Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science
Mathematical Image Analysis Group

Bachelor's Thesis

Depth from Blur

Combining Image Deblurring and Depth Estimation

submitted by
Steffen Lösch

submitted
on July 24, 2009

Supervisor
Joachim Weickert

Advisor
Martin Welk

Reviewers
Joachim Weickert
Martin Welk

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement in Lieu of an Oath

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit verffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken,..... .....

(Datum/Date) (Unterschrift/Signature)

Abstract

We propose a novel method for combined deblurring and depth estimation in a variational framework. Our goal is to retrieve a deblurred image and the corresponding depth map by only using information from a single blurred image. In many practical applications, for example for defocus or camera motion, the shape of the blurring remains constant, but only its spatial extent is varying. A new blurring model based on this assumption, in which the depth is included explicitly, is introduced. Depth is related to the spatial extent of the blur by assuming that the stronger an object is blurred, the closer it is to the camera. We also present a reduction of the model to one-dimensional blurring, as it is present for example in motion blur. The deblurred image and the depth map is then estimated variationally by means of gradient descent, where a novel partial differential equation occurs. Our method is evaluated experimentally, and the results confirm that the combination of deblurring and depth estimation from just one image is possible.

Acknowledgements

First of all, I would like to thank my family, Charlotte and my friends for their enduring support in every kind of way. Without their help I would never have come so far.

I also want to thank Joachim Weickert for providing me with the interesting topic of this thesis and suggesting me several other topics. He was always very helpful and accessible whenever I had a request. I am also very grateful for the descent mathematical education I received from him.

Not to forget is the advisor of my bachelor's thesis, Martin Welk. He has been a constant support in the whole working process for this thesis and he was very patient in answering all my questions. He always gave me the little nudge I needed to keep on going. Many important ideas laid out in this thesis were motivated by him. I thank him very much.

My thanks are also dedicated to all the other members of the Mathematical Image Analysis Group, in particular Markus Mainberger. They have always been helpful and friendly and created a great working atmosphere in the chair.

I would like to thank Gert Smolka for being my tutor from the Honours Program for Bachelor Students. I really appreciated the term meetings I had with him and he has been a great help in many different aspects of my studies.

Last but not least, I want to thank the German National Academic Foundation (Studienstiftung des deutschen Volkes) for funding my studies and providing me with the opportunity to meet many new people and to make new experiences.

Contents

List of Figures

1 Introduction

> The camera has an interest in turning history into spectacle, but none in reversing the process. At best, the picture leaves a vague blur in the observer's mind; strong enough to send him into battle perhaps, but not to have him understand why he is going.
>
> *Denis Donoghue*

Image deblurring and depth estimation are both tasks with a long tradition in image processing. In deblurring, as the name already indicates, we want to recover a sharp image back from an image, that is deviated under some blur. Depth estimation is about retrieving 3-dimensional information out of one or several 2-dimensional images.

Everyone who has already taken images with a small hand-held camera knows what image blur is. It appears very frequently in practice that images are blurred. The reasons range from the shaking hands with a small camera over ordinary defocus to images taken from a moving vehicle or even atmospheric perturbations when a telescope is used. Obviously it is very desirable for a photographer to have a deblurring algorithm to get nice and sharp images out of blurry ones. Unfortunately deblurring is a very hard problem, and it appears in many different variants. The blur may be the same for the whole image or it may vary over space. We may know how the blur was caused or not. We may know the shape of the blur or not or we may know it only approximately. Additionally, even the simplest deblurring task, where the blur is everywhere the same and we know it exactly, is an ill-posed problem.

But anyway there is still a great demand in having a good deblurring algorithm. This fact has attracted many researches and a lot of papers have been written about deblurring. They differ in what assumptions they make, what variant of the problem they actually want to solve, how they cope with the ill-posedness and what modelling techniques they use. Some of these approaches are mentioned in the next section. No method works well in all cases and so all the different deblurring techniques have their right to exist.

Depth estimation is also a very important task. We live in a 3-dimensional environment, but are only able to take 2-dimensional images (at least with a standard camera). Therefore we want to bring these two worlds together. If the camera position is fixed, then we should not expect an algorithm to reconstruct the complete 3-D space, because we cannot look behind objects. In this setting a depth map for an taken image is the best what we can hope for.

Many different information cues have been investigated to retrieve 3-D information from 2-D images. These cues include stereo images, shading, motion, focus, texture and some more. Also the depth reconstruction from blur has been already investigated, including defocus and motion blur. Basically any way to retrieve depth is useful because often just a single cue is available. One could also construct specific devices that fulfil one's

assumptions exactly. Such devices can be used for example in medical imaging or robotics, where it is very important to know the real 3-D world instead of a simple 2-D projection.

Deblurring and depth estimation are indeed highly related, if the right setting is chosen. Imagine for example a car driving by a house. If one takes a picture out of a moving car, it will be blurred, because the car is driving. But a tree in front of the house will be stronger blurred than the house, because it is closer to the camera, and a aircraft in the sky will be probably not blurred at all. In this case, the assumption that the closer an object to the camera is, the more it is blurred, is true. But this assumption can also simply be violated. If a child is running next to the tree, it will be blurrier than the tree, even if it is at the same depth.

The assumption that closer objects are more blurred is nevertheless true in many practical situations, as for example in the described car scene. If this is fulfilled, then depth and blur are basically in a one-to-one relation: if we know the depth then we can estimate the blurriness of an object, and if we know its blur then we can reconstruct its depth. Our goal is to use this observation to design an algorithm that combines deblurring and depth estimation in a variational framework.

1.1 Related Work

As already stated before, deblurring is a very research-intensive topic and there are still improvements and novel methods.

The most classical methods are linear approaches, that are based on the Fourier transform. They are only applicable to spatially invariant blur, because they use the convolution theorem, which says that the convolution of two signals turns out to be a multiplication in Fourier space. The most prominent example of this filter class is the Wiener filter [25]. Another often used approach is the Richardson-Lucy algorithm, that was introduced in [12] and [14]. It is based on Bayes' theorem and performs deblurring in an iterative way. The Wiener filter as well as the Richardson-Lucy algorithm assume that the blur is known exactly.

There are also methods that try to estimate spatially invariant blur and the unblurred image from a single image. Fergus et al. [8] model their method using probability distributions and by means of Bayes' theorem. They include prior knowledge in assuming that the deblurred image follows natural image statistics. These statistics are also modelled by Shan et al. [17], but they also introduce some other interesting concepts that makes their algorithm quite successful. Even though they motivate their model in a Bayesian way, the resulting energy is very similar to the ones in pure variational approaches.

These variational approaches include the methods from You and Kaveh (see [26] and [27]), that focus on the regularisation effect of the smoothness terms. Images are also deblurred in a spatially variant and blind way, where the ill-posedness is reduced by parametrising the blur. The most influential methods for this thesis are the works by Welk et al. (see [22], [23] and [24]). They perform spatially invariant and variant deblurring under known blur. The use of robust data terms for deblurring is justified by them and they propose the stepwise reduction of the smoothness weights during image evolution. Boundaries are treated in a detailed way and [22] derives stability conditions for the gradient descent. Bar et al. introduced the notion of robust data terms in [2]. Money

and Kang [13] try to improve the deblurring result by using a shock filtered version of the blurred image as initialisation.

We now come to approaches that include some notion of depth into deblurring. In Jin et al. [9] a variational energy is not based on a 2-dimensional model as in the methods before, but on a 3-dimensional one. They already try to estimate a depth map and an unblurred image, but they are using multiple defocussed images. Multiple images are also used by Favaro et al. in [4], [5] and [7], where they assume that the blur for defocus has Gaussian shape, which is a quite strong assumption. Therefore they are able to use the relation between Gaussian convolution and diffusion to use this in their variational energy to recover some 3-D shape from blurred images. The focus of Lin and Chang is on geometric optics, in [11] they show in a detailed way why the relation between depth and blur is reasonable for defocus and motion blur. Their depth estimation method only estimates one depth value for the whole image by trying to detect the blurriness of edges, it is therefore spatially invariant. The method can only handle quite restricted test patterns with strong edges. Bae and Durand [1] also try to detect the blurriness of edges to get some blur measure for the image, but they do it in a spatially variant way, such that they get a "blurriness map". They actually do not use this for deblurring, but for defocus magnification to increase the quality of a photograph. The blurriness of edges is detected by a variant of the edge detector from Elder and Zucker [3], that is based on Gaussian derivatives. Levin [10] does blind spatially variant deblurring of motion blurred images with a hands-on approach. She compares the image statistics in the direction of the blur to the statistics in the direction rectangular to it, which is not blurred. Doing this she gets a measure how well a certain blur fits to the image and she uses this measure in a segmentation algorithm to get regions with different blur extents.

Machine learning approaches have been used to perform depth estimation, as well. Favaro and Soatto [6] follow a quite discrete idea, where they interpret the blurring process as matrix multiplication and use singular value decomposition to detect the blurriness. Saxena et al. [15] perform classical supervised learning, by constructing a reasonable feature vector and using a Markov Random Field. They do not even assume that their images are blurred, because they just train their algorithm with ground truth images taken with a 3-D scanner and hence try to reconstruct depth with a single unblurred image. The machine learning approaches have in common that they need a ground truth set of images to train their method.

That the relation between blur and depth exists is one more time justified by Schechner and Kiryati [16]. They show that depth recovery from stereo images is actually very similar to depth recovery from blurred images.

1.2 Overview

The thesis was started with the current Chapter 1, where we motivated our problem and introduced already existing approaches to solve it or some variants of it. Chapter 2 gives the most important mathematical background, that is needed to understand this thesis. Our mathematical language, in terms of notation, is laid out therein as well. In Chapter 3 blur and the deblurring task are studied in more detail and important observations for our approach are gained. Additionally the standard blurring model is introduced. In contrast

to that, our new model is set up in Chapter 4, and we explain the relation to depth there. Chapter 5 gives the corresponding variational energy functional and derives the gradient descent equations for the unblurred image and the depth map. The optimisation part is completed by Chapter 6, where we discretise the gradient descents for the reduced model, that is designed specifically for motion blur in horizontal direction. In Chapter 7 the implementation of the discretised gradient descents is discussed and results of our algorithm are shown. This thesis ends with Chapter 8 where we look back to what we have done, interpret the results and give a perspective of future work. Appendix A shows some proofs that are needed in the other chapters.

2 Mathematical Background and Notation

Without understanding the mathematical background and notation given in this chapter, it will not be possible to follow the rest of this thesis. Knowledge of the contents of this chapter is assumed in the other chapters and the notation will be used without any further comments. Therefore the basis of the thesis is formed here.

2.1 Images and Functions

In visual computing the ordinary way to work with images is to consider them as functions. For example a greyscale image, that ranges horizontally from 1 to 800, vertically from 1 to 600 and contains grey values between 0 and 255 can be modelled as a discrete function f:

$$f : \{1, 2, .., 800\} \times \{1, 2, .., 600\} \to \{0, 1, .., 255\}$$

This interpretation allows us to work with images in a mathematical framework. However, in the physical, real world, images are not discrete. When one looks around, one can always come closer and closer to an object and the image of the object will become more and more detailed. There is no discrete grid in real life. So it makes sense to reason with continuous functions, that are defined in the whole space, and interpret them as continuous images without boundaries:

$$f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$

By this, we can model visual computing algorithms in continuous space. Because these functions still correspond to observable images, we can assume that they are "reasonable". This means that they have nice properties like

- smoothness (e.g. $f \in C^\infty$)
- boundedness (e.g. $f \in L^\infty$)

So we are able to derive and integrate such functions and use other continuous calculus. But in the end we always have to transform a continuous framework back to the discrete setting, because digital images are discrete.

2.2 Notation

This section contains conventions about notation, that will be used throughout the whole thesis. For better readability, we will stick to mathematical standards as far as possible.

2.2.1 Vectors

Images discussed in this paper will be 2-dimensional. Hence the arguments of most functions will be 2-dimensional vectors as well. For shorter notation, vectors are denoted by **bold** variables:

$$\mathbf{x} := \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

2.2.2 Functions

We will mainly consider two types of functions: discrete and continuous ones. For better discriminability between these two types, we will denote them in a different way, by writing their arguments in different ways.

For discrete functions, the arguments are given as subscripts. So the value of a discrete function f at the point $(x_1, x_2)^\top$ is denoted by f_{x_1,x_2}, whereas $f(\mathbf{x}) = f\left((x_1, x_2)^\top\right)$ is the value of a continuous function at the point $(x_1, x_2)^\top$.

Please note that the horizontal direction will be often called "x-direction" and the vertical direction will be called "y-direction", although we will often just denote the arguments of these directions by x_1 and x_2.

2.2.3 Operators

Let us assume that f is a 2-dimensional function, $f : \mathbb{R}^2 \to \mathbb{R}$, and g is a 1-dimensional function, $g : \mathbb{R} \to \mathbb{R}$. We define the following differential operators:

Derivative:	$g'(x) := \partial_x g(x) := \dfrac{d}{dx} g(x)$
Partial derivative by x_1:	$\partial_{x_1} f(\mathbf{x}) := \dfrac{\partial}{\partial x_1} f(\mathbf{x})$
Gradient:	$\nabla f(\mathbf{x}) := \begin{pmatrix} \frac{\partial}{\partial x_1} f(\mathbf{x}) \\ \frac{\partial}{\partial x_2} f(\mathbf{x}) \end{pmatrix}$
Divergence:	$\mathrm{div} f(\mathbf{x}) := \dfrac{\partial}{\partial x_1} f(\mathbf{x}) + \dfrac{\partial}{\partial x_2} f(\mathbf{x})$

2.3 Calculus of Variations

The calculus of variations is about finding the minimiser of a real-valued functional, e.g. a mapping from a function $u(x)$ into the real numbers $\mathbb{R}$. This section and its notation is based on [21].

2.3.1 One-dimensional Calculus

The theorem of Euler-Lagrange says, that if a functional is given in the following form

$$E(u) := \int_a^b F(x, u, u')\ dx \tag{2.1}$$

then a minimising function u necessarily fulfils the Euler-Lagrange equation

$$\frac{\partial}{\partial u}F - \frac{d}{dx}\left(\frac{\partial}{\partial u'}F\right) = 0$$

and the so-called natural boundary conditions:

$$\frac{\partial}{\partial u'}F = 0 \text{ , for } x = a \vee x = b$$

For better readability, we omitted the arguments of F, so $F := F(x, u, u')$. The functional (2.1) is often called energy, as well.

2.3.2 Two-dimensional Calculus

In case of multiple, two-dimensional functions the theorem of Euler-Lagrange can be easily extended. For a two-dimensional energy depending on two variables

$$E(u, s) := \int_\Omega F(x_1, x_2, u, u_{x_1}, u_{x_2}, s, s_{x_1}, s_{x_2})\ d\mathbf{x}, \tag{2.2}$$

a minimiser has to satisfy the Euler-Lagrange equations

$$\frac{\partial}{\partial u}F - \frac{d}{dx_1}\left(\frac{\partial}{\partial u_{x_1}}F\right) - \frac{d}{dx_2}\left(\frac{\partial}{\partial u_{x_2}}F\right) = 0 \tag{2.3}$$

$$\frac{\partial}{\partial s}F - \frac{d}{dx_1}\left(\frac{\partial}{\partial s_{x_1}}F\right) - \frac{d}{dx_2}\left(\frac{\partial}{\partial s_{x_2}}F\right) = 0 \tag{2.4}$$

and the natural boundary conditions

$$\mathbf{n}^\top \begin{pmatrix} \frac{\partial}{\partial u_{x_1}}F \\ \frac{\partial}{\partial u_{x_2}}F \end{pmatrix} = 0, \text{ for } x \in \partial\Omega, \tag{2.5}$$

$$\mathbf{n}^\top \begin{pmatrix} \frac{\partial}{\partial s_{x_1}}F \\ \frac{\partial}{\partial s_{x_2}}F \end{pmatrix} = 0, \text{ for } x \in \partial\Omega, \tag{2.6}$$

for the boundary $\partial\Omega$ with normal vector $\mathbf{n}$. In Chapter 5 the above equations will be used to derive an algorithm.

2.4 Gradient Descent

The Euler-Lagrange equations from Section 2.3 can be interpreted in the sense of Gâteaux derivatives, that are a generalisation of directional derivatives to function spaces. If we define

$$\frac{\delta}{\delta u}E(u, s) := \frac{\partial}{\partial u}F - \frac{d}{dx_1}\left(\frac{\partial}{\partial u_{x_1}}F\right) - \frac{d}{dx_2}\left(\frac{\partial}{\partial u_{x_2}}F\right)$$

$$\frac{\delta}{\delta s}E(u, s) := \frac{\partial}{\partial s}F - \frac{d}{dx_1}\left(\frac{\partial}{\partial s_{x_1}}F\right) - \frac{d}{dx_2}\left(\frac{\partial}{\partial s_{x_2}}F\right)$$

then we get for equations (2.3) and (2.4)

$$\frac{\delta}{\delta u}E(u,s) = 0$$
$$\frac{\delta}{\delta s}E(u,s) = 0.$$

$\frac{\delta}{\delta u}E(u,s)$ and $\frac{\delta}{\delta s}E(u,s)$ are now the Gâteaux derivatives of E. The gradient descent corresponding to these derivatives is given by

$$\frac{\partial}{\partial t}u = -\gamma \frac{\delta}{\delta u}E(u,s)$$
$$\frac{\partial}{\partial t}s = -\gamma \frac{\delta}{\delta s}E(u,s),$$

where t is the time and $\gamma > 0$ is some speed factor. In our case, we set γ to be 1 and hence get as gradient descent equations:

$$\frac{\partial}{\partial t}u = -\frac{\delta}{\delta u}E(u,s) \tag{2.7}$$
$$\frac{\partial}{\partial t}s = -\frac{\delta}{\delta s}E(u,s) \tag{2.8}$$

These equations can be used to find a minimiser of (2.2). If the energy E is strictly convex, then the gradient descent will globally converge and find the unique minimiser.

2.5 Finite Differences

To solve systems like (2.7) and (2.8) for discrete images, we need to approximate partial derivatives with pixel values. We assume the pixel values to be given on an equidistant grid with gridsize h_1 in x-direction and gridsize h_2 in y-direction. With the help of the Taylor expansion, one can derive the following discrete expressions that approximate one-dimensional derivatives:

First derivative, forward difference: $u'(x) = \frac{u_{x+1} - u_x}{h} + O(h)$

First derivative, backward difference: $u'(x) = \frac{u_x - u_{x-1}}{h} + O(h)$

First derivative, central difference: $u'(x) = \frac{u_{x+1} - u_{x-1}}{2h} + O(h^2)$

Second derivative, central difference: $u''(x) = \frac{u_{x+1} - 2u_x + u_{x-1}}{h^2} + O(h^2)$

There are many more finite difference expressions for all orders of derivatives, incorporating forward, backward and central differences, or using more pixels to get a more precise estimation.

In case of two-dimensional images, we will just apply the above one-dimensional differences in the corresponding direction, so for example:

$$\frac{\partial}{\partial x_1}u(\mathbf{x}) = \frac{u_{x_1+1,x_2} - u_{x_1-1,x_2}}{2h_1} + O(h_1^2)$$
$$\frac{\partial}{\partial t}u(\mathbf{x}) = \frac{u^{k+1}_{x_1,x_2} - u^k_{x_1,x_2}}{\tau} + O(\tau)$$

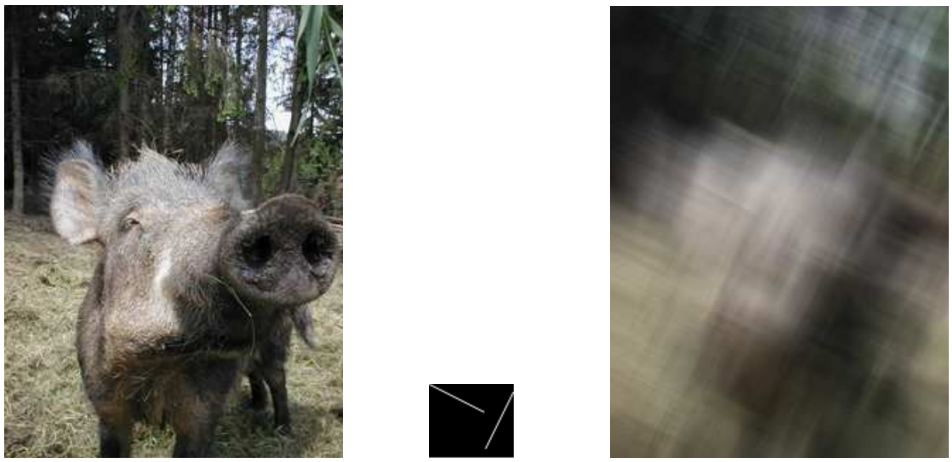
We have introduced a time index k as superscript and a timestep size τ in the last equation. This notation will be kept for time discretisations.

3 Blur and Deblurring

We will now take a closer look on blur itself and the deblurring task. By studying this topics in more detail technical terms are introduced and important observations for our forthcoming approach are obtained. We also give the standard deblurring model, that is vital for the understanding of the modified model, which will be developed in Chapter 4.

3.1 What is Blur?

Blur is an artefact that occurs in images due to unregularities in image acquisition. The blurring process can be described as some kind of weighted averaging of pixel values in a certain neighbourhood. Figure 3.1 illustrates this. Each pixel of the blurred picture on the right is a weighted average of pixels from the original image on the left. The weights of this averaging process are visualised in the blur image in the middle, where the neighbourhood of a pixel is shown. White pixels give a large contribution, whereas dark pixels give a small contribution.



original image blur blurred image

Figure 3.1: A blurring process. **Left:** Original, unblurred image. **Middle:** Visualisation of the blur. **Right:** Blurred image. *Author: D. Theis* [23]

3.1.1 Spatial Variance

If the blur is the same for every pixel in the blurred image, then we talk about spatially invariant blur. Figure 3.1 is an example for such a process.

