Technische Universität München
Zentrum Mathematik
Wissenschaftliches Rechnen

An Implicit Proximal Method for Motion Segmentation

Karin Barbara Tichmann

Vollständiger Abdruck der von der Fakultät für Mathematik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. Daniel Matthes
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. Oliver Junge
	2.	Univ.-Prof. Dr. Joachim Weickert Universität des Saarlandes (nur schriftliche Beurteilung)
	3.	Hon.-Prof. Dr. Carsten Steger

Die Dissertation wurde am 27.05.2015 bei der Technischen Universität München eingereicht und durch die Fakultät für Mathematik am 14.08.2015 angenommen.

Acknowledgment

It is a pleasure for me to thank those who made this thesis possible:

In particular, I deeply thank my supervisor Prof. Dr. Oliver Junge for encouragement and guidance. With endless motivation through endless discussions, he always suggested lines of action on how to carry on, and with this made it possible to perform the necessary work.

Very special thanks goes to Dr. Markus Ulrich and Dr. Carsten Steger of the company MVTec Software GmbH for fruitful discussions and ideas and for fostering this work.

I am grateful to many colleagues at the TUM, in particular I thank Dr. Maximilian Baust, Dr. Péter Koltai and Andre Milzarek for helpful discussions and encouragement. I thank my colleagues at the chair of scientific computing for providing a pleasant and enjoyable atmosphere to work in.

Finally, my most sincere thanks goes to my family and friends for never-ending support, encouragement and patience during the last months.

Zusammenfassung

Diese Arbeit entwickelt einen neuen iterativen Algorithmus für eine Klasse konvexer Optimierungsprobleme mit Nebenbedingungen, die unter anderem bei Variationsmodellen aus der Bildverarbeitung auftreten. Das neue Verfahren überwindet Probleme etablierter Methoden wo Schrittweitenbeschränkungen zu langsamer Konvergenz führen. Basierend auf der Theorie monotoner Operatoren werden Konvergenzeigenschaften entwickelt. Numerische Experimente, primär mit Modellen für Bewegungssegmentierung in Echtzeit, die in dieser Arbeit ebenfalls entwickelt werden, zeigen die Leistung des neuen Verfahrens.

Der neue Algorithmus ist ein implizites proximal Verfahren. Er ist nicht anfällig gegenüber steifen Systemen, im Gegensatz zu einigen proximalen Verfahren, wie zum Beispiel das proximale Gradientenverfahren, oder aktuelle primal-duale Methoden. Solche steifen Systeme können unter anderem auch von Regularisierungstermen aus Bildverarbeitungsproblemen herrühren. Schwierigkeiten, die bei der Berechnung des impliziten Verfahrens auftreten, können mit einer Näherung überwunden werden. Diese Näherung bewahrt die Eigenschaften des exakten Verfahrens und ist zusätzlich effektiv berechenbar, liefert aber nur eine nicht-exakte Lösung. Wir zeigen jedoch, dass der Abstand zwischen exakter und nicht-exakter Lösung beschränkt und direkt proportional zur verwendeten Schrittweite ist.
Das implizite proximal Verfahren kann auch als Update-Schritt in eine „Splitting"-Methode, wie die „Alternating Direction Method of Multipliers", (ADMM), integriert werden. In dieser Form wird der neue Algorithmus auf das Problem der Bewegungssegmentierung angewandt, wo er im Vergleich zum primal-dualen Algorithmus, sowie dem klassischen ADMM um einen Faktor fünf bis zehn schneller ist, gleichzeitig aber mindestens gleich gute Bewegungssegmentierungen liefert. Letztendlich arbeiten wir auf einen echtzeitfähigen Algorithmus für die Bewegungssegmentierung hin.
Zu diesem Zweck werden Modelle für zwei- und mehrteilige Bewegungssegmentierung vorgeschlagen. Aus unterschiedlichen variationellen Modellen zur Bewegungssegmentierung wird ein Modell aus zwei Gebieten („2L-Modell") aufgebaut. Damit erhält man eine Segmentierung in einen bewegten Vordergrund und einen bewegten Hintergrund zusammen mit einer Abschätzung für einen konstanten oder affinen Bewegungsvektor für jedes der beiden Gebiete. Zur Erweiterung auf mehrere Gebiete („nL-Modell") werden zwei Modelle vorgeschlagen. Basierend auf existierenden Ideen aus der Mehrgebietssegmentierung und jüngsten Ansätzen zur Bewegungssegmentie-

rung wird das 2L-Modell mit zusätzlichen Label-Funktionen erweitert. Im zweiten Modell kommt ein Verfahren zur Anwendung, das direkt aus dem erwähnten 2L-Modell hervorgeht. Dazu wird zunächst nur ein Gebiet mit einem konstanten oder affinen Bewegungsvektor versehen. Die sich aus der Unterteilung des Restgebietes ergebenden, bewegten Gebiete, erhalten ihre zugehörigen Bewegungsvektoren in einem Nachbearbeitungsschritt. Während das erste nL-Modell mit zusätzlichen Labelfunktionen für n Gebiete in den numerischen Testumgebungen eine etwa n-mal längere Rechenzeit als das 2L-Modell benötigte, war das zweite Modell fast genauso schnell wie das 2L-Modell, dafür aber weniger robust.

Abstract

This thesis develops a new iterative method for a class of convex constrained optimization problems, which occur inter alia in imaging models based on variational approaches. The new method overcomes problems of established optimization methods, such as step size restrictions and resulting slow convergence. Convergence properties are derived, based on monotone operator theory. Numerical experiments, primarily on the application of real-time motion segmentation, for which suitable models are developed, show the performance of the new method.

The developed optimization algorithm is an implicit proximal method. It is not prone to stiff systems, to which some proximal methods, e.g. the proximal gradient method or recent primal-dual methods, are sensitive. Such stiff systems can also arise from regularization terms in imaging problems. Challenges with calculations required for the implicit method are overcome by an approximation, which preserves the properties of the exact method and is efficiently computable, but gives only an inexact solution. However, we show that the distance between the exact and inexact solution is bounded and proportional to the step size.
The implicit proximal method can also be included into the update steps of splitting methods as the alternating direction method of multipliers (ADMM). In this form, it is applied to the motion segmentation application, where it outperforms the primal-dual algorithm and the classical ADMM by a factor of five to ten in runtime, while the resulting motion segmentations are equally good or better. Eventually, we aim for a real-time capable algorithm for the motion segmentation problem.
For this purpose, suitable models for two-label and multi-label motion segmentation are proposed. From different variational motion segmentation models a two-label motion segmentation model is assembled, which yields a segmentation into a moving foreground and a moving background, as well as an estimate for a constant or affine motion vector for each of these two regions. For the task of multi-label motion segmentation, two models are proposed. The first works with an adaption of a two-label model with multiple labeling functions, which is based on existing ideas of multi-label segmentation and also more recent motion segmentation approaches. The second model uses a technique which is derived directly from the mentioned two-label model. Only one area is modeled with a constant or affine motion vector, while the remaining moving regions are segmented and provided with motion vectors in a post processing step. While the first multi-label approach for n regions usually takes about n times as long as

the two-label approach, the second approach is almost as fast as the two-label approach — but less robust.

Contents

1 Introduction

> *"Much like Newton's method is a standard tool for solving unconstrained smooth minimization problems of modest size, proximal algorithms can be viewed as an analogous tool for nonsmooth, constrained, large-scale, or distributed versions of these problems."*
> [*Parikh and Boyd, 2013*]

Many of the algorithms used and evolved for recent imaging models are based on optimization methods developed since the 1950s, in particular on the proximal point method. Bit by bit, it turned out that between these recently used algorithms many connections, often even equivalences exist, and some are special cases of others, which is in general not an easy task to show. Nevertheless some of the algorithms are more easy to adapt to problems and to implement, and therefore became well established and used in many applications throughout different manuscripts. However, it turns out that for some applications these algorithms work only for very small step sizes, causing slow convergence. For other applications, the step size restrictions are less severe or can be violated without loosing the convergence properties of the algorithms in practice.
Such problems have also arisen in the theory of ordinary differential equations, where the characteristic of a problem yielding the small step size restriction which prevents the algorithm from converging in reasonable time was called stiffness. It turned out that explicit schemes, as the explicit Euler scheme, are prone to stiffness, while implicit schemes are not. Even though proximal algorithms are of implicit nature by construction, many of the above optimization algorithms contain explicit gradient steps.
The task we consider is to find a segmentation of an image sequence into moving objects with associated motion vectors preferably in real-time. This problem is a combination of motion estimation, where a motion vector is defined for every pixel of the image domain, and segmentation, where the image is partitioned into regions with similar properties. Both tasks have been studied intensively and sought to be solved in real-time, for example in advanced driver assistance systems, (traffic) surveillance, medical imaging and assistance systems, or motion sensing input devices mostly known from video game consoles. Even though the models for both separated tasks, together with suitable algorithms, have experienced large development, the combination of both tasks turns out to be one of those problems, where the step size restriction of the used algorithm is a crucial point.
In the following work, we analyze some well established methods for variational imag-

ing models with special attention to explicit schemes, and develop a new proximal method with a fully implicit scheme that overcomes the problems with stiff systems, while trying to preserve an easily implementable structure. Since the iterates of implicit schemes tend to be computationally involved, we focus in particular on efficient solvability and approximations that are fast to compute.

Related work. The name proximal operators was probably coined in the work of [Moreau, 1962] in the 1960s while the relationship between proximal operators and resolvents was explored in [Rockafellar, 1976] through the handling of the subdifferential as a monotone operator. Already in the 1950s gradient methods were investigated through numerical methods for ordinary differential equations in [Arrow et al., 1958]. Algorithms, such as the alternating iterative scheme of [Bregman, 1967], were invented independently while connections were explored later.
In [Esser, 2009, Esser et al., 2010, Setzer, 2011, Yin et al., 2008] similarities of Bregman iterations, the method of multipliers and first order primal-dual methods, which are based on the proximal point method, are analyzed. A connection of proximal methods and the Douglas–Rachford splitting is provided in [Eckstein and Bertsekas, 1992]. A recent survey by [Boyd et al., 2011] on the alternating direction method of multipliers (ADMM), and its sequel [Parikh and Boyd, 2013] on proximal methods give a comprehensive overview on algorithms and applications including detailed references on previous work and historical development. In the recent monograph of [Bauschke and Combettes, 2011], monotone operator theory is explored in detail with connections to the mentioned algorithms.
These algorithms are extensively utilized in computer vision applications, and in particular in motion estimation and segmentation. For example, in [Goldstein et al., 2010] the split Bregman method is proposed for the segmentation of images, and in [Möllenhoff et al., 2013] the ADMM is compared with the primal-dual algorithm on minimal partition problems. In [Pock et al., 2009, Chambolle and Pock, 2011] a primal-dual algorithm based on proximal methods is presented and applied to many different imaging problems, such as optical flow, segmentation and inpainting.
The combination of motion estimation and segmentation has been addressed in many works, among them are [Brox and Weickert, 2004] and [Cremers and Soatto, 2005], where the motion segmentation problem has been formulated in the variational framework for segmentation by [Chan et al., 2000, Vese and Chan, 2002] with optical flow data terms. More recently, a motion segmentation model has been proposed in [Unger et al., 2012], where the authors propose a convex constraint model, which is solved by the primal-dual algorithm from [Chambolle and Pock, 2011].

Contribution. The contribution of this thesis lies in the analysis of the optimization methods and convergence properties, and the resulting development of a new implicit

algorithm, which overcomes the problem of explicit algorithms with stiff systems. Further, a convergence analysis for the new method is provided. Possible simplifications and approximations of parts which are challenging to compute, due to the implicit step, are proposed and analyzed. The new algorithm is tested and compared with other state of the art algorithms, primarily on motion segmentation examples, but also on further imaging tasks such as denoising and inpainting.
In addition, different (convex) models for two-label and multi-label motion segmentation are presented, and a new model for fast multi-label motion segmentation is proposed, tested and compared with the other models, utilizing our newly developed algorithm.

Outline. This thesis is structured as follows:

- In the second and third chapter models for different imaging problems are presented. Chapter 2 introduces the concept of variational methods in imaging and presents, in preparation for the motion segmentation task, different models for optical flow and segmentation. Also further imaging tasks such as denoising and inpainting are addressed briefly. In chapter 3, two-label and multi-label motion segmentation models are adapted from the combination of optical flow and segmentation models given in chapter 2. Further, a new model for multi-label motion segmentation is proposed.

- The fourth and fifth chapter are about algorithms for convex optimization problems, which often arise from imaging problems. In chapter 4, well known algorithms used for imaging tasks are presented and compared. Based on the theory of monotone operators, proximal methods are introduced and the primal-dual algorithm and the ADMM are described and compared. In chapter 5, a new proximal algorithm is developed, which has no step size restriction and hence is not prone to stiff systems. Further, convergence properties and fixed points of the new algorithm are analyzed.

- Chapter 6 connects application and algorithms. The algorithms given in chapters 4 and 5, are adapted to the motion segmentation problem and compared on several examples, with respect to runtime and accuracy.

- Chapter 7 finally summarizes our findings and gives an outlook on future work.

2 Tools and Models for Mathematical Imaging

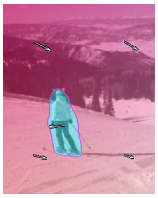
In this chapter, a technique for imaging problems based on solving partial differential equations is described, the concept of variational methods. All applications described and analyzed in this work use a variational approach, thus a brief introduction is given here using the example of image denoising.
Further, different approaches for segmentation and optical flow are explained which are the basis of our motion segmentation model which is given in chapter 3.

2.1 Variational Methods

In variational approaches, model assumptions are formulated as penalty terms and added in a so called *energy functional*. These add a high value (or cost) to the functional if the model assumption is not fulfilled and a low value otherwise. Therefore, the function minimizing this energy functional, possibly respecting additional constraints, fulfills the formulated assumptions best and is the solution to the given problem.
Mathematically speaking, the penalizer terms are formulated as functions of the form $F_i : \mathcal{H} \to [0, \infty)$ on a real Hilbert space $\mathcal{H}$ with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and induced norm $\|\cdot\|_{\mathcal{H}}$. The energy functional is given by

$$J(u) = F_1(u) + F_2(u) + \dots, \tag{2.1}$$

and the solution u_* minimizes $J(u)$. In order to find a minimizer u_*, one analyzes (local) optimality conditions derived from the variation of u around u_* - hence the name variational method.
In image processing applications, the energy functionals can often be divided into two parts, the data terms and the regularization terms. The data terms consist of all penalizer terms measuring the pixelwise error according to the application, for example, for a denoising application, the difference between the noisy image and the reconstruction. The regularization term also includes neighboring attributes, such as smoothness assumptions.
The result of the optimization depends highly on the regularization and thus on the assumptions we make.
Many tasks in image processing are inverse problems such as deconvolution, which are often ill-posed. [Chan and Shen, 2005]



A very common assumption is that large parts of the image are considered smooth, which means that every pixel is in a way connected to its neighboring pixel. In the following we introduce different regularization terms using this assumption.
As an example for an energy functional with data term and regularization term, we consider a variational model for denoising. Let $f : \Omega \to \mathbb{R}$ be a noisy gray-value image, and the image domain $\Omega \subset \mathbb{R}^2$. We seek to reconstruct the original image $u : \Omega \to \mathbb{R}$ without noise. As a first assumption, the difference between the noisy image f and u should be small, yielding the data term. Thus, a function is used, that penalizes high differences. In this example, we use the quadratic function

$$F_{\text{data}}(u) = \frac{1}{2} \int_\Omega (f(x) - u(x))^2 \, \mathrm{d}x. \tag{2.2}$$

This data term alone is of course not sufficient to reconstruct a noise-free image. Therefore a second assumption is introduced, the regularization. As already said, we assume, that most areas in the image will be smooth, i.e. differences in the gray-values between neighboring pixels are small. The gradient of the image provides information on neighboring pixels and again, we choose a quadratic function to penalize large gray-value changes, i.e. large values in the gradient of the reconstructed image u:

$$F_{\text{reg}}(u) = \frac{1}{2} \int_\Omega \|\nabla u(x)\|_2^2 \, \mathrm{d}x. \tag{2.3}$$

This regularization is sometimes referred to as the classical Tikhonov regularization, cf. [Bredies and Lorenz, 2011, Chan and Shen, 2005, Tikhonov, 1963]. The energy functional for the denoising problem reads

$$J(u) = \frac{1}{2} \int_\Omega (u(x) - f(x))^2 + \mu \|\nabla u(x)\|_2^2 \, \mathrm{d}x, \tag{2.4}$$

where $\mu > 0$ is a weighting parameter.
The Euler-Lagrange equation, yielding an elliptic partial differential equation, is given by

$$\mu \Delta u(x) = u(x) - f(x), \tag{2.5}$$

with Neumann boundary conditions. The Euler-Lagrange equation is a necessary condition for a sufficiently smooth minimizer of J. In some cases, the quadratic function in the regularization term forces too much blurring on the edges. This suggests another regularization term, the so called *total variation regularization (TV)*. The TV-regularization term is given by

$$F_{TV}(u) = \sup \left\{ \int_\Omega u \operatorname{div} \varphi \, \mathrm{d}x \;\middle|\; \varphi \in \mathcal{D}(\Omega, \mathbb{R}^2), \|\varphi\|_\infty \leq 1 \right\}, \tag{2.6}$$

where $\mathcal{D}(\Omega, \mathbb{R}^2)$ is the space of compactly supported infinitely often differentiable functions $\varphi : \Omega \to \mathbb{R}^2$ and $\|\varphi\|_\infty = \sup_{x \in \Omega} \|\varphi(x)\|_2$ the supremum norm, see e.g. [Bredies

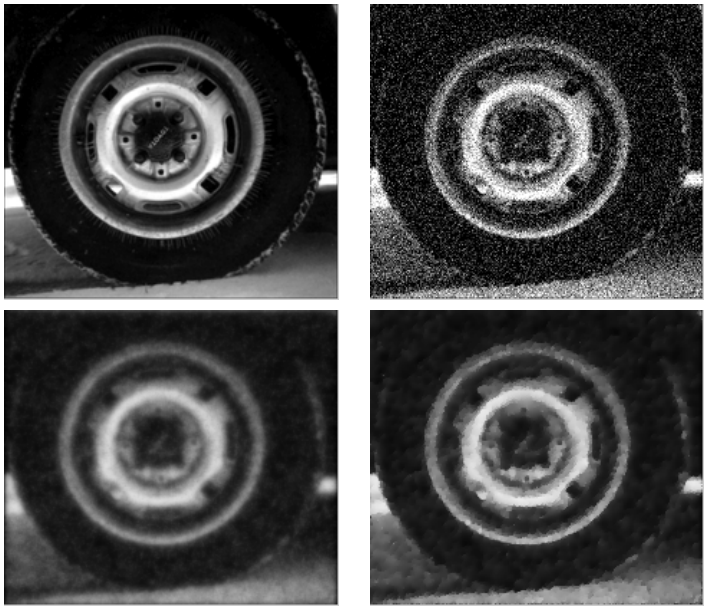


Figure 2.1: Upper row: original and noisy image. Lower row: denoising with quadratic regularization term (left) and TV regularization (right).

and Lorenz, 2011]. For sufficiently smooth functions u, there is a more intuitive representation for the TV-regularization (2.6):

$$F_{TV}(u) = \int_\Omega \|\nabla u(x)\|_2 \, \mathrm{d}x, \tag{2.7}$$

but this representation is not applicable for functions u that are discontinuous, i.e. functions that can represent images with edges or small structures. Thus, for functions with jump discontinuities, the total variation regularization from (2.6) has to be used. Also, functional (2.7) is not lower-semicontinuous on the associated Sobolev space of weakly differentiable functions. Lower semicontinuity is an essential property for the theory of the algorithms presented in chapters 4 and 5, which can be retained for functions for which (2.6) is well-defined and finite, cf. [Bredies and Lorenz, 2011, section 6.3.3]. These functions are so called functions with *bounded variation*. In Figure 2.1 a denoising example is shown with a quadratic regularization (left) and a TV-regularization (right). We can observe that in the denoising result with TV-regularization there are more sharp edges than in the denoising result with quadratic regularization.
All functions considered so far where defined on an open domain $\Omega \subset \mathbb{R}^2$. However, images consist of a finite number of pixels, represented as matrices or long vectors, $f \in \mathbb{R}^{m\times n}$, or $f \in \mathbb{R}^N$, where m, n denotes the number of pixels in two dimensions, while $N = m \cdot n$ is the total number of pixels in the image domain Ω. Using this discrete setting with appropriate discrete adaptations of the gradient and divergence

operator, cf. chapter 6, the total variation representations in Equations (2.6) and (2.7) are equivalent. However, since the continuous notation is more intuitive and easy to read, we keep using it, although Ω is going to be a finite set in what follows.

2.2 Segmentation

The segmentation of an image is the partition of the image domain into pairwise disjoint regions. One considered case is gray values to be similar inside one region. The mean gray value over each region is used to approximate the original image through the segmented regions.

2.2.1 Piecewise Constant Mumford–Shah Model

One of the first variational image models is the model by Mumford and Shah [Mumford and Shah, 1989] for denoising and segmentation.
Let Ω be the two-dimensional image domain. The image domain shall be divided into areas with piecewise constant color / gray value. The more areas the model includes, the more detailed the segmentation. Let $f : \Omega \to \mathbb{R}$ denote the image. The segmentation model with only two areas is given by

$$J(c_1, c_2, C) = \underbrace{\int_\omega (f(x) - c_1)^2 \, \mathrm{d}x + \int_{\Omega\setminus\omega} (f(x) - c_2)^2 \, \mathrm{d}x}_{J_{\text{data}}} + \mu \underbrace{\mathcal{L}(C)}_{J_{\text{reg}}}, \tag{2.8}$$

where $\mathcal{L}(C)$ is the length of the contour dividing the image domain Ω into two areas, ω and $\Omega\setminus\omega$, and c_1 and c_2 are the mean color- / gray-values in the areas. $\mu > 0$ is a weighting parameter. The model consists of the data term, J_{data}, which models the image data, and a regularization term, J_{reg}, which prevents the contour from getting fractal, i.e. prefers short contours.
By minimizing this functional, a segmentation can be found. Since the functional is not convex, the solution is not unique. In some applications not only global, but also local minima of the segmentation functional are interesting. In this case the user has to decide, which solution is best for the application.
In order to find more than two regions the following model by [Mumford and Shah, 1989] is given.

$$J(c_i, C) = \sum_{i=1}^{n} \int_{\omega_i} (f(x) - c_i)^2 \, \mathrm{d}x + \mu \mathcal{L}(C) \tag{2.9}$$

n is the number of areas and c_i the mean color-/gray-value in ω_i.

2.2.2 Segmentation with Level Sets - Chan–Vese

The introduction of a level set function $\varphi : \Omega \to \mathbb{R}$ by Chan–Vese [Chan et al., 2000] simplifies the Mumford-Shah model, since there is only one integral left. The contour $\mathcal{C}$, which divides the different areas is defined by the zero level set $\mathcal{C} = \{x \in \Omega | \ \varphi(x) = 0\}$ of the level set function. For the level set function signed distance functions can be used. By introducing a Heaviside function H and a Dirac measure δ

$$H(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}, \qquad \delta(z) = \frac{\mathrm{d}}{\mathrm{d}z} H(z), \tag{2.10}$$

the functional in (2.8) can be explained by an integral over the whole domain Ω

$$\begin{aligned} J(c_1, c_2, \varphi) &= \int_\Omega (f(x) - c_1)^2 H(\varphi) + (f(x) - c_2)^2 (1 - H(\varphi)) \, \mathrm{d}x \\ &\quad + \mu \int_\Omega \delta(\varphi) \, \|\nabla \varphi\| \, \mathrm{d}x \\ &= \underbrace{\int_\Omega \left((f(x) - c_1)^2 - (f(x) - c_2)^2 \right) H(\varphi) + (f(x) - c_2)^2 \, \mathrm{d}x}_{J_{\text{data}}} \\ &\quad + \mu \underbrace{\int_\Omega \delta(\varphi) \, \|\nabla \varphi\| \, \mathrm{d}x}_{J_{\text{reg}}} \end{aligned} \tag{2.11}$$

with c_1, c_2 are the mean gray values in the areas and $\mu > 0$ is a parameter. The functional is divided into the same two terms as (2.8), J_{data}, the data term and J_{reg}, the regularization of the level set. The regularization term in this case prevents the zero level set from getting fractal.
In order to minimize the functional, typically the Heaviside function $H(z)$ and the Dirac measure $\delta(z)$ are replaced by regularized versions, H_ε and δ_ε, see e.g. [Chan et al., 2000]. The minimization with respect to the mean gray-values and the level set function is alternated until convergence.
The functional (2.11) is not convex in φ, therefore the algorithm may get stuck in a local minimum depending on the initialization.
If one is only interested in a global solution and does not have good initializations, a convex model should be chosen.

2.2.3 Convexification: Labeling Model

Since the functional (2.11) is depending on the level set function φ only indirectly through the Heaviside function H, [Chan et al., 2004] proposed replacing the Heaviside function of the level set function by a labeling function $u : \Omega \to [0, 1]$ that is directly

connected to x. The new functional

$$J(u) = \int_\Omega ((f(x) - c_1)^2 - (f(x) - c_2)^2)u(x) + \mu \|\nabla u(x)\| \, \mathrm{d}x \tag{2.12}$$

is convex in u, which can be seen by analyzing the terms in $J(u)$: the first term is linear in u and thus convex, and the second term is the norm which is also convex in u. Since the sum of convex functions is convex, $J(u)$ is convex in u.
The segmented areas are found by thresholding with a parameter $\vartheta \in [0, 1]$:

$$\begin{aligned} \Omega_1 &= \{x \in \Omega | u(x) \geq \vartheta\} \\ \Omega_2 &= \{x \in \Omega | u(x) < \vartheta\} \end{aligned}$$

The model is also convex in c_1, c_2, but not in the joint variable (u, c_1, c_2). Thus c_1, c_2 are fixed and are calculated separately by minimizing the functional

$$J(c_1, c_2) = \int_\Omega ((f(x) - c_1)^2 - (f(x) - c_2)^2)u(x) + (f(x) - c_2)^2 \, \mathrm{d}x \tag{2.13}$$

for fixed u. A minimizer (u^*, c_1^*, c_2^*) can be computed by alternating minimization with respect to u and c_1, c_2.
Due to convexity, functional (2.12) has a global minimum (not necessarily unique) in u for every c_1, c_2. In [Chan et al., 2004] the authors proved, that for any given c_1, c_2 a global minimizer for (2.8) is given by the minimum of equation (2.12).
Open Problem: An open question is, whether the alternating minimization with respect to c_1, c_2 and u respectively leads to a global minimum.
This convex approach with the labeling function u will from now on be called the *labeling model*.

2.2.4 Multilabel Segmentation

Using the presented models, the Chan–Vese model and the labeling model, only two areas can be separated, $\varphi < 0$ and $\varphi > 0$ or $u \geq \vartheta$ and $u < \vartheta$ respectively. To adapt the model for segmentation with more than two areas, different approaches can be used.
The first idea is to include more level set functions or labeling functions. In [Vese and Chan, 2002] the authors suggested defining the areas by an overlapping of different level set functions, i.e. with two level set functions, φ_1, φ_2, four areas Ω_i, $i = 1, \ldots, 4$ can be segmented:

$$\begin{aligned} \Omega_1 &= \{x \in \Omega | \varphi_1(x) > 0 \wedge \varphi_2(x) > 0\} \\ \Omega_2 &= \{x \in \Omega | \varphi_1(x) < 0 \wedge \varphi_2(x) > 0\} \\ \Omega_3 &= \{x \in \Omega | \varphi_1(x) > 0 \wedge \varphi_2(x) < 0\} \\ \Omega_4 &= \{x \in \Omega | \varphi_1(x) < 0 \wedge \varphi_2(x) < 0\} \end{aligned} \tag{2.14}$$

Thus up to n areas can be defined with a set of only $m = \lceil \log_2(n) \rceil$ level set functions, $\varphi_i : \Omega \to \mathbb{R}$.
This idea can also be adapted to the labeling model (2.12), such that it can be adapted for multi-label segmentation by including multiple labeling functions u.

Convexified Chan–Vese. The multi-label model by Chan–Vese can be adapted to the convex model directly, but one has to find a way to define the areas (2.14), and include them into the model. In [Li et al., 2010], a multi-label model based on this idea is introduced

$$J(u,c) = \sum_{i=1}^{m} \int_\Omega \|\nabla u_i(x)\|_2 \, \mathrm{d}x + \mu \sum_{k=1}^{2^m} \int_\Omega (f(x) - c_k)^2 M_k \, \mathrm{d}x, \tag{2.15}$$

where e.g. for $m = 2$ the M_k are explicitly given by

$$\begin{aligned} M_1 &= u_1 u_2 & M_2 &= u_1(1-u_2) \\ M_3 &= (1-u_1)u_2 & M_4 &= (1-u_1)(1-u_2). \end{aligned}$$

The drawback is, that since the areas are defined by multiplication of the labeling functions u_i the functional is no longer convex in the variable (u_1, u_2).

Second idea. Another idea is to introduce one labeling function $u_i \in [0,1]^{|\Omega|}$ for every label $\Omega_i \subset \Omega$, thus for a partition of Ω into n labels, n labeling functions u_i are needed. Also an additional constraint has to be introduced in order to avoid labeling functions to overlap. This constraint can be realized by the sum of the labeling functions, as in e.g. [Nieuwenhuis et al., 2013]

$$\sum_i u_i(x) = 1. \tag{2.16}$$

Functional lift idea. In several manuscripts, as e.g. [Brown et al., 2012, Brown et al., 2009, Pock et al., 2008], an idea is presented that defines the different labels by lifting the labeling function into a higher dimension. Therefore a super-level set function $\phi : \Omega \times \Gamma \to [0,1]$ is introduced. Since we do not follow this approach, details are omitted here, but to apply this idea also on the motion segmentation problem could be a topic of future studies.

2.3 Optical Flow

We now address the problem of motion estimation, also called optical flow estimation. The optical flow of an image sequence is a vector field on the image domain. The vector field describes for each pixel the motion displacement vector between two or more images. Here, we only take the information from two images to calculate the displacement field.

2.3.1 Aperture Problem and Occlusions

Pixels with the same color/gray-value, or areas with the same texture cannot be matched uniquely to pixels in the consecutive images. This is called the aperture problem, because it occurs if the observed area is too small and not connected to its surroundings. For example, the motion of the roof of a car, which is homogenous in color and texture, can only be estimated by connecting it to the motion of the rest of the car. Thus, if the observed area is too small, the movement cannot be determined uniquely. Therefore additional assumptions to estimate the optical flow are necessary.
Further, during an image sequence some regions may appear or disappear over time, i.e. the region is visible in one image but not in the consecutive image of the sequence. These regions are called occlusions. The pixels in this area cannot be matched to pixels in other images. The optical flow in these regions is undefined. Here, we do not model occlusions. See [Ayvaci et al., 2010] for an optical flow model with occlusions.

2.3.2 Local and Global Methods

Let $f : \Omega \times \mathbb{R} \to \mathbb{R},\ \Omega \in \mathbb{R}^2$ be the image sequence depending on the spatial coordinate $x \in \Omega$ and the time $t \in \mathbb{R}$. The displacement field defining the optical flow is given by the flow field $v : \Omega \times \mathbb{R} \to \mathbb{R}^3$. In order to estimate the vector field $v(x,t)$, it is necessary to make assumptions on constancy over time. A natural choice is the assumption, that the gray-values of objects do not change over time, i.e. the gray-value $f(x,t)$ in a point $x \in \Omega$ and time t stays the same, if it has moved to $x(t+\Delta t) = x(t) + v_{\Delta t}(x)$ in the consecutive image $f(\cdot\ , t+\Delta t)$. This assumption is called *gray value constancy assumption*:

$$f(x,t) = f(x + v_{\Delta t}(x), t + \Delta t), \qquad \forall x \in \Omega,\ t \in \mathbb{R} \tag{2.17}$$

where $v_{\Delta t}(x)$ is the spatial flow vector and Δt is the time step between the images in the sequence.
In order to resolve for $v(x,t)$, the equation is linearized by a first order Taylor expansion of the right hand side

$$f(x + v_{\Delta t}(x), t + \Delta t) = f(x,t) + \langle \nabla f(x,t), v(x,t) \rangle + \mathcal{O}(\|v\|^2). \tag{2.18}$$

Assuming differentiability in Ω, we denote by ∇f the derivative of the image sequence in two spatial and one time direction, $\nabla f = (\partial_{x_1} f, \partial_{x_2} f, \partial_t f)^T$, with $(x_1, x_2) \in \Omega$.
Putting equations (2.17) and (2.18) together, and omitting terms of higher order, one obtains the so called *optical flow constraint (OFC)*

$$\langle \nabla f(x,t), v(x,t) \rangle = 0, \qquad \forall x \in \Omega,\ t \in \mathbb{R}. \tag{2.19}$$

Note, that due to the linearization, the optical flow constraint is only reasonable for small displacements. Further, since occlusions and other changes in the image are not

modeled by the OFC, one should not expect equation (2.19) to hold for the whole image domain and thus, there will always be errors.
On its own the OFC is not sufficient to determine a unique solution. Often, large areas of the image are smooth (low texture) and the image derivative is zero there. Thus, in these areas equation (2.19) is fulfilled for every v, which is the effect of the aperture problem. Consequently, a second assumption is needed, connecting the pixel to its surroundings. We discuss two possibilities, a global and a local assumption.

Global assumption. In the early work of Horn and Schunck [Horn and Schunck, 1981], the authors suggested to introduce a spatial regularizer, and formulate the problem with the regularizer and the optical flow constraint as a variational model. Their regularizer is based on a global smoothness assumption for the estimated flow field penalizing the squared norm of the gradient of the flow field. Their variational model for optical flow estimation reads:

$$J(v) = \int_\Omega \langle \nabla f(x,t), v\rangle^2 + \mu \left(\left\| \nabla (v_x)^{(1)} \right\|_2^2 + \left\| \nabla (v_x)^{(2)} \right\|_2^2 \right) \mathrm{d}x \tag{2.20}$$

where $\mu > 0$ is a regularization parameter, and $(v_x)^{(1)}$ is the first component of the vector field $((v_x)^{(1)}, (v_x)^{(2)})$. The chosen regularization term yields dense, smooth flow fields. It is also possible to choose a TV regularization term, which allows discontinuities in the optical flow field, as done e.g. in [Zach et al., 2007].

Local assumption. In contrary to the Horn–Schunck method, Lucas and Kanade [Lucas et al., 1981] developed a local method. They overcome the aperture problem by assuming, that the optical flow vectors are constant on small parts of the image, i.e. a small part D of the image domain Ω is chosen to compute a constant flow vector v on the selected area D which yields a linear system. Let $x_i \in D$, for indices $i = 1, \dots, k$:

$$\begin{pmatrix} \nabla f(x_1,t)^T \\ \vdots \\ \nabla f(x_k,t)^T \end{pmatrix} \begin{pmatrix} v_{\Delta t} \\ \Delta t \end{pmatrix} = 0. \tag{2.21}$$

If the area D is chosen large enough, such that the matrix consisting of the spatial and temporal image derivatives has full rank, a vector v can be determined for D through least squares. Thus, the optical flow is computed on a local and not a global neighborhood.
In [Bruhn et al., 2005], the authors suggested a combination of local and global methods. In order to model the local neighborhood, a convolution with a Gaussian kernel[1]

[1] A discrete 2-dimensional Gaussian kernel can be computed by $K_\rho(i,j) = \frac{g_{i,j}}{\sum_{i,j} g_{i,j}}$, where $g_{i,j} = e^{-\frac{i^2+j^2}{2\rho^2}}$.

$\mathcal{K}_\rho$ with standard deviation ρ is used together with the regularization from the Horn–Schunck method:

$$\min_v \int_\Omega \mathcal{K}_\rho \star \langle \nabla f(x,t), v\rangle^2 + \mu(\|\nabla v_1\|_2^2 + \|\nabla v_2\|_2^2)\,\mathrm{d}x, \tag{2.22}$$

where $\star$ denotes the convolution. Through a sufficiently large smoothing, or a scaling of the image, also large displacements can be modeled, which are usually not covered by the optical flow constraint.
Since smoothing and scaling can improve the calculated optical flow, and can speed up calculations, in [Bruhn et al., 2006] the authors propose several multigrid schemes. In an earlier article, [Bruhn et al., 2005] used a sophisticated coarse-to-fine strategy, to handle large displacements. Further, a presmoothing step, i.e. a convolution of the input images with a Gaussian kernel, is suggested to remove noise and get better results.

