logarithmic regularizer $R_{\log}(u)$ and the one of the learned deconvolution network are already very close to 1. For piece-wise constant images the logarithmic regularization works quite well, as can be seen in Figure 3.2. The logarithmic regularizer is very narrow, hence it is a very close approximation to the Potts model, which basically counts the intensity changes within an image. The blurry observation $f = u * a$, depicted in Figure 3.2b, has more different grayvalues than the sharp image due to the blurring. Consequently, its regularization energy is much higher than the one of the sharp image and the blind deconvolution energy minimization algorithms work. The resulting image, shown in Figure 3.2c, has sharper edges than the original one because of the heavy tailed prior. Natural images posses a large variant of grayvalues, though. Therefore this idea cannot be mapped to natural images if details have to be recovered.

Additionally, the modeling of the first order gradient statistics is not sufficient to characterize natural images, as the following experiment demonstrates. In Chapter 1 we computed the statistics of the first order x-directional gradient of two images. The x-directional gradient of the second image is depicted in Figure 3.3b. The corresponding negative logarithmic Probability Density Function (PDF) is plotted below in black. Since we can estimate this probability distribution based on the samples within an image, also

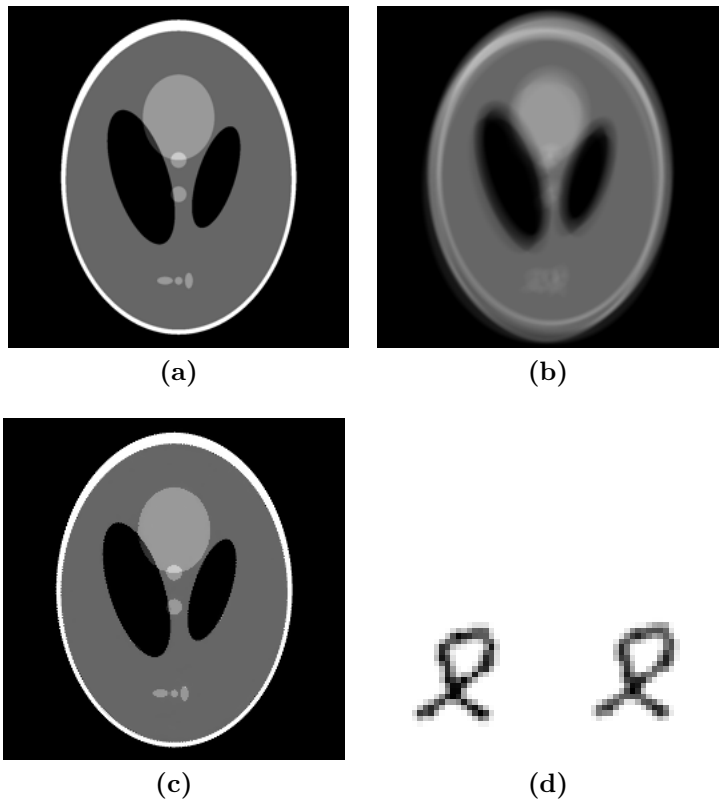


Figure 3.2: Blind deconvolution demonstration of a piece-wise constant image. (a) true sharp Shepp-Logan phantom. (b) blurry observation $f = u * a$. It was generated using the true kernel, which is depicted on the left in (d). The resulting image of the blind deconvolution is depicted in (c) and the estimated kernel on the right in (d). The deconvolution operation used a heavy tailed prior of the form $R_{\log}(u) = \log(1 + \alpha((\nabla_x u)^2 + (\nabla_y u)^2))$ with $\alpha = 10^4$.

random samples can be drawn from it. The image depicted in Figure 3.3d was exactly generated by drawing random samples from this distribution. Obviously it does not look as if it corresponds to a natural image. However, the majority of its values are close to zero (grayish) and just some larger gradients (white and black dots) are present. Furthermore, the estimated *PDF* of this image, red line in Figure 3.3c, is almost identical to the one of the naturally looking gradient image. Thus, also their regularization energies are almost the same. As a result, this first order gradient related prior models are not suitable for distinguishing between artificial and natural images. Consequently, higher order models are necessary to better describe natural image structures.

Despite this major problem of blind image deconvolution in the field of energy minimization, there exits some methods that deliver good results by exploiting other properties, as we will see in the next section.

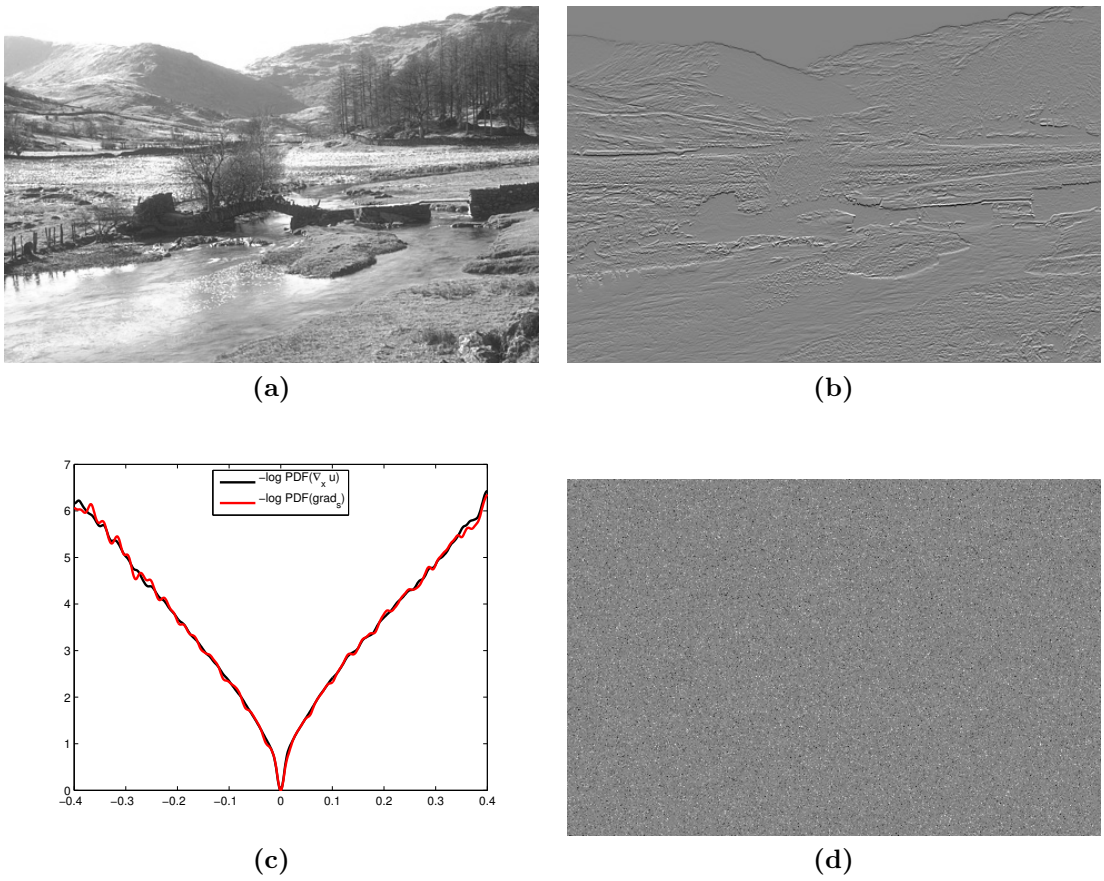


Figure 3.3: Demonstration of the gradient distribution of a natural image and an artificial sample. (b) Illustration of the x-directional gradient of image (a). It is computed by convolving the image with the filter $f_x = \begin{pmatrix} 1 & -1 \end{pmatrix}$. (d) Random samples from the distribution of $\nabla_x u$. (c) Illustration of the two *PDF*s of image (b) and (d). The black one is computed from the natural image, while, the red one corresponds to the gradient image consisting of random samples.

3.2 Recent blind image deconvolution methods

Blind image deconvolution is a fundamental problem in image restoration, thus it has been extensively studied in the past. A broad overview of recently proposed methods can be found in [10, 26, 46]. In this section we provide an outline of blind deconvolution methods influencing our approach. Therefore, they have been grouped according to their fundamental kernel/image restoration procedures.

3.2.1 Blur kernel estimation

As pointed out by Levin et al. [28], the naive *MAP* joint estimation of the blur kernel and the true image favors the trivial solution, defined as the delta kernel and the blurry image, if typical sparse image priors are used. That is because these priors prefer solutions with

small gradients, hence blurred images, as we have demonstrated in the previous section. As a consequence, many recent methods [15, 21, 22, 24, 27, 28, 43, 47] circumvent this problem by first estimating the blur kernel and then performing a non-blind deconvolution using the estimate. This lead to a paradigm change in blind-deconvolution: From prior selection to better blur kernel estimation.

3.2.1.1 MAP_a estimation

Based on information theory, Levin et al. [28] state that large enough observed images provide a sufficient amount of cues to accurately estimate the underlying blur kernel using a *MAP* scheme because the number of measurements (pixel in the observed blurry image) is usually much larger than the number of unknowns (blur kernel elements). The MAP_a estimation is then defined as

$$a^* = \arg\max_a p(a|f), \tag{3.3}$$

where $p(a|f)$ is the a posteriori probability distribution of the kernel a given the blurry observation f, which can be computed by marginalizing over the joint distribution

$$p(a|f) = \int p(u, a|f)du.$$

However, this marginalization is computationally infeasible because it requires all possible u explanations. Therefore, just a few algorithms [15, 27] tackle this problem, although Levin et al. [28] showed that $p(a|f)$ is maximized by the true kernel due to the strong asymmetry between image and blur kernel size if a local prior is used.

A good method to compute an approximate solution of (3.3) was proposed by Levin et al. [27]. They adopt an Expectation Maximization (EM) approach to optimize the MAP_a estimate. In the expectation step a non-blind deconvolution problem is solved yielding the mean image given a blur kernel and the associated covariance around it. The maximization step then uses the mean image and the covariance to estimate the true kernel given the blurry observation. This iterative scheme is very similar to the $\text{MAP}_{u,a}$ estimations, which first set the kernel constant and estimate the best image and then fix it and determine the best blur kernel. In contrast, this algorithm embeds the covariance around the image estimate when computing the best kernel, which makes the kernel estimation robust. At first glance the computational complexity of computing the marginalization over the joint distribution $p(u, a|f)$ is avoided, however, it was transfered to computing the covariance matrix around the image estimates. This can be done efficiently, though, by considering a suitable estimate of it. Levin and colleagues use a diagonal approximation, which can be computed fast. Thus, they embed a mixed strategy. While the blur kernel is based on the *MAP* estimate, the mean image is defined as the mean squared estimate given the blurry image and the current blur kernel. Their resulting robust kernel and also the mean image

estimate can then be further used in a non-blind deconvolution step.

3.2.1.2 Edge based kernel estimation

To estimate the blur kernel more robust, many recent blind-deconvolution algorithms concentrate on image regions where suitable edges are located. A very popular edge based kernel estimator was proposed by Xu and Jia [47], which consists of two phases. In the first phase an accurate initial blur kernel is estimated by carefully selecting sharp edges that guide the kernel estimation, which is followed by a coarse image restoration. These steps are repeated on each layer of a multi-scale representation of the problem for a certain amount of iterations. The interesting point in their approach is that they do not use all edges in the images because they pointed out that those of objects having a smaller scale than the kernel could flaw kernel estimation. Therefore, they introduced the r-metric which is defined as

$$r(x) = \frac{\left|\left|\sum_{y \in N_{H\times W}(x)} (\nabla f)(y)\right|\right|_2}{\sum_{y \in N_{H\times W}(x)} ||(\nabla f)(y)||_2 + 0.5}, \tag{3.4}$$

where f is the blurry observation, ∇ a linear operator computing the first order derivatives in x/y-direction and $N_{H\times W}(x)$ defines a $H \times W$ window centered at x (it is assumed that the unknown blur kernel is of size $H \times W$). This measure, is small for flat regions and spikes in the observed image f, thus it can be used to sort out these edges. Furthermore, small edges are omitted by thresholding the edges of a shock-filtered [32] version of the blurry observation f, yielding an edge image ∇u^S. Using these extracted edges, the blur kernel can be estimated by minimizing the following energy

$$a^* = \arg\min_a \frac{1}{2} \left|\left|\nabla U^S a - \nabla f\right|\right|_2^2 + \frac{\tau}{2} ||a||_2^2,$$

where $\nabla U^S a$ is the matrix-vector representation of the convolution $\nabla u^S * a$ and τ is a weighting parameter used to balance data fidelity and regularization. Since this energy functional is smooth and convex in a, it can be solved in closed form, yielding a fast estimation. Its solution is given by

$$a^* = \left((\nabla U^S)^\mathsf{T} \nabla U^S + \tau I\right)^{-1} (\nabla U^S)^\mathsf{T} \nabla f.$$

The kernel estimate is then used to compute a coarse estimation of the true image by minimizing

$$u^* = \arg\min_u \frac{1}{2} ||A^* u - f||_2^2 + \frac{\lambda}{2} \left|\left|\nabla u - \nabla u^S\right|\right|_2^2,$$

where A^* is the associated matrix when convolving u with a^*. This minimization problem can be also solved in closed form and its minimizer is given by

$$u^* = \left((A^*)^\mathsf{T} A^* + \lambda \nabla^\mathsf{T} \nabla\right)^{-1} \left((A^*)^\mathsf{T} f + \nabla^\mathsf{T} \nabla u^S\right).$$

Their approach yields a fast initial kernel estimate, however, the accuracy is not good enough due to the Gaussian regularization on the kernel and image. To overcome this shortage, in a second phase the kernel is refined using an iterative support detection method which maintains the deblurring quality while removing kernel noise. As we have seen, this estimation method uses a lot of handcrafted operations to ensure a good kernel estimate.

Xu and Jia's method demonstrated the basic principle of edge based kernel estimation approaches. First, suitable edges are selected using a mask. Then the blur kernel is estimated by solving an optimization problem, whose structure varies among the different estimation methods. Finally, the true image is partly reconstructed by optimizing another energy functional depending on the blur kernel estimate and the structure of the method.

A fascinating kernel estimation method, which is also kind of edge based, was introduced by Lai et al. [24]. They first compute an edge mask M based on the r-map (3.4) and the responses of Gaussian derivative filters [43] to exactly locate edges that are present in the true image and not caused by some artifacts such as ringing. This mask yields step edges between two color clusters. In a blurred observation these edges are typically smeared, thus the color distance between two pixels in a patch, centered at an edge pixel, is reduced. The colors of all pixels within such patches must approximately ly on a line connecting the two cluster centers because at step edges are per definition just two colors involved. Within this patch the pixels can be clustered and based on the clustering the original step edge is restored by maximizing the margin between these two clusters. This patch-based edge refinement is performed for many edge pixels yielding a latent coarsely deblurred image, which is used to estimate the unknown blur kernel, see Figure 3.4. The iterative refinement is repeated from coarse-to-fine in an image pyramid to ease kernel estimation in the coarse layers and up-sample good initializations for the next finer one. This method is strongly related to the one proposed by Sun et al. [43]. They also extract patches at step edges and generate a partially deblurred image by restoring the selected step edge's contrast. The so coarsely restored image is then used to restore the blur kernel. The resulting blur kernel estimate at the finest layer is finally used as input to a non-blind image deconvolution method.

3.2.2 Joint image/kernel estimation ($\mathrm{MAP}_{a,u}$)

Despite the problem of the joint *MAP* estimation and its associated energy minimization formulation, there exist some recent methods [33, 34, 42], which deliver solutions on par with State-of-the-Art (SotA) methods.

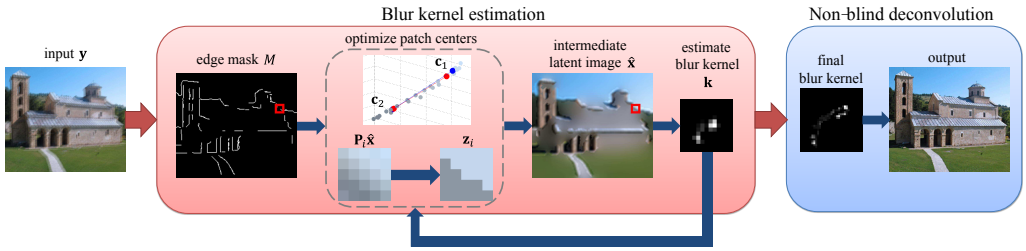


Figure 3.4: Outline of Lai et al. method, Image taken from [24]. For each layer in a coarse-to-fine pyramid an edge mask M is estimated. Then patches at step edge pixel locations are extracted and the step edge is reconstructed by maximizing the distance between the two color cluster centers c_1 and c_2. Based on the so obtained coarsely deblurred image $\hat{x}$, the kernel is estimated. These two steps are repeated until a proper blur kernel estimate is found, which can be used to non-blindly deblur the image by any suitable method.

Shan et al. [42] proposed a successful and interesting method, which was especially designed to avoid ringing artifacts due to poor kernel estimates. Their approach transfered to our energy minimization notation is defined by

$$\begin{aligned}(u^*, a^*) = \arg\min_{u,a} \lambda_1 \left|\left|\Phi(\nabla u)\right|\right|_1 + \left|\left|a\right|\right|_1 + \\ \lambda_2 \left(\left|\left|(\nabla_x u - \nabla_x f) \otimes M\right|\right|_2^2 + \left|\left|(\nabla_y u - \nabla_y f) \otimes M\right|\right|_2^2 \right) + \\ \sum_{i=1}^{N_k} w_i \left|\left|k_i * (u * a - f)\right|\right|_2^2 , \end{aligned} \tag{3.5}$$

where $\otimes$ denotes the point-wise multiplication, λ_1, λ_2 and w_i are weights balancing between the individual components and M is mask. The purpose of this mask is to select regions, where the gradients of the estimated image u should match the blurry ones. This local prior helps to reduce ringing artifacts because M masks out regions containing edges and texture. Thus, it solely constrains flat regions. The global regularizer is the absolute sum of a function $\Phi(x)$ applied to the image gradients. $\Phi(x)$ is a heavy-tailed function, which was trained to best match the gradient statistics of natural images. It is basically a combination of a quadratic and an absolute function. As a consequence, also this prior suffers form the fact that the blurry image has smaller regularization energies.

The more interesting idea in this formulation is the use of different filter k_i in the data term in Equation (3.5). They employ all derivative filters of order zero to two, to better model the Gaussian noise constraints. In their alternating minimization scheme, also first the image is estimated based on the energy (3.5) by holding the blur kernel a fixed. Then a is restored based on the updated image estimate. For a fixed image the energy (3.5) reduces to

$$E(a) = \left|\left|a\right|\right|_1 + \sum_{i=1}^{N_k} w_i \left|\left|k_i(Ua - f)\right|\right|_2^2 .$$

Algorithm 3: Projected Alternating Minimization Algorithm of Perrone and Favaro [34].

Data: f, blur size, initail large λ, λ_{min}
Result: u,a
$u_0 \leftarrow \text{pad}(f)$;
$a_0 \leftarrow$ uniform;
while *not converged* **do**
  $u_{t+1} \leftarrow u_t - \epsilon_u \left(A_t^\mathsf{T}(A_t u_t - f) - \lambda \nabla \cdot \frac{\nabla u_t}{|\nabla u_t|} \right)$;
  $a_{t+1/3} \leftarrow k_t - \epsilon_a \left(U_{t+1}^\mathsf{T}(U_{t+1} a_t - f) \right)$;
  $a_{t+2/3} \leftarrow \max(a_{t+1/3}, 0)$;
  $a_{t+1} \leftarrow \frac{a_{t+2/3}}{||a_{t+2/3}||_1}$;
  $\lambda \leftarrow \max(0.99\lambda, \lambda_{min})$;
  $t \leftarrow t + 1$;
$u \leftarrow u_{t+1}$;
$a \leftarrow a_{t+1}$;

There is a loose relation to the edge-based kernel estimation methods of the previous section because if minimizing this energy, the kernel is especially influenced by edge regions.