There is also another type of blur, namely spatially variant blur. In this case the blur may vary over the image. An example of an image that results from this type of blur is

figure 3.2. Clearly, the television in the background is much more blurred than the pets in the foreground.

Figure 3.2: Spatially variant blur. *Authors: S. Bae, F. Durand* [1]

3.1.2 Causes

Blurring arises from certain conditions in the image taking process. Possible causes of blur are:

- camera or object motion
- defocussed objects
- atmospheric perturbations
- lens errors
- other reasons

Although two different causes can give the same blur, in most cases the type of a blur is significant for a cause. So one can conclude from an image, how it got blurred. Figure 3.3 shows some common types of blur.

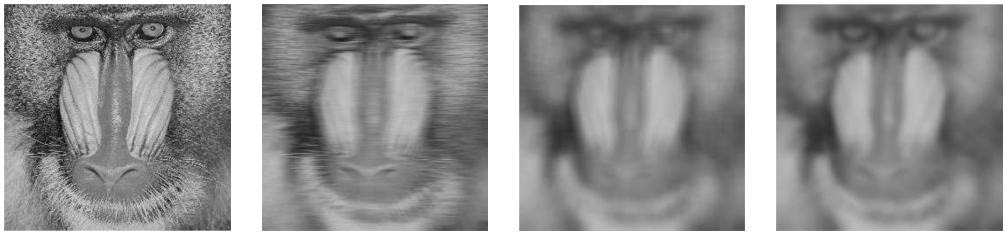
motion blur | defocus blur | Gaussian blur | natural "shake" blur

Figure 3.3: Different types of blur. *Authors: Q. Shan et al.* [17]

Motion blur of this kind is normally caused by camera or object motion in x-direction during the camera exposure time, whereas the disc-like defocus blur comes from photographing objects that are outside the focus of the camera. Atmospheric perturbations

or lens errors are normally the cause of Gaussian blurs and the "shake" blur arises e.g. when one takes a picture with a small hand-held camera and his hands are shaking.

One now may ask oneself how pictures deviated by such blurs look like. This is illustrated by figure 3.4. In this example, the images are blurred in a spatially invariant way. It is easy to observe that the blurred images look differently.



sharp image | motion blurred image | defocussed image | Gaussian blurred image

Figure 3.4: Blurred images. *Author: J. Weickert* [20]

3.2 Model

As described previously in Section 3.1, blurring can be modelled as a weighted average. In the continuous model for blurring, which will be introduced now, this averaging will lead to an integration with a weighting function. The following models and the notation are oriented on the formulations in [23].

3.2.1 Spatially Variant Blur

Spatially variant blurring can be described as a linear process:

$$f(\mathbf{x}) = \int_{\mathbb{R}^2} u(\mathbf{y})\ H(\mathbf{x}, \mathbf{y})\ d\mathbf{y} + n(\mathbf{x}) \tag{3.1}$$

The observed (blurred) image f results from an averaging of the original (unblurred) image u, weighted by the point spread function H. For the sake of shortness, any point spread function will be just called PSF. H may vary for each pixel of the observed image and therefore has both, $\mathbf{x}$ and $\mathbf{y}$, as arguments. It is 4-dimensional. Actually, the blurs in figures 3.1 and 3.3 visualise the PSF for a given pixel $\mathbf{x}$, so the figures show $H(\mathbf{x}, \cdot)$. We also assume that that the additive noise n is small and can later be neglected. The integral ranges over the whole space $\mathbb{R}^2$, u and H are assumed to be defined on this domain. Boundary considerations will not be included at this point of the modelling.

3.2.2 Spatially Invariant Blur

In case of spatially invariant blur the model reduces to:

$$f(\mathbf{x}) = \int_{\mathbb{R}^2} u(\mathbf{y})\ h(\mathbf{x} - \mathbf{y})\ d\mathbf{y} + n(\mathbf{x}) \tag{3.2}$$

The PSF h is now only depending on the relative distance between $\mathbf{x}$ and $\mathbf{y}$. Such a two-dimensional PSF is denoted by a lower-case letter. If one ignores the noise n, then f is just the convolution of u with h, so we can rewrite equation (3.2) as follows, using the symbol $*$ as convolution operator:

$$f(\mathbf{x}) = (u * h)(\mathbf{x}) + n(\mathbf{x}) \tag{3.3}$$

3.2.3 Normalisation

The spatially invariant PSF H and the one for the spatially invariant case h have to fulfil an important requirement: the normalisation property.

$$\forall\ \mathbf{y} \in \mathbb{R}^2 : \int_{\mathbb{R}^2} H(\mathbf{x}, \mathbf{y})\ d\mathbf{x} = 1 \tag{3.4}$$

$$\forall\ \mathbf{y} \in \mathbb{R}^2 : \int_{\mathbb{R}^2} h(\mathbf{x} - \mathbf{y})\ d\mathbf{x} = 1 \tag{3.5}$$

The normalisation property follows from the energy conservation law and basically says that no pixel of the original image f can contribute more energy to the blurred image f than it has.

3.2.4 Motion Blur

We will explicitly distinguish the case of blur that occurs only in x-direction of an image. Because this blurring is only happening horizontally, the model reduces to be just one-dimensional:

$$f(\mathbf{x}) = \int_{\mathbb{R}} u\begin{pmatrix} y \\ x_2 \end{pmatrix}\ H(\mathbf{x}, y)\ dy + n(\mathbf{x}) \tag{3.6}$$

$$f(\mathbf{x}) = \int_{\mathbb{R}} u\begin{pmatrix} y \\ x_2 \end{pmatrix}\ h(x_1 - y)\ dy + n(\mathbf{x}) \tag{3.7}$$

For notational convenience, two parentheses have been omitted in argument of two-dimensional functions, e.g. $u\begin{pmatrix} y \\ x_2 \end{pmatrix} := u\left(\begin{pmatrix} y \\ x_2 \end{pmatrix}\right)$. Of course we still have to fulfil the normalisation property:

$$\forall\ x_2 y \in \mathbb{R} : \int_{\mathbb{R}} H(\mathbf{x}, y)\ dx_1 = 1 \tag{3.8}$$

$$\forall\ y \in \mathbb{R} : \int_{\mathbb{R}} h(x_1 - y)\ dx_1 = 1 \tag{3.9}$$

As visualised in figure 3.3, different blur causes give different PSFs. For motion blur, the PSF h, which is now just one-dimensional, is assumed to be a symmetric box function with radius r:

$$h(x) = \begin{cases} \frac{1}{2r} & \text{if } -r \leq x \leq r \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

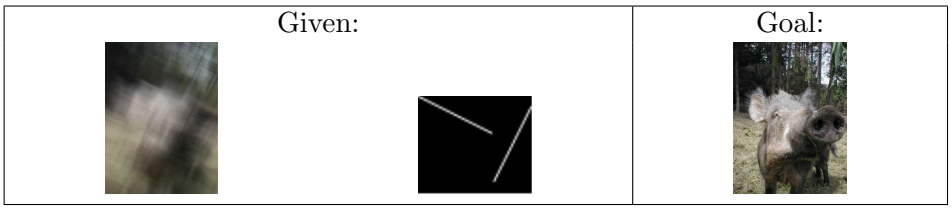
3.3 Deblurring

In real images, blurring artefacts appear quite frequently. Thus deblurring, the task of making a blurred image sharp again, is a very important problem in image processing. The deblurring problem can be posed in the spatially variant as well as in the spatially invariant setting. Due to the strong relation of blurring to convolution (see equation (3.3)), deblurring is often just called deconvolution, even in the spatially variant case.

3.3.1 Non-Blind and Blind Deblurring

There are mainly two different kinds of deblurring, depending on how much information is assumed to be known. Figure 3.3.1 shows both variants: non-blind and blind deblurring.

- Non-blind deblurring:



- Blind deblurring:

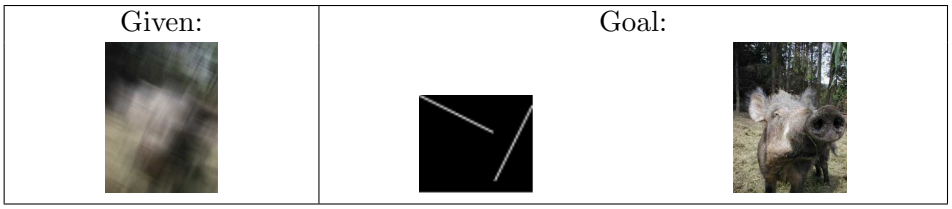


Figure 3.5: Non-blind and blind deblurring. *Author: D. Theis* [23]

In non-blind deblurring we assume that the observed image and the PSF is known. So the input of a non-blind deblurring algorithm would be a blurred image as well as the corresponding PSF, the output would be the unblurred image. Whereas in the case of blind deblurring we only have a blurred image and we are searching for the corresponding PSF and the unblurred image.

3.3.2 Ill-Posedness

Unfortunately the deblurring problem is severely ill-posed, even in the spatially invariant and non-blind setting without noise. To illustrate this we consider the Fourier transform $\mathcal{F}$.

Equation (3.3) shows that blurring an image with a spatially invariant PSF without the presence of noise is equivalent to convolving the image with the PSF. We know by the

convolution theorem that convolution in the spatial domain corresponds to multiplication in the Fourier domain:

$$\mathcal{F}[u * h] = \mathcal{F}[u] \cdot \mathcal{F}[h] \tag{3.11}$$

By combining equations (3.3) and (3.11) we get for the blurred image:

$$f = \mathcal{F}^{-1}\left[\mathcal{F}[u] \cdot \mathcal{F}[h]\right] \tag{3.12}$$

If now some frequencies of h are zero (which is often the case for real blurs), we could change the corresponding frequency components of u arbitrarily without changing f, because we would just multiply these changed components with zero (see equation (3.12)). All versions of u, changed in this way, would be a correct solution of the deblurring problem. Thus the solution of the deblurring problem is not unique and the problem itself is ill-posed. The ill-posedness of the problem gets even worse if one allows noise, spatial variance or blindness.

Anyway, the deblurring problem is still a very important task, which arises in many practical applications. Therefore loads of different approaches have been developed to solve it. They differ in what assumptions they make and how they cope with the ill-posedness by including for example prior knowledge or regularisation.

3.4 Depth

The crucial observation of this thesis is that blur can contain depth information. We will illustrate this for the case of motion blur in the pinhole camera model. Figure 3.6 shows a blurring process caused by camera motion in the pinhole model.

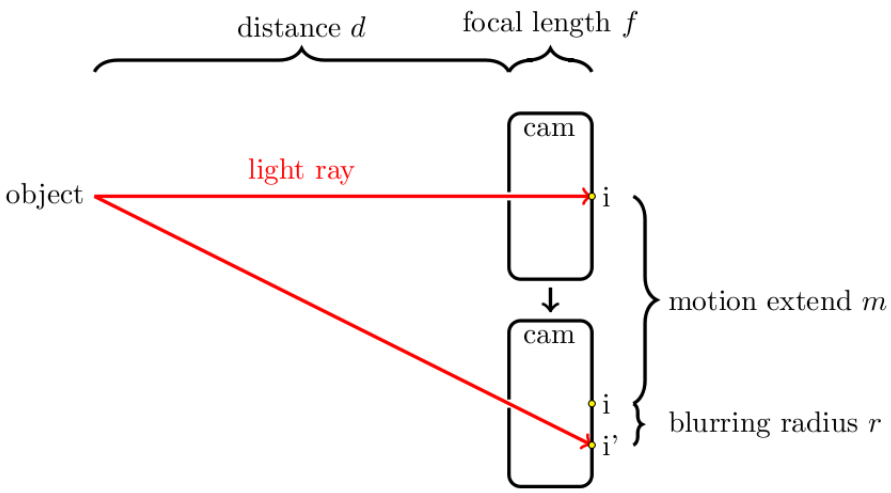


Figure 3.6: A schematic figure of motion blur, caused by camera motion in the pinhole model. The camera is moving downwards and covers a distance of m.

According to the notation of figure 3.6, we assume that a pinhole camera with focal length f is moving during the exposure time, covering a distance of m. An object of

distance d gets blurred by this motion, the corresponding blurring radius is r. By the intercept theorem it follows that

$$\frac{d+f}{d} = \frac{m+r}{m},$$

this is equivalent to

$$r = \frac{f \cdot m}{d}.$$

So we know that the blurring radius r times the object distance d is a constant for an image taken under camera motion. This leads to the fact that there exists an inversely proportional dependency between r and d and thus blur and depth are related. If we know the depth we can estimate the blur and if we know the blur we can estimate the depth.

A synthetic image, that is blurred under uniform camera motion in horizontal direction, is shown in figure 3.7, together with the sharp original image. It is clearly observable, that for example the tower in the back is lesser blurred than the sign in front of the house, which is closer to the camera. This illustrates the correspondence between blur and depth.

Figure 3.7: Image blurred under camera motion. **Left:** Original, sharp image. **Right:** Blurred image. *Author: T. Becker* (2008)

Similar considerations can be made for defocus and motion blur with an aperture camera. Lin and Chang [11] show that the assumption of inverse proportionality is still valid here. But often we will need some additional approximations and assumptions. For instance, it could be possible that sharp objects are in the foreground and blurred objects are in the background in the defocal setting. Figure 3.2 is an example for this.

4 Underlying Model

Our goal now is to introduce some notion of depth in our current model from Section 3.2. Later, this depth containing model will be used to retrieve the sharp original image and the depth map of the scene from a single spatially variant blurred image. So basically we want to retrieve a sharp three-dimensional image from a blurred two-dimensional one.

4.1 Idea

There are quite successful approaches for blind image deblurring with a spatially invariant blur (see for example [8], [17] or [26]). In case of blind, spatially variant deblurring the situation is different. This task is just too difficult, because we want to retrieve a four-dimensional PSF and a two-dimensional sharp image by just using the information from a single two-dimensional blurred image. As far as we know, no method successfully dealt with this problem up to now, although an effective algorithm to cope with unknown spatially variant blur would be very desirable. Anyway, we have to come up with some way to work with spatial variance, to estimate depth only by using one blurred image.

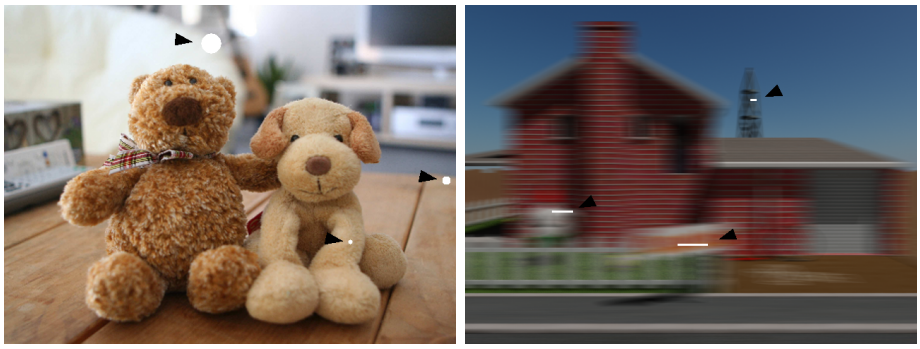


Figure 4.1: Different blur extents. The blur for 3 specific pixels in each image is visualised in white and marked with a black arrow. **Left:** Defocussed image, the PSF shape is a disc. **Right:** Motion blurred image, the PSF shape is a horizontal line.

The remedy is the observation that in many practical cases of spatially variant blur, the blur extent changes, but the shape of the blur remains constant. We showed in Section 3.1.2, that different causes give different types of blur. So it is the case that in practise normally just the strength of the blur is varying, but not the type, because we have just one blurring cause for the whole image. In terms of the mathematical model from Section 3.2.1 the shape of the PSF does not change, but its spatial extent does. Additionally, the shape is often known in advance (like a disc for defocus), or can be estimated quite easily from the image. Figure 4.1 illustrates this. On the left side, we have a defocussed image

and whereas on the right side the image is deviated under motion blur. In both cases we have visualised the approximated blur in red colour for three pixels. For defocus the blur remains to be a circle, but the radius is varying, whereas for motion blur, the blur is always a line, but its length is changing.

4.2 Modelling

As discussed in the section before, we would like to have the following properties for the combined deblurring and depth estimation approach. The model should be:

- spatially invariant and non-blind regarding the shape of the PSF
- spatially variant and blind regarding the spatial extent of the PSF

Deblurring with such assumptions is often called semi-blind deblurring. The idea is to extend the model of spatially invariant blur (3.2) with a scaling factor s in the argument of the PSF. So we assume that a certain PSF is given and we just scale it to get spatial variance.

$$f(\mathbf{x}) = \int_{\mathbb{R}^2} u(\mathbf{y})\ h\left((\mathbf{x}-\mathbf{y})s(\mathbf{y})\right)\ d\mathbf{y} \tag{4.1}$$

Figure 4.2 demonstrates what effect the scaling factor s has for a circle-like PSF.

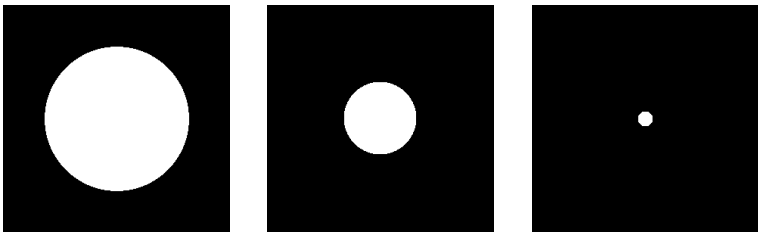


Figure 4.2: Effect of s to a PSF. The factor s scales the PSF, here a defocus-like circle, to a certain size. **Left:** PSF for a small value of s. **Middle:** PSF for a medium value of s. **Right:** PSF for a large value of s.

The scaling may vary for every pixel of the original image u, so s is a function depending on $\mathbf{y}$. However, equation (4.1) is not a blurring model any more, because we have lost the normalisation property (equations (3.4) and (3.5)). The remedy is an additional factor to scale the values of the PSF:

$$f(\mathbf{x}) = \int_{\mathbb{R}^2} u(\mathbf{y})\ s(\mathbf{y})h\left((\mathbf{x}-\mathbf{y})s(\mathbf{y})\right)\ d\mathbf{y}$$

This is now our final blurring model that allows spatial variance by the scaling factor s. It can be regarded as spatially variant blurring model as given in equation (3.1) by defining

$$\tilde{H}(\mathbf{x},\mathbf{y}) := s(\mathbf{y})h\left((\mathbf{x}-\mathbf{y})s(\mathbf{y})\right) \tag{4.2}$$

and setting H to be $\tilde{H}$. The appendix shows in Section A.1, that the normalisation property is now fulfilled, under the assumption that the invariant blur h is normalised.

4.3 Reduction to Motion Blur

In the same way as in Section 3.2.4 we can reduce the new model to one-dimensional blur in x-direction.

$$f(\mathbf{x}) = \int_{\mathbb{R}} u\begin{pmatrix} y \\ x_2 \end{pmatrix} \; s\begin{pmatrix} y \\ x_2 \end{pmatrix} h\left((x_1 - y)s\begin{pmatrix} y \\ x_2 \end{pmatrix}\right) \; dy$$

In this case, s is still a two-dimensional function, but h reduces to be one-dimensional and the integral is only ranging over one dimension, as well.

We can again fit this model into the spatially variant formulation of equation 3.6 by setting H to be $\hat{H}$ and defining $\hat{H}$ as:

$$\hat{H}(\mathbf{x}, y) := s\begin{pmatrix} y \\ x_2 \end{pmatrix} h\left((x_1 - y)s\begin{pmatrix} y \\ x_2 \end{pmatrix}\right) \tag{4.3}$$

4.4 Relation to Depth

The above definition of our model corresponds exactly to the observations from Section 3.4, where we have shown that under some reasonable assumptions, the equation

$$\hat{r} = \frac{f \cdot m}{d} \tag{4.4}$$

holds for the depth of a blurred object d, the blurring radius $\hat{r}$, the focal length f and the motion extent m. We will illustrate this correspondence for a one-dimensional motion blur PSF h, which is defined via equation (3.10). h has radius r and can be plugged into the definition of the spatially variant PSF $\hat{H}$ from equation (4.3). Using the definition $d := s\begin{pmatrix} y \\ x_2 \end{pmatrix}$, $\hat{H}$ has now the following form:

$$\begin{aligned}
&\hat{H}(\mathbf{x}, y) \\
&= d \; h\left((x_1 - y)d\right) && \text{by (4.3)} \\
&= \begin{cases} d\frac{1}{2r} & \text{if } -r \le xd \le r \\ 0 & \text{otherwise} \end{cases} && \text{by (3.10)} \\
&= \begin{cases} \frac{1}{2\hat{r}} & \text{if } -\hat{r} \le x \le \hat{r} \\ 0 & \text{otherwise} \end{cases} && \text{by defining } \hat{r} := \frac{r}{d}
\end{aligned}$$

So the actual blurring radius of $\hat{H}$ is given by $\hat{r} = \frac{r}{d}$. This corresponds perfectly to equation (4.4), where the blurring radius r of the invariant blur h is the constant $f \cdot m$ and the depth d is the scaling factor s. Hence the depth of a scene point at the position $(y, x_2)^\top$ can be modelled by s and thus s is a depth map for the whole scene. However, we can only estimate the absolute depth if the following equation holds:

$$r = f \cdot m \tag{4.5}$$

If this is not the case, then s can only be a relative depth. We conclude that the initial blurring radius r is somewhat arbitrary, if we do not know the exact camera parameters,

but it scales the depth map s. The inverse proportionality between the distance and the blurring radius (equation (4.4)) holds for our model, and the relative depth of two objects at different distances is given by $\frac{r_1}{r_2} = \frac{d_2}{d_1}$.