Gradient constancy assumption. Unfortunately, for some sequences, or parts of the sequences, the optical flow constraint is not valid, if for example the gray-values change over time due to illumination changes. Then, further assumptions on the flow field can be made to increase the quality of the optical flow estimation. In case of illumination changes over time, e.g. reflections, the spatial image gradient is assumed to be constant over time.

$$\nabla_x f(x,t) = \nabla_x f(x + v_x, t + \Delta t)$$

This model is discussed in several works, cf. for example [Otte and Nagel, 1995]. Further, higher order data terms can be found in [Bruhn, 2006].

Remark 2.1 (Color Images). For color images different approaches can be used. The above equations can for example be formulated for each color channel resulting in a linear system over all three channels. Here we only assume gray value images.

2.4 Further Imaging Applications

Further applications such as image zooming, deconvolution or inpainting can be modeled through a variational formulation with different regularization terms. The total variation is often used as regularization, but as for the denoising problem at the beginning of this chapter, also other regularizations can be used. Also the gradient operator can be replaced by another operator. For example, the problem of joint inpainting and denoising, where missing parts of an image f should be reconstructed in a natural way, can be modeled with different linear operators: The image domain is given by Ω and the inpainting domain, i.e. the missing part of the image, is given by $\Omega' \subset \Omega$. A possible model is given by

$$\min_u \int_\Omega \|\Phi u(x)\| + \frac{\mu}{2} \int_{\Omega\setminus\Omega'} (u(x) - f(x))^2\,\mathrm{d}x, \tag{2.23}$$

where the operator Φ can be the gradient operator yielding the total variation regularization, or, as suggested in [Chambolle and Pock, 2011], the fast discrete curvelet transform, which according to [Chambolle and Pock, 2011] leads to better results.

3 Models for Motion Segmentation

In this chapter we discuss models for real-time motion segmentation, i.e. the segmentation of moving objects on a moving background. The motion segmentation models are related closely to the segmentation models. Instead of penalizing a deviation of the mean gray value, a penalization for flow vectors is used. In order to achieve real-time performance, on the one hand, the model must be simple such that the calculations are fast and, on the other hand, the optimization algorithm has to converge fast. Different models are presented in this chapter, and we discuss optimization algorithms in the next two chapters.
At first, we analyze models dividing only foreground and background, thus a segmentation of only two areas. One area does not necessarily consist of one connected component. Further, we assume that the movement in each of these two areas can be modeled by a constant motion vector.
Then, we examine different multi-label models, which allow to find a segmentation for multiple objects moving in multiple directions on a moving background.
At last, we discuss an affine motion model as replacement for the constant motion vector.

3.1 Two-Label Model

In the two-label model the image domain should be divided into two areas, a moving object, or multiple objects with similar motion, and the rest of the image with a different motion. These two areas can be seen as foreground and background. In the following, several approaches for the two-label model are presented.

3.1.1 Chan–Vese Model with Optical Flow

The Chan–Vese Model with an optical flow data term inside the segmentation model was proposed in [Cremers and Soatto, 2005] and [Brox et al., 2006]. Inside the areas ω_i a constant flow v_i is assumed.
In [Cremers and Soatto, 2005], the authors suggest calculating the flow vectors as follows. The data term used in their functional for every area $\omega_i \subset \Omega$ is

$$J_{\omega_i}(v_i) = \int_{\omega_i} \frac{\langle v_i, T(x,t)v_i \rangle_2}{\|v_i\|_2^2} \, \mathrm{d}x, \quad \text{with} \quad T(x,t) = \frac{\nabla f(x,t) \nabla f(x,t)^T}{\|\nabla f(x,t)\|_2^2}. \tag{3.1}$$

These data terms are a normalized version of the OFC. The motion segmentation model with TV-regularization then reads

$$J(v_1, v_2, \varphi) = \int_\Omega \frac{\langle v_1, T(x,t)v_1\rangle_2}{\|v_1\|_2^2} H(\varphi(x)) + \frac{\langle v_2, T(x,t)v_2\rangle_2}{\|v_2\|_2^2} \Big(1 - H\big(\varphi(x)\big)\Big)\, \mathrm{d}x \\ + \mu \int_\Omega \delta(\varphi(x))\, \|\nabla\varphi(x)\|_2\, \mathrm{d}x. \tag{3.2}$$

Here φ is the level set function and H is the Heaviside function.
The model given in [Brox et al., 2006], is much more complex. It is also based on level set functions, but in addition to the OFC, a gradient constancy assumption is included into the model with good results. Their results were accurate but the proposed algorithm was not fast.
Both models use multiple level set functions to model more than two areas. Further, both models are, as the level-set model for segmentation, not convex. Thus, the solution depends on the initialization and will probably be a local and not a global minimum.

3.1.2 Labeling Model with Optical Flow

We choose a convex labeling model for motion segmentation, as done in [Tichmann and Junge, 2014]. Starting with a two-label model, the labeling function $u : \Omega \to [0,1]$ is introduced. As the computations should be fast, a simple model is chosen, and thus, we choose a squared optical flow constraint as data term. For the regularization term, we choose a total variation regularization since we want to allow sharp edges. The resulting minimization problem reads

$$\begin{aligned} &\min_{u,v_1,v_2} \quad \int_\Omega \|\nabla u(x)\|_2 + \mu\Big(\langle \nabla f(x,t), v_1\rangle^2 u(x) + \langle \nabla f(x,t), v_2\rangle^2 \big(1 - u(x)\big)\Big)\, \mathrm{d}x \\ &\text{subject to} \quad u(x) \in [0,1] \end{aligned} \tag{3.3}$$

where $f(x,t)$ denotes the image sequence at a fixed time t and $\mu > 0$ is a weighting parameter. v_1 and v_2 are the constant motion vectors for the regions $\omega = \{x \in \Omega | u(x) > \vartheta\}$ and $\Omega\backslash\omega = \{x \in \Omega | u(x) \leq \vartheta\}$, where $\vartheta \in (0,1)$ is a threshold usually chosen as $\vartheta = 0.5$.
The functional is convex in u and in v_1, v_2, but not in the joint variable (u, v_1, v_2). The nonconvexity can be seen on the following example: if (u_*, v_1, v_2) is a solution to (3.3), then $(1 - u_*, v_2, v_1)$ is also a solution. For fixed (v_1, v_2) a global solution exists with respect to u, and vice versa, for fixed u, a unique solution with respect to (v_1, v_2). Typically, the variables are minimized alternatingly. If a good segmentation u is already given, the motion vectors can be calculated explicitly by minimizing over v_1 and v_2. Otherwise, the segmentation u is calculated by minimization with respect to u and afterwards u is fixed and the functional is minimized with respect to v_1, v_2. The algorithms are explicitly given in chapter 6.

Remark 3.1. If the test sequence has big illumination changes, the gradient constancy assumption can be included into this model, as done for the level set model in [Brox et al., 2006].

3.2 Multi-Label Model

In contrast to the two-label model, which can only handle one moving object, the multi-label model should be able to model multiple objects with different motion and a moving background.

3.2.1 Multiple Labeling Functions

Usually, the two-label models are adapted straightforward to a multi-label model, using techniques from multi-label segmentation by introducing multiple labeling functions u_i. These models tend to need some a priori knowledge about the number of labels.
The adaption of the Chan–Vese multiphase model has been proposed in e.g. [Cremers and Soatto, 2005] and [Brox et al., 2006]. A convex multi-label model has been proposed in [Unger et al., 2012]. However, their algorithm needs good initializations to be reasonably fast.
It turns out that for a direct adaption from two-label to multi-label motion segmentation, the initialization is a crucial point. With good initializations, which means an initialization which is close to the desired solution, the algorithms might work well. Without a good initialization, the alternating minimization between the flow vectors v_i and the labeling functions u_i turns out to slow down convergence. This is due to the fact that most multi-label models are only convex in the single labeling function u_i and not in all labeling functions simultaneously.
In the following, we review some multi-label models with different methods of handling the regions to be segmented.

Introducing multiple labeling functions. We start with the multi-label model, which models every region by a labeling function. The resulting optical flow errors and regularizations are summed in the objective functional.

$$\min_{u_i, v_i} \int_\Omega \sum_i \left(\|\nabla u_i(x)\|_2 + \mu \langle \nabla f(x,t), v_i \rangle^2 u_i(x) \right) \mathrm{d}x \tag{3.4}$$

$$\text{subject to} \quad u_i(x) \in [0,1] \tag{3.5}$$

Here, a constraint or term has to be added to assure that the labels cover the whole image domain Ω and do not overlap. Different approaches are possible and presented in the following.

Convex Chan–Vese adaption. In [Li et al., 2010], the authors propose a multi-label model for segmentation inspired by the Chan–Vese level-set model from [Vese and Chan, 2002]. As for the segmentation multi-label model m labeling functions are introduced for 2^m areas, and we include the optical flow data term:

$$J(u,v) = \sum_{i=1}^{m} \int_\Omega \|\nabla u_i(x)\|_2 \, \mathrm{d}x + \mu \sum_{k=1}^{2^m} \int_\Omega \langle \nabla f(x,t), v_i \rangle^2 \, M_k \, \mathrm{d}x, \tag{3.6}$$

where e.g. for $m = 2$ the M_k are explicitly given by

$$\begin{aligned} M_1 &= u_1 u_2 & M_2 &= u_1(1-u_2) \\ M_3 &= (1-u_1)u_2 & M_4 &= (1-u_1)(1-u_2). \end{aligned}$$

This model is nonconvex in (u_1, u_2), and in our experiments, it did not work with the motion segmentation data term.

Labeling functions sum up to 1. Another possibility is to add the constraint that the sum of all labels equals one as done for example in [Unger et al., 2012]. The minimization problem is given by

$$\begin{aligned} &\min_{u_i, v_i} \int_\Omega \sum_i \left(\|\nabla u_i(x)\|_2 + \mu \, \langle \nabla f(x,t), v_i \rangle^2 \, u_i(x) \right) \, \mathrm{d}x, \\ &\text{subject to} \quad u_i(x) \geq 0, \, \sum_i u_i(x) = 1. \end{aligned} \tag{3.7}$$

This enforces on the one hand, that the whole image domain is covered, but on the other hand, since u can take values in the interval $[0,1]$, does not necessarily enforce that the labeling functions do not overlap.

Prevent overlapping of u_i. The background can also be calculated through the other labels via $(1 - \sum_i u_i)$, as done in the two-label model, with an additional vector v_0 for the background:

$$\begin{aligned} &\min_{u_i, v_i} \int_\Omega \sum_i \Big(\|\nabla u_i(x)\|_2 + \mu \, \langle \nabla f(x,t), v_i \rangle^2 \, u_i(x) \Big) + \mu \, \langle \nabla f(x,t), v_0 \rangle^2 \left(1 - \sum_i u_i(x) \right) \mathrm{d}x, \\ &\text{subject to} \quad u_i(x) \in [0,1] \end{aligned} \tag{3.8}$$

Unfortunately, this also does not prevent the labeling functions u_i from overlapping. Thus, we propose to add an additional term penalizing overlapping labels through

pairwise multiplication of the labeling functions u_i:

$$\begin{aligned}\min_{u_i,v_i} \int_\Omega \sum_i \Big(\|\nabla u_i(x)\|_2 + \mu \left\langle \nabla f(x,t), v_i \right\rangle^2 u_i(x) \Big) \, \mathrm{d}x \\ + \int_\Omega \mu \left\langle \nabla f(x,t), v_0 \right\rangle^2 \Big(1 - \sum_i u_i(x) \Big) + \sum_{i,j,\, i \neq j} u_i(x) u_j(x) \, \mathrm{d}x, \\ \text{subject to} \quad u_i(x) \in [0,1]\end{aligned} \tag{3.9}$$

As with the Chan–Vese like adaption from [Li et al., 2010], this model is only convex in u_i, but not in the joint variable $u = (u_1, u_2, \dots)$.
For computation, the labels u_i and v_i are updated alternatingly, i.e. for a given set of motion vectors v_i the labels u_i are updated one by one and afterwards the v_i are updated. We observe, that during this alternating minimization the labels u_i do not converge simultaneously but sequentially, if at all. Thus, the labels can be calculated consecutively instead of simultaneously, i.e. we start with the two label model and calculate a vector v_1, a background vector v_0 and a labeling function u_1. When the first label u_1 has converged, u_1 is fixed and a second label u_2 together with a motion vector v_2 is introduced and so on. The advantage of this scheme is that the number of labels does not have to be known a priori. The drawback is, that the computation time rises at least linearly with the number of labels.

3.2.2 Error-Label Function

We propose a multi-label model directly derived from the two label model. We do not introduce multiple labeling functions and vectors, but reduce the two label model even further, and only calculate one vector. The second error term, given by $\langle \nabla f, v_2 \rangle$ in the two-label model, is replaced by a constant parameter $\xi > 0$. The model reads as follows:

$$\begin{aligned}\min_{u,v} \int \|\nabla u(x)\|_2 + \mu \Big(\xi u(x) + \left\langle \nabla f(x,t), v \right\rangle^2 \left(1 - u(x) \right) \Big) \, \mathrm{d}x, \\ \text{subject to} \quad u(x) \in [0,1].\end{aligned} \tag{3.10}$$

One segmented area corresponds to vector v. Every part of the image domain, for which the vector v does not fit, i.e. areas in which the error $\langle \nabla f, v \rangle^2$ is larger than ξ will belong to the second area.

Remark 3.2. A similar technique is also used in the task of classification, where a so called rejection class is introduced, where all image elements, which do not fulfill the criteria of the classification are put, cf. [Steger et al., 2007].

The areas of the image, which do not belong to the vector v, i.e. which are part of the error-label, can include multiple moving objects, and are possibly not connected. These areas have to be analyzed in a post processing step, in order to possibly find multiple

regions and corresponding motion vectors. In this post processing step, the resulting error-label is decomposed into connected components, and constant vectors for each connected component are calculated. The calculation of a constant vector for a given region can be realized efficiently.

This model is advantageous in different aspects but also has some drawbacks. First, the minimization is independent of the number of moving objects. Also, it is almost as fast as the two-label model. A drawback is, that a new parameter, ξ, has been introduced, which has to be chosen appropriately. Further, objects moving close together in different directions will be represented in the error-label through one connected component. Thus, theses areas are not separated in the post processing step, and have to be separated in further post processing steps with a different approach.

3.3 Affine Motion Model

Instead of assuming a piecewise constant movement, i.e. a constant vector for each label, an affine model can be used. The advantage of the affine model is, that also zooming or rotating motions can be captured. Affine motion models can be added easily into the motion segmentation models, as done in [Cremers and Soatto, 2005] or [Unger et al., 2012].
We use a similar affine model: For each region the motion vector v inside the region is defined via the matrix $H \in \mathbb{R}^{3\times 2}$:

$$v = Hx = \begin{pmatrix} h_1 & h_3 & h_5 \\ h_2 & h_4 & h_6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}, \tag{3.11}$$

where x_1 and x_2 are the two image coordinates. Now instead of two parameters for each motion vector, the six parameters h_i have to be calculated.
The resulting two-label minimization problem reads as follows:

$$\begin{aligned} &\min_{u,H_1,H_2} \int_\Omega \|\nabla u(x)\|_2 + \mu \left(\langle \nabla f(x,t), H_1 x \rangle^2 u(x) + \langle \nabla f(x,t), H_2 x \rangle^2 \left(1 - u(x)\right) \right) \, \mathrm{d}x, \\ &\text{subject to} \quad u(x) \in [0,1]. \end{aligned} \tag{3.12}$$

Depending on the application, it is also possible to add an affine motion model only for the background and use piecewise constant models for the foreground motion.
If we include the affine motion model only into one of the two segmented areas, we have to include an additional assumption. Here, we want to apply the affine model to the background and we include the additional assumption that the background is always the largest area in the image. The two areas are defined by $\omega = \{x \in \Omega | u(x) >$

$0.5\}$ and $\Omega\backslash\omega = \{x \in \Omega | u(x) \leq 0.5\}$. Since $u(x) \in [0,1]$, these terms are included in the functional via multiplication of the associated error terms with u and $(1-u)$. The background should be modeled by $\Omega\backslash\omega = \{x \in \Omega | u(x) \leq 0.5\}$, thus, the affine motion model has to be included with the term $(1-u)$. In order to guarantee that the background is calculated with the affine model, we include a term penalizing $\int_\Omega u(x)\,\mathrm{d}x$.

$$J(u) = \int_\Omega \|\nabla u(x)\|_2 + \mu \left(\langle \nabla f(x,t), v \rangle^2 u(x) + \langle \nabla f(x,t), Hx \rangle^2 \left(1 - u(x)\right) \right) + \nu u(x)\,\mathrm{d}x, \tag{3.13}$$

where $\nu > 0$ is a parameter.
The error label model can also be combined with the affine motion model for affine background motion or, in the post processing step, for the connected components from the error label.

$$\begin{aligned} &\min_{u,H} \int \|\nabla u(x)\|_2 + \mu \left(\xi u(x) + \langle \nabla f(x,t), Hx \rangle^2 \left(1 - u(x)\right) \right)\,\mathrm{d}x, \\ &\text{subject to} \quad u(x) \in [0,1]. \end{aligned} \tag{3.14}$$

The given models for two-label and multi-label motion segmentation can be solved through established methods for imaging problems, as e.g. the primal-dual algorithm or the alternating direction method of multipliers. However, the performance of these algorithms is not fast enough for a real-time application. In the next two chapters, we analyze properties of existing algorithms for convex constrained problems of the form usually arising from imaging models, and present a new algorithm.

4 Optimization Algorithms for Convex Problems

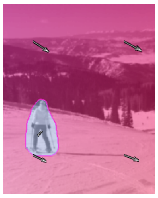
Optimization problems arising from variational imaging models are nowadays sought to be convex in order to derive a global solution, such as the models presented in chapter 3. In this chapter, we analyze methods based on the theory of monotone operators associated with convex functionals. In the first part, we present some fundamental methods such as the proximal point method and the proximal gradient method, of which well-established methods are often special cases. We analyze, which requirements on the objective functionals are essential for the existence of a minimizer and how the minimizer can be related to a fixed point of an operator. Further, the theory providing convergence results for the iteration with averaged operators is given. In the second part, the handling of convex set constraints and equality constraints in optimization problems is given. Augmented Lagrangian functions and the concept of duality is therefor presented. In the third part, composite functionals are analyzed, which are either transformed into a primal-dual formulation, or through a splitting of the variables, into an augmented Lagrangian formulation. These transformed problems are solved with a primal-dual method, and the alternating direction method of multipliers (ADMM), which are both special cases of the proximal point method. In the last part, we analyze two variants of these methods—which will turn out to be equivalent—in order to achieve a better understanding and intuition on how the algorithms work.

4.1 Minimization of Convex Functionals

We start with some basic definitions, which can be found in detail in [Bauschke and Combettes, 2011].

Definition 4.1 (Convex proper closed functional)**.**

1. A functional $F : \mathcal{H} \to (-\infty, \infty]$ on a real Hilbert space $\mathcal{H}$, with scalar product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, and induced norm $\|\cdot\|_{\mathcal{H}}$, is called *convex* if for all $x, y \in \mathcal{H}$ and $\lambda \in [0, 1]$
$$F(\lambda x + (1-\lambda)y) \leq \lambda F(x) + (1-\lambda)F(y). \tag{4.1}$$
It is called *strictly convex* if for all $x, y \in \mathcal{H}$, $x \neq y$ and $\lambda \in (0, 1)$
$$F(\lambda x + (1-\lambda)y) < \lambda F(x) + (1-\lambda)F(y). \tag{4.2}$$



2. The effective domain of F is

$$\operatorname{dom} F = \{x \in \mathcal{H} | F(x) < \infty\}. \tag{4.3}$$

   A functional F is called *proper* if its effective domain is not empty, $\operatorname{dom} F \neq \emptyset$.

3. The functional F is *closed*, if its *epigraph*

$$\operatorname{epi} F = \{(x,t) \in \mathcal{H} \times \mathbb{R} | F(x) \leq t\} \tag{4.4}$$

   is a closed set. Further, if the epigraph is a closed convex nonempty set, the functional is proper convex and closed.

4. The functional F is *lower semicontinuous* at $x \in \mathcal{H}$ if, for every sequence $x_n \to x$, it holds that $F(x) \leq \liminf F(x_n)$.

Lemma 4.2. *A proper convex function is closed, if and only if it is lower semicontinuous.*

For the convex proper and lower semicontinuous function $F(x)$ we want to solve the problem

$$\min_x F(x). \tag{4.5}$$

First, we analyze which assumptions are needed for the existence of a minimizer in problem (4.5).

Definition 4.3 (Coercive function)**.** The function $F : \mathcal{H} \to (-\infty, \infty]$ on the Hilbert space $\mathcal{H}$ is *coercive*, if

$$\lim_{\|x\| \to \infty} F(x) = \infty. \tag{4.6}$$

Using the above definitions, we can formulate the theorem on the existence of a minimizer.

Theorem 4.4 ([Bauschke and Combettes, 2011, prop. 11.14])**.** *Let $\mathcal{H}$ be a real Hilbert space, $F : \mathcal{H} \to (-\infty, \infty]$ proper convex and lower semicontinuous, and let $\mathcal{C}$ be a closed convex subset of $\mathcal{H}$ such that $\mathcal{C} \cap \operatorname{dom} F \neq \emptyset$. Suppose that one of the following holds:*

1. *F is coercive.*
2. *$\mathcal{C}$ is bounded.*

Then F has a minimizer over $\mathcal{C}$.

With this theorem the existence of a not necessarily unique minimizer can be shown. Next, we analyze optimality conditions for a minimizer of (4.5).

The functional F does not need to be differentiable. If the functional F is not differentiable a generalization of the gradient is used, the subgradient:

Definition 4.5 (Subgradient, Subdifferential)**.** A *subgradient* $w \in \mathcal{H}$ at x of a convex functional $F : \mathcal{H} \to (-\infty, \infty]$ is defined by

$$F(x) + \langle w, y - x \rangle \leq F(y) \quad \forall y \in \mathcal{H}. \tag{4.7}$$

The *subdifferential* of F at x is defined by the set of all subgradients in x.

$$\partial F(x) = \{w \in \mathcal{H} \mid F(x) + \langle w, y - x \rangle \leq F(y), \forall y \in \mathcal{H}\} \tag{4.8}$$

We assume that the functional F is subdifferentiable, i.e. the subdifferential is not empty, $\partial F(x) \neq \emptyset$.
A simple characterization of the minimizers of a proper functional is provided by Fermat's Rule, cf. [Bauschke and Combettes, 2011, chapter 16]:

Theorem 4.6 (Optimality condition, Fermat's rule)**.** *$x^* \in \mathcal{H}$ minimizes the convex proper functional $F : \mathcal{H} \to (-\infty, \infty)$, if and only if $0 \in \partial F(x^*)$.*

Proof. $x^* \in \mathcal{H}$ is a minimizer of $F(x)$ if and only if $F(x^*) \leq F(x), \forall x \in \mathcal{H}$.
Thus $F(x^*) \leq F(x) + \langle 0, x - x^* \rangle$, $\forall x \in \mathcal{H}$, which is by Definition 4.5 equivalent to $0 \in \partial F(x^*)$. □

Remark 4.7. The subdifferential is defined in general also for non-convex functionals, but we only use convex functionals here.

Now, if a minimizer exists we want to find one. We seek to solve these optimization problems through iterative methods which we analyze in the following.

4.1.1 Monotone Operators

In order to find minimizers we use iterative methods where the iteration is defined by the application of operators. The iteration with an appropriate operator should converge to a fixed point of the operator and its fixed point should be a minimizer of the original problem. We therefore analyze which operators fulfill these conditions.
As is known, iterations with contractions, i.e. operators with Lipschitz constant smaller than 1, converge to a fixed point, cf. the Banach fixed point theorem from [Banach, 1922]. But also operators that are not contractive but nonexpansive can be useful for iterations. Properties for nonexpansive operators are analyzed in the following definitions and theorems, which can be found in [Bauschke and Combettes, 2011].

Definition 4.8 (Nonexpansive operator)**.** Let $\mathcal{K}$ be a nonempty subset of $\mathcal{H}$. An operator $T : \mathcal{K} \to \mathcal{H}$ is called nonexpansive if the following inequality holds:

$$\|Tx - Ty\| \leq \|x - y\| \quad \forall x, y \in \mathcal{K}. \tag{4.9}$$

Iterations with nonexpansive operators do not necessarily converge to a fixed point of the operator such as for example with rotations or reflections. However, iterations with a damping of a nonexpansive operator, called averaged operator, converge to a fixed point.

Definition 4.9 (Averaged operator)**.** Let $\mathcal{K}$ be a nonempty subset of $\mathcal{H}$ and $\alpha \in (0,1)$. An operator $T : \mathcal{K} \to \mathcal{H}$ is called α-averaged, if there exists a nonexpansive operator $N : \mathcal{K} \to \mathcal{H}$ such that

$$T = (1-\alpha)I + \alpha N, \tag{4.10}$$

where I is the identity operator.

The averaged operator T has the same fixed points as N. The next theorem states the convergence of an iteration with an averaged operator T to a fixed point.

Theorem 4.10 ([Bauschke and Combettes, 2011, section 5.2])**.** *Let an α-averaged operator $T : \mathcal{K} \to \mathcal{K}$, and $x_0 \in \mathcal{K}$ be given. Set*

$$x_{k+1} := T(x_k), \quad \forall k \in \mathbb{N}. \tag{4.11}$$

Then, $(x_k)_{k\in\mathbb{N}}$ converges weakly to a fixed point of T, and $(Tx_k - x_k)_{k\in\mathbb{N}}$ converges strongly to 0.

Therefore, iterations with nonexpansive operators can be shown to converge to a fixed point if they can be formulated as an averaged operator.

Remark 4.11. In our setting the real Hilbert space $\mathcal{H} = \mathbb{R}^n$ is finite dimensional, and thus weak convergence implies strong convergence.

Theorem 4.12 ([Parikh and Boyd, 2013])**.** *The class of averaged operators is closed under composition.*

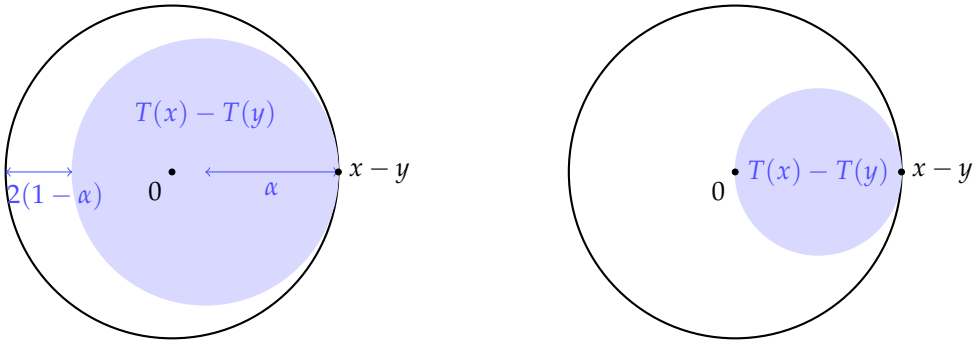
Thus, given the averaged operators T and P, iterations of the composition $T \circ P$ will converge to a fixed point of $T \circ P$.

Remark 4.13. Clearly, contractions are a subset of the class of averaged operators and thus a composition of an averaged operator with a contraction will, with theorem 4.10 also converge to a fixed point.

A special class of averaged operators are firmly nonexpansive operators. Firm nonexpansiveness is a stronger property than nonexpansiveness and can be written as follows:

Definition 4.14 (Firmly nonexpansive operator)**.** An operator $T : \mathcal{H} \to \mathcal{H}$ is called *firmly nonexpansive* if

$$\|Tx - Ty\|^2 \leq \langle x - y, Tx - Ty \rangle \tag{4.12}$$



α-averaged operator T, $\alpha = 0.8$ firmly nonexpansive operator T, with $\alpha = 0.5$

Figure 4.1: Illustration of the effect of averaged operators and firmly nonexpansive operators inspired by [Eckstein and Bertsekas, 1992]. If, without loss of generality, $\|x - y\| = 1$, and $x - y$ is on the unit circle (shown in black), then $T(x) - T(y)$ will be an element of the inner blue disk including its boundary, which has radius α and is shifted to $x - y$.

The connection between firmly nonexpansive operators and averaged operators is given by the following theorem from [Bauschke and Combettes, 2011, Chapter 4]:

Theorem 4.15. *An operator $T : \mathcal{K} \to \mathcal{H}$ is α-averaged with $\alpha \in (0, \frac{1}{2}]$, if and only if T is firmly nonexpansive.*

In Figure 4.1 the effect of an α-averaged operator in $\mathbb{R}^2$ with $\alpha = 0.8$ and a firmly nonexpansive operator is illustrated. Applying a nonexpansive operator N, $N(x) - N(y)$ will be an element of the large disk including the black boundary. But, by applying an α-averaged operator T, $T(x) - T(y)$ will be an element of the inner blue disk, which has radius α, or, for a firmly nonexpansive operator, radius $\alpha \leq 0.5$. The blue disk is touching the black boundary only at the point $x - y$. Therefore, $\|x - y\|$ will decrease by applying an averaged operator except if it does not change at all. This supports the intuition why iteration with α-averaged operators converge.

Remark 4.16. Note that nonexpansive operators are necessarily single-valued.

One type of firmly nonexpansive operators, particularly interesting in our case are proximal operators, also called proximal mappings, cf. [Parikh and Boyd, 2013].

Definition 4.17 (Proximal Operator)**.** The *proximal operator* of a lower semicontinuous proper convex functional $F : \mathcal{H} \to (-\infty, \infty]$ is defined by

$$\operatorname{prox}_{\tau F}(y) = \operatorname*{argmin}_{x \in \mathcal{H}} \left\{ F(x) + \frac{1}{2\tau} \|x - y\|_{\mathcal{H}}^2 \right\}, \tag{4.13}$$

with $\tau > 0, y \in \mathcal{H}$.

The next propositions provides a rule for computing subdifferentials of transformations of convex functions, in particular, when linearity and the chain rule can be applied.

Definition 4.18 ([Bauschke and Combettes, 2011]). Let $\mathcal{C}$ be a convex subset of $\mathcal{H}$.

1. The smallest closed linear subspace of $\mathcal{H}$ containing $\mathcal{C}$ is denoted by $\overline{\text{span}}(\mathcal{C})$.
2. The *conical hull* of $\mathcal{C}$ is the smallest cone in $\mathcal{H}$ containing $\mathcal{C}$, denoted by $\text{cone}(\mathcal{C})$.
3. The *interior* of $\mathcal{C}$ is the largest open set that is contained in $\mathcal{C}$, denoted by $\text{int}(\mathcal{C})$.
4. The *strong relative interior* of $\mathcal{C}$ is given by

$$\text{sri}(\mathcal{C}) = \{x \in \mathcal{C} \mid \text{cone}(\mathcal{C} - x) = \overline{\text{span}}(\mathcal{C} - x)\}, \tag{4.14}$$

where $A - B := \{x - y | x \in A, y \in B\}$ for two sets A and B.

Proposition 4.19 ([Bauschke and Combettes, 2011, Section 16.4]). *Let $\mathcal{X}, \mathcal{Y}$ be real Hilbert spaces, let $F : \mathcal{X} \to (-\infty, \infty]$ and $G : \mathcal{Y} \to (-\infty, \infty]$ be convex, proper and lower semicontinuous, and let $K : \mathcal{X} \to \mathcal{Y}$ be a bounded linear operator. If $0 \in \text{sri}(\text{dom}\, G - K(\text{dom}\, F))$, then*

$$\partial(F + G \circ K) = \partial F + K^* \circ (\partial G) \circ K, \tag{4.15}$$

where K^* is the adjoint of K.

The following result provides a more intuitive way to handle the strong relative interior condition in terms of the interior of $\text{dom}(G) - K\,\text{dom}(F)$:

Proposition 4.20 ([Bauschke and Combettes, 2011, Proposition 6.19]). *Let $\text{dom}(F) \subset \mathcal{X}$ and $\text{dom}(G) \subset \mathcal{Y}$ be convex sets, and $K : \mathcal{X} \to \mathcal{Y}$ be a bounded linear operator. Then*

$$0 \in \text{int}(\text{dom}\, G - K(\text{dom}\, F)) \quad \Rightarrow \quad 0 \in \text{sri}(\text{dom}\, G - K(\text{dom}\, F)). \tag{4.16}$$

Especially, if $\text{dom}(F) \neq \emptyset$, i.e. F is proper, and $\text{dom}(G) = \mathcal{Y}$, then the requirements of Proposition 4.19 are fulfilled.

Definition 4.21 (Resolvent). Let a possibly multivalued operator $S : \mathcal{H} \to \mathcal{H}$ be given. The mapping

$$(I + \tau S)^{-1} : y \mapsto x \tag{4.17}$$

where x is a solution of $y \in (I + \tau S)(x)$ is called the *resolvent* of S with parameter $\tau > 0$. Note, that the resolvent might not exist for every combination of S and τ.

With the following theorems from [Eckstein and Bertsekas, 1992], it can be seen that the proximal operator is firmly nonexpansive.

Definition 4.22. An operator $S : \mathcal{H} \to \mathcal{H}$ is monotone if

$$\langle x_1 - x_2, y_1 - y_2 \rangle \geq 0 \quad \forall x_1, x_2 \in \mathcal{H},\ y_1 \in Sx_1,\ y_2 \in Sx_2. \tag{4.18}$$

Theorem 4.23. *Let a possibly multivalued operator $S : \mathcal{H} \to \mathcal{H}$ and $\tau > 0$ be given. Then S is monotone if and only if the resolvent $(I + \tau S)^{-1}$ is single-valued and firmly nonexpansive.*

Theorem 4.24. *If F is proper convex and lower semicontinuous, its subdifferential ∂F is monotone.*

From these theorems we can see, that for a proper convex and lower semicontinuous function F the resolvent operator $(I + \tau\partial F)^{-1}$ is single-valued and firmly nonexpansive. Further, it can be shown, that the proximal operator is equivalent to the resolvent of ∂F,

$$\text{prox}_{\tau F} = (I + \tau\partial F)^{-1}, \tag{4.19}$$

which can be seen directly by applying Fermat's rule to (4.13). Thus we can formulate the following corollary:

Corollary 4.25. *If F is proper convex and lower semicontinuous, the proximal operator $\text{prox}_{\tau F}$ is firmly nonexpansive.*

Since the proximal operator is firmly nonexpansive it is an averaged operator, with Theorem 4.15.
A connection between the proximal operator and the minimizer of the functional F associated with the proximal operator is given by the following theorem:

Theorem 4.26 (Fixed Point, [Parikh and Boyd, 2013])**.** *The point x^* minimizes F if and only if it is a fixed point of $\text{prox}_{\tau F}$, i.e. $x^* = \text{prox}_{\tau F}(x^*)$.*

From Theorems 4.10 and 4.26, we can see that the iteration with the proximal operator will converge to a fixed point, hence to a minimizer of F.

Corollary 4.27. *Let F be convex proper and lower semicontinuous. The iteration*

$$x_{k+1} := \text{prox}_{\tau F}(x_k) \tag{4.20}$$

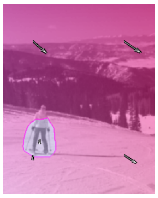
with arbitrary x_0 will converge for $k \to \infty$ to a minimizer of F, if one exists.

This corollary immediately suggests the *proximal point algorithm* (cf. [Rockafellar, 1976]), which is given by

Algorithm 1 proximal point algorithm, [Rockafellar, 1976]

1. choose a step size $\tau > 0$ and an initial point $x_0 \in \mathcal{X}$
2. iterate for $k = 0, 1, \ldots$

$$x_{k+1} := \text{prox}_{\tau F}(x_k). \tag{4.21}$$



The proximal point algorithm can be interpreted as an implicit gradient descent step with step size τ. By definition of the proximal operator in 4.17, the iterates are given by

$$x_{k+1} = \text{prox}_{\tau F}(x_k) = \underset{x \in \mathcal{H}}{\text{argmin}} \left\{ F(x) + \frac{1}{2\tau} \|x - x_k\|_{\mathcal{H}}^2 \right\}. \tag{4.22}$$

The necessary and sufficient condition for x_{k+1} to minimize the strongly convex function on the right hand side of (4.22) is:

$$0 \in \tau \partial F(x_{k+1}) + x_{k+1} - x_k \tag{4.23}$$

Since the resolvent operator is single valued and thus x_{k+1} is unique, this is equivalent to the implicit gradient step

$$x_{k+1} = x_k - \tau \partial F(x_{k+1}). \tag{4.24}$$

We call this update implicit, since the subgradient is evaluated at the point x_{k+1} instead of x_k as for an explicit update step.

