Graz University of Technology

Institute for Computer Graphics and Vision

Dissertation

Convex Optimization for Image Segmentation

Markus Unger

Graz, October 2012

Thesis Supervisors

Horst Bischof
Hugues Talbot

Thesis Advisor

Thomas Pock

To my family.

Abstract

Segmentation is one of the fundamental low level problems in computer vision. Extracting objects from an image gives rise to further high level processing as well as image composing. A segment not always has to correspond to a real world object, but can fulfill any coherency criterion (e.g. similar motion). Segmentation is a highly ambiguous task, and usually requires some prior knowledge. This can either be obtained by interactive user input in an supervised manner, or completely unsupervised using strong prior knowledge. In this thesis we use continuous energy minimization to tackle all of these problems.

Continuous energy minimization provides an elegant way to model a problem like image segmentation. If the problem is convex, there are powerful optimization algorithms available. Additionally, we are guaranteed to find the globally optimal solution. We give an extensive introduction to convex optimization methods in computer vision. A great part of this thesis is devoted to basic image segmentation. We investigate the continuous maximum flow model for the two label segmentation, as well as optimization problems for multi-label segmentation.

To obtain good segmentation results in a reasonable time, it is important that the energy, optimization algorithm and implementation are perfectly matched. For non-smooth convex optimization problems, primal-dual optimization methods deliver very good convergence rates. Furthermore, they are inherently parallel, and therefore perfectly suited for modern parallel hardware like graphics processors. While we achieve good results with existing optimization methods, we also demonstrate how to further speed up convergence for some specific energies. We have to keep all this in mind during the design of the energy. As a result this thesis tackles the whole process from designing an energy minimization problem, over the algorithmic optimization to the final implementation on the GPU.

Apart from the general segmentation algorithms, we also show four different applications: First, interactive color image segmentation, where the developed segmentation models are applied. Second, tracking as segmentation of spatio-temporal volumes. In this case a video is interpreted as a volume, and objects are segmented in a semi-supervised

manner. Third, we develop an unsupervised approach for segmenting 2.5D depth images. Finally, we present an approach for joint motion estimation, segmentation and occlusion handling.

Keywords. image segmentation, convex optimization, variational methods, interactive, tracking, motion estimation, parallel computing

Statutory Declaration

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

Graz, October 1, 2012 (signature)

Eidesstattliche Erklärung

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

Graz, Oktober 1, 2012 (Unterschrift)

Acknowledgments

Working at the institute for computer graphics and vision over the course of the last years has been one of the greatest times in my life. There is so much I learned on optimization, computer vision, doing proper research etc. Pursuing my PhD meant lots of hours of hard work, several setbacks and (especially while writing this thesis) a sacrifice of my leisure time. But it also meant meeting great people, hours of entangling discussions, moments of great excitement while working on new ideas and finally the satisfaction of solving a problem and pushing knowledge a tiny little bit. All this would not have happened without the support, guidance, criticism and friendship of the people I met along the way.

I like to thank Horst Bischof for persuading me to computer vision with his very first lecture, and his continuous guidance till the end of my thesis. With his knowledge, ideas and management, he created an environment that was simply a great place to be and do research. I also like to thank Hugues Talbot for being my second supervisor.

I owe very much to Thomas Pock for inspiring, guiding and challenging me. His enthusiasm and dedication always acted as a great example. Working with him was a truly exceptional experience. I also like to thank Jakob Santner and Manuel Werlberger for sharing an office, research interests and much more. Thanks, to all my co-authors as well as colleagues at the institute. You all made this a great and fun place to work!

A very special thanks goes to my parents for enabling me to study and supporting me in everything I do. I also like to thank my brother Michael and my friends for their support and friendship. I am deeply grateful for the love of my wife Damaris and her continuously backing me during demanding times. Finally, I like to thank my daughter Sophie for making all trouble go away with a single smile.

Contents

List of Algorithms

Chapter 1

Introduction

1.1 What is image segmentation?

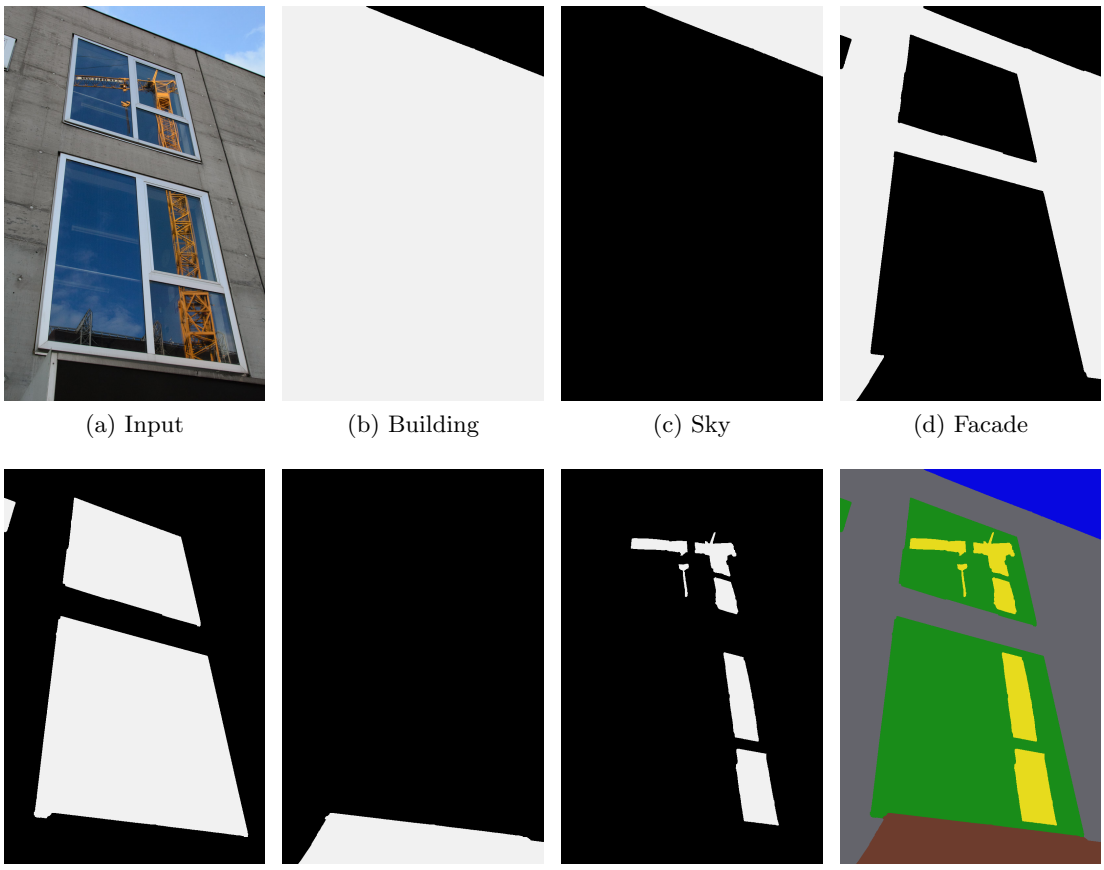
This thesis deals with convex optimization methods for image segmentation. As a start, we first consider a definition of the general image segmentation problem. We assume that images are functions $I : \Omega \to \mathbb{R}^c$ given on some domain Ω. For 2D images, the domain Ω is of dimensionality 2, and for volumetric data or videos, Ω is of dimensionality 3. The function maps to some real valued vector. For color images we have $c = 3$, and for gray-scale images $c = 1$, the mapping reduces to a scalar.