To overcome the problem of blind-deconvolution in energy minimization, Shan et al. reduce the influence of the regularizer after each iteration of their optimization algorithm by decreasing the parameters λ_1 and λ_2. Ergo they reduce the over-smoothing effect of the regularizer. Nevertheless, this approach is strictly speaking no energy minimization method more.

Another method which also exploits this idea was introduced by Perrone and Favaro [34]. Their variational formulation is very simple and defined as

$$(u^*, a^*) = \arg\min_{u,a} E(u, a) = \lambda \, ||\nabla u||_{2,1} + ||a * u - f||_2^2 + \delta_\Delta(a), \tag{3.6}$$

where they moved the weighting parameter λ to the *TV* regularizer. To minimize this energy, they proposed a projected alternating minimization algorithm, see Algorithm 3. In contrast to the previous methods, no optimization problems are solved during the individual iterates of their algorithm. They simply alternate between a gradient descent step with respect to the estimated image and one for the kernel followed by a back-projection onto the unit simplex. At the end of each iterate the influence of the regularizer is reduced by decreasing λ, which mimics the behavior of Shan and colleagues. Their algorithm works because it exploits the favor of piece-wise constant solutions of the *TV*. Initially the weight λ is very high, thus the first intermediate images are favored to be piece-wise constant. As a consequence, their edges are sharper and closer to the true solution. In this fashion the blur kernel estimate gets closer to the true one, avoiding the delta solution. If the kernel estimate is good enough, the regularization on the image

gradients can be decreased to better reconstruct image details.

An advantage of this algorithm is that it has very simple update rules in each iteration, however, quite many iterations are necessary due to the re-weighting of the energy. Furthermore, no subgradient information is incorporated when computing the derivative of the *TV* and the projection on the simplex is not a proper projection. The method produces results close to *SotA*, though.

3.2.3 Learning based

Recently, more and more learning based methods are proposed due to their success in other image processing fields. A very promising and efficient algorithm was proposed by Schelten et al. [39]. The novel idea is that they learn the parameters of the posterior probability directly in a discriminate way using Regression Tree Fields (RTFs). It can be expressed as an Gaussian Conditional Random Fields (CRFs) of the form

$$p(u|f,a) \propto \mathcal{N}(u|\mu(f,a), C(f,a)),$$

where the parameters of the mean $\mu(f,a)$ and the associated covariance matrix $C(f,a)$ are partly derived from the learned *RTFs*, while keeping the kernel a fixed. Details about training and evaluating *RTFs* can be found in [19, 20]. To estimate a sharp image based on the Gaussian *CRFs*, they maximize the posterior likelihood

$$u^* = \arg\max_u p(u|f,a) = \mu(f,a).$$

The mean can be computed by

$$\begin{aligned} C(f,a) &= \left(W(f) + \frac{1}{\sigma^2} A^\mathsf{T} A\right)^{-1} \\ \mu(f,a) &= C(f,a)\left(w(f) + \frac{1}{\sigma^2} A^\mathsf{T} f\right), \end{aligned}$$

where the matrix $W(f)$ and the vector $w(f)$ are regressed from the input image features by the *RTFs*. Due to the complexity of the blind deconvolution problem, a single estimation of the mean is not sufficient. Thus, they stack some *RTFs* together to form a cascade, which leads to subsequently better estimates. As a result, also the intermediate images u_i and kernels a_i, which are the output of the i-th cascade, can be used to better estimate the parameters of the next stage $W_{i+1}(f,u_i)$ and $w_{i+1}(f,u_i)$. Algorithm 4 outlines the basic computations within the method of Schelten and colleagues. By comparing this algorithm to the one introduced by Levin et al. [27], we see that the estimation of the latent image u_i is quite similar because in both algorithm it is computed incorporating the covariance. The difference lies in the prior as Schelten et al. use *RTFs* and Levin et al. count on sparse image gradient priors. In contrast, the kernel estimation is quite different. Schelten et

Algorithm 4: Blind deconvolution using interleaved *RTFs* cascade of [39].

Data: f, initial blur kernel a_0
Result: u,a
for $i = 1 \ldots N$ **do**
  $u_i \leftarrow \left(W_i(f, u_{i-1}) + \frac{1}{\sigma^2} A_{i-1}^{\mathsf{T}} A_{i-1}\right)^{-1} \left(w_i(f, u_{i-1}) + \frac{1}{\sigma^2} A_{i-1}^{\mathsf{T}} f\right)$;
  $a_i = \arg\min_a \lambda \left|\left|a\right|\right|_1 + \left|\left|\nabla f - a * (\nabla u_i)\right|\right|_2^2$;
$u \leftarrow u_N$;
$a \leftarrow a_N$;

al. estimate the blur kernel in each stage by minimizing

$$a_i = \arg\min_a \lambda \left|\left|a\right|\right|_1 + \left|\left|\nabla f - a * (\nabla u_i)\right|\right|_2^2 .$$

So, they compute a sparse prior based on the edges of the blurry input f and the current image estimate u_i. In this manner a subsequent kernel refinement is done within the cascade. To sum up, their method achieves the best blind deconvolution results at the moment. Its success lies in the iterative refinement of an already good initial blur kernel estimate and the expressive power of the *RTFs* which model the posterior probabilities.

Another learning based blind deconvolution method which takes the idea of learning a bit further was proposed by Schuler et al. [41]. Instead of separating the feature extraction, kernel estimation and image reconstruction phases of the previous algorithms, they embed all into a Neural Network (NN) framework. Thus, all parameters of the individual stages can be trained jointly. Therefore, they introduced a network architecture consisting of three modules.

The first module is used to extract suitable features for blur kernel estimation. Within it, a convolutional layer is used to extract features, which are then combined using a linear network layer to form the features for kernel estimation. In the second module the blur kernel is estimated by solving

$$\tilde{a} = \arg\min_a \sum_i \left|\left| a * \tilde{u}_i - \tilde{f}_i \right|\right|_2^2 + \beta_a \left|\left|a\right|\right|_2^2 ,$$

where $\tilde{u}_i$ and $\tilde{f}_i$ are the computed features of the feature extraction module. This problem is quadratic in the kernel a, thus it can be solved in closed form and efficiently computed in Fourier space.

$$\tilde{a} = \mathcal{F}^{-1}\left(\frac{\sum_i \overline{\mathcal{F}(\tilde{u}_i)} \odot \mathcal{F}(\tilde{f}_i)}{\sum_i \left|\mathcal{F}(\tilde{u}_i)\right|^2 + \beta_a}\right) \tag{3.7}$$

In this formulation the Fourier transform is denoted by $\mathcal{F}(\cdot)$ and its inverse by $\mathcal{F}^{-1}(\cdot)$, $\bar{\cdot}$ denotes the complex conjugate and $\odot$ the Hadamard product. The hyper-parameter

β_a is trained during network training and is responsible to balance between Tikhonov regularization and data-fitting. This formulation can be easily transfered into the *NN* layer structure, yielding the kernel estimation module. In their work, they call this layer type 'quotient'-layer. The final third module can be derived in a similar manner. Based on the resulting blur kernel estimate and the computed features, the image is restored by minimizing

$$\tilde{u} = \arg\min_u ||\tilde{a} * u - f||_2^2 + \beta_u ||u||_2^2 ,$$

which can be solved again by using a quotient layer. To get the final network architecture, a series of those three modules are stacked together, where the output of the image restoration module is the input to the feature extraction of the next stage. Thus, the entire network can be interpreted as computing a fixed amount of iterates of a simple blind deconvolution algorithm. The advantage of this formulation is that it can be easily trained end-to-end. As all other blind deconvolution methods, the network is applied using a coarse-to-fine image pyramid. Despite the unsuitable Tikhonov regularization when computing the kernel and image estimate, their network performs quite well.

3.3 Blind image deconvolution using iteratively adapted energy minimization

We motivate our algorithm by the outstanding results of the learning based blind deconvolution methods and those of the joint image/kernel estimation ($\text{MAP}_{a,u}$). Therefore, we want to minimize the energy in Equation (3.1) in a joint fashion. The success of the prior proposed by Chen et al. [13], drives its application in our model. By introducing this prior in (3.1), one gets

$$(u^*, a^*) = \arg\min_{u,a} E(u,a) = \sum_{p=1}^{MN} \sum_{i=1}^{N_k} \rho_i \left((k_i * u)_p\right) + \frac{\lambda}{2} ||a * u - f||_2^2 + \delta_\Delta(a), \tag{3.8}$$

where $\rho_i(x)$ is an arbitrarily shaped penalty function and k_i its associated filter kernel. The function $\delta_\Delta(a)$ is the indicator function of the unit simplex, defined in Equation (1.21). This energy is in general non-convex due to the freedom of the penalty functions $\rho_i(x)$. Furthermore, $\delta_\Delta(a)$ is non-smooth, thus computing the true minimizer is rather hard. However, Bolte et al. [5] recently proposed an algorithm called Proximal Alternating Linearized Minimization (PALM), which is able to solve a certain class of non-convex and non-smooth problems. They proved that each bounded sequence generated by *PALM* globally converges to a critical point. These points are those in the solution space, whose subdifferential contains 0, thus local/global minimum/maximum and saddle points. It can be applied to minimize problem (3.8), as we show in the next sections.

Algorithm 5: Proximal Alternating Linearized Minimization algorithm of [5].

Data: start with any $(x^0, y^0) \in \mathbb{R}^n \times \mathbb{R}^m$
Result: (x, y)
while *not converged* **do**
  Take $\gamma_1 > 1$, set $c_k = \gamma_1 L_1(y_k)$ and compute
  $x_{k+1} \in \text{prox}_{c_k f}\left(x_k - \frac{1}{c_k}\nabla_x H(x_k, y_k)\right)$;
  Take $\gamma_2 > 1$, set $d_k = \gamma_2 L_2(x_{k+1})$ and compute
  $y_{k+1} \in \text{prox}_{d_k g}\left(y_k - \frac{1}{d_k}\nabla_y H(x_{k+1}, y_k)\right)$;
  $k \Leftarrow k + 1$;
$x \leftarrow x_k$;
$y \leftarrow y_k$;

3.3.1 Details on *PALM*

In order to validate the usage of *PALM* to minimize energy (3.8), we first have to ensure that the problem fulfills the constraints. As demonstrated in [5], *PALM* is a powerful algorithm for solving non-convex and non-smooth problems of the form

$$\min_{x,y} \Psi(x, y) := f(x) + g(y) + H(x, y). \tag{3.9}$$

It is assumed that the functions $f : \mathbb{R}^n \to (-\infty, \infty]$ and $g : \mathbb{R}^m \to (-\infty, \infty]$ are proper and lower semicontinuous functions and $H : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is C^1. Algorithm 5 sketches their proposed scheme to solve problem (3.9), where the proximal operator, defined as

$$x_{k+1} = \text{prox}_{tf}\left(x_k - \frac{1}{t}\nabla h(x_k)\right),$$

is equivalent to solving the following problem

$$x_{k+1} \in \arg\min_x \langle x - x_k, \nabla h(x_k)\rangle + \frac{t}{2}\,||x - x_k||^2 + f(x). \tag{3.10}$$

The application of *PALM* is tied to the justification of the following problem assumptions:

(i) $\inf_{\mathbb{R}^n \times \mathbb{R}^m} \Psi > -\inf$, $\inf_{\mathbb{R}^n} f > -\inf$ and $\inf_{\mathbb{R}^m} g > -\inf$

(ii) For any fixed y the function $x \to H(x, y)$ is $C^{1,1}_{L_1(y)}$, namely the partial gradient $\nabla_x H(x^k, y^k)$ is globally Lipschitz with moduli $L_1(y)$. Likewise, for any fixed x the function $y \to H(x, y)$ is assumed to be $C^{1,1}_{L_2(x)}$ either.

(iii) If H is C^2, also the other assumptions in [5] are satisfied.

If they are fulfilled, Algorithm 5 can be employed to solve the desired problem.

3.3.2 Applying *PALM* to blind image deconvolution

First, we need to map the energy minimization problem for blind image deconvolution (3.8) to the general form (3.9). If we set x to be the image u and y to reflect the estimated blur kernel a, a suitable choice of the functions is

$$\begin{aligned} H(a,u) &= \frac{1}{N_k}\sum_{p=1}^{MN}\sum_{i=1}^{N_k}\rho_i(k_i * u) + \frac{\lambda}{2}\,||u*a-f||_2^2 \\ f(u) &= 0 \\ g(a) &= \delta_\Delta(a). \end{aligned}$$

With this setup, we have ensured that $f(u)$ and $g(a)$ are proper and lower semicontinuous functions and moreover, that $H(a,u)$ is C^1.

3.3.2.1 Proof of applicability

Due to the chosen mapping, the basic assumption are validated. It remains to show that this mapping satisfies the necessary assumptions (i) to (iii).

(i) Since $H(a,u) \geq 0\ \forall a,u$ and by the definition of the functions f and g, assumption (i) is proven.

(ii) To fulfill this constraint, we need to compute the Lipschitz constants of the gradients $\nabla_u H(a,u)$ and $\nabla_a H(a,u)$. We start by stating the gradient of $H(a,u)$ with respect to a

$$\nabla_a H(a,u) = \lambda U^\mathsf{T}(Ua-f),$$

where $u*a \Leftrightarrow Ua$. The Lipschitz constant $L_2(u)$ is characterized by satisfying the following property

$$||\nabla_a H(a_1,u) - \nabla_a H(a_2,u)||_* \leq L_2(u)\,||a_1-a_2||\,, \tag{3.11}$$

where $||\cdot||_*$ is the dual norm of the applied norm in the proximal map $||\cdot||$. Hence, the Lipschitz constant of the kernel update can be computed as

$$\begin{aligned} \left|\left|\lambda U^\mathsf{T}(Ua_1-f) - \lambda U^\mathsf{T}(Ua_2-f)\right|\right| &\leq |\lambda|\left|\left|\lambda U^\mathsf{T}U(a_1-a_2)\right|\right| \\ &\leq \lambda\left|\left|U^\mathsf{T}U\right|\right|\,||a_1-a_2|| \\ L_2(u) &= \lambda\left|\left|U^\mathsf{T}U\right|\right|, \end{aligned}$$

which concludes the first part of this proof. Note that we do not specify this norm, since it is related to the norm used in the proximal map (3.10).

The computation of the Lipschitz constant $L_1(a)$ is a bit more elaborate. Again, we start by computing $\nabla_u H(a,u)$

$$\nabla_u H(a,u) = \frac{1}{N_k}\sum_{i=1}^{N_k} K_i^T \rho_i'(K_i u) + \lambda A^T(Au - f),$$

whereby $u * a \Leftrightarrow Au$, $k_i * u \Leftrightarrow K_i u$ and $\rho'(x)$ denotes the first order derivative of the function $\rho(x)$. By substituting $\phi_i(x) = \rho_i'(x)$ and applying the Lipschitz definition for $\nabla_u H$ one gets

$$\left|\left|\frac{1}{N_k}\sum_{i=1}^{N_k} K_i^T(\phi_i(K_i u_1) - \phi_i(K_i u_2)) + \lambda A^T A(u_1 - u_2)\right|\right|_2 \leq L_1(a)\,||u_1 - u_2||_2$$

$$\frac{1}{N_k}\sum_{i=1}^{N_k} \left|\left|K_i^T\right|\right|_2 ||\phi_i(K_i u_1) - \phi_i(K_i u_2)||_2 + \lambda \left|\left|A^T A\right|\right|_2 ||u_1 - u_2||_2 \leq L_1(a)\,||u_1 - u_2||_2 .$$

Here the ℓ_2-norm is used because the proximal map of the image update is solved using it. To estimate the Lipschitz constant we need to compute an upper bound for

$$||\phi_i(K_i u_1) - \phi_i(K_i u_2)||_2 .$$

Therefore, we assume that $\phi_i(x) \in C^1$ and hence $\rho_i \in C^2$. In other words, $\phi_i(x)$ has a smooth derivative. Then its Lipschitz constant is defined as

$$L_i = \sup_x \left|\phi_i'(x)\right| .$$

Thus, an upper bound can be formulated as

$$||\phi_i(K_i u_1) - \phi_i(K_i u_2)||_2 \leq L_i\,||K_i u_1 - K_i u_2||_2 \leq L_i\,||K_i||_2\,||u_1 - u_2||_2 .$$

By putting all pieces together we can define $L_1(a)$ as

$$L_1(a) = \frac{1}{N_k}\sum_{i=1}^{N_k} \left|\left|K_i^T\right|\right|_2 L_i\,||K_i||_2 + \lambda \left|\left|A^\intercal A\right|\right|_2 .$$

Due to the physical properties of the blur kernel a and the parametrization of the filters k_i, all this kernels have norm 1. This property is shared between a convolution kernel and its associated matrix representation. Consequently, the Lipschitz constant

simplifies to

$$L_1(a) = \lambda + \frac{1}{N_k} \sum_{i=1}^{N_k} L_i$$

and assumption (ii) is proven.

(iii) Since we require $\rho_i(x)$ to be C^2 and $||a * u - f||_2^2 \in C^2$ with respect to a and u, also $H(a, u) \in C^2$ and assumption (iii) follows.

As a result, the *PALM* algorithm can be applied to solve problem (3.8).

3.3.2.2 Derivation of proximal maps

As we have proven, the blind image deconvolution problem can be solved by the *PALM* algorithm. In order to apply it, we first need to compute the proximal maps, which are stated in Algorithm 5.