4.1.2 Convex Composite Functionals $F(x) + G(x)$

In this section we consider composite functionals, where the proximal operator of functional J is hard to compute, but the objective functional can be split into F and G, $J(x) = F(x) + G(x)$. It might be more easy to handle F or G separately.
We consider the problem where the objective functional can be split into two functionals, one of which is differentiable:

$$\min_x F(x) + G(x), \tag{4.25}$$

where $F : \mathcal{H} \to (-\infty, \infty]$, $G : \mathcal{H} \to \mathbb{R}$, F convex proper and lower semicontinuous, and G is convex and differentiable.
The optimality conditions are given by the following theorem from [Bauschke and Combettes, 2011, section 26.1]:

Theorem 4.28 (optimality conditions)**.** *Let F and G be defined as in problem (4.25) and $0 \in \text{sri}(\text{dom}\, G - \text{dom}\, F)$. Then*

$$x_* \text{ is a solution to problem (4.25)} \quad \Leftrightarrow \quad 0 \in \partial F(x_*) + \nabla G(x_*). \tag{4.26}$$

These optimality conditions can be derived from Fermat's rule, Theorem 4.6, and Proposition 4.19.

Proximal gradient method. An algorithm handling the functionals F and G separately is the *proximal gradient method*, also called *forward-backward splitting*.

Algorithm 2 proximal gradient method, cf. [Parikh and Boyd, 2013]

1. choose a step size $\tau > 0$ and an initial point $x_0 \in \mathcal{X}$
2. iterate for $k = 0, 1, \ldots$

$$x_{k+1} := \text{prox}_{\tau F}(x_k - \tau \nabla G(x_k)), \tag{4.27}$$

The splitting of the objective function into differentiable and non-differentiable terms is not unique, thus different splittings lead to different algorithms. The proximal gradient method is a combination of an implicit and explicit gradient step, therefore it is also known as forward-backward splitting.
As described in [Parikh and Boyd, 2013], the proximal gradient method can be used as a fixed point iteration to find a minimizer of the objective functional:

Theorem 4.29 ([Parikh and Boyd, 2013, Chapter 4.2.1]). *A point x_* minimizes $F(x) + G(x)$ if and only if*

$$x_* = \text{prox}_{\tau F}\,(x_* - \tau \nabla G(x_*)), \tag{4.28}$$

with $\tau > 0$.

Proof. The proof analyzes and rearranges the optimality condition to derive a proximal formulation through the resolvents of F and G:

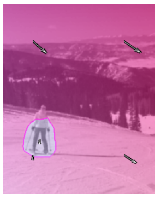
$$0 \in \partial F(x_*) + \nabla G(x_*) \tag{4.29}$$
$$0 \in \tau \partial F(x_*) - x_* + x_* + \tau \nabla G(x_*) \tag{4.30}$$
$$(I + \tau \partial F)(x_*) \ni (I - \tau \nabla G)(x_*) \tag{4.31}$$

Since the resolvent operator of F is single valued, the following equations can be derived:

$$x_* = (I + \tau \partial F)^{-1}(I - \tau \nabla G)(x_*) \tag{4.32}$$
$$= \text{prox}_{\tau F}\,(x_* - \tau \nabla G(x_*)) \tag{4.33}$$

□

If ∇G is Lipschitz continuous with constant L, this method is guaranteed to converge for step sizes $\tau \in (0, 2/L)$, see also [Parikh and Boyd, 2013] for further details.



Proximal Newton-type method. The proximal gradient method is a first order method. For composite functions with a smooth part, [Lee et al., 2012] proposed a proximal Newton-type method, which also uses second order information on the smooth part. Consider the problem

$$\min_x F(x) + G(x), \tag{4.34}$$

which has a smooth part $G : \mathcal{X} \to \mathbb{R}$ and a non-smooth part $F : \mathcal{X} \to \mathbb{R}$, i.e. G is a closed, convex, continuously differentiable function and its gradient ∇G is Lipschitz continuous. F is a closed, convex but not necessarily differentiable function, but its proximal operator can be evaluated efficiently. The scaled proximal operator is defined as follows.

Definition 4.30. Let a convex proper lower semicontinuous function $F : \mathcal{X} \to (-\infty, \infty]$, and a symmetric positive definite matrix H be given. The *scaled proximal operator* with matrix H is defined by

$$\text{prox}_{\tau F}^H(v) = \arg\min_x F(x) + \frac{1}{2\tau} \|x - v\|_H^2, \tag{4.35}$$

where $\|x\|_H = \langle x, Hx \rangle_{\mathcal{X}}^{1/2}$ is the norm on the Hilbert space $\mathcal{X}$ weighted by the symmetric positive definite matrix H.

A line search method is given by

$$x_{k+1} = x_k + \tau \Delta x_k, \tag{4.36}$$

where τ is a step length and Δx_k is a search direction, where τ is chosen adaptively to the search direction. The proximal Newton-type method approximates only the smooth part G with a local quadratic model:

$$\hat{G}_k(y) = G(x_k) + \langle \nabla G(x_k), y - x_k \rangle + \frac{1}{2} \|y - x_k\|_{H_k}^2, \tag{4.37}$$

where $H_k = \nabla^2 G(x_k)$. A proximal Newton-type search direction Δx_k solves the subproblem

$$\begin{aligned} \Delta x_k &= \arg\min_d F(x_k + d) + \hat{G}_k(x_k + d) \\ &= \arg\min_d F(x_k + d) + G(x_k) + \langle \nabla G(x_k), d \rangle_{\mathcal{X}} + \frac{1}{2} \|d\|_{H_k}^2 \\ &= \arg\min_d F(x_k + d) + \langle \nabla G(x_k), d \rangle_{\mathcal{X}} + \frac{1}{2} \|d\|_{H_k}^2 \end{aligned} \tag{4.38}$$

The proximal Newton-type search directions in terms of the scaled proximal operator, (4.35), reads

$$\Delta x = \text{prox}_F^H \left(x - H^{-1} \nabla G(x) \right) - x. \tag{4.39}$$

The generic proximal Newton-type method is given by

Algorithm 3 proximal Newton-type method, cf. [Lee et al., 2012]

1. choose an initial point $x_0 \in \mathcal{X}$
2. iterate for $k = 0, 1, \ldots$
   (i) choose H_k, a positive definite approximation to the Hessian
   (ii) compute a search direction:

$$\Delta x_k = \operatorname{prox}_F^{H_k}\left(x_k - H_k^{-1}\nabla G(x_k)\right) - x_k \tag{4.40}$$

   (iii) select a step size τ_k with line search
   (iv) $x_{k+1} = x_k + \tau_k \Delta x_k$

In [Lee et al., 2012], the proximal Newton-type method is proved to converge globally, if the subproblems are evaluated exactly. Further, if the functionals F and G are strongly convex the method converges quadratically.

4.2 Convex Constrained Optimization

The problems under consideration in this section are those of minimizing a convex proper and lower semicontinuous function F with respect to additional constraints. In the first part, affine constraints are analyzed, while in the second part the case of closed convex set type constraints is investigated.

4.2.1 Equality Constraint

Consider the primal formulation of the convex constrained minimization problem:

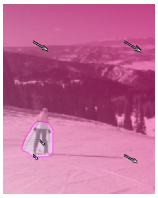
$$\min_x F(x), \qquad \text{subject to} \quad Kx = b, \tag{4.41}$$

where $F : \mathcal{X} \to (-\infty, \infty]$ is proper convex and lower semicontinuous and $K : \mathcal{X} \to \mathcal{Y}$ is a bounded linear operator, and $\mathcal{X}, \mathcal{Y}$ are real Hilbert spaces. The constraint is modeled through the *Lagrangian*

$$\mathcal{L}(x, y) = F(x) + \langle Kx - b, y\rangle_{\mathcal{Y}}\,. \tag{4.42}$$

The introduced variable y is called the *Lagrangian multiplier*. It can be seen as a penalization for violating the constraint. Therefore, the new problem is a saddle point problem. We want to minimize w.r.t. x and maximize w.r.t. the Lagrangian multiplier y.

$$\max_y \min_x \mathcal{L}(x, y) \tag{4.43}$$



The optimality conditions for non differentiable F, stated in the following theorem, can be found in [Bauschke and Combettes, 2011, sections 19 and 26].

Theorem 4.31 (Optimality conditions)**.** *Let the functional F and operator K be defined as in problem (4.41), and suppose $b \in K(\operatorname{dom} F)$. Then x_* is a solution to (4.41) if and only if*

$$Kx_* = b \quad \text{and} \quad 0 \in \partial F(x_*) + K^* y_*, \tag{4.44}$$

in which case y_* is a Lagrangian multiplier associated with x_*, and (x_*, y_*) is a saddle point of the Lagrangian $\mathcal{L}(x, y)$.

The Lagrangian multiplier is also called the *dual variable*, and problem (4.41) is also called the *primal problem*. With the dual variable, the *dual problem* can be formulated in terms of the convex conjugate of function F, cf. e.g. [Bauschke and Combettes, 2011, chapter 13].

Definition 4.32 (Convex conjugate)**.** Let $F : \mathcal{X} \to (-\infty, \infty]$ be a proper functional on a real Hilbert space $\mathcal{X}$. Then $F^* : \mathcal{X} \to (-\infty, \infty]$ defined by

$$F^*(z) = \sup_{x \in \mathcal{X}} \langle z, x \rangle_{\mathcal{X}} - F(x) \tag{4.45}$$

is called the *convex conjugate* or *(Fenchel-) conjugate* of F.

Remark 4.33. Note that the adjoint of the linear operator $K : \mathcal{X} \to \mathcal{Y}$, denoted by K^* should not be confused with the convex conjugate of functional $F : \mathcal{X} \to (-\infty, \infty]$, denoted by F^*, cf. Definition 4.32.

The *dual function* is given by

$$G(y) = \inf_x \mathcal{L}(x, y) = -F^*(-K^* y) - \langle b, y \rangle_{\mathcal{Y}}. \tag{4.46}$$

Then, the *dual problem* or dual formulation is given by

$$\max_y G(y), \qquad \text{subject to} \quad y \in \mathcal{Y}. \tag{4.47}$$

Note, that for the optimal value of the primal problem (4.41), $F(x_*)$, and of the dual problem (4.47), $G(y_*)$, usually a duality gap exists, i.e. $F(x_*) \neq -G(y_*)$. However, if the requirements of Theorem 4.31 are fulfilled, there is no duality gap.

Proximal operators of the convex conjugate F^*, which might be easier to compute, can be derived through Moreau's identity, also known as Moreau's decomposition, cf. [Bauschke and Combettes, 2011, section 14.1]

Theorem 4.34 (Moreau's identity)**.** *Let F be convex proper and lower semicontinuous, $\tau > 0$ and the conjugate of F be denoted by F^*. Then the following equality holds:*

$$\operatorname{prox}_{\tau F}(x) = x - \tau \operatorname{prox}_{\frac{1}{\tau} F^*}\left(\frac{x}{\tau}\right) \tag{4.48}$$

Augmented Lagrangian. The following method, which should solve the constrained problem in (4.41), is based on an augmented Lagrangian, with a quadratic augmentation term.

Definition 4.35 (Augmented Lagrangian)**.** Consider again the constrained minimization problem

$$\min_x F(x), \qquad \text{subject to} \quad Kx = b. \tag{4.41}$$

The *augmented Lagrangian* is given by

$$\mathcal{L}_\lambda(x, y) = F(x) + \langle y, Kx - b\rangle_{\mathcal{Y}} + \frac{\lambda}{2} \|Kx - b\|_{\mathcal{Y}}^2, \tag{4.49}$$

with the augmentation parameter $\lambda > 0$.

A solution x_* with Lagrangian multiplier y_* for problem (4.41) is given by a saddle point of

$$\max_y \min_x \mathcal{L}_\lambda(x, y). \tag{4.50}$$

The augmented Lagrangian can also be seen as the Lagrangian for the problem

$$\min_x F(x) + \frac{\lambda}{2} \|Kx - b\|_{\mathcal{Y}}^2, \qquad \text{subject to} \quad Kx = b, \tag{4.51}$$

which has the same minimizer(s), x_*, as problem (4.41), since the quadratic augmentation term is zero if the constraint is fulfilled.
A basic algorithm working on the augmented Lagrangian formulation is the *method of multipliers*, see i.e. [Boyd et al., 2011] for further details. The method of multipliers solves the saddle point problem by alternatingly minimizing with respect to x and maximizing with respect to the dual variable y:

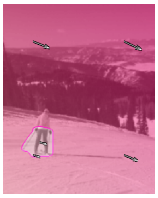
Algorithm 4 Method of Multipliers, cf. [Boyd et al., 2011]

1. choose an augmentation parameter $\lambda > 0$, an initial point $x_0 \in \mathcal{X}$ and set $y_0 = 0 \in \mathcal{Y}$.

2. iterate for $k = 0, 1, \ldots$

$$x_{k+1} = \arg\min_x F(x) + \langle y_k, Kx - b\rangle_{\mathcal{Y}} + \frac{\lambda}{2} \|Kx - b\|_{\mathcal{Y}}^2$$
$$y_{k+1} = y_k + \lambda(Kx_{k+1} - b)$$

The update of y is a gradient ascent step with step size λ. Analyzing the optimality conditions, it can be observed, that this step size leads to a fulfillment of the second optimality condition, cf. [Boyd et al., 2011]:



Optimality Conditions. The optimality conditions for problem (4.41) in the differentiable case are given by

$$Kx_* - b = 0, \qquad \nabla F(x_*) + K^* y_* = 0 \tag{4.52}$$

After the minimization with respect to x in iteration k, x_{k+1} minimizes $\mathcal{L}_\lambda(x, y_k)$ by definition. Thus,

$$\begin{aligned} 0 &= \nabla_x \mathcal{L}_\lambda(x_{k+1}, y_k) && (4.53)\\ &= \nabla F(x_{k+1}) + K^*\big(y_k + \lambda(Kx_{k+1} - b)\big) && (4.54)\\ &= \nabla F(x_{k+1}) + K^* y_{k+1} && (4.55) \end{aligned}$$

By using the step size λ, the second optimality condition in (4.52) is fulfilled automatically after updating y with the gradient ascent step.

Remark 4.36. The fulfillment of the optimality condition with the method of multipliers also works for non-differentiable functions with the subgradient, cf. [Boyd et al., 2011].

The augmented Lagrangian can also be transformed into an equivalent scaled form, which can be more convenient to use. Completing the square yields the form

$$\begin{aligned} \langle u, v\rangle + \frac{\lambda}{2}\|u\|^2 &= \frac{\lambda}{2}\left(2\left\langle u, \lambda^{-1}v\right\rangle + \langle u, u\rangle + \left\langle \lambda^{-1}v, \lambda^{-1}v\right\rangle - \left\langle \lambda^{-1}v, \lambda^{-1}v\right\rangle\right) && (4.56)\\ &= \frac{\lambda}{2}\left\|u + \lambda^{-1}v\right\|^2 - \frac{\lambda}{2}\left\|\lambda^{-1}v\right\|^2 && (4.57) \end{aligned}$$

For $u := Kx - b$ and the scaled dual variable $\hat{y} := \lambda^{-1}v$ we obtain the *scaled form* of the augmented Lagrangan:

$$\mathcal{L}_\lambda(x, \hat{y}) = F(x) + \frac{\lambda}{2}\|Kx - b + \hat{y}\|_{\mathcal{Y}}^2 - \frac{\lambda}{2}\|\hat{y}\|_{\mathcal{Y}}^2 . \tag{4.58}$$

If the method of multipliers is applied to the scaled augmented Lagrangian, by replacing the dual variable with the scaled dual variable $\hat{y} := \lambda^{-1}y$, the step size in the update for $\hat{y}$ will be 1:

$$\hat{y}_{k+1} = \hat{y}_k + Kx_{k+1} - b \tag{4.59}$$

4.2.2 Convex Set

Consider the problem of minimizing a function over a closed convex set:

$$\min_x F(x), \qquad \text{subject to} \quad x \in \mathcal{C}, \tag{4.60}$$

where $\mathcal{C}$ is a convex closed set and $F : \mathcal{H} \to (-\infty, \infty]$ is a convex, proper and lower semicontinuous functional.

The optimality condition for the constrained problem is given by the following theorem [Bauschke and Combettes, 2011, section 26.2]. For the differentiable case, the optimality conditions can be found in [Boyd and Vandenberghe, 2004, Chapter 4.2.3].

Theorem 4.37 (Optimality condition)**.** *Let F be a function as defined in problem (4.60), $\mathcal{C}$ a convex closed subset of $\mathcal{H}$ and let $0 \in \operatorname{sri}(\mathcal{C} - \operatorname{dom} F)$. A point x_* is optimal if and only if*

$$x_* \in \mathcal{C} \quad \text{and} \quad \exists w \in \partial F(x_*) : \ \langle w, y - x_* \rangle_{\mathcal{H}} \geq 0 \quad \forall y \in \mathcal{C}. \tag{4.61}$$

The constraint can be modeled by adding an *indicator function* for $\mathcal{C}$ to the objective function.

Definition 4.38 (Indicator function)**.** The indicator function on a convex set $\mathcal{C}$ is given by

$$\delta_{\mathcal{C}}(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ \infty & \text{if } x \notin \mathcal{C}. \end{cases} \tag{4.62}$$

Using the indicator function on the constrained optimization problem, the problem can be rewritten as an unconstrained optimization problem:

$$\min_x \delta_{\mathcal{C}}(x) + F(x). \tag{4.63}$$

The indicator function is not differentiable. In the following, we will see, that using the proximal point method on the indicator function yields a projection onto the set $\mathcal{C}$.

Definition 4.39 (Orthogonal projection)**.** The orthogonal projection onto a set $\mathcal{C} \subset \mathcal{H}$, where $\mathcal{H}$ is a Hilbert space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is a linear mapping $\mathcal{P}_{\mathcal{C}} : \mathcal{H} \to \mathcal{H}$ where for all $v \in \mathcal{H}$ the following conditions hold:

1. $\mathcal{P}_{\mathcal{C}}(v) \in \mathcal{C}$
2. $\langle v - \mathcal{P}_{\mathcal{C}}(v), y - \mathcal{P}_{\mathcal{C}}(v) \rangle_{\mathcal{H}} \leq 0 \quad \forall y \in \mathcal{C}$

Assume F is differentiable. The proximal gradient method, given in algorithm 2, applied on the problem (4.63) with the substitution for the gradient step $v_k := x_k - \tau \nabla F(x_k)$ yields

$$x_{k+1} = \operatorname{prox}_{\tau \delta_{\mathcal{C}}}(x_k - \tau \nabla F(x_k)) = \arg\min_x \left(\delta_{\mathcal{C}}(x) + \frac{\|x - v_k\|_{\mathcal{H}}^2}{2\tau} \right) \tag{4.64}$$

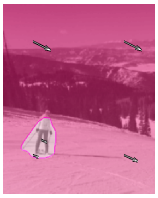
$$= \arg\min_{x \in \mathcal{C}} \left(\frac{\|x - v_k\|_{\mathcal{H}}^2}{2\tau} \right) = \mathcal{P}_{\mathcal{C}}(v_k) = \mathcal{P}_{\mathcal{C}}(x_k - \tau \nabla F(x_k)). \tag{4.65}$$

The argument is minimized for $x = v_k$. If $v_k \notin \mathcal{C}$, then the orthogonal projection $P_{\mathcal{C}}(v_k)$ is the unique element in $\mathcal{C}$ that minimizes the distance to v_k.

Proposition 4.40. *Let $\mathcal{H}$ be a real Hilbert space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and norm $\|\cdot\|_{\mathcal{H}}$, and $\mathcal{C} \subset \mathcal{H}$ a convex set. The point x_* is the unique solution to*

$$\min_{x \in \mathcal{C}} F(x), \quad F(x) := \frac{1}{2} \|x - v\|_{\mathcal{H}}^2,$$

if and only if $x_* = \mathcal{P}_{\mathcal{C}}(v)$.



Proof. F is strictly convex and bounded from below, thus has a unique minimizer and since $\mathcal{C}$ is convex and closed, has a unique minimizer on $\mathcal{C}$. There are two different cases whether v lies inside or outside the set $\mathcal{C}$: If $v \in \mathcal{C}$, the orthogonal projection of v is given by $\mathcal{P}_{\mathcal{C}}(v) = v$ and v minimizes the unconstrained problem $\min_x F(x)$.
If v minimizes the unconstrained problem $\min_x F(x)$, $x_* = v$ and $x_* \in \mathcal{C}$: $\mathcal{P}_{\mathcal{C}}(v) = \mathcal{P}_{\mathcal{C}}(x_*) = x_*$

If $v \notin \mathcal{C}$, the optimality condition for a minimizer x_* to the constrained optimization problem $\min_{x \in \mathcal{C}} F(x)$ is given by $\langle \nabla F(x_*), y - x_* \rangle_{\mathcal{H}} \geq 0$, $\forall y \in \mathcal{C}$. With $\nabla F(x_*) = x_* - v$ we have

$$\langle x_* - v, y - x_* \rangle_{\mathcal{H}} \geq 0 \quad \forall y \in \mathcal{C}. \tag{4.66}$$

Assume $x_* = \mathcal{P}_{\mathcal{C}}(v)$:

$$\langle \mathcal{P}_{\mathcal{C}}(v) - v, y - \mathcal{P}_{\mathcal{C}}(v) \rangle_{\mathcal{H}} \geq 0 \quad \forall y \in \mathcal{C}, \tag{4.67}$$

which is the definition of the orthogonal projection, $\langle v - \mathcal{P}_{\mathcal{C}}(v), y - \mathcal{P}_{\mathcal{C}}(v) \rangle_{\mathcal{H}} \leq 0$ for all $y \in \mathcal{C}$. □

4.3 Convex Composite with Operator K: $F(Kx) + G(x)$

Consider the problem

$$\min_x F(Kx) + G(x), \tag{4.68}$$

where $\mathcal{X}$ is a real Hilbert space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{X}}$ and induced norm $\| \cdot \|_{\mathcal{X}}$, $K : \mathcal{X} \to \mathcal{Y}$ is a continuous linear operator into another Hilbert space $\mathcal{Y}$, $F : \mathcal{Y} \to (-\infty, \infty]$ is a convex lower semicontinuous functional and $G : \mathcal{X} \to (-\infty, \infty]$ is proper, convex and lower semicontinuous. The optimality conditions given in this section can be found in [Bauschke and Combettes, 2011, chapters 19 and 26].

Theorem 4.41 (Optimality conditions)**.** *Let F, G and K be defined as in problem (4.68) and let $0 \in \operatorname{sri}(\operatorname{dom} F - K(\operatorname{dom} G))$. Then*

$$x_* \text{ is a solution to problem (4.68)} \quad \Leftrightarrow \quad 0 \in K^* \partial F(Kx_*) + \partial G(x_*). \tag{4.69}$$

In the following, two different method classes for problem (4.68) are presented, those operating on a primal-dual formulation and methods operating on an augmented Lagrangian formulation through a splitting.

4.3.1 Augmented Lagrangian

The idea behind this method is to introduce a new variable together with an equality constraint and formulate the problem through an augmented Lagrangian. The new variable is created through a splitting of the primal variable x.

Splitting. A new variable z is introduced together with the constraint $z = Kx$:

$$\min_{x,z} F(z) + G(x), \qquad \text{subject to} \quad z = Kx \tag{4.70}$$

For the split problem, the (scaled) quadratic augmented Lagrangian, with augmentation parameter $\sigma > 0$ is given by:

$$\mathcal{L}_\sigma(x, z, y) = F(z) + G(x) + \frac{\sigma}{2}\|Kx - z + y\|_{\mathcal{Y}}^2 - \frac{\sigma}{2}\|y\|_{\mathcal{Y}}^2 \tag{4.71}$$

The associated optimization problem reads

$$\max_{y} \min_{x,z} \mathcal{L}_\sigma(x, z, y) \tag{4.72}$$

The optimality conditions for the equality constrained minimization problem (4.70) are given by

Theorem 4.42 (Optimality conditions)**.** *Let the functionals F, G and operator K be defined as in problem (4.68), and $0 \in \operatorname{sri}(\operatorname{dom} F - K(\operatorname{dom} G))$. Then (x_*, z_*) is a solution to (4.68) if and only if*

$$Kx_* = z_* \quad \text{and} \quad \begin{cases} 0 \in \partial F(z_*) - y_*, \text{ and} \\ 0 \in \partial G(x_*) + K^* y_*, \end{cases} \tag{4.73}$$

where y_* is a Lagrangian multiplier associated with (x_*, z_*).

The augmented Lagrangian is minimized with respect to x, z alternatingly and, as done in the method of multipliers, given in algorithm 4, maximized in the dual variable y via a gradient ascent step.
The update steps for the *alternating direction method of multipliers (ADMM)* using the scaled form of the augmented Lagrangian with $\sigma\hat{y} = y$ read as follows

Algorithm 5 alternating direction method of multipliers(ADMM), cf. [Boyd et al., 2011]

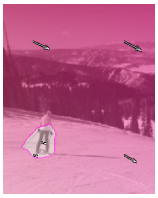
1. choose augmentation parameter $\sigma > 0$, and an initial point $x_0 \in \mathcal{X}, z_0 = Kx_0 \in \mathcal{Y}$ and set $y_0 = 0 \in \mathcal{Y}$.

2. iterate for $k = 0, 1, \ldots$

$$x_{k+1} = \arg\min_x G(x) + \frac{\sigma}{2}\|Kx - z_k + \hat{y}_k\|_{\mathcal{Y}}^2 \tag{4.74}$$

$$z_{k+1} = \arg\min_z F(z) + \frac{\sigma}{2}\|Kx_{k+1} - z + \hat{y}_k\|_{\mathcal{Y}}^2 \tag{4.75}$$

$$\hat{y}_{k+1} = \hat{y}_k + Kx_{k+1} - z_{k+1} \tag{4.76}$$



Convergence. In several manuscripts, the algorithm is shown to converge, e.g. [Eckstein and Bertsekas, 1992]. In [Boyd et al., 2011], a general proof is given and their convergence result is presented here:

Theorem 4.43 (Convergence ADMM, [Boyd et al., 2011])**.** *If the extended real valued functions $\mathcal{F} : \mathcal{Y} \to (-\infty, \infty]$ and $G : \mathcal{X} \to (-\infty, \infty]$ are closed, proper, and convex and the Lagrangian $\mathcal{L}$ has a saddle point (x_*, z_*, y_*), i.e.*

$$\mathcal{L}(x_*, z_*, \hat{y}) \leq \mathcal{L}(x_*, z_*, \hat{y}_*) \leq \mathcal{L}(x, z, \hat{y}_*) \tag{4.77}$$

holds for all x, z, y, then the ADMM iterates satisfy:

1. *Residual convergence: $r_k := Kx_k - z_k \to 0$ as $k \to \infty$.*
2. *Objective convergence: $F(z_k) + G(x_k) \to p_*$ as $k \to \infty$, i.e. , the objective function of the iterates approaches the optimal value p_*.*
3. *Dual variable convergence: $y_k \to y_*$ as $k \to \infty$, where y_* is a dual optimal point.*

The assumptions of this theorem include, that $\mathcal{L}(x_*, z_*, y_*)$ is finite for any saddle point (x_*, z_*, y_*). This also implies, that $F(z_*) < \infty$ and $G(x_*) < \infty$.
The ordering of the updates in the ADMM is not important for convergence. As described in [Boyd et al., 2011], the update ordering can be changed and also multiple iterations per update are possible.
As for the method of multipliers, the gradient ascent step in the update of y or $\hat{y}$ with step size σ or 1, respectively, leads to a fulfillment of one of the three optimality conditions, cf. [Boyd et al., 2011]: After the minimization with respect to z in iteration k, z_{k+1} minimizes $\mathcal{L}_\sigma(x_{k+1}, z, \hat{y}_k)$ by definition. Thus,

$$\begin{aligned} 0 &\in \partial F(z_{k+1}) - \sigma(Kx_{k+1} - z_{k+1} + \hat{y}_k) && (4.78)\\ &= \partial F(z_{k+1}) - \sigma\hat{y}_{k+1}, && (4.79) \end{aligned}$$

which, with $\sigma\hat{y} = y$ gives the second optimality condition.
Further, the algorithm will also converge for suitable inexact minimization in the update steps, which can be shown with the convergence results on the Douglas–Rachford splitting, which is equivalent to the ADMM, by [Eckstein and Bertsekas, 1992].

Remark 4.44. The ADMM is not only equivalent to the Douglas–Rachford splitting but also many different methods, such as the split Bregman method, cf. [Goldstein and Osher, 2009], which is sometimes not easy to see. For example, performing a Douglas–Rachford splitting on the dual formulation is equivalent to the ADMM on the primal formulation and vice versa. These equivalences have been intensively studied in [Boyd et al., 2011, Esser et al., 2010, Setzer, 2011].

Minimizations w.r.t. x, z. The minimization with respect to z, in equation (4.75), can be transferred directly into a proximal point algorithm:

$$z_{k+1} = \operatorname{prox}_{\frac{1}{\sigma}F}(Kx_{k+1} + y_k). \tag{4.80}$$

Due to the operator K, the minimization with respect to x, in equation (4.74), cannot be written as a proximal operator directly. Therefore, different methods can be used:

Gauss–Seidel Iteration(s). If G is differentiable the optimality condition or Euler–Lagrange equations lead to

$$\nabla G(x) + \sigma K^* K x = \sigma K^*(z_k - y_k). \tag{4.81}$$

Depending on G, in particular if G is linear or quadratic, this leads to a system of linear equations which is solved by a Gauss–Seidel iteration:

$$(\nabla G + \sigma K^* K)x_{k+1} = \sigma K^*(z_k - y_k), \tag{4.82}$$

or, if $G(x)$ is linear in x and ∇G constant, we have:

$$(\sigma K^* K)x_{k+1} = \sigma K^*(z_k - y_k) - \nabla G. \tag{4.83}$$

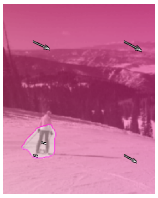
In the literature some authors claim, that one step of the Gauss–Seidel iteration during the update of x, i.e. before updating z and y (which we call a Gauss–Seidel sweep in the following), is enough for the ADMM to converge, cf. [Goldstein et al., 2010]. Thus, with one Gauss–Seidel sweep only an approximate solution of the linear system is calculated. With the partition $(\nabla G + \sigma K^* K) = L_\llcorner + L_\urcorner$, where $L_\llcorner$ and $L_\urcorner$ are lower and upper triangular matrices, the Gauss–Seidel sweep reads

$$x_{k+1} = {L_\llcorner}^{-1}\left(\sigma K^*(z_k - y_k) - L_\urcorner x_k\right). \tag{4.84}$$

In [Boyd et al., 2011], the authors suggest as an alternative also to do more than one Gauss–Seidel sweep. We will analyze the effect of different numbers of Gauss–Seidel sweeps on the convergence of the method in chapter 6.
In [Goldstein et al., 2010], also further constraints are involved which are updated through projections after the Gauss–Seidel step.

Linearization. Another possibility for the update of (4.74) in algorithm 5 is to do a linearization of the quadratic norm, or a proximal gradient step for the x-update, where the quadratic penalization term is used as the differentiable term for the proximal gradient method:

$$x_{k+1} = \operatorname{prox}_{\tau G}\left(x_k - \sigma K^*(Kx_k - z_k + y_k)\right) \tag{4.85}$$



Proximal Newton-type update. As with the linearization, also a proximal Newton-type method can be integrated into the update step. Assume that G is nonsmooth, the gradient and the Hessian H of the smooth part are given by

$$\nabla \mathcal{L}_{smooth}(x, z_k, y_k) = \sigma K^* K x - \sigma K^*(z_k - y_k), \tag{4.86}$$
$$H(x) = \sigma K^* K, \tag{4.87}$$

and the proximal Newton-type search direction reads

$$\Delta x = \operatorname{prox}_G^{\sigma K^* K}\left(x_k - \frac{1}{\sigma}(K^*K)^{-1}(\sigma K^* K x_k - \sigma K^*(z_k - y_k))\right) \tag{4.88}$$
$$= \operatorname{prox}_G^{\sigma K^* K}\left((K^*K)^{-1} K^*(z_k - y_k)\right). \tag{4.89}$$

This search direction and thus the update step might be hard to compute due to the scaled proximal operator and the inverse of K^*K. In chapter 6, we will see that if G is a composition of a nonsmooth and a linear function, the proximal Newton-type method is equivalent to the Gauss–Seidel method.

The ADMM with the proximal gradient step update in x, (4.85), is called the linearized ADMM and is also known as split inexact Uzawa method, cf. [Parikh and Boyd, 2013]. The algorithm with the proximal gradient update in x in terms of proximal operator is given by:

Algorithm 6 Linearized ADMM, cf. [Parikh and Boyd, 2013]

1. choose step sizes $\tau > 0$, augmentation parameter $\sigma > 0$, and an initial point $x_0 \in \mathcal{X}, z_0 = Kx_0 \in \mathcal{Y}$ and set $y_0 = 0 \in \mathcal{Y}$.

2. iterate for $k = 0, 1, \ldots$

$$x_{k+1} = \operatorname{prox}_{\tau G}\left(x_k - \tau\sigma K^*(Kx_k - z_k + y_k)\right) \tag{4.90}$$
$$z_{k+1} = \operatorname{prox}_{\frac{1}{\sigma}F}\left(Kx_{k+1} + y_k\right) \tag{4.91}$$
$$y_{k+1} = y_k + Kx_{k+1} - z_{k+1} \tag{4.92}$$

4.3.2 Primal-Dual Formulation

The primal-dual formulation for problem (4.68) reads as follows

$$\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} \; -F^*(y) + G(x) + \langle Kx, y\rangle_{\mathcal{Y}}. \tag{4.93}$$

The optimal point (x_*, y_*) is a saddle point, since it maximizes (4.93) with respect to y and minimizes (4.93) with respect to x.

The optimality conditions for an optimal point (x_*, y_*) can be derived with Theorem 4.6:

$$0 \in -\partial F^*(y_*) + Kx_* \tag{4.94}$$
$$0 \in \partial G(x_*) + K^* y_* \tag{4.95}$$

A fixed point iteration based on the resolvent operator can be derived by introducing two step sizes $\sigma, \tau > 0$

$$0 \in -\sigma\partial F^*(y_*) + y_* - y_* + \sigma K x_* \tag{4.96}$$
$$0 \in -\tau\partial G(x_*) + x_* - x_* - \tau K^* y_* \tag{4.97}$$

Rearranging leads to

$$(I + \sigma\partial F^*)y_* \ni y_* + \sigma K x_* \tag{4.98}$$
$$(I + \tau\partial G)x_* \ni x_* - \tau K^* y_* \tag{4.99}$$

Since the resolvent is single valued, we have

$$y_* = (I + \sigma\partial F^*)^{-1}(y_* + \sigma K x_*) = \operatorname{prox}_{\sigma F^*}(y_* + \sigma K x_*) \tag{4.100}$$
$$x_* = (I + \tau\partial G)^{-1}(x_* - \tau K^* y_*) = \operatorname{prox}_{\tau G}(x_* - \tau K^* y_*) \tag{4.101}$$

The update steps can also be interpreted as an alternating proximal gradient descent and proximal gradient ascent method on the saddle point formulation (4.93).

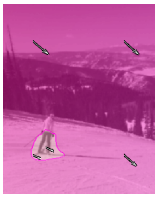
From this fixed point equations, different algorithms can be derived. Among the oldest are the explicit Arrow–Hurwicz method, and the semi-implicit Arrow–Hurwicz method, cf. [Arrow et al., 1958].

Later, different extrapolation or preconditioning steps were included, as e.g. in [Esser et al., 2010, Nesterov, 2005, Pock et al., 2009]

One algorithm became popular recently, and is introduced here explicitly. It is also based on proximal gradient descent and ascent steps, but includes an additional extrapolation step for acceleration:

$$\bar{x}_{k+1} = x_{k+1} + \theta(x_{k+1} - x_k)$$

where $\theta \in [0, 1]$ is an extrapolation parameter. In [Chambolle and Pock, 2011] convergence properties are analyzed and it is applied to different imaging problems.



Algorithm 7 primal-dual (PD), [Chambolle and Pock, 2011, Pock et al., 2009, Popov, 1980]

1. choose step sizes $\tau, \sigma > 0$, an extrapolation parameter $\theta \in [0,1]$, an initial point $(x_0, y_0) \in \mathcal{X} \times \mathcal{Y}$, and set $\bar{x}_0 = x_0$.

2. iterate for $k = 0, 1, \ldots$

$$\begin{aligned} y_{k+1} &= \text{prox}_{\sigma F^*}(y_k + \sigma K \bar{x}_k) \\ x_{k+1} &= \text{prox}_{\tau G}(x_k - \tau K^* y_{k+1}) \\ \bar{x}_{k+1} &= x_{k+1} + \theta(x_{k+1} - x_k) \end{aligned}$$

Convergence. The primal dual algorithm is a first order method and, for $\theta = 1$, it is shown in [Chambolle and Pock, 2011] to converge as $\mathcal{O}(k^{-1})$. In [Chambolle and Pock, 2011], it has also been shown that for uniformly convex problems a convergence rate of $\mathcal{O}(k^{-2})$ can be achieved. Following [Nesterov, 2005, Chambolle and Pock, 2011], this algorithm has the optimal asymptotic convergence rate, and cannot be improved further. However, it comes with a step size restriction which yields a large number of iteration steps in praxis. In the following, we examine equivalences of this algorithm with the ADMM.