In one sentence: We assume that near objects are stronger blurred than far objects and can therefore combine depth estimation and deblurring.

Another way to interpret the model is to consider it as model with parametrised PSF, similar to the one in [27]. For every pixel the PSF has just one parameter: the depth at this point.

4.5 Recapitulation

Following the discussion from the sections before, the model and the notation underlying this thesis are given by:

$$f(\mathbf{x}) = \int_{\mathbb{R}^2} u(\mathbf{y})\ s(\mathbf{y}) h\left((\mathbf{x}-\mathbf{y})s(\mathbf{y})\right)\ d\mathbf{y} \tag{4.6}$$

- f: Observed, blurred image
- u: Original, unblurred image
- h: Spatially invariant point spread function (PSF), normalised.
- s: Depth map of the image
- $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, $\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$

The names f, u, h and s will be used as meta-variables for the rest of this thesis and hence will keep the interpretation given above. Assuming one-dimensional blur in horizontal direction, the model reduces to:

$$f(\mathbf{x}) = \int_{\mathbb{R}} u\begin{pmatrix} y \\ x_2 \end{pmatrix}\ s\begin{pmatrix} y \\ x_2 \end{pmatrix} h\left((x_1 - y)s\begin{pmatrix} y \\ x_2 \end{pmatrix}\right)\ dy \tag{4.7}$$

5 Continuous Optimisation

To solve the problem that was stated at the beginning of this thesis, i.e. to find an unblurred version of a single spatially variant blurred image and to combine this with depth estimation, we have to come up with an algorithm. This task will be tackled in the two following chapters. Therefore we will formulate an energy in sense of Section 2.3 and use the calculus of variations to retrieve a solution in continuous space by gradient descent.

5.1 The Functional

5.1.1 General Functional

If we take the model for spatial variant blur (4.6) and reorder it,

$$\begin{aligned} & f(\mathbf{x}) = \int_{\mathbb{R}^2} u(\mathbf{y})\ s(\mathbf{y}) h\left((\mathbf{x}-\mathbf{y}) s(\mathbf{y})\right)\ d\mathbf{y} \\ \Leftrightarrow \quad & 0 = f(\mathbf{x}) - \int_{\mathbb{R}^2} u(\mathbf{y})\ s(\mathbf{y}) h\left((\mathbf{x}-\mathbf{y}) s(\mathbf{y})\right)\ d\mathbf{y} \\ \Leftrightarrow \quad & 0 = \left(f(\mathbf{x}) - \int_{\mathbb{R}^2} u(\mathbf{y})\ s(\mathbf{y}) h\left((\mathbf{x}-\mathbf{y}) s(\mathbf{y})\right)\ d\mathbf{y} \right)^2 \end{aligned}$$

then we get a non-negative expression, that is minimised by the original image u and the depth map s. So we could take the squared expression and search for some u and s, that are minimising it. Unfortunately this task is highly ill-posed, because the squared expression is for example minimised by many possible u, as shown in Section 3.3.2.

As a remedy we propose an energy that incorporates regularisation terms for both unknown functions, u and s. The energy will fit in the framework of equation (2.2), such that we can apply the calculus of variations to it. We follow the conventions of other variational approaches, the formalism is particularly based on the formulations in [24]. The energy functional reads:

$$\begin{aligned} E[u,s] \ := \ \frac{1}{2} \int_{\mathbb{R}^2} & \underbrace{\Phi\left(\left(f(\mathbf{x}) - \int_{\mathbb{R}^2} u(\mathbf{y})\ s(\mathbf{y}) h((\mathbf{x}-\mathbf{y}) s(\mathbf{y}))\ d\mathbf{y} \right)^2 \right)}_{\text{data term}} \\ & + \underbrace{\alpha\ \Psi_1\left(|\nabla u(\mathbf{x})|^2\right) + \beta\ \Psi_2\left(|\nabla s(\mathbf{x})|^2\right)}_{\text{smoothness terms}}\ d\mathbf{x} \end{aligned} \tag{5.1}$$

The algorithm of this thesis is basically about finding the unblurred image and the depth map by minimising this energy: $(u, s) = \min_{u',s'} E[u', s']$. Unfortunately, the given functional is not convex, such that we could get stuck in local minima, when we are searching for the minimiser.

If u and s do not fulfil the model assumptions (4.6), then these violations get penalised by the data term. The smoothness terms penalise high deviations and therefore lead to a smooth solution. α and β are regularisation parameters (or also called smoothness weights) and should be positive. The penalising functions Φ, Ψ_1 and Ψ_2 steer the penalisation and can help to make the energy less sensitive against outliers or to enforce a piecewise smooth solution. These functions should be positive, differentiable and increasing, in order to give a reasonable penalisation.

For clearness, we will now introduce some abbreviations. The data term

$$E_1[u, s] := \frac{1}{2} \int_{\mathbb{R}^2} \Phi\left((R_{f,h}[u, s](\mathbf{x}))^2 \right) \, d\mathbf{x} \tag{5.2}$$

depends on the residual R:

$$\begin{aligned} \mathbb{R}_{f,h}[u, s](\mathbf{x}) &:= f(\mathbf{x}) - \int_{\mathbb{R}^2} u(\mathbf{y}) \; s(\mathbf{y}) h((\mathbf{x} - \mathbf{y}) s(\mathbf{y})) \, d\mathbf{y} \qquad (5.3) \\ &= f(\mathbf{x}) - \int_{\mathbb{R}^2} u(\mathbf{y}) \; \tilde{H}(\mathbf{x}, \mathbf{y}) \, d\mathbf{y}. \qquad \text{by equation (4.2)} \end{aligned}$$

The smoothness terms are denoted as:

$$E_2[u] := \frac{\alpha}{2} \int_{\mathbb{R}^2} \Psi_1 \left(|\nabla u(\mathbf{x})|^2 \right) \, d\mathbf{x} \tag{5.4}$$

$$E_3[s] := \frac{\beta}{2} \int_{\mathbb{R}^2} \Psi_2 \left(|\nabla s(\mathbf{x})|^2 \right) \, d\mathbf{x} \tag{5.5}$$

So in the end the functional can be formulated as:

$$E[u, s] = E_1[u, s] + E_2[u] + E_3[s]$$

In terms of equation (2.2) we also define:

$$\begin{aligned} &F(x_1, x_2, u, u_{x_1}, u_{x_2}, s, s_{x_1}, s_{x_2}) \qquad (5.6) \\ &:= F_1(x_1, x_2, u, s) + F_2(x_1, x_2, u_{x_1}, u_{x_2}) + F_3(x_1, x_2, s_{x_1}, s_{x_2}) \\ &:= \frac{1}{2} \Phi((f(\mathbf{x}) - \int_{\mathbb{R}^2} u(\mathbf{y}) \; s(\mathbf{y}) h((\mathbf{x} - \mathbf{y}) s(\mathbf{y})) \, d\mathbf{y})^2) + \frac{\alpha}{2} \Psi_1 \left(|\nabla u(\mathbf{x})|^2 \right) + \frac{\beta}{2} \Psi_2 \left(|\nabla s(\mathbf{x})|^2 \right) \end{aligned}$$

5.1.2 Motion Blur Functional

The energy in Section 5.1.1 was based on the general blur model (4.6). By assuming instead the more restricted model from (4.7), we get:

$$\hat{E}[u,s] := \frac{1}{2}\int_{\mathbb{R}^2} \underbrace{\Phi\left(\left(f(\mathbf{x}) - \int_{\mathbb{R}} u\begin{pmatrix} y \\ x_2 \end{pmatrix}\ s\begin{pmatrix} y \\ x_2 \end{pmatrix} h\left((x_1-y)s\begin{pmatrix} y \\ x_2 \end{pmatrix}\right)\ dy\right)^2\right)}_{\text{data term}} + \underbrace{\alpha\ \Psi_1\left(|\nabla u(\mathbf{x})|^2\right) + \beta\ \Psi_2\left(|\nabla s(\mathbf{x})|^2\right)}_{\text{smoothness terms}}\ d\mathbf{x} \tag{5.7}$$

As before, we will abbreviate the residual

$$\begin{aligned}\hat{R}_{f,h}[u,s](\mathbf{x}) :&= f(\mathbf{x}) - \int_{\mathbb{R}} u\begin{pmatrix} y \\ x_2 \end{pmatrix}\ s\begin{pmatrix} y \\ x_2 \end{pmatrix} h\left((x_1-y)s\begin{pmatrix} y \\ x_2 \end{pmatrix}\right)\ dy \qquad (5.8)\\ &= f(\mathbf{x}) - \int_{\mathbb{R}} u\begin{pmatrix} y \\ x_2 \end{pmatrix}\ \hat{H}(\mathbf{x},y)\ dy \qquad \text{by equation (4.3)}\end{aligned}$$

and the data term

$$\hat{E}_1[u,s] := \frac{1}{2}\int_{\mathbb{R}^2} \Phi\left(\left(\hat{R}_{f,h}[u,s](\mathbf{x})\right)^2\right)\ d\mathbf{x} \tag{5.9}$$

such that the complete energy can be shortened to

$$\hat{E}[u,s] = \hat{E}_1[u,s] + E_2[u] + E_3[s].$$

5.2 Euler-Lagrange Equations

The functional (5.7) fits into the variational framework of Section 2.3.2. In order to use this calculus, we have to derive the Euler-Lagrange equations (2.3) and (2.4). Section 2.4 shows that these equations can be interpreted in the sense of Gâteaux derivatives, this is why we will sometimes call equation (2.3) "gradient by u" and equation (2.4) "gradient by s" respectively. These gradients will be then used to get a solution for the unblurred image u and the depth map s.

Our energy (5.7) is formulated in such a way, that the data Term E_1 is the only term that is depending on u and s directly. Also, the smoothness term E_2 is the only term depending on derivatives of u, and the smoothness term E_3 is the only term depending on derivatives of s. Therefore we can consider these terms separately:

$$\frac{\delta}{\delta u}E(u,s) = \frac{\delta}{\delta u}E_1(u,s) - \frac{\delta}{\delta u}E_2(u) \tag{5.10}$$

$$\frac{\delta}{\delta s}E(u,s) = \frac{\delta}{\delta s}E_1(u,s) - \frac{\delta}{\delta s}E_3(s) \tag{5.11}$$

5.2.1 Diffusion

We start with looking at the parts of the Euler-Lagrange equations, that are resulting from the smoothness terms. Regarding equations (2.3) and (2.4), these parts are

$$\frac{\delta}{\delta u}E_2[u] := -\frac{d}{dx_1}\left(\frac{\partial}{\partial u_{x_1}}F\right) - \frac{d}{dx_2}\left(\frac{\partial}{\partial u_{x_2}}F\right)$$
$$\frac{\delta}{\delta s}E_2[s] := -\frac{d}{dx_1}\left(\frac{\partial}{\partial s_{x_1}}F\right) - \frac{d}{dx_2}\left(\frac{\partial}{\partial s_{x_2}}F\right)$$

By cancelling out all terms of the energy, that do not depend on u_{x_1} or u_{x_2} (see also equation (5.6)), we get for the equation depending on u:

$$\begin{aligned}
&-\frac{d}{dx_1}\left(\frac{\partial}{\partial u_{x_1}}F\right) - \frac{d}{dx_2}\left(\frac{\partial}{\partial u_{x_2}}F\right)\\
&= -\frac{d}{dx_1}\left(\frac{\partial}{\partial u_{x_1}}F_2(x_1,x_2,u_{x_1},u_{x_2})\right) - \frac{d}{dx_2}\left(\frac{\partial}{\partial u_{x_2}}F_2(x_1,x_2,u_{x_1},u_{x_2})\right)\\
&= -\frac{d}{dx_1}\left(\frac{\partial}{\partial u_{x_1}}\frac{\alpha}{2}\Psi_1\left(|\nabla u(\mathbf{x})|^2\right)\right) - \frac{d}{dx_2}\left(\frac{\partial}{\partial u_{x_2}}\frac{\alpha}{2}\Psi_1\left(|\nabla u(\mathbf{x})|^2\right)\right)\\
&= -\frac{d}{dx_1}\left(\frac{\alpha}{2}\Psi_1'\left(|\nabla u(\mathbf{x})|^2\right)\frac{\partial}{\partial u_{x_1}}|\nabla u(\mathbf{x})|^2\right) - \frac{d}{dx_2}\left(\frac{\alpha}{2}\Psi_1'\left(|\nabla u(\mathbf{x})|^2\right)\frac{\partial}{\partial u_{x_2}}|\nabla u(\mathbf{x})|^2\right)\\
&= -\frac{d}{dx_1}\left(\alpha\Psi_1'\left(|\nabla u(\mathbf{x})|^2\right)u_{x_1}(\mathbf{x})\right) - \frac{d}{dx_2}\left(\alpha\Psi_1'\left(|\nabla u(\mathbf{x})|^2\right)u_{x_2}(\mathbf{x})\right)\\
&= -\alpha\ \mathrm{div}\left(\Psi_1'\left(|\nabla u(\mathbf{x})|^2\right)\nabla u(\mathbf{x})\right)
\end{aligned}$$

Identically we get an equation depending on s. In summary we have:

$$\frac{\delta}{\delta u}E_2[u] := -\alpha\ \mathrm{div}\left(\Psi_1'\left(|\nabla u(\mathbf{x})|^2\right)\nabla u(\mathbf{x})\right) \tag{5.12}$$
$$\frac{\delta}{\delta s}E_3[s] := -\beta\ \mathrm{div}\left(\Psi_2'\left(|\nabla s(\mathbf{x})|^2\right)\nabla s(\mathbf{x})\right) \tag{5.13}$$

At this point it should be clear why this section is called “diffusion”. It is because the divergence terms resulting from the Euler-Lagrange equations of the smoothness terms correspond to the non-linear isotropic diffusion equation. The expression containing the divergence is just the same. We will use this observation later in the discretisation part.

5.2.2 Equation with Respect to u

To derive the whole Euler-Lagrange equation for u (2.3), we have to determine the gradient by u for the data term E_1. This is done in appendix A.2.1, the resulting equation is given as:

$$\frac{\delta}{\delta u}E_1[u,s] := -\int_{\mathbb{R}^2}\Phi'\left(\left(R_{f,h}[w,s](\mathbf{y})\right)^2\right)\ R_{f,h}[u,s](\mathbf{y})\ s(\mathbf{x})h\left((\mathbf{y}-\mathbf{x})s(\mathbf{x})\right)\ d\mathbf{y} \tag{5.14}$$

Now we can formulate the gradient descent by u for our model, as it is described in Section 2.4. Following equations (2.7) and (5.10) we get:

$$\frac{\partial}{\partial t}u = \int_{\mathbb{R}^2} \Phi'\left((R_{f,h}[u,s](\mathbf{y}))^2\right)\ R_{f,h}[u,s](\mathbf{y})\ s(\mathbf{x})h((\mathbf{y}-\mathbf{x})s(\mathbf{x})\ d\mathbf{y} + \alpha\ \mathrm{div}\left(\Psi_1'\left(|\nabla u(\mathbf{x})|^2\right)\nabla u(\mathbf{x})\right) \tag{5.15}$$

5.2.3 Equation with Respect to s

The complete derivation of the Euler-Lagrange equation with respect to s is again shown in the appendix in Section A.2.2. Following this argumentation we state that the expression corresponding to the data term Euler-Lagrange equation for s reads:

$$\frac{\delta}{\delta s}E_1[u,s] := -\int_{\mathbb{R}^2} \Phi'\left((R_{f,h}[u,s](\mathbf{y}))^2\right)\ R_{f,h}[u,s](\mathbf{y})\ u(\mathbf{x})\ (h((\mathbf{y}-\mathbf{x})s(\mathbf{x})) + s(\mathbf{x})\ \langle\nabla h((\mathbf{y}-\mathbf{x})s(\mathbf{x})), \mathbf{y}-\mathbf{x}\rangle)d\mathbf{y} \tag{5.16}$$

By combining equations (2.8) and (5.11), we get the gradient descent for s:

$$\frac{\partial}{\partial t}s = \int_{\mathbb{R}^2} \Phi'\left((R_{f,h}[u,s](\mathbf{y}))^2\right)\ R_{f,h}[u,s](\mathbf{y})\ u(\mathbf{x})\ \big(h((\mathbf{y}-\mathbf{x})s(\mathbf{x})) + s(\mathbf{x})\ \langle\nabla h((\mathbf{y}-\mathbf{x})s(\mathbf{x})), \mathbf{y}-\mathbf{x}\rangle\big)d\mathbf{y} + \beta\ \mathrm{div}\left(\Psi_2'\left(|\nabla s(\mathbf{x})|^2\right)\nabla s(\mathbf{x})\right) \tag{5.17}$$

As far as we know, this is a novel partial differential equation, that was not published in the literature yet.

5.2.4 Motion Blur Equations

We can also derive the gradient descent based on the motion blur functional from Section 5.1.2. The diffusion parts of the gradient descent equations remain the same as for the general model, because the smoothness terms in the both functionals, (5.7) and (5.1), are the same. The expressions corresponding to the data term are different and their exact derivation is given in Sections A.2.3 and A.2.4 of the appendix. In particular they are given by:

$$\frac{\delta}{\delta u}\hat{E}_1[u,s] = -\int_{\mathbb{R}} \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2\end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2\end{pmatrix}\ s(\mathbf{x})h((y-x_1)s(\mathbf{x}))\ dy \tag{5.18}$$

$$\frac{\delta}{\delta s}\hat{E}_1[u,s] = -\int_{\mathbb{R}} \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2\end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2\end{pmatrix}\ u(\mathbf{x})\ \big(h\left((y-x_1)s(\mathbf{x})\right) + s(\mathbf{x})\ h'\left((y-x_1)s(\mathbf{x})\right)\ (y-x_1)\big)\ dy \tag{5.19}$$

Plugging all parts together, we get the reduced gradient descent equations for u and s, which are particularly designed for blurring only in horizontal direction, e.g. motion blur:

$$\begin{aligned}\frac{\partial}{\partial t}u = &\int_{\mathbb{R}} \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2\end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2\end{pmatrix}\ s(\mathbf{x})h((y-x_1)s(\mathbf{x}))\ dy \\ &+ \alpha\ \mathrm{div}\left(\Psi_1'\left(|\nabla u(\mathbf{x})|^2\right)\nabla u(\mathbf{x})\right) \qquad (5.20)\\ \frac{\partial}{\partial t}s = &\int_{\mathbb{R}} \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2\end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2\end{pmatrix}\ u(\mathbf{x})\ \left(h\left((y-x_1)s(\mathbf{x})\right)\right. \\ &\left.+ s(\mathbf{x})\ h'\left((y-x_1)s(\mathbf{x})\right)\ (y-x_1)\right)\ dy + \beta\ \mathrm{div}\left(\Psi_2'\left(|\nabla s(\mathbf{x})|^2\right)\nabla s(\mathbf{x})\right) \qquad (5.21)\end{aligned}$$

6 Discretisation

To make our algorithm usable, we have to discretise the continuous framework from Chapter 5, because we are only able to work on discrete images on a computer. By now, we will concentrate exclusively on the reduced motion blur model, which was introduced in 4.3. A discretisation for the general model is beyond the scope of this thesis and is left for future work.

The gradient descent equations (5.20) and (5.21) are able to find a solution in the continuous setting. Therefore we discretise these equations to get a solution for the discrete case. It is assumed that f, u and s are discrete images defined on an equidistant grid, where h_1 is the grid size in x-direction and h_2 is the grid size in y-direction.

6.1 Boundaries

In the discrete setting the image and the PSF have of course only a finite domain. Hence we have to introduce some boundaries and some boundary treatment. The original image u and the observed image f are assumed to have the same domain, which ranges from 1 to n in x-direction and from 1 to m in y-direction. The domain of the PSF h is normally smaller. Even though the whole idea behind this algorithm is to allow the PSF to have varying spatial extent, we anyway assume a maximal blurring radius r for computational convenience. This restriction is not a problem in practice, because r can be chosen quite large and even for large blurs it is still much smaller than the image domain. So the PSF h is assumed to range from $-r$ to r.

For the pixels right next to the boundaries we have to think about some boundary treatment. We decided to enforce some reflecting (Neumann) boundary conditions for both, the original image u and the depth map s. So if we assume that the domain of u and s is a rectangle Ω with boundary $\partial\Omega$ and $\mathbf{n}$ is the (outer) normal vector on $\partial\Omega$, then the reflecting boundary conditions can be formulated as:

$$\frac{\partial u}{\partial \mathbf{n}} = 0 \text{ on } \partial\Omega \times [0, \infty)$$
$$\frac{\partial s}{\partial \mathbf{n}} = 0 \text{ on } \partial\Omega \times [0, \infty)$$

These conditions are called reflecting boundary conditions, because one can think of them like that: the original image is reflected at the boundary. For simple blurs like motion blur in horizontal direction or defocus blur, reflecting boundary conditions are a reasonable choice. For more complex blurs like motion blur in some general direction, reflecting boundary conditions will fail, because in this case the blur outside the boundary has a different direction than the blur inside the boundary. Then one has to put more effort in finding some appropriate boundary conditions. We avoid this problem here by assuming

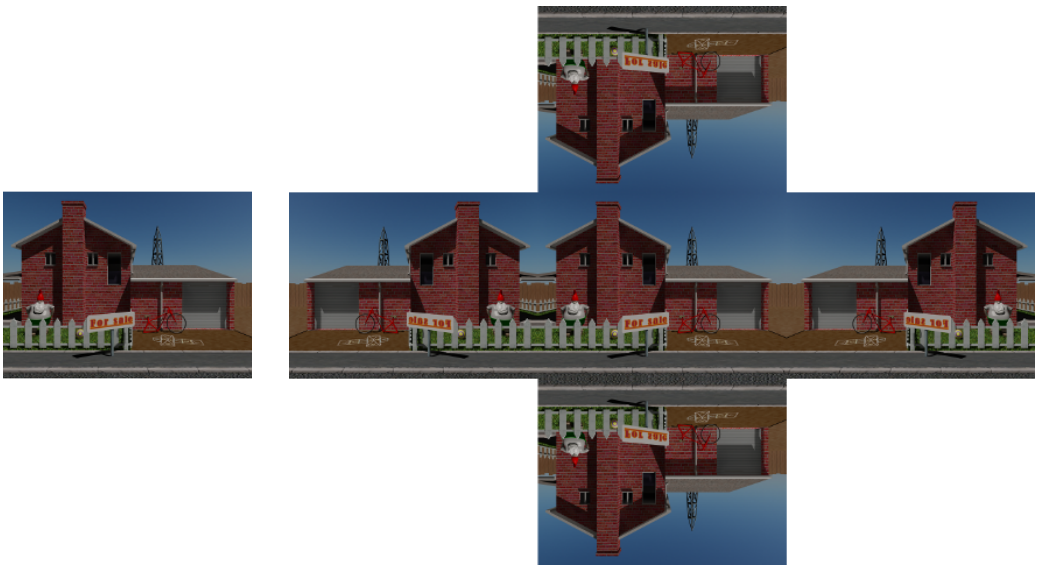


Figure 6.1: Reflecting boundary conditions. **Left:** Image without any boundary conditions. **Right:** Image extended by reflecting boundary conditions.

motion blur in x-direction and applying reflecting boundary conditions. See [22] for some more detailed discussion of boundary treatment for deblurring.