Proximal map of the kernel: Let us start with the derivation of the proximal map related to the blur kernel a.

$$a_{k+1} \in \text{prox}_{d_k g}\left(a_k - \frac{1}{d_k} \nabla_a H(a_k, u_{k+1})\right)$$

This map can be computed by solving the problem

$$a_{k+1} \in \arg\min_a \left\langle a - a_k, \lambda U_{k+1}^{\mathsf{T}}(U_{k+1}a_k - f)\right\rangle + \frac{d_k}{2} ||a - a_k|| + g(a),$$

which is equivalent to solve the following smooth constrained optimization problem

$$a_{k+1} = \arg\min_{a \in \Delta} \left\langle a - a_k, \lambda U_{k+1}^{\mathsf{T}}(U_{k+1}a_k - f)\right\rangle + \frac{d_k}{2} ||a - a_k||, \tag{3.12}$$

since $g(a) = \delta_\Delta(a)$ is the indicator function of the unit simplex. This kind of constrained optimization problems has been well studied [3, 44], since the unit simplex is a convex set which is widely used, e.g. at the optimization of probability distributions. Unfortunately, a closed form solution does not exists for the above problem due to the simplex constraint. However, as shown by Beck and Teboulle [2] and Ben-Tal et al. [3] a closed form solution of this convex problem can be computed by introducing a more general distance metric. Therefore, we reformulate the proximal map problem to

$$a_{k+1} = \arg\min_{a \in \Delta} \left\langle a - a_k, \lambda U_{k+1}^{\mathsf{T}}(U_{k+1}a_k - f)\right\rangle + d_k D_f(a, a_k), \tag{3.13}$$

where D_f denotes the Bregman divergence associated with the function $f(x) = \sum_i x_i \log(x_i)$, which is also known as the entropy. As introduced by Bregman [7]

in 1967, D_f is defined as

$$D_f(a,b) = f(a) - f(b) - \langle \nabla f(b), a - b \rangle ,$$

where he already proved that the entropy $f(x)$ yields proper Bregman distances D_f. Furthermore, Teboulle proved in [44] that problem (3.12) and (3.13) share important properties regarding minimization, i.e. they share the set of minimizer. Therefore, we further on consider problem (3.13). When plugging the function f into the definition of the Bregman divergence, we get

$$\begin{aligned} D_f(a,b) &= \sum_i a_i \log(a_i) - \sum_i b_i \log(b_i) - \langle \log(b) + 1, a - b \rangle \\ &= \sum_i a_i (\log(a_i) - \log(b_i)) - (a_i - b_i). \end{aligned}$$

This reveals the advantage of problem (3.13). If any element a_i of the blur kernel a is below or equal to zero, $D_f(a, a_k) \to \infty$. Consequently, we can omit the $a_i \geq 0$ constraint of the unit simplex Δ, as it is implicitly handled by the distance function. Hence, the problem simplifies to

$$a_{k+1} = \arg \min_{\sum_i a_i = 1} \left\langle a - a_k, \lambda U_{k+1}^{\mathsf{T}} (U_{k+1} a_k - f) \right\rangle + d_k D_f(a, a_k),$$

which can be solved using a Lagrange multiplier. The associated Lagrange function is given by

$$L(a,\nu) = \left\langle a - a_k, \lambda U_{k+1}^{\mathsf{T}} (U_{k+1} a_k - f) \right\rangle + d_k D_f(a, a_k) + \nu \left(\sum_i a_i - 1 \right).$$

A minimizer of the above problem has to fulfill the following conditions $\nabla_a L(a,\nu) = 0$ and $\nabla_\nu L(a,\nu) = 0$.

$$\begin{aligned} \nabla_a L(a,\nu) &= \lambda U_{k+1}^{\mathsf{T}} (U_{k+1} a_k - f) + d_k (\log(a) - \log(a_k)) + \nu = 0 \\ \nabla_\nu L(a,\nu) &= \sum_i a_i - 1 = 0 \end{aligned}$$

Let us introduce a new variable $\nabla a_k = \lambda U_{k+1}^{\mathsf{T}} (U_{k+1} a_k - f)$ to ease writing. In order to compute a solution, we have to calculate the Lagrange multiplier ν beforehand.

$$\begin{aligned} \nabla a_k + d_k (\log(a) - \log(a_k)) + \nu &= 0 \\ -\frac{1}{d_k} \nabla a_k + \log(a_k) - \frac{1}{d_k} \nu &= \log(a) \\ \operatorname{diag} \left(\exp \left(-\frac{1}{d_k} (\nabla a_k + \nu) \right) \right) a_k &= a \end{aligned}$$

This equation must hold for all elements in the vector a.

$$\exp\left(-\frac{1}{d_k}\left(\nabla a_k + \nu\right)\right)_i a_{ki} = a_i$$

By summing up all these individual equations, we can express the remaining simplex constraint.

$$\sum_i \exp\left(-\frac{1}{d_k}\left(\nabla a_k + \nu\right)\right)_i a_{ki} = \sum_i a_i \overset{!}{=} 1$$

The Lagrange multiplier ν can then be computed by

$$\begin{aligned}
\exp\left(-\frac{1}{d_k}\nu\right)\sum_i \exp\left(-\frac{1}{d_k}(\nabla a_k)_i\right) a_{ki} &= 1 \\
\exp\left(\frac{1}{d_k}\nu\right) &= \sum_i \exp\left(-\frac{1}{d_k}(\nabla a_k)_i\right) a_{ki} \\
\nu &= d_k \log\left(\sum_i \exp\left(-\frac{1}{d_k}(\nabla a_k)_i\right) a_{ki}\right).
\end{aligned}$$

If we plug this result in the initial equation, we get

$$\begin{aligned}
0 &= \nabla a_k + d_k(\log(a) - \log(a_k)) + \nu \\
0 &= \nabla a_k + d_k(\log(a) - \log(a_k)) + d_k \log\left(\sum_i \exp\left(-\frac{1}{d_k}(\nabla a_k)_i\right) a_{ki}\right) \\
\log(a) &= -\frac{1}{d_k}\nabla a_k + \log(a_k) - \log\left(\sum_i \exp\left(-\frac{1}{d_k}(\nabla a_k)_i\right) a_{ki}\right)
\end{aligned}$$

By computing the exponential function of this equation we get the final proximal map

$$\begin{aligned}
a &= \operatorname{diag}\left(\exp\left(-\frac{1}{d_k}\nabla a_k\right)\right) a_k \left(\sum_i \exp\left(-\frac{1}{d_k}(\nabla a_k)_i\right) a_{ki}\right)^{-1} \\
a &= \frac{\operatorname{diag}\left(\exp\left(-\frac{1}{d_k}\nabla a_k\right)\right) a_k}{\sum_i \exp\left(-\frac{1}{d_k}(\nabla a_k)_i\right) a_{ki}}.
\end{aligned}$$

Again, this equation must be valid for any element in a.

$$a_j = \frac{a_{kj} \exp\left(-\frac{1}{d_k}(\nabla a_k)_j\right)}{\sum_i a_{ki} \exp\left(-\frac{1}{d_k}(\nabla a_k)_i\right)}$$

It is clear, that this entropic proximal map computes a blur kernel $a \in \Delta$, as long as $a_k \in \Delta$ is true.

As we have pointed out during the computation of the Lipschitz constant $L_2(u)$, the norms applied in Equation (3.11) depend on the distance function used in the proximal map. Although the ℓ_2-norm also generates suitable results, we can do better if the ℓ_1-norm is used. To validate its usage, we have to show that the Bregman divergence $D_f(a, b)$ is bounded from below by the associated ℓ_1-norm distance for all elements in the domain.

$$D_f(a, b) \geq \frac{1}{2} ||a - b||_1^2 \ \forall a, b \in \Delta$$

In other words, the ℓ_1-norm distance never over-estimates the actual used one. To do so, we start by plugging in the definition of D_f

$$f(a) - f(b) - \langle \nabla f(b), a - b \rangle \geq \frac{1}{2} ||a - b||_1^2 \ \forall a, b \in \Delta,$$

which is equivalent to

$$f(a) \geq f(b) + \langle \nabla f(b), a - b \rangle + \frac{1}{2} ||a - b||_1^2 \ \forall a, b \in \Delta.$$

Thus, the function $f(a)$ must be 1-strongly convex with respect to the ℓ_1-norm on the unit simplex Δ. This circumstance can also be expressed as the strongly monotone property

$$\langle \nabla f(a) - \nabla f(b), a - b \rangle \geq ||a - b||_1^2 \ \forall a, b \in \Delta,$$

see [31], which has been proven by Beck and Teboulle [2]. As a result, we can update $L_2(u)$. The updated version of Equation (3.11) is then given by

$$||\nabla_a H(a_1, u) - \nabla_a H(a_2, u)||_\infty \leq L_2(u) \, ||a_1 - a_2||_1 \,,$$

since $||\cdot||_\infty$ is the dual norm of the ℓ_1-norm. Hence the Lipschitz constant can be computed by

$$L_2(u) = \lambda \left|\left| U^\mathsf{T} U \right|\right|_{1,\infty} = \lambda \, ||u||_2^2 \,,$$

where the $||\cdot||_{1,\infty}$ is defined as

$$||A||_{1,\infty} = \sup_{||x||_1 = 1} ||Ax||_\infty \,.$$

Thus the Lipschitz constant simplifies to the sum over all image pixel squares. This can be seen if the structure of the convolution matrix U is analyzed. It basically consists of shifted versions of the image u, therefore, if multiplied with its transpose, the correlations of the shifted image variants are computed. Since the $||\cdot||_{1,\infty}$ extracts the maximum element of its

argument, the Lipschitz constant is given by the maximal correlation, which is given for no shift. Hence, the Lipschitz constant is equivalent to the maximum of the autocorrelation function. This Lipschitz constant is very important for the further derivations, since it reflects the data dependency of the step size d_k.

Proximal map of the image: The proximal map with respect to the image can be computed much easier. It is defined as

$$u_{k+1} \in \operatorname{prox}_{c_k f}\left(u_k - \frac{1}{c_k}\nabla_u H(a_k, u_k)\right),$$

which is equivalent to solving the following problem

$$u_{k+1} = \arg\min_u \left\langle u - u_k, \frac{1}{N_k}\sum_{i=1}^{N_k} K_i^\mathsf{T}\phi_i(K_i u_k) + \lambda A_k(A_k u_k - f)\right\rangle + \frac{c_k}{2}\,||u - u_k||_2^2.$$

This smooth unconstrained convex optimization problem can be solved in closed form and its solution is given by

$$u_{k+1} = u_k - \frac{1}{c_k}\left\{\frac{1}{N_k}\sum_{i=1}^{N_k} K_i^\mathsf{T}\phi_i(K_i u_k) + \lambda A_k(A_k u_k - f)\right\}.$$

It is basically a gradient descent step with the blur kernel estimate of the previous iteration a_k in the data term. This iterative update rule is almost the same as the one in the non-blind image deconvolution problem, see Equation (2.13).

3.3.2.3 *PALM* for blind image deconvolution

By putting all the bits and pieces of the previous derivations together, we can state the *PALM* algorithm for blind image deconvolution, see Algorithm 6. The corresponding Lipschitz constants are defined as

$$L_1(a_k) = \lambda + \frac{1}{N_k}\sum_{i=1}^{N_k} L_i, \tag{3.14}$$

$$L_2(u_{k+1}) = \lambda\,||u_{k+1}||_2^2 \tag{3.15}$$

where $L_i = \sup_x |\phi_i'(x)|$ is the Lipschitz constant of the i-th influence function. If we compare this algorithm to the projected alternating minimization method of Perrone and Favaro 3, we see that the image update is almost identical, whereas, the kernel refinement can be done in closed form with our approach. Furthermore, our projection method is theoretically justified and differentiable.

Algorithm 6: Proximal Alternating Linearized Minimization algorithm for blind image deconvolution.

Data: start with (a^0, u^0)
Result: (a, u)
while *not converged* **do**
  Take $\gamma_1 > 1$, set $c_k = \gamma_1 L_1(a_k)$ and compute
  $u_{k+1} = u_k - \frac{1}{c_k}\left\{\frac{1}{N_k}\sum_{i=1}^{N_k} K_i^\mathsf{T}\phi_i(K_i u_k) + \lambda A_k(A_k u_k - f)\right\}$;
  Take $\gamma_2 > 1$, set $d_k = \gamma_2 L_2(u_{k+1})$ and compute
  $\nabla a_k = \lambda U_{k+1}^\mathsf{T}(U_{k+1}a_k - f)$;
  $a_{(k+1),j} = \frac{a_{kj}\exp\left(-\frac{1}{d_k}(\nabla a_k)_j\right)}{\sum_i a_{ki}\exp\left(-\frac{1}{d_k}(\nabla a_k)_i\right)}$;
  $k \Leftarrow k+1$;
$a \leftarrow a_k$;
$u \leftarrow u_k$;

3.3.3 Iteratively adapted energy minimization approach

As we have seen, the blind image deconvolution problem can be solved by iterating between two gradient descent steps. While, the update rule of the image estimate u in Algorithm 6 is almost identical to the non-blind deconvolution scheme (2.13), the gradient descent step of the blur kernel a is multiplicative. In general, such schemes require hundreds or thousands of iterations to get a good estimate for u and a due to the relatively poor convergence rate of this simple gradient related algorithms. The iterative approach of Perrone and Favaro [34] takes for instance $10,000$ iterations. Although this iterates can be computed very fast, it is impractical for real applications.

To overcome this problem, the implementation of the same idea as in the non-blind case is an obvious choice. Therefore, the amount of iterations are fixed to T and by means of discriminative supervised learning the output of the last iteration is tuned to best fit a certain objective criterion. Hence, the iterative approach is transfered to a network consisting of T stages, where each computes a single iterate. Furthermore, the parameters of the data term λ_t and the regularization k_{ti} and ϕ_{ti} are allowed to freely change within each stage. As a consequence, the update rules of a single stage are defined by

$$u_t = u_{(t-1)} - \frac{1}{c_t}\left\{\frac{1}{N_k}\sum_{i=1}^{N_k} K_{ti}^\mathsf{T}\phi_{ti}(K_{ti}u_{(t-1)}) + \lambda_t A_{(t-1)}(A_{(t-1)}u_{(t-1)} - f)\right\}$$
$$\nabla a_{(t-1)} = \lambda_t U_t^\mathsf{T}(U_t a_{(t-1)} - f)$$
$$a_{t,j} = \frac{a_{(t-1)j}\exp\left(-\frac{1}{d_t}(\nabla a_{(t-1)})_j\right)}{\sum_i a_{(t-1)i}\exp\left(-\frac{1}{d_t}(\nabla a_{(t-1)})_i\right)}.$$

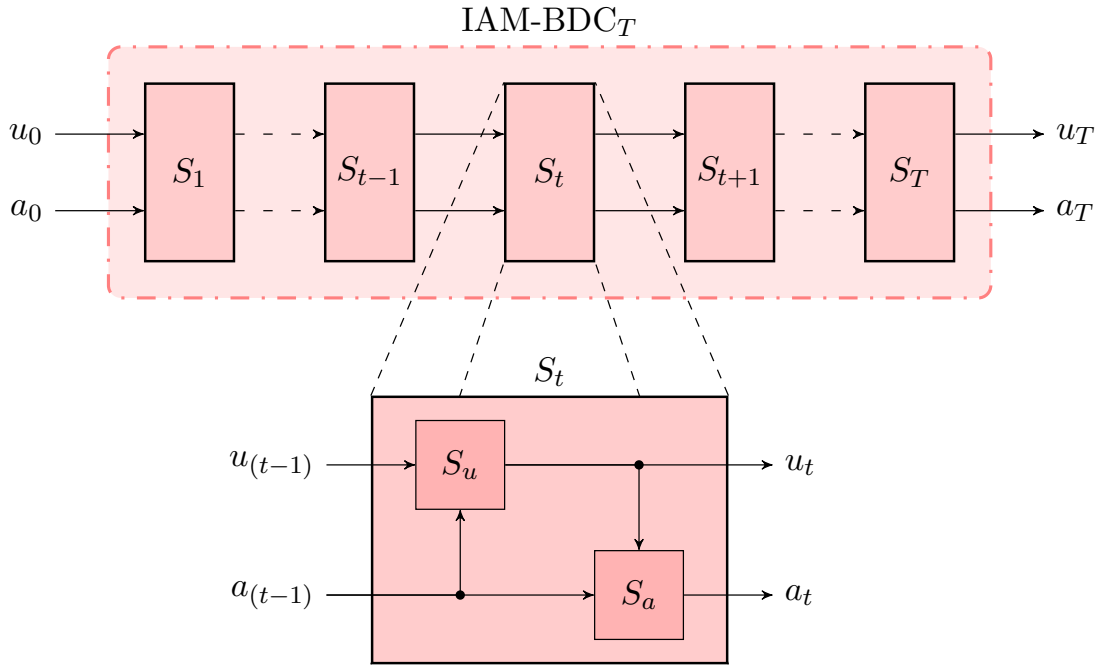


Figure 3.5: Demonstration of the Iteratively Adapted Minimization for blind Deconvolution (IAM-BDC) network structure. The input the procedure is the blurry observation u_0 and the initial blur estimate a_0. These are processed by a series of stages S_t, which implement the image update scheme S_u, see Equation (3.16), and the kernel one S_a, depicted in Equation (3.17). The output of the net after performing T updates is the deconvolved image u_T and the blur kernel estimate a_T.

The step sizes c_t can be omitted, since it can be expressed by appropriately scaling the influence functions ϕ_{ti} and the weighting parameter λ_t. However, the step size d_t reflects a data dependency of the kernel update on the input image, as shown during the computation of the Lipschitz constant $L_2(u)$. In oder to keep this data dependency, we express it by $1/d_t = \tau_t / \left|\left|u_{(t-1)}\right|\right|_2^2$. Hence, the update scheme changes to

$$u_t = u_{(t-1)} - \left\{ \frac{1}{N_k} \sum_{i=1}^{N_k} K_{ti}^\mathsf{T} \phi_{ti}(K_{ti} u_{(t-1)}) + \lambda_t A_{(t-1)}(A_{(t-1)} u_{(t-1)} - f) \right\} \tag{3.16}$$

$$\begin{aligned} \nabla a_{(t-1)} &= U_t^\mathsf{T}(U_t a_{(t-1)} - f) \\ a_{t,j} &= \frac{a_{(t-1)j} \exp\left(-\frac{\tau_t}{\left|\left|u_{(t-1)}\right|\right|_2^2} (\nabla a_{(t-1)})_j \right)}{\sum_i a_{(t-1)i} \exp\left(-\frac{\tau_t}{\left|\left|u_{(t-1)}\right|\right|_2^2} (\nabla a_{(t-1)})_i \right)}. \end{aligned} \tag{3.17}$$

The resulting network structure of the proposed *IAM-BDC* approach is depicted in Figure 3.5. The blurry observation u_0 and the initial blur kernel a_0 is processed by the first stage S_1 and its output is further refined by successive stages until the final stage S_T is

reached. Moreover, Figure 3.5 outlines the interrelation of the image and kernel stage update rules. The image propagation rule S_u is based on the output of the previous stage, whereas, the kernel update S_a already utilizes the updated image u_t. This scheme favors the kernel estimation because within a single stage the image $u_{(t-1)}$ can be modified such that the blur kernel estimate at the output of the stage a_t is better.

However, experiments showed that the kernel estimates a_T of this network are very poor if the update rules (3.16) and (3.17) are used. If we compare our approach to *SotA* methods such as [24, 39, 41, 42, 47], one difference pops out. All these algorithm estimate the unknown blur kernel on filtered versions of the blurry observation f and the current estimate u. The additional filtering is necessary because strong edges are better suited for the estimation of blur kernels as we have seen in the discussion of the related methods, see Section 3.2. Therefore, we alter the data term in a similar way as in the non-blind deconvolution network by introducing data term filters $\bar{k}_{ti}$, $\tilde{k}_{ti}$ and its associated influence functions $\bar{\phi}_{ti}$, $\tilde{\phi}_{ti}$. The modified stage update rules are then given by

$$\begin{aligned} u_t = T\Bigg\{ u_{p(t-1)} - \Bigg(& \frac{1}{N_k}\sum_{i=1}^{N_k} K_{ti}^\mathsf{T}\phi_{ti}(\eta K_{ti}u_{p(t-1)}) + \dots \\ & \frac{1}{N_d}\sum_{i=1}^{N_d} A_{(t-1)}^\mathsf{T}\bar{K}_{ti}^\mathsf{T}\bar{\phi}_{ti}\left(\bar{\eta}\bar{K}_{ti}\left(A_{(t-1)}u_{p(t-1)} - f\right)\right)\Bigg)\Bigg\} \end{aligned} \tag{3.18}$$

$$\begin{aligned} \nabla a_{(t-1)} &= \frac{1}{N_d}\sum_{i=1}^{N_d} \frac{1}{\left|\left|\tilde{K}_{ti}u_t\right|\right|_2^2} U_t^\mathsf{T}\tilde{K}_{ti}^\mathsf{T}\tilde{\phi}_{ti}\left(\tilde{\eta}\tilde{K}_{ti}\left(U_t a_{(t-1)} - f\right)\right) \\ a_{t,j} &= \frac{a_{(t-1)j}\exp\left(-(\nabla a_{(t-1)})_j\right)}{\sum_i a_{(t-1)i}\exp\left(-(\nabla a_{(t-1)})_i\right)}, \end{aligned} \tag{3.19}$$

where $\eta, \bar{\eta}$ and $\tilde{\eta}$ are scalar factors to scale the argument of its associated influence function such that all can be constructed from the same Gaussian radial basis functions. The updated expression for the image u_t is almost identical to the non-blind case, see Equation (2.25) but the latest kernel estimate $a_{(t-1)}$ is used instead of the true one. Again, the image update is performed on a padded version of the input image defined as $u_{p(t-1)} = Pu_{(t-1)}$. Thus, also the truncation operation T is employed. In contrast to the simple kernel update scheme depicted in Equation (3.17), the step size moved into the computation of the kernel gradient $\nabla a_{(t-1)}$. Furthermore, the step size parameter τ_t was neglected as it can be reflected in the scale of the according influence functions $\tilde{\phi}_{ti}$. Nevertheless, the data dependency of the Lipschitz constant $L_2(u)$ must still be maintained within this approach. For a single element of the data term $U_t^\mathsf{T}\tilde{K}_{ti}^\mathsf{T}\tilde{\phi}_{ti}(\tilde{\eta}\tilde{K}_{ti}(U_t a_{(t-1)} - f))$ it can be computed by following the same idea as in the derivation of this method and is defined as

$$L_{2,i}(u_t) = \left|\left|\tilde{k}_{ti} * u_t\right|\right|_2^2.$$

This additional refinement of the step size already highlights one advantage of the modified approach, since in general the Lipschitz constant $L_{2,i}(u_t)$ is much smaller due to the filtering with a derivative kernel. Hence, larger steps are possible which speed up kernel estimation.