4.3.3 Equivalences

As described in [Parikh and Boyd, 2013], the described methods are special cases of the proximal point and the proximal gradient method. As already mentioned, connections and equivalences between several primal-dual algorithms and augmented Lagrangian methods are analyzed and discussed in e.g. [Esser et al., 2010].
For a better understanding of the connections between the different methods, we show the equivalence of the primal-dual algorithm and the linearized ADMM here, cf. [Tichmann and Junge, 2014].
We start with the primal dual formulation of optimization problem (4.68)

$$\min_x \max_y \langle Kx, y \rangle - F^*(y) + G(x) \tag{4.102}$$

and the primal-dual algorithm described in algorithm 7

$$y^{k+1} = \text{prox}_{\sigma F^*}\left(y^k + \sigma K((1+\theta)x^k - \theta x^{k-1})\right) \tag{4.103}$$

$$x^{k+1} = \text{prox}_{\tau G}\left(x^k - \tau K^* y^{k+1}\right). \tag{4.104}$$

We reformulate Moreau's Identity (4.34)

$$\text{prox}_{\sigma F^*}(\sigma x) = \sigma x - \sigma \text{prox}_{\frac{1}{\sigma}F}(x). \tag{4.105}$$

Applying Moreau's Identity to equation (4.103) gives

$$y^{k+1} = \operatorname{prox}_{\sigma F^*}\left(y^k + \sigma K((1+\theta)x^k - \theta x^{k-1})\right) \tag{4.106}$$

$$= \operatorname{prox}_{\sigma F^*}\left\{\sigma\left(\underbrace{\frac{1}{\sigma}y^k + \theta K(x^k - x^{k-1})}_{=:\eta^k} + Kx^k\right)\right\} \tag{4.107}$$

$$= \operatorname{prox}_{\sigma F^*}\left(\sigma(\eta^k + Kx^k)\right) \tag{4.108}$$

$$\overset{(4.105)}{=} \sigma(\eta^k + Kx^k) - \sigma\underbrace{\operatorname{prox}_{\frac{1}{\sigma}F}\left(\eta^k + Kx^k\right)}_{=:z^{k+1}} \tag{4.109}$$

$$= \sigma(\eta^k + Kx^k - z^{k+1}). \tag{4.110}$$

Equation (4.110) applied to the definition $\eta^k := \frac{1}{\sigma}y^k + \theta K(x^k - x^{k-1})$ gives for $k+1$

$$\eta^{k+1} = \frac{1}{\sigma}y^{k+1} + \theta K(x^{k+1} - x^k) = \eta^k + Kx^k - z^{k+1} + \theta K(x^{k+1} - x^k) \tag{4.111}$$

$$= \eta^k - z^{k+1} + K(x^k + \theta(x^{k+1} - x^k)) \tag{4.112}$$

$$= \eta^k + \theta K x^{k+1} - z^{k+1} + (1-\theta)Kx^k. \tag{4.113}$$

We apply equation (4.110) to the update step given in equation (4.104):

$$x^{k+1} = \operatorname{prox}_{\tau G}\left(x^k - \tau K^* y^{k+1}\right) = \operatorname{prox}_{\tau G}\left(x^k - \tau\sigma K^*(\eta^k + Kx^k - z^{k+1})\right) \tag{4.114}$$

The algorithm now reads:

$$z^{k+1} = \operatorname{prox}_{\frac{1}{\sigma}F}\left(Kx^k + \eta^k\right) \tag{4.115}$$

$$x^{k+1} = \operatorname{prox}_{\tau G}\left(x^k - \tau\sigma K^*(Kx^k - z^{k+1} + \eta^k)\right) \tag{4.116}$$

$$\eta^{k+1} = \eta^k - z^{k+1} + K(x^k + \theta(x^{k+1} - x^k)), \tag{4.117}$$

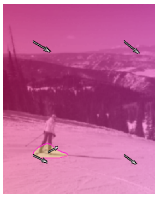
which is a more general version of the linearized ADMM with the extrapolation parameter $\theta \in [0,1]$ from the primal-dual algorithm.
For $\theta = 1$ we obtain the linearized ADMM using a proximal point method in the first update step and a proximal gradient method in the second update step:

$$z^{k+1} = \operatorname{prox}_{\frac{1}{\sigma}F}\left(Kx^k + \eta^k\right) \tag{4.118}$$

$$x^{k+1} = \operatorname{prox}_{\tau G}\left(x^k - \tau\sigma K^*(Kx^k - z^{k+1} + \eta^k)\right) \tag{4.119}$$

$$\eta^{k+1} = \eta^k + Kx^{k+1} - z^{k+1} \tag{4.120}$$

From this formulation, we can see that the step size given in the primal-dual algorithm is also the step size for the proximal gradient step in the x-update of the linearized



ADMM. We study the restriction on the step-size τ, given in the primal-dual algorithm, and associated problems in detail in the next chapter.
Let us summarize our findings for this section:

Proposition 4.45. *For $\theta = 1$ the linearized ADMM given in algorithm 6 and the primal-dual algorithm given in algorithm 7 are equivalent.*

Further, we have a general version of the linearized ADMM including the extrapolation step with parameter θ, which is designed to speed-up the convergence of the primal-dual algorithm, even though the convergence is only proven for $\theta = 1$.

5 Implicit Algorithms for Convex Problems

The methods presented in the previous section usually work well but come with a step size restriction. In some cases the step size has to be very small and the algorithms need a lot of iterations. Therefore, we investigate this phenomenon and the step size restriction and propose a new stable algorithm in the following. The Hilbert spaces in this chapter are finite dimensional.

5.1 Step Size Restrictions and Stiffness

First we examine, what causes the step size restriction in the described methods by analyzing the proximal gradient method on an example. Then, we consider stiffness—a problem often encountered during the numerical treatment of ordinary differential equations—which is related to the step size restriction and affects the required number of iteration steps in explicit methods.

5.1.1 Example: Proximal Gradient Method

The linearized ADMM, which is equivalent to the primal-dual method, uses a proximal gradient step with a step size restriction in the update of variable x. We analyze this update step first. The problem is given by (cf. equation (4.74))

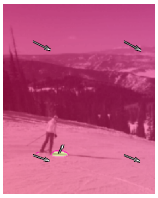
$$\min_x F(x) + \frac{\sigma}{2} \|Kx - z + y\|_{\mathcal{Y}}^2, \tag{5.1}$$

where $F : \mathcal{X} \to (-\infty, \infty]$ is a proper convex and lower semicontinuous functional on a finite dimensional Hilbert space $\mathcal{X}$ and $K : \mathcal{X} \to \mathcal{Y}$ is a linear operator into another finite dimensional Hilbert space $\mathcal{Y}$. $\sigma > 0$ is the parameter of the quadratic penalty in the augmented Lagrangian. The associated proximal gradient update step, described in equation (4.85), is given by:

$$x_{k+1} = \operatorname{prox}_{\tau F}(x_k - \tau\sigma K^*(Kx_k - z_k + y_k)), \tag{5.2}$$

with guaranteed convergence for step sizes $\tau < \frac{2}{\sigma\|K^*K\|}$, where $\|K^*K\|$ is the operator norm induced by the norms on the Hilbert spaces, cf. [Parikh and Boyd, 2013].
In order to simplify the analysis of this method, we consider the related problem

$$\min_x F(x) + \underbrace{\frac{1}{2} \|Kx\|_{\mathcal{Y}}^2 + \langle b, x \rangle_{\mathcal{X}}}_{G(x)}. \tag{5.3}$$



The proximal gradient method is given by

$$\begin{aligned} x_{k+1} &= \operatorname{prox}_{\tau F}\left(x_k - \tau \nabla G(x)\right) \\ &= \operatorname{prox}_{\tau F}\left(x_k - \tau(K^*Kx_k + b)\right) = (I + \tau\partial F)^{-1}\big((I - \tau K^*K)x_k - \tau b\big), \end{aligned} \tag{5.4}$$

with ∇G Lipschitz continuous with constant L, to be specified below.

5.1.2 Convergence

In the view of averaged operators and the convergence theory we can get an idea, where the step size restriction comes from. Therefore, in the following, we examine for which step sizes τ the update step of the proximal gradient method can be described as an averaged operator, and thus the convergence theory of the previous chapter is applicable. In this section, for better readability, the norm $\|\cdot\|_{\mathcal{H}}$ on the Hilbert space $\mathcal{H}$ is written without the subscript.

The proximal gradient method can be seen as the composition of two operators. From Theorem 4.12, we know that the composition of averaged operators is also an averaged operator. The proximal gradient method in consideration is a composition of the operators P and T:

$$(I + \tau\partial F)^{-1}((I - \tau K^*K)x_k - \tau b) = P \circ T(x_k). \tag{5.5}$$

The operator $P := (I + \tau\partial F)^{-1}$ is the resolvent of the subgradient ∂F, thus P is firmly nonexpansive and an averaged operator. Now we analyze if $T := (I - \tau K^*K)(\,\cdot\,) - \tau b$ is also an averaged operator. With the following lemma, we deal with translations:

Lemma 5.1. *Let $v \in \mathcal{H}$, and the averaged operator $M = (1-\alpha)I + \alpha N$ be given, with N nonexpansive. Then the operators $S_1 := M(\,\cdot\,) + v$, and $S_2 := M(\,\cdot\, + v)$ are averaged operators.*

Proof. First the operator $S_1 := M(\,\cdot\,) + v$ is written in the form of an averaged operator:

$$M(\,\cdot\,) + v = (1-\alpha)I(\,\cdot\,) + \alpha N(\,\cdot\,) + v = (1-\alpha)I(\,\cdot\,) + \alpha \underbrace{\left(N(\,\cdot\,) + \frac{v}{\alpha}\right)}_{=:\hat{N}(\,\cdot\,)} \tag{5.6}$$

Now, it can be seen, that the operator $\hat{N}(\,\cdot\,)$ is nonexpansive:

$$\left\|\hat{N}(x_1) - \hat{N}(x_2)\right\| = \left\|N(x_1) + \frac{v}{\alpha} - N(x_2) - \frac{v}{\alpha}\right\| = \|N(x_1) - N(x_2)\| \leq \|x_1 - x_2\|. \tag{5.7}$$

The proof of the second statement, that S_2 is an averaged operator, is analogous to the first. □

Hence, we rewrite T without the translation and show that $\hat{T}$ is an averaged operator

$$\hat{T} := I - \tau K^*K = (1-\alpha)I + \alpha \underbrace{(I - \frac{\tau}{\alpha}K^*K)}_{=:N}, \tag{5.8}$$

and try to find out, whether N is nonexpansive for some $\alpha \in (0,1)$. The operator K^*K is symmetric positive semidefinite, i.e. its spectrum satisfies $\sigma(K^*K) \geq 0$, $(K^*K)^* = K^*K$, and it can be orthogonally diagonalized[1] with the diagonal matrix D, $K^*K = S^*DS$, $\|K^*K\| = \|D\|$.

$$\|N(x) - N(y)\| = \left\|x - y - \frac{\tau}{\alpha}K^*K(x-y)\right\| = \left\|S^*S(x-y) - \frac{\tau}{\alpha}S^*DS(x-y)\right\| \tag{5.9}$$

$$\overset{z=S(x-y)}{=} \left\|z - \frac{\tau}{\alpha}Dz\right\| = \left\|(I - \frac{\tau}{\alpha}D)z\right\| \leq \max\left|1 - \frac{\tau}{\alpha}\sigma(D)\right| \|z\| \tag{5.10}$$

$$= \max\left|1 - \frac{\tau}{\alpha}\sigma(D)\right| \|S(x-y)\| = \max\left|1 - \frac{\tau}{\alpha}\sigma(D)\right| \|x-y\|. \tag{5.11}$$

The largest eigenvalue of D is defined by $\lambda_{\max} := \max\sigma(D)$, which is in our case equal to the Lipschitz constant $L = \lambda_{\max}$. We have

$$\sigma(D) \subset [0, L] \quad \Leftrightarrow \quad \sigma(I - \frac{\tau}{\alpha}D) \subset [1 - \frac{\tau}{\alpha}L, 1] \tag{5.12}$$

$$|1 - \frac{\tau}{\alpha}L| \leq 1 \quad \Leftrightarrow \quad \tau \leq \frac{2\alpha}{L} \overset{\alpha<1}{<} \frac{2}{L} \tag{5.13}$$

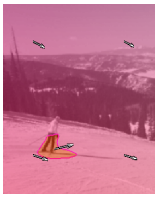
Since $\alpha < 1$, N is nonexpansive if the step size $\tau < 2/L$. With Theorem 4.10 the iteration with $P \circ T$ converges to a fixed point, if one exists.

Remark 5.2. If the operator K^*K does not have the eigenvalue 0, it can be shown, that the operator $T := I - \tau K^*K$ is a contraction for $\tau < 2/L$. The composition of a contractive operator and a nonexpansive operator is a contraction, and, with the Banach fixed point theorem, will converge to the unique fixed point.

5.1.3 Eigenvalue Analysis - Stiffness

The term stiffness is used, if for an iterative method, the step sizes become too small such that to many steps are required for a sufficiently accurate solution. What exactly is meant by "too small", depends on the problem. In the following part, this is studied further and possible consequences are investigated.
In the previous section, we have seen, that the step size restriction arises from the gradient descent step, the explicit step of the proximal gradient method, which handles the

[1] The orthogonal diagonalization does only work in the described way, if the norm on the Hilbert space is the Euclidian norm. However, the main result needed in this context, $\|A^*A\| = \max|\sigma(A^*A)|$, also holds for a norm induced by the scalar product $\langle\cdot,\cdot\rangle_M$, with an arbitrary symmetric positive semidefinite matrix M: $\|A^*A\|_M = \max|\sigma(A^*A)|$, where the adjoint is of course taken with respect to $\langle\cdot,\cdot\rangle_M$.



minimization of functional $G(x)$ in equation (5.3), and can be determined by the largest eigenvalue $\lambda_{\max}$ of the operator K^*K. Consider the problem

$$\min_x \frac{1}{2} \|Kx\|_{\mathcal{Y}}^2, \tag{5.14}$$

which is minimized through the explicit gradient method. As can easily be seen, the minimizer of problem (5.14) is given by 0, provided K has full rank. Therefore, the iterates should converge to zero. The explicit gradient method for this problem is given by

$$x_{k+1} = x_k - \tau K^* K x_k = (I - \tau K^* K) x_k, \tag{5.15}$$

with the step size restriction $0 < \tau < 2/\lambda_{\max}$. The first iterate, i.e. the initial value $x_0 \in \mathcal{X}$, where $\mathcal{X}$ has finite dimension, $\dim(\mathcal{X}) = n$, can be represented in terms of the eigenvectors of K^*K:

$$x_0 = \sum_i v_i, \quad \text{with} \quad K^* K v_i = \lambda_i v_i, \tag{5.16}$$

where v_i denotes the (scaled) eigenvectors and λ_i the eigenvalues. The first gradient step then reads

$$x_1 = (I - \tau K^* K) \sum_i v_i = \sum_i (1 - \tau\lambda_i) v_i, \tag{5.17}$$

and the kth step is given by

$$x_k = \sum_i (1 - \tau\lambda_i)^k v_i. \tag{5.18}$$

Considering the step size restriction, it can again be seen, that $|1 - \tau\lambda_i| < 1$ for all i, and $|1 - \tau\lambda_i|^k \to 0$ for $k \to \infty$. Now, the crucial point is how fast $|1 - \tau\lambda_i|^k$ will converge to zero. Since the iterates x_k are represented through a basis of eigenvectors, we can see how fast every eigenvector v_i will converge to zero depending on how close to zero its corresponding eigenvalue $|1 - \tau\lambda_i|^k$ is:

$$x_k = (1 - \tau\lambda_0)^k v_0 + \ldots + (1 - \tau\lambda_{n-1})^k v_{n-1}. \tag{5.19}$$

Assume that the spectrum of the matrix K^*K ranges over several scales, i.e. there are eigenvalues close to zero and also large eigenvalues, such that the quotient $\lambda_{\max}/\lambda_{\min}$ is large, e.g. $\lambda_{\max}/\lambda_{\min} = \mathcal{O}(10^3)$ or more, which gives a criterion for stiffness. Further, assume that the eigenvalues are sorted, i.e. λ_0 is the largest eigenvalue and λ_{n-1} is the smallest. In Figure 5.1, the eigenvalues defined by $\lambda_i = \exp(-\frac{i}{4})$, $i = 0, \ldots, n-1$, are plotted together with $|1 - \tau\lambda_i|$ (left) and $|1 - \tau\lambda_i|^{100}$ (right) for $\tau = 1 = 1/\lambda_{\max}$ and for $\tau = 2 - \epsilon < 2/\lambda_{\max}$, $\epsilon \ll 1$, which reaches almost the step size restriction. We can see, that for large λ_i, the factor, $(1 - \tau\lambda_i)$, in front of the corresponding eigenvector v_i after 100 iterations is almost zero, thus for those i corresponding to the larger eigenvalues, the iteration converges fast. However, for the smaller eigenvalues of K^*K, the iteration almost does not converge at all. After 100 iterations, $|1 - \tau\lambda_i|^{100}$ is still almost one for some λ_i, i.e. the iterate x_k will consist of large parts of the v_i corresponding to small eigenvalues, while the v_i corresponding to large eigenvalues are damped to zero.

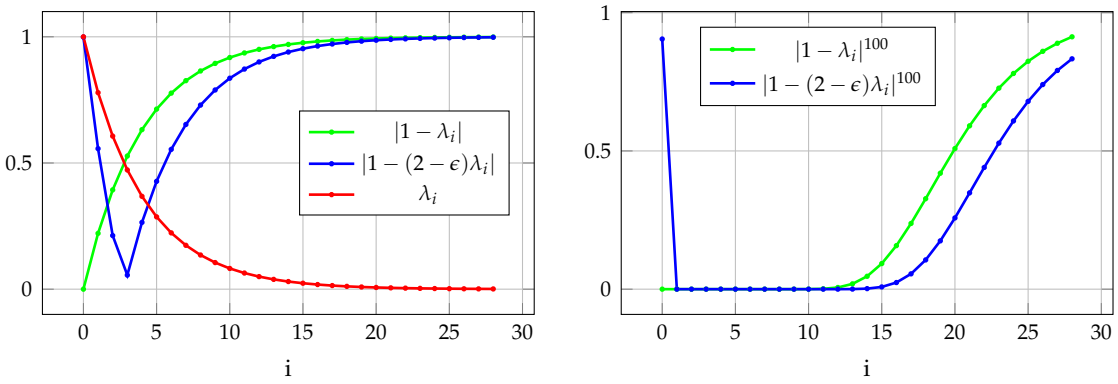


Figure 5.1: Eigenvalues λ_i of the matrix K^*K, and eigenvalues $(1-\tau\lambda_i)^k$ of the iteration matrix $(I-\tau K^*K)^k$ for $k=1$ (left) and $k=100$ (right) for step sizes $\tau \in \{1, 2-\epsilon\}$.

5.2 Implicit Proximal Method

In order to circumvent the step size restriction and obtain an unconditionally stable method we propose a fully implicit update step. Consider again the problem

$$\min_x F(x) + \frac{1}{2}\|Kx\|_{\mathcal{Y}}^2 + \langle b, x\rangle_{\mathcal{X}}, \tag{5.20}$$

where $F : \mathcal{X} \to (-\infty, \infty]$ is proper convex and lower semicontinuous, $K : \mathcal{X} \to \mathcal{Y}$ is a linear operator between Hilbert spaces and $b \in \mathcal{X}$. Our implicit proximal gradient algorithm is given by:

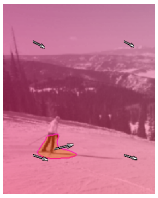
Algorithm 8 implicit proximal method

1. choose a step size $\tau > 0$ and an initial point $x_0 \in \mathcal{X}$
2. iterate for $k = 0, 1, \ldots$

$$x_{k+1} = \operatorname{prox}_{\tau F}^{(I+\tau K^*K)}\Big((I+\tau K^*K)^{-1}(x_k - \tau b)\Big), \tag{5.21}$$

with the scaled proximal operator from Definition 4.30.

Note, that there is no restriction on τ in this update.
At first we show, that a fixed point of the implicit proximal iteration (5.21) minimizes the objective functional (5.20). Then we show, that the fixed point iteration will converge to a fixed point.



5.2.1 Fixed Points

In order to show, that a fixed point of (5.21) minimizes (5.20), we rewrite the scaled proximal operator with the following lemma:

Lemma 5.3. *If there exists a decomposition $H = I + \tau K^*K$, then the scaled proximal operator can be written as*

$$\operatorname{prox}_{\tau F}^{H}\left(H^{-1}v\right) = \arg\min_x F(x) + \frac{1}{2}\|Kx\|_{\mathcal{Y}}^2 + \frac{1}{2\tau}\|x - v\|_{\mathcal{X}}^2 . \tag{5.22}$$

This can be seen as the proximal operator with function $G(x) = F(x) + \frac{1}{2}\|Kx\|_{\mathcal{Y}}^2$, i.e.

$$\operatorname{prox}_{\tau F}^{H}\left(H^{-1}v\right) = \operatorname{prox}_{\tau G}(v). \tag{5.23}$$

Proof. We show that the terms in (5.22) and (4.35) only differ by a constant, thus the following equation holds:

$$\tau\|Kx\|^2 + \|x - v\|^2 = \left\|x - H^{-1}v\right\|_H^2 + C, \tag{5.24}$$

with constant C, and the norm on the finite dimensional Hilbert space $\mathcal{X}$ weighted by the symmetric positive definite matrix H is given by $\|x\|_H = \langle x, Hx\rangle_{\mathcal{X}}^{1/2}$. The matrix $H = (I + \tau K^*K)$ is symmetric, i.e. $H^* = H$.

$$\begin{aligned}
\tau\|Kx\|^2 + \|x - v\|^2 &= \tau\langle x, K^*Kx\rangle + \langle x, x\rangle - 2\langle x, v\rangle + \langle v, v\rangle && (5.25)\\
&= \langle x, (I + \tau K^*K)x\rangle - 2\langle x, v\rangle + c_1 && (5.26)\\
&= \langle x, Hx\rangle - 2\langle x, v\rangle + c_1, && (5.27)
\end{aligned}$$

where c_1 is constant in x. Further,

$$\begin{aligned}
\left\|x - H^{-1}v\right\|_H^2 &= \left\langle x - H^{-1}v, Hx - v\right\rangle && (5.28)\\
&= \langle x, Hx\rangle - \langle x, v\rangle - \left\langle Hx, H^{-1}v\right\rangle + \left\langle v, H^{-1}v\right\rangle && (5.29)\\
&= \langle x, Hx\rangle - 2\langle x, v\rangle + c_2 && (5.30)
\end{aligned}$$

where c_2 is constant in x. □

With this lemma, we have that if $G(x) = F(x) + \frac{1}{2}\|Kx\|_{\mathcal{Y}}^2$ is proper convex and lower semicontinuous, the scaled proximal operator inherits the properties of the proximal operator, stated in the previous chapter. This follows directly from the properties of F. Now, we study fixed points of the scaled proximal operator.

Theorem 5.4. *A point $x_* \in \mathcal{X}$ solves the minimization problem (5.20), if and only if*

$$x_* = \operatorname{prox}_{\tau F}^{H}\left(H^{-1}(x_* - \tau b)\right), \tag{5.31}$$

with $H := (I + \tau K^*K)$, i.e. if x_* is a fixed point of the implicit proximal gradient iteration.

Proof. The optimality condition for x_* solving (5.20) is

$$0 \in \partial F(x_*) + K^*Kx_* + b \tag{5.32}$$
$$0 \in -\tau\partial F(x_*) + x_* - x_* - \tau K^*Kx_* - \tau b \tag{5.33}$$
$$x_* + \tau\partial F(x_*) + \tau K^*Kx_* \ni x_* - \tau b \tag{5.34}$$
$$x_* + \tau\partial\big(F(x_*) + \frac{1}{2}\|Kx_*\|^2\big) \ni x_* - \tau b \tag{5.35}$$
$$x_* \in (I + \tau\partial\hat{F})^{-1}(x_* - \tau b), \tag{5.36}$$

in the last step, we used the substitution $\hat{F}(x) := F(x) + \frac{1}{2}\|Kx\|^2$. With Lemma 5.3, the last equation is equivalent to

$$x_* = \operatorname{prox}_{\tau\hat{F}}(x_* - \tau b) = \operatorname{prox}^H_{\tau F}\Big(H^{-1}(x_* - \tau b)\Big). \tag{5.37}$$

□

5.2.2 Convergence

The implicit proximal gradient method can also be seen as a composition of the two operators

$$P := \operatorname{prox}^{(I+\tau K^*K)}_{\tau F}(\,\cdot\,), \text{ and} \tag{5.38}$$
$$T := (I + \tau K^*K)^{-1}(\,\cdot\, - \tau b). \tag{5.39}$$

i.e. the iteration reads

$$x_{k+1} = P \circ T(x_k). \tag{5.40}$$

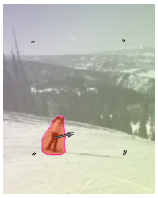
We have

Theorem 5.5. *Let $F : \mathcal{X} \to (-\infty, \infty]$ be proper convex and lower semicontinuous, $K : \mathcal{X} \to \mathcal{Y}$ a linear operator between Hilbert spaces and $b \in \mathcal{X}$. The implicit proximal gradient method*

$$x_{k+1} = \operatorname{prox}^{(I+\tau K^*K)}_{\tau F}\Big((I + \tau K^*K)^{-1}(x_k - \tau b)\Big),$$

converges to a fixed point $x_* \in \mathcal{X}$ as $k \to \infty$, for $\tau > 0$.

To show Theorem 5.5, we again analyze, if the operators P and T are averaged operators, cf. Theorem 4.10. With Lemma 5.3, the scaled operator can be written as a proximal operator with function $\hat{F}(x) := F(x) + \frac{1}{2}\|Kx\|^2$. $\hat{F}$ is convex proper and lower semicontinuous, thus the scaled proximal operator inherits the properties of the proximal operator. In particular, P is firmly nonexpansive and hence an averaged operator. Thus, it remains to show that T is an averaged operator.



Proof of Theorem 5.5. With Lemma 5.1 an averaged operator is averaged after translation with a constant. Therefore we only discuss the operator $T = (I + \tau K^*K)^{-1}(\,\cdot\,)$ in the next part.
Since K^*K is positive semidefinit, and $\|K^*K\| = \max\sigma(K^*K)$, we have that $\|K^*K\| \geq 0$. The spectrum of $(I + \tau K^*K)$ is given by

$$\sigma(I + \tau K^*K) = \{1 + \tau\lambda \mid \lambda \text{ is eigenvalue of } K^*K\}. \tag{5.41}$$

Further, the eigenvalues of H^{-1} can be determined through

$$(I + \tau K^*K)v = (1 + \tau\lambda)v \quad \Rightarrow \quad (I + \tau K^*K)^{-1}v = \frac{1}{1 + \tau\lambda}v. \tag{5.42}$$

Now, we rewrite $T = (1 - \alpha)I + \alpha N$ in the form of an averaged operator and show that N is nonexpansive for at least one $\alpha \in (0, 1)$. We use the substitution $H := (I + \tau K^*K)$ in the following.

$$H^{-1} = (1 - \alpha)I \underbrace{-(1 - \alpha)I + H^{-1}}_{=:\alpha N} \quad \Rightarrow \quad N = \frac{1}{\alpha}(H^{-1} - (1 - \alpha)I). \tag{5.43}$$

Since N is a linear operator, it is sufficient to determine for which alpha $\|N\| \leq 1$:

$$\|N\| = \frac{1}{\alpha}\left\|H^{-1} - (1 - \alpha)I\right\| = \frac{1}{\alpha}\left\|(I + \tau K^*K)^{-1} - (1 - \alpha)I\right\| \tag{5.44}$$

We examine the spectra of the matrices:

$$\sigma(K^*K) \geq 0 \quad \Rightarrow \quad \sigma(H) \geq 1 \quad \Rightarrow \quad 0 < \sigma((I + \tau K^*K)^{-1}) \leq 1 \tag{5.45}$$
$$\Rightarrow \quad -1 < \sigma((I + \tau K^*K)^{-1} - I) \leq 0 \tag{5.46}$$
$$\Rightarrow \quad \alpha - 1 < \sigma\big((I + \tau K^*K)^{-1} - I + \alpha I\big) \leq \alpha \tag{5.47}$$
$$\Rightarrow \quad 1 - \frac{1}{\alpha} < \frac{1}{\alpha}\sigma\big((I + \tau K^*K)^{-1} - (1 - \alpha)I\big) \leq 1 \tag{5.48}$$

We conclude that with

$$1 - \frac{1}{\alpha} \geq -1 \quad \Leftrightarrow \quad \alpha \geq \frac{1}{2}, \tag{5.49}$$

N is a nonexpansive operator for $\alpha \geq \frac{1}{2}$:

$$\|N\| \leq 1 \text{ if } \alpha \geq \frac{1}{2}. \tag{5.50}$$

Choosing $\alpha = 1/2$, N is a nonexpansive operator and thus T is an averaged operator and, with Theorem 4.15, T is a firmly nonexpansive operator. □

Remark 5.6 (Proof of Theorem 5.5 with Resolvent Argument). The operator $T := (I + \tau K^*K)^{-1}$ can also be seen as the resolvent $(I + \tau\nabla\hat{G})^{-1}$ for the function $\hat{G}(x) = \frac{1}{2}\|Kx\|^2$. Hence, for every proper convex and lower semicontinuous function $\hat{G}$, the resolvent $(I + \tau\nabla\hat{G})^{-1}$ is equivalent to the proximal operator $\text{prox}_{\tau\hat{G}}$, which is firmly nonexpansive, by Corollary 4.25, and thus an averaged operator, by Theorem 4.15. Thus, the implicit proximal gradient method converges also for more general proper, convex and lower semicontinuous functions $\hat{G}$.

5.2.3 Eigenvalue Analysis

Now, we analyze, how the iterates of the implicit method behave with respect to stiffness. Consider again the minimization problem

$$\min_x \frac{1}{2}\|Kx\|_Y^2 . \tag{5.14}$$

As done in section 5.1.3, we study the eigenvalues of the iteration matrix. The implicit iteration is given by

$$x_{k+1} = (I + \tau K^*K)^{-1}x_k, \tag{5.51}$$

for $\tau > 0$. The initial x_0 can again be represented in terms of eigenvectors of K^*K

$$x_0 = \sum_i v_i, \quad \text{with} \quad K^*Kv_i = \lambda_i v_i, \tag{5.52}$$

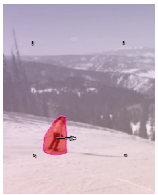
where v_i denotes the (scaled) eigenvectors and λ_i the eigenvalues. The first gradient step then reads

$$x_1 = (I + \tau K^*K)^{-1}\sum_i v_i = \sum_i (1 + \tau\lambda_i)^{-1}v_i, \tag{5.53}$$

and the k-th step is given by

$$x_k = \sum_i (1 + \tau\lambda_i)^{-k}v_i. \tag{5.54}$$

Again, we can examine, how fast the components of x_k corresponding to the ith eigenvector converge to zero. In Figure 5.2 the values for $|1 + \tau\lambda_i|^{-1}$ and $|1 + \tau\lambda_i|^{-100}$ are shown for $\tau \in \{1, 20\}$. The eigenvalues $\lambda_i = \exp(-\frac{i}{4})$, $i = 0, \dots, n-1$ are chosen as in Figure 5.1. We can observe, that the convergence for $\tau = 1 = 1/\lambda_{\max}$ has no advantage over the explicit gradient method for small λ_i, however, since there is no step size restriction, for $\tau = 20$ the convergence is much better, even for the smallest λ_i.



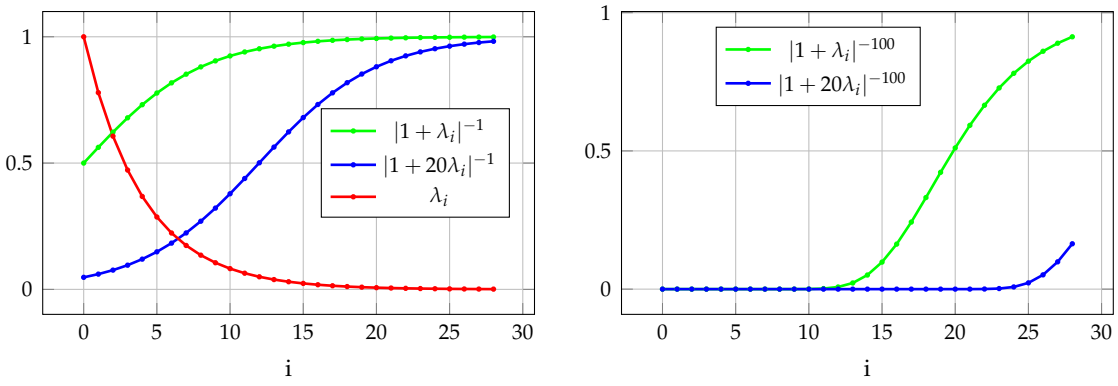


Figure 5.2: Eigenvalues λ_i of the matrix K^*K and the iteration matrix $(I+\tau K^*K)^{-k}$, i.e. $(1+\tau\lambda_i)^{-k}$ for $k=1$ (left) and $k=100$ (right), for step sizes $\tau \in \{1, 20\}$.

5.3 Inexact Evaluation

The scaled proximal operator is possibly difficult to evaluate, thus we propose an approximation of the scaled proximal operator with a non-scaled one. The implicit iteration then reads

$$x_{k+1} = \tilde{P}(T(x_k)), \tag{5.55}$$

with

$$\tilde{P} := \operatorname{prox}_{\tau F}(\,\cdot\,), \text{ and} \tag{5.38a}$$
$$T := (I+\tau K^*K)^{-1}(\,\cdot\, - \tau b), \tag{5.39}$$

where the proximal operator can be evaluated more easily and possibly explicitly. The inexact implicit proximal method is given by

Algorithm 9 inexact implicit proximal method

1. choose a step size $\tau > 0$ and an initial point $x_0 \in \mathcal{X}$
2. iterate for $k = 0, 1, \ldots$

$$x_{k+1} = \operatorname{prox}_{\tau F}\Big((I+\tau K^*K)^{-1}(x_k - \tau b)\Big). \tag{5.56}$$

At first we have to show the existence of a fixed point for the inexact iteration. Unfortunately, we could not find an optimization problem corresponding to the inexact fixed point iteration. Therefore, we analyze the existence of a fixed point for the inexact iteration and compare the inexact fixed point to the exact fixed point.

5.3.1 Existence of Fixed Points

The existence of a fixed point of the inexact iteration can be shown through the following fixed point theorem:

Theorem 5.7 (Browder-Göhde-Kirk). *Let $\mathcal{D}$ be a nonempty bounded closed convex subset of a real Hilbert space $\mathcal{H}$ and let $S : \mathcal{D} \to \mathcal{D}$ be a nonexpansive operator. Then the set of fixed points is not empty:*

$$\{x \in \mathcal{D} | S(x) = x\} \neq \emptyset. \tag{5.57}$$

For further details, see [Bauschke and Combettes, 2011, Chapter 4.3]. The operator in consideration is the concatenation of the operators $\tilde{P}$ and T, i.e. $S = \tilde{P} \circ T$. As shown in the previous section, T is an averaged operator, hence nonexpansive, and the proximal operator $\tilde{P}$ for the convex proper lower semicontinuous functional F is nonexpansive. Therefore, the restriction $T : \mathcal{D} \to \mathcal{H}$ is also nonexpansive for any $\mathcal{D} \subset \mathcal{H}$. Thus, it would suffice to find a convex closed set $\mathcal{D} \subset \mathcal{H}$ such that $S(\mathcal{D}) \subset \mathcal{D}$.
Suppose $\operatorname{dom}(F) = \mathcal{D}$ is a bounded convex set, then the requirements are fulfilled. If $\operatorname{dom}(F) = \mathcal{H}$, the domain $\operatorname{dom} F$ has to be restricted in a way, that for a sufficiently large closed convex bounded subset $\mathcal{D}$ the function is defined by e.g. $\tilde{F}(x) = F(x)$, if $x \in \mathcal{D}$, and $\tilde{F}(x) = \infty$, otherwise. This restriction does not affect the properties of F, i.e. $\tilde{F}$ is still proper, convex and lower semicontinuous, but the proximal operator of $\tilde{F}$ is nonexpansive with $\operatorname{prox}_{\tilde{F}} : \mathcal{H} \to D$, and thus the requirements of Theorem 5.7 are fulfilled. For practical purposes this restriction of F is reasonable, since in applications we usually are not interested in solutions with arbitrarily large norm.