6.2 Value of the PSF

For the whole implementation we will assume that the PSF h is a PSF for motion blur in x-direction. Therefore it is one-dimensional and has the shape of a box function. The exact definition of h is given in equation (3.10).

Remember, the value of h is chosen such that the integral of h over the space is one. If we now want to determine the value of h for a certain pixel at position x we have to be careful, this value will not always be just $h(x)$. The reason for this is that each pixel has a certain width, in our case each pixel has the width h_1 in x-direction and we normally set h_1 to be 1. So a pixel at position x will range from $x - \frac{h_1}{2}$ to $x + \frac{h_1}{2}$. To determine the value of h for a pixel at x we have to determine how much the pixel is covered by h, so the right value will be the convolution of h with a box function of width h_1, evaluated at position x. A pixel that is right at the border of h will have a smaller value than a pixel in the middle of h has. With these observations the right value for a pixel will be computed as follows (assuming that $h_1 = 1$):

$$h(x) = \begin{cases} \frac{1}{2r} & \text{if } -r + \frac{1}{2} \leq x \leq r - \frac{1}{2} \\ 0 & \text{if } -r - \frac{1}{2} > x \vee x > r + \frac{1}{2} \\ \frac{1}{2r}\left(|r| + \frac{1}{2} - |x|\right) & \text{otherwise} \end{cases}$$

In this setting, $|r| + \frac{1}{2} - |x|$ is exactly the value between 0 and 1 that measures how much the pixel at x with width 1 is covered by h.

6.3 Residual

The residual is given in equation (5.8) and we can reformulate it in the following way:

$$\hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2 \end{pmatrix} = f\begin{pmatrix} y \\ x_2 \end{pmatrix} - \int_{\mathbb{R}} u\begin{pmatrix} y' \\ x_2 \end{pmatrix}\ s\begin{pmatrix} y' \\ x_2 \end{pmatrix}\ h\left((y-y')\, s\begin{pmatrix} y' \\ x_2 \end{pmatrix}\right)\ dy'$$

$$= f(\mathbf{x}) - \int_{\mathbb{R}} u\begin{pmatrix} y-p \\ x_2 \end{pmatrix}\ s\begin{pmatrix} y-p \\ x_2 \end{pmatrix}\ h\left(p\ s\begin{pmatrix} y-p \\ x_2 \end{pmatrix}\right)\ dp \qquad \text{by substituting } p := y - y'$$

It is used in various forms in our equations. Therefore it is very important to discretise the residual properly. We substituted $y - y'$ by p to let the integral range over the domain of h. By doing this we safe computational time in the forthcoming implementation, because the domain of h is assumed to range from $-sup$ to sup, which is normally smaller than the domain of u and s. Direct discretisation gives us:

$$\hat{R}_{f,h}[u,s]_{y,x_2} := f_{y,x_2} - \sum_{p=-sup}^{sup} u_{y-p,x_2}\ s_{y-p,x_2}\ h\,(p\ s_{y-p,x_2})$$

Note that the argument of h is not restricted to discrete values and the value of h thus needs to be determined by interpolation. In the current form the arguments of u and s could be out of their domain. Thus we include reflecting boundary conditions by defining

$$i := \begin{cases} |y-p|+1 & \text{if } y-p<1 \\ 2n+1-(y-p) & \text{if } y-p>n \\ y-p & \text{otherwise} \end{cases}$$

and including this definition into the equation for the residual

$$\hat{R}_{f,h}[u,s]_{y,x_2} := f_{y,x_2} - \sum_{p=-sup}^{sup} u_{i,x_2}\ s_{i,x_2}\ h\,(p\ s_{i,x_2}) \tag{6.1}$$

6.4 Smoothness Terms

In this section we want to concentrate on the discretisation of terms resulting from the Euler-Lagrange equations of the smoothness terms (5.12) and (5.13). Because of the symmetry of the two equations above we will only derive the discretisation with respect to u. The scheme for the second equation is then totally analogous. The equation

$$-\frac{\delta}{\delta u}E_2[u](\mathbf{x}) = \operatorname{div}\left(\Psi_1'(|\nabla u(\mathbf{x})|^2)\nabla u(\mathbf{x})\right)$$
$$= \partial_x\left(\Psi_1'\left(|\nabla u(\mathbf{x})|^2\right)\partial_x u\right) + \partial_y\left(\Psi_1'\left(|\nabla u(\mathbf{x})|^2\right)\partial_y u\right)$$

would be some nonlinear isotropic diffusion equation, if one replaced $\frac{\delta}{\delta u}E_2[u]$ by $\frac{\partial}{\partial t}u$. Therefore we use the standard spatial discretisation from [19] for some inner pixel (x_1, x_2):

$$
\begin{aligned}
&-\frac{\delta}{\delta u}E_2[u]_{x_1,x_2} = \\
&\frac{1}{h_1}\left(\frac{\Psi'_{x_1+1,x_2}+\Psi'_{x_1,x_2}}{2}\frac{u_{x_1+1,x_2}-u_{x_1,x_2}}{h_1} - \frac{\Psi'_{x_1x_2}+\Psi'_{x_1-1,x_2}}{2}\frac{u_{x_1,x_2}-u_{x_1-1,x_2}}{h_1}\right) \\
&+\frac{1}{h_2}\left(\frac{\Psi'_{x_1,x_2+1}+\Psi'_{x_1,x_2}}{2}\frac{u_{x_1,x_2+1}-u_{x_1,x_2}}{h_2} - \frac{\Psi'_{x_1,x_2}+\Psi'_{x_1,x_2-1}}{2}\frac{u_{x_1,x_2}-u_{x_1,x_2-1}}{h_2}\right)
\end{aligned}
$$

This discretisation is even valid for boundary pixels, if reflecting boundary conditions are realised by reflected dummy pixels. Ψ'_{x_1,x_2} approximates the diffusivity $\Psi'_1(|\nabla u|^2)$ in the pixel (x_1, x_2). The derivation of the formula uses first order central differences, that have been introduced in Section 2.5.

6.5 Gradient by u

Finally we want to find some algorithmic scheme for the complete gradient descent as given in equation (5.20):

$$\frac{\partial}{\partial t}u = -\frac{\delta}{\delta u}\hat{E}[u,s]$$

We will follow the simplest way and discretise this by an explicit scheme. This means that the time derivative will be discretised by a forward difference and the whole gradient is assumed to be in the old timestep. The timestep size is τ_u.

$$\frac{u^{k+1}-u^k}{\tau_u} \;=\; -\frac{\delta}{\delta u}\hat{E}[u^k,s] \;=\; -\frac{\delta}{\delta u}\hat{E}_1[u^k,s] - \frac{\delta}{\delta u}E_2[u^k]$$

Data term: The missing brick to get a complete discretisation is still the part of the gradient, that corresponds to the data term and is given by equation (5.18). With the discretisation of the residual $\hat{R}$ this equation can be discretised directly to:

$$-\frac{\delta}{\delta u}\hat{E}_1[u,s]_{x_1,x_2} = \sum_{y=1}^{n} \Phi'\left(\left(\hat{R}_{f,h}[u,s]_{y,x_2}\right)^2\right)\; \hat{R}_{f,h}[u,s]_{y,x_2}\; s_{x_1,x_2}\; h\left((y-x_1)s_{x_1,x_2}\right)$$

Complete Formula: By plugging all parts of the gradient descent together, we get:

$$\begin{aligned}
\frac{u^{k+1}_{x_1,x_2} - u^k_{x_1,x_2}}{\tau_u} &= -\frac{\delta}{\delta u}\hat{E}_1[u^k_{x_1,x_2}, s](\mathbf{x}) - \frac{\delta}{\delta u}E_2[u^k_{x_1,x_2}](\mathbf{x}) \\
&= \sum_{y=1}^{n} \Phi'\left(\left(\hat{R}_{f,h}[u^k, s]_{y,x_2}\right)^2\right)\ \hat{R}_{f,h}[u^k, s]_{y,x_2}\ s_{x_1,x_2}\ h\left((y - x_1)s_{x_1,x_2}\right) \\
&\quad + \frac{\alpha}{2h_1^2}\Big(\left(\Psi'_{x_1+1,x_2} + \Psi'_{x_1,x_2}\right)\left(u^k_{x_1+1,x_2} - u^k_{x_1,x_2}\right) \\
&\quad - \left(\Psi'_{x_1,x_2} + \Psi'_{x_1-1,x_2}\right)\left(u^k_{x_1,x_2} - u^k_{x_1-1,x_2}\right)\Big) \\
&\quad + \frac{\alpha}{2h_2^2}\Big(\left(\Psi'_{x_1,x_2+1} + \Psi'_{x_1,x_2}\right)\left(u^k_{x_1,x_2+1} - u^k_{x_1,x_2}\right) \\
&\quad - \left(\Psi'_{x_1,x_2} + \Psi'_{x_1,x_2-1}\right)\left(u^k_{x_1,x_2} - u^k_{x_1,x_2-1}\right)\Big)
\end{aligned}$$

So we end up with the iterative scheme:

$$\begin{aligned}
u^{k+1}_{x_1,x_2} &= u^k_{x_1,x_2} + \tau_u \sum_{y=1}^{n} \Phi'\left(\left(\hat{R}_{f,h}[u^k, s]_{y,x_2}\right)^2\right)\ \hat{R}_{f,h}[u^k, s]_{y,x_2}\ s_{x_1,x_2}\ h\left((y - x_1)s_{x_1,x_2}\right) \\
&\quad + \frac{\tau_u\alpha}{2h_1^2}\Big(\left(\Psi'_{x_1+1,x_2} + \Psi'_{x_1,x_2}\right)\left(u^k_{x_1+1,x_2} - u^k_{x_1,x_2}\right) \\
&\quad - \left(\Psi'_{x_1,x_2} + \Psi'_{x_1-1,x_2}\right)\left(u^k_{x_1,x_2} - u^k_{x_1-1,x_2}\right)\Big) \\
&\quad + \frac{\tau_u\alpha}{2h_2^2}\Big(\left(\Psi'_{x_1,x_2+1} + \Psi'_{x_1,x_2}\right)\left(u^k_{x_1,x_2+1} - u^k_{x_1,x_2}\right) \\
&\quad - \left(\Psi'_{x_1,x_2} + \Psi'_{x_1,x_2-1}\right)\left(u^k_{x_1,x_2} - u^k_{x_1,x_2-1}\right)\Big)
\end{aligned} \tag{6.2}$$

6.6 Gradient by s

By proceeding in the same way as for u we come to the following equation:

$$\begin{aligned}
\frac{s^{k+1} - s^k}{\tau_s} &= -\frac{\delta}{\delta s}\hat{E}[u, s^k] \\
&= -\frac{\delta}{\delta s}\hat{E}_1[u, s^k] - \frac{\delta}{\delta s}E_3[s^k]
\end{aligned}$$

Data Term: According to equation (5.21), in the derivation of the data term by s

$$
\begin{aligned}
&-\frac{\delta}{\delta s}\hat{E}_1[u,s](\mathbf{x}) \\
&= \int_{\mathbb{R}} \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2 \end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2 \end{pmatrix}\ u(\mathbf{x})\ \big(h\left((y-x_1)s(\mathbf{x})\right) \\
&\quad + s(\mathbf{x})\ h'\left((y-x_1)s(\mathbf{x})\right)\ (y-x_1)\big)\ dy \\
&= \int_{\mathbb{R}} \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2 \end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2 \end{pmatrix}\ u(\mathbf{x})\ h\left((y-x_1)s(\mathbf{x})\right)\ dy \\
&+ \int_{\mathbb{R}} \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2 \end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2 \end{pmatrix}\ u(\mathbf{x})\ s(\mathbf{x})\ h'\left((y-x_1)s(\mathbf{x})\right)\ (y-x_1)\ dy
\end{aligned}
\tag{6.3}
$$

a novelty occurs: we have to determine $h'\left((y-x_1)s(\mathbf{x})\right)$. This is quite challenging because of two reasons. First, for motion blur the PSF h is assumed to be a box function and it is not clear how the take the derivative of this box function in the discrete setting. Second, the argument of the derived box function h' is continuous and not discrete.

We came up with three different ways to discretise the data term gradient containing h'. These approaches will be introduced now, one by another.

6.6.1 Discretisation Based on Distributions

Let us first take a closer look at h. For motion blur with radius r, h is defined in equation (3.10). Then the continuous derivative of h reads:

$$
h'(x) = \frac{1}{2r}\left(\delta(x+r) - \delta(x-r)\right) = \begin{cases} \frac{1}{2r}\delta & \text{if } x = -r \\ -\frac{1}{2r}\delta & \text{if } x = r \\ 0 & \text{otherwise} \end{cases} \tag{6.4}
$$

Here, $\delta(x)$ is the delta-function

$$
\delta(x) = \begin{cases} \infty & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases}, \tag{6.5}
$$

which is often used by physicists. The symbol δ without an argument denotes the value of the delta-function at 0, so basically infinity. If one feels uncomfortable in deriving box functions like h and handling delta-functions, one can take a look at the theory of distributions [18].

With this definition of h and h', the second integral of equation (6.3) changes signifi-

cantly. The exact derivation of the first rewriting step is given in the appendix.

$$\int_{\mathbb{R}} \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2 \end{pmatrix}\right)^2\right) \hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2 \end{pmatrix} u(\mathbf{x})\ s(\mathbf{x})\ h'\left((y-x_1)s(\mathbf{x})\right)\ (y-x_1)\ dy$$

see appendix A.3

$$\begin{aligned}
&= \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix} x_1 - \frac{r}{s(\mathbf{x})} \\ x_2 \end{pmatrix}\right)^2\right) \hat{R}_{f,h}[u,s]\begin{pmatrix} x_1 - \frac{r}{s(\mathbf{x})} \\ x_2 \end{pmatrix} u(\mathbf{x})\ s(\mathbf{x})\frac{1}{2r}\left(x_1 - \frac{r}{s(\mathbf{x})} - x_1\right) \\
&- \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix} x_1 + \frac{r}{s(\mathbf{x})} \\ x_2 \end{pmatrix}\right)^2\right) \hat{R}_{f,h}[u,s]\begin{pmatrix} x_1 + \frac{r}{s(\mathbf{x})} \\ x_2 \end{pmatrix} u(\mathbf{x})\ s(\mathbf{x})\frac{1}{2r}\left(x_1 + \frac{r}{s(\mathbf{x})} - x_1\right) \\
&= \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix} x_1 - \frac{r}{s(\mathbf{x})} \\ x_2 \end{pmatrix}\right)^2\right) \hat{R}_{f,h}[u,s]\begin{pmatrix} x_1 - \frac{r}{s(\mathbf{x})} \\ x_2 \end{pmatrix} u(\mathbf{x})\ s(\mathbf{x})\frac{1}{2r}\left(-\frac{r}{s(\mathbf{x})}\right) \\
&- \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix} x_1 + \frac{r}{s(\mathbf{x})} \\ x_2 \end{pmatrix}\right)^2\right) \hat{R}_{f,h}[u,s]\begin{pmatrix} x_1 + \frac{r}{s(\mathbf{x})} \\ x_2 \end{pmatrix} u(\mathbf{x})\ s(\mathbf{x})\frac{1}{2r}\frac{r}{s(\mathbf{x})} \\
&= -\frac{1}{2}u(\mathbf{x})\left(\Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix} x_1 - \frac{r}{s(\mathbf{x})} \\ x_2 \end{pmatrix}\right)^2\right) \hat{R}_{f,h}[u,s]\begin{pmatrix} x_1 - \frac{r}{s(\mathbf{x})} \\ x_2 \end{pmatrix}\right. \\
&\quad \left. + \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix} x_1 + \frac{r}{s(\mathbf{x})} \\ x_2 \end{pmatrix}\right)^2\right) \hat{R}_{f,h}[u,s]\begin{pmatrix} x_1 + \frac{r}{s(\mathbf{x})} \\ x_2 \end{pmatrix}\right)
\end{aligned}$$

In the last expression, the argument of the residual adopts non-integral values. So we have to determine the value of the residual by interpolation. The practical implementation uses linear interpolation to do this.

The first integral from equation (6.3) can be directly discretised in the way we did it for the gradient by u in Section 6.5. The discretised data term gradient reads then:

$$\begin{aligned}
-\frac{\delta}{\delta s}\hat{E}_1[u,s]_{x_1,x_2} = &\left(\sum_{y=1}^{n} \Phi'\left(\left(\hat{R}_{f,h}[u,s]_{y,x_2}\right)^2\right)\ \hat{R}_{f,h}[u,s]_{y,x_2}\ u_{x_1,x_2}\ h\left((y-x_1)\,s_{x_1,x_2}\right)\right) \\
&- \frac{1}{2}u_{x_1,x_2}\left(\Phi'\left(\left(\hat{R}_{f,h}[u,s]_{x_1-\frac{r}{s(\mathbf{x})},x_2}\right)^2\right)\ \hat{R}_{f,h}[u,s]_{x_1-\frac{r}{s(\mathbf{x})},x_2}\right. \\
&\left. + \Phi'\left(\left(\hat{R}_{f,h}[u,s]_{x_1+\frac{r}{s(\mathbf{x})},x_2}\right)^2\right)\ \hat{R}_{f,h}[u,s]_{x_1+\frac{r}{s(\mathbf{x})},x_2}\right) \qquad (6.6)
\end{aligned}$$

6.6.2 Direct Discretisation

The next idea is to neglect the special form of h and to discretise it directly. This means that we interpret h as ordinary one-dimensional image. According to definition (3.10) this image can have negative coordinates, has a grey value of $\frac{1}{2r}$ between $-r$ and r and 0 everywhere else. We now use a first order central difference with a grid size of h_s (see Section 2.5 for a definition) to get the derivative and define this discretisation of

$h'\left((y-x_1)s(\mathbf{x})\right)$ as function p in the following way:

$$p(x_1,x_2,y) := \begin{cases} \frac{1}{2rh_s} & \text{if } (y-x_1)s_{x_1,x_2} + \frac{h_s}{2} > -r \wedge (y-x_1)s_{x_1,x_2} - \frac{h_s}{2} < -r \\ -\frac{1}{2rh_s} & \text{if } (y-x_1)s_{x_1,x_2} + \frac{h_s}{2} > r \wedge (y-x_1)s_{x_1,x_2} - \frac{h_s}{2} < r \\ 0 & \text{otherwise} \end{cases}$$

With this definition we can just exchange h' by p in the formulation and are then able to discretise the data term gradient directly:

$$\begin{aligned} -\frac{\delta}{\delta s}\hat{E}_1[u,s]_{x_1,x_2} = \sum_{y=1}^{n} \Phi'\left(\left(\hat{R}_{f,h}[u,s]_{y,x_2}\right)^2\right)\ \hat{R}_{f,h}[u,s]_{y,x_2}\ u_{x_1,x_2}\left(h\left((y-x_1)s_{x_1,x_2}\right)\right. \\ \left. +\ s_{x_1,x_2}\ p(x_1,x_2,y)\ (y-x_1)\right) \end{aligned}$$

6.6.3 Discretisation Based on the Chain Rule

We try to determine the derivative of h with some workaround. The idea is to use the chain rule in the following way:

$$\frac{\partial}{\partial y}h((y-x_1)s(\mathbf{x})) = s(\mathbf{x})h'\left((y-x_1)s(\mathbf{x})\right)$$

The left-hand side of the above equation is approximated quite easily. It can be done by a central difference expression:

$$\frac{\partial}{\partial y}h((y-x_1)s(\mathbf{x})) = \frac{h\left(\left(y+\frac{h_s}{2}-x_1\right)s(\mathbf{x})\right) - h\left(\left(y-\frac{h_s}{2}-x_1\right)s(\mathbf{x})\right)}{h_s}$$