Although the influence functions and kernels of the two data terms are not shared, this approach can still be seen as gradient descent steps of related iteratively adapted energy minimization problems. The associated energies to stage t are defined by

$$E_t(u) = \frac{1}{N_k} \sum_{p=1}^{MN} \sum_{i=1}^{N_k} \rho_{ti}\left((K_{ti}u)_p\right) + \frac{1}{N_d} \sum_{p=1}^{M_f N_f} \sum_{i=1}^{N_d} \bar{\rho}_{ti}(\bar{K}_{ti}(A_{(t-1)}u - f))$$

$$E_t(a) = \delta_\Delta(a) + \frac{1}{N_d} \sum_{p=1}^{M_f N_f} \sum_{i=1}^{N_d} \tilde{\rho}_{ti}(\tilde{K}_{ti}(U_t a - f)),$$

whereby $M_f = M - H + 1$ and $N_f = N - W + 1$ if the image is of size $M \times N$ and the kernel of $H \times W$. In this formulation the penalty functions $\rho, \bar{\rho}$ and $\tilde{\rho}$ are the integrated version of their associated influence functions.

3.3.3.1 Training

In analogy to the non-blind deconvolution network training, the *IAM-BDC* net is trained in a discriminative fashion. Therefore, the network parameters Θ are learned such that a certain objective function is optimized. In this case it is formulated as

$$L(\Theta) = \sum_{s=1}^{S} \ell(u_T(\Theta, u_0^s, a_0^s), u_{gt}^s, a_T(\Theta, u_0^s, a_0^s), a_{gt}^s), \tag{3.20}$$

where one training sample consists of the blurry observation u_0^s, the initial blur estimate a_0^s, the true sharp image u_{gt}^s and the true blur kernel a_{gt}^s. In our case we minimize the squared error of the image and the blur kernel, hence the obejctive function is given by

$$\ell(u_T, u_{gt}, a_T, a_{gt}) = \frac{\alpha}{2} ||M(u_T - u_{gt})||_2^2 + \frac{\beta}{2} ||a_T - a_{gt}||_2^2, \tag{3.21}$$

whereby α and β are weights which can be adapted to balance between kernel and image fitting. The matrix M is a diagonal one that sets the error at border regions to zero, since every common border handling inserts inconsistencies. The network parameters Θ can be grouped into stages as in the non-blind case.

$$\Theta = \{\theta_t,\ t = 1 \dots T\}$$
$$\theta_t = \{(\phi_{ti}, k_{ti})\ i = 1 \dots N_k, (\bar{\phi}_{ti}, \bar{k}_{ti}), (\tilde{\phi}_{ti}, \tilde{k}_{ti})\ i = 1 \dots N_d\}$$

Thus, each stage is setup using the parameters θ_t, which hold the parametrization of the filters and influence functions of the regularization and both data terms of stage t.

Due to the massive number of parameters of the proposed net, an additional pre-training is necessary. Consequently, the training is once more split into a greedy pre-training phase followed by joint training.

Greedy pre-training: The pre-training phase is vital to steer the network parameters towards a good initialization to ease the successive joint training of the network. Therefore, we adopt the same greedy manner as in the non-blind case. Starting with a single stage, its parameters are tuned such that its output best fits under an objective function. Then the parameters are fixed and so is the output of this stage. Another stage is added on top of it and just the parameters of the newly added stage are tuned. This procedure is repeated until the final block depth is reached. Within this scheme, the stages are trained by minimizing

$$L(\theta_t) = \sum_{s=1}^{S} \ell(u_t^s, u_{gt}^s, a_t^s, a_{gt}^s). \tag{3.22}$$

As a result, each stage tries to modify the image and the blur estimate (u_t, a_t) such that they best match the true ones.

Joint training: After the greedy pre-training of a network block consisting of 5 stages, its parameters Θ are refined by jointly training all the block parameters. Hence, the parameter vector Θ is tuned such that the overall objective function, defined in Equation (3.20), is minimized. This approach enables the entire expressive power of the network since the intermediate results (u_t, a_t) are not constrained anymore. Therefore, they can be arbitrarily changed to improve the final output of the block. Thus, this stage is most important within the training framework.

3.3.3.2 Gradient derivation

In both training phases an objective function is minimized. The optimization of those objectives is done by applying the well known L-BFGS [29] algorithm as in the non-blind case. It is built upon the gradients of the objective function with respect to its parameters. Hence, the derivation of these gradients is mandatory for both pre-training and joint training.

Let us start with the greedy pre-training phase. The minimization of (3.22) requires the computation of the following gradient.

$$\frac{\partial L}{\partial \theta_t} = \sum_{s=1}^{S} \frac{\partial \ell(u_t^s, u_{gt}^s, a_t^s, a_{gt}^s)}{\partial \theta_t}. \tag{3.23}$$

For the sake of simplicity, we further analyze the gradient $\partial\ell/\partial\theta_t$ just for a single training sample, as the overall gradient can be computed by summation. Consequently, the derivative of the loss function ℓ of a single training sample with respect to the stage parameters is defined as

$$\frac{\partial\ell(u_t^s, u_{gt}^s, a_t^s, a_{gt}^s)}{\partial\theta_t} = \frac{\partial\ell}{\partial\theta_t} = \frac{\partial u_t}{\partial\theta_t}\alpha M^\mathsf{T} M(u_t - u_{gt}) + \frac{\partial a_t}{\partial\theta_t}\beta(a_t - a_{gt}).$$

Since, the kernel estimate a_t is a function of u_t, this gradient can be further refined to

$$\begin{aligned}
\frac{\partial\ell}{\partial\theta_t} &= \frac{\partial u_t}{\partial\theta_t}\alpha M(u_t - u_{gt}) + \left(\frac{\partial a_t}{\partial\theta_t} + \frac{\partial u_t}{\partial\theta_t}\frac{\partial a_t}{\partial u_t}\right)\beta(a_t - a_{gt}) \\
&= \frac{\partial u_t}{\partial\theta_t}\underbrace{\left(\alpha M(u_t - u_{gt}) + \frac{\partial a_t}{\partial u_t}\beta(a_t - a_{gt})\right)}_{e_{ut}} + \frac{\partial a_t}{\partial\theta_t}\underbrace{\beta(a_t - a_{gt})}_{e_{at}} \\
&= \frac{\partial u_t}{\partial\theta_t}e_{ut} + \frac{\partial a_t}{\partial\theta_t}e_{at},
\end{aligned}$$

whereby $\partial a_t/\partial\theta_t$ does not need to take care for the relation of u_t and θ_t anymore. The gradient $\partial a_t/\partial u_t$ can be derived as follows

$$\frac{\partial a_t}{\partial u_t} = \frac{\partial\nabla a_{(t-1)}}{\partial u_t}\frac{\partial a_t}{\partial\nabla a_{(t-1)}}. \tag{3.24}$$

In order to compute $\partial a_t/\partial\nabla a_{(t-1)}$ let us reformulate the kernel update rule, defined in Equation (3.19), to

$$a_t = \frac{1}{a_{(t-1)}^\mathsf{T} v}\operatorname{diag}\left(a_{(t-1)}\right)v,$$

where $v = \exp(-\nabla a_{(t-1)})$. Within this formulation the updated kernel a_t is constructed from the old kernel $a_{(t-1)}$, which forms a basis $\operatorname{diag}\left(a_{(t-1)}\right)$, followed by a normalization. Hence, $\frac{\partial a_t}{\partial\nabla a_{(t-1)}}$ can be computed as

$$\begin{aligned}
\frac{\partial a_t}{\partial\nabla a_{(t-1)}} &= \frac{\partial v}{\partial\nabla a_{(t-1)}}\frac{\partial a_t}{\partial v} \\
\frac{\partial a_t}{\partial v} &= \frac{1}{a_{(t-1)}^\mathsf{T} v}\left(I - \frac{a_{(t-1)}v^\mathsf{T}}{a_{(t-1)}^\mathsf{T} v}\right)\operatorname{diag}\left(a_{(t-1)}\right) \\
\frac{\partial v}{\partial\nabla a_{(t-1)}} &= \frac{\partial}{\partial\nabla a_{(t-1)}}\left(\exp(-\nabla a_{(t-1)})\right) = -\operatorname{diag}(v) \\
\frac{\partial a_t}{\partial\nabla a_{(t-1)}} &= -\operatorname{diag}(v)\frac{1}{a_{(t-1)}^\mathsf{T} v}\left(I - \frac{a_{(t-1)}v^\mathsf{T}}{a_{(t-1)}^\mathsf{T} v}\right)\operatorname{diag}\left(a_{(t-1)}\right).
\end{aligned} \tag{3.25}$$

If we compare this gradient to the derivative $\partial k_{ti}/\partial c_{ti}$ of the kernel filters in the none-blind case, we see that they resemble each other in terms of structure. This circumstance originates from the similar definition, as both are constructed from a basis and normalized vectors. Nevertheless, the missing gradient $\partial \nabla a_{(t-1)}/\partial u_t$ can be expressed as

$$\begin{aligned}\frac{\partial \nabla a_{(t-1)}}{\partial u_t} &= \frac{1}{N_d}\sum_{i=1}^{N_d}\frac{\partial}{\partial u_t}\left(\underbrace{\frac{1}{\left|\left|\tilde{K}_{ti}u_t\right|\right|_2^2}}_{g}\underbrace{U_t^\mathsf{T}\tilde{K}_{ti}^\mathsf{T}\tilde{\phi}_{ti}\left(\tilde{\eta}\tilde{K}_{ti}(U_t a_{(t-1)}-f)\right)}_{h}\right)\\ &= \frac{1}{N_d}\sum_{i=1}^{N_d}\left(g\frac{\partial h}{\partial u_t}+\frac{\partial g}{\partial u_t}h^\mathsf{T}\right).\end{aligned}$$

The gradient of the introduced variable h can be computed based on the results of the non-blind gradient derivation. Thus, $\partial h/\partial u_t$ is defined as

$$\frac{\partial h}{\partial u_t} = R_{180}^\mathsf{T}\left(\tilde{K}_{ti}^\mathsf{T}\tilde{\phi}_{ti}\left(\tilde{\eta}\tilde{K}_{ti}(U_t a_{(t-1)}-f)\right)\right)^\mathsf{T}+\tilde{\eta}A_{(t-1)}^\mathsf{T}\tilde{K}_{ti}^\mathsf{T}\tilde{\Lambda}_{ti}\tilde{K}_{ti}U_t,$$

where the diagonal matrix $\tilde{\Lambda}_{ti}$ holds the derivatives $\tilde{\phi}'_{ti}(\tilde{\eta}\tilde{K}_{ti}(U_t a_{(t-1)}-f))$ in its diagonal. The computation of $\partial g/\partial u_t$ is given by

$$\frac{\partial g}{\partial u_t} = \frac{\partial}{\partial u_t}\frac{1}{u_t^\mathsf{T}\tilde{K}_{ti}^\mathsf{T}\tilde{K}_{ti}u_t} = (-1)\frac{1}{(u_t^\mathsf{T}\tilde{K}_{ti}^\mathsf{T}\tilde{K}_{ti}u_t)^2}2\tilde{K}_{ti}^\mathsf{T}\tilde{K}_{ti}u_t = \frac{-2}{\left|\left|\tilde{k}_{ti}*u_t\right|\right|_2^4}\tilde{K}_{ti}^\mathsf{T}\tilde{K}_{ti}u_t.$$

By putting all the above derivatives together we get

$$\begin{aligned}\frac{\partial \nabla a_{(t-1)}}{\partial u_t} = \frac{1}{N_d}\sum_{i=1}^{N_d}\frac{1}{\left|\left|\tilde{K}_{ti}u_t\right|\right|_2^2}\Bigg\{&R_{180}^\mathsf{T}\left(\tilde{K}_{ti}^\mathsf{T}\tilde{\phi}_{ti}\left(\tilde{\eta}\tilde{K}_{ti}(U_t a_{(t-1)}-f)\right)\right)^\mathsf{T}+\tilde{\eta}A_{(t-1)}^\mathsf{T}\tilde{K}_{ti}^\mathsf{T}\tilde{\Lambda}_{ti}\tilde{K}_{ti}U_t-\ldots\\ &\left(\frac{2}{\left|\left|\tilde{k}_{ti}*u_t\right|\right|_2^2}\tilde{K}_{ti}^\mathsf{T}\tilde{K}_{ti}u_t\right)\left(\tilde{\phi}_{ti}\left(\tilde{\eta}\tilde{K}_{ti}(U_t a_{(t-1)}-f)\right)\right)^\mathsf{T}\tilde{K}_{ti}U_t\Bigg\}.\end{aligned}\quad(3.26)$$

Thus all parts of the derivative $\partial a_t/\partial u_t$ are defined. We do not state the complete gradient explicitly since it would span multiple lines. It can be constructed by multiplying all the above parts.

After this exhausting derivation, we can finally compute the derivatives of a_t and u_t with respect to the stage parameters. The gradient of u_t with respect to the regularization

parameters have already been computed in the non-blind case and are given by

$$\frac{\partial u_t}{\partial c_{ti}} = -\frac{1}{N_k \left|\left|c_{ti}\right|\right|_2}\left(I - \frac{c_{ti}c_{ti}^\mathsf{T}}{c_{ti}^\mathsf{T}c_{ti}}\right) B^\mathsf{T}\left(R_{180}^\mathsf{T} H^\mathsf{T} + \eta U_{p(t-1)}^\mathsf{T}\Lambda_{ti}K_{ti}\right)T^\mathsf{T} \tag{3.27}$$

$$\frac{\partial u_t}{\partial w_{ti}} = -\frac{1}{N_k}\Phi^\mathsf{T}(K_{it}u_{(t-1)})K_{ti}T^\mathsf{T}, \tag{3.28}$$

where $H = \phi_{ti}(\eta K_{ti}u_{p(t-1)})$, $\Lambda_{ti} = \operatorname{diag}\left(\phi'_{ti}(\eta K_{ti}u_{p(t-1)})\right)$ and $\Phi(K_{it}u_{p(t-1)})$ is the basis matrix the influence functions are created from. Note that the kernels k_{ti} are parametrized by c_{ti} and the influence functions are constructed using the weights w_{ti}. Furthermore, the gradient with respect to the data term filters $\bar{k}_{ti}$ and its according influence function $\bar{\phi}_{ti}$ have also been computed beforehand and are defined as

$$\frac{\partial u_t}{\partial \bar{c}_{ti}} = -\frac{1}{N_d \left|\left|\bar{c}_{ti}\right|\right|_2}\left(I - \frac{\bar{c}_{ti}\bar{c}_{ti}^\mathsf{T}}{\bar{c}_{ti}^\mathsf{T}\bar{c}_{ti}}\right) B_d^\mathsf{T}\left(R_{180}^\mathsf{T} H^\mathsf{T} + \bar{\eta}\left(A_{(t-1)}u_{p(t-1)} - f\right)^\mathsf{T}\bar{\Lambda}_{ti}\bar{K}_{ti}\right)A_{(t-1)} \tag{3.29}$$

$$\frac{\partial u_t}{\partial \bar{w}_{ti}} = -\frac{1}{N_d}\Phi^\mathsf{T}\left(\bar{\eta}\bar{K}_{ti}(A_{(t-1)}u_{p(t-1)} - f)\right)\bar{K}_{ti}A_{(t-1)}T^\mathsf{T}, \tag{3.30}$$

whereby $H = \bar{\phi}_{ti}(\bar{\eta}\bar{K}_{ti}(A_{(t-1)}u_{p(t-1)} - f))$, $\bar{\Lambda}_{ti} = \operatorname{diag}\left(\bar{\phi}'_{ti}(\bar{\eta}\bar{K}_{ti}(A_{(t-1)}u_{p(t-1)} - f))\right)$ and $\Phi(\bar{\eta}\bar{K}_{ti}(A_{(t-1)}u_{p(t-1)} - f))$ is the matrix representation of the Gaussian radial basis functions. We compute the gradients with respect to $\bar{c}_{ti}$ and $\bar{w}_{ti}$ because the data term filters $\bar{k}_{ti}$ and influence functions $\bar{\phi}_{ti}$ are parametrized by those. As a result it remains to compute $\partial a_t/\partial\tilde{k}_{ti}$ and $\partial a_t/\partial\tilde{\phi}_{ti}$ because u_t is not influenced by those parameters. Moreover, the influence of a_t on the parameters (c_{ti}, w_{ti}) and $(\bar{c}_{ti}, \bar{w}_{ti})$ is already handled within the error e_{ut}.

Data term filters $\tilde{k}_{ti}$: Since $\partial u_t/\partial\tilde{k}_{ti} = 0$, we just have to compute

$$\frac{\partial a_t}{\partial\tilde{k}_{ti}} = \frac{\partial\nabla a_{(t-1)}}{\partial\tilde{k}_{ti}}\frac{\partial a_t}{\partial\nabla a_{(t-1)}}.$$

The gradient $\partial a_t/\partial\nabla a_{(t-1)}$ has already been computed, see Equation (3.25) and the remaining gradient can be computed in a similar manner as $\partial\nabla a_{(t-1)}/\partial u_t$ and is given by

$$\frac{\partial\nabla a_{(t-1)}}{\partial\tilde{k}_{ti}} = \frac{1}{N_d}\sum_{i=1}^{N_d}\frac{1}{\left|\left|\tilde{K}_{ti}u_t\right|\right|_2^2}\Bigg\{R_{180}^\mathsf{T}\left(\tilde{\phi}_{ti}\left(\tilde{\eta}\tilde{K}_{ti}(U_t a_{(t-1)} - f)\right)\right)^\mathsf{T}U_t + \tilde{\eta}(U_t a_{(t-1)} - f)^\mathsf{T}\tilde{\Lambda}_{ti}\tilde{K}_{ti}U_t - \ldots$$
$$\left(\frac{2}{\left|\left|\tilde{K}_{ti}u_t\right|\right|_2^2}U_t^\mathsf{T}U_t\tilde{k}_{ti}\right)\left(\tilde{\phi}_{ti}\left(\tilde{\eta}\tilde{K}_{ti}(U_t a_{(t-1)} - f)\right)\right)^\mathsf{T}\tilde{K}_{ti}U_t\Bigg\}. \tag{3.31}$$

Akin to the other data term filters $\bar{k}_{ti}$, this kernels are constructed from a discrete cosine transform basis in order to limit their norm. Thus, they are defined as

$$\tilde{k}_{ti} = B_d \frac{\tilde{c}_{ti}}{||\tilde{c}_{ti}||_2}.$$

Consequently, the training parameter changes to $\tilde{c}_{ti}$ and we need to compute $\partial a_t / \partial \tilde{c}_{ti}$. Based on the above results and the derivations in Section 2.2.1.2 this gradient can be expressed as

$$\begin{aligned}
\frac{\partial a_t}{\partial \tilde{c}_{ti}} &= \frac{\partial \tilde{k}_{ti}}{\partial \tilde{c}_{ti}} \frac{\partial \nabla a_{(t-1)}}{\partial \tilde{k}_{ti}} \frac{\partial a_t}{\partial \nabla a_{(t-1)}} \\
\frac{\partial \tilde{k}_{ti}}{\partial \tilde{c}_{ti}} &= \frac{1}{||\tilde{c}_{ti}||_2} \left(I - \frac{\tilde{c}_{ti} \tilde{c}_{ti}^\mathsf{T}}{\tilde{c}_{ti}^\mathsf{T} \tilde{c}_{ti}} \right) B_d^\mathsf{T} \\
\frac{\partial a_t}{\partial \tilde{c}_{ti}} &= \frac{1}{||\tilde{c}_{ti}||_2} \left(I - \frac{\tilde{c}_{ti} \tilde{c}_{ti}^\mathsf{T}}{\tilde{c}_{ti}^\mathsf{T} \tilde{c}_{ti}} \right) B_d^\mathsf{T} \frac{\partial \nabla a_{(t-1)}}{\partial \tilde{k}_{ti}} \frac{\partial a_t}{\partial \nabla a_{(t-1)}}.
\end{aligned} \tag{3.32}$$

So, the gradient with respect to the data term filter parameters $\tilde{c}_{ti}$ can be computed by first propagating the error to the level of $\nabla a_{(t-1)}$ and then computing the influence of the filters.