5.3.2 Convergence to Exact Solution

In [Rockafellar, 1976], inexact evaluations of the resolvent operator in the proximal point algorithm are analyzed. The resolvents are allowed to be evaluated approximately, if the sum of all errors is finite. In [Eckstein and Bertsekas, 1992], these results are extended and carried over to splitting methods, which are special cases of the proximal point algorithm, in particular the Douglas–Rachford splitting and the ADMM. Their result, adapted to our setting, is given in the following theorem:

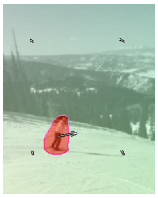
Theorem 5.8 ([Eckstein and Bertsekas, 1992]). *Let $F : \mathcal{X} \to (-\infty, \infty]$ be a convex proper lower semicontinuous function, $S_k = (I + \tau_k \partial F)^{-1}$ a firmly nonexpansive operator, and let (x_k) be such that*

$$x_{k+1} = \tilde{S}_k(x_k) \quad \forall k \geq 0, \quad \textit{where} \quad \left\| \tilde{S}_k(x_k) - S_k(x_k) \right\| \leq \varepsilon_k \quad \forall k \geq 0, \tag{5.58}$$

and (ε_k), (τ_k) are sequences such that

$$\sum_{k=0}^{\infty} \varepsilon_k < \infty, \quad \textit{and} \quad \inf_{k \geq 0} \tau_k > 0. \tag{5.59}$$

Then (x_k) converges weakly to a fixed point of S, if one exists.



The following theorem gives an estimate for the scaled and non-scaled proximal operator:

Theorem 5.9 ([Milzarek, 2015]). *The scaled proximal operator is Lipschitz continuous with respect to the scaling matrix H*

$$\left\|\text{prox}_F^{H_1}(x) - \text{prox}_F^{H_2}(x)\right\| \leq L \left\|H_1 - H_2\right\|_F, \tag{5.60}$$

where $\|\cdot\|_F$ is the Frobenius norm.

With this theorem, we can estimate that the error of the inexact iteration depends linearly on τ:

$$\left\|\text{prox}_F^{H}(x_k) - \text{prox}_F^{I}(x_k)\right\| \leq L \left\|H - I\right\|_F = \tau L \left\|K^*K\right\|_F \tag{5.61}$$

However, there is no sequence $(\tau_k) \to 0$, with $\sum_{k=0}^{\infty} \tau_k < \infty$ and $\inf_{k\geq 0} \tau_k > 0$. Thus, Theorem 5.9 does not yield a way to fulfill the requirements of Theorem 5.8. We have to assume that the fixed points of the exact and the inexact iteration are generally not identical. Therefore, in the next part, we analyze if the distance of the fixed points of the inexact iteration and the exact iteration is bounded.

5.3.3 Comparison of Fixed Points

As shown above, for a fixed $\tau > 0$ a fixed point of the inexact iteration, $\tilde{x}$, will exist, but will not necessarily be a fixed point of the correct iteration and thus, will not minimize (5.20). We compare the fixed points and examine if $\tilde{x}$ lies close to x, and thus, gives a good approximated result through an easy iteration.
The fixed points of the different iterations are given by

$$x = \text{prox}_{\tau F}^{H}\left(H^{-1}(x - \tau b)\right), \text{ and} \tag{5.62}$$

$$\tilde{x} = \text{prox}_{\tau F}\left(H^{-1}(\tilde{x} - \tau b)\right), \tag{5.63}$$

with $H = I + \tau K^*K$, symmetric positive definite. For better readability, we substitute the constant $\hat{b} = \tau H^{-1} b$.

Estimation. The estimate for the fixed points reads as

$$\begin{aligned}
\|x - \tilde{x}\| &= \left\|\text{prox}_{\tau F}^{H}\left(H^{-1}x - \hat{b}\right) - \text{prox}_{\tau F}\left(H^{-1}\tilde{x} - \hat{b}\right)\right\| && (5.64)\\
&\leq \left\|\text{prox}_{\tau F}^{H}\left(H^{-1}x - \hat{b}\right) - \text{prox}_{\tau F}\left(H^{-1}x - \hat{b}\right)\right\| && (5.65)\\
&\quad + \left\|\text{prox}_{\tau F}\left(H^{-1}x - \hat{b}\right) - \text{prox}_{\tau F}\left(H^{-1}\tilde{x} - \hat{b}\right)\right\| && (5.66)\\
&\overset{\text{NE}}{\leq} \left\|\text{prox}_{\tau F}^{H}\left(H^{-1}x - \hat{b}\right) - \text{prox}_{\tau F}\left(H^{-1}x - \hat{b}\right)\right\| + \left\|H^{-1}x - H^{-1}\tilde{x}\right\|. && (5.67)
\end{aligned}$$

We used the nonexpansiveness of the proximal operator in the last inequality. To estimate $\left\|\text{prox}_{\tau F}^{H}\left(H^{-1}x-\hat{b}\right)-\text{prox}_{\tau F}\left(H^{-1}x-\hat{b}\right)\right\|$ in (5.67), we use Theorem 5.9. Since we have no information on the constant L, we follow the ideas from [Milzarek, 2015,Lee et al., 2012] to obtain a refined estimate of the first term in (5.67):

For $y = H^{-1}x-\hat{b}$, we define $p := \text{prox}_{\tau F}^{H}(y)-y$ and $\tilde{p} := \text{prox}_{\tau F}(y)-y$. Through the optimality conditions of the scaled proximal operator and the proximal operator at $p+y$ and $\tilde{p}+y$, respectively, p and $\tilde{p}$ can be written as an element of the subgradient of F:

$$p+y = \text{prox}_{\tau F}^{H}(y) = \arg\min_{x} F(x)+\frac{1}{2\tau}\|x-y\|_H^2 \tag{5.68}$$

$$\tilde{p}+y = \text{prox}_{\tau F}(y) = \arg\min_{x} F(x)+\frac{1}{2\tau}\|x-y\|^2 \tag{5.69}$$

The optimality conditions are given by

$$0 \in \tau\partial F(p+y)+H(p+y-y) \quad \Leftrightarrow \quad -\frac{1}{\tau}Hp \in \partial F(p+y) \tag{5.70}$$

$$0 \in \tau\partial F(\tilde{p}+y)+\tilde{p}+y-y \quad \Leftrightarrow \quad -\frac{1}{\tau}\tilde{p} \in \partial F(\tilde{p}+y) \tag{5.71}$$

Due to the definition of the subgradient in Definition 4.5, one derives the following inequalities:

$$F(p+y)-\frac{1}{\tau}\langle Hp,\tilde{p}-p\rangle \leq F(\tilde{p}+y) \tag{5.72}$$

$$F(\tilde{p}+y)-\frac{1}{\tau}\langle \tilde{p},p-\tilde{p}\rangle \leq F(p+y) \tag{5.73}$$

These inequalities are summed and rearranged to give

$$\langle \tilde{p},\tilde{p}-p\rangle \leq \langle Hp,\tilde{p}-p\rangle \tag{5.74}$$

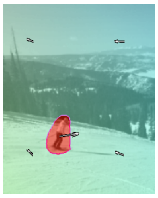
Completing the square on the left side by adding $\langle p,p-\tilde{p}\rangle$ and applying the Cauchy–Schwarz inequality as done in [Milzarek, 2015] yields

$$\|p-\tilde{p}\|^2 \leq \langle (H-I)p,p-\tilde{p}\rangle \overset{\text{C-S}}{\leq} \|(H-I)p\|\cdot\|p-\tilde{p}\| \tag{5.75}$$

Substituting the definitions of p and $\tilde{p}$ we have

$$\|p-\tilde{p}\| = \left\|\text{prox}_{\tau F}^{H}\left(H^{-1}x-\hat{b}\right)-\text{prox}_{\tau F}\left(H^{-1}x-\hat{b}\right)\right\| \tag{5.76}$$

$$\leq \left\|(H-I)\left(\text{prox}_{\tau F}^{H}\left(H^{-1}x-\hat{b}\right)-H^{-1}x+\hat{b}\right)\right\|. \tag{5.77}$$



Next, we use the fixed point equation, (5.62), i.e. $x = \text{prox}^H_{\tau F}\left(H^{-1}x - \hat{b}\right)$, which gives

$$\left\|\text{prox}^H_{\tau F}\left(H^{-1}x - \hat{b}\right) - \text{prox}_{\tau F}\left(H^{-1}x - \hat{b}\right)\right\| \leq \left\|(H-I)((I-H^{-1})x + \hat{b})\right\| \tag{5.78}$$

$$= \left\|(H-I)^2H^{-1}x + (H-I)\hat{b}\right\| \tag{5.79}$$

We now return to the comparison of the fixed points from inequality (5.67). With the estimate (5.79) and the substitutions $H = I + \tau K^*K$, and $\hat{b} = \tau H^{-1}b$ we have

$$\|x - \tilde{x}\| \leq \left\|\tau^2(K^*K)^2H^{-1}x + \tau^2K^*KH^{-1}b\right\| + \left\|H^{-1}x - H^{-1}\tilde{x}\right\|, \tag{5.80}$$

Let λ_i denote the eigenvalues of K^*K. The eigenvalues of $(K^*K)^2H^{-1}$ are given by $\frac{\lambda_i^2}{1+\tau\lambda_i}$, and the maximal value is given by $\max\left\{\sigma((K^*K)^2H^{-1})\right\} = \frac{\lambda_{\max}^2}{1+\tau\lambda_{\max}}$, where $\lambda_{\max}$ denotes the largest eigenvalue of K^*K. Then we have

$$\|x - \tilde{x}\| \leq \frac{\tau^2\lambda_{\max}^2}{1+\tau\lambda_{\max}}\Big(\|x\| + \underbrace{\frac{1}{\lambda_{\max}}\|b\|}_{=:c_1}\Big) + \left\|H^{-1}(x-\tilde{x})\right\| \tag{5.81}$$

The eigenvalues of the matrix H^{-1} are given by $\frac{1}{1+\tau\lambda}$. We assume for now, that the smallest eigenvalue of K^*K, denoted by $\lambda_{\min}$ is not zero, i.e. $\lambda_{\min} > 0$. This yields an upper bound for the relative error of the fixed points depending on the eigenvalues of the matrix K^*K with the constant c_1:

$$\|x - \tilde{x}\| \leq \frac{\tau^2\lambda_{\max}^2}{1+\tau\lambda_{\max}}(\|x\| + c_1) + \frac{1}{1+\tau\lambda_{\min}}\|x - \tilde{x}\| \tag{5.82}$$

$$(1+\tau\lambda_{\min})\|x - \tilde{x}\| \leq \tau^2\lambda_{\max}^2\frac{1+\tau\lambda_{\min}}{1+\tau\lambda_{\max}}(\|x\| + c_1) + \|x - \tilde{x}\| \tag{5.83}$$

This yields

$$\frac{\|x - \tilde{x}\|}{\|x\|} \leq \tau\,\frac{\lambda_{\max}^2}{\lambda_{\min}}\,\frac{1+\tau\lambda_{\min}}{1+\tau\lambda_{\max}}\left(1 + \frac{c_1}{\|x\|}\right) \tag{5.84}$$

This relative error of the fixed points depends linearly on the step size τ. In the worst case, for $\lambda_{\min} = 0$, the constant $\frac{\lambda_{\max}}{\lambda_{\min}}$ will be infinity. But one can bring forward the argument that this is only the case if $(x - \tilde{x})$ is an element of the kernel of K^*K, i.e. $(x - \tilde{x}) \in \text{Ker}(K^*K)$ and is thus collinear with the eigenvector corresponding to the eigenvalue $\lambda_{\min} = 0$. In the next paragraph, this is studied further.

Eigenvalue Analysis. To get a better estimate, especially when $\lambda_{\min} = 0$, we investigate the eigenvalues and associated eigenvectors of H^{-1} in (5.81) and estimate the part of $(x - \tilde{x})$ that is collinear with the eigenvector of H^{-1} corresponding to eigenvalue 0.

We treat the first part of (5.81) as a constant $c = \frac{\tau^2\lambda_{\max}^2}{1+\tau\lambda_{\max}}(\|x\| + c_1)$ for now and substitute $z = x - \tilde{x}$

$$\|z\| \leq c + \left\|H^{-1}z\right\|. \tag{5.85}$$

The spectrum of $H^{-1} \in \mathbb{R}^{n\times n}$ satisfies $\sigma(H^{-1}) \subset (0, 1]$. The eigenvalue 1 corresponds to the eigenvalue 0 of the matrix K^*K. Let $v \in \mathbb{R}^n$ be an eigenvector to eigenvalue 1, and let $\|v\| = 1$, then $H^{-1}v = v$. Further, let $v_\perp$ denote the orthogonal subspace to v, with $\dim(v_\perp) = n - 1$ and $v \perp v_\perp$. Then, z can be represented through the eigenvectors

$$\mathbb{R}^n = v \oplus v_\perp \quad \Leftrightarrow \quad \forall z \in \mathbb{R}^n : \quad \exists \eta, \eta_\perp \in \mathbb{R} \text{ and } w \in v_\perp : z = \eta v + \eta_\perp w. \tag{5.86}$$

Without loss of generality, we assume, that $\|w\| = 1$ and thus $\|z\|^2 = \eta^2 + \eta_\perp^2$. Next, we consider the quadratic norm:

$$\left\|H^{-1}z\right\|^2 = \left\|H^{-1}(\eta v + \eta_\perp w)\right\|^2 = \|\eta v\|^2 + \left\|\eta_\perp H^{-1}w\right\|^2 = \eta^2 + \eta_\perp^2 \left\|H^{-1}w\right\|. \tag{5.87}$$

Since w is by definition orthogonal to v, the last norm can be estimated with the second largest eigenvalue of H^{-1}, denoted by $\Lambda_2 < 1$, and substitute $\eta_\perp^2 = \|z\|^2 - \eta^2$

$$\left\|H^{-1}z\right\|^2 = \eta^2 + \eta_\perp^2 \left\|H^{-1}w\right\| \leq \eta^2 + \eta_\perp^2\Lambda_2^2 \tag{5.88}$$

$$= \eta^2 + (\|z\|^2 - \eta^2)\Lambda_2^2 = \Lambda_2^2\|z\|^2 + (1 - \Lambda_2^2)\eta^2. \tag{5.89}$$

Now, we get back to the original estimate (5.81) and rewrite it as follows

$$c \geq \|z\| - \left\|H^{-1}z\right\| = \|z\| - \sqrt{\Lambda_2^2\|z\|^2 + (1 - \Lambda_2^2)\eta^2} \tag{5.90}$$

$$= \left(1 - \sqrt{\Lambda_2^2 + (1 - \Lambda_2^2)\frac{\eta^2}{\|z\|^2}}\right)\|z\|. \tag{5.91}$$

We consider the term $\eta^2/\|z\|^2$. By definition, $\eta^2 \leq \|z\|^2$, however, if z is not collinear with v, then $\eta^2 < \|z\|^2$. Thus, the quotient will be smaller than one, $\eta^2/\|z\|^2 < 1$. Further, if the quotient is very small, i.e. $\eta^2/\|z\|^2 \ll 1$ the estimate can be written as

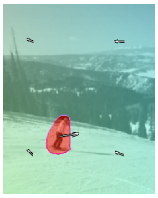
$$\|z\| \lesssim \frac{c}{1 - \Lambda_2}. \tag{5.92}$$

The second largest eigenvalue of H^{-1} can be calculated from the second smallest eigenvalue of K^*K, denoted by λ_2, and $\Lambda_2 = \frac{1}{1+\tau\lambda_2}$. Substituting the constant

$$c = \frac{\tau^2\lambda_{\max}^2}{1 + \tau\lambda_{\max}}(\|x\| + c_1), \tag{5.93}$$

estimate (5.92) reads in terms of λ_2

$$\|z\| = \|x - \tilde{x}\| \lesssim \tau\,\frac{\lambda_{\max}^2}{\lambda_2}\,\frac{1 + \tau\lambda_2}{1 + \tau\lambda_{\max}}(\|x\| + c_1). \tag{5.94}$$



This calculation can be repeated with the same arguments for larger eigenvalues than λ_2. Let z be the partition of the eigenvectors v_i, where $\|v_i\| = 1$:

$$z = \sum_i \eta_i v_i. \tag{5.95}$$

If the first $\eta_1, \dots, \eta_{k-1}$ are all very small, i.e. the part of z in direction of v_i, with $i = 1, \dots, k-1$, is small, then estimate (5.94) can be written in terms of λ_k:

$$\|x - \tilde{x}\| \lesssim \tau \frac{\lambda_{\max}^2}{\lambda_k} \frac{1 + \tau\lambda_k}{1 + \tau\lambda_{\max}} \left(\|x\| + \frac{\|b\|}{\lambda_{\max}} \right). \tag{5.96}$$

However, for which k this estimate holds, depends on $(x - \tilde{x})$.
Further refinements depend on the eigenvalues and eigenvectors of the matrix, and therefore require further knowledge of the eigenvalues of the matrix K^*K. We will analyze this estimate of the fixed points for the motion segmentation model in chapter 6.

The result of this section is summarized in the following proposition:

Proposition 5.10. *Let x_* be a fixed point of the exact implicit proximal method (5.21), and let $\tilde{x}_*$ be a fixed point of the inexact implicit proximal method (5.55). Suppose there exists an $\epsilon > 0$ and a decomposition $x - \tilde{x} = v + w$, where $v \in \ker(K^*K)$, and $w \perp \ker(K^*K)$, such that $\|v\| < \|\tilde{x} - x\| - \epsilon$ for all τ, then*

$$\|x - \tilde{x}\| = \mathcal{O}(\tau), \quad \text{as } \tau \to 0. \tag{5.97}$$

5.4 ADMM with Implicit Update

The implicit proximal gradient method can be included into the update step of different splitting methods. Here we replace the x-update step of the ADMM by the new method, using a more general formulation of the method, where the differentiable parts of the objective function are split into stiff and non-stiff parts. We again consider the subproblem for the minimization of the augmented Lagrangian with respect to x, (4.74):

$$\min_x G(x) + \frac{\sigma}{2}\|Kx - z + y\|_{\mathcal{Y}}^2.$$

We define $G = G_1 + G_2$, where G_2 is differentiable and ∇G_2 is non-stiff, and G_1 is non-differentiable. The problem now reads:

$$\min_x G_1(x) + G_2(x) + \frac{\sigma}{2}\|Kx - z + y\|_{\mathcal{Y}}^2. \tag{5.98}$$

We split[2] the (sub-)gradient into stiff, non-stiff and non differentiable terms for the implicit proximal algorithm from equation (5.21):

$$\underbrace{\partial G_1(x)}_{\text{non-diff.}} + \underbrace{\sigma K^*Kx}_{\text{stiff}} + \underbrace{\sigma K^*(-z + y) + \nabla G_2(x)}_{\text{non-stiff}}, \tag{5.99}$$

[2]Before this step, we have to ensure that the requirements of proposition 4.19 are fulfilled.

which can be written as

$$\partial\left(G_1(x) + \frac{\sigma}{2}\|Kx\|_{\mathcal{Y}}^2\right) + \sigma K^*(-z+y) + \nabla G_2(x). \tag{5.100}$$

The resulting update step reads

$$\begin{aligned} x_{k+1} &= \left(I + \tau\partial(G_1 + \frac{\sigma}{2}\|K(\cdot)\|_{\mathcal{Y}}^2)\right)^{-1}(x_k - \tau(\sigma K^*(y_k - z_{k+1}) + \nabla G_2(x_k))) && (5.101)\\ &= \operatorname{prox}_{\tau(G_1 + \frac{\sigma}{2}\|K(\cdot)\|_{\mathcal{Y}}^2))}\left(x_k - \tau(\sigma K^*(y_k - z_{k+1}) + \nabla G_2(x_k))\right) && (5.102)\\ &= \operatorname{prox}_{\tau\hat{F}}\left(x_k - \tau(\sigma K^*(y_k - z_{k+1}) + \nabla G_2(x_k))\right), && (5.103) \end{aligned}$$

where $\hat{F}(x) = G_1(x) + \frac{\sigma}{2}\|Kx\|_{\mathcal{Y}}^2$. In terms of the scaled proximal operator for function G_1, we obtain the following algorithm

Algorithm 10 implicit ADMM

1. choose a step size $\tau > 0$, an augmentation parameter $\sigma > 0$, and an initial point $(x_0, y_0) \in \mathcal{X} \times \mathcal{Y}$, $z_0 = Kx_0 \in \mathcal{Y}$.

2. iterate for $k = 0, 1, \ldots$

$$x_{k+1} = \operatorname{prox}_{\tau G_1}^{(I+\tau\sigma K^*K)}\left((I + \tau\sigma K^*K)^{-1}\left(x_k - \tau(\sigma K^*(y_k - z_k) + \nabla G_2(x_k))\right)\right) \tag{5.104}$$

$$z_{k+1} = \operatorname{prox}_{\sigma^{-1}F}\left(Kx_{k+1} + y_k\right) \tag{5.105}$$

$$y_{k+1} = y_k + Kx_{k+1} - z_{k+1} \tag{5.106}$$

Note that again, since we treat the operator K^*K implicitly, there is no major restriction on the step sizes τ and the penalizing parameter σ in order to guarantee convergence.

Convergence. The update step with the scaled proximal operator in (5.104) is exact, thus the convergence results given in Theorem 4.43 hold for the fully implicit ADMM. If the scaled proximal operator cannot be evaluated explicitly, an approximation has to be used. The convergence results for the ADMM state, that the ADMM also converges for suitable inexact minimization in the update steps, cf. [Boyd et al., 2011].

Inexact Convergence. Convergence to the same fixed point with inexact minimization requires, that the sum of the errors between the exact and inexact iterates for a fixed step size τ is finite, cf. [Eckstein and Bertsekas, 1992]. These requirements are similar to those cited in section 5.3.2, and we have already seen, that one cannot expect the algorithm to converge to the same fixed point as the exact iteration. The implicit ADMM with an inexact update is given in algorithm 11.

Algorithm 11 inexact implicit ADMM

1. choose a sequence of step sizes $\tau_k > 0$ or a fixed step size $\tau > 0$, an augmentation parameter $\sigma > 0$, and an initial point $(x_0, y_0) \in \mathcal{X} \times \mathcal{Y}$, $z_0 = Kx_0 \in Y$.

2. iterate for $k = 0, 1, \ldots$

$$x_{k+1} = \text{prox}_{\tau_k G_1}\left((I + \tau_k \sigma K^* K)^{-1}\left(x_k - \tau_k(\sigma K^*(y_k - z_k) + \nabla G_2(x_k))\right)\right) \quad (5.107)$$

$$z_{k+1} = \text{prox}_{\sigma^{-1}F}\left(Kx_{k+1} + y_k\right) \quad (5.108)$$

$$y_{k+1} = y_k + Kx_{k+1} - z_{k+1} \quad (5.109)$$

As shown in section 5.3, a fixed point of the inexact iteration exists, to which the algorithm will converge. In chapter 6, we will see, that for larger step sizes τ, the approximated fixed point $\tilde{x}$ will still give reasonable results for the motion segmentation example. But due to the approximation error growing with the step size τ, the values for τ and σ will at some point be restricted by accuracy requirements.
However, an adaptive step size strategy can be chosen, where τ_k decreases during the iteration. A sequence τ_k, where for small k the step size τ_k is large, and, once the iteration starts to converge, τ_k decreases, could have the advantage of fast convergence, and its final solution could be close to the fixed point of the exact iteration.
Further, the feasibility of the implicit proximal method will depend on whether the resulting linear systems involving the operator K^*K will be efficiently solvable. For total variation regularized problems this is true, because the linear systems can be solved through a fast Fourier transform.

6 Numerical Results and Evaluation

In this chapter, we apply the different algorithms and variations, presented in the previous chapters, to the motion segmentation problem and compare performance, with respect to speed and robustness.
In the first part, we discuss the existence of minimizers and how the presented methods are applicable on the motion segmentation models given in chapter 3. The tested algorithms are the primal-dual algorithm, the ADMM with Gauss–Seidel update and the new implicit ADMM. Also, implementational details of the different algorithms are given, and a fast method to solve the linear system arising from the implicit step is provided.
Further, we analyze performance of the different algorithms on the two-label model in comparison to each other, considering the theory derived in chapters 4 and 5. We examine the influence of different step sizes on the performance of the implicit ADMM. In the end, we show the performance on the multi label model, since it is based directly on the two-label model.
Finally, we compare the primal-dual algorithm with the implicit ADMM on the inpainting and the denoising model.

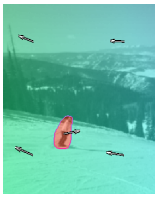
6.1 Algorithms Applied to Motion Segmentation

We now apply the algorithms from chapters 4 and 5 to the two-label motion segmentation model from equation (3.3).
The labeling function is denoted by $u : \Omega \to [0,1]$ and the two motion vectors are called $v = (v_1, v_2)$. The operator K from chapters 4 and 5 is given by a discrete approximation of the gradient operator, with ∇, and the adjoint, K^*, is given by a discrete approximation of the divergence operator, $-\operatorname{div}$, cf. section 6.1.4. The functional $F : \mathcal{Y} \to [0, \infty]$ together with the operator K defines the total variation regularization of the labeling function u,

$$F(Ku) = \int_\Omega \|\nabla u(x)\|_2 \, \mathrm{d}x. \tag{6.1}$$

The functional $G : \mathcal{X} \to (-\infty, \infty]$ represents the data term penalizing the squared optical flow constraint:

$$G(u) = \mu \int_\Omega \langle \nabla f(x,t), v_1 \rangle^2 \, u(x) + \langle \nabla f(x,t), v_2 \rangle^2 \left(1 - u(x)\right) \mathrm{d}x, \tag{6.2}$$



The objective functional is given by

$$J(u,v) = \int_\Omega \|\nabla u(x)\|_2 + \mu \left(\langle \nabla f(x,t), v_1 \rangle^2 u(x) + \langle \nabla f(x,t), v_2 \rangle^2 \left(1 - u(x)\right) \right) \, \mathrm{d}x, \tag{6.3}$$

and the minimization problem is given by

$$\min_{u(x)\in[0,1],v} J(u,v). \tag{6.4}$$

The constraint $u(x) \in [0,1]$ is included through the indicator function $\delta_{\mathcal{C}}$ on the convex hypercube $\mathcal{C} = [0,1]^{|\Omega|}$, as given in section 4.2.2:

$$\delta_{\mathcal{C}}(u) = \begin{cases} 0 & \text{if } u \in [0,1]^{|\Omega|} \\ \infty & \text{otherwise.} \end{cases} \tag{6.5}$$

Thus, the unconstrained minimization problem reads

$$\min_{u,v} J(u,v) + \delta_{\mathcal{C}}(u). \tag{6.6}$$

The optimization problem is solved by alternating minimization with respect to u and v, i.e. v_n is fixed and $u_n = \arg\min_u J(u, v_n)$ is solved. Afterwards $v_{n+1} = \arg\min_v J(u_n, v)$ is computed while u_n is fixed. The minimizer of $\min_v J(u_n, v)$ (and u_n fixed) is explicitly given by solving two 2×2-systems (cf. section 6.1.1) and since $J(v, u_n)$ is convex in v, the solution is the global minimizer with respect to v for a fixed u_n. The minimization with respect to u, given in section 6.1.2, is carried out through the optimization algorithms from chapters 4 and 5.
Since the minimization with respect to u for fixed v is sometimes also carried out by an iterative algorithm, we call the alternating minimization between u and v the *outer iteration* denoted with indices n, while the iteration for solving $\arg\min_u J_v(u)$ with fixed v is called the *inner iteration*, denoted with indices k.

6.1.1 Motion Vectors

While minimizing $J_u(v) := J(u_n, v)$ with respect to v for fixed u_n the terms independent of v can be omitted, since the value of the objective function is irrelevant, and the resulting objective functional is given by

$$J_u(v) = \int_\Omega \mu \langle \nabla f(x,t), v_1 \rangle^2 u_n(x) + \mu \langle \nabla f(x,t), v_2 \rangle^2)(1 - u_n(x)) \, \mathrm{d}x. \tag{6.7}$$

The unique minimizer of $J_u(v)$ with respect to $v = (v_1, v_2) \in \mathbb{R}^2 \times \mathbb{R}^2$ is calculated through

$$v_{n+1} = \arg\min_v J_u(v). \tag{6.8}$$

The solution of (6.8) is given by the linear system

$$A_u \hat{v}_1 = b_u \tag{6.9}$$

$$(A - A_u)\hat{v}_2 = b - b_u, \tag{6.10}$$

yielding $v_1 = (\hat{v}_1, 1)^T$ and $v_2 = (\hat{v}_2, 1)^T$, where

$$A = \begin{bmatrix} \int_\Omega (f_{x_1})^2 & \int_\Omega f_{x_1} f_{x_2} \\ \int_\Omega f_{x_2} f_{x_1} & \int_\Omega (I_{x_2})^2 \end{bmatrix}, \quad b = \begin{bmatrix} \int_\Omega f_{x_1} f_t \\ \int_\Omega f_{x_2} f_t \end{bmatrix}$$

and

$$A_u = \begin{bmatrix} \int_\Omega u (f_{x_1})^2 & \int_\Omega u f_{x_1} f_{x_2} \\ \int_\Omega u f_{x_2} f_{x_1} & \int_\Omega u (f_{x_2})^2 \end{bmatrix}, \; b_u = \begin{bmatrix} \int_\Omega u f_{x_1} f_t \\ \int_\Omega u f_{x_2} f_t \end{bmatrix},$$

where f_{x_1}, f_{x_2}, f_t denote the partial derivatives of the image, $\partial_{x_1} f, \partial_{x_2} f, \partial_t f$, respectively. The linear system can be solved analytically and can be computed very fast. For further speedup, the matrix A and vector b can be precalculated and the matrix A_u and b_u can be calculated from previous steps with increments $du = u_{n+1} - u_n$:

$$A_{u_{n+1}} = A_{u_n} + A_{du}. \tag{6.11}$$

Since the increments du will become sparse during the iteration, this can speed up the calculation.

6.1.2 Labeling Function

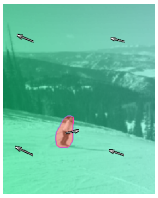
For the computation of $\arg\min_u J(u, v_n)$, the constant term can also be omitted and the objective functional reduces to

$$J_v(u) = \int_\Omega \|\nabla u(x)\|_2 + \mu \left(\langle \nabla f(x,t), (v_n)_1 \rangle^2 - \langle \nabla f(x,t), (v_n)_2 \rangle^2 \right) u(x) \, \mathrm{d}x + \delta_{\mathcal{C}}(u) \tag{6.12}$$

$$= \int_\Omega \|\nabla u(x)\|_2 + \mu s(v_n) u(x) \, \mathrm{d}x + \delta_{\mathcal{C}}(u), \tag{6.13}$$

where $s(v_n) = \langle \nabla f(x,t), (v_n)_1 \rangle^2 - \langle \nabla f(x,t), (v_n)_2 \rangle^2$ is the optical flow error. In the following, the optical flow error for a fixed v_n is denoted by $s_v := s(v_n)$.

Requirements. First, we check whether the requirements of the theory presented in chapter 4 are fulfilled. Therefore, we have to check, if the functionals F and G are proper, convex and lower semicontinuous, if the condition in Proposition 4.19, $0 \in \operatorname{sri}(\operatorname{dom} G - K(\operatorname{dom} F))$, is satisfied, and if a minimizer of $J_v(u)$ exists.



The functional term corresponding to $F \circ K : \mathcal{Y} \to [0, \infty]$ is given by the total variation regularization and $G : \mathcal{X} \to (-\infty, \infty]$ is the data term. The Hilbert spaces $\mathcal{X}, \mathcal{Y}$ can be associated with $\mathbb{R}^N$ and $\mathbb{R}^{2N}$ respectively, with the corresponding Euclidean scalar product and norm $\|\cdot\|_{\mathcal{X}}$ and $\|\cdot\|_{\mathcal{Y}}$. The pixelwise Euclidean norm is denoted by $\|\cdot\|_2$. As given in chapter 2, the total variation regularization is convex, proper and lower semicontinuous. The operator $K : \mathcal{X} \to \mathcal{Y}$, defined through a discrete approximation of the gradient, is a bounded linear operator. The function $G(u)$ consists of two parts, the linear term $\mu s_v u(x)$, and the indicator function $\delta_{\mathcal{C}}(u)$. Both terms are clearly convex, proper and lower semicontinuous.
We have

$$F(u) = \|u\|_{\mathcal{Y}} \quad \Rightarrow \quad \operatorname{dom}(F) = \mathcal{Y} = \mathbb{R}^{2N} \tag{6.14}$$

$$G(u) = \int_\Omega \mu s_v u \, \mathrm{d}x + \delta_{\mathcal{C}}(u) \quad \Rightarrow \quad \operatorname{dom}(G) = \mathcal{C} \subset \mathbb{R}^N. \tag{6.15}$$

The domain of F is the whole space $\mathbb{R}^{2N}$, and the domain of G is not empty, therefore $\operatorname{dom} F - K(\operatorname{dom} G)) = \mathbb{R}^{2N}$, hence $0 \in \operatorname{int}(\operatorname{dom} F - K(\operatorname{dom} G))$ and thus Proposition 4.20 yields $0 \in \operatorname{sri}(\operatorname{dom} F - K(\operatorname{dom} G))$.
Further, $J_v(u)$ should have a minimizer on the hypercube $\mathcal{C} = [0,1]^{|\Omega|}$. Since $\mathcal{C}$ is a closed, convex and bounded subset of $\mathcal{X}$, with Theorem 4.4, $J_v(u)$ has a minimizer on $\mathcal{C}$.

Thus, all requirements are fulfilled, and we can apply the primal-dual algorithm, the ADMM with a Gauss–Seidel update and the implicit ADMM to the motion segmentation model.

The primal-dual algorithm for functional J_v. In order to apply the primal-dual algorithm to the motion segmentation model, we use the primal-dual formulation

$$\min_u \max_w \langle \nabla u, w \rangle_{\mathcal{Y}} + G(u) - F^*(w), \tag{6.16}$$

with $F(u) = \int_\Omega \|\nabla u(x)\|_2 \, \mathrm{d}x$, thus the convex conjugate F^* is the indicator function of the dual norm unit ball. The functional $G(u)$ is given by $G(u) = \int_\Omega \mu s_v u(x) \, \mathrm{d}x + \delta_{\mathcal{C}}(u)$. The optimality conditions for a saddle point (u_*, w_*) are given by

$$0 \in \nabla u_* - \partial F^*(w_*) \tag{6.17}$$

$$0 \in \partial \delta_{\mathcal{C}}(u_*) + \mu s_v + \operatorname{div}(w_*), \tag{6.18}$$

where the subgradient of $G(u)$ is given by $\partial G(u) = \partial \delta_{\mathcal{C}}(u) + \mu s_v$. The update for w is performed by the proximal gradient ascent method, and the update for u is calculated through the proximal gradient descent method. The primal-dual algorithm with

extrapolation step from algorithm 7 is given by

$$w_{k+1} = \text{prox}_{\sigma F^*}\left(w_k + \sigma \nabla \bar{u}_k\right) \tag{6.19}$$
$$u_{k+1} = \text{prox}_{\tau \delta_{\mathcal{C}}}\left(u_k + \tau \text{div}(w_{k+1}) - \tau \mu s_v\right) \tag{6.20}$$
$$\bar{u}_{k+1} = u_{k+1} + \theta(u_{k+1} - u_k), \tag{6.21}$$

for step sizes $\tau, \sigma > 0$ and extrapolation parameter $\theta \in [0, 1]$.
The update of w can be easily computed through a so called *shrinkage* or soft thresholding, $S_\lambda : \mathcal{Y} \to \mathcal{Y}$ with parameter λ, cf. [Esser, 2009]. Consider the problem

$$S_\lambda(z) = \text{prox}_{\lambda\|\cdot\|_{\mathcal{Y}}}(z) = \arg\min_w \int_\Omega \left(\|w\|_2 + \frac{1}{2\lambda}\|w - z\|_2 \right)^2, \tag{6.22}$$

then the solution is given by

$$S_\lambda(z) = \begin{cases} (\|z\|_2 - \lambda)\frac{z}{\|z\|_2} & \text{if } \|z\|_2 > \lambda \\ 0 & \text{otherwise.} \end{cases} \tag{6.23}$$

This can be written as

$$S_\lambda(z) = \max\{0, \|z\|_2 - \lambda\}\frac{z}{\|z\|_2}, \tag{6.24}$$

and, such that there are no issues with $\|z\|_2 = 0$, we can rewrite it as

$$S_\lambda(z) = \left(1 - \min\left\{1, \frac{\lambda}{\|z\|_2}\right\}\right) z = \left(1 - \frac{1}{\max\{1, \frac{1}{\lambda}\|z\|_2\}}\right) z, \tag{6.25}$$

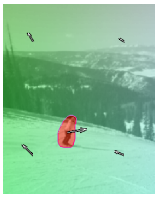
which has implementational advantages over (6.24). With Moreau's identity given in equation 4.34, we can derive the shrinkage formula for (6.19) and thus the proximal operator for the conjugate function F^*:

$$\text{prox}_{\sigma F^*}\{\underbrace{w_k + \sigma \nabla \bar{u}_k}_{z}\} = z - \sigma \text{prox}_{\frac{1}{\sigma}F}\left(\frac{z}{\sigma}\right) = z - \sigma\left(1 - \frac{1}{\max\{1, \sigma\left\|\frac{z}{\sigma}\right\|_2\}}\right)\frac{z}{\sigma} \tag{6.26}$$
$$= \frac{w_k + \sigma \nabla \bar{u}_k}{\max\{1, \|w_k + \sigma \nabla \bar{u}_k\|_2\}}. \tag{6.27}$$

In the update step of u, the proximal operator for the indicator function results in the projection $\mathcal{P}_{\mathcal{C}}$, which is defined by

$$\mathcal{P}_{\mathcal{C}}(u) = \underset{\hat{u} \in \mathcal{C}}{\text{argmin}} \|u - \hat{u}\|_{\mathcal{X}}, \tag{6.28}$$

as described in section 4.2.2. The step size restriction can be estimated with $\tau\sigma < 2/\|K^*K\| = 1/4$, where $\|K^*K\| = 8$ can be calculated in the experiments, or by an



eigenvalue analysis. With these instantiations the primal-dual algorithm 7 now more concretely reads as

Algorithm 12 primal-dual (PD) for motion segmentation

1. choose step sizes $\tau, \sigma > 0$, with $\tau\sigma < 1/4$, an extrapolation parameter $\theta \in [0,1]$, an initial point $(u_0, w_0) \in \mathcal{X} \times \mathcal{Y}$, and set $\bar{u}_0 = u_0$.