So we determine the value of the larger expression $s(\mathbf{x})h'\left((y-x_1)s(\mathbf{x})\right)$ instead of the value for $h'\left((y-x_1)s(\mathbf{x})\right)$. With the definition of h (as given in equation (3.10)) we get:

$$\begin{aligned} s(\mathbf{x})h'\left((y-x_1)s(\mathbf{x})\right) &= \frac{\partial}{\partial y}h((y-x_1)s(\mathbf{x})) \\ &= \begin{cases} \frac{1}{2h_sr} & \text{if } \left(y+\frac{h_s}{2}-x_1\right)s(\mathbf{x}) > -r \wedge \left(y-\frac{h_s}{2}-x_1\right)s(\mathbf{x}) < -r \\ -\frac{1}{2h_sr} & \text{if } \left(y+\frac{h_s}{2}-x_1\right)s(\mathbf{x}) > r \wedge \left(y-\frac{h_s}{2}-x_1\right)s(\mathbf{x}) < r \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} \frac{1}{2h_sr} & \text{if } (y-x_1)s(\mathbf{x}) + \frac{s(\mathbf{x})h_s}{2} > -r \wedge (y-x_1)s(\mathbf{x}) - \frac{s(\mathbf{x})h_s}{2} < -r \\ -\frac{1}{2h_sr} & \text{if } (y-x_1)s(\mathbf{x}) + \frac{s(\mathbf{x})h_s}{2} > r \wedge (y-x_1)s(\mathbf{x}) - \frac{s(\mathbf{x})h_s}{2} < r \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

To end up with a discretised term we define

$$g(x_1,x_2,y) := \begin{cases} \frac{1}{2h_sr} & \text{if } (y-x_1)s_{x_1,x_2} + \frac{s_{x_1,x_2}h_s}{2} > -r \wedge (y-x_1)s_{x_1,x_2} - \frac{s_{x_1,x_2}h_s}{2} < -r \\ -\frac{1}{2h_sr} & \text{if } (y-x_1)s_{x_1,x_2} + \frac{s_{x_1,x_2}h_s}{2} > r \wedge (y-x_1)s_{x_1,x_2} - \frac{s_{x_1,x_2}h_s}{2} < r \\ 0 & \text{otherwise} \end{cases}$$

and discretise the data term gradient by:

$$-\frac{\delta}{\delta s}\hat{E}_1[u,s]_{x_1,x_2} = \sum_{y=1}^{n} \Phi'\left(\left(\hat{R}_{f,h}[u,s]_{y,x_2}\right)^2\right)\ \hat{R}_{f,h}[u,s]_{y,x_2}\ u_{x_1,x_2}\left(h\left((y-x_1)s_{x_1,x_2}\right)\right.$$
$$\left. + g(x_1,x_2,y)\ (y-x_1)\right)$$

6.6.4 Complete Formula

Analogous the equation (6.7), we get an iterative scheme for the gradient descent with respect to s:

$$\begin{aligned} s^{k+1}_{x_1,x_2} &= s^k_{x_1,x_2} - \tau_s \frac{\delta}{\delta s}\hat{E}_1[u,s^k]_{x_1,x_2} \\ &\quad + \frac{\tau_s\beta}{2h_1^2}\Big(\left(\Psi'_{x_1+1,x_2}+\Psi'_{x_1,x_2}\right)\left(s^k_{x_1+1,x_2}-s^k_{x_1,x_2}\right) \\ &\quad - \left(\Psi'_{x_1,x_2}+\Psi'_{x_1-1,x_2}\right)\left(s^k_{x_1,x_2}-s^k_{x_1-1,x_2}\right)\Big) \\ &\quad + \frac{\tau_s\beta}{2h_2^2}\Big(\left(\Psi'_{x_1,x_2+1}+\Psi'_{x_1,x_2}\right)\left(s^k_{x_1,x_2+1}-s^k_{x_1,x_2}\right) \\ &\quad - \left(\Psi'_{x_1,x_2}+\Psi'_{x_1,x_2-1}\right)\left(s^k_{x_1,x_2}-s^k_{x_1,x_2-1}\right)\Big) \end{aligned} \tag{6.7}$$

The appropriate data term gradient $\frac{\delta}{\delta s}\hat{E}_1[u,s^k]_{x_1,x_2}$ from Section 6.6.1, 6.6.2 or 6.6.3 has to be plugged into the formula, depending on the chosen discretisation.

Remarks: We have seen three different discretisations and now have to decide which one we want to choose. This will be done by experimental evaluation. Please notice that even though the discretisations look different, they are quite similar. If for the direct and chain rule discretisations h_s is chosen appropriately, such that p or g are non-zero for just two summands, then they are approximately equal to the discretisation based on distributions. The effect of the three discretisations will be shown in Chapter 7.

6.7 Descent Velocities

Two possibly different timestep sizes τ_u and τ_s were introduced for the gradient descents. Actually this is not consistent with our continuous model any more, because in equations (5.20) and (5.21) we derive u and s by the same time t. So the two descents should decrease with the same velocity. But if we have different timestep sizes in the discretisation this is not given anymore, because we can perform one descent faster than the other. The remedy is to insert a new velocity parameter $\eta > 0$ in the continuous model and to reinterpret τ_u and τ_s. If we are working with two different timestep sizes then we are actually not

solving a system including equation (5.21), but a system with

$$\frac{\partial}{\partial t}s = \eta\Bigg(\int_{\mathbb{R}} \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2 \end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s]\begin{pmatrix} y \\ x_2 \end{pmatrix}\ u(\mathbf{x})\ \big(h\left((y-x_1)s(\mathbf{x})\right) \\ + s(\mathbf{x})\ h'\left((y-x_1)s(\mathbf{x})\right)\ (y-x_1)\big)\ dy + \beta\ \mathrm{div}\left(\Psi_2'\left(|\nabla s(\mathbf{x})|^2\right)\nabla s(\mathbf{x})\right)\Bigg) \tag{6.8}$$

as gradient descent equation for s. η is a model parameter that gives the ratio how much faster the gradient descent by s is performed compared to the one by u. If we now discretised the system of equations given by (5.20) and (6.8) with the same timestep size τ, then we would just get same as we had before with the definition:

$$\tau_u := \tau \quad \wedge \quad \tau_s := \tau\eta \tag{6.9}$$

So we actually have instead of two numerical parameters τ_u and τ_s, just one numerical parameter τ in combination with a model parameter η. In the experimental chapter we will stick to τ_u and τ_s anyway, because stability conditions are easier formulated with this definition.

7 Experiments

A new model for image deblurring has been introduced in Chapter 4 and the optimisation of an energy based on this model is discussed in Chapter 5 and 6. This chapter now is about the algorithm and its results. We have implemented the minimisation of the restricted functional given in (5.7). It realises the iterative schemes for the unblurred image u and the depth map s, as given in equation (6.2) and (6.7).

7.1 Implementation

The test implementation that has been developed to compute our algorithm on a real machine is written in the C programming language. Some functions in the code have been generously provided by Joachim Weickert. These functions are about memory management, analysing images and Gaussian smoothing, they are clearly marked in the source code. All other code has been implemented by the author.

Due to the simple explicit scheme, that we have used to implement the gradient descent equations, the code is not optimised for speed at all. So the computation times ranged approximately between 10 minutes and 1 hour to deblur a 320×240 image, depending on the discretisation and the number of iterations.

7.1.1 Penalisation Functions

One degree of freedom in the algorithm is the choice of the robust function Φ and the regularisation functions Ψ_1 and Ψ_2 for the energy. As stated in Section 5.1.2, these functions should be positive, differentiable and increasing.

For the data term function Φ, we tried two different functions:

$$\Phi(s^2) = s^2 \qquad \Phi'(s^2) = 1 \qquad \text{quadratic penalisation}$$

$$\Phi(s^2) = \sqrt{s^2 + \varepsilon^2} \qquad \Phi'(s^2) = \frac{1}{2\sqrt{s^2 + \varepsilon^2}} \qquad \text{robust penalisation}$$

The second choice of Φ is called robust function, because it is more robust against outliers and deviations from the model assumptions. The reason is that it approximates the L^1 penaliser $\Phi(s^2) = |s|$ and hence outliers are less weighted with this function. ε is chosen to be small, in our implementation it is 0.001. We will use the robust penalisation by default and notice the reader if we use quadratic penalisation for the data term.

The penalising functions Ψ_1 and Ψ_2 play obviously a similar role. They result in the diffusivities Ψ_1' and Ψ_2' for the diffusion terms of the Euler-Lagrange equations (5.12) and

(5.13). Three different functions are chosen to test our algorithm:

$$\Psi(s^2) = s^2 \qquad \Psi'(s^2) = 1 \qquad \text{Whittaker-Tikhonov}$$

$$\Psi(s^2) = 2\lambda^2\sqrt{1+\frac{s^2}{\lambda^2}} - 2\lambda^2 \qquad \Psi'(s^2) = \frac{1}{\sqrt{1+\frac{s^2}{\lambda^2}}} \qquad \text{Charbonnier}$$

$$\Psi(s^2) = \lambda^2 \ln\left(1+\frac{s^2}{\lambda^2}\right) \qquad \Psi'(s^2) = \frac{1}{1+\frac{s^2}{\lambda^2}} \qquad \text{Perona-Malik}$$

To make the algorithm more robust under noise, we apply little bit of Gaussian smoothing of $\sigma = 0.5$ to the gradient magnitude, before we apply the diffusivity function. So actually we do not compute $\Psi(|\nabla u|^2)$, but $\Psi(|\nabla u|_\sigma^2)$ in the diffusion terms. This leads to regularised non-linear isotropic diffusion as it is described in [19]. The contrast parameter $\lambda > 0$ distinguishes between forward and backward diffusion and allows edge enhancement for gradients, that are larger than λ. No contrast enhancement is possible for the Whittaker-Tikhonov and Charbonnier regularisers, because these smoothness terms are convex and hence they always perform forward diffusion. The non-convex Perona-Malik regulariser leads to multiple possible solutions, but the data term itself is already so ill-posed (see Section 3.3.2) that this is negligible and so we will mainly use Perona-Malik. If not stated different, Ψ_1 and Ψ_2 are Perona-Malik regularisers.

7.1.2 Parameters

The algorithm has quite a lot of parameters, that can be used to tune the result. While this can have advantages for experienced users, it can be confusing for beginners to choose the right parameters. We will give now the parameters that worked well for the experiments.

The most important parameters are probably α and β, that steer the smoothness of the resulting unblurred image u and depth map s respectively. Values of $\alpha \approx 0.03$ and $\beta \approx 15$ worked well for most test images.

We have chosen a spatial grid size of $h_1 = h_2 = 1$ for the finite differences. For the chain rule discretisation (see Section 6.6.2) of the gradient by s a grid size of $h_s \approx 1$ is chosen and for the direct discretisation (see Section 6.6.3) we took a grid size of $h_s \approx 5$ and is chosen. The timestep sizes τ_u and τ_s are depending on the smoothness weights α and β. In [22] the following necessary stability condition has been derived:

$$\tau_u \leq \frac{2}{1+8\alpha}$$

In practice we mainly just used $\tau_u = 0.5$, because α was chosen quite small. We do not have a stability result for τ_s, but the experimental observations showed, that the bound for the explicit diffusion scheme

$$\tau_s \leq \frac{1}{4\beta}$$

is sharp. Artefacts arose for larger values of τ_s and the algorithm performed well when the condition was fulfilled.

Regarding the diffusion terms, we have to give a value for the contrast parameter λ. A reasonable choice is to take $\lambda \approx 0.03$ for the diffusion by u and $\lambda \approx 10$ for the diffusion by

s. The parameter for s is larger, because the range of the depth map is normally smaller than the one for the image and additionally the depth map has normally large areas with the same depth. This behaviour is emphasized by a larger diffusion weight. As stated before in Section 7.1.1, a Gaussian presmoothing with $\sigma = 0.5$ is applied to the gradients to get regularised diffusion.

In Section 4.4 we explained that our blurring model can only give us a relative depth, that is depending on the initial blurring radius of h. This blurring radius r should be chosen such that it is still larger than the maximal observable blur. If this is the case, then the depth is always larger than one and our algorithm behaves well. But if r is chosen smaller, then the real values of s are smaller than one and it could easily happen that negative values occur in the evolution of s. However, a negative depth makes no sense and the implementation is not adjusted to cope with such values. Actually, we avoid values smaller than 1 explicitly. So r should be chosen large enough, for most test images we just set $r = 30$, but it has always to depend on the maximal blur in the image.

The total number of iterations should be very large. We are interesting in the steady state of the gradient descent equations and therefore have to perform many iterations. In our practical implementation, iteration numbers ranging from 500 up to 5000 were sufficient.

7.1.3 Initialisation

A good initialisation for the unblurred image is quite easy to find: we just take the blurred image. This will be true for all deblurred images in this thesis.

To find a reasonable initialisation for the depth map s is more difficult. The severe ill-posedness of the deblurring problem makes this difficulty even worse, because the initialisation is highly influencing the result of the algorithm. By default the depth map gets initialised with random values between 5 and 10. We will indicate whenever we use another initialisation.

7.1.4 Test Images

Our test set of images has been created under realistic conditions with a ray-tracer. Therefore the original sharp image and a ground truth depth map are available. The test set has been provided by Tobias Becker, the author of the images.

Figure 7.1 shows the test images. The original unblurred image shows a house on a road. There is a fence with a sign in front of the house. A tower is standing in the background and a garden gnome is sitting on the lawn. The whole test set is available as colour image, but our algorithm is designed for greyscale images and so we are operating on the greyscale version of the test set. The test images are blurred under motion blur in horizontal direction, as if they were taken from a car that is driving by the house. There are three versions of the blurred images available, one with a small blur, one with a medium blur and one with a large blur. One may think of a car driving by with three different speeds. We mainly tested our algorithm on the image with the small blur, it will be mentioned if this is not the case. The last image in figure 7.1 has been blurred synthetically in a spatially invariant way by the author of this thesis. It was created to test the capabilities of the program without the presence of spatially variant blur.

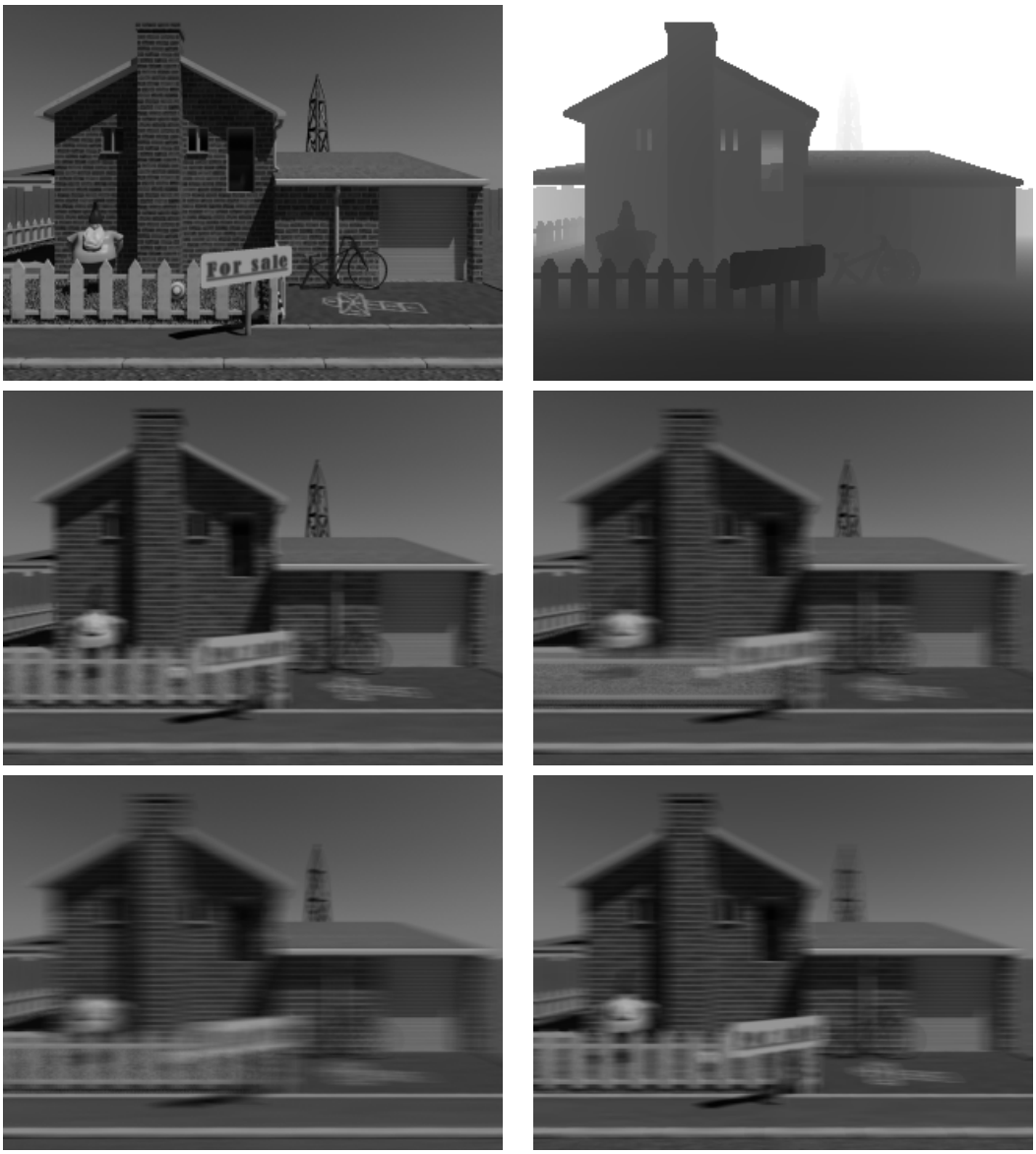


Figure 7.1: Test images. **Upper left:** Ground truth unblurred image. **Upper right:** Ground truth depth map. Dark pixels are near the camera, whereas brighter pixels are farer away. **Central left:** Image blurred under small motion blur. **Central right:** Image blurred under medium motion blur. **Lower left:** Image blurred under large motion blur. *Author: T. Becker* (2008) **Lower right:** Image blurred under spatially invariant motion blur. *Author: S. Lösch* (2009)

7.2 Results

This section is now about the actual images, that our program produces. We tested the gradient descent by u (6.2) and the gradient descent for s (6.7) seperately in Section 7.2.1 and Section 7.2.2 respectively. Of course, we also tested the complete algorithm, that performs the two gradient descents in an alternating way in Section 7.2.3. So the final program really only takes a single blurred image and tries to return the unblurred image, as well as the depth map of the scene. This task is obviously not easy to solve.

The algorithms will be tested on the images shown in figure 7.1. Due to the lack of other test images, that fulfil the model assumptions and provide a ground truth, no other images are considered.

7.2.1 Deblurring

There are many algorithms that perform non-blind spatially invariant deblurring and there are still a lot of algorithms that perform non-blind spatially variant deblurring. We now want to evaluate how good our own program handles these tasks. Therefore, the ground truth depth map is just taken as additional input and only the gradient descent by u, as given in equation (6.2), is computed iteratively.

Please notice that this part of the algorithm is actually equivalent to the one given in [24], with the small difference that the PSF is represented in another way and the boundary treatment works in a slightly different manner.

Spatially Invariant Deblurring

We first test our method on the image that is deviated by spatially invariant blur from the lower right corner of figure 7.1. The amount of blur is known and is an input of the algorithm. Figure 7.2 shows the results and illustrates the advantages of the robust data term.

Figure 7.2: Spatially invariant deblurring. **Left:** Image deblurred with quadratic data term. **Right:** Image deblurred with robust data term.

At the first glance the deblurred images look very convincing, but actually they were quite easy to produce, because the blur is known and everywhere the same. Another

reason for the nice behaviour is that the blur was created synthetically by just averaging pixel values in a neighbourhood. So we cannot have occlusion effects. It is observable that the robust data term is very successful in reducing image errors, like the ringing artefacts around the tower.

Spatially Variant Deblurring

By taking the original depth map as input, we can test the gradient descent by u in the case of spatially variant blur. Two results are given in figure 7.3, where we deblurred the image on the central left side of figure 7.1.

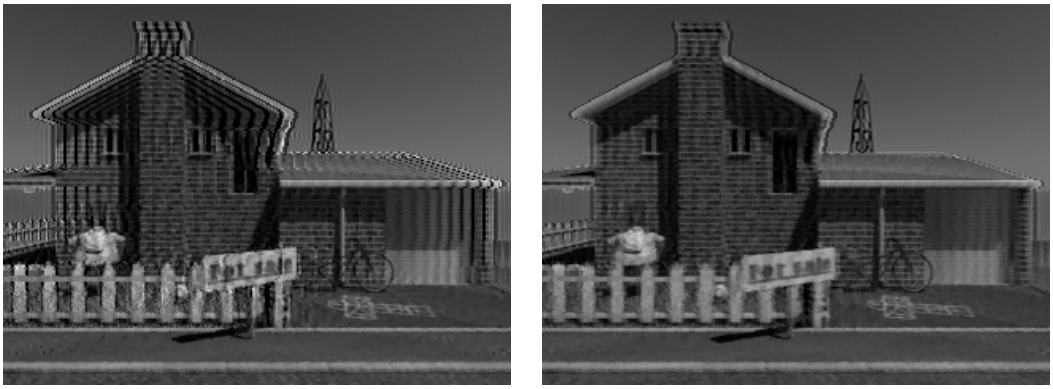


Figure 7.3: Spatially variant deblurring. **Left:** Image deblurred with quadratic data term. **Right:** Image deblurred with robust data term.

Please note that even though we took the exact absolute depth map as input, it still could have errors for our method. The reason is that we do not know the exact focal length f and the exact motion extent m and hence cannot choose r in such a way that it fulfils equation (4.5) properly. We have to rescale the depth map by hand and therefore it could still contain little errors. But this could only give a small contribution to the strong ringing artefacts that are still present in the deblurred image (look at the roof of the house). They are probably due to occlusions that appear around strong edges, e.g. the wall of the house. However one can observe that these artefacts are much better suppressed with the robust data term, because it can better cope with deviations from the model assumptions.

The dominance of our spatially variant model to the spatially invariant one becomes obvious if the assumption of spatial invariance is not fulfilled any more. Figure 7.4 provides an example of such a deblurring with wrong assumptions.