We define the segmentation problem such that the image domain Ω is partitioned into K disjoint regions Ω_i:

$$\Omega = \bigcup_{i=1}^{K} \Omega_i \ , \quad \Omega_i \cap \Omega_j = \emptyset \quad \forall i \neq j \ . \tag{1.1}$$

The case $K = 2$ is referred to as binary segmentation. It partitions the image domain into foreground and background. The case $K > 2$ is also called multi-label segmentation.

The segmentation problem is highly ambiguous. As an example, see Figure 1.1, where different objects of a natural scene are segmented. In (b-g), binary segmentation problems have been solved to obtain segmentations of building, sky, facade, windows, porch and the reflection of a crane. We depicted the foreground in white and the background in black. In (h), a multi-label segmentation ($K = 5$) was performed, where each color corresponds to a label. We can make several observations: Segmentations can be equivalent to each other in the sense that they are inverse e.g. building and sky. Some segmentation regions represent subsets of other regions e.g. the windows are a subset of the building, the reflection of



(a) Input (b) Building (c) Sky (d) Facade

(e) Windows (f) Porch (g) Crane (h) Multi-label

Figure 1.1: Illustration of different segmentations of a natural scene. (b-g) depict binary segmentation problems with bright foreground and black background. (h) shows a multi-label segmentation result.

the crane is a subset of the windows as well as the building. A region Ω_i does not need to be spatially connected e.g. windows and crane. As stated in (1.1), we assume regions to be disjoint $\Omega_i \cap \Omega_j = \emptyset$. This means that only one label can be assigned to a pixel, as can be seen in the multi-label segmentation result in Figure 1.1(h). As a consequence objects may occlude each other (e.g. the crane occludes the window). Of course another definition of the segmentation problem would be possible, that allows to assign multiple regions to a single pixel. We restrict ourselves to the segmentation problem definition in (1.1).

We could go on with additional questions like: Why was the reflection of the sky in the window not labeled as sky? Should not a crane form a single coherent region? etc. It

is not possible to model the segmentation problem in general, as it is highly ambiguous. As a consequence, one has to utilize additional information for each problem separately. The prior information reaches from user input, over priors on the connectivity, to complex models of the label data.

We will present segmentation models that include assumptions on the boundary of objects in Chapter 3. In Chapter 4, we will show how user input can be utilized to perform interactive segmentation. Fully automatic approaches, that rely on special models for the labels, are presented in Chapter 5.

1.2 Related work on image segmentation

There is a large body of different segmentation methods available. In the following, we put the focus only on methods relevant to the topic of this thesis.

One of the most simple segmentation can be obtained by thresholding the image based on some color or texture information [Sezgin and Sankur, 2004]. While thresholding methods generally perform on a pixel level, they could also incorporate spatial information. Similarly, clustering or classification methods can be used for image segmentation. A typical clustering based segmentation method is the mean shift segmentation [Comaniciu et al., 2002].

Instead of basing decisions on the pixel level, region based methods take into account homogeneity criteria inside a complete area. Examples are the split and merge approach [Horowitz and Pavlidis, 1974], the watershed transform [Vincent and Soille, 1991] or segmentation based on the maximally stable extremal region (MSER) [Donoser et al., 2006]. Shape based segmentation additionally constraints the segmentation to be close to some prior given outline of the segmentation.

Edge based methods are originally based on previously extracted edges that are refined and traced to obtain a segmentation. Instead of tediously processing extracted contours, gradient information can be used to model the spatial relation ship or edge information. Modern segmentation approaches typically use edge information and contours. We will therefore give a more detailed overview on active contours in Section 1.2.1. Often the problem is formulated as some form of energy that is either minimized or maximized. This allows to model the desired properties of the segmentation in advance. We give a short overview on different energy minimization methods for image segmentation in Section 1.2.2, and deal with the topics in more detail when needed.

1.2.1 Active Contours

In the following we introduce previous work on active contours. The main idea is to model segmentation in terms of contours that are deformed according to various forces. These forces are either data (image) based, or model some form of higher level information and regularization.