2. iterate for $k = 0, 1, \ldots$

$$w_{k+1} = \frac{w_k + \sigma \nabla \bar{u}_k}{\max(1, \|w_k + \sigma \nabla \bar{u}_k\|_2)} \tag{6.29}$$

$$u_{k+1} = \mathcal{P}_{\mathcal{C}}(u_k + \tau \mathrm{div}(w_{k+1}) - \tau \mu s_v) \tag{6.30}$$

$$\bar{u}_{k+1} = u_{k+1} + \theta(u_{k+1} - u_k), \tag{6.31}$$

Since the linearized ADMM, which uses the proximal gradient method in the u-update, is equivalent to the primal-dual algorithm, if the updates for u and d are exchanged, cf. section 4.3.3, we do not give this algorithm here explicitly for the motion segmentation model.

Remark 6.1. For the motion segmentation application, changing the update ordering in the ADMM slows down convergence a little bit. Thus, in the experiments, we stick to the update ordering, where u is updated first and afterwards d and b.

The ADMM with Gauss–Seidel for functional J_v. In the ADMM, the new variable $d := \nabla u$ is introduced together with the equality constraint $\nabla u - d = 0$. The quadratic augmented Lagrangian (scaled form) for the motion segmentation problem reads

$$\mathcal{L}(u, d, b) = \int_\Omega \Big(\|d\|_2 + \mu s_v u + \frac{\sigma}{2} \|\nabla u - d + b\|_2^2 \Big), \tag{6.32}$$

where b is the Lagrangian multiplier. The associated optimization problem is given by

$$\max_b \min_{u \in [0,1]^{|\Omega|},\, d} \mathcal{L}(u, d, b). \tag{6.33}$$

The ADMM for the motion segmentation model is given by

$$u_{k+1} = \operatorname*{arg\,min}_{u \in [0,1]^{|\Omega|}} \int_\Omega \Big(\mu s_v u + \frac{\sigma}{2} \|\nabla u + b_k - d_k\|_2^2 \Big) \tag{6.34}$$

$$d_{k+1} = \mathrm{prox}_{\frac{1}{\sigma} F}(\nabla u_{k+1} + b_k) \tag{6.35}$$

$$b_{k+1} = b_k + \nabla u_{k+1} - d_{k+1}, \tag{6.36}$$

where $F(d) = \int_\Omega \|d\|_2$. The proximal operator of functional F can be computed through the shrinkage formula given in (6.25), which yields a well-defined update for d, even if $\|\nabla u_{k+1} + b_k\|_2 = 0$:

$$d_{k+1} = \left(1 - \frac{1}{\max(1, \sigma\|\nabla u_{k+1} + b_k\|_2)}\right)(\nabla u_{k+1} + b_k). \tag{6.37}$$

The gradient ascent step in the dual variable b can also be computed easily. As already seen in chapter 4, the update with respect to u is more complicated. A traditional variant of the ADMM uses a Gauss–Seidel method for the update of u, as done in [Goldstein et al., 2010] for the segmentation problem.
The optimal u in the unconstrained case must satisfy the Euler-Lagrange equation

$$\Delta u_{k+1} = \frac{\mu}{\sigma} s_v + \operatorname{div}(d_k - b_k), \tag{6.38}$$

where $\Delta = -K^*K$ is the Laplacian matrix. An approximate solution of this system can be obtained by one step of the Gauss–Seidel iteration for the unconstrained problem. To enforce the constraint $u \in [0,1]^{|\Omega|}$, the iterates u_k are projected onto $\mathcal{C} = [0,1]^{|\Omega|}$ through an orthogonal projection after the Gauss–Seidel iteration. In matrix form, the scheme then reads

Algorithm 13 ADMM with Gauss–Seidel for motion segmentation

1. choose penalty parameter $\sigma > 0$ and initial points $u_0 \in \mathcal{X}$, $d_0, b_0 \in \mathcal{Y}$, where $b_0 = 0$.

2. iterate for $k = 0, 1, \ldots$

$$u_{k+1} = \mathcal{P}_{\mathcal{C}}\left[{L_\llcorner}^{-1}\left(\sigma^{-1}\mu s_v + \operatorname{div}(d_k - b_k) - L^\urcorner u_k\right)\right] \tag{6.39}$$

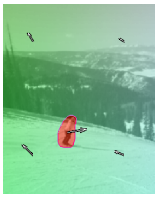
$$d_{k+1} = \left(1 - \frac{1}{\max(1, \sigma\|\nabla u_{k+1} + b_k\|_2)}\right)(\nabla u_{k+1} + b_k) \tag{6.40}$$

$$b_{k+1} = b_k + \nabla u_{k+1} - d_{k+1}, \tag{6.41}$$

   with $L_\llcorner$ and $L^\urcorner$ are the lower and upper triangular matrices of the Laplacian matrix Δ, such that $\Delta = L_\llcorner + L^\urcorner$ (the diagonal is included in $L_\llcorner$).

Sometimes more than one Gauss–Seidel sweep yields faster convergence. Then, some iterations of the Gauss–Seidel algorithm are calculated before the projection onto $[0,1]^{|\Omega|}$. Afterwards, b and d are updated. We study this in the experiments.

The implicit ADMM for the functional J_v. We now apply our ideas from chapter 5 to the motion segmentation problem. As for the ADMM with Gauss–Seidel iterations,



the augmented Lagrangian is deduced from the substitution of the variable $d = \nabla u$ (splitting). As done for the primal-dual algorithm, we model the constraint $u \in [0,1]^{|\Omega|}$ through the indicator function $\delta_{\mathcal{C}}(u)$, with $\mathcal{C} = [0,1]^{|\Omega|}$. The augmented Lagrangian then reads

$$\mathcal{L}(u,d,b) = \int_\Omega \Big(\|d\|_2 + \mu s_v u \Big) + \delta_{\mathcal{C}}(u) + \int_\Omega \Big(\frac{\sigma}{2} \|\nabla u - d + b\|_2^2 \Big), \tag{6.42}$$

where b is the Lagrangian multiplier. The associated optimization problem is given by

$$\max_b \min_{u,\, d} \mathcal{L}(u,d,b). \tag{6.43}$$

The ADMM is then given by

$$u_{k+1} = \arg\min_u \int_\Omega \Big(\mu s_v u \Big) + \delta_{\mathcal{C}}(u) + \int_\Omega \Big(\frac{\sigma}{2} \|\nabla u + b_k - d_k\|_2^2 \Big) \tag{6.44}$$

$$d_{k+1} = \nabla u_{k+1} + b_k - \frac{1}{\sigma} \text{prox}_{\sigma F^*} \left(\sigma(\nabla u_{k+1} + b_k) \right) \tag{6.45}$$

$$b_{k+1} = b_k + \nabla u_{k+1} - d_{k+1}. \tag{6.46}$$

As for the ADMM with Gauss-Seidel update, the update for d is performed as given in equation (6.37). The implicit update step of u, given in equation (5.104), can be written in terms of the scaled proximal operator for $\delta_{\mathcal{C}}(u)$, or in terms of the proximal operator for function $\hat{G}(u) = \delta_{\mathcal{C}}(u) + \frac{\sigma}{2} \int_\Omega \|\nabla u\|_2^2 \, \mathrm{d}x$. The update is given as follows:

$$u^{k+1} = \text{prox}_{\tau \hat{G}} \{ \overbrace{u_k - \tau[\mu s_v - \sigma \, \text{div}(b_k - d_k)]}^{=:p} \} \tag{6.47}$$

$$= \arg\min_u \left(\delta_{\mathcal{C}}(u) + \frac{\lambda}{2} \|\nabla u\|_{\mathcal{Y}}^2 + \frac{\|u-p\|_{\mathcal{X}}^2}{2\tau} \right) \tag{6.48}$$

$$= \arg\min_{u \in \mathcal{C}} \left(\frac{\lambda}{2} \|\nabla u\|_{\mathcal{Y}}^2 + \frac{\|u-p\|_{\mathcal{X}}^2}{2\tau} \right) \tag{6.49}$$

$$= \arg\min_{u \in \mathcal{C}} \frac{1}{2\tau} \left\| u - (I - \tau\sigma\Delta)^{-1} p \right\|_{(I-\tau\sigma\Delta)}^2 \tag{6.50}$$

$$= \mathcal{P}_{\mathcal{C}}^{(I-\tau\sigma\Delta)} \left[(I - \tau\sigma\Delta)^{-1} p \right] \tag{6.51}$$

$$= \mathcal{P}_{\mathcal{C}}^{(I-\tau\sigma\Delta)} \left[(I - \tau\sigma\Delta)^{-1} \Big(u_k + \tau\lambda \, \text{div}(b_k - d_k) - \tau\mu s_v \Big) \right] \tag{6.52}$$

$$= \text{prox}_{\tau\delta_{\mathcal{C}}}^{(I-\tau\sigma\Delta)} \Big((I - \tau\sigma\Delta)^{-1} \Big(u_k + \tau\lambda \, \text{div}(b_k - d_k) - \tau\mu s_v \Big) \Big), \tag{6.53}$$

where Δ is the discrete approximation of the Laplacian operator derived from $-\Delta = -\,\text{div} \cdot \nabla = K^*K$. The implicit ADMM yields a projection using the norm induced by the matrix $I - \tau\sigma\Delta$ onto set $\mathcal{C}$. This projection is much more complicated to compute than the orthogonal one using the norm induced by the identity matrix. Thus, we use

the approximation described in chapter 5.3 yielding the orthogonal projection for the motion segmentation example. The algorithm 10 then reads

Algorithm 14 inexact implicit ADMM for motion segmentation

1. choose penalty parameter $\sigma > 0$, step size $\tau > 0$ and initial points $u_0 \in \mathcal{X}$, $d_0, b_0 \in \mathcal{Y}$.

2. iterate for $k = 0, 1, \ldots$

$$u_{k+1} = \mathcal{P}_{\mathcal{C}}[(I - \tau\sigma\Delta)^{-1}(u_k + \tau\sigma \mathrm{div}(b_k - d_k) - \tau\mu s_v)] \tag{6.54}$$

$$d_{k+1} = \left(1 - \frac{1}{\max(1, \sigma\|\nabla u_{k+1} + b_k\|_2)}\right)(\nabla u_{k+1} + b_k) \tag{6.55}$$

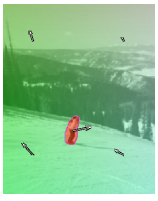
$$b_{k+1} = b_k + \nabla u_{k+1} - d_{k+1}. \tag{6.56}$$

The update of u requires the solution of a linear system, which can be demanding if one aims at real-time application. Here, if we assume periodic boundary conditions, the operator $I - \tau\sigma\Delta$ has a circular structure and the linear system can be solved efficiently via the Fast Fourier Transform (FFT), which is described in detail in the next section.
In section 5.3, we predicted, that the error between the exact minimizer and the approximate solution will increase with the step size τ. We will analyze these differences numerically in the next section and compare the results to the theory given in 5.3. Although the accuracy of the segmentation decreases with larger step sizes $\tau\sigma$, we still obtain reasonable segmentations also with large step sizes.

Remark 6.2. One might also try to evaluate the projection with respect to the matrix $I - \tau\sigma\Delta$ explicitly. This yields very expensive and difficult calculations, or again an iterative algorithm. During experiments, it turned out, that the additional cost for calculating the exact projection was too high for a real-time application and did not yield better results. Thus, in this work, we do not follow this trail any further.

The proximal Newton-type method for functional J_v. As suggested in section 4.3, we use the proximal Newton-type method for the update step with respect to u in the ADMM, and include the constraint $[0,1]^{|\Omega|}$ through the indicator function $\delta_{[0,1]^{|\Omega|}}$. Then, the functional F and the smooth functional G are given by

$$u_{k+1} = \arg\min_u \underbrace{\delta_{[0,1]^{|\Omega|}}(u)}_{=:F(u)} + \underbrace{\int_\Omega \mu s_v u + \frac{\sigma}{2}\|\nabla u + b_k - d_k\|_2^2}_{=:G(u)}. \tag{6.57}$$



The updates for b and d are handled as in the ADMM with Gauss–Seidel update. The gradient and Hessian of G are given by

$$\nabla_u G(u_k) = \mu s_v - \sigma \Delta u_k + \sigma \mathrm{div}(d_k - b_k)) \tag{6.58}$$

$$H = -\sigma\Delta. \tag{6.59}$$

The proximal Newton-type update step reads:

$$u_{k+1} = \mathcal{P}_{\mathcal{C}}^{-\sigma\Delta}\left[\Delta^{-1}\left(\frac{\mu}{\sigma}s_v + \mathrm{div}(d_k - b_k)\right)\right] \tag{6.60}$$

If the projection $\mathcal{P}_{\mathcal{C}}^{-\sigma\Delta}$, which is difficult to solve, is approximated as for the implicit ADMM by $\mathcal{P}_{\mathcal{C}}$, and the inner linear system is solved by a Gauss–Seidel iteration, the proximal Newton-type update step is equivalent to the ADMM with Gauss–Seidel update.

Remark 6.3. Periodic boundary conditions should not be applied in this case, since the linear system will become singular.

6.1.3 Fast Solution of the Linear System

A fast solution of the linear system, arising from the implicit update step in equation (6.54) is important for the efficiency of the implicit ADMM. Fortunately, if we choose periodic boundary conditions, the operator $(I - \tau\sigma\Delta)$ is a circulant matrix and can be diagonalized by a fast Fourier transform, and solved efficiently.
For a circulant matrix the multiplication with a vector is equivalent to a convolution with the first column of the Matrix

$$Ax = \alpha \star x, \quad \alpha = A(1:m\,,\,1), \tag{6.61}$$

with the circulant matrix

$$A = \begin{pmatrix} a & b & & \dots & & c \\ c & a & b & & \dots & \\ & c & a & b & & \dots \\ & & & & & \\ & & & & \ddots & \\ b & & \dots & & c & a \end{pmatrix}$$

In the Fourier space a convolution is a multiplication of the Fourier transforms

$$\mathcal{F}\{\alpha \star x\} = \mathcal{F}\{\alpha\}\mathcal{F}\{x\}. \tag{6.62}$$

In this part, u is assumed to have periodic boundary conditions in every dimension. The discrete Laplacian is given by

$$\Delta u = L_x u + u L_y, \tag{6.63}$$

with $u \in R^{m\times n}$ and the matrices $L_x \in \mathbb{R}^{m\times m}$ and $L_y \in \mathbb{R}^{n\times n}$, which both have the same circular structure:

$$L_x = \begin{pmatrix} -2 & 1 & 0 & \dots & 0 & 1 \\ 1 & -2 & 1 & 0 & \dots & 0 \\ & & & \ddots & & \\ 1 & 0 & \dots & 0 & 1 & -2 \end{pmatrix} \in \mathbb{R}^{m\times m}. \tag{6.64}$$

Consider the problem, equivalent to the update step in u in equation (6.54) in algorithm 14 without the projection,

$$(I - \tau\sigma\Delta)u_{k+1} = B, \tag{6.65}$$

where $B \in \mathbb{R}^{m\times n}$ and $u_{k+1} \in \mathbb{R}^{m\times n}$. Let $l_x = L_x(1:m\ ,\ 1)$ be the first column of L_x, and $l_y = L_y(1\ ,\ 1:n)$ is the first row of L_y. A two-dimensional Fourier transform for the equation reads

$$\begin{aligned} \mathcal{F}_x\{\mathcal{F}_y\{B\}\} &= \mathcal{F}_x\{\mathcal{F}_y\{u_{k+1} - \tau(L_x u_{k+1} + u_{k+1}L_y)\}\} && (6.66)\\ &= \mathcal{F}_x\{\mathcal{F}_y\{u_{k+1}\}\} - \tau\sigma\mathcal{F}_x\{\mathcal{F}_y\{(l_x \star u_{k+1} + u_{k+1} \star l_y)\}\} && (6.67)\\ &= \mathcal{F}_x\{\mathcal{F}_y\{u_{k+1}\}\}[1 - \tau\sigma(\mathcal{F}_x\{l_x\} + \mathcal{F}_y\{l_y\})], && (6.68) \end{aligned}$$

where $l_x \star u_{k+1}$ is the convolution of l_x with each column of u_{k+1}, and $u_{k+1} \star l_y$ is the convolution of each row of u_{k+1} with l_y. The solution $u_{k+1} \in \mathbb{R}^{m\times n}$ for the linear system in (6.54) is given by the componentwise division

$$u_{k+1} = \mathcal{F}_x^{-1}\left\{\mathcal{F}_y^{-1}\left\{\frac{\mathcal{F}_x\{\mathcal{F}_y\{B\}\}}{1 - \tau\sigma(\mathcal{F}_x\{l_x\} + \mathcal{F}_y\{l_y\})}\right\}\right\}, \tag{6.69}$$

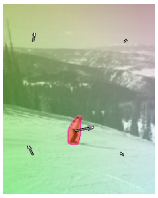
with $B = u_k + \tau\sigma\mathrm{div}(b_k - d_k) - \tau\mu s_v$.

Remark 6.4. The linear system can be solved in a similar way with a discrete cosine transform in case Neumann boundary conditions are applied.

6.1.4 Implementational Details

Discretization. The images are given as a pixel grid, thus this grid is used for discretization. The gradient and the divergence operator are approximated with finite differences on the pixel grid. The mesh size, i.e. the distance between two pixels, is chosen to be one. As already mentioned in the previous section, the operator $K = \nabla$ and its adjoint $K^* = -\,\mathrm{div}$. Thus, it would be convenient, if the discrete approximations fulfill

$$\langle \nabla u, w\rangle_{\mathbb{R}^{2N}} = -\langle u, \mathrm{div}\, w\rangle_{\mathbb{R}^N}, \tag{6.70}$$



which holds if the discrete gradient and divergence are chosen as follows. The gradient of the labeling function u is calculated through forward differences in the spatial directions x and y with homogeneous Dirichlet boundary conditions:

$$\nabla_x u(x,y) = \begin{cases} u(x+1,y) - u(x,y) & \text{if } x = 1,\dots,m-1 \\ 0 - u(x,y) & \text{if } x = m \end{cases}, \quad y = 1,\dots,n \tag{6.71}$$

$$\nabla_y u(x,y) = \begin{cases} u(x,y+1) - u(x,y) & \text{if } y = 1,\dots,n-1 \\ 0 - u(x,y) & \text{if } y = n \end{cases}, \quad x = 1,\dots,m \tag{6.72}$$

The divergence operator is calculated through backward differences:

$$\operatorname{div}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \nabla_x^* u_1 + \nabla_y^* u_2, \tag{6.73}$$

with

$$\nabla_x^* u_1(x,y) = \begin{cases} u_1(x,y) - u_1(x-1,y) & \text{if } x = 2,\dots,m \\ u_1(x,y) & \text{if } x = 1 \end{cases}, \quad y = 1,\dots,n \tag{6.74}$$

$$\nabla_y^* u_2(x,y) = \begin{cases} u_2(x,y) - u_2(x,y-1) & \text{if } y = 2,\dots,n \\ u_2(x,y) & \text{if } y = 1 \end{cases}, \quad x = 1,\dots,m \tag{6.75}$$

The discrete approximation of the Laplacian matrix is given by $K^*K = -\operatorname{div}\cdot\nabla = -\Delta$.

Remark 6.5. The choice of the discretization is crucial for the validity of equation (6.70). A proof of (6.70) for the given discretization can be found in [Bredies and Lorenz, 2011].

Orthogonal Projection. The orthogonal projection $\mathcal{P}_{[0,1]^{|\Omega|}}$ onto the set $[0,1]^{|\Omega|}$, with $|\Omega| = N$ is realized component wise

$$\mathcal{P}_{[0,1]^N}(u(x_i)) = \max\{\min\{u(x_i),\, 1\},\, 0\}, \quad i = 1,\dots,N. \tag{6.76}$$

Initialization. There are different variants of initializing the algorithms. Since the objective functional is convex in u, the algorithm can be initialized with a randomly chosen u and will converge to a global minimizer. However, the theory does not cover the outer iteration, which alternatingly minimizes the motion vectors and the segmentation. An initialization closer to the segmentation result might yield faster convergence for the motion vectors. One possibility is to initialize the algorithms with a smoothed version of the difference of two consecutive images $u_0 = \mathcal{K}_\rho \star |f(x,t+1) - f(x,t)|$, where $\mathcal{K}_\rho$ is a Gaussian kernel with standard deviation ρ and $\star$ denotes the convolution. This initialization for the segmentation u will at least cover the edges of the moving objects. This may lead to a better approximation of the motion vectors in the first loop and thus, may lead to a faster convergence.

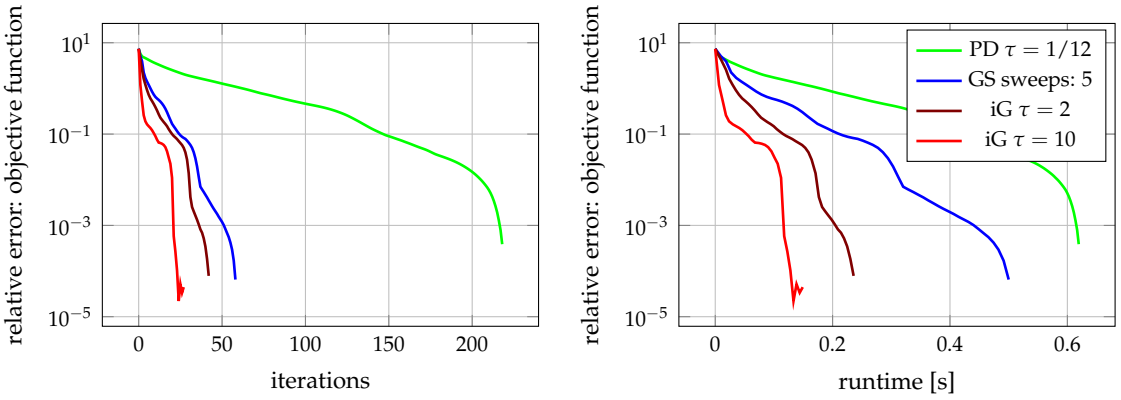


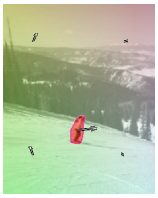
Figure 6.1: Comparison of convergence of different algorithms with fixed precalculated motion vectors v on the "cars6" sequence. The relative error of the objective function, $\|J - J_*\| \,/\, \|J_*\|$, is plotted against the iterations (left) and runtime in seconds (right).

For the ADMM, the second primal variable is always initialized with the constraint $d = \nabla u$. The dual variable is initialized by $b = 0$, but other initializations are also possible, e.g. $b = 1$. For the primal-dual algorithm, the dual variable w is initialized with $w = \nabla u$ and $\bar{u} = u$.

Stopping Criterion. As a stopping criterion for all algorithms throughout the numerical experiments, the pixelwise error of the iterates $\|u_k - u_{k-1}\|_{\mathcal{X}} \,/ N < \text{tol}$ is tested, where N is the total number of pixels in the image domain Ω and tol is a predefined tolerance.

6.2 Performance Analysis

In this section, the performance of the primal-dual algorithm, the ADMM with a Gauss–Seidel update and the implicit ADMM are measured by the value of the objective functional the algorithm achieved at the end. We assume, that similar functional values belong to similar good segmentation results.
The performance of the algorithms is first analyzed with fixed motion vectors, since the convergence results do not cover the alternating minimization with respect to u and v. We expect, that the convergence with fixed motion vectors will be as predicted in theory.
All convergence results in this section are compared on the "cars6" sequence from the Berkeley segmentation dataset from [Brox and Malik, 2010]. In the segmentation re-



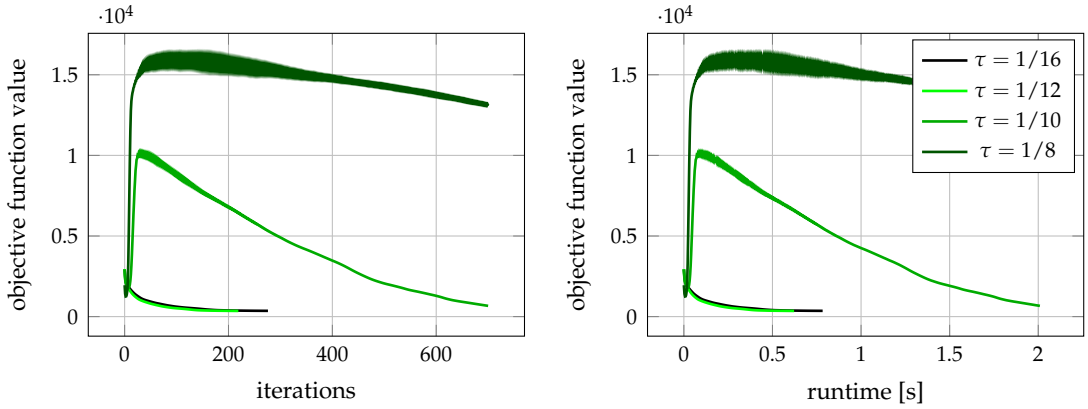


Figure 6.2: Comparison of convergence of primal-dual algorithms with fixed precalculated motion vectors v for different step sizes τ on the "cars6" sequence. The objective function value is plotted against the iterations (left) and runtime in seconds (right).

sults, further sequences from this dataset, and the self-made sequences "bird", "plane" and "skiing" are used.
In order to compare the different algorithms, the weighting parameters are set to $\mu = 2$ and $\sigma = 2$ in the following experiments for all algorithms. Further, all algorithms are implemented in Matlab and run on a 2.2 GHz Intel Core i7.

6.2.1 Convergence of Segmentation for Fixed Motion Vectors

For the convergence results with fixed motion vectors, the motion vectors from previously calculated motion segmentation results are used. The motion vectors are fixed and the algorithms only calculate the segmentations u. The convergence results for the primal-dual algorithm (PD) with step size $\tau = 1/12$, $\theta = 1$, the ADMM with 5 Gauss–Seidel sweeps (GS) and the implicit ADMM (iG) with $\tau \in \{2, 10\}$ are shown in Figure 6.1, where the relative errors of the objective functional, $\|J - J_*\| / \|J_*\|$, is plotted against the iterations (left) and the runtime (right). For all algorithms $\sigma = 2$ is fixed, otherwise the functional values would vary for different σ. Due to its small step size τ, the primal-dual algorithm needs a lot of steps to converge. Although each iteration can be computed efficiently and fast, the overall runtime is longer compared to the other methods. The ADMM with Gauss–Seidel update needs only a few iterations, but each iteration needs more computation time. The implicit ADMM takes around 40 iterations with $\tau = 2$ and only half of that for $\tau = 10$, while each iteration can be computed sufficiently fast. Further, for all methods a linear convergence can be observed in Figure 6.1.

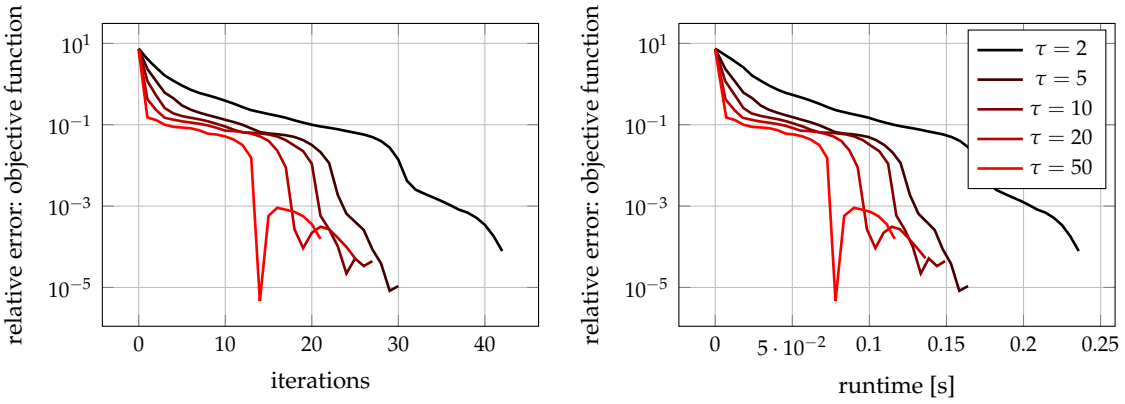
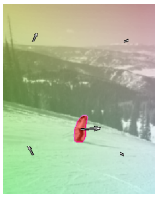


Figure 6.3: Comparison of convergence of stable ADMM with fixed precalculated motion vectors v for different step sizes τ on the "cars6" sequence. The relative error of the objective function is plotted against the iterations (left) and runtime in seconds (right).

The algorithms are also tested individually with different step sizes. The primal-dual algorithm is analyzed for different step sizes $\tau \in \{1/16, 1/12, 1/10, 1/8\}$, for $\sigma = 2$ and $\theta = 1$. In Figure 6.2, the objective function value is plotted against the number of iterations and runtime. We can observe, that for $\tau = 1/10$, the algorithm converges, but only after many iterations, and for $\tau = 1/8$ an oscillatory, non-convergent behavior can be observed. Therefore, in this example, the effect of stiffness, described in section 5.1.3, can be observed nicely. In order to prevent the iterates from oscillating, the step size has to be reduced, but then the algorithm takes many iterations to converge. Even though each update in the primal-dual algorithm can be computed very fast, the overall runtime cannot be reduced.
The convergence of the inexact implicit ADMM, given in algorithm 14, is analyzed for different step sizes $\tau \in \{2, 5, 10, 20, 50\}$. The algorithm has no step size restriction, however, as theoretically discussed in section 5.3, the error of the approximation increases with the step size τ. In Figure 6.3, the relative error of the objective function, $\|J - J_*\| / \|J_*\|$, is again plotted against iterations and runtime. We can observe, that for larger step sizes, the algorithm takes less iterations and, since the runtime only depends on the number of iterations, the runtime decreases with increasing τ. However, analyzing the minimum values of the objective functional achieved by the algorithm, it can be observed, that the minima also grow with τ, but only barely. We will analyze this error yielded by larger step sizes in more detail in the next sections. It can also be observed in Figure 6.3 that near the optimal solution for larger step sizes τ, the algorithm does not converge directly, but jumps back a few times.
As mentioned in section 4.3, different numbers of Gauss–Seidel sweeps on the segmen-



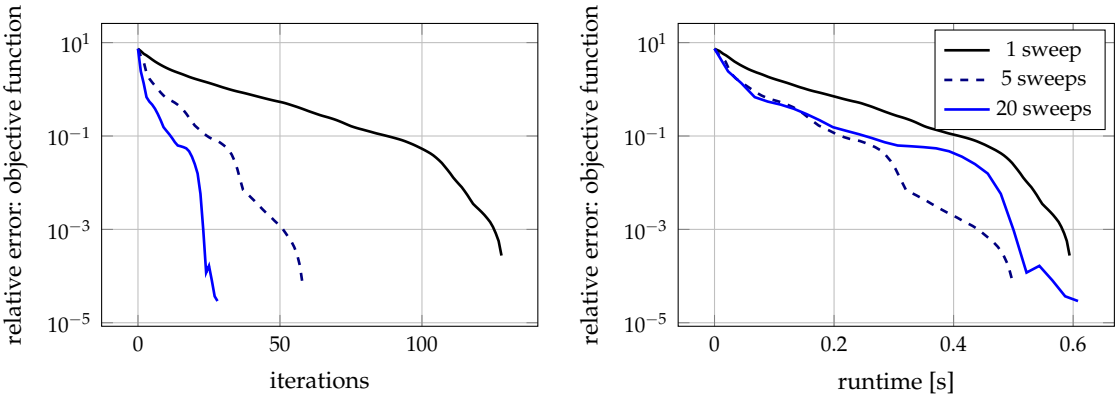


Figure 6.4: Comparison of convergence of ADMM with Gauss–Seidel update with fixed precalculated motion vectors v with different numbers of Gauss–Seidel sweeps in each iteration on the "cars6" sequence. The relative error of the objective function is plotted against the iterations (left) and runtime in seconds (right).

tation u before updating d and b, can lead to faster convergence. This can be seen in Figure 6.4. The ADMM with Gauss–Seidel update was tested with only one sweep of the Gauss–Seidel step, with 5 sweeps, and with 20 sweeps. The number of iterations decreases by more than a factor of two from 1 to 5 sweeps and also from 5 sweeps to 20 sweeps. However, while the runtime with 5 sweeps decreases significantly in view of the update with one Gauss–Seidel sweep, the update with 20 sweeps takes longer per iteration and increases the overall runtime. In this example, the update with 5 Gauss–Seidel sweeps was the fastest.

6.2.2 Convergence of Segmentation and Motion Vectors

For the convergence of the segmentation u with fixed motion vectors v the algorithms converge, as shown in the theoretical results. The convergence of the alternating minimization scheme between u and v is not covered by the given theory, but we observe that convergence seems to be given in this case. In Figure 6.5 the convergence results for the objective functional value of the primal-dual algorithm, the ADMM with five Gauss–Seidel sweeps and the implicit ADMM with step sizes $\tau \in \{2, 20\}$ are shown. In the lower plot, the convergence of the relative error of the vectors, $\|v_k - v_*\| \,/\, \|v_*\|$ is plotted, and in the upper plot the convergence of the objective functional value is plotted. Again, the algorithms are compared according to the number of iterations (left plots) and the runtime (right plots). We can observe, that the segmentation starts to converge only after the motion vectors have converged, i.e. the relative error of the mo-

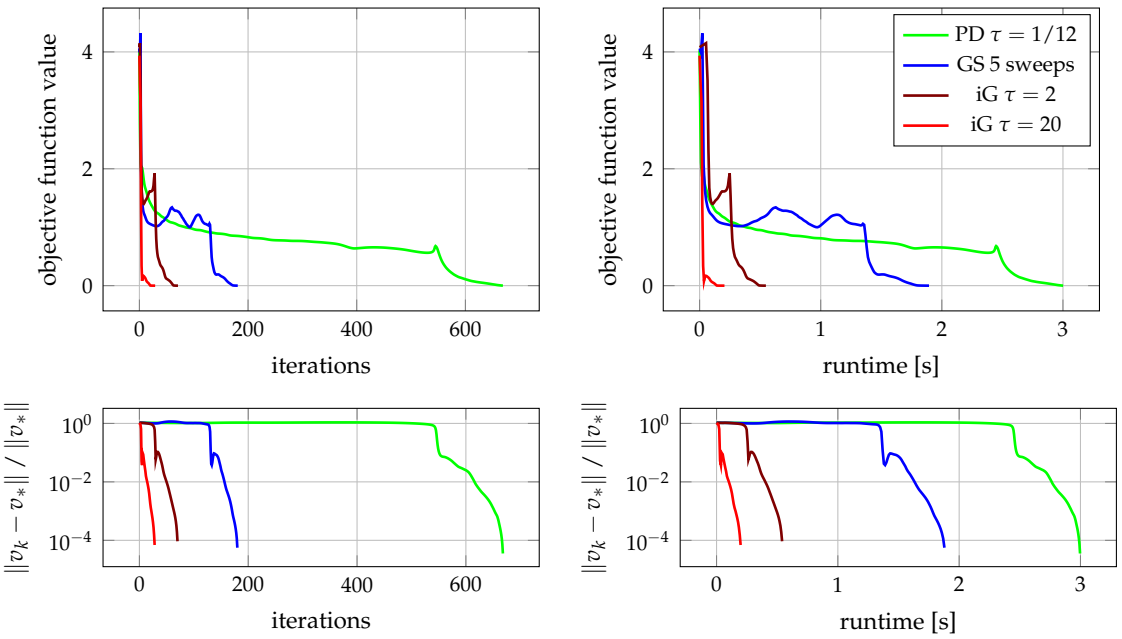
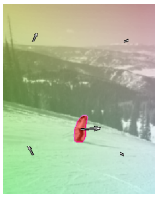


Figure 6.5: Comparison of convergence of different algorithms on the "cars6" sequence. Upper plots: relative error of the objective function value is plotted against the iterations (left) and runtime (right); Lower plots: relative error of the motion vectors, $\|v - v_*\| / \|v_*\|$ is plotted against the iterations (left) and runtime in seconds (right)

tion vector is near zero. This behavior can be observed for all three algorithms. For the primal-dual algorithm, the phase until the motion vectors converge takes longest, and the ADMM with Gauss–Seidel update takes second longest. For the implicit ADMM with $\tau = 2$, the vectors converge fast, while with $\tau = 20$ the vectors and thus the segmentation converges almost immediately. One can argue that this behavior correlates with the shape and the smoothness of the iterates of u.