So we took the spatially variant blurred image on the central left side of figure 7.1 and tried to deblur it in a spatially invariant way. Although the garden gnome and the bike in front of the house have been recovered quite well, the tower in the back is deblurred too much and the sign in the front is not deblurred enough. In contrast to that, the sign as well as the tower have been reasonably reconstructed in figure 7.3, where we recovered the image with our spatially variant model.

Figure 7.4: Deblurring under wrong assumptions. The above image is an example of what happens when one tries to recover an image that was blurred under spatially variant blur with spatially invariant deblurring.

7.2.2 Depth Estimation

The main novelty of this thesis is the gradient descent by s. Algorithms very similar to the one described in Section 7.2.1 have already been developed to perform non-blind deblurring, e.g. in [24]. But as far as we know, the way how we model spatial variance (see equations (4.6) and (4.7)), the Euler-Lagrange equation by s (see equations (5.17) and (5.21)) and our resulting gradient descent algorithm for s (see equation (6.7)) is new to the literature. We will now evaluate the performance of this novel approach. Section 6.6 describes three different ways to discretise the gradient descent by s, each of them will be discussed separately. For better visualisation we have scaled the depth maps by a factor of 10.

Distribution Discretisation

At the first glance the discretisation based on distribution from Section 6.6.1 is superior to the other two methods. We don't have to evaluate functions like p or g and the additional parameter of h_s is missing. Unfortunately, practical experiments showed that this discretisation is not working directly. What happens when we run a program based on equation (6.6) is shown on the left side of figure 7.5. The algorithm seems to get out of control and hence it produces useless results. Only after we modified equation (6.6) to

$$\begin{aligned}
-\frac{\delta}{\delta s}\hat{E}_1[u,s]_{x_1,x_2} = \Bigg(& \sum_{y=1}^{n} \Phi'\left(\left(\hat{R}_{f,h}[u,s]_{y,x_2}\right)^2\right)\ \hat{R}_{f,h}[u,s]_{y,x_2}\ u_{x_1,x_2}\ h\left((y-x_1)\, s_{x_1,x_2}\right)\Bigg) \\
& - \frac{1}{15} u_{x_1,x_2}\Bigg(\Phi'\left(\left(\hat{R}_{f,h}[u,s]_{x_1-\frac{r}{s(\mathbf{x})},x_2}\right)^2\right)\ \hat{R}_{f,h}[u,s]_{x_1-\frac{r}{s(\mathbf{x})},x_2} \\
& + \Phi'\left(\left(\hat{R}_{f,h}[u,s]_{x_1+\frac{r}{s(\mathbf{x})},x_2}\right)^2\right)\ \hat{R}_{f,h}[u,s]_{x_1+\frac{r}{s(\mathbf{x})},x_2}\Bigg)
\end{aligned} \tag{7.1}$$

we could get reasonable depth maps. What we did is to decrease the absolute weight of the h' term by dividing this term not only by 2, but by the larger value 15.

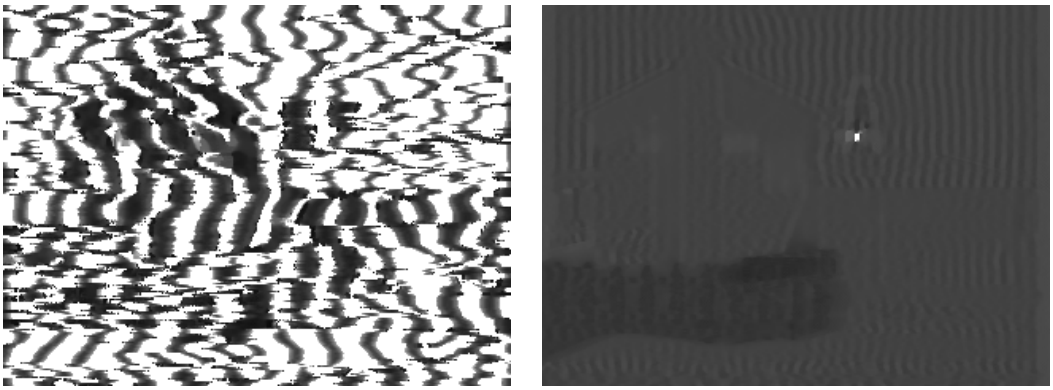


Figure 7.5: Depth estimation with discretisation based on distributions. **Left:** Original discretisation. **Right:** Modified discretisation.

We do not have a proper explanation why the modification in equation 7.1 is reasonable and why the original discretisation is not working. The idea of reducing the weight of the h' term came from the observation that both other discretisation give a much smaller contribution and produce good depth maps. Further research will be necessary to explain this phenomenon.

Direct Discretisation

A depth map resulting from the direct discretisation is given in Figure 7.6. Please note that we could achieve the best results with a grid size of $h_s \approx 5$, which is larger than we had expected.

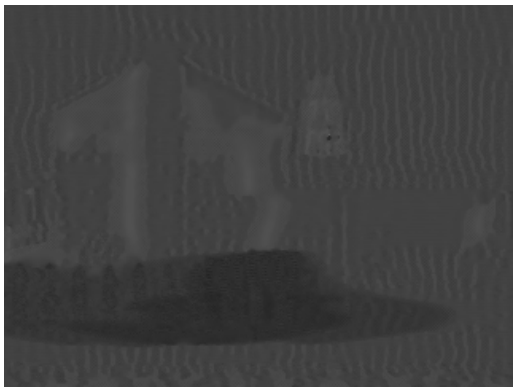


Figure 7.6: Depth estimation with direct discretisation.

Chain Rule Discretisation

Also our third discretisation, the discretisation based on the chain rule from Section 6.6.3, gives reasonable results. On the right upper side of figure 7.7 an example of the capabilities

of this discretisation is shown. We used the same setting as for figures 7.5 and 7.6 for this image.

Even though the images resulting from all three discretisations look very similar, it seems like the chain rule discretisation is slightly better. The tower in the back is better detected and one is able to distinguish the garden gnome and the house. Also the ringing artefacts, that are clearly visible at the horizon for the other two discretisations, are better suppressed. Experiments with other parameters and a close look at the image evolutions strengthen this conjecture.

Effect of the Initialisation

Due to the severe ill-posedness of the deblurring task and the fact that our energy functional given in (5.1) is far from being convex, the result of the algorithm is quite sensitive to the initialisation.

Figure 7.7 gives four examples of initialisations (left side) and their resulting depth map (right side). We only used the discretisation based on the chain rule, because it was the superior discretisation before. The first row shows an initialisation with random values between 5 and 10, whereas in the second row depth map is initialised with the constant value 7.5. The last two rows are again initialisation with random values, where the third row ranges from 5 to 20 and the fourth row ranges from 5 to 40.

Larger Blurs

We can apply our depth estimation algorithm to larger blurs as well. Figure 7.8 shows some results, where we have deblurred the image with the medium sized blur from the right side of the central row in figure 7.1, as well as the image deviated by a large blur from the lower left side in the same figure. Although the depth maps are not very accurate, it is observable that our method still performs reasonably, in particular for the medium sized blur. The depth map of the large blur is brighter than the one for the medium blur, because we initialised the initial blur with $r = 100$ instead of $r = 30$.

General Observations

We saw that the gradient descent by s is able to give reasonable results. It clearly detects that the fence and the sign from the original image are closer to the camera and that the tower is far away. It is not surprising, that the horizon is not detected well, because it is just one homogeneous area, that is not blurred at all. Therefore it also makes no sense to compute some error measure between the ground truth depth and the estimated depth, because the horizon will dominate the result. We rely on our visual perception to rank the results.

The depth estimation method is also successful when there is actually nothing to detect. The blurry input image that formed the basis for figure 7.9 is the spatially invariant blurred image from the lower right side of figure 7.1, so the ground truth depth map is just a constant signal. Our method performs also well in this situation, because no clear structure is visible in the estimated depth map and everything except the sky is approximately at the same depth.

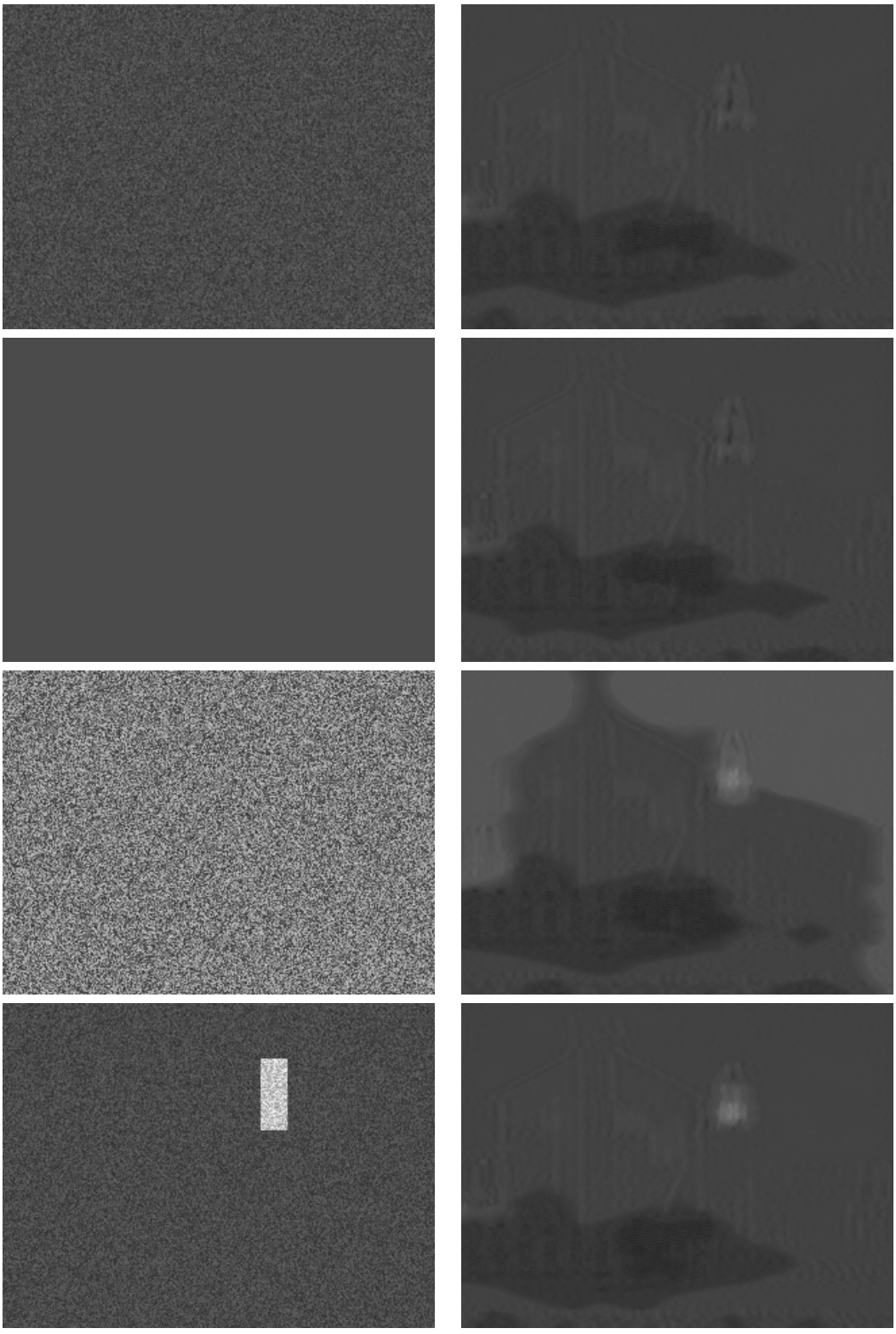


Figure 7.7: Effect of the initialisation. See Section 7.2.2 for more details.

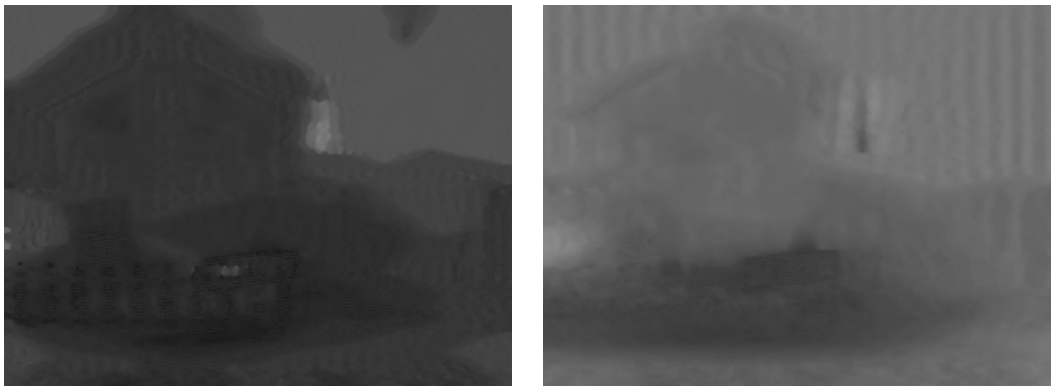


Figure 7.8: Depth estimation with larger blurs. **Left:** Estimated depth for a medium sized blur. **Right:** Estimated depth for a large blur.

However, the results for the depth map are often far from being perfect. The depth map is quite inaccurate and not well localised. Strong ringing artefacts are present, especially for the direct and distribution discretisation. The results of the algorithm are dependent on the initialisation. We reckon that the data term does not yet give enough information to get a more accurate depth map. Improvements in this area are desirable.

Figure 7.9: Depth estimation under spatially invariant blur.

7.2.3 Combined Deblurring and Depth Estimation

The heart of this thesis is the combined estimation of the unblurred image u and the depth map s. To tackle this task we just perform the gradient descents by u and the one by s in an alternating manner, such that both tasks can benefit from each other. Due to the superior results of the discretisation based on the chain rule in Section 7.2.2, we will only use this discretisation.

We observe that the smoothness parameter for the unblurred image α has a big influence on the deblurring result. Figure 7.10 illustrates this by giving the output of our algorithm for two different values of α.

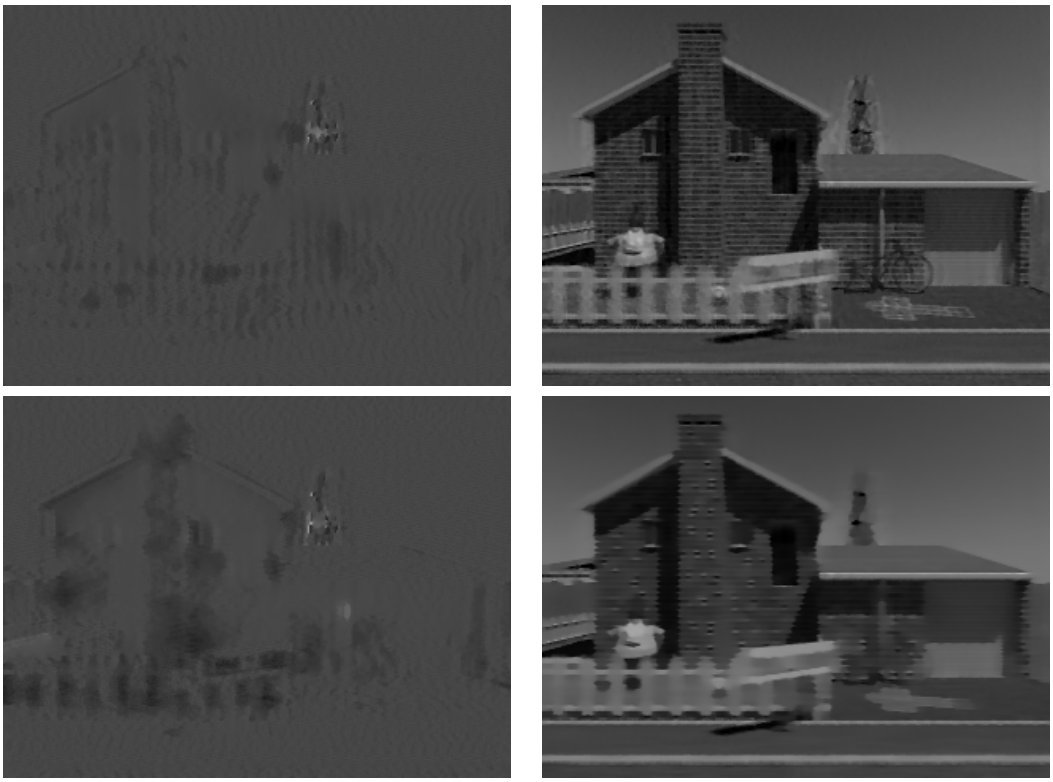


Figure 7.10: Combined deblurring with constant α. **Left side:** Estimated depth maps. **Right side:** Deblurred images. **Upper row:** Result for $\alpha = 0.01$. **Lower row:** Result for $\alpha = 0.1$.

A large value of α clearly gives a better depth map, but it is bought by an overdiffusion of the unblurred image, though. Obviously the image on the lower right side of figure 7.10 is lacking small scale details, but the estimated depth map is more accurate than the one computed with a small alpha. On the other hand, the image unblurred with small alpha has many details (see for example the bike), but the deblurring is not convincing because it suffers under the inaccurate depth map.

We try to avoid this trade-off by applying the following strategy: we start with a large value of α and then decrease it step by step. This can be regarded as applying the algorithm with a large α and taking both results of it, the deblurred image and the depth map, as new input for a new algorithm that is applied with a smaller α. This is then iterated several times. The proposed method is basically the same as the continuation strategy described in [24], but there it was only applied to the gradient descent by u.

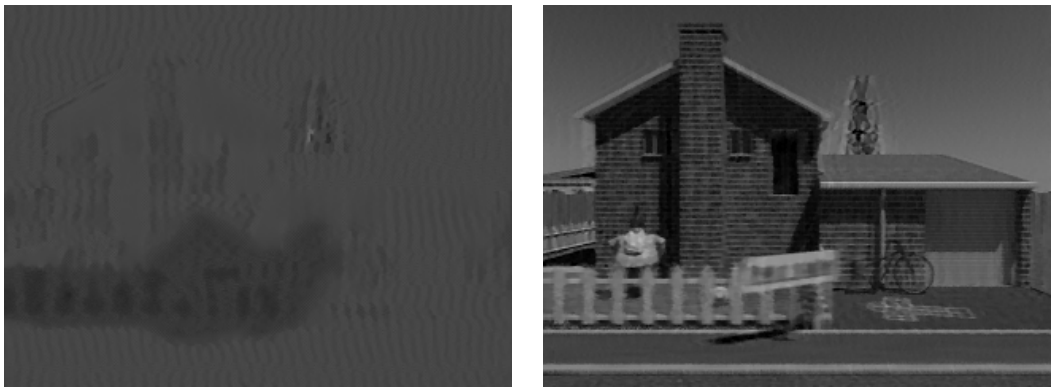


Figure 7.11: Combined deblurring with decreasing α. **Left:** Estimated depth map. **Right:** Deblurred image.

To compute the images in figure 7.11 we started with $\alpha = 0.8$ and $\tau_u = 0.0625$ and divided α every 100 iterations by 2. τ_u got doubled every 100 iterations till it reached 0.5, which was at iteration 300. By this doubling of τ_u with fixed τ_s we are also divide the modelling variable η from Section 6.7 by two. The decreasing of α was stopped after iteration 700 when it had a value of 0.00625. The total number of iterations was chosen to be 1500. The depth map has been initialised with random values between 5 and 10. The strategy of decreasing α step by step gives clearly better results than having a constant α. Both, the unblurred image and the depth map are estimated in a more accurate way.

However, the tower in the background is still not deblurred in a satisfying way. It is clearly overdeblurred, because the gradient descent by s was not sensitive enough to detect that it is quite far away. In this case we propose to help our algorithm with a little supervision. Figure 7.7 shows that the initialisation of the depth map has a significant impact on the result. Therefore we just change the initialisation in such a way, that the area around the tower is brighter than the rest of the initialisation and hence the modified initialisation is closer to the ground truth depth map. So we take the image on the lower left side of figure 7.7 as initialisation for the depth map, where the area around the tower contains random values between 15 and 25. The rest of the initialisation still consists of random values between 5 and 10. The result is shown in figure 7.12.

Now both, the tower and the fence, have been deblurred quite accurately. This would

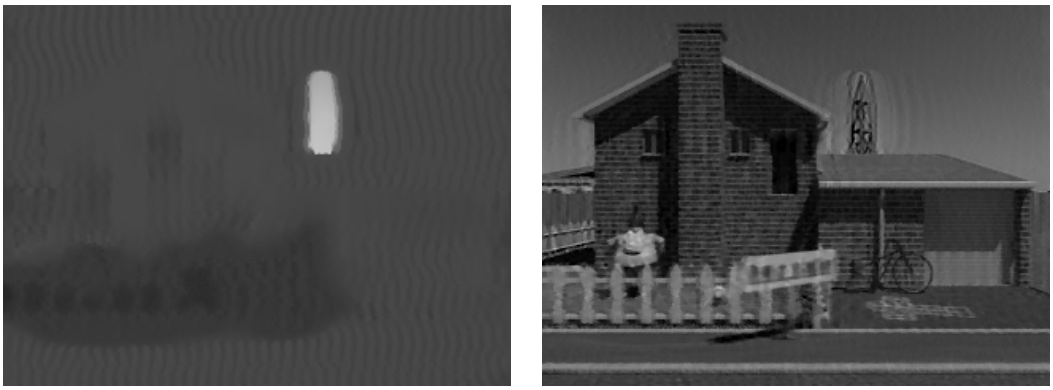


Figure 7.12: Combined deblurring with decreasing α and modified initialisation. **Left:** Estimated depth map. **Right:** Deblurred image.

have never been possible with spatially invariant deblurring. There are still some artefacts in the image, like the area around the tower, or the sign that is still a bit blurred, but they are quite minor if one compares this to other deblurred images. Even though ringing artefacts are still present in the obtained depth map, they are almost entirely suppressed in the deblurred image.