1.2.1.1 Snakes

In [Kass et al., 1988], one of the first active contour models was introduced. The so called snakes are splines that are exposed to different forces that deform the snake in an iterative process. Using the spline, the contour is parametrized as $C(s) = \boldsymbol{x}(s) \in \Omega,\ s \in [0, 1]$. The snake model can be stated as the following variational problem:

$$\min_C \{E_{snake}\} = \min_C \{E_{int}(C(s)) + E_{img}(C(s)) + E_{con}(C(s))\}. \tag{1.2}$$

The energy consists of three separate forces. First, the internal energy of the snake is given by E_{int} and maintains the smoothness as well as the tension of the contour and is given as

$$E_{int} = \int_0^1 \frac{\alpha(s)}{2} \left| \frac{\partial C(s)}{\partial s} \right|^2 ds + \int_0^1 \frac{\beta(s)}{2} \left| \frac{\partial^2 C(s)}{\partial^2 s} \right|^2 ds. \tag{1.3}$$

This energy penalizes both first-order and second-order derivatives. The free parametrization functions $\alpha(s)$ and $\beta(s)$ are typically set to constant values. Inherently there is a shrinking bias caused by this internal forces. The image term E_{img} ensures that the boundary is drawn towards significant image features:

$$E_{img} = -\lambda \int_0^1 |\nabla (G_\sigma * I)(C(s))|^2 ds. \tag{1.4}$$

The ∇ operator computes the gradient of the image I convolved with a Gaussian kernel G_σ. This ensures that the contour aligns with edges in the image. Finally, the third term E_{con} is used to model constraints imposed by the user.

The energy defined in (1.2) is generally non-convex. As a result it is very difficult to find the globally optimum. [Kass et al., 1988] suggested to solve the snake model by an implicit finite differences method. While this method is very fast, it highly depends on a good initialization. Therefore, the user first has to manually specifies an initial boundary, hopefully close to the desired segmentation. For some special cases, it can be shown that a globally optimal solution of the snake model can be found [Schoenemann and Cremers,

2007].

Another problem of the approach by [Kass et al., 1988], is that the parametrization of the curve does not allow for automatic topology changes. Therefore, the algorithm has to analyze the topology during all iterations for self-intersections. This is an computationally expensive step.

Despite its drawbacks, the snake model has received much attention (especially in medical image segmentation [Jayadevappa et al., 2011; McInerney and Terzopoulos, 1996; Xu et al., 2000]). There exist several extensions, like an additional balloon force [Cohen, 1991] that prevents the typical shrinking bias. There also exist early extensions to 3D [Cohen et al., 1992; Cohen and Cohen, 1993; Terzopoulos et al., 1988] by the same authors.

1.2.1.2 Geodesic Active Contours

The geodesic active contours (GAC) model was introduced in [Caselles et al., 1997a,b] and [Kichenassamy et al., 1995, 1996]. The main idea, is to use an additional weighting function to guide the contour. While in 2D one is speaking of active contours, the 3D equivalent is referred to as minimal weighted surfaces.

Using the line integral, the GAC model can be formulated as the following optimization problem.

$$\min_C \left\{ \oint_C g(C) dC \right\} = \min_C \left\{ \int_0^{|C|_\mathcal{E}} g(|\nabla I(C(s))|) dl \right\}, \tag{1.5}$$

with the Euclidean length of the contour C defined as $|C|_\mathcal{E} = \oint \left| \frac{\partial C(s)}{\partial s} \right| ds = \oint dl$, and dl the Euclidean element of length. Here $g : \Omega \to \mathbb{R}^+$ is a weighting function, that is used to incorporate edge information. It should be low for strong edges, and high in flat areas. This way, the contour is pulled towards the edges during the minimization process. As suggested by [Caselles et al., 1997a] the edge detector function for an image I could be of the form

$$g(I) = \frac{1}{1 + \delta |\nabla G_\sigma * I|^2} + \epsilon. \tag{1.6}$$

The GAC problem is equivalent to finding a geodesic curve in a Riemannian space, where the length of the contour is given as

$$|C|_\mathcal{R} = \int_0^{|C|_\mathcal{E}} \sqrt{\mathcal{T}^T D(C(s)) \mathcal{T}}\, dl. \tag{1.7}$$

In the case of $D = \mathrm{diag}(g(|\nabla I|))$, the contour length in the Riemannian space equals the energy of the GAC model. Here $\mathcal{T}$ denote the unit tangent vector of the contour C.

There exists a very close relationship of the GAC model to the snake model [Aubert and Blanc-Féraud, 1998, 1999; Caselles et al., 1997a]. If we assume constant α and $\beta = 0$ in (1.3), and additionally add the weighting function g to (1.4), we arrive at the following variant of the snake model

$$E_{snake} = \alpha \int_0^1 \left| \frac{\partial C(s)}{\partial s} \right|^2 ds - \lambda \int_0^1 g|\nabla (G_\sigma * I)(C(s))|^2 ds. \tag{1.8}$$

[Caselles et al., 1997a] already showed the equivalence of the snake model in (1.8) and the GAC model in (1.5).

The simplest optimization approach is to apply gradient descent on the Euler-Lagrange equation of the GAC model. Another option, the level set method [Osher and Sethian, 1988; Sethian, 1999], was already suggested in the original paper [Caselles et al., 1997a]. The level set method represents a contour, as the level set of a higher dimensional function. Instead of evolving the contour directly, the higher dimensional function is updated. As a result topological changes of the contour are possible.

In [Appleton and Talbot, 2005, 2006], a globally optimal solution to the GAC model was introduced. The work of Appleton and Talbot also reveals the close connection to the continuous maximum flow problem. We will discuss this relationships and algorithms in more detail in Section 3.1.

1.2.2 Energy minimization methods

The labeling or segmentation problem can be written as an energy minimization problem. In fact, all algorithms presented in this thesis are based on energy minimization. Energy minimization methods typically try to minimize a composite criterion consisting of a data fidelity term and a regularization term that models prior knowledge. The expression of a composite energy term has a long history, dating from Maximum A Posteriori Bayesian models, which were formulated in the late 1980s as MRF models, and solved with iterative solvers, and more recently by Graph Cuts.

The energy formulation can either be set in a continuous or a discrete setting. The continuous setting provides an intuitive approach to model the real world that lies behind an image. Popular representatives of continuous segmentation formulations are the Mumford Shah model (Section 2.2.3), the continuous max flow problem (Section 3.1) and the continuous Potts model (Section 3.2). Chapter 2 and Chapter 3 will tackle these methods in detail.