Data term influence functions $\tilde{\phi}_{ti}$: This gradient can be computed in the same fashion. Since the influence functions $\tilde{\phi}_{ti}(x)$ are parametrized by a weight vector $\tilde{w}_{ti}$, we have to compute the gradient $\partial a_t / \partial \tilde{w}_{ti}$, which is expressed as

$$\frac{\partial a_t}{\partial \tilde{w}_{ti}} = \frac{\partial \nabla a_{(t-1)}}{\partial \tilde{w}_{ti}} \frac{\partial a_t}{\partial \nabla a_{(t-1)}}, \tag{3.33}$$

where the gradient $\partial \nabla a_{(t-1)} / \partial \tilde{w}_{ti}$ can be computed as follows

$$\begin{aligned}
\frac{\partial \nabla a_{(t-1)}}{\partial \tilde{w}_{ti}} &= \frac{1}{N_d} \frac{\partial}{\partial \tilde{w}_{ti}} \left(\sum_{i=1}^{N_d} U_t^\mathsf{T} \tilde{K}_{ti}^\mathsf{T} \Phi \left(\tilde{\eta} \tilde{K}_{ti} (U_t a_{(t-1)} - f) \right) \tilde{w}_{ti} \right) \\
&= \Phi^\mathsf{T} \left(\tilde{\eta} \tilde{K}_{ti} (U_t a_{(t-1)} - f) \right) \tilde{K}_{ti} U_t.
\end{aligned}$$

This gradient concludes the derivation of the required gradients for the greedy pre-training phase.

Joint training: Also in the joint training phase the L-BFGS algorithm is used to minimize (3.23). For the same reasons as in the pre-training, we consider just a single training example. Therefore, we have to compute the derivative $\partial \ell(u_T, u_{gt}, a_T, a_{gt}) / \partial \Theta$ with T

being the output of the current block, which is defined as

$$\frac{\partial \ell(u_T, u_{gt}, a_T, a_{gt})}{\partial \Theta} = \frac{\partial \ell}{\partial \Theta} = \frac{\partial u_T}{\partial \Theta} \underbrace{\alpha M(u_t - u_{gt})}_{e_{uT}} + \frac{\partial a_T}{\partial \Theta} \underbrace{\beta(a_t - a_{gt})}_{e_{aT}} = \frac{\partial u_T}{\partial \Theta} e_{uT} + \frac{\partial a_T}{\partial \Theta} e_{aT}.$$

The direct derivation of these gradients is difficult, thus we employ a divide and conquer approach. For the moment we are just interested in computing the gradient with respect to the parameters of stage t, hence the gradient changes to

$$\frac{\partial \ell(u_T, u_{gt}, a_T, a_{gt})}{\partial \theta_t} = \frac{\partial u_T}{\partial \theta_t} e_{uT} + \frac{\partial a_T}{\partial \theta_t} e_{aT}. \tag{3.34}$$

Based on the stage update rule we know that u_T is a function of $u_{(T-1)}$ and $a_{(T-1)}$ and a_T depends on u_T and $a_{(T-1)}$. Thus, the gradients with respect to θ_t can be refined to

$$\begin{aligned}
\frac{\partial u_T}{\partial \theta_t} &= \frac{\partial u_{(T-1)}}{\partial \theta_t} \frac{\partial u_T}{\partial u_{(T-1)}} + \frac{\partial a_{(T-1)}}{\partial \theta_t} \frac{\partial u_T}{\partial a_{(T-1)}} \\
\frac{\partial a_T}{\partial \theta_t} &= \frac{\partial u_T}{\partial \theta_t} \frac{\partial a_T}{\partial u_T} + \frac{\partial a_{(T-1)}}{\partial \theta_t} \frac{\partial a_T}{\partial a_{(T-1)}}.
\end{aligned}$$

If we plug this result into Equation (3.34), we get

$$\begin{aligned}
\frac{\partial \ell}{\partial \theta_t} &= \frac{\partial u_T}{\partial \theta_t} e_{uT} + \frac{\partial a_T}{\partial \theta_t} e_{aT} \\
&= \frac{\partial u_T}{\partial \Theta} e_{uT} + \left(\frac{\partial u_T}{\partial \theta_t} \frac{\partial a_T}{\partial u_T} + \frac{\partial a_{(T-1)}}{\partial \theta_t} \frac{\partial a_T}{\partial a_{(T-1)}} \right) e_{aT} \\
&= \frac{\partial u_T}{\partial \theta_t} \underbrace{\left(e_{uT} + \frac{\partial a_T}{\partial u_T} e_{aT} \right)}_{\omega} + \frac{\partial a_{(T-1)}}{\partial \theta_t} \frac{\partial a_T}{\partial a_{(T-1)}} e_{aT} \\
&= \left(\frac{\partial u_{(T-1)}}{\partial \theta_t} \frac{\partial u_T}{\partial u_{(T-1)}} + \frac{\partial a_{(T-1)}}{\partial \theta_t} \frac{\partial u_T}{\partial a_{(T-1)}} \right) \omega + \frac{\partial a_{(T-1)}}{\partial \theta_t} \frac{\partial a_T}{\partial a_{(T-1)}} e_{aT} \\
&= \frac{\partial u_{(T-1)}}{\partial \theta_t} \underbrace{\frac{\partial u_T}{\partial u_{(T-1)}} \omega}_{e_{u(T-1)}} + \frac{\partial a_{(T-1)}}{\partial \theta_t} \underbrace{\left(\frac{\partial u_T}{\partial a_{(T-1)}} \omega + \frac{\partial a_T}{\partial a_{(T-1)}} e_{aT} \right)}_{e_{a(T-1)}} \\
&= \frac{\partial u_{(T-1)}}{\partial \theta_t} e_{u(T-1)} + \frac{\partial a_{(T-1)}}{\partial \theta_t} e_{a(T-1)}.
\end{aligned}$$

The same derivation can be repeated if the dependencies of $u_{(T-1)}$ and $a_{(T-1)}$ on $u_{(T-2)}$ and $a_{(T-2)}$ are inserted. The result would be the same, with updated indexes though. This procedure can be stopped as soon as stage t is reached, since the images u_i and kernels a_i

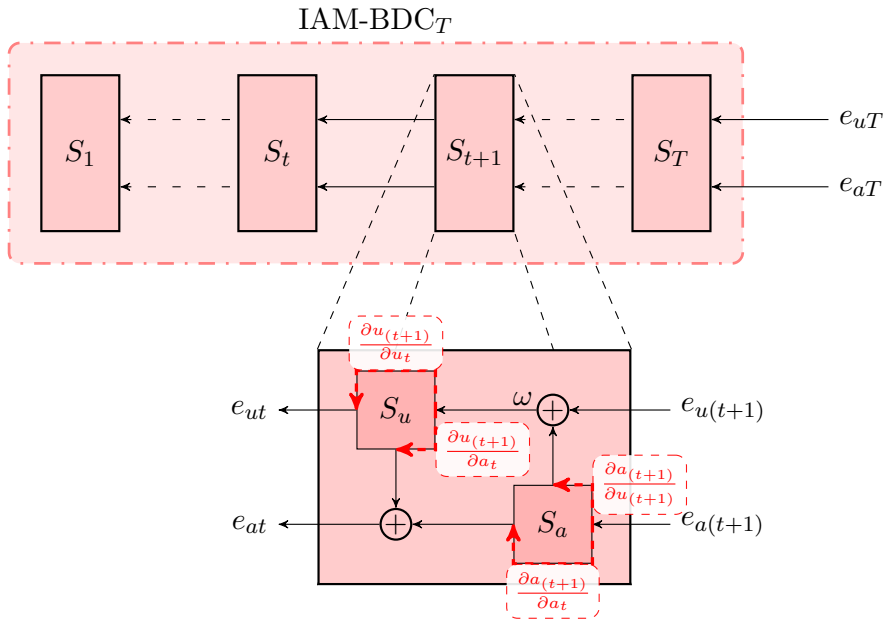


Figure 3.6: Illustration of the error back propagation within an *IAM-BDC* net. The errors at the stage e_{uT} and e_{aT} are back propagated through the network by following the reverse path of the stage updates. Furthermore, inside the stage updates the error is distributed by following the reverse information flow.

for $i < t$ do not depend on θ_t. So, we end up with the following error propagation rules:

$$
\begin{aligned}
e_{ut} &= \frac{\partial u_{(t+1)}}{\partial u_t}\omega \\
e_{at} &= \frac{\partial u_{(t+1)}}{\partial a_t}\omega + \frac{\partial a_{(t+1)}}{\partial a_t}e_{a(t+1)} \\
\omega &= e_{u(t+1)} + \frac{\partial a_{(t+1)}}{\partial u_{(t+1)}}e_{a(t+1)}
\end{aligned}
$$

Compared to the non-blind error propagation, this result is much more evolved. Figure 3.6 illustrates the error propagation through the *IAM-BDC* network. The error at the stage T is pushed into the back propagation scheme and the output of the algorithm is an error at each stage t, which denotes the portion of the error generated by this stage. The error propagation has to follow exactly the reverse paths as in the forward computation in order to distribute the errors correctly. Thus, it has to take both paths for the image and kernel update S_u and S_a as illustrated by the red arrows. In contrast to the Iteratively Adapted Minimization for non-blind Deconvolution (IAM-DC) net, the error propagation requires the computation of four gradients. Fortunately, some of those gradients are closely related to derivatives we already computed before. The gradient $\partial a_{(t+1)}/\partial u_{(t+1)}$

was earlier derived within the greedy training stage and can be computed as

$$\frac{\partial a_{(t+1)}}{\partial u_{(t+1)}} = \frac{\partial \nabla a_t}{\partial u_{(t+1)}} \frac{\partial a_{(t+1)}}{\partial \nabla a_t}$$
$$\frac{\partial a_{(t+1)}}{\partial \nabla a_t} = -\operatorname{diag}(v) \frac{1}{a_t^\mathsf{T} v} \left(I - \frac{a_t v^\mathsf{T}}{a_t^\mathsf{T} v} \right) \operatorname{diag}(a_t)$$

$$\frac{\partial \nabla a_t}{\partial u_{(t+1)}} = \frac{1}{N_d} \sum_{i=1}^{N_d} \frac{1}{\left|\left| \tilde{K}_{(t+1)i} u_t \right|\right|_2^2} \Bigg\{ R_{180}^\mathsf{T} \left(\tilde{K}_{(t+1)i}^\mathsf{T} \tilde{\phi}_{(t+1)i} \left(\tilde{\eta} \tilde{K}_{(t+1)i} (U_{(t+1)} a_t - f) \right) \right)^\mathsf{T} + \dots$$
$$\tilde{\eta} A_t^\mathsf{T} \tilde{K}_{(t+1)i}^\mathsf{T} \tilde{\Lambda}_{(t+1)i} \tilde{K}_{(t+1)i} U_{(t+1)} - \dots$$
$$\left(\frac{2}{\left|\left| \tilde{k}_{(t+1)i} * u_{(t+1)} \right|\right|_2^2} \tilde{K}_{(t+1)i}^\mathsf{T} \tilde{K}_{(t+1)i} u_{(t+1)} \right) \left(\tilde{\phi}_{(t+1)i} \left(\tilde{\eta} \tilde{K}_{(t+1)i} (U_{(t+1)} a_t - f) \right) \right)^\mathsf{T} \tilde{K}_{(t+1)i} U_{(t+1)} \Bigg\},$$

where $v = \exp(-\nabla a_t)$ and $\tilde{\Lambda}_{(t+1)i} = \operatorname{diag}\left(\tilde{\phi}'_{(t+1)i} (\tilde{\eta} \tilde{K}_{(t+1)i} (U_{(t+1)} a_t - f)) \right)$. Based on the gradients of the joint training in the non-blind case, we can easily derive $\partial u_{(t+1)} / \partial u_t$, which is defined as

$$\frac{\partial u_{(t+1)}}{\partial u_t} = P^\mathsf{T} \left\{ I - \left(\frac{1}{N_k} \sum_{i=1}^{N_k} \eta K_{(t+1)i}^\mathsf{T} \Lambda_{(t+1)i} K_{(t+1)i} + \frac{1}{N_d} \sum_{i=1}^{N_d} \bar{\eta} A_t^\mathsf{T} \bar{K}_{(t+1)i}^\mathsf{T} \bar{\Lambda}_{(t+1)i} \bar{K}_{(t+1)i} A_t \right) \right\} T^\mathsf{T},$$

where $\Lambda_{(t+1)i}$ and $\bar{\Lambda}_{(t+1)i}$ are diagonal matrices which hold the derivatives of the influence functions $\phi'_{(t+1)i}(\eta u_{tp} * k_{(t+1)i})$ and $\bar{\phi}'_{(t+1)i}(\bar{\eta}(u_{tp} * a_t - f) * \bar{k}_{(t+1)i})$ respectively.

The two missing gradients are required to propagate the kernel error back. The gradient $\partial u_{(t+1)} / \partial a_t$ can be derived analogue to the data term filters $\bar{k}_{ti}$ and is given by

$$\frac{\partial u_{(t+1)}}{\partial a_t} = -\frac{1}{N_d} \left\{ \sum_{i=1}^{N_d} R_{180}^\mathsf{T} \left(\bar{\phi}_{(t+1)i} \left(\bar{\eta} \bar{K}_{(t+1)i} (A_t u_{tp} - f) \right) \right)^\mathsf{T} \bar{K}_{(t+1)i} + \bar{\eta} U_t^\mathsf{T} \bar{K}_{(t+1)i}^\mathsf{T} \bar{\Lambda}_{(t+1)i} \bar{K}_{(t+1)i} A_t \right\} T^\mathsf{T}.$$

In order to derive the remaining derivative $\partial a_{(t+1)} / \partial a_t$ let us consider to following variants of the stage kernel update

$$a_{(t+1)} = \frac{1}{a_t^\mathsf{T} v} \operatorname{diag}(a_t)\, v = \frac{1}{a_t^\mathsf{T} v} \operatorname{diag}(v)\, a_t,$$

whereby $v = \exp(-\nabla a_t)$. Since both $a_{(t+1)}$ and v are functions of a_t, we have to apply the chain rule once more.

$$\frac{\partial a_{(t+1)}}{\partial a_t} = \frac{\partial a_t}{\partial a_t} \frac{\partial a_{(t+1)}}{\partial a_t} + \frac{\partial \nabla a_t}{\partial a_t} \frac{\partial v}{\partial \nabla a_t} \frac{\partial a_{(t+1)}}{\partial v}$$

The first part of the chain rule looks a bit unfamiliar but it states the derivative with respect to a_t for a constant v. It can be easily computed by using the second variant of

the kernel update rule.

$$\frac{\partial a_{(t+1)}}{\partial a_t} = \frac{1}{a_t^\mathsf{T} v}\left(I - \frac{v a_t^\mathsf{T}}{a_t^\mathsf{T} v}\right)\operatorname{diag}(v)$$

To compute the second part of the chain rule, we can build upon the previous results. However, we still have to calculate $\partial \nabla a_t / \partial a_t$, which can be computed similarly to the data term kernels $\tilde{k}_{ti}$ and is defined as

$$\frac{\partial \nabla a_t}{\partial a_t} = \frac{1}{N_d}\sum_{i=1}^{N_d} \frac{1}{\left|\left|\tilde{k}_{(t+1)i} * u_t\right|\right|_2^2} \tilde{\eta} U_t^\mathsf{T} \tilde{K}_{(t+1)i}^\mathsf{T} \tilde{\Lambda}_{(t+1)i} \tilde{K}_{(t+1)i} U_t.$$

Eventually, we can write down the last missing gradient.

$$\frac{\partial a_{(t+1)}}{\partial a_t} = \frac{1}{a_t^T v}\left(I - \frac{v a_t^T}{a_t^T v}\right)\operatorname{diag}(v) - \frac{\partial \nabla a_t}{\partial a_t}\operatorname{diag}(v)\frac{1}{a_t^T v}\left(I - \frac{a_t v^T}{a_t^T v}\right)\operatorname{diag}(a_t)$$

As a result, the network can be trained jointly by first propagating the error information though all stages, which yields e_{ut} and e_{at}. Then the gradients of the individual stage parameters can be computed in the same way as in the greedy pre-training using the propagated errors as input.

3.3.3.3 Evaluation

To demonstrate the influence of different hyper-parameters of the proposed network such as net depth and kernel size, we trained all variants on the same training dataset. It consists of 200 training samples of the form $(u_0^s, u_{gt}^s, f^s, a_0^s, a_{gt}^s)$, where u_0^s is the initial image estimate, u_{gt}^s the true sharp image,f^s the blurry observation, a_0^s the initial blur kernel estimate and a_{gt}^s the true blur kernel. Half of this samples use the blurry observation as initial image estimate $u_0^s = f^s$ and the blur kernel estimate of Xu and Jia [47]. The rest of the training is initialized using the ground truth data $u_0^s = u_{gt}^s$, $a_0^s = a_{gt}^s$. The second half of the training set helps to guide the network parameters to favor natural sharp images and avoid delta blur kernels. All the training sample images consist of patches extracted from the BSD500 dataset [30] and the blur kernels are randomly sampled from the dataset proposed by Schelten et al. [39]. Note that the initial blur kernel estimates of Xu and Jia's method [47] are computed using the entire image instead of the cropped version.

Similar to the non-blind case, we use $N_w = 63$ Gaussian radial basis functions to form the basis for all influence functions for all trained networks. They are uniformly placed in the interval $[-1, 1]$ and have a standard deviation of $\gamma = 2/N_w$. To account for the different argument scales, the influence function argument parameters are set to $\eta = 1/1.25$, $\bar{\eta} = 1$ and $\tilde{\eta} = 1$. The influence functions of the regularization are initialized by the gradient of the heavy tailed function $\log(1 + \alpha x^2)$ and the filters k_{ti} were initially defined by the i-th basis if the corresponding discrete cosine transform basis. The initial

	IAM-BDC$_{10}^{3\times3}$		IAM-BDC$_{15}^{3\times3}$		IAM-BDC$_{20}^{3\times3}$	
Dataset	PSNR	σ	PSNR	σ	PSNR	σ
Levin	28.621	1.407	28.665	1.378	28.670	1.373
BSD500	28.436	3.382	28.582	3.413	28.583	3.410

Table 3.2: Comparison of the experimental results for the proposed *IAM-BDC*$^{3\times3}$ nets. Depicted are the average PSNR(dB) and the corresponding standard deviation σ for the dataset of Levin et al. [28] and 28 image samples from the BSD500 dataset [30] convolved with the kernels of Levin. All the network filters $k_{ti}, \bar{k}_{ti}$ and $\tilde{k}_{ti}$ are of size 3×3, however the depth of the nets varies, e.g. the IAM-BDC$_{10}^{3\times3}$ consists of 10 stages.

	IAM-BDC$_{10}^{5\times5}$		IAM-BDC$_{15}^{5\times5}$		IAM-BDC$_{20}^{5\times5}$	
Dataset	PSNR	σ	PSNR	σ	PSNR	σ
Levin	28.746	1.301	28.820	1.264	28.785	1.247
BSD500	28.633	3.479	28.779	3.520	28.825	3.521

Table 3.3: Comparison of the experimental results for the proposed *IAM-BDC*$^{5\times5}$ nets. Depicted are the average PSNR(dB) and the corresponding standard deviation σ for the dataset of Levin et al. [28] and 28 image samples from the BSD500 dataset [30] convolved with the kernels of Levin. All the network filters $k_{ti}, \bar{k}_{ti}$ and $\tilde{k}_{ti}$ are of size 5×5, however the depth of the nets varies.

influence functions of the data term $\bar{\phi}_{ti}$ and $\tilde{\phi}_{ti}$ were set to follow the gradient of $1/2x^2$. Moreover, the filters $\bar{k}_{ti}$ and $\tilde{k}_{ti}$ were also initialized by the i-th basis vector of the related discrete cosine transform basis. However, instead of the constant basis vector the delta kernel is used. The padding operation associated with the matrix P is in analogy to the non-blind case defined by replicating its border pixels. The corresponding truncation operator T removes this padded pixels from the intermediate results. Furthermore, it is ensured that the intermediate images have intensity values in $[0, 1]$ by truncating those outside.