Therefore, we analyze the Fourier transforms of the iterates. In Figure 6.6, some iterates and their frequencies of the implicit ADMM (left) and the primal-dual (right) are shown. The frequencies are only shown on a 20×20 frequency grid, where the constant frequency is in the center. Dark color represents a low value and yellow and white a high value in the frequency plot. The values of iterates are between 0 (blue) and 1 (red), the value 0.5 is shown in green. On the bottom of the figure, the convergence of the motion vectors is plotted against the iterates and the chosen iterates are highlighted with red dots. The first picture in both methods shows the initialization with the time derivative of the image sequence, $|f(x, t+1) - f(x, t)|$. Further iterates are chosen at similar states of the iteration with the two algorithms. For both methods,



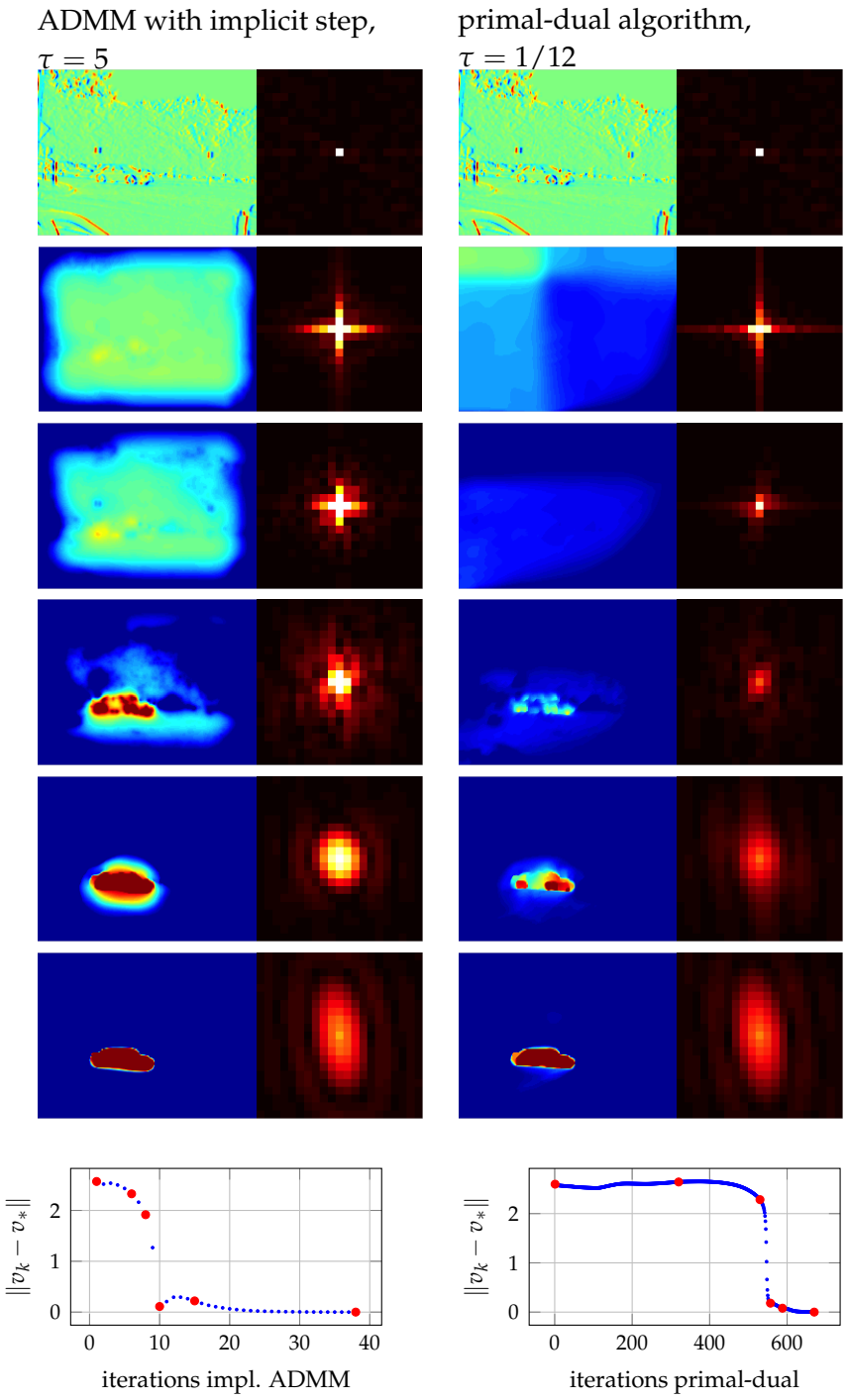
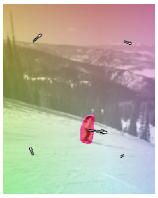


Figure 6.6: Six iterates and corresponding Fourier transforms of the ADMM with implicit proximal method and primal-dual algorithm on an image pair of the "cars6" sequence. Only a grid of the largest 21×21 frequencies is shown next to the iterates, with the constant frequency in the center. The convergence of the motion vectors is shown on the bottom, where the chosen iterates are highlighted. Left: implicit ADMM, right: primal-dual. Colors: iterates: blue = 0, green = 0.5 , red = 1; frequencies: black = 0, white = 2100.

the upper three pictures show iterates before the motion vector has converged and the lower three pictures show iterates after, or during the convergence of the motion vector. The frequency-values of the iterates of the implicit ADMM are large around the center, which means that the segmented area in the iterates is large and smooth. Analyzing the frequencies in the iterates of the primal-dual algorithm, we can observe that they also gather around the center, but the values are much smaller than the iterates of the implicit ADMM. Further, the frequency in the center, corresponding to the constant part of the iterates, is much larger for the implicit ADMM than for the primal-dual algorithm, thus the segmented area is larger. This can also be observed in the plot of the iterates itself. It seems that the iterates of the primal-dual algorithm converge to the solution from the inside, while the iterates of the implicit ADMM approach the solution from outside, i.e. the segmentation is growing during the iteration of the primal-dual algorithm and shrinking during the iteration of the implicit ADMM.

In chapter 2.3, methods for optical flow calculation were given, where local and global methods are combined, hence a smoothing with a Gaussian kernel was beneficial for optical flow calculations. According to the aperture problem the estimation of a vector is better for a large region. Since we have seen in the Fourier plots of the iterates, the iterates of the implicit ADMM cover a larger area than the iterates of the primal dual, this might be the reason, why the motion vectors converge faster while iterating with the implicit ADMM. In Figure 6.6, it can also be observed that in the end of the iteration, the solution is almost binary.

In Figure 6.7 the motion segmentation results are given for two-label motion segmentation with constant motion vectors in each region. In table 6.1 the corresponding values for iterations, runtime and functional values are given explicitly for the tested sequences using the primal-dual algorithm, the ADMM with Gauss–Seidel update with different numbers of sweeps and the implicit ADMM for step sizes $\tau \in \{2, 20\}$. All image sequences are scaled. The best result in each category is printed in bold font. The implicit ADMM with both step sizes is faster than the primal-dual, and the classical ADMM (Gauss–Seidel update) for all sequences. Only once, in particular for the "bird" sequence, the best objective functional value J is produced by the primal dual algorithm. It can also be seen, that while the runtime decreases for the implicit ADMM with $\tau = 20$, the functional value increases, i.e. we assume, that the segmentation results are less accurate than with $\tau = 2$, or by using the other algorithms. This can be verified in the motion segmentation results in Figure 6.7, where the segmentations are given. However, this criterion does not hold for all cases, e.g. for the "cars1" sequence, as a human, we would rate the segmentation produced by the implicit ADMM with $\tau = 20$ better than the one produced by the primal-dual algorithm, while the objective functional values are worse. Hence, as predicted in chapter 3, the values of the objective function produced by the implicit ADMM and a fixed step sizes τ, increase with the step size τ. A few examples of the corresponding segmentations can be seen in



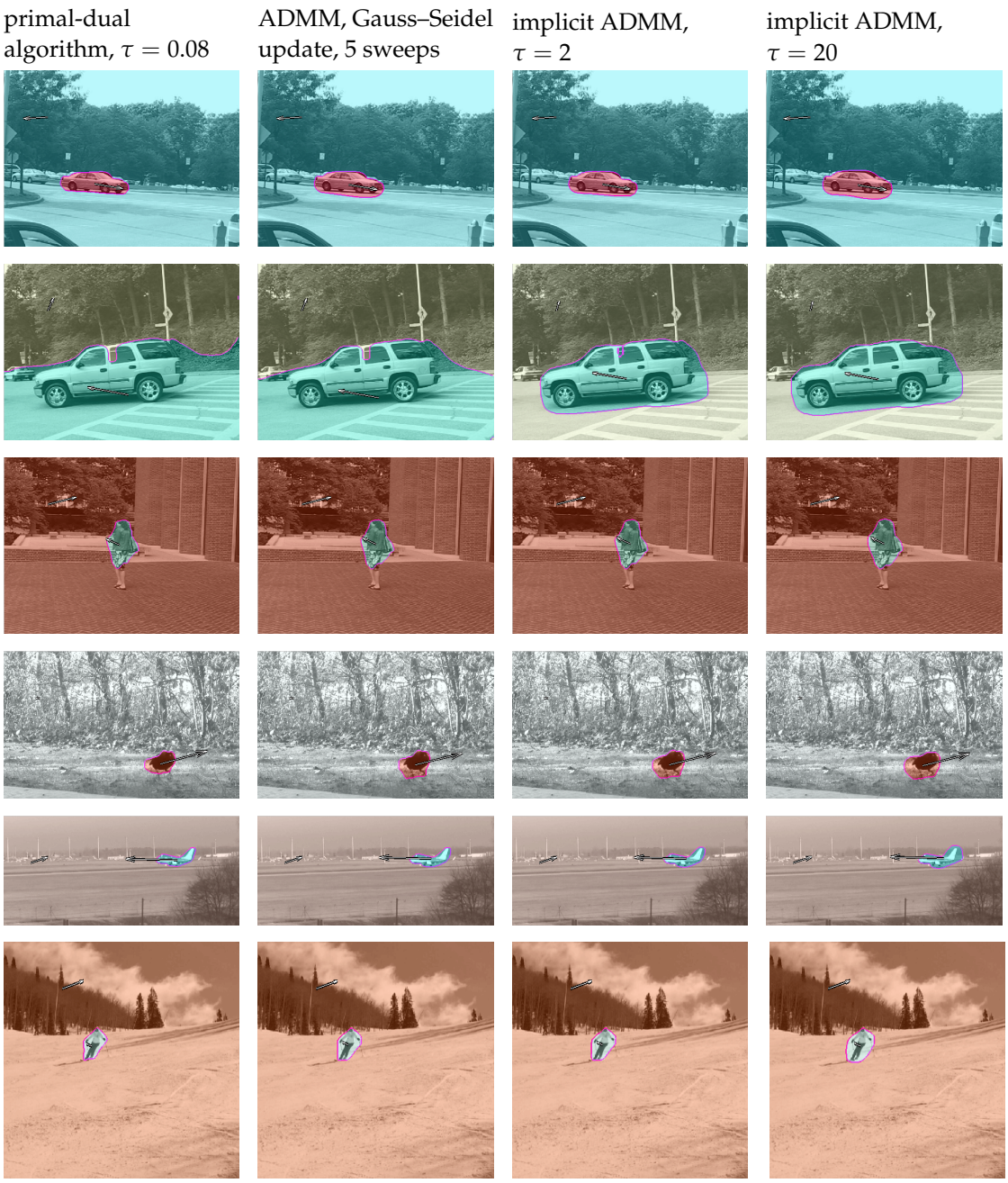


Figure 6.7: Motion segmentation for different examples. Top to bottom: "cars6", "cars1", "people1", "bird", "plane", "skiing1". Algorithms with different step sizes τ, parameters for all examples: $\sigma = 2, \mu = 2$. Motion vectors are color coded according to a color wheel: red indicates motion to the right, blue to the lower left corner, green to the upper left corner. Color intensity is proportional to the length of the motion vector.

sequence	algorithm	τ	iterations	time [s]	J
cars6, size: 480×640, scaled: 160×213	primal-dual	1/12	669	3.04	353.4
	ADMM (Gauss–Seidel 5)	-	181	1.80	351.3
	ADMM (Gauss–Seidel 20)	-	41	0.98	351.1
	implicit ADMM	2	71	0.51	**349.8**
	implicit ADMM	20	**29**	**0.22**	360.1
cars1, size: 480×640, scaled: 120×160	primal-dual	1/12	305	0.80	1281
	ADMM (Gauss–Seidel 5)	-	175	1.05	1275
	ADMM (Gauss–Seidel 20)	-	**52**	0.74	1295
	implicit ADMM	2	58	0.25	**1264**
	implicit ADMM	20	54	**0.23**	1341
people1, size: 480×640, scaled: 160×213	primal-dual	1/12	658	3.02	1077
	ADMM (Gauss–Seidel 5)	-	54	0.57	1070
	ADMM (Gauss–Seidel 20)	-	26	0.65	1073
	implicit ADMM	2	37	0.27	**1070**
	implicit ADMM	20	**19**	**0.15**	1076
bird, size: 500×801, scaled: 167×267	primal-dual	1/12	775	5.04	205.5
	ADMM (Gauss–Seidel 5)	-	131	2.11	**200.4**
	ADMM (Gauss–Seidel 20)	-	74	2.54	205.5
	implicit ADMM	2	69	0.77	202.7
	implicit ADMM	20	**17**	**0.22**	223.1
plane, size: 301×651, scaled: 151×326	primal-dual	1/12	999	7.10	542.2
	ADMM (Gauss–Seidel 5)	-	87	1.44	526.4
	ADMM (Gauss–Seidel 20)	-	45	1.70	528.2
	implicit ADMM	2	59	0.75	**525.4**
	implicit ADMM	20	**35**	**0.48**	536.7
skiing1, size: 601×601, scaled: 200×200	primal-dual	1/12	574	3.69	757
	ADMM (Gauss–Seidel 5)	-	58	0.77	750.2
	ADMM (Gauss–Seidel 20)	-	33	0.98	752.6
	implicit ADMM	2	35	0.30	**750**
	implicit ADMM	20	**19**	**0.17**	755.3

Table 6.1: Step size, number of iterations, runtime in seconds and final value of the objective functional J for the three algorithms under comparison and 6 different image sequences. The best values are written in bold font.

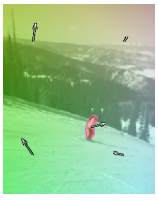


Figure 6.7 and in table 6.1, and further results are given in section 6.3.1.

6.2.3 Analysis of Inexact Fixed Points

As shown in chapter 5.3, the difference between the inexact solution $\tilde{u}$ and the exact solution u can be estimated by

$$\frac{\|u-\tilde{u}\|}{\|u\|} \leq \tau\sigma \, \frac{\lambda_{\max}^2}{\lambda_k} \, \frac{1+\tau\sigma\lambda_k}{1+\tau\sigma\lambda_{\max}} \left(1 + \frac{\left\|\frac{\mu}{\sigma}s_v - \operatorname{div}(b-d)\right\|}{\lambda_{\max}\|u\|}\right), \tag{6.77}$$

where $\lambda_{\max}$ is the maximal eigenvalue of the matrix K^*K which is the negative Laplacian matrix, i.e. $\lambda_{\max} = 8$, and $0 < \lambda_k \leq \lambda_{\max}$ is also an eigenvalue of K^*K, but not the smallest. This estimate only holds if the component of $(u-\tilde{u})$ in the direction of the eigenvectors associated with the smallest eigenvalues $\lambda_1, \ldots, \lambda_{k-1}$ is negligible, cf. equation (5.96).
First, we analyze, which values the left hand side can take. If u is a proper solution, which is not thresholded, i.e. pixels belonging to the foreground are 1, pixels on the background are 0, and only a few pixels on the border are in $(0,1)$, then, the worst case for $\tilde{u}$, i.e. an $\tilde{u} \in [0,1]^N$ maximizing $\|u-\tilde{u}\|$ is approximately $\tilde{u} = 1-u$, thus, roughly speaking, every pixel has a wrong value. This yields

$$\frac{\|u-\tilde{u}\|}{\|u\|} \approx \frac{\sqrt{N}}{\|u\|}. \tag{6.78}$$

However, since $(1-\tilde{u})$ models one area while $\tilde{u}$ models the second area, $\tilde{u} = (1-u)$ is also a valid solution and will therefore be flipped to $\tilde{u} = 1-\tilde{u}$ to match u. Thus, we assume that for u and $\tilde{u}$ it always holds that $\|u\| < \sqrt{N/2}$. Otherwise the segmentations are flipped to $u = 1-u$. If $\tilde{u}$ is the constant zero vector, i.e. no segmentation has been found, the left side of (6.77) will be equal to 1. The worst case error is attained if half of the pixels are wrong:

$$\|\tilde{u}-u\| \leq \sqrt{\frac{N}{2}}. \tag{6.79}$$

If $\|u\| = \alpha\sqrt{N}$, with $\alpha < 1$, describing the size of the correct area, we can write the worst case for the left side of (6.77) in terms of the size of the correct area in relation to $\sqrt{N}$:

$$\frac{\|u-\tilde{u}\|}{\|u\|} \leq \frac{1}{\alpha\sqrt{2}}. \tag{6.80}$$

For a specific motion segmentation problem ("cars6", scaled size 240×320), the eigenvalue λ_k, and the factor α can be approximated through experiments. Even though the smallest eigenvalue larger than zero of $-\Delta$ is $\lambda_2 = \mathcal{O}(10^{-4})$, it turns out that through an a posteriori calculation, the factor $1/(1+\tau\lambda_k)$ in equation (5.81) can be estimated with $\left\|(I+\tau\Delta)^{-1}(u-\tilde{u})\right\| / \|u-\tilde{u}\| \approx 0.67$, where for $\tau = 1$, we obtain $\lambda_k \approx 0.5$. The

factor of the area size is approximately $\alpha \approx 0.15$. However, even for a relatively small $\tau = 1$, the right hand side of estimation (6.77) will be larger than 10, while the left hand side is smaller than 5:

$$\underbrace{\frac{\|u - \tilde{u}\|}{\|u\|}}_{\leq 5 \ \forall u,\tilde{u}} \leq \underbrace{\tau \frac{\lambda_{\max}^2}{\lambda_k} \frac{1 + \tau\lambda_k}{1 + \tau\lambda_{\max}}}_{>10 \text{ for } \tau=1} \underbrace{\left(1 + \frac{\left\|\frac{\mu}{\sigma} s_v - \operatorname{div}(b - d)\right\|}{\lambda_{\max} \|u\|}\right)}_{>1}.$$

Thus, (6.77) holds for the motion segmentation example, but is not helpful in estimating errors of the inexact solution, especially for large step sizes $\tau > 10$, since we have already seen on a few examples in the previous chapter, that the results with large step sizes are not bad at all. Therefore, a better a priori estimation of $\|u - \tilde{u}\|$ in equation (5.67) is desirable, since the approximation due to the nonexpansiveness of the proximal operator in this inequality is already too rough. Further experimental results on the quality of the motion segmentations for larger step sizes τ can be found in section 6.3.1.

6.3 Parameter Study

In this section, the results of the implicit ADMM are measured on the segmentation results compared to a "pseudo" ground truth. Since most real-world sequences do not have a ground truth, i.e. manually segmented images, a "pseudo" ground truth is generated by a long iteration of an algorithm with a small step size. The resulting pseudo ground truth segmentations are verified visually and corrected by hand, if necessary.

Accuracy. As measurement of the segmentation result, we choose an accuracy measure. After the thresholding of the labeling function u with threshold $\vartheta = 0.5$, the segmentation at each pixel is either 0 or 1, i.e. $u \in \{0,1\}^N$. The thresholded segmentation is compared pixel-wise to the pseudo ground truth. A pixel which is 0 in the segmentation and 1 in the ground truth (gt) is marked "false" and vice versa, otherwise it is marked "correct". Thus the pixel-wise difference $|u - gt|$ is in each pixel either 0 for correct or 1 for false. To measure the accuracy, the number of correct pixels is compared to the number of false pixels. A pixel is correct, if it is not false, thus

$$\#correct = N - \#false \tag{6.81}$$

where N is the total number of pixels. The accuracy is given by

$$acc = \frac{\#correct}{\#correct + \#false} = \frac{N - \#false}{N} \tag{6.82}$$

Further, we want to mark the solution $u = 1 - gt$ also as correct, thus both solutions, u and $1 - u$, are measured and the minimum is taken.

$$acc = \frac{N - \min\{\#false(u, gt),\ \#false(1 - u, gt)\}}{N} \in [\tfrac{1}{2}, 1] \tag{6.83}$$

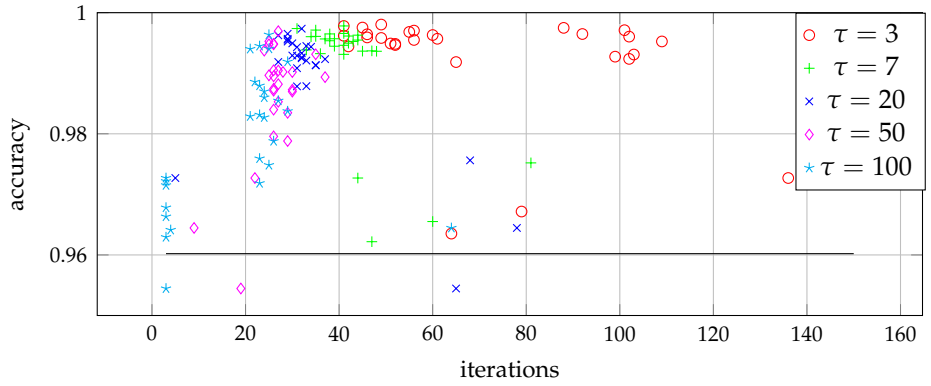


Figure 6.8: For every image pair of the "cars6" sequence the accuracy is plotted against the number of iterations for different step sizes $\tau \in \{3, 7, 20, 50, 100\}$ for the implicit ADMM.

Therefore, the worst accuracy will be $1/2$ instead of zero, since $(1 - u)$, which would yield the largest error, is also a valid solution and therefore has accuracy 1.

Weighting Parameters. The weighting parameter μ and the penalization parameter σ yield good results if they are chosen in $[1, 5]$. If the objects in the sequences have low contrast (e.g. a dark car on dark background), it might be beneficial to increase the data term parameter μ, but in general for the two-label model a good choice is $\mu = 2$ and $\sigma = 2$. For the multi-label models, this choice is more critical for good results, as we will see in section 6.4.

6.3.1 Step Size

Since we have already seen, that the value of the objective functional will increase with the step size τ, we now analyze the influence of the step size on the accuracy of the segmentation result. Since the algorithm has no direct step size restriction, we use the step sizes $\tau \in \{3, 7, 20, 50, 100\}$ on different image sequences and compare the results. In Figure 6.8 the results for the different step sizes are plotted for the "cars6" sequence. The accuracy for one image pair with step size τ is plotted against the number of iterations, i.e. for every consecutive image pair of the "cars6" sequence, a red circle is plotted for the calculation with step size $\tau = 3$. The number of iterations is chosen over the runtime here, since the computation of one iterate takes roughly the same amount of time for every τ, but the runtime can be disturbed by background activities during the computation of a long test series, while the number of iterations cannot be affected at all. The black horizontal line shows the accuracy for $u = 0$, i.e. when no objects are found. This accuracy depends on the size of the objects in the sequence. The closer a

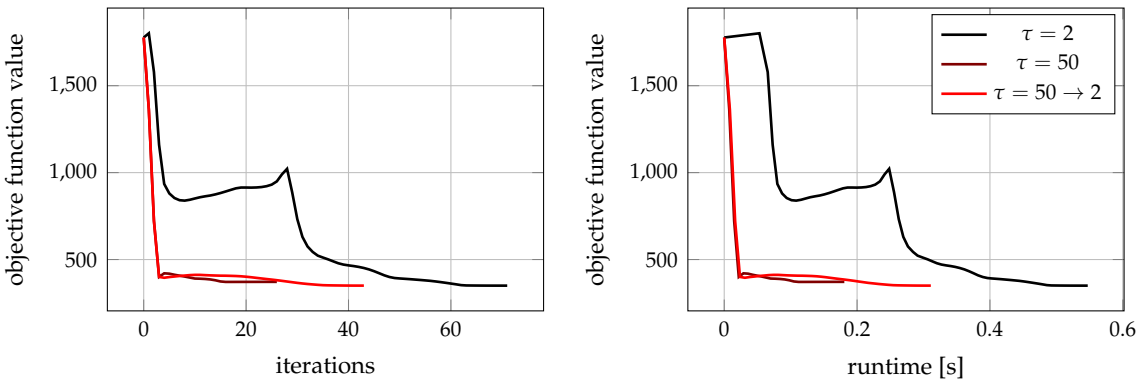
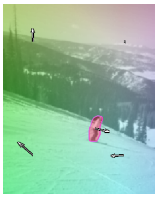


Figure 6.9: Comparison of convergence of the implicit ADMM on the "cars6" sequence with and without a reduction of the step size during the iteration. Objective function value is plotted against the iterations.

point lies to the black line, the smaller is the correct part of the segmented area, or the error around the segmented object is about as large as the object itself.

It can be seen that with increasing step size τ the number of iterations and accuracy decreases. Comparing $\tau = 3$ and $\tau = 7$, the runtime decreases while the accuracy stays almost the same. For some larger τ, this behavior turns around, i.e. the runtime stays almost the same, while the accuracy decreases significantly. This behavior has its optimum in the upper left corner, hence the points nearest to this corner are optimal which are in this case $\tau = 20$. The largest step size $\tau = 100$ does in general not give good results on this example.

Step size adaption. In this paragraph, a step size adaption is tested. The iteration is started with a large step size, and then decreased abruptly to a small step size. The challenging part is to identify the right timing when the step size should be decreased. The problem is that after the reduction of the step size, the objective function value increases a little bit before it decreases. We assume that this is due to a bad initialization of the dual variable, but could not find an explanation or a way to prevent this behavior. Initializations of the dual variable are also studied in section 6.3.3. In this example we chose the iterate when the motion vectors are almost converged to decrease the step size. In Figure 6.9 the convergence of an iteration starting with $\tau = 50$ and then decreasing to $\tau = 2$ is shown, with the convergence of $\tau = 50$ and $\tau = 2$ without changing the step size. In the left plot the objective functional value is plotted against the number of iterations and in the right plot against the runtime. In table 6.2 the number of iterations, the runtime and the value of the objective function is given for fixed step sizes $\tau \in \{2, 5, 50\}$ and for the iterations with a reduction of the step size



algorithm	τ	iterations	time [s]	J
implicit ADMM	2	71	0.64	349.83
implicit ADMM	5	37	0.27	351.01
implicit ADMM	50	26	0.19	371.67
implicit ADMM step size reduction	50→2	43	0.32	349.96
implicit ADMM step size reduction	50→5	39	0.29	350.57

Table 6.2: Step size, number of iterations and objective function value for the implicit ADMM with and without a reduction of the step size during the iteration on the "cars6" sequence.

from $\tau = 50$ to $\tau = 5$ and from $\tau = 50$ to $\tau = 2$. For the reduction from $\tau = 50$ to $\tau = 2$, the objective functional value decreased in comparison to the objective functional value of $\tau = 50$, and decreased the number of iterations and runtime in comparison to the iteration with $\tau = 2$, but, of course takes longer than the iteration with $\tau = 50$. The reduction from $\tau = 50$ to $\tau = 5$ has almost no benefit over the iteration with $\tau = 5$ without a step size reduction for this example. Thus, if a high accuracy is needed, for some examples, it might be beneficial to start the iteration with a large step size and decrease the step size, after a few iterations.

6.3.2 Scaling and Gaussian Smoothing

Another factor, which has an influence on the runtime of the algorithm is the preprocessing of the images by scaling or the convolution with a Gaussian kernel. As already discussed in chapter 2.3 the convolution with a Gaussian kernel or a scaling can be beneficial for optical flow estimation, especially if large displacements are to be expected in the image sequence.
In the shown experiments, all image sequences /image pairs are scaled and smoothed through a convolution with a Gaussian kernel in a preprocessing step. Too much scaling or smoothing causes the moving objects to vanish, and it is not possible to achieve a segmentation. Up to now, the scaling and smoothing was chosen arbitrarily, but since the smoothing may have a positive, but also a negative effect on the segmentation, this parameter is analyzed in the experiments. The results are given in Figure 6.10. The "cars6" sequence is scaled down to three different sizes by a scaling factor of 1, $1/2$ and $1/4$, (top to bottom in Figure 6.10), and smoothed with a Gaussian kernel with different standard deviations $\rho \in \{1, 3, 6\}$. The accuracy of the segmentation results compared to the pseudo ground truth, which is computed for the whole sequence, i.e. for every image pair in the sequence, is plotted against the runtime. For all image sizes the most accurate results are achieved by the smallest standard deviation ρ, shown with red circles. It can be observed, that the accuracy decreases on the smaller images, but

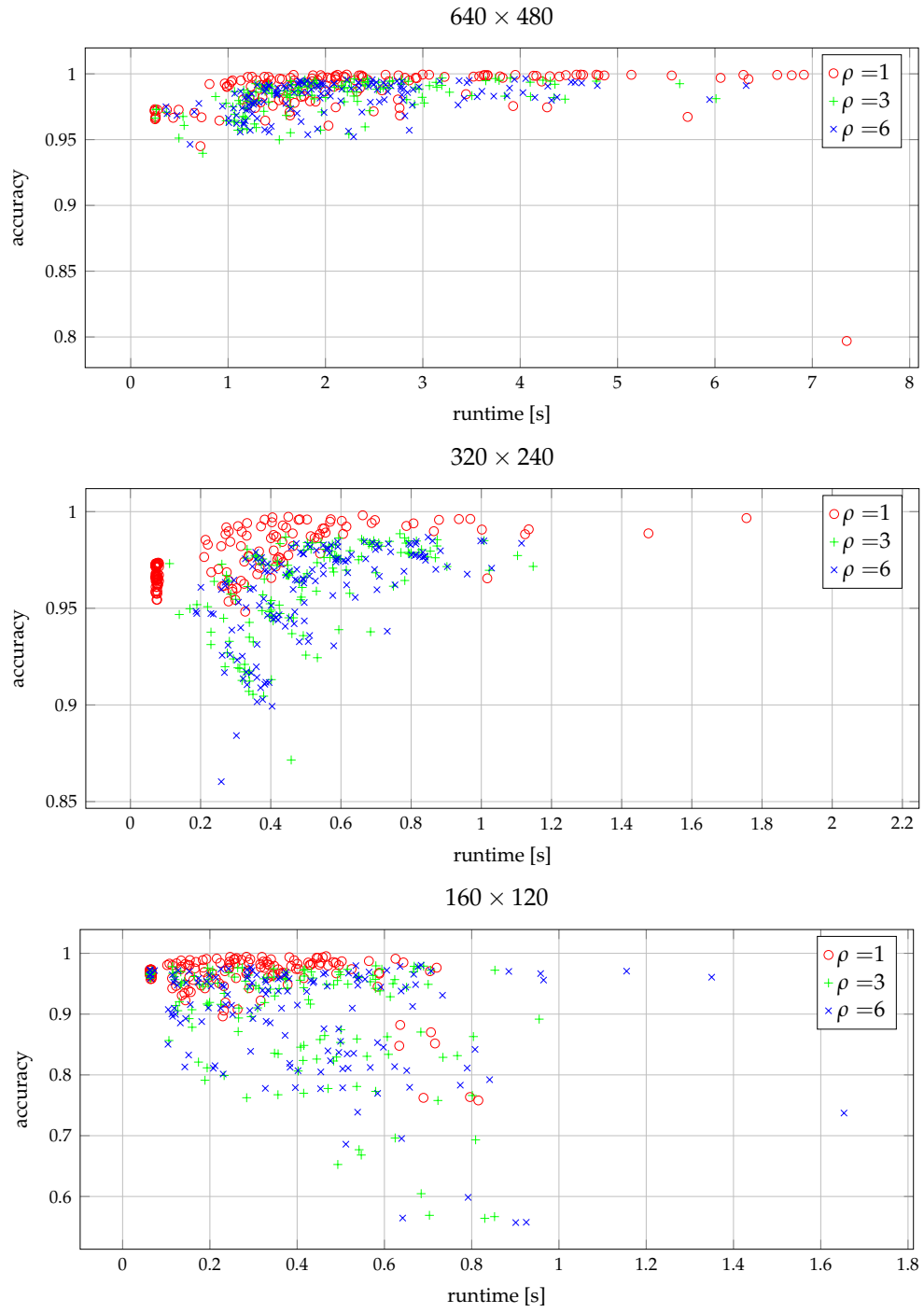
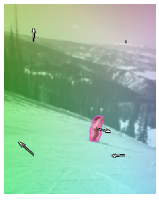


Figure 6.10: For every image pair of the sequence the error is plotted against the accuracy for different sizes and different Gaussian filters with standard deviation ρ for the "cars6" sequence with the implicit ADMM.



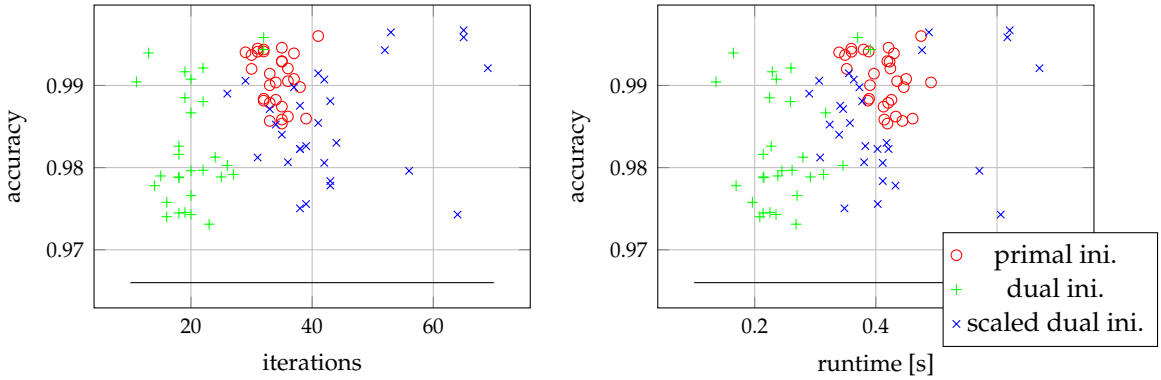


Figure 6.11: For every image pair of the "cars6" sequence the accuracy is plotted against the number of iterations (left) and runtime in seconds (right) with and without a dual initialization in addition to the primal initialization.

the computation times are much faster. However, between the largest and the second largest image size, the accuracy decreases only a little, while the runtime decreases significantly by a factor of 4. Between the second largest and the smallest images, the decrease in runtime is very low, while the accuracy decreases significantly (note the scaling of the vertical axis).
Thus, an optimal parameter combination can be chosen for every image sequence, but the parameter choice is always a choice between speed and accuracy.

6.3.3 Initializations and Pyramid scheme

If a whole image sequence should be segmented, the segmentations u, the primal variable, of previously calculated images can be used as an initialization of consecutive image pairs. The dual variable can be initialized either by $b = 0$, or also with the result of the previously calculated images. Further, in order to speed up the iteration and improve the segmentation, a pyramid scheme can be applied on the images. The motion segmentation is calculated on a scaled image, and the results are used to initialize the unscaled image to get more accurate results. In the experiments the primal variable u and the dual variable b are scaled to the larger grid and used as an initialization. The idea of pyramid schemes is related to multigrid calculations.
In Figure 6.11 the results are shown for a primal initialization with the segmentation u of the previous image pair (red), and for a combined primal and dual initialization with the results of the previous image pair. The black line at the bottom of the plot again visualizes the error of the "empty segmentation", where $u = 0$. The accuracy is plotted against iteration and runtime. The initialization with both, primal and dual variable, gains a plus in speed but loses accuracy.