Discrete methods are quite popular in computer vision, as images on a computer are typically sampled on a regular grid, and have a limited number of radiometric values. Thus the input data for computer vision algorithms is already given in a spatially as well as radiometric discretized domain. This perfectly fits the concept of graphs and Markov random fields (MRFs) or conditional random fields (CRFs). MRFs are a class of statistical models mainly developed in physics. Recently they have become quite popular in computer vision [Li, 2009], as they provide a general framework to model low level as well as high level computer vision problems. This includes labeling problems based on graph cuts or Total Variation (TV) based energy minimization.

Comparing discrete to continuous methods cannot be done in general, although both methods are used to solve very similar problems. In practice discrete methods are often preferred, as there is a huge choice of algorithms available, they are easy to apply. But there are reasons that make us use continuous approaches throughout this thesis, the most prominent being:

- There is no inherent bias towards the grid (discretization artifacts).
- Continuous methods are easy to parallelize.
- They reduce memory consumption compared to discrete methods.

We will give more details later, when all algorithms have been introduced.

Graph cuts are the discrete counterpart to the continuous max flow approaches used in this thesis. We therefore review the graph cut approach to image segmentation in the next Section.

1.2.2.1 Graph Cuts

Ignited by the work of [Veksler, 1999] and later [Boykov and Kolmogorov, 2003, 2004] graph cuts have become very popular in computer vision (e.g. 'GrabCut' proposed by [Rother et al., 2004]). Graph cuts are also referred to as the discrete max-flow/min-cut problem. [Ford and Fulkerson, 1956] already showed that the problem of finding a minimum cut in a weighted graph, is equivalent to finding the maximal flow through a graph. Max-flow/min-cut optimization already has a long history in computer vision as it can be applied to a wide range of optimization problems [Greig et al., 1989]. A more recent study in the general context of Markov random fields can be found in [Szeliski et al., 2008]. We will see in Section 3.1 that the discrete max flow problem is very closely related to the continuous approaches used in this thesis.

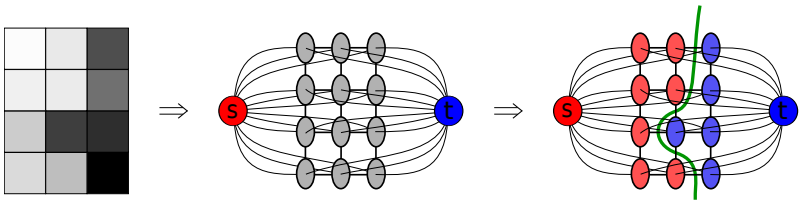


Figure 1.2: Illustration of the graph cut problem for a 2D image. On the left hand side the gray value image is illustrated. A graph is constructed by connecting all pixels in a 4-neighborhood. Additionally each pixel is connected to the source s and sink t terminals. The minimum cut on the right hand side, defines the final segmentation.

A graph $\mathcal{G}$ is a pair $(\mathcal{V}, \mathcal{E})$, that consists of a vertex set $\mathcal{V}$ and an edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. In image processing the vertices usually correspond to discrete pixel locations and two special terminal vertices, the source s and the sink t. See Fig. 1.2 for an illustration of graph construction and the graph cut problem. The edge set $\mathcal{E}$ consists of different types of edges. First, the spatial edges $e_b = (r, q) \,|\, r, q \in \mathcal{V} \backslash \{s, t\}$ that define the pixel neighborhood (e.g. 4-connected, 8-connected). Second, there are edges connecting every pixel with the source $e_s = (s, r)$ and with the sink $e_t = (r, t)$. In case of a weighted graph, all edges have some assigned costs $C(e) \geq 0$. In case of e_b these will correspond to image gradient information (edges), and for e_s and e_t the costs are used to model the foreground and background affinity.

A cut partitions the set of vertices $\mathcal{V}$ into two disjoint regions, $\mathcal{V}_s \cap \mathcal{V}_t = 0$ and $\mathcal{V} = \mathcal{V}_s \bigcup \mathcal{V}_t$, assigned to the source s and the sink t. We can define a cut $\mathcal{E}_c$ as the cost of all edges $e_c \in \mathcal{E}_c$ whose end points belong to two different regions. The cut has an assigned energy as the sum of all corresponding costs $C(e_c)$. Therefore the min-cut problem can be written as

$$\min_{\mathcal{E}_c \subset \mathcal{E}} \left\{ \sum_{e_c \in \mathcal{E}_c} C(e_c) \right\}. \tag{1.9}$$

To formulate the maximum flow problem, we can reinterpret the edge costs $C(e)$ as capacities. The goal is now to push as much flow through the graph (from the source s to the sink t) as the capacities allow. A flow can then be defined as a mapping $F : \mathcal{E} \rightarrow \mathbb{R}$ that fulfills the following constraints:

1. Conservation of the flow: For each vertex $q \in \mathcal{V} \backslash \{s, t\}$, the flow into the vertex

equals the flow leaving the vertex

$$\sum_{e:(r,q)\in\mathcal{E}} F(e) = \sum_{e:(q,r)\in\mathcal{E}} F(e). \tag{1.10}$$

2. Capacity constraint: For all edges $e \in \mathcal{E}$, the flow has to be less or equal to the corresponding capacity

$$F(e) \leq C(e). \tag{1.11}$$

[Ford and Fulkerson, 1956] showed that the min-cut and max-flow problems are equivalent to each other, and suggested to use an augmenting path algorithm. Other well known algorithms include the algorithm of [Edmonds and Karp, 1972], that improves the sort order of the Ford-Fulkerson algorithm, and the push-relabel algorithm [Goldberg and Tarjan, 1988]. All these algorithms rely on solving the max-flow problem. Recently, graph cuts for image segmentation have also been implemented on parallel hardware [Dixit et al., 2005; Vineet and Narayanan, 2008]. The discrete graph cut problem can also be written as an L_1 norm minimization problem as shown by [Bhusnurmath and Taylor, 2008], and it is thus possible to solve graph cuts with off-the-shelf LP solvers. We will see in Section 3.1.1 that it is also possible to solve the discrete min-cut/max-flow problem using continuous convex optimization methods.