The so configured networks were then trained as described in Section 3.3.3.1. We set $\alpha = 1$ and $\beta = 100$ in the loss function of a single training sample, see Equation (3.21), to balance between image and kernel fitting. The greedy pre-training of the stage parameters θ_t was done by performing 100 iterates of L-BFGS. The subsequent joint training of a network block, consisting of 5 stages, involved another 100 L-BFGS iterates. This alternating training strategy was repeated until the final network depth was reached.

The trained networks were evaluated on the dataset of Levin et al. [28]. The blurry observations of this dataset have a spatially varying blur due to their acquisition process. To evaluate also the performance for uniform blur, we generated a test dataset consisting of random samples of the BSD500 validation set [30] and the blur kernels of Levin et al. [28]. We added 0.2% white Gaussian noise just as in the generation of the training data. Both test datasets are constructed from images and kernels which are not included in the training data. The resulting average Peak Signal Noise Ratio (PSNR) performances and the associated standard deviation across both test sets are shown in Table 3.2 and 3.3. While Table 3.2 depicts the results of various IAM-BDC$^{3\times3}$ networks, Table 3.3 states

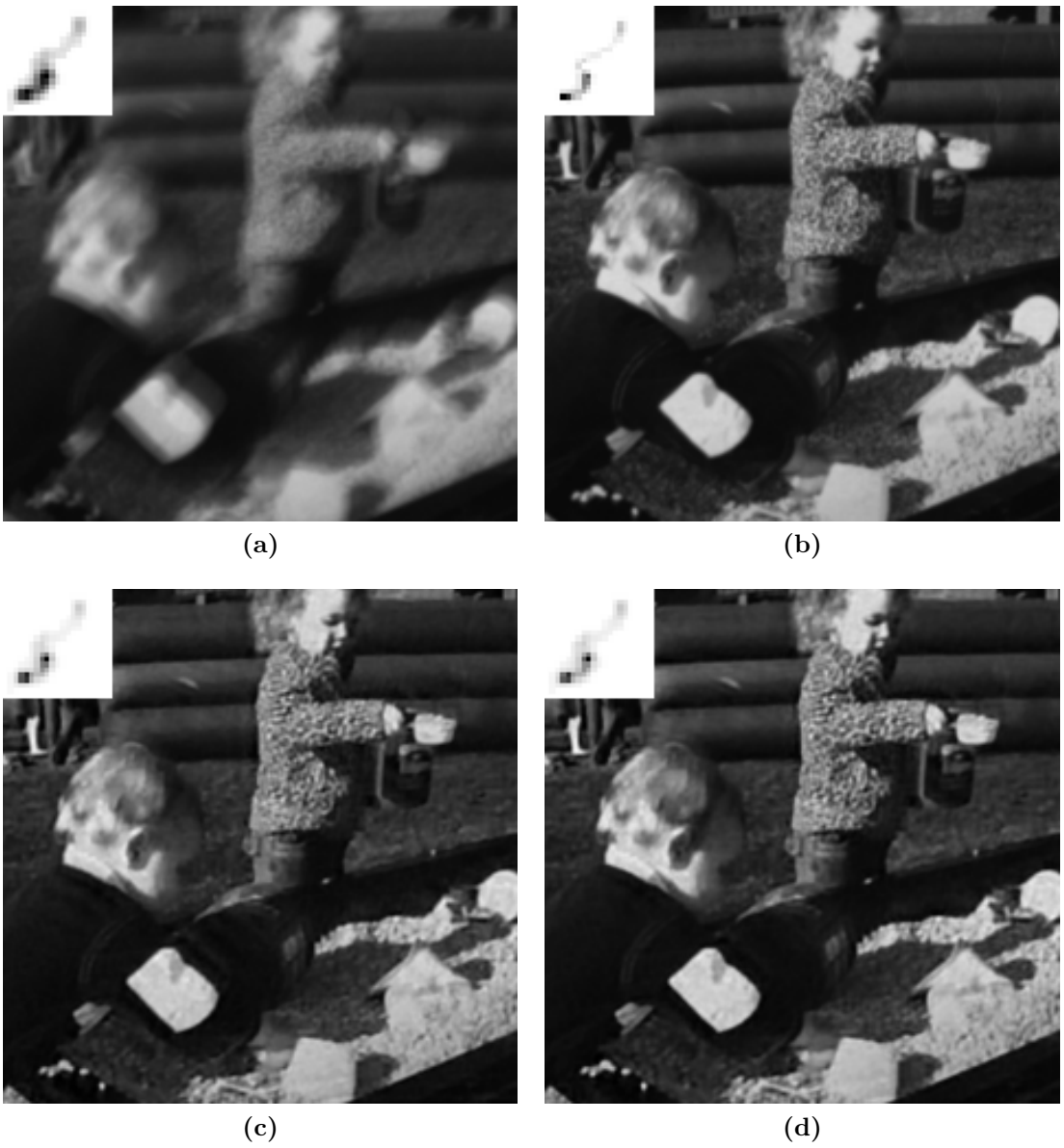


Figure 3.7: Blind deconvolution results for image 1 and kernel 6 of the dataset of Levin et al. [28]. (a) blurry observation with initial blur kernel estimate of Xu and Jia [47] in the upper left corner. (b) ground truth sharp image together with true blur kernel. Deconvolution results of (c) IAM-DC$_{20}^{3\times3}$ and (d) IAM-DC$_{20}^{5\times5}$.

those of the 5×5 nets, which have a larger filter support. The average *PSNR* of the 3×3 nets stays almost constant with increasing net depth. For the dataset of Levin et al. there is just a slight increase of 0.05dB and the average *PSNR* of the 28 BSD500 images increases just by 0.15dB. The average *PSNR* values of the 5×5 networks also increases slightly along with the net depth. For the BSD500 it increases up to 0.2dB if the network

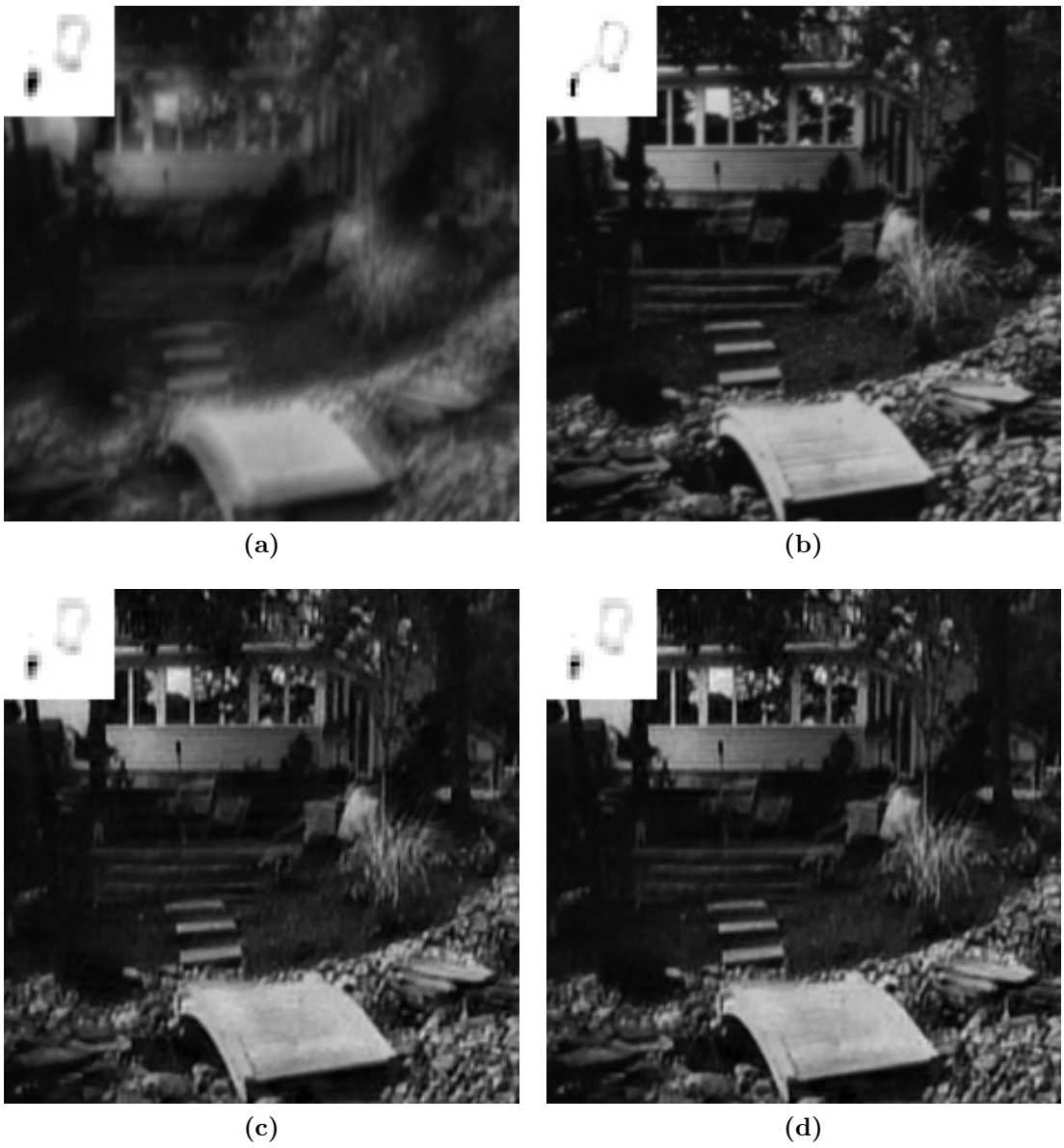


Figure 3.8: Blind deconvolution results for image 2 and kernel 7 of the dataset of Levin et al. [28]. (a) blurry observation with initial blur kernel estimate of Xu and Jia [47] in the upper left corner. (b) ground truth sharp image together with true blur kernel. Deconvolution results of (c) IAM-BDC$_{20}^{3\times3}$ and (d) IAM-BDC$_{20}^{5\times5}$.

depth is increased from 10 to 20 stages. For the test set of Levin et al. [28] the average *PSNR* increases by almost 0.08dB from 10 stages to 15. However, if the depth is further increased, the average *PSNR* values decrease. We later show that our network produces sharper edges than the ground truth images in the test set of Levin and colleagues. The standard deviation σ is almost identical for the investigated network depths. If we compare

Figure 3.9: Blind deconvolution results for a second sample of the generated BSD500 test set. The blurry observation (a) was constructed by using the second kernel of Levin's dataset. The initial blur kernel estimate of Xu and Jia [47] is added in the upper left corner. (b) ground truth sharp image together with true blur kernel. Deconvolution results of (c) IAM-BDC$_{20}^{3\times3}$ and (d) IAM-BDC$_{20}^{5\times5}$.

the resulting *PSNR* values to the scores of the non-blind deconvolution results, we see that the non-blind scores are by a large margin better. The performance difference originates from the much higher complexity of the blind image deconvolution problem. Furthermore, here we use the original dataset of Levin et al. [28], whose blurry observations are non-uniformly blurred. Nevertheless, the resulting images and the blur kernel estimates confirm that the iteratively adapted networks are also suitable for blind image deconvolution, see Figures 3.7 to 3.12. The outputs of the IAM-BDC$_{20}^{5\times5}$ net seem to have even sharper edges than the ground truth images in the dataset of Levin et al. [28]. For instance the face of the girl in Figure 3.7 and her pants are much sharper or the background in Figure 3.8 posses sharper edges than the ground truth. This difference reduces the *PSNR* scores, although from a human point of view the results of our networks seem superior.

Despite the relatively coarse initial blur kernel estimates of Xu and Jia [47], the deconvolved images and the refined blur kernels of the *IAM-BDC* nets are of remarkable quality. However, some ringing artifacts occur in the deconvolved images, for instance to the left of

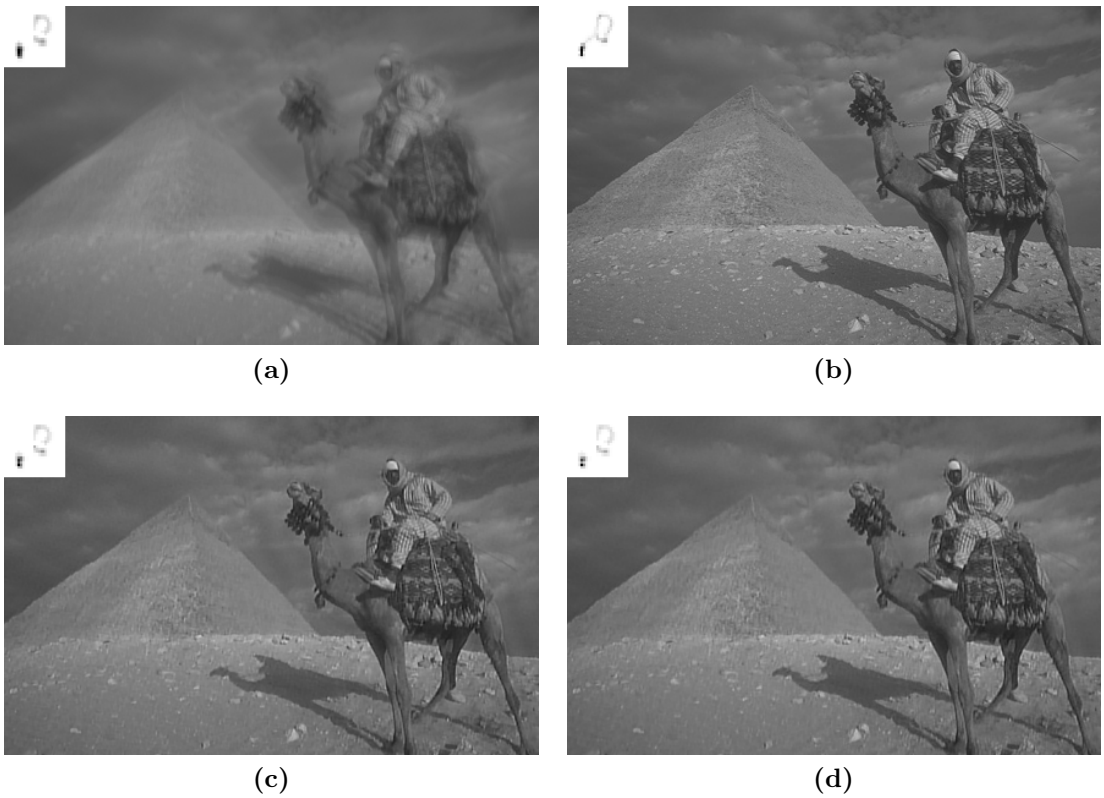


Figure 3.10: Blind deconvolution results for another sample image of the BSD500 validation set. It was generated using the 7-th kernel of the dataset proposed by Levin et al. [28]. (a) blurry observation with initial blur kernel estimate of Xu and Jia [47] in the upper left corner. (b) ground truth sharp image together with true blur kernel. Deconvolution results of (c) IAM-BDC$_{20}^{3\times3}$ and (d) IAM-BDC$_{20}^{5\times5}$.

the children's jacket in Figure 3.7c or at the wings of the plane in Figure 3.9c. Note that the majority of this ringing artifacts are suppressed in the output of the IAM-BDC$_{20}^{5\times5}$ networks, see Figure 3.7d and 3.9d. Consequently, larger filters help to avoid ringing artifacts. Also fine image structures such as the leftmost propeller blade in Figure 3.9 or the fine edges of the wooden slats of the bridge in Figure 3.8 are filtered out along with the ringing artifacts.

Moreover, the plane test image has a bad initial blur estimate because the blurry observation, depicted in Figure 3.9a has very few large step edges, which are favored by the approach of Xu and Jia [47]. Despite that, the initial blur kernel estimate is refined such that the output is closer to the ground truth. This effect can also be seen in Figures 3.7, 3.8 and 3.12. The blur kernel estimates are refined to better meet the main properties of the true blur kernels. One very successfully deconvolved sample is depicted in Figure 3.10. There is almost no difference visible between the network outputs and the ground truth image despite the imperfect blur kernel estimate. The same observation is valid for the

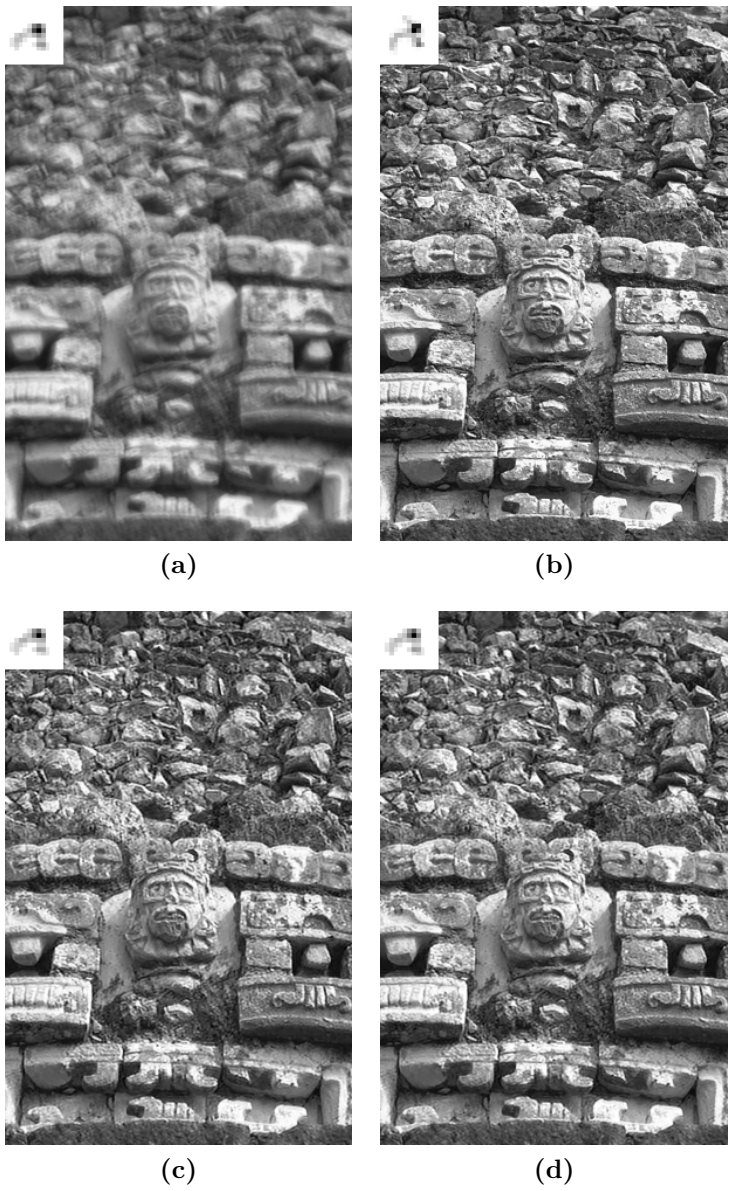


Figure 3.11: Blind deconvolution results for a third sample image of the BSD500 validation set. It was generated using the 5-th kernel of the dataset proposed by Levin et al. [28]. (a) blurry observation with initial blur kernel estimate of Xu and Jia [47] in the lower left corner. (b) ground truth sharp image together with true blur kernel. Deconvolution results of (c) IAM-BDC$_{20}^{3\times3}$ and (d) IAM-BDC$_{20}^{5\times5}$.

images depicted in Figure 3.11. We think that the resulting image of the IAM-BDC$_{20}^{5\times5}$ net matches the ground truth quite well, however in terms of *PSNR* this test sample yields the worst result, which is just above 21dB. This explains the large standard deviations in Tables 3.2 and 3.3 for the BSD500 dataset. The last example from the BSD500 test set