7.3 Evaluation

We already argued that a quantitative evaluation by means of a global error measure between the ground truth depth map and the estimated one is pointless, because the results will be dominated by the large areas in the sky, where no depth can be recovered. So we decided to compare some results by only considering a one-dimensional cut through an interesting image region. This cut was done horizontally at the pixels with y-coordinate 169, which is right at the tips of the fence poles. The cut is shown in figure 7.13.

Figure 7.13: Cut for the evaluation. The cut itself is illustrated in white colour.

In figure 7.14 we compare the ground truth with the pure depth estimation algorithm, that only performs the gradient descent by s, as well as with the combined approach,

where the depth map and the deblurred image are computed. We chose as reference depth map the image in the third row of figure 7.7 for the descent by s, and the image on the left side of figure 7.11 for the combined method.

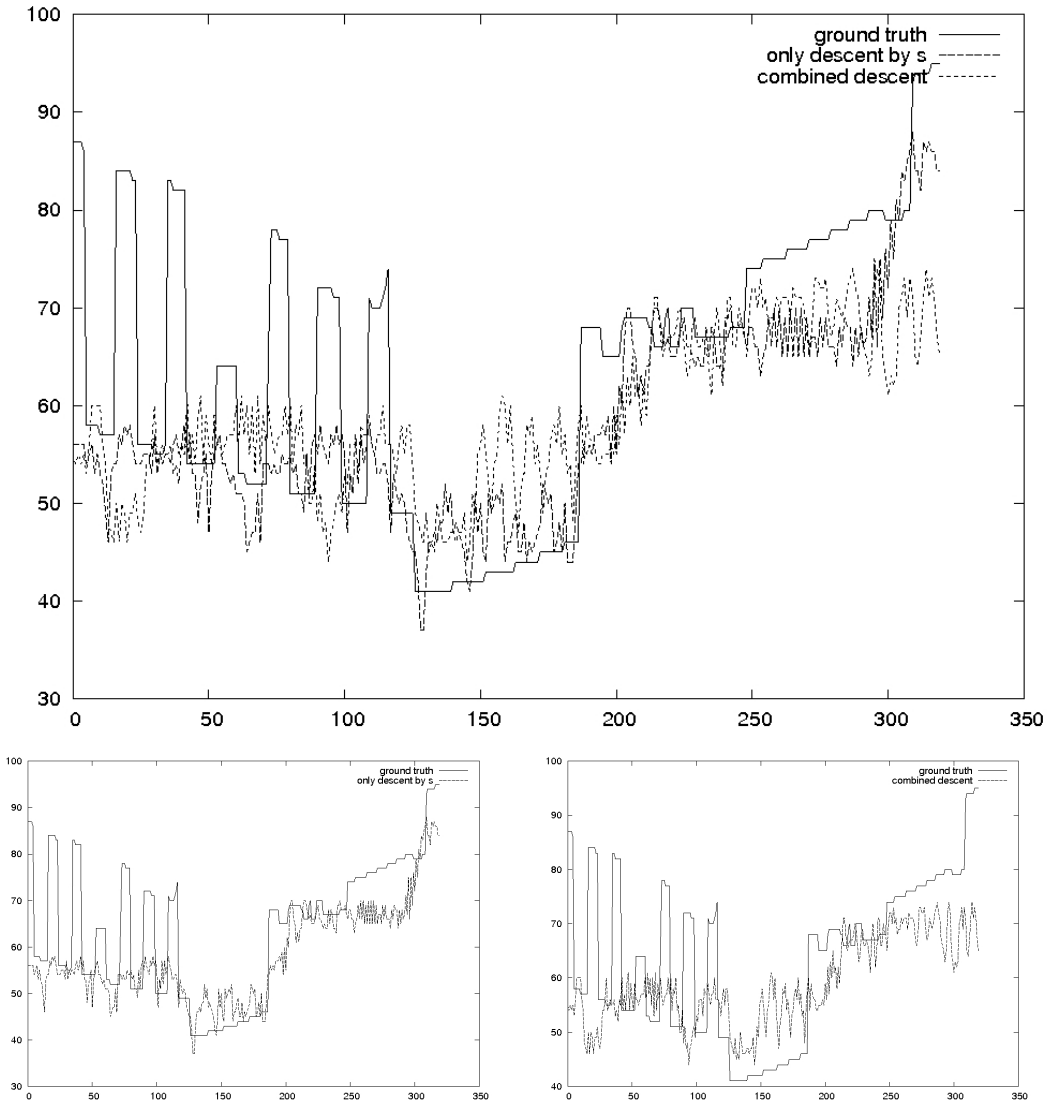


Figure 7.14: Experimental evaluation. **Top:** Ground truth compared to gradient descent by s and combined descent. **Lower left side:** Ground truth compared to gradient descent by s. **Lower right side:** Ground truth compared to combined descent.

8 Conclusions

A novel method for image deblurring and depth estimation from a single blurred image has been introduced. We will review the achievements, discuss the results and give a outlook on further research in this chapter.

8.1 Achievements

Our algorithm is based on a blurring model that includes the depth of the scene explicitly. The PSF is allowed to vary with respect to its spatial extent, but not with respect to its shape. Therefore our method can be regarded as semi-blind deblurring. The relation to depth estimation has been justified in case of camera motion in the pinhole model, but can be extended to other settings.

An energy functional based on the novel model has been developed, containing the unblurred image and the depth map as unknown variables. It contains regularisation terms for both unknowns to cope with the severe ill-posedness of the deblurring task. We try to find the unblurred image and the depth map by minimising the energy by means of gradient descent. Therefore we derived the Euler-Lagrange equations for the energy. We got a system of two partial differential equations that evolve the unblurred image and the depth map respectively. The equation for the unblurred image is in principle already known to the literature, for example in [24], but the gradient descent for the depth map is new.

Discretising both gradient descent equations gave us an algorithm implementable on a computer. For the discretisation and the actual implementation we concentrated on the reduced model, that is designed for motion blur in horizontal direction. We came up with three different ways to discretise the novel gradient descent for the depth map. Standard discretisations have been used for the other gradient descent and the diffusion terms in both descents.

The actual results of our method have been produced with a C implementation. Both gradients were tested separately as well as the combined approach, where the gradients were performed in an alternating manner. We have shown that it is actually possible to get reasonable estimates for the unblurred image and the depth map of the scene, by only incorporating information from a single blurred image.

8.2 Discussion

Even though our combined approach gives good results in some cases, the depth maps and deblurred images it produces are not always satisfying. We think the reason is that the gradient descent for the depth map is still too inaccurate, due to a lack of information. If the depth map is unreliable, the deblurring cannot be accurate, because it is depending

directly on the depth map. Our method is then close to spatially invariant deblurring or performs even worse.

But we have shown that our novel method actually does work and have therefore given a proof of concept. Combined deblurring and depth estimation from a single blurred image is possible. This is not obvious, because we are basically trying to restore a 3-dimensional sharp surface only by incorporating a 2-dimensional blurred image.

Furthermore, if the depth map is well approximated, then convincing deblurred images can be obtained. This shows that generally our new blurring model is correct and useful. In appropriate situations, the spatial extent of blur can be directly related to depth and the relation is inversely proportional. The setting of constant shape but varying spatial extent of blur is definitely worth further research, because it occurs in many practical situations like defocus or general motion blur. The spatially variant deblurring task is considerably simplified by assuming shape constance and therefore becomes solvable. Our model can be also interpreted as a parametrisation of the blur, where we have to estimate one parameter per pixel, namely the depth.

8.3 Ongoing Work

The main future goal should be the improvement of the gradient descent by the depth map. A success in this task will most probably enhance the combined method as well. Maybe already the extension of the model to colour images will bring some advantage.

Another idea is to introduce some kind of gradient constancy assumption. In the data term of our energy formulation we assume that the grey value of the observed image and the one of the image modelled by the blurring process is the same. Additionally we could assume that the gradient is the same and model this in the data term. Such a gradient constancy assumption gives good results in optical flow algorithms and [17] used it successfully for spatially invariant image deblurring.

The deblurring result of the combined gradient descent could also be improved by using a better initialisation for the unblurred image. A promising approach is the one from [13], where a shock filtered version of the blurred images is used as initialisation.

To reduce the ringing artefacts in the depth map we could include a local smoothness term, similar to the one already applied in [17]. The idea is to enforce some stronger smoothness assumption for the depth map in regions where the observed image is smooth. So we assume that the depth is not varying when the greyvalues of the observed image are not varying. This could be in particular helpful for reducing ringing artefacts in the horizon area of our test images.

Our algorithm was not optimised for speed at all. It is implemented by a simple explicit scheme for the gradient descents. Therefore the computation times are quite large, even for small images. More advanced schemes, that incorporate for example the solving of linear systems, would lead to faster computation.

Last but not least we come back to our general model, in contrast to the model designed for one-dimensional motion blur. It is desirable to find a discretisation of this model and its continuous gradient descents, such that we could evaluate how good our method performs for general blurring in horizontal and vertical direction. One useful application would be deblurring under defocus, but there are many more.

A Appendix

A.1 Normalisation of H

We want to show that the spatially variant PSF $\tilde{H}$, defined by equation (4.2), fulfils the normalisation property (3.4), if the invariant PSF h is normalised. So we can assume that equation (3.5) is true. Under these assumption we can proof:

$$\begin{aligned}
&\int_{\mathbb{R}^2} \tilde{H}(\mathbf{x}, \mathbf{y})\ d\mathbf{x} \\
&= \int_{\mathbb{R}^2} s(\mathbf{y}) h((\mathbf{x}-\mathbf{y})s(\mathbf{y}))\ d\mathbf{x} && \text{by equation (4.2)} \\
&= \int_{\mathbb{R}^2} s(\mathbf{y}) h(\mathbf{x}')\ \frac{d\mathbf{x}'}{s(\mathbf{y})} && \text{by substitution} \\
&= \int_{\mathbb{R}^2} h(\mathbf{x}')\ d\mathbf{x}' \\
&= 1 && \text{by equation (3.5)}
\end{aligned}$$

In the third line, we have substituted the argument of h by $\mathbf{x}'$. We used integration by substitution and substituted

$$\mathbf{x}' = (\mathbf{x}-\mathbf{y})s(\mathbf{y}),$$

where the following equation

$$\frac{d\mathbf{x}'}{d\mathbf{x}} = s(\mathbf{y}) \ \Leftrightarrow\ d\mathbf{x} = \frac{d\mathbf{x}'}{s(\mathbf{y})}$$

arises. This concludes the proof. □

A.2 Derivation of the Euler-Lagrange Equations

A.2.1 General Derivative with Respect to u

The aim is to determine $\frac{\delta}{\delta u} E_1$. Therefore we just derive the equation for the spatially variant model and then plug the definition of $\tilde{H}$ (4.2) in. Additionally we define $w := u + \varepsilon v$, where v is an unknown perturbation function and u is a minimiser. So we know that functional has a minimum for $\varepsilon = 0$. Using this we can show:

$$0 = \frac{d}{d\varepsilon} \left(E_1[u+\varepsilon v, s]\right)|_{\varepsilon=0}$$

$$= \frac{d}{d\varepsilon}\left(\frac{1}{2}\int_{\mathbb{R}^2}\Phi\left(\left(R_{f,h}[u+\varepsilon v,s](\mathbf{x})\right)^2\right)\ d\mathbf{x}\right)|_{\varepsilon=0}$$

$$= \frac{d}{d\varepsilon}\left(\frac{1}{2}\int_{\mathbb{R}^2}\Phi\left(\left(R_{f,h}[w,s](\mathbf{x})\right)^2\right)\ d\mathbf{x}\right)|_{\varepsilon=0}$$

$$= \frac{d}{d\varepsilon}\left(\frac{1}{2}\int_{\mathbb{R}^2}\Phi\left(\left(f(\mathbf{x})-\int_{\mathbb{R}^2} w(\mathbf{y})\ \tilde{H}(\mathbf{x},\mathbf{y})\ d\mathbf{y}\right)^2\right)\ d\mathbf{x}\right)|_{\varepsilon=0}$$

$$= \left(\frac{1}{2}\int_{\mathbb{R}^2}\frac{d}{d\varepsilon}\Phi\left(\left(f(\mathbf{x})-\int_{\mathbb{R}^2} w(\mathbf{y})\ \tilde{H}(\mathbf{x},\mathbf{y})\ d\mathbf{y}\right)^2\right)\ d\mathbf{x}\right)|_{\varepsilon=0}$$

$$= \left(-\int_{\mathbb{R}^2}\Phi'\left(\left(R_{f,h}[w,s](\mathbf{x})\right)^2\right)\ R_{f,h}[w,s](\mathbf{x})\left(\int_{\mathbb{R}^2} v(\mathbf{y})\ \tilde{H}(\mathbf{x},\mathbf{y})\ d\mathbf{y}\right)\ d\mathbf{x}\right)|_{\varepsilon=0}$$

$$= -\int_{\mathbb{R}^2}\Phi'\left(\left(R_{f,h}[w,s](\mathbf{x})\right)^2\right)\ R_{f,h}[u,s](\mathbf{x})\left(\int_{\mathbb{R}^2} v(\mathbf{y})\ \tilde{H}(\mathbf{x},\mathbf{y})\ d\mathbf{y}\right)\ d\mathbf{x}$$

$$= -\int_{\mathbb{R}^2}\int_{\mathbb{R}^2}\Phi'\left(\left(R_{f,h}[w,s](\mathbf{x})\right)^2\right)\ R_{f,h}[u,s](\mathbf{x})\ \tilde{H}(\mathbf{x},\mathbf{y})\ d\mathbf{x}\ v(\mathbf{y})\ d\mathbf{y}$$

Because this has to hold for every v

$$\Rightarrow 0 = -\int_{\mathbb{R}^2}\Phi'\left(\left(R_{f,h}[w,s](\mathbf{x})\right)^2\right)\ R_{f,h}[u,s](\mathbf{x})\ \tilde{H}(\mathbf{x},\mathbf{y})\ d\mathbf{x}$$

Variable substitution

$$= -\int_{\mathbb{R}^2}\Phi'\left(\left(R_{f,h}[w,s](\mathbf{y})\right)^2\right)\ R_{f,h}[u,s](\mathbf{y})\ \tilde{H}(\mathbf{y},\mathbf{x})\ d\mathbf{y}$$

By summarising

$$\forall x\in\mathbb{R}^2 : 0 = -\int_{\mathbb{R}^2}\Phi'\left(\left(R_{f,h}[w,s](\mathbf{y})\right)^2\right)\ R_{f,h}[u,s](\mathbf{y})\ \tilde{H}(\mathbf{y},\mathbf{x})\ d\mathbf{y}$$

$$= -\int_{\mathbb{R}^2}\Phi'\left(\left(R_{f,h}[w,s](\mathbf{y})\right)^2\right)\ R_{f,h}[u,s](\mathbf{y})\ s(\mathbf{x})h\left((\mathbf{y}-\mathbf{x})s(\mathbf{x})\right)\ d\mathbf{y}$$

we observe that definition (5.14) is correct in the sense of the calculus of variations. □

A.2.2 General Derivative with Respect to s

With $w := s + \varepsilon v$ we want to derive $\frac{\delta}{\delta s}E_1$. Then we get in the same way as in Section A.2.1:

$$\frac{d}{d\varepsilon}\left(E_1[u,s+\varepsilon v]\right)|_{\varepsilon=0}$$

$$= \frac{d}{d\varepsilon}\left(\frac{1}{2}\int_{\mathbb{R}^2}\Phi\left((R_{f,h}[u,s+\varepsilon v](\mathbf{x}))^2\right)\ d\mathbf{x}\right)|_{\varepsilon=0}$$

$$= \frac{d}{d\varepsilon}\left(\frac{1}{2}\int_{\mathbb{R}^2}\Phi\left((R_{f,h}[u,w](\mathbf{x}))^2\right)\ d\mathbf{x}\right)|_{\varepsilon=0}$$

Definition (5.3)

$$= \frac{d}{d\varepsilon}\left(\frac{1}{2}\int_{\mathbb{R}^2}\Phi\left((f(\mathbf{x})-\int_{\mathbb{R}^2}u(\mathbf{y})\ w(\mathbf{y})\ h((\mathbf{x}-\mathbf{y})w(\mathbf{y}))\ d\mathbf{y})^2\right)\ d\mathbf{x}\right)|_{\varepsilon=0}$$

Pull the derivative operator into the integral

$$= \left(-\int_{\mathbb{R}^2}\Phi'\left((R_{f,h}[u,w](\mathbf{x}))^2\right)\ R_{f,h}[u,w](\mathbf{x})\right.$$
$$\left.\frac{d}{d\varepsilon}\left(\int_{\mathbb{R}^2}u(\mathbf{y})\ w(\mathbf{y})\ h((\mathbf{x}-\mathbf{y})w(\mathbf{y}))\ d\mathbf{y}\right)\ d\mathbf{x}\right)|_{\varepsilon=0}$$

Again derivative into the integral, evaluate first part at $\varepsilon = 0$

$$= -\int_{\mathbb{R}^2}\Phi'\left((R_{f,h}[u,s](\mathbf{x}))^2\right)\ R_{f,h}[u,s](\mathbf{x})\int_{\mathbb{R}^2}u(\mathbf{y})\ \frac{d}{d\varepsilon}\left(w(\mathbf{y})\ h((\mathbf{x}-\mathbf{y})w(\mathbf{y}))\right)|_{\varepsilon=0}\ d\mathbf{y}\ d\mathbf{x}$$

Product rule

$$= -\int_{\mathbb{R}^2}\Phi'\left((R_{f,h}[u,s](\mathbf{x}))^2\right)\ R_{f,h}[u,s](\mathbf{x})$$
$$\int_{\mathbb{R}^2}u(\mathbf{y})\left(v(\mathbf{y})\ h((\mathbf{x}-\mathbf{y})w(\mathbf{y}))+w(\mathbf{y})\frac{d}{d\varepsilon}h((\mathbf{x}-\mathbf{y})w(\mathbf{y}))\right)|_{\varepsilon=0}\ d\mathbf{y}\ d\mathbf{x}$$

Multidimensional chain rule

$$= -\int_{\mathbb{R}^2}\Phi'\left((R_{f,h}[u,s](\mathbf{x}))^2\right)\ R_{f,h}[u,s](\mathbf{x})\int_{\mathbb{R}^2}u(\mathbf{y})\ \left(v(\mathbf{y})\ h((\mathbf{x}-\mathbf{y})w(\mathbf{y}))\right.$$
$$+w(\mathbf{y})\left(\frac{\partial}{\partial\ ((x_1-y_1)w(\mathbf{y}))}h((\mathbf{x}-\mathbf{y})w(\mathbf{y}))\ (x_1-y_1)\ v(\mathbf{y})\right.$$
$$\left.\left.+\frac{\partial}{\partial\ ((x_2-y_2)w(\mathbf{y}))}h((\mathbf{x}-\mathbf{y})w(\mathbf{y}))\ (x_2-y_2)\ v(\mathbf{y})\right)\right)|_{\varepsilon=0}\ d\mathbf{y}\ d\mathbf{x}$$

Reformulation

$$= -\int_{\mathbb{R}^2}\Phi'\left((R_{f,h}[u,s](\mathbf{x}))^2\right)\ R_{f,h}[u,s](\mathbf{x})\int_{\mathbb{R}^2}u(\mathbf{y})\ \left(v(\mathbf{y})\ h((\mathbf{x}-\mathbf{y})w(\mathbf{y}))\right.$$
$$\left.+w(\mathbf{y})\left\langle\nabla h((\mathbf{x}-\mathbf{y})w(\mathbf{y})),(\mathbf{x}-\mathbf{y})v(\mathbf{y})\right\rangle\right)|_{\varepsilon=0}\ d\mathbf{y}\ d\mathbf{x}$$

Evaluation at $\varepsilon = 0$

$$= -\int_{\mathbb{R}^2} \Phi'\left((R_{f,h}[u,s](\mathbf{x}))^2\right)\ R_{f,h}[u,s](\mathbf{x}) \int_{\mathbb{R}^2} u(\mathbf{y})\ (v(\mathbf{y})\ h((\mathbf{x}-\mathbf{y})s(\mathbf{y}))$$
$$+ s(\mathbf{y})\,\langle \nabla h((\mathbf{x}-\mathbf{y})s(\mathbf{y})), (\mathbf{x}-\mathbf{y})v(\mathbf{y})\rangle)\ d\mathbf{y}\ d\mathbf{x}$$

Factor out v

$$= -\int_{\mathbb{R}^2} \Phi'\left((R_{f,h}[u,s](\mathbf{x}))^2\right)\ R_{f,h}[u,s](\mathbf{x}) \int_{\mathbb{R}^2} u(\mathbf{y})\ v(\mathbf{y})\ (h((\mathbf{x}-\mathbf{y})s(\mathbf{y}))$$
$$+ s(\mathbf{y})\,\langle \nabla h((\mathbf{x}-\mathbf{y})s(\mathbf{y})), \mathbf{x}-\mathbf{y}\rangle)\ d\mathbf{y}\ d\mathbf{x}$$
$$= -\int_{\mathbb{R}^2}\int_{\mathbb{R}^2} \Phi'((R_{f,h}[u,s](\mathbf{x}))^2)\ R_{f,h}[u,s](\mathbf{x})\ u(\mathbf{y})\ (h((\mathbf{x}-\mathbf{y})s(\mathbf{y}))$$
$$+ s(\mathbf{y})\,\langle \nabla h((\mathbf{x}-\mathbf{y})s(\mathbf{y})), \mathbf{x}-\mathbf{y}\rangle)\ v(\mathbf{y})\ d\mathbf{y}\ d\mathbf{x}$$