1.2.2.2 Extensions to multi label segmentation

While multi label segmentation could be treated as a completely independent problem, it can also be reduced to solving multiple graph cut problems. This is typically performed using so called move-based algorithms. Move-based algorithms apply a number of changes to the labeling such that they always decrease the corresponding energy. Finding the optimal move is not trivial, as the multi label segmentation problem is usually NP-hard. Therefore a reasonable set of possible moves has to be found, to find a good approximate solution of the original problem. The most popular representatives are $\alpha\beta$-swap and α-expansion [Boykov et al., 2001].

The $\alpha\beta$-swap works by selecting two labels $\alpha\beta \in K$. All pixels in the region $\Omega_\alpha \bigcup \Omega_\beta$ are then allowed to change their value to $\{\alpha, \beta\}$. No other labels are affected by this move. Additionally, the move is only accepted if it decreases the overall energy. The algorithm performs $\alpha\beta$-swaps iteratively for all labels until no more changes occur.

The α-expansion move is different as it considers only one label α against the others.

Each pixel in Ω has the choice to either keep the current label, or switch to the label α. Thus the current label can only expand. In practice the α-expansion move seems to be the faster and more robust choice. However, the types of energies that expansion moves can optimize is more restricted than swap moves.

Other approaches have been proposed, as e.g. roof duality as shown in [Rother et al., 2007]. There also exist several other extensions to graph cuts and multi label segmentation, that we mention when needed (e.g. label costs Section 3.2.3). An excellent recent overview on progress in graph cut segmentation is given in [Delong, 2011].

1.3 Contributions of the thesis

Global relabeling for binary image segmentation: In Section 3.3.2, we discuss a global relabeling algorithm for binary image segmentation that we first introduced in [Unger et al., 2011]. Continuous optimization methods have become popular to deal with non-smooth convex optimization problems. They are inherently parallel and therefore well suited for GPU implementations. Most continuous optimization approaches have in common that they are very fast in the beginning, but tend to get slow as the solution gets close to the optimum. We therefore propose to apply global relabeling steps to speed up the convergence close to the optimum. We apply this algorithm to both the continuous max flow energy as well as to the discrete graph cut formulation. An evaluation comparing convergence rates reveals significant speedups over the fast primal dual algorithm of [Chambolle and Pock, 2010]. This is especially true for the discrete max flow, resulting in a fast parallel graph cut solver.

Interactive image segmentation: Section 4.1 covers the contributions to interactive image segmentation. It is based on the work in [Unger et al., 2008a,b] as well as parts of [Santner et al., 2009; Werlberger et al., 2011b]. For interactive image segmentation, it is important to have an efficient way to apply user constraints. We show that different kinds of constraints can be incorporated into our segmentation models. Constraints can be either hard to enforce pixel labels, or soft, giving only hints to which label the pixel belongs. In this context we also briefly discuss features suited for image segmentation. User constraints can have different forms e.g. scribbles or bounding boxes. We show how this forms of interaction can be integrated into our segmentation framework.

Interactive image segmentation requires immediate feedback for an optimal workflow. As a consequence the speed of the segmentation algorithm is very important, and requires

to match model, optimization method and implementation. With hardware getting more and more parallel, we have a strong focus on GPU implementations that are discussed in Section 3.3.1. Continuous convex optimization algorithms are inherently parallel and are therefore discussed in Chapter 2. An important choice is a good segmentation model that not only delivers good segmentation results, but can also be efficiently solved. Chapter 3 deals with variational segmentation models and how they can be solved efficiently. By treating all this parts in a coordinated way, we are able to interactively segment images with minimal response times.

Tracking as spatio-temporal segmentation: Tracking is usually interpreted as finding an object in single consecutive frames. Regularization is done by enforcing temporal smoothness of appearance, shape and motion. In Section 4.2, we propose a tracker, by interpreting the task of tracking as segmentation of a volume in 3D. Inherently temporal and spatial regularization is unified in a single regularization term. Segmentation is performed by a variational approach using anisotropic weighted Total Variation (TV) regularization, and thus closely related to the standard maximum flow approach. The proposed convex energy is solved by a fast primal-dual algorithm already discussed in Section 2.3.3. As demonstrated in our experiments, our tracking approach is able to handle large variations in shape and size, as well as partial and complete occlusions. While the algorithm works reasonably well by interactively segmenting the first frame, it allows interaction at all times. Thus it is possible to further refine the results, or manually recover the algorithm. The work of this Section was already introduced in [Unger et al., 2009], with a similar approach adapted in [Roberts et al., 2011].

Depth image segmentation: In Section 5.1, we present a novel approach for segmenting buildings in depth images. Depth ($2.5D$) images of the earth are usually referred to as digital surface model (DSM). They are obtained using either Lidar technology, or stereo / multi-view reconstruction from aerial imagery. In this Section we propose a multi label approach to automatically segment building footprints into coherent regions. The segmentation is based on the Potts model with label costs, we introduce in Section 3.2.3. Each region is modeled by the affine parameters of a plane. The resulting algorithm iteratively optimizes for the segmentation as well as the affine parameters.

A segmentation into meaningful regions gives rise to further processing such as semantics and scene understanding. We demonstrate that the algorithm delivers good results preserving small details such as chimneys. Additionally, we demonstrate that the proposed

approach can also be used for some stereo problems. The limiting factor is only the affine model for the regions.

Joint Motion Estimation and Motion Segmentation: In Section 5.2, we propose a unified variational formulation for joint motion estimation and segmentation with explicit occlusion handling, that we first presented in [Unger et al., 2012]. This is done by a multi-label representation of the flow field, where each label corresponds to a parametric representation of the motion, similar to the depth image segmentation approach. We again use a convex formulation of the multi-label Potts model with label costs and show that occlusion constraints can be integrated into our formulation via convex constraints. This is done for the asymmetric map-uniqueness criterion as well as the backmatch criterion. Explicit occlusion handling eliminates errors otherwise created by the regularization. As occlusions can occur only at segmentation boundaries, a large number of objects may be required. By using a fast primal-dual algorithm we are able to handle hundreds of motion segments.