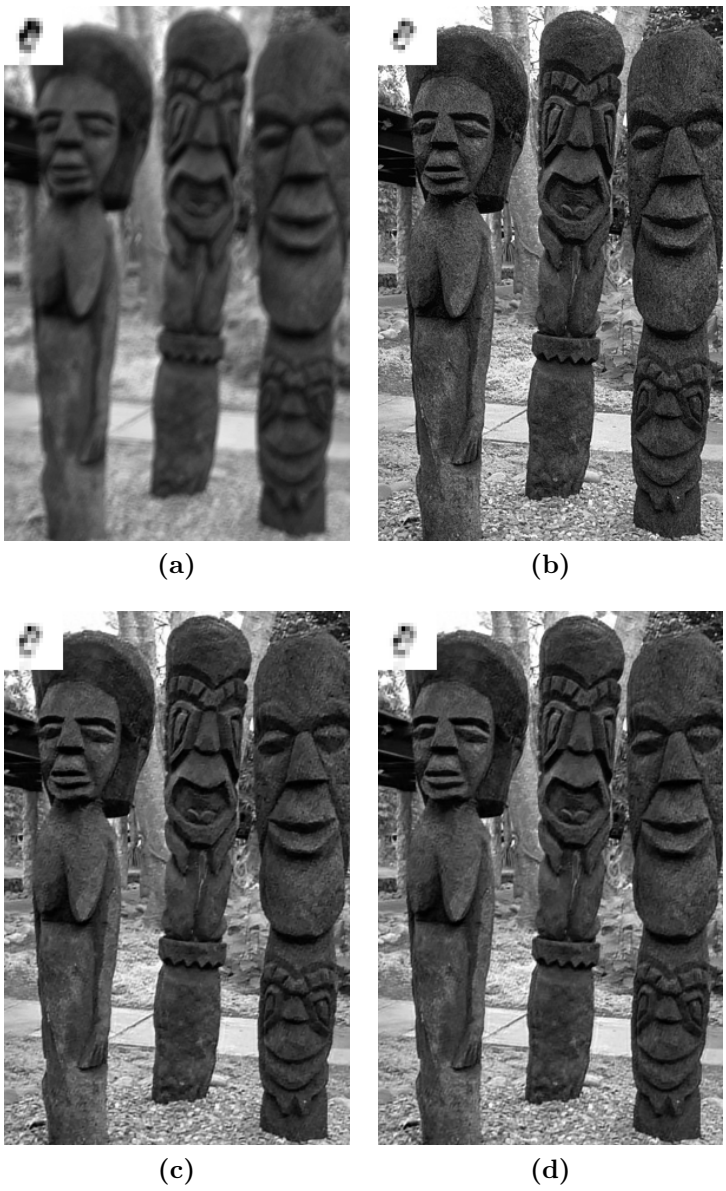


Figure 3.12: Blind deconvolution results for a fourth sample image of the BSD500 validation set. It was generated using the 3-rd kernel of the dataset proposed by Levin et al. [28]. (a) blurry observation with initial blur kernel estimate of Xu and Jia [47] in the lower left corner. (b) ground truth sharp image together with true blur kernel. Deconvolution results of (c) IAM-BDC$_{20}^{3\times 3}$ and (d) IAM-BDC$_{20}^{5\times 5}$.

is depicted in Figure 3.12. It has an average *PSNR* performance and some tiny ringing artifacts are present close to strong edges. However, the visual impression of the image and blur kernel estimate is quite astonishing.

In spite of the complex model derivation, its parameters can still be interpreted such

as those of the *IAM-DC* nets. Figure 3.13 illustrates the regularization parameters of the first, second and last stage of the IAM-BDC$_{20}^{5\times5}$ net. Similar to the Improved Iteratively Adapted Minimization for non-blind Deconvolution (I^{2}AM-DC) networks, the penalty functions ρ_{1i} of the first stage are dominated by heavy-tailed functions with little nibbles in the center. Additionally, the filters k_{1i} are almost the same as in the non-blind case, see Figure 2.15. However, the two penalty functions associated with step edges (first and last kernel-function pair in the first row) are differently shaped than in the non-blind case. They look like an inverse Mexican hat function. Consequently, this functions either favor smooth regions or large edges.

In the second stage the shape of the step edge related functions changes again. They are shaped like in the I^{2}AM-DC$_2^{5\times5}$ net and favor large step edges in order to suppress ringing artifacts. The remaining filter and function pairs are difficult to interpret due to their irregular filter shape but it seems as if they model the sparse gradient constraint of natural images. Hence, they help to suppress noise and artifacts.

The parameters of the final stage resemble those of the non-blind I^{2}AM-DC$_2^{5\times5}$ net. The first and last filter-function pair in the first row filter out small, smooth edges such as ringing artifacts. The second filter-function pair in the second row and the first in the third row are also related to edges. However, their filters are much steeper and their penalty function has minimal cost for medium sized edges. Hence, they suppress noise and enhance edges.

The filter $\bar{k}_{ti}$ of the image update data term and their corresponding penalty functions $\bar{\phi}_{ti}$ are depicted in Figure 3.14 for selected stages of the IAM-BDC$_{20}^{5\times5}$ net. In the first stages the shape of the penalty functions $\bar{\phi}_{ti}$ is a combination of the absolute, quadratic and heavy tailed function, like in the non-blind case. The absolute and tailed functions posses the most influence due to their scale. Additionally, the filters $\bar{k}_{ti}$ have shapes closer related to derivative filters. The influence of the quadratic penalty function grows with increasing depth. The intermediate images u_t and the blur kernel estimates a_t get closer to the true solution for increasing t. Thus, just the noise remains in the data term and it is known that the quadratic penalty function is optimal for denoising Gaussian noise. Consequently, the penalty functions preserve their quadratic shape.

The behavior of the kernel update data term filters $\tilde{k}_{ti}$ and their associated penalty functions $\tilde{\rho}_{ti}$ is totally different. It seems as if the initial discrete cosine transform filters and the quadratic penalty functions perform already quite well because they are almost not modified. All the functions in each stage preserve their initial quadratic shape, just their scale is slightly modified. Obviously, the parameters of the image update steps have a larger influence on the results. Therefore, the training algorithm concentrates on tuning those parameters. Nevertheless, the blur kernel refinement works quite well, as we have seen in the examples depicted in Figure 3.7 to 3.12.

The intermediate results of the first six stages of the IAM-BDC$_{20}^{5\times5}$ net are depicted in Figure 3.16. The sequence was generated by pushing the blurry observation shown in Figure 3.9a into the network. If we look at the output of the first stage, which is shown in

the upper left corner, we see that the blur kernel was slightly refined and just strong edges were enhanced. The second stage steadily continues the kernel refinement and enhances strong edges even more. Additionally, some more ringing artifacts are introduced. The next two stages keep on enhancing edges, while, the fifth stage filters out the majority of ringing artifacts. This effect originates from the block training approach. Note that the blur kernel estimate of the fifth stage is still rather coarse compared to the final estimate, which is depicted in Figure 3.9d. At the fifth stage the network has already found the sharp edges of the plain wings. The successive stages are used to further refine the blur kernel estimate and the deblurred image. Shan et al. [42] pointed out that the better the blur kernel estimates are, the fewer ringing artifacts appear due to kernel estimation errors. Consequently, the remaining ringing artifacts can be suppressed by improving the blur kernel estimate, which is done by the proposed network.

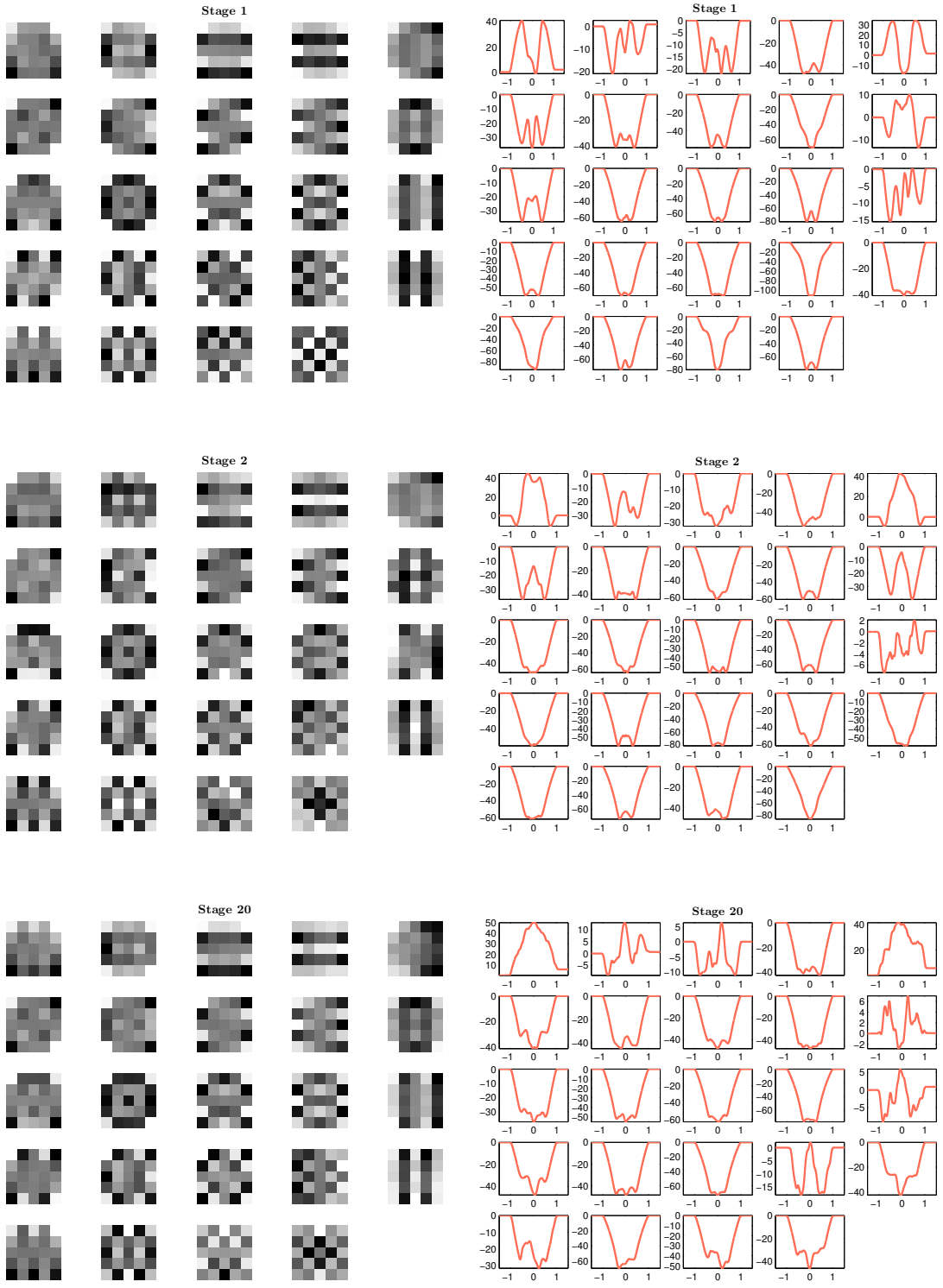


Figure 3.13: Filters k_{ti} and associated penalty functions ρ_{ti} of some stages of the IAM-BDC$_{20}^{5\times5}$ net. Top row illustrates the parameters of the first stage, in the central row are those depicted of the second stage, while, the last row shows the filters and penalty functions of the final stage.

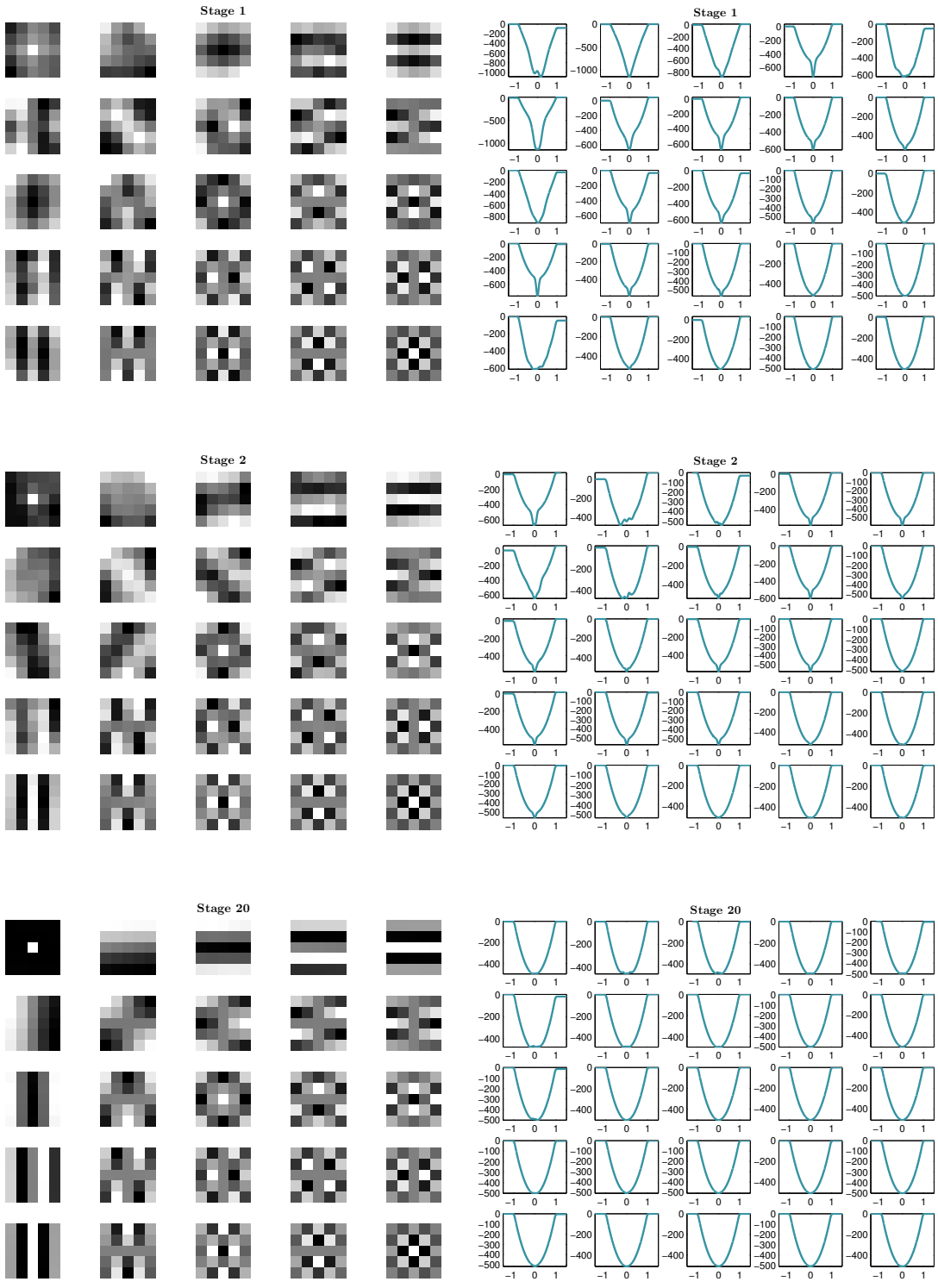


Figure 3.14: Filters $\bar{k}_{ti}$ and associated penalty functions $\bar{\rho}_{ti}$ of some stages of the IAM-BDC$_{20}^{5\times5}$ net. Top row illustrates the parameters of the first stage, in the central row are those depicted of the second stage, while, the last row shows the image update data term filters and penalty functions of the final stage.

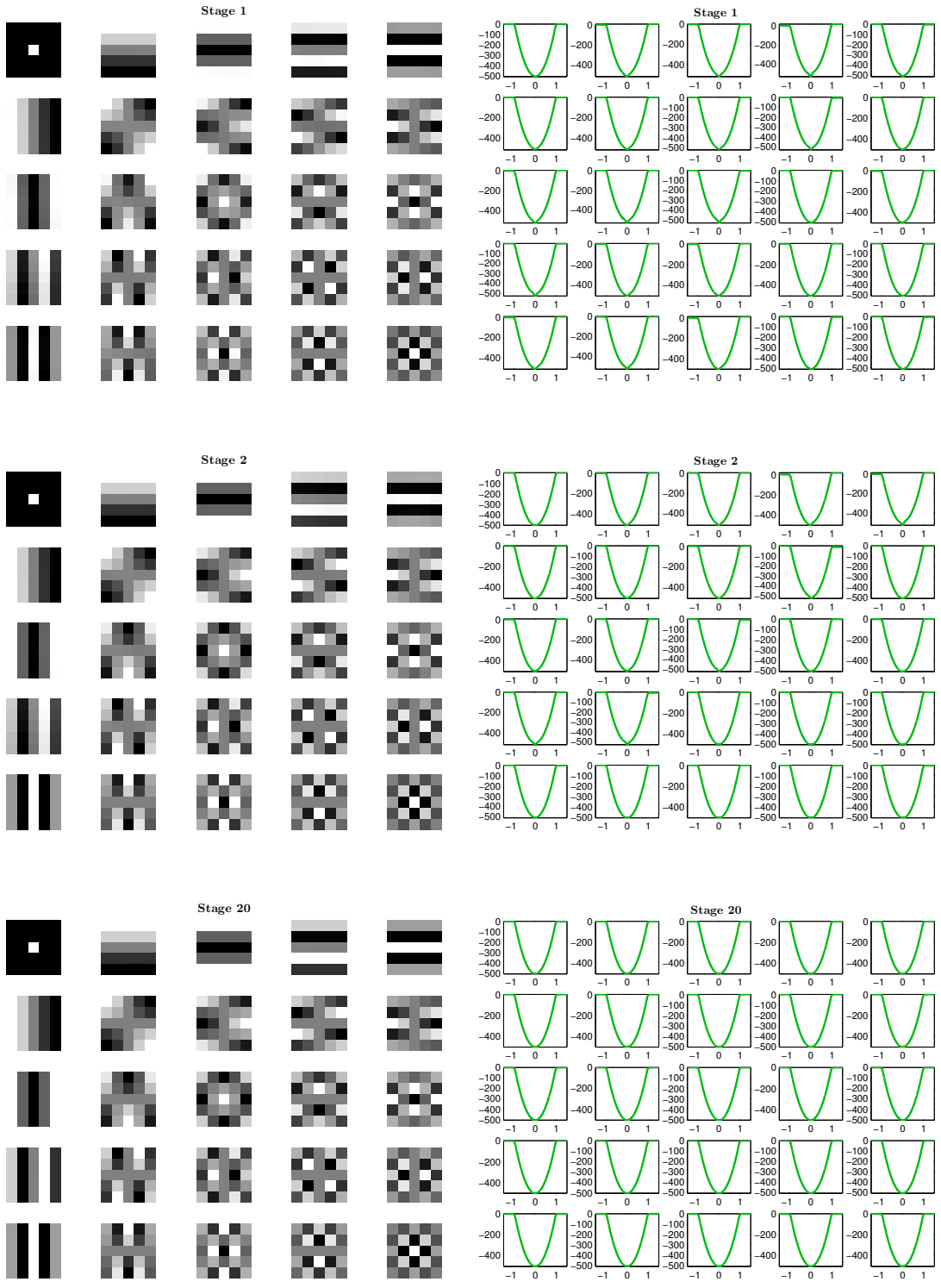


Figure 3.15: Filters $\tilde{k}_{ti}$ and associated penalty functions $\tilde{\rho}_{ti}$ of some stages of the IAM-BDC$_{20}^{5\times5}$ net. Top row illustrates the parameters of the first stage, in the central row are those depicted of the second stage, while, the last row shows the kernel update data term filters and penalty functions of the final stage.

Figure 3.16: Intermediate results of the first 6 stages of the IAM-BDC$_{20}^{5\times5}$ net for the image depicted in Figure 3.9. Top left image is the output of the first stage, to the left is the one of the second stage, while u_3 is below and so forth.