Change integral order

$$= -\int_{\mathbb{R}^2}\int_{\mathbb{R}^2} \Phi'\left((R_{f,h}[u,s](\mathbf{x}))^2\right)\ R_{f,h}[u,s](\mathbf{x})\ u(\mathbf{y})\ (h((\mathbf{x}-\mathbf{y})s(\mathbf{y}))$$
$$+ s(\mathbf{y})\,\langle \nabla h((\mathbf{x}-\mathbf{y})s(\mathbf{y})), \mathbf{x}-\mathbf{y}\rangle)\ d\mathbf{x}\ v(\mathbf{y})\ d\mathbf{y}$$

Because this has to hold for every v

$$\Rightarrow 0$$
$$= -\int_{\mathbb{R}^2} \Phi'\left((R_{f,h}[u,s](\mathbf{x}))^2\right)\ R_{f,h}[u,s](\mathbf{x})\ u(\mathbf{y})\ (h((\mathbf{x}-\mathbf{y})s(\mathbf{y}))$$
$$+ s(\mathbf{y})\,\langle \nabla h((\mathbf{x}-\mathbf{y})s(\mathbf{y})), \mathbf{x}-\mathbf{y}\rangle)\ d\mathbf{x}$$

Variable substitution

$$= -\int_{\mathbb{R}^2} \Phi'\left((R_{f,h}[u,s](\mathbf{y}))^2\right)\ R_{f,h}[u,s](\mathbf{y})\ u(\mathbf{x})\ (h((\mathbf{y}-\mathbf{x})s(\mathbf{x}))$$
$$+ s(\mathbf{x})\,\langle \nabla h((\mathbf{y}-\mathbf{x})s(\mathbf{x})), \mathbf{y}-\mathbf{x}\rangle)\ d\mathbf{y}$$

And finally we have shown:

$$\forall x \in \mathbb{R}^2 : 0 = -\int_{\mathbb{R}^2} \Phi'\left((R_{f,h}[u,s](\mathbf{y}))^2\right)\ R_{f,h}[u,s](\mathbf{y})\ u(\mathbf{x})\ (h((\mathbf{y}-\mathbf{x})s(\mathbf{x}))$$
$$+ s(\mathbf{x})\,\langle \nabla h((\mathbf{y}-\mathbf{x})s(\mathbf{x})), \mathbf{y}-\mathbf{x}\rangle)\ d\mathbf{y} \qquad \square$$

A.2.3 Motion Blur Derivative with Respect to u

We basically approach in the same way as in Section A.2.1, the main difference is that we define $w\begin{pmatrix} y \\ x_2 \end{pmatrix} := u\begin{pmatrix} y \\ x_2 \end{pmatrix} + \varepsilon v\begin{pmatrix} y \\ x_2 \end{pmatrix}$ and use $\hat{H}$ instead of $\tilde{H}$.

$$\frac{d}{d\varepsilon}\left(\hat{E}_1[w,s]\right)|_{\varepsilon=0}$$

$$= \frac{d}{d\varepsilon}\left(\frac{1}{2}\int_{\mathbb{R}^2}\Phi\left((\hat{R}_{f,h}[w,s](\mathbf{x}))^2\right)d\mathbf{x}\right)|_{\varepsilon=0}$$

$$= \frac{d}{d\varepsilon}\left(\frac{1}{2}\int_{\mathbb{R}^2}\Phi\left(\left(f(\mathbf{x})-\int_{\mathbb{R}} w\begin{pmatrix}y\\x_2\end{pmatrix}\ \hat{H}(\mathbf{x},y)\ dy\right)^2\right)\ d\mathbf{x}\right)|_{\varepsilon=0}$$

$$= \left(-\int_{\mathbb{R}^2}\Phi'\left((\hat{R}_{f,h}[w,s](\mathbf{x}))^2\right)\ \hat{R}_{f,h}[w,s](\mathbf{x})\left(\int_{\mathbb{R}} v\begin{pmatrix}y\\x_2\end{pmatrix}\ \hat{H}(\mathbf{x},y)\ dy\right)\ d\mathbf{x}\right)|_{\varepsilon=0}$$

$$= -\int_{\mathbb{R}^2}\Phi'\left((\hat{R}_{f,h}[u,s](\mathbf{x}))^2\right)\ \hat{R}_{f,h}[u,s](\mathbf{x})\left(\int_{\mathbb{R}} v\begin{pmatrix}y\\x_2\end{pmatrix}\ \hat{H}(\mathbf{x},y)\ dy\right)\ d\mathbf{x}$$

$$= -\int_{\mathbb{R}}\int_{\mathbb{R}}\int_{\mathbb{R}}\Phi'\left((\hat{R}_{f,h}[u,s](\mathbf{x}))^2\right)\ \hat{R}_{f,h}[u,s](\mathbf{x})\ v\begin{pmatrix}y\\x_2\end{pmatrix}\ \hat{H}(\mathbf{x},y)\ dy\ dx_1\ dx_2$$

Change integral order

$$= -\int_{\mathbb{R}}\int_{\mathbb{R}}\int_{\mathbb{R}}\Phi'\left((\hat{R}_{f,h}[u,s](\mathbf{x}))^2\right)\ \hat{R}_{f,h}[u,s](\mathbf{x})\ \hat{H}(\mathbf{x},y)\ d\mathbf{x}_1\ v\begin{pmatrix}y\\x_2\end{pmatrix}\ dy\ dx_2$$

Because this has to hold for every v

$\Rightarrow 0$

$$= -\int_{\mathbb{R}}\Phi'\left((\hat{R}_{f,h}[u,s](\mathbf{x}))^2\right)\ \hat{R}_{f,h}[u,s](\mathbf{x})\ \hat{H}(\mathbf{x},y)\ dx_1$$

Variable substitution (substitute x_1 by y)

$$= -\int_{\mathbb{R}}\Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix}y\\x_2\end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s]\begin{pmatrix}y\\x_2\end{pmatrix}\ \hat{H}\left(\begin{pmatrix}y\\x_2\end{pmatrix},x_1\right)\ dy$$

This gives us the Euler-Lagrange equation for the data term with respect to u:

$$\forall x\in\mathbb{R}^2: 0 = -\int_{\mathbb{R}}\Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix}y\\x_2\end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s]\begin{pmatrix}y\\x_2\end{pmatrix}\ \hat{H}\left(\begin{pmatrix}y\\x_2\end{pmatrix},x_1\right)\ dy \quad \square$$

A.2.4 Motion Blur Derivative with Respect to s

Again using $w\begin{pmatrix}y\\x_2\end{pmatrix} := s\begin{pmatrix}y\\x_2\end{pmatrix} + \varepsilon v\begin{pmatrix}y\\x_2\end{pmatrix}$ and analogous to Section A.2.2 we get:

$$\frac{d}{d\varepsilon}\left(\hat{E}_1[u,w]\right)|_{\varepsilon=0}$$

$$= \frac{d}{d\varepsilon}\left(\frac{1}{2}\int_{\mathbb{R}^2}\Phi\left((\hat{R}_{f,h}[u,w](\mathbf{x}))^2\right)d\mathbf{x}\right)|_{\varepsilon=0}$$

Definition

$$= \frac{d}{d\varepsilon}\left(\frac{1}{2}\int_{\mathbb{R}^2}\Phi\left(\left(f(\mathbf{x}) - \int_{\mathbb{R}} u\begin{pmatrix}y\\x_2\end{pmatrix}\ w\begin{pmatrix}y\\x_2\end{pmatrix}\ h\left((x_1-y)w\begin{pmatrix}y\\x_2\end{pmatrix}\right)dy\right)^2\right)d\mathbf{x}\right)|_{\varepsilon=0}$$

Pull the derivative operator into the intergral

$$= \left(-\int_{\mathbb{R}^2}\Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix}y\\x_2\end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,w](\mathbf{x})$$
$$\frac{d}{d\varepsilon}\left(\int_{\mathbb{R}} u\begin{pmatrix}y\\x_2\end{pmatrix}\ w\begin{pmatrix}y\\x_2\end{pmatrix}\ h\left((x_1-y)w\begin{pmatrix}y\\x_2\end{pmatrix}\right)\ dy\right)\ d\mathbf{x}\Bigg)|_{\varepsilon=0}$$

Again derivative into the integral, evaluate first part at $\varepsilon = 0$

$$= -\int_{\mathbb{R}^2}\Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix}y\\x_2\end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s](\mathbf{x})$$
$$\int_{\mathbb{R}} u\begin{pmatrix}y\\x_2\end{pmatrix}\ \frac{d}{d\varepsilon}\left(w\begin{pmatrix}y\\x_2\end{pmatrix}\ h\left((x_1-y)\ w\begin{pmatrix}y\\x_2\end{pmatrix}\right)\right)|_{\varepsilon=0}\ dy\ d\mathbf{x}$$

Product rule

$$= -\int_{\mathbb{R}^2}\Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix}y\\x_2\end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s](\mathbf{x})\int_{\mathbb{R}} u\begin{pmatrix}y\\x_2\end{pmatrix}\left(v\begin{pmatrix}y\\x_2\end{pmatrix}\ h\left((x_1-y)w\begin{pmatrix}y\\x_2\end{pmatrix}\right)\right.$$
$$\left.+\ w\begin{pmatrix}y\\x_2\end{pmatrix}\ h'\left((x_1-y)w\begin{pmatrix}y\\x_2\end{pmatrix}\right)\ (x_1-y)\ v\begin{pmatrix}y\\x_2\end{pmatrix}\right)|_{\varepsilon=0}\ dy\ d\mathbf{x}$$

Evaluation at $\varepsilon = 0$

$$= -\int_{\mathbb{R}^2}\Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix}y\\x_2\end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s](\mathbf{x})\int_{\mathbb{R}} u\begin{pmatrix}y\\x_2\end{pmatrix}\left(v\begin{pmatrix}y\\x_2\end{pmatrix}h\left((x_1-y)s\begin{pmatrix}y\\x_2\end{pmatrix}\right)\right.$$
$$\left.+\ s\begin{pmatrix}y\\x_2\end{pmatrix}\ h'\left((x_1-y)s\begin{pmatrix}y\\x_2\end{pmatrix}\right)\ (x_1-y)\ v\begin{pmatrix}y\\x_2\end{pmatrix}\right)\ dy\ d\mathbf{x}$$

Factor out v

$$= -\int_{\mathbb{R}^2}\Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix}y\\x_2\end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s](\mathbf{x})\int_{\mathbb{R}} u\begin{pmatrix}y\\x_2\end{pmatrix}\ \left(h\left((x_1-y)s\begin{pmatrix}y\\x_2\end{pmatrix}\right)\right.$$
$$\left.+\ s\begin{pmatrix}y\\x_2\end{pmatrix}\ h'\left((x_1-y)s\begin{pmatrix}y\\x_2\end{pmatrix}\right)\ (x_1-y)\right)\ v\begin{pmatrix}y\\x_2\end{pmatrix}\ dy\ d\mathbf{x}$$
$$= -\int_{\mathbb{R}}\int_{\mathbb{R}}\int_{\mathbb{R}}\Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix}y\\x_2\end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s](\mathbf{x})u\begin{pmatrix}y\\x_2\end{pmatrix}\ \left(h\left((x_1-y)s\begin{pmatrix}y\\x_2\end{pmatrix}\right)\right.$$
$$\left.+\ s\begin{pmatrix}y\\x_2\end{pmatrix}\ h'\left((x_1-y)s\begin{pmatrix}y\\x_2\end{pmatrix}\right)(x_1-y)\right)\ v\begin{pmatrix}y\\x_2\end{pmatrix}\ dy\ dx_1\ dx_2$$

Change integral order

$$= -\int_{\mathbb{R}}\int_{\mathbb{R}}\int_{\mathbb{R}} \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix}y\\x_2\end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s](\mathbf{x})u\begin{pmatrix}y\\x_2\end{pmatrix}\ \left(h\left((x_1-y)s\begin{pmatrix}y\\x_2\end{pmatrix}\right)\right.$$
$$\left. + s\begin{pmatrix}y\\x_2\end{pmatrix}\ h'\left((x_1-y)s\begin{pmatrix}y\\x_2\end{pmatrix}\right)(x_1-y)\right)\ dx_1\ v\begin{pmatrix}y\\x_2\end{pmatrix}\ dx_2\ dy$$

Because this has to hold for every v

$\Rightarrow 0$

$$= -\int_{\mathbb{R}} \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix}y\\x_2\end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s](\mathbf{x})u\begin{pmatrix}y\\x_2\end{pmatrix}\ \left(h\left((x_1-y)s\begin{pmatrix}y\\x_2\end{pmatrix}\right)\right.$$
$$\left. + s\begin{pmatrix}y\\x_2\end{pmatrix}\ h'\left((x_1-y)s\begin{pmatrix}y\\x_2\end{pmatrix}\right)(x_1-y)\right)\ dx_1$$

Variable substitution (substitute x_1 by y)

$$= -\int_{\mathbb{R}} \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix}y\\x_2\end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s]\begin{pmatrix}y\\x_2\end{pmatrix}\ u(\mathbf{x})\ \left(h\left((y-x_1)s(\mathbf{x})\right)\right.$$
$$\left. + s(\mathbf{x})\ h'\left((y-x_1)s(\mathbf{x})\right)(y-x_1)\right)\ dy$$

So finally we get the motion blur equation with respect to s:

$$\forall x \in \mathbb{R}^2 : 0 = -\int_{\mathbb{R}} \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix}y\\x_2\end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s]\begin{pmatrix}y\\x_2\end{pmatrix}\ u(\mathbf{x})\ \left(h\left((y-x_1)s(\mathbf{x})\right)\right.$$
$$\left. + s(\mathbf{x})\ h'\left((y-x_1)s(\mathbf{x})\right)(y-x_1)\right)\ dy$$

□

A.3 Distributions and Integrals

In this paragraph we will not keep perfect mathematical strictness. If one is interested in the backgrounds and the whole theory of distributions, we refer to the book [18].

First of all we take the derived PSF h' as given in equation (6.4) and plug the argument $(y-x_1)s(\mathbf{x})$ into it. The derivative contains the delta-function, as it is defined in (6.5).

$$\begin{aligned} h'\left((y-x_1)s(\mathbf{x})\right) &= \frac{1}{2r}\left(\delta\left((y-x_1)s(\mathbf{x})+r\right)-\delta\left((y-x_1)s(\mathbf{x})-r\right)\right) \\ &= \begin{cases} \frac{1}{2r}\delta & \text{if } y = x_1 - \frac{r}{s(\mathbf{x})} \\ -\frac{1}{2r}\delta & \text{if } y = x_1 + \frac{r}{s(\mathbf{x})} \\ 0 & \text{otherwise} \end{cases} \\ &= \frac{1}{2r}\left(\delta\left(y-x_1+\frac{r}{s(\mathbf{x})}\right)-\delta\left(y-x_1-\frac{r}{s(\mathbf{x})}\right)\right) \end{aligned}$$

By defining

$$\Theta(y) := \Phi'\left(\left(\hat{R}_{f,h}[u,s]\begin{pmatrix}y\\x_2\end{pmatrix}\right)^2\right)\ \hat{R}_{f,h}[u,s]\begin{pmatrix}y\\x_2\end{pmatrix}\ u(\mathbf{x})\ s(\mathbf{x})\ (y-x_1) \tag{A.1}$$

and using the following definition of the heaviside function $\dot{H}$ (do not mix this $\dot{H}$ up with a spatially variant PSF)

$$\dot{H}(x) := \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \qquad \dot{H}'(x) = \delta(x)$$

we can reduce the above integral as follows:

$$\begin{aligned}
&\int_{\mathbb{R}} \Theta(y)\ h'\left((y - x_1)s(\mathbf{x})\right)\ dy \\
&= \frac{1}{2r} \int_{\mathbb{R}} \Theta(y) \left(\delta\left(y - x_1 + \frac{r}{s(\mathbf{x})}\right) - \delta\left(y - x_1 - \frac{r}{s(\mathbf{x})}\right)\right)\ dy \\
&= \frac{1}{2r} \int_{\mathbb{R}} \Theta(y) \left(\dot{H}'\left(y - x_1 + \frac{r}{s(\mathbf{x})}\right) - \dot{H}'\left(y - x_1 - \frac{r}{s(\mathbf{x})}\right)\right)\ dy \\
&\text{Partial integration} \\
&= 0 - \frac{1}{2r} \int_{\mathbb{R}} \Theta'(y) \left(\dot{H}\left(y - x_1 + \frac{r}{s(\mathbf{x})}\right) - \dot{H}\left(y - x_1 - \frac{r}{s(\mathbf{x})}\right)\right)\ dy \\
&\text{Definition of } \dot{H} \\
&= -\frac{1}{2r} \int_{x_1 - \frac{r}{s(\mathbf{x})}}^{x_1 + \frac{r}{s(\mathbf{x})}} \Theta'(y) \underbrace{\left(\dot{H}\left(y - x_1 + \frac{r}{s(\mathbf{x})}\right) - \dot{H}\left(y - x_1 - \frac{r}{s(\mathbf{x})}\right)\right)}_{=1}\ dy \\
&\text{Fundamental theorem of calculus} \\
&= \frac{1}{2r} \left(\Theta\left(x_1 - \frac{r}{s(\mathbf{x})}\right) - \Theta\left(x_1 + \frac{r}{s(\mathbf{x})}\right)\right)
\end{aligned}$$

This identity is now exactly what we had to show. One only has to plug in the definition of Θ as given in equation (A.1) and one gets the right equation. □

Bibliography

[1] S. Bae and F. Durand. Defocus magnification. *Computer Graphics Forum*, 26(3):571–579, 2007.

[2] L. Bar, N. Sochen, and N. Kiryati. Image deblurring in the presence of salt-and-pepper noise. In *Scale Space and PDE Methods in Computer Vision*, volume 3459 of *Lecture Notes in Computer Science*, pages 107–118, Berlin, 2005. Springer.

[3] J. H. Elder and S. W. Zucker. Local scale control for edge detection and blur estimation, 1996.

[4] P. Favaro, M. Burger, and S. Soatto. Scene and motion reconstruction from defocused and motion-blurred images via anisotropic diffusion. In *Computer Vision - ECCV 2004*, volume 3021 of *Lecture Notes in Computer Science*, pages 257–269, Berlin, 2004. Springer.

[5] P. Favaro, S. Osher, S. Soatto, and L. A. Vese. 3D shape from anisotropic diffusion. *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 1:179–186, 2003.

[6] P. Favaro and S. Soatto. Learning shape from defocus. In *Computer Vision — ECCV 2002*, volume 2351 of *Lecture Notes in Computer Science*, pages 823–824, Berlin, 2002. Springer.

[7] P. Favaro, S. Soatto, M. Burger, and S. Osher. Shape from defocus via diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):518–531, 2008.

[8] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. *ACM Transactions on Graphics*, 25(3):787–794, 2006.

[9] H. Jin and P. Favaro. A variational approach to shape from defocus. In *Computer Vision — ECCV 2002*, volume 2351 of *Lecture Notes in Computer Science*, pages 18 – 30, Berlin, 2002. Springer.

[10] A. Levin. Blind motion deblurring using image statistics. In *Advances in Neural Information Processing Systems*, volume 19, pages 841–848. MIT Press, Cambridge, MA, 2007.

[11] H.-Y. Lin and C.-H. Chang. Depth recovery from motion and defocus blur. In *Image Analysis and Recognition*, volume 4142 of *Lecture Notes in Computer Science*, pages 122–133, Berlin, 2006. Springer.

[12] L. B. Lucy. An iterative technique for the rectification of observed distributions. *The Astronomical Journal*, 79:745–754, 1974.

[13] J. H. Money and S. H. Kang. Total variation minimizing blind deconvolution with shock filter reference. *Image and Vision Computing*, 26(2):302–314, 2008.

[14] W. H. Richardson. Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America*, 62(1):55–59, January 1972.

[15] A. Saxena, S. H. Chung, and A. Y. Ng. 3-d depth reconstruction from a single still image. *International Journal of Computer Vision*, 76(1):53–69, 2008.

[16] Y. Y. Schechner and N. Kiryati. Depth from defocus vs. stereo: How different really are they? In *Proceedings of the International Conference on Pattern Recognition*, volume 2, pages 1784–1786, 1998.

[17] Q. Shan, J. Jia, and A. Agarwala. High-quality motion deblurring from a single image. In *ACM Transactions on Graphics*, volume 27, pages 1–10, New York, NY, USA, 2008. ACM.

[18] W. Walter. *Einführung in die Theorie der Distributionen.* Spektrum Akademischer Verlag, 1994.

[19] J. Weickert. *Anisotropic Diffusion in Image Processing.* European Consortium for Mathematics in Industry. Teubner, 1998.

[20] J. Weickert. Image processing and computer vision. Lecture Notes, 2007. Lecture 2.

[21] J. Weickert. Differential equations in image processing and computer vision. Lecture Notes, 2009. Lecture 10.

[22] M. Welk. *Dynamic and Geometric Contributions to Digital Image Processing.* Habilitation thesis, Saarland University, 2007. Chapter 6.

[23] M. Welk, D. Theis, T. Brox, and J. Weickert. PDE-based deconvolution with forward-backward diffusivities and diffusion tensors. In *Scale Space and PDE Methods in Computer Vision*, volume 3459 of *Lecture Notes in Computer Science*, pages 585–597, Berlin, 2005. Springer.

[24] M. Welk, D. Theis, and J. Weickert. Variational deblurring of images with uncertain and spatially variant blurs. In *Pattern Recognition*, volume 3663 of *Lecture Notes in Computer Science*, pages 485–492, Berlin, 2005. Springer.

[25] N. Wiener. *Extrapolation, Interpolation and Smoothing of Stationary Time Series with Engineering Applications.* The MIT Press, Cambridge (Mass.), 1949.

[26] Y.-L. You and M. Kaveh. A regularization approach to joint blur identification and image restoration. *IEEE Transactions on Image Processing*, 5(3):416–428, 1996.

[27] Y.-L. You and M. Kaveh. Blind image restoration by anisotropic regularization. *IEEE Transactions on Image Processing*, 8(3):396–407, 1999.