Results are shown on several classical motion segmentation and optical flow examples. Different extensions to the basic model with affine parameters are presented. This includes the extension to quadratic parameters, as well as regions without any parametrization. Although the algorithm is rather slow, it opens exciting opportunities by simultaneously delivering optical flow, motion segmentation and occlusions.

Chapter 2

Continuous convex optimization

2.1 Inverse problems

Throughout this thesis we will deal with inverse problems. Thus, let us first start with some considerations on these kind of problems. We therefore first define inverse problems, and then discuss approaches to find a solution.

2.1.1 Problem definition

One of the earliest descriptions of an inverse problem can be found in the famous 'cave allegory':

> Behold! human beings living in a underground den, which has a mouth open towards the light and reaching all along the den; here they have been from their childhood, and have their legs and necks chained so that they cannot move, and can only see before them, being prevented by the chains from turning round their heads. Above and behind them a fire is blazing at a distance, and between the fire and the prisoners there is a raised way; and you will see, if you look, a low wall built along the way, like the screen which marionette players have in front of them, over which they show the puppets. [...] And do you see, I said, men passing along the wall carrying all sorts of vessels, and statues and figures of animals made of wood and stone and various materials, which appear over the wall? [...] Like ourselves, I replied; and they see only their own shadows, or the shadows of one another, which the fire throws on the opposite wall of the cave? [Plato, 2008]

Plato uses this allegory to make philosophical considerations if and how human beings can discern truth. But the setting also gives a good example of an inverse problem. According to [Keller, 1976], two problems are inverses to each other if the formulation involves all or part of the solution of the other. He called the better understood problem the direct problem, and the other one the inverse problem. While the direct problem often describes a physical process and is easy to model, the inverse problem is often more complicated. We speak of inverse problems whenever we have to deduce the state of a system (the solution x) from one or multiple observations (the data y). This process is also called inference. Inverse problems have already been well studied in various context, see for example [Bertero and Boccacci, 1998; Chambolle, 2000; Fitzpatrick, 1991; Idier, 2008; Kaipio and Somersalo, 2005; Kirsch, 2011; Knapik et al., 2011; Stuart, 2010] as a starting point.

From the definition of inverse problems, it is obvious that we have to deal with inverse problems all day long: When trying to catch a ball we have to infer its future location from our visual observations. When trying to understand an image we have to infer the single components it is composed of. When talking with your wife/partner, you have to infer what she/he really tries to convey.

We already see that inverse problems are by no means trivial to solve. Inverse problems are typically *ill-posed*. Let us first recall the definition of this notion. According to [Hadamard, 1902] there are three conditions that have to be fulfilled for a problem to be *well-posed*.

Existence: A solution x to the problem exists.

Uniqueness: The solution x is unique.

Stability: The dependence of the solution x on the data y is continuous. (An infinitesimal small change δy of the data induces only an infinitesimal error δx on the solution.)

All other problems are considered *ill-posed*. While for a lot of ideal problems, the condition of *Existence* is fulfilled, a solution might cease to exist if noise is added to the observation. For a lot of problems the *Uniqueness* condition does not hold (e.g. think of quantization). Also the condition of *Stability* is often violated. This poses additional difficulties, as small deviations are amplified and can lead to arbitrary large errors. Note that although even if problems can be modeled exactly, this does not mean that they are stable.

2.1.2 Minimization of a composite criterion

When dealing with ill-posed inverse problems, one has to rely on prior knowledge on the structure of the problem and the corresponding direct problem. One way to do so is by means of an composite criterion. In the seminal work of [Tikhonov, 1963] the usage of an optimization problem was proposed as follows:

$$\min_u \{\mathcal{R}(u) + \lambda \mathcal{D}(u, f)\} \; . \tag{2.1}$$

The observation is denoted with f and the solution with u. The right hand term $\mathcal{D}(u, f)$ denotes a data fidelity term that models the relationship between solution u and observation f. To convert an ill-posed problem into a well-posed problem we have to make use of a-priori information and make certain assumptions on the solution. In other words, we have to impose some regularity on the solution. In (2.1), this is done by the regularization term $\mathcal{R}(u)$. [Tikhonov, 1963; Tikhonov and Arsenin, 1977] used a quadratic regularization of the solution that we will discuss among others in Section 2.2.1. The parameter λ is used to find a tradeoff between regularization and fidelity to the observation.

The challenge now is to model this composite criterion such that the optimization result is very close to the true solution. Therefore statistical properties on the space of solutions as well as the physical process described by the corresponding direct problem are important. Finding the right model for the actual problem is not a trivial task. George E. P. Box once said: 'Essentially, all models are wrong, but some are useful.'

All problems in this thesis will be based on the formulation in (2.1). But modeling an appropriate composite criterion does not only involve to find a model that has a solution close to the desired solution, but we also have to be able to find this solution in a feasible time. One important criterion is convexity of the energy posed by the composite criterion. If convex, a globally optimal solution exists and can be found (e.g. by means of first order derivatives). But if the energy is non-convex we additionally have to decide between local and global optima. This process is much more complex and in general can only be solved by comparing all possible solutions. Of course the number of solutions can be infinite making non-convex optimization problems unpractical.

We will give an introduction to convex optimization methods in Section 2.3. The rest of this thesis will deal with modeling appropriate convex optimization problems for segmentation problems.

2.1.3 A Bayesian approach

While the above description of a composite criterion minimization seems obvious, another natural way to approach inverse problems is in a probabilistic setting. Indeed, the Bayesian approach to inverse problems has been well studied [Bayes and Price, 1763; Markov, 1971; Ramsey, 1931] and more recently [Chambolle et al., 2009; Idier, 2008; Kaipio and Somersalo, 2005; Pock, 2008]. We therefore treat u as a random variable that has some underlying probability distribution. The optimization problem can then be formulated as finding the hypothesis u, that maximizes the probability based on the observation f:

$$\max_u \{p(u|f)\} \; . \tag{2.2}$$

We also speak of the maximum a posteriori (MAP) estimation, and $p(u|f)$ the posterior probability.