3.4 Conclusion

In this chapter we introduced a novel approach to blind image deconvolution. The idea originates from the denoising models of Chen et al. [13] and is inspired by the *TV* blind deconvolution of Perrone and Favaro [34]. While the approach of Chen and colleagues is based on reaction diffusion models, we motivate our algorithm by minimizing an energy functional. The proposed iteratively adapted energy minimization model is derived by applying the *PALM* algorithm of Bolte et al. [5] to minimize the according energy. We ended up with an efficient and fast method for blind image deconvolution. The introduced *IAM-BDC* networks are related to the *I²AM-DC* networks for non-blind image deconvolution, since both the regularization and data term parameters are trainable. The blind deconvolution networks can also be seen as an expansion to its non-blind pendant because we simply added a blur kernel update in each stage. The additional differentiable blur kernel refinement was tricky to derive, however the resulting blur kernel estimates after just a view updates demonstrate its power. Moreover, the reconstructed images appeal naturally and posses sharp edges.

In terms of the *PSNR* image performance measure, our approach is a bit below state of the art algorithms. For instance the algorithm of Fergus et al. [15] achieves an average *PSNR* value of 29.38dB on the dataset of Levin et al. [28]. The very recent and successful interleaved regression tree field cascade of Schelten et al. [39] obtained an average *PSNR* value of 31.50dB on this dataset. Hence, there is still some space for improvement. Nevertheless, the proposed IAM-BDC$_{20}^{5\times5}$ net delivers good results despite inaccurate initial blur kernel estimates, as we have seen in the examples above.

The proposed iteratively adapted minimization networks are closely related to energy minimization just like their non-blind pendants. Therefore, each parameter of the network can be interpreted in an easily understandable way, which helps to understand suitable blind deconvolution methods. For instance we demonstrated how the regularization changes from stage to stage at the end of the previous section. The first stages try to filter out noise and small textures of the blurry observation and start sharpening strong edges. The successive stages keep on enhancing edges and refine the blur kernel. The deeper network stages remove artifacts such as ringing around strong edges, see Figure 3.16.

To sum up, the proposed *IAM-BDC* networks are suitable for efficient blind image deconvolution. They generate remarkable results even for inaccurate initial blur kernel estimates. Furthermore, we have to note that the experiments summarized here are just a small subset of the already performed ones. However, there are still a lot of interesting questions open: Is this approach also able to robustly estimate initial kernels, since it refines the estimates of Xu and Jia [47] quite well? Does the method also work if the influence functions and kernels of both data terms are kept constant?

4 Conclusion and outlook

4.1 Conclusion

In this work we first profoundly analyzed different reasons for blur in images and outlined the problems which make the image deblurring task so difficult. We reformulated the uniform blurring process as a mathematical convolution operation and derived a variational formulation of the deconvolution problems. We extended it by introducing the regularization structure of Chen et al. [13] and derived an iterative energy minimization approach by a simple steepest gradient descent algorithm. To speed up computation, we followed the idea of Chen and colleagues [13] to fixed the amount of iterations and allowed the regularization and step size parameters of each stage to be arbitrary. We ended up with the Iteratively Adapted Minimization for non-blind Deconvolution (IAM-DC) network-like structure, which can be trained in a similar fashion as neural networks. To account for the large depth of the proposed networks, we applied a block training strategy, consisting of a greedy pre-training and a joint training of all stage parameters within a block of stages. Basically any objective function and training algorithm can be used. We decided to apply the well known L-BFGS algorithm [29] for training and optimize the pixel-wise squared error between an estimated image and the ground truth. The evaluation of various *IAM-DC* nets showed that more stages and larger regularization filters improved the Peak Signal Noise Ratio (PSNR). The IAM-DC$_{10}^{3\times3}$ net, which is the simplest, has already average *PSNR* scores which are almost 1dB better than those of the frequently used TV-based non-blind image deconvolution.

We further improved the results by expanding the data term. Instead of a simple ℓ_2-norm of the image formation process, we applied different filtered versions and allowed the penalty function to change arbitrarily, just like the regularization in the *IAM-DC* nets. This Improved Iteratively Adapted Minimization for non-blind Deconvolution (I^{2}AM-DC) networks are able to recover fine details better and suppress ringing artifacts at strong edges even more. The resulting data term penalty functions showed that initially an ℓ_1-norm of the filtered image formation process is suitable. However, at deeper network stages

the ℓ_2-norm is used because it is optimal to suppress Gaussian noise. This deep analysis of the network parameters is possible due to the close relation to energy minimization of the proposed methods. Hence, every single piece of the network such as the image regularization or the data term filters can be easily interpreted, which is a fundamental advantage compared to conventional neural networks. As a result, the trained networks help to deepen the knowledge about natural image regularization.

The relation to energy minimization is maintained in the Iteratively Adapted Minimization for blind Deconvolution (IAM-BDC) networks because they also originate from an energy minimization problem. The derivation of a differentiable iterative algorithm to optimize the variational formulation of the blind image deconvolution problem is quite challenging. We applied the Proximal Alternating Linearized Minimization (PALM) algorithm of Bolte et al. [5] to infer two alternating gradient descent steps. The first step updates the unknown image and the second refines the blur kernel estimate based on the current image. Moreover, we presented a differentiable projection onto the unit simplex, which is required for updating the blur kernel. When the same ideas as in the non-blind case are applied to the iterative update rules, we end up with the *IAM-BDC* networks. Its structure is different from those of Chen et al. [13] and the *IAM-DC* nets due to the alternating optimization. Nevertheless, this network can be trained like the non-blind *I²AM-DC* networks. The training algorithm has to take the alternating minimization into account, though. We evaluated those trained networks for various depth and different kernel sizes. Like in the non-blind case, deeper networks and larger kernels enhance the quality of the results, in terms of noise and ringing artifacts. However, the improvement is not as high as for the *I²AM-DC* networks. Nevertheless, the networks produce naturally appealing images with sharp edges. The *PSNR* values for the dataset of Levin and colleagues [28] decrease slightly for very deep networks because the outputs posses sharper edges than the corresponding ground truth. Furthermore, the proposed network is capable of refining the sometimes inaccurate initial blur kernel estimates of Xu and Jia [47] very well. All in all, we introduced a novel approach for non-blind and blind image deconvolution which generates remarkable results and can be easily analyzed due to its close relation to energy minimization.

4.2 Future work

We have already mentioned that the experiments performed in this work are just an outline and still many interesting questions are unanswered. Moreover, we demonstrated that the proposed networks are efficient to evaluate. However, we just use a pure MATLAB implementation so far. Since the networks consist of simple mathematical operations, a GPU port should decrease the runtime tremendously, see Chen et al. [13].

Additionally, a cascade of *IAM-DC* networks and iteratively adapted minimization nets for blur kernel estimation can be set up and trained. Also other objective training functions have to be investigated. For instance, instead of the *PSNR* we could maximize the

Structural Similarity Index (SSIM). To get a deeper insight into the deconvolution problem, we may constrain the network parameters to change linearly throughout the stages by setting up a suitable training objective. In addition, other training algorithms such as the stochastic conjugate gradient method could be incorporated to speed up training. The high flexibility of our approach leaves room for creativity and further improvement.

A

List of Acronyms

CRFs	Conditional Random Fields
DFT	Discrete Fourier Transform
DoF	Depth of Field
EM	Expectation Maximization
EPLL	Expected Patch Log Likelihood
FoE	Field of Experts
GMM	Gaussian Mixture Models
I^2AM-DC	Improved Iteratively Adapted Minimization for non-blind Deconvolution
IAM-BDC	Iteratively Adapted Minimization for blind Deconvolution
IAM-DC	Iteratively Adapted Minimization for non-blind Deconvolution
iid	independent and identically distributed
MAP	Maximum A Posteriori
MRF	Markov Random field
MRI	Magnetic Resonance Imaging
NN	Neural Network
PALM	Proximal Alternating Linearized Minimization
PDF	Probability Density Function
PSF	Point Spread Function
PSNR	Peak Signal Noise Ratio
RTFs	Regression Tree Fields
SotA	State-of-the-Art
TV	Total Variation

B Convolution in matrix vector notation

B.1 Formulation as matrix vector product

In the discrete setting we define the valid convolution of two 2D signals as

$$f(i,j) = (u * a)(i,j) = \sum_{h=0}^{H-1} \sum_{w=0}^{W-1} u(i+h, j+w) a(H-h, W-w), \tag{B.1}$$

for $i = 1 \ldots M-H+1,\ j = 1 \ldots N-W+1$ whereby the kernel a is of size $H \times W$ and the image u of $M \times N$ and all dimensions are odd. This formulation is a bit unconventional, however, it exactly matches the behavior of the MATLAB command `f=conv2(u,a,'valid')`, which is used throughout our implementations.

Equation (B.1) can be reformulated as

$$f = Au = Ua \Leftrightarrow f = u * a, \tag{B.2}$$

where A is a $(M-H+1)(N-W+1) \times MN$ matrix, $u \in \mathbb{R}^{MN}$, $a \in \mathbb{R}^{HW}$ and the matrix U is of size $(M-H+1)(N-W+1) \times HW$. The vector u holds the image pixels in lexicographical ordering and a the kernel elements respectively. This ordering scheme is used to benefit from the powerful matrix calculus. In order to express the convolution as matrix vector product, we need to setup the matrices correctly. The matrix A is built up of row vectors a_n^T such that

$$f(i,j) = (u * a)(i,j) = \sum_{l=1}^{MN} a_n(l) u(l)$$

is fulfilled for $n = j(M-H+1)+i$. Therefore, the vector a_n holds zeros for each element of u being not under the blur kernel mask if image pixel n is processed and the associated blur kernel values else. Consequently, A is a Toeplitz matrix which holds in each row either zeros or the kernel coefficients. The matrix U can be constructed in the same way.

It is built up of row vectors u_n and each of those must satisfy

$$f(i,j) = (u * a)(i,j) = \sum_{l=1}^{HW} u_n(l)a(l),$$

where $n = j(M - H + 1) + i$. Thus, each of those row vectors holds the image pixels that are under the blur kernel mask when the n-th output pixel is computed. So, the columns of the matrix U are basically shifted versions of the vector u. As a result, all the formulations in Equation (B.2) are equivalent. The advantage of the reformulation using matrices is the applicability of the matrix calculus along with its derivative rules.

B.2 Gradients and their relation to convolution operations

The basic formulation of the image deconvolution problem is given by

$$E(u,a) = \frac{1}{2} ||u * a - f||_2^2 .$$

This model tries to find an image u and a blur kernel a such that the convolution yields the observation f. An equivalent formulation using the matrix vector notation of the convolution operations is

$$E(u,a) = \frac{1}{2} ||Au - f||_2^2 = \frac{1}{2} ||Ua - f||_2^2 .$$

Based on this formulation the gradients with respect to u and a can be easily expressed as

$$\begin{aligned} \nabla_u E &= A^\mathsf{T}(Au - f) \\ \nabla_a E &= U^\mathsf{T}(Ua - f). \end{aligned}$$

Although this formulation is very helpful for deriving the gradients, in practice the direct convolution operation, see Equation (B.1), is much faster as its matrix-vector pendant. So, we need to express the operations which correspond to A^T and U^T multiplied with a column vector from the right. Let us denote this vector as $\bar{f} \in \mathbb{R}^{(M-H+1)(N-W+1)}$, which is basically a vector representation of the convolution $u * a$. If A^T is multiplied with this vector $\bar{f}$, one gets

$$A^\mathsf{T}\bar{f} = \bar{u},$$

where $\bar{u} \in \mathbb{R}^{MN}$. Due to the insertion of zeros in the construction of A, this operation can be interpreted as a full convolution with zero padding as boundary handling. Moreover, the transposition implies a 180° rotation of the blur kernel a. Hence, this operation can be computed by the MATLAB command `conv2(`$\bar{f}$`,rot90(a,2),'full')`.

In contrast, if U^T is multiplied with the vector $\bar{f}$, we get

$$U^\mathsf{T}\bar{f} = \bar{a},$$

where the vector $\bar{a} \in \mathbb{R}^{HW}$. Also this operation can be expressed as a convolution with a rotated image u. In MATLAB it can be computed by `conv2(rot90(u,2),`$\bar{f}$`,'valid')`. Consequently, the time consuming matrix representation of the convolution can be avoided by using the above commands instead.

Bibliography

[1] Barber, D. (2012). *Bayesian Reasoning and Machine Learning*. Cambridge University Press. (page 13)

[2] Beck, A. and Teboulle, M. (2003). Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175. (page 72, 75)

[3] Ben-Tal, A. and Nemirovski, A. (2005). Non-euclidean restricted memory level method for large-scale convex optimization. *Mathematical Programming*, 102(3):407–456. (page 72)

[4] Bertero, M. and Boccacci, P. (1998). *Introduction to Inverse Problems in Imaging*. CRC Press. (page 6)

[5] Bolte, J., Sabach, S., and Teboulle, M. (2014). Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494. (page 68, 69, 103, 106)

[6] Bredies, K., Kunisch, K., and Pock, T. (2010). Total Generalized Variation. (page 19)

[7] Bregman, L. (1967). The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. (page 72)

[8] Chambolle, A. and Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145. (page 18, 20, 35)

[9] Charbonnier, P., Blanc-Féraud, L., Aubert, G., and Barlaud, M. (1997). Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on Image Processing*, 6(2):298–311. (page 22)

[10] Chaudhuri, S., Velmurugan, R., and Rameshan, R. M. (2014). *Blind Image Deconvolution - Methods and Convergence*. Springer. (page 9, 60)

[11] Chen, J., Dong, W., Feng, H., Xu, Z., and Li, Q. (2013). High quality non-blind image deconvolution using the Fields of Experts prior. *Optik*, 124(18):3601–3606. (page 22)

[12] Chen, Y. (2015). *Learning fast and effective image restoration models*. PhD thesis, Institute for Computer Graphics and Vision, Graz University of Technology, Graz, Austria. (page 24, 29, 30)

[13] Chen, Y., Yu, W., and Pock, T. (2015). On learning optimized reaction diffusion processes for effective image restoration. In *IEEE Conference on Computer Vision and Pattern Recognition*. (page 1, 22, 24, 26, 27, 54, 68, 103, 105, 106)

[14] Dong, W., Feng, H., Xu, Z., and Li, Q. (2012). Blind image deconvolution using the Fields of Experts prior. *Optics Communications*, 285(24):5051–5061. (page 27)

[15] Fergus, R., Singh, B., Hertzmann, A., Roweis, S. T., and Freeman, W. T. (2006). Removing camera shake from a single photograph. (page 61, 103)

[16] Geman, D. and Reynolds, G. (1992). Constrained restoration and the recovery of discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(3):367–383. (page 21)

[17] Geman, D. and Yang, C. (1995). Nonlinear image recovery with half-quadratic regularization. *IEEE Transactions on Image Processing*, 4(7):932–946. (page 21)

[18] Geman, S. and Geman, D. (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6). (page 21)

[19] Jancsary, J., Nowozin, S., and Rother, C. (2012a). Loss-Specific Training of Non-Parametric Image Restoration Models: A New State of the Art. In *12th European Conference on Computer Vision*, volume 7578 of *Lecture Notes in Computer Science*, pages 112–125. Springer Berlin Heidelberg. (page 66)

[20] Jancsary, J., Nowozin, S., Sharp, T., and Rother, C. (2012b). Regression Tree Fields An efficient, non-parametric approach to image labeling problems. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2376–2383. (page 66)

[21] Joshi, N., Szeliski, R., and Kriegman, D. J. (2008). PSF estimation using sharp edge prediction. In *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. (page 61)

[22] Krishnan, D. and Fergus, R. (2009). Fast Image Deconvolution using Hyper-Laplacian Priors. In Bengio, Y., Schuurmans, D., Lafferty, J. D., Williams, C. K. I., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22*, pages 1033–1041. Curran Associates, Inc. (page 21, 61)

[23] Krishnan, D., Tay, T., and Fergus, R. (2011). Blind deconvolution using a normalized sparsity measure. *CVPR 2011*, pages 233–240. (page 15)

[24] Lai, W.-S., Ding, J.-J., Lin, Y.-Y., and Chuang, Y.-Y. (2015). Blur Kernel Estimation Using Normalized Color-line Priors. In *Proceedings of IEEE Conferene on Computer Vision and Pattern Recognition (CVPR 2015)*, page to appear. (page 15, 61, 63, 64, 79)

[25] Levin, A., Fergus, R., Durand, F., and Freeman, W. T. (2007). Image and Depth from a Conventional Camera with a Coded Aperture. *ACM Trans. Graph.*, 26(3). (page 20, 43, 58)

[26] Levin, A., Weiss, Y., Durand, F., and Freeman, W. T. (2009). Understanding and evaluating blind deconvolution algorithms. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*, pages 1964–1971. (page 60)

[27] Levin, A., Weiss, Y., Durand, F., and Freeman, W. T. (2011a). Efficient marginal likelihood optimization in blind deconvolution. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2657–2664. (page 61, 66)

[28] Levin, A., Weiss, Y., Durand, F., and Freeman, W. T. (2011b). Understanding blind deconvolution algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2354–2367. (page 23, 24, 35, 36, 46, 60, 61, 90, 91, 92, 93, 94, 95, 96, 103, 106)

[29] Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528. (page 81, 105)

[30] Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, 2. (page 34, 35, 46, 57, 58, 89, 90)

[31] Nesterov, Y. (2004). *Introductory lectures on convex optimization: A basic course*, volume 0. (page 75)

[32] Osher, S. and Rudin, L. I. (1990). Feature-Oriented Image Enhancement Using Shock Filters. (page 62)

[33] Perrone, D., Diethelm, R., and Favaro, P. (2015). Blind Deconvolution via Lower-Bounded Logarithmic Image Priors. In *International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*. (page 63)

[34] Perrone, D. and Favaro, P. (2014). Total Variation Blind Deconvolution: The Devil is in the Details. In *EEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 23, 63, 65, 77, 103)

[35] Roggemann, M., Welsh, B., and Hunt, B. (1996). *Imaging Through Turbulence*. CRC Press. (page 4)

[36] Roth, S. and Black, M. (2009). Fields of Experts. *International Journal of Computer Vision*, 82(2):205–229. (page 22)

[37] Rudin, L. and Osher, S. (1994). Total variation based image restoration with free local constraints. *Proceedings of 1st International Conference on Image Processing*, 1. (page 17)

[38] Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. (page 17)

[39] Schelten, K., Nowozin, S., Jancsary, J., Rother, C., and Roth, S. (2015). Interleaved Regression Tree Field Cascades for Blind Image Deconvolution. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, pages 494–501. (page 27, 34, 57, 66, 67, 79, 89, 103)

[40] Schmidt, U., Rother, C., Nowozin, S., Jancsary, J., and Roth, S. (2013). Discriminative Non-blind Deblurring. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 604–611. (page)

[41] Schuler, C. J., Hirsch, M., Harmeling, S., and Schölkopf, B. (2014). Learning to Deblur. *CoRR*, abs/1406.7. (page 27, 67, 79)

[42] Shan, Q., Jia, J., and Agarwala, A. (2008). High-quality motion deblurring from a single image. (page 43, 63, 64, 79, 98)

[43] Sun, L., Cho, S., Wang, J., and Hays, J. (2013). Edge-based blur kernel estimation using patch priors. In *2013 IEEE International Conference on Computational Photography, ICCP 2013*. (page 61, 63)

[44] Teboulle, M. (1992). Entropic Proximal Mappings with Applications to Nonlinear Programming. (page 72, 73)

[45] Tikhonov, A. N. and Arsenin, V. I. A. (1977). *Solutions of ill-posed problems*. Scripta series in mathematics. Winston. (page 16)

[46] Wang, R. and Tao, D. (2014). Recent Progress in Image Deblurring. *Computing Research Repository*, abs/1409.6. (page 60)

[47] Xu, L. and Jia, J. (2010). Two-phase kernel estimation for robust motion deblurring. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6311 LNCS, pages 157–170. (page 15, 16, 18, 61, 62, 79, 89, 91, 92, 93, 94, 95, 96, 103, 106)

[48] Zhu, X. and Milanfar, P. (2013). Removing Atmospheric Turbulence via Space-Invariant Deconvolution. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):157–170. (page 4)

[49] Zoran, D. and Weiss, Y. (2011). From learning models of natural image patches to whole image restoration. In *2011 International Conference on Computer Vision*, pages 479–486. Ieee. (page 23)