6.4 Multi-label Motion Segmentation

The experiments for multi-label segmentation are carried out on a multi-label model with multiple labeling functions and on the error-label model. The optimization problems are solved with the inexact implicit ADMM given in algorithm 11.

6.4.1 Models

Multiple labeling functions. As described in chapter 3.2, multiple labeling functions u_i are introduced. We use the following multi-label model from equation (3.9):

$$\begin{aligned} \min_{u_i, v_i} \int_\Omega \sum_i \Big(\|\nabla u_i(x)\|_2 + \mu \langle \nabla f, v_i \rangle^2 u_i(x) \Big) + \mu \langle \nabla f, v_0 \rangle^2 \Big(1 - \sum_i u_i(x) \Big) \\ + \sum_{i,j,\, i \neq j} u_i(x) u_j(x) \, \mathrm{d}x, \\ \text{subject to} \quad u_i(x) \in [0,1] \end{aligned} \tag{3.9}$$

In order to minimize this functional with the (inexact) implicit ADMM, for each labeling function u_i a second variable d_i and a dual variable b_i has to be included. The minimization for the motion vectors v_i and the labeling functions u_i is also carried out alternatingly. The updates for the u_i for fixed motion vectors v_i is carried out sequentially for all i. The inexact implicit ADMM for multi-label motion segmentation is given by:

Algorithm 15 inexact implicit ADMM for multi-label motion segmentation

1. choose penalty parameter $\sigma > 0$, step size $\tau > 0$, number of labels K and initial points $(u_i)_0 \in \mathcal{X}$, $(d_i)_0, (b_i)_0 \in \mathcal{Y}$, for $i = 1, \ldots, K$.
2. iterate for $n = 0, 1, \ldots$
   (i) calculate motion vectors $(v_i)_n$ for every $(u_i)_n$ with (6.9)
   (ii) calculate motion errors $(s_v)_i$ for $i = 0, \ldots, K$
   (iii) iterate for $i = 0, 1, \ldots$,K
   calculate $(u_i, d_i, b_i)_{n+1}$ with algorithm 14

This update scheme can also be performed with any other optimization algorithm presented in the previous sections. However, since the implicit ADMM performed best on the two-label model, and the update steps are equivalent to a two-label update for every labeling function u_i, the implicit ADMM is also used here.

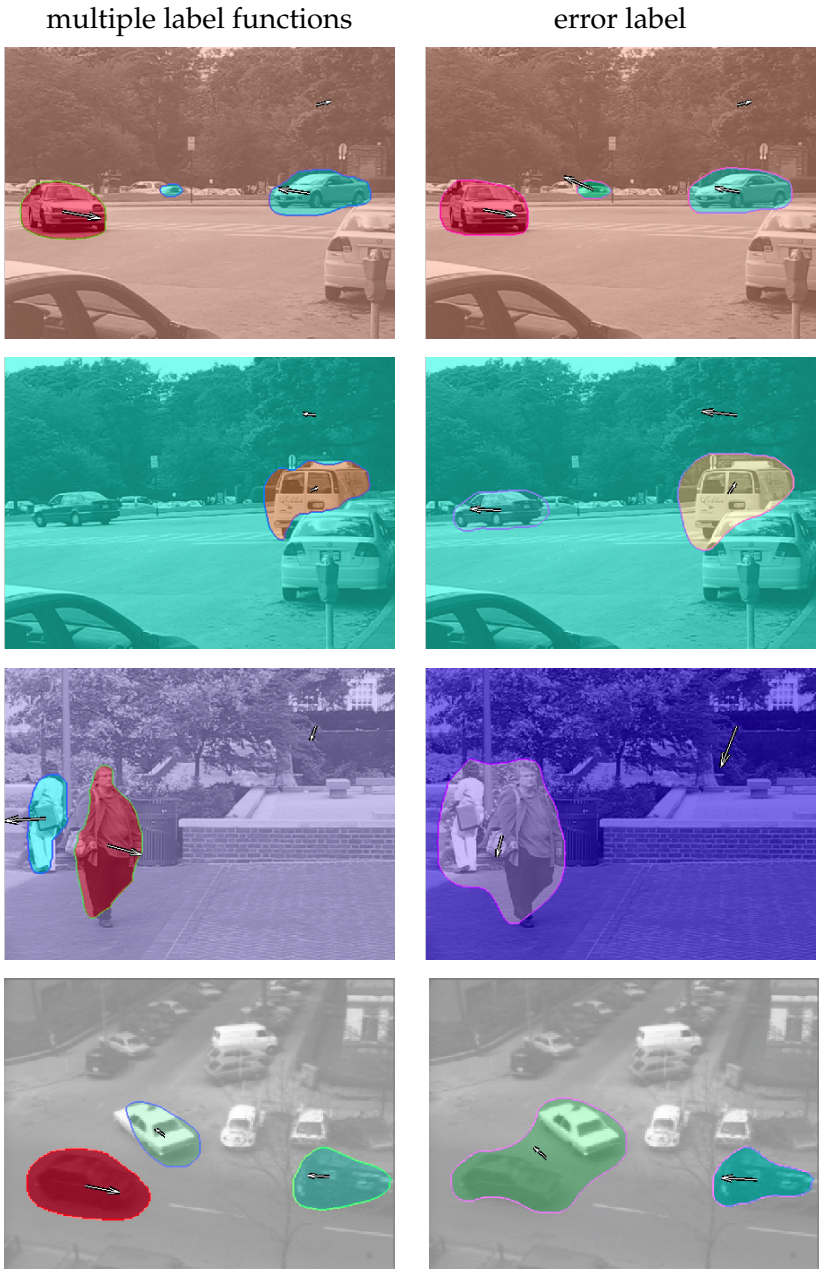


Figure 6.12: Motion segmentation for different examples. Top to bottom: “cars5”, “cars3”, “people2”, “Hamburg taxi”. Multi-label segmentation with different algorithms: left: multiple labeling functions, right: method with one error label. Motion vectors are color coded according to a color wheel: red indicates motion to the right, blue to the lower left corner, green to the upper left corner. Color intensity is proportional to the length of the motion vector.

sequence	algorithm	τ	μ	#labels / ξ	iterations	time [s]
cars5, size: 480×640,	multiple labels	20	3.5	3	28	0.67
scaled: 160×213	error label	20	3.5	1.1	35	0.27
cars3, size: 480×640,	multiple labels	20	2	2	25	0.34
scaled: 160×213	error label	20	3.5	1.1	32	0.23
people2, size: 480×640,	multiple labels	4	2	2	59	0.93
scaled: 160×213	error label	20	3.5	1.1	23	0.17
Hamburg taxi,	multiple labels	10	2.5	3	102	3.80
size: 191×256	error label	20	3.5	1.1	41	0.45

Table 6.3: Step size, parameter μ, ξ or number of labels, number of iterations and runtime in seconds for the two algorithms under comparison and 4 different image sequences.

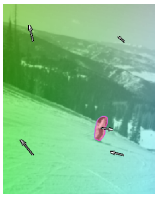
Error-label method. The error-label model from section 3.2.2 is implemented similarly to the two-label model. The objective functional uses a replacement for the data term $s_v = \langle f(x,t), v\rangle^2 - \xi$

$$J(u,v) = \int_\Omega \|\nabla u(x)\|_2 + \mu\Big(\langle \nabla f(x,t), v\rangle^2 - \xi\Big)u(x)\,\mathrm{d}x + \delta_C(u), \tag{6.84}$$

with the error $\xi > 0$. The segmentation is calculated with the inexact implicit ADMM in algorithm 14. The vector v is calculated as described in section 6.1.1, with only one vector. Hence, only the 2 by 2 linear system in equation (6.9) has to be solved.

6.4.2 Results

The motion segmentations are performed on the "cars5", "cars3", and "people2" sequence from the Berkeley segmentation dataset, and on the "Hamburg taxi" sequence. The segmentation results for the model with multiple labeling functions and the error-label model are shown in Figure 6.12. The corresponding parameters, in particular, the step size τ, weighting parameter μ, the number of labeling functions used in the computation and the parameter ξ for the error-label, are given in table 6.3 together with the number of iterations and the runtime. The parameter σ for the quadratic penalty of the augmentation term in the Lagrangian is $\sigma = 2$ for all calculations. Due to the new parameter ξ, the parameter μ in the error label model has to be chosen larger than in the model with multiple labels. Since the modeling is different, the objective function value cannot be compared.
In Figure 6.12, the segmentation produced by the model with multiple labeling functions (left), is shown by the contour lines $u_i = 0.5$, marked with different colors. In the



segmentation of the "cars5" sequence, two objects that move in similar directions are labeled with only one function u_i.
In table 6.3, it can be seen, that the error-label model is faster on every sequence. In the first sequences, i.e. "cars5", the motion segmentation is equally good for both methods, and for the "cars3" sequence, the motion segmentation of the error-label model is better, since the moving car in the left of the image is also detected, which moves almost in the same direction as the moving background. However, for moving objects that are close to each other the error-label model only computes one object and the mean vector in this region. This can be observed in the "people2" and "Hamburg taxi" sequence. In order to separate these objects, further post processing steps are needed. One idea is to perform a two-label segmentation on the segmented region, or to test the deviations from the optical flow constraint for different test vectors, but the post processing step for the error-label model is a topic of future work.

6.5 Affine Motion Segmentation

The affine motion model is used in the two-label model and in the error-label model. While for small foreground objects a constant motion vector is usually a good choice, the background motion can be a rotation or a more complex camera motion. Thus, in the following experiments, the background is modeled by the affine model, while the foreground is modeled by a constant vector. In order to enforce the affine motion model on the background, an additional weighting parameter is used, as described in chapter 2. As for the models with constant vectors, the minimization of the segmentation and the affine motion parameters is performed in an alternating scheme.

6.5.1 Models

The models for motion segmentation with affine background movement from chapter 3.3 are used here, in particular the two-label model and the error-label model. Both models can be solved by the algorithms for the two-label model, but since the implicit ADMM performed best on the two-label model, we also use this algorithm here.

Affine two-label model. For the two-label model with affine motion, the following functional is used in the experiments: The resulting two-label minimization problem reads as follows:

$$J(u) = \int_\Omega \|\nabla u(x)\|_2 + \mu \langle \nabla f, v\rangle^2 u(x) + \mu \langle \nabla f, Hx\rangle^2 \Big(1 - u(x)\Big) + \nu u(x)\, \mathrm{d}x + \delta_C(u),$$

where $\nu > 0$ is a parameter enforcing the background to be modeled with the affine motion model.

sequence	algorithm	τ	μ	ξ	iterations	time [s]
cars7, size: 480×640,	two-label affine	10	2	-	19	0.18
scaled: 160×213	error label affine	20	1.8	1.1	49	0.53
skiing2, size: 1001×800,	two-label affine	10	2	-	14	0.50
scaled: 334×267	error label affine	20	2	1.5	30	1.03

Table 6.4: Step size, weighting parameters μ, ξ, number of iterations and runtime in seconds for the two label algorithm and the error label algorithm with an affine background model for the motion vector.

Affine error-label model. The error-model also consists of a penalty for the background to be modeled by an affine motion model, i.e. the background is modeled with an affine motion, while the foreground will be segmented by the error label. The model used in the experiments is given by:

$$\min_{u,H} \int \|\nabla u(x)\|_2 + \mu\Big(\xi u(x) + \langle \nabla f, Hx\rangle^2 \left(1 - u(x)\right)\Big) + \delta_{\mathcal{C}}(u)\, \mathrm{d}x, \tag{6.85}$$

where $\xi > 0$ is the weighting for the error-label.

6.5.2 Affine Parameters

The affine motion is given by the matrix $H \in \mathbb{R}^{3,2}$:

$$v = Hx = \begin{pmatrix} h_1 & h_3 & h_5 \\ h_2 & h_4 & h_6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}, \tag{6.86}$$

where x_1 and x_2 are the two image coordinates.
The entries of the matrix H, h_i, $i = 1, \dots 6$, for a fixed segmentation u are calculated through a system of linear equations, similar to the calculation of the constant motion vectors. Let $X = (x_1, x_2, 1)^T$ and I_{x_1}, I_{x_2}, I_t denote the partial derivatives of the image, $\partial_{x_1} I, \partial_{x_2} I, \partial_t I$, respectively. With

$$XX^T = \begin{pmatrix} x_1^2 & x_1 x_2 & x_1 \\ x_1 x_2 & x_2^2 & x_2 \\ x_1 & x_2 & 1 \end{pmatrix} \tag{6.87}$$

the matrix $A_u \in \mathbb{R}^{6\times 6}$ and vector $b_u \in \mathbb{R}^6$ are given by

$$A_u = \begin{bmatrix} \int_\Omega u \left(I_{x_1}\right)^2 [XX^T] & \int_\Omega u I_{x_1} I_{x_2} [XX^T] \\ \int_\Omega u I_{x_2} I_{x_1} [XX^T] & \int_\Omega u \left(I_{x_2}\right)^2 [XX^T] \end{bmatrix}, \quad b_u = \begin{bmatrix} \int_\Omega u I_{x_1} I_t X \\ \int_\Omega u I_{x_2} I_t X \end{bmatrix}, \tag{6.88}$$

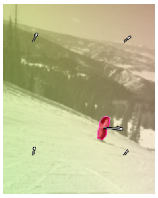


Figure 6.13: Motion segmentation for different examples. Top to bottom: “cars7”, “skiing2”. Left: Two-label segmentation with affine background, right: error-label segmentation with affine background. Motion vectors are color coded according to a color wheel: red indicates motion to the right, blue to the lower left corner, green to the upper left corner. Color intensity is proportional to the length of the motion vector.

The matrix $h \in \mathbb{R}^6$ is given by

$$A_u h = b_u. \tag{6.89}$$

As in the calculation of the constant motion vectors, the matrix H for the second region is calculated through

$$A_{1-u} h = b_{1-u}. \tag{6.90}$$

6.5.3 Results

The calculations are performed by the implicit ADMM on the "cars7" sequence from the Berkeley segmentation database, and a self-made sequence called "skiing2". In both sequences, the camera is rotated around the optical axis while following the moving object roughly. In Figure 6.13 the motion segmentations are given for the sequences with affine background motion. The vectors are color-coded, as for the previous motion segmentations, i.e. in the case of affine motion, the background is multicolored according to the movement of the background induced by the camera rotation. In the image pair of the "cars7" sequence depicted in 6.13, the background rotates clockwise and moves to the right, and in the image pair of "skiing2" sequence, the background rotates clockwise and moves to the left. Both segmentations appear to be equally good in quality. In table 6.4, the values for the step size τ, parameter μ, and ξ, for the error-label, as well as the iterations and runtimes are given. It turns out, that the two-label model is faster than the error-label model, even though the error-label model uses a larger step-size.
In the lower right corner the affine motion segmentation of the "skiing2" sequence is included as a flip-book.

6.6 Implicit ADMM for Denoising and Inpainting

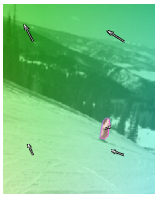
In this section, we compare the primal-dual algorithm with the implicit ADMM on further imaging problems.

Denoising. First we revisit the denoising example from Chapter 2, with TV-regularization, also called the Rudin-Osher-Fatemi (ROF) model, cf. [Rudin et al., 1992]

$$J(u) = \int_\Omega \|\nabla u(x)\|_2 + \frac{\mu}{2}(u(x) - f(x))^2 \, \mathrm{d}x, \tag{6.91}$$

where $f : \Omega \to \mathbb{R}$ is the noisy image, u is the reconstruction and μ is a parameter.
The primal-dual algorithm for ROF denoising can be found in [Chambolle and Pock, 2011].
A splitting is used to introduce the variable $d = \nabla u$, and the augmented Lagrangian reads

$$\mathcal{L}_\sigma(u, d, b) = \int_\Omega \|d(x)\|_2 + \frac{\mu}{2}(u(x) - f(x))^2 + \frac{\sigma}{2}\|\nabla u(x) - d(x) + b(x)\|^2 \, \mathrm{d}x. \tag{6.92}$$



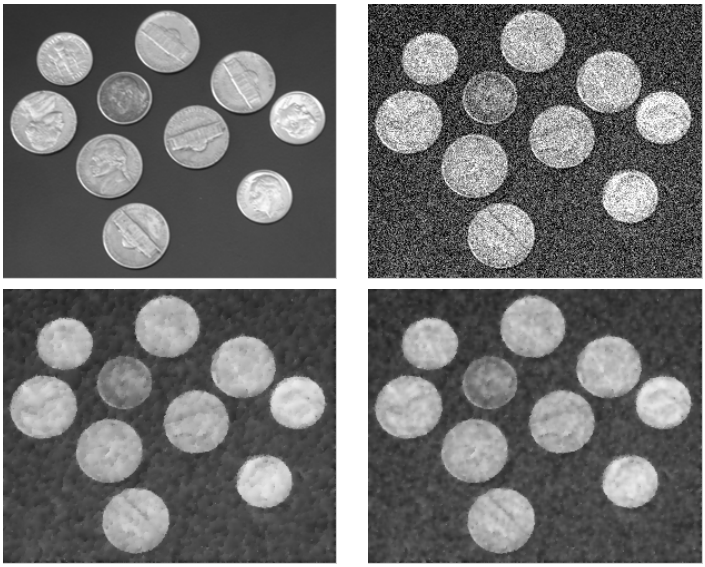


Figure 6.14: Upper row: original and noisy image. Lower row: denoising with the primal-dual algorithm (left) and the implicit ADMM (right).

Since there are no constraints on the function u, the implicit update step can be evaluated explicitly, i.e. without an approximation. The implicit ADMM for the denoising model is given by

Algorithm 16 implicit ADMM for denoising

1. choose penalty parameter $\sigma > 0$, step size $\tau > 0$ and initial points $u_0 \in \mathcal{X}$, $d_0, b_0 \in \mathcal{Y}$.

2. iterate for $k = 0, 1, \ldots$

$$u_{k+1} = (I + \tau\mu I - \tau\sigma\Delta)^{-1}\Big(u_k + \tau\big(\sigma\mathrm{div}(b_k - d_k) - \mu f\big)\Big) \tag{6.93}$$

$$d_{k+1} = \left(1 - \frac{1}{\max(1, \sigma\|\nabla u_{k+1} + b_k\|_2)}\right)(\nabla u_{k+1} + b_k) \tag{6.94}$$

$$b_{k+1} = b_k + \nabla u_{k+1} - d_{k+1}. \tag{6.95}$$

The convergence results are given in Figure 6.15. Even though the implicit ADMM needs about half as many iterations as the primal-dual algorithm, the overall runtime of the implicit ADMM is longer than the primal-dual algorithm. The results of the denoising are equally good, which can be seen in Figure 6.14. In this example the extrapolation parameter for the primal-dual algorithm is chosen as $\theta = 0.8$ which reduced

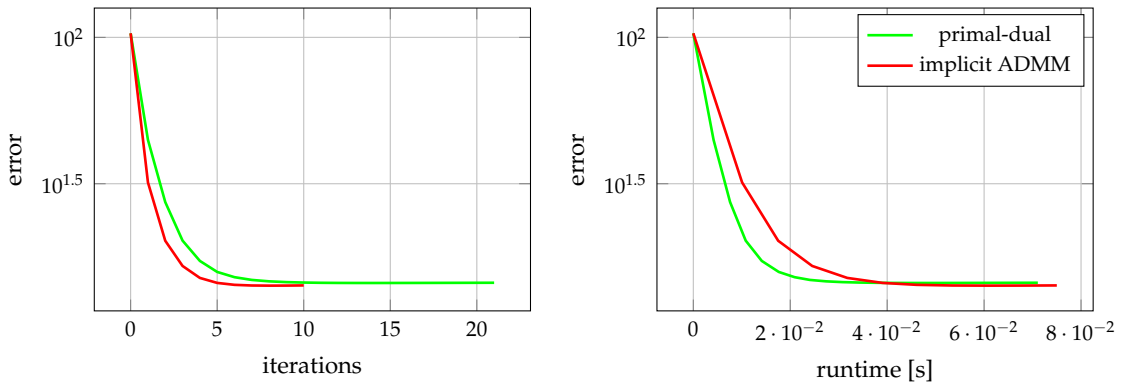


Figure 6.15: Comparison of convergence of the primal-dual algorithm and the implicit ADMM on the denoising problem. The error is plotted against the iterations (left) and runtime in seconds (right).

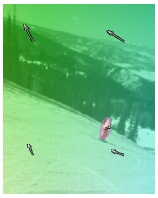
the number of iterations. A topic of future work is, whether the extrapolation parameter can be included in the implicit ADMM through similar calculations as in section 4.3.3, and it also yields an acceleration for the implicit ADMM.

Inpainting. As a second example we analyze the task of joint inpainting and denoising. The variational model, already given in section 2.4, reads

$$\min_u \int_\Omega \|\Phi u\| + \frac{\mu}{2} \int_{\Omega \setminus \Omega'} (u(x) - f(x))^2 \, \mathrm{d}x, \tag{6.96}$$

where the operator Φ is chosen as a fast discrete curvelet transform, which according to [Chambolle and Pock, 2011] leads to better results than a total variation regularization. An advantage of the discrete curvelet transformation is that the inverse curvelet transform is simply computed as the adjoint of the forward transform $\Phi^*\Phi = I$, cf. [Candes et al., 2006]. In [Chambolle and Pock, 2011], also the primal-dual algorithm for the given inpainting model can be found.

The inpainting domain Ω' is modeled through the diagonal matrix $D \in \{0,1\}^{N \times N}$. As for the denoising problem, the implicit update step can be evaluated explicitly. The implicit ADMM for the inpainting model reads



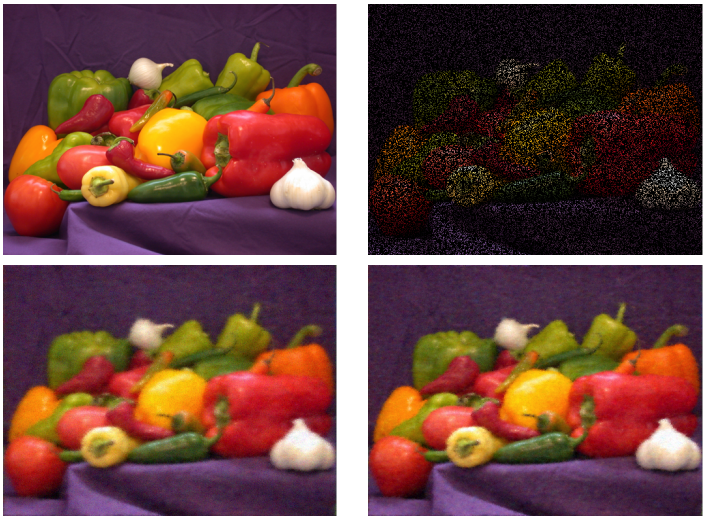


Figure 6.16: Upper row: original and noisy image. Lower row: inpainting with the primal-dual algorithm (left) and the implicit ADMM (right).

Algorithm 17 implicit ADMM for inpainting

1. choose penalty parameter $\sigma > 0$, step size $\tau > 0$ and initial points $u_0 \in \mathcal{X}$, $d_0, b_0 \in \mathcal{Y}$.

2. iterate for $k = 0, 1, \ldots$

$$u_{k+1} = (I - \tau\sigma I + \tau\mu D)^{-1}\Big(u_k - \tau\big(\sigma\Phi^*(b_k - d_k) - \mu D f\big)\Big) \tag{6.97}$$

$$d_{k+1} = \left(1 - \frac{1}{\max(1, \sigma\|\Phi u_{k+1} + b_k\|_2)}\right)(\Phi u_{k+1} + b_k) \tag{6.98}$$

$$b_{k+1} = b_k + \Phi u_{k+1} - d_{k+1}. \tag{6.99}$$

The inpainting results together with the original and the image with missing data are given in Figure 6.16. The lower left picture shows the result of the primal-dual algorithm and the lower right picture shows the result of the implicit ADMM. In Figure 6.17 the convergence results of both algorithms are given. The implicit ADMM again only needs about half of the iterations of the primal-dual algorithm, but for the inpainting problem with a discrete curvelet transform, the computation of one iteration is equally fast.

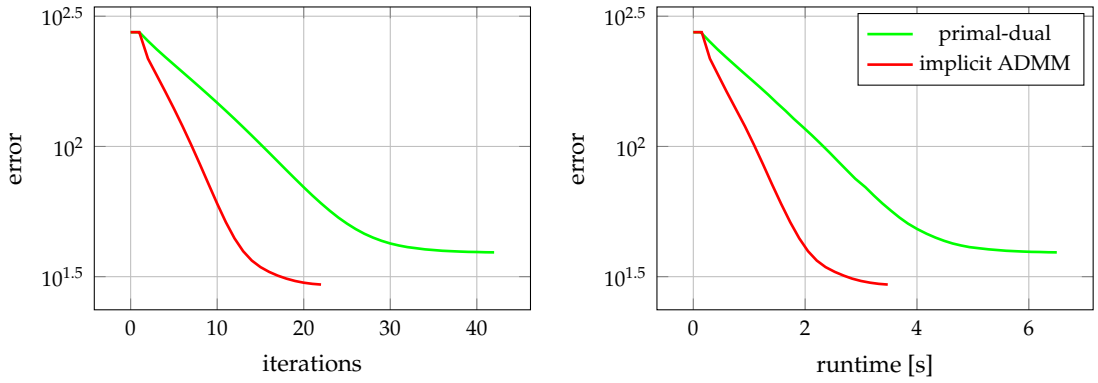


Figure 6.17: Comparison of convergence of the primal-dual algorithm and the implicit ADMM on the inpainting problem. The error is plotted against the iterations (left) and runtime in seconds (right).

7 Conclusion

Conclusion. We have proposed a new proximal algorithm for the minimization of convex functionals over convex sets which uses an implicit gradient update instead of a half explicit one in one descent step. The new algorithm is stable without restriction on the step size, hence it is not prone to stiff systems. Computational challenges with a scaled proximal operator appearing in our algorithm can be handled through an approximation. We investigated convergence properties of the exact and the approximate algorithm, thereby relying on the theory of monotone and averaged operators.
In the numerical experiments, the implicit algorithm is included into an alternating direction method of multipliers (ADMM) routine, where it requires significantly fewer iterates than standard schemes. As a tradeoff, in each step a linear system of equations has to be solved. In problems where this can be performed efficiently significant runtime improvements can be achieved. This is backed by our numerical experiments on the motion segmentation problem with several standard and self-made image sequences. Further, we proposed variational two-label and multi-label motion segmentation models and developed a new model for multi-label motion segmentation based on the two-label model, the error-label model, which performs a multi-label motion segmentation almost as fast as a two-label motion segmentation.

Future Work. We have shown, that the linearized ADMM is equivalent to the primal-dual algorithm for the extrapolation parameter $\theta = 1$, and that this extrapolation step can also be included in the linearized ADMM. The extrapolation can speed up convergence for the primal-dual algorithm, and the question is, if it can be included also into the implicit ADMM, and whether it can improve the convergence.
Also, as indicated by the numerical results in sections 6.3.3 and 6.3.1, the inititialization of the dual variable influences the convergence. Further studies on the initialization of the dual variable could lead to faster convergence in pyramid schemes, and could improve step size adaptation strategies.
Moreover, the implicit proximal method can be extended to solve problems of the form $F(Ax) + G(Bx)$, where F and G are convex functions, and A and B are linear operators. With this extension, the implicit proximal method can be applied to the function lifting models. Also, a multi-label motion segmentation model based on the function lifting idea could be developed.
Furthermore, the error-label model can be extended with additional post processing steps to separate remaining areas in the error-label. This can be achieved by a two-label

motion segmentation on the remaining area, or, if the areas are small, by a successive partitioning of the area and calculating vectors on each partition. Depending on the error produced by the calculated vector, the area is partitioned again, until the error is small.

Bibliography

[Arrow et al., 1958] Arrow, K. J., Hurwicz, L., and Uzawa, H. (1958). *Studies in linear and non-linear programming*. Cambridge Univ. Press.

[Ayvaci et al., 2010] Ayvaci, A., Raptis, M., and Soatto, S. (2010). Occlusion detection and motion estimation with convex optimization. In Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23*, pages 100–108. Curran Associates, Inc.

[Banach, 1922] Banach, S. (1922). Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fund. Math*, 3(1):133–181.

[Bauschke and Combettes, 2011] Bauschke, H. H. and Combettes, P. L. (2011). *Convex analysis and monotone operator theory in Hilbert spaces*. Springer.

[Boyd et al., 2011] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122.

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, New York, NY, USA.

[Bredies and Lorenz, 2011] Bredies, K. and Lorenz, D. (2011). *Mathematische Bildverarbeitung*. Vieweg+ Teubner.

[Bregman, 1967] Bregman, L. (1967). The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200 – 217.

[Brown et al., 2009] Brown, E. S., Chan, T. F., and Bresson, X. (2009). Convex formulation and exact global solutions for multi-phase piecewise constant Mumford-Shah image segmentation. Technical report, DTIC Document.

[Brown et al., 2012] Brown, E. S., Chan, T. F., and Bresson, X. (2012). Completely convex formulation of the Chan-Vese image segmentation model. *International Journal of Computer Vision*, 98:103–121.

[Brox et al., 2006] Brox, T., Bruhn, A., and Weickert, J. (2006). Variational motion segmentation with level sets. In Leonardis, A., Bischof, H., and Pinz, A., editors, *Computer Vision - ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 471–483. Springer Berlin Heidelberg.

[Brox and Malik, 2010] Brox, T. and Malik, J. (2010). Object segmentation by long term analysis of point trajectories. In Daniilidis, K., Maragos, P., and Paragios, N., editors, *Computer Vision - ECCV 2010*, volume 6315 of *Lecture Notes in Computer Science*, pages 282–295. Springer Berlin Heidelberg.

[Brox and Weickert, 2004] Brox, T. and Weickert, J. (2004). Level set based image segmentation with multiple regions. In Rasmussen, C., Bülthoff, H., Schölkopf, B., and Giese, M., editors, *Pattern Recognition*, volume 3175 of *Lecture Notes in Computer Science*, pages 415–423. Springer Berlin Heidelberg.

[Bruhn, 2006] Bruhn, A. (2006). *Variational optic flow computation: accurate modelling and efficient numerics*. PhD thesis, Department of Mathematics and Computer Science, Saarland University, Saarbrücken.

[Bruhn et al., 2006] Bruhn, A., Weickert, J., Kohlberger, T., and Schnörr, C. (2006). A multigrid platform for real-time motion computation with discontinuity-preserving variational methods. *International Journal of Computer Vision*, 70(3):257–277.

[Bruhn et al., 2005] Bruhn, A., Weickert, J., and Schnörr, C. (2005). Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231.

[Candes et al., 2006] Candes, E., Demanet, L., Donoho, D., and Ying, L. (2006). Fast discrete curvelet transforms. *Multiscale Modeling & Simulation*, 5(3):861–899.

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

[Chan et al., 2004] Chan, T. F., Esedoglu, S., and Nikolova, M. (2004). Algorithms for finding global minimizers of image segmentation and denoising models. Technical report, SIAM Journal on Applied Mathematics.

[Chan et al., 2000] Chan, T. F., Sandberg, B. Y., and Vese, L. A. (2000). Active contours without edges for vector-valued images. *Journal of Visual Communication and Image Representation*, 11:130–141.

[Chan and Shen, 2005] Chan, T. F. and Shen, J. J. (2005). *Image processing and analysis: variational, PDE, wavelet, and stochastic methods*. Siam.

[Cremers and Soatto, 2005] Cremers, D. and Soatto, S. (2005). Motion competition: a variational approach to piecewise parametric motion segmentation. *International Journal of Computer Vision*, 62:249–265.

[Eckstein and Bertsekas, 1992] Eckstein, J. and Bertsekas, D. P. (1992). On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3):293–318.

[Esser, 2009] Esser, E. (2009). Applications of Lagrangian-based alternating direction methods and connections to split Bregman. *CAM report*, 9:31.

[Esser et al., 2010] Esser, E., Zhang, X., and Chan, T. (2010). A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science. *SIAM Journal on Imaging Sciences*, 3(4):1015–1046.

[Goldstein et al., 2010] Goldstein, T., Bresson, X., and Osher, S. (2010). Geometric applications of the split Bregman method: segmentation and surface reconstruction. *Journal of Scientific Computing*, 45(1-3):272–293.

[Goldstein and Osher, 2009] Goldstein, T. and Osher, S. (2009). The split Bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343.

[Horn and Schunck, 1981] Horn, B. K. P. and Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence*, 17:185–203.

[Lee et al., 2012] Lee, J. D., Sun, Y., and Saunders, M. A. (2012). Proximal Newton-type methods for minimizing composite functions. *arXiv preprint arXiv:1206.1623*.

[Li et al., 2010] Li, F., Shen, C., and Li, C. (2010). Multiphase soft segmentation with total variation and H1 regularization. *Journal of Mathematical Imaging and Vision*, 37(2):98–111.

[Lucas et al., 1981] Lucas, B. D., Kanade, T., et al. (1981). An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679.

[Milzarek, 2015] Milzarek, A. (2015). Second-order conditions for a class of nonsmooth optimization problems. unpublished.

[Möllenhoff et al., 2013] Möllenhoff, T., Nieuwenhuis, C., Töppe, E., and Cremers, D. (2013). Efficient convex optimization for minimal partition problems with volume constraints. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 94–107. Springer.

[Moreau, 1962] Moreau, J.-J. (1962). Fonctions convexes duales et points proximaux dans un espace hilbertien. *CR Acad. Sci. Paris Sér. A Math*, 255:2897–2899.

[Mumford and Shah, 1989] Mumford, D. and Shah, J. (1989). Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, 42(5):577–685.

[Nesterov, 2005] Nesterov, Y. (2005). Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152.

[Nieuwenhuis et al., 2013] Nieuwenhuis, C., Töppe, E., and Cremers, D. (2013). A survey and comparison of discrete and continuous multi-label optimization approaches for the Potts model. *International journal of computer vision*, 104(3):223–240.

[Otte and Nagel, 1995] Otte, M. and Nagel, H.-H. (1995). Estimation of optical flow based on higher-order spatiotemporal derivatives in interlaced and non-interlaced image sequences. *Artificial Intelligence*, 78(1):5–43.

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

[Pock et al., 2009] Pock, T., Cremers, D., Bischof, H., and Chambolle, A. (2009). An algorithm for minimizing the Mumford-Shah functional. In *IEEE 12th International Conference on Computer Vision*, pages 1133–1140, Kyoto, Japan. IEEE.

[Pock et al., 2008] Pock, T., Schoenemann, T., Graber, G., Bischof, H., and Cremers, D. (2008). A convex formulation of continuous multi-label problems. In Forsyth, D., Torr, P., and Zisserman, A., editors, *Computer Vision - ECCV 2008*, volume 5304 of *Lecture Notes in Computer Science*, pages 792–805. Springer Berlin Heidelberg.

[Popov, 1980] Popov, L. D. (1980). A modification of the Arrow-Hurwitz method of search for saddle points. *Mat. Zametki*, 28(5):777–784, 803.

[Rockafellar, 1976] Rockafellar, R. T. (1976). Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898.

[Rudin et al., 1992] Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268.

[Setzer, 2011] Setzer, S. (2011). Operator splittings, Bregman methods and frame shrinkage in image processing. *International Journal of Computer Vision*, 92(3):265–280.

[Steger et al., 2007] Steger, C., Ulrich, M., and Wiedemann, C. (2007). *Machine Vision Algorithms and Applications*. Wiley-VCH, Weinheim.

[Tichmann and Junge, 2014] Tichmann, K. and Junge, O. (2014). A fully implicit alternating direction method of multipliers for the minimization of convex problems with an application to motion segmentation. In *2014 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 823–830. IEEE.

[Tikhonov, 1963] Tikhonov, A. N. (1963). On the regularization of ill-posed problems (russian). *Dokl. Akad. Nauk SSSR*, 153:49–52.

[Unger et al., 2012] Unger, M., Werlberger, M., Pock, T., and Bischof, H. (2012). Joint motion estimation and segmentation of complex scenes with label costs and occlusion modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1878–1885. IEEE.

[Vese and Chan, 2002] Vese, L. A. and Chan, T. F. (2002). A multiphase level set framework for image segmentation using the Mumford and Shah model. *International Journal of Computer Vision*, 50:271–293.

[Yin et al., 2008] Yin, W., Osher, S., Goldfarb, D., and Darbon, J. (2008). Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing. *SIAM Journal on Imaging Sciences*, 1(1):143–168.

[Zach et al., 2007] Zach, C., Pock, T., and Bischof, H. (2007). A duality based approach for realtime TV-L1 optical flow. In *Pattern Recognition*, pages 214–223. Springer.