The well known Bayes theorem states that

$$p(u|f) = \frac{p(f|u)p(u)}{p(f)} \; , \tag{2.3}$$

where $p(u)$, the prior probability of u, corresponds to the regularization (2.1) or the a priori known information. The likelihood that the observation f can be explained by the hypothesis u is denoted as $p(f|u)$, and is also called the conditional probability. It directly corresponds to the data fidelity term in (2.1). For the optimization in u, we can neglect the constant $p(f)$.

As shown in [Chambolle et al., 2009], computing the MAP solution of u is not always the best thing to do, as the solution with the highest probability might be very rare. A better solution could rely on the expectation of u (e.g. [Protter et al., 2010]). Unfortunately a direct optimization is not feasible and one has to rely on statistical optimization methods like Markov Chain Monte Carlo (MCMC).

2.2 Classical variational problems

In this section we start with some classical variational problems in computer vision. As an introduction we first focus on image denoising. The important contribution of [Rudin et al., 1992] on image denoising paved the way for variational methods in image processing. We will then show the close relationship to shape denoising, that already gives a motivation for variational image segmentation.

2.2.1 Image denoising

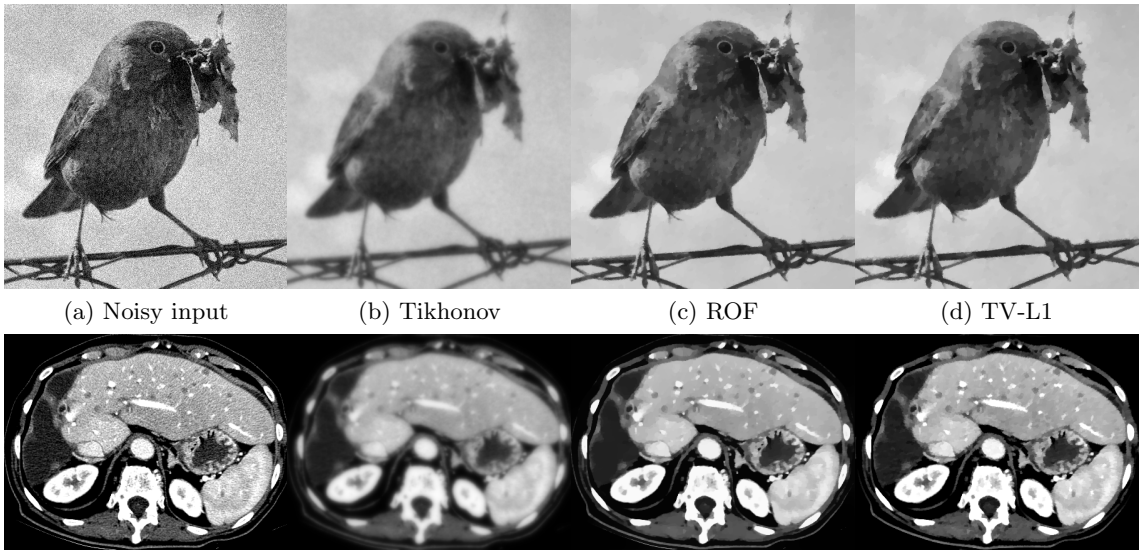
The most basic example is the one of image denoising or restoration. We assume that the true image $u^* : \Omega \to \mathbb{R}$ was degraded by additive noise $n : \Omega \to \mathbb{R}$. Note that Ω represents the image domain throughout this thesis. Thus, we observe an image $f : \Omega \to \mathbb{R}$ as $f = u^* + n$. In the following, we will introduce three well known variational image denoising problems.

2.2.1.1 Tikhonov model

Using the ideas of [Tikhonov, 1963] we can come up with the following optimization problem:

$$\min_u \left\{ \frac{1}{2} \int_\Omega |\nabla u|^2 + \frac{\lambda}{2} \int_\Omega (u - f)^2 dx \right\}. \tag{2.4}$$

The right term ensures that the solution $u : \Omega \to \mathbb{R}$ is close to the input f, and is therefore called the data fidelity term. The left term does regularization in the Tikhonov sense by quadratically penalizing the image gradient ∇u. Finally, a free parameter λ is used to find a tradeoff between regularization and data term. In Figure 2.1b, two examples of Tikhonov denoising using (2.4) are depicted. One for a natural image with artificial Gaussian noise,



(a) Noisy input (b) Tikhonov (c) ROF (d) TV-L1

Figure 2.1: Comparison of different variational denoising models. The top row shows an example with artificial Gaussian noise. On the bottom an CT image with natural noise was used as an input.

and one CT image with natural noise. While the Tikhonov denoising approach efficiently removes the noise, it also results in a strong blur and loss of small details.

We now reconsider the Bayesian approach discussed in Section 2.1.3. If we assume that the observation f was degraded by additive Gaussian noise of variance σ^2, we can formulate the likelihood as

$$p(f|u) = \prod_{\boldsymbol{x}\in\Omega} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(u(\boldsymbol{x})-f(\boldsymbol{x}))^2}{2\sigma^2}}. \tag{2.5}$$

For the image prior, we again assume quadratic smoothness of the image gradients. If we again assume a Gaussian distribution with variance μ^2 of the image gradients $|\nabla u|$, we obtain a prior probability as

$$p(u) = \prod_{\boldsymbol{x}\in\Omega} \frac{1}{\sqrt{2\pi}\mu} e^{-\frac{|\nabla u(\boldsymbol{x})|^2}{2\mu^2}}. \tag{2.6}$$

As a result of the Bayes theorem (2.3), we can write the posterior as

$$p(u|f) = p(f|u)p(u) = \prod_{\boldsymbol{x}\in\Omega} \frac{1}{\sqrt{4\pi}\sigma\mu} e^{-\frac{((u(\boldsymbol{x})-f(\boldsymbol{x}))^2}{2\sigma^2)^2}-\frac{|\nabla u(\boldsymbol{x})|^2}{2\mu^2}}. \tag{2.7}$$

Interestingly this equation corresponds to the Gibbs function [Gibbs, 1902] that is used in thermodynamics.

Using the MAP optimization approach (2.2), we arrive at the following optimization problem.

$$\begin{aligned}
\max_u\{p(u|f)\} &= \max_u \left\{ \prod_{\boldsymbol{x}\in\Omega} e^{-\frac{(u(\boldsymbol{x})-f(\boldsymbol{x}))^2}{2\sigma^2}-\frac{|\nabla u(\boldsymbol{x})|^2}{2\mu^2}} \right\} \\
&= \max_u \left\{ e^{-\int_\Omega \left(\frac{(u(\boldsymbol{x})-f(\boldsymbol{x}))^2}{2\sigma^2}+\frac{|\nabla u(\boldsymbol{x})|^2}{2\mu^2}\right) d\boldsymbol{x}} \right\} \\
&= \min_u \left\{ \int_\Omega \left(\frac{(u(\boldsymbol{x})-f(\boldsymbol{x}))^2}{2\sigma^2} + \frac{|\nabla u(\boldsymbol{x})|^2}{2\mu^2} \right) d\boldsymbol{x} \right\}.
\end{aligned} \tag{2.8}$$

One immediately sees that this formulation is equivalent to the Tikhonov denoising model in (2.4).

2.2.1.2 ROF model

In the seminal work of [Rudin et al., 1992], the quadratic regularization of (2.4) was replaced with an $L1$ norm:

$$\min_u \left\{ \int_\Omega |\nabla u| + \frac{\lambda}{2} \int_\Omega (u - f)^2 dx \right\}. \tag{2.9}$$

This model is often referred to as the ROF model according to Rudin, Osher and Fatemi. While the original formulation of [Rudin et al., 1992] proposes a constrained optimization problem, the strictly convex formulation in (2.9) was proposed in [Chambolle and Lions, 1997]. The Total Variation (TV) term represents the sum over all absolute image gradients:

$$\mathrm{TV}(u) = \int_\Omega |\nabla u| = \int_\Omega \sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2} \ , \tag{2.10}$$

with the 2D image gradient $\nabla u = \left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\right)^T$. We assume here for the moment, that u can be differentiated at least once. As depicted in Figure 2.1c, the ROF denoising model according to (2.9) provides a much better result, as it not only removes the noise but also preserves the edges. This fact is also well known in robust statistics [Huber, 1981]. For an illustration of this edge preserving effect see Figure 2.2. Here three different functions are depicted with different step sizes. Due to the quadratic regularization in the Tikhonov model (2.4), the energy with a lot of small steps is significantly lower than one big step. As a result the Tikhonov model will favor smooth transitions resulting in a blur of the

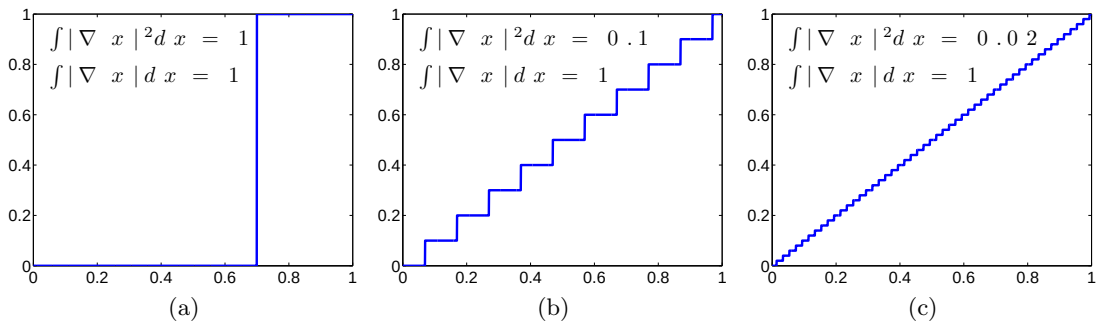


Figure 2.2: Comparison of quadratic and TV regularization. TV regularization always costs the total amount of all jumps, that is here 1 in all three examples. For the quadratic regularization smaller jumps mean lower costs.

image. On the other hand, TV counts the absolute gradients, making the resulting energy for the three examples of Figure 2.2 always 1. As the regularization does not favor any of these three functions only the data term is relevant for the final result, and edges or smooth transitions of the image are preserved.

The denoising effect of TV regularization relies on the increased costs of small dips that are caused by the noise. But this is of course also true for small structures, as TV regularization prefers flat over rippled functions. As can be seen in Figure 2.1c, the denoising results contains a lot of small flat regions. This so called stair-casing effect is a well known side-effect of TV regularization.

The TV regularization term has also been extended to vector norms for color image processing. See for example the publications of [Aujol and Kang, 2006; Blomgren and Chan, 1998; Bresson and Chan, 2008; Chan et al., 2001; Goldluecke et al., 2012] as a starting point.

2.2.1.3 TV-L1 model

The L1 norm can not only be used for the regularization, but also in the data term. When the L2 data term in the ROF model (2.9) is replaced with an L1 norm, we arrive at the so called TV-L1 model [Alliney, 1992, 1997; Aujol et al., 2006; Chan and Esedoglu, 2005; Nikolova, 2002, 2004]:

$$\min_u \left\{ \int_\Omega |\nabla u| + \lambda \int_\Omega |u - f| dx \right\}. \tag{2.11}$$

While this problem is still convex, it is not strictly convex any more. This means that the minimizers of (2.11) are not unique any more. In Figure 2.1d, the denoising result using the TV-L1 model is depicted. When comparing to the results of the ROF model, the differences are marginal. But [Nikolova, 2002] shows, that for certain types of noise (e.g. impulse noise), the TV-L1 model outperforms the ROF model.

The artificial example in Figure 2.3 reveals another interesting property of the TV-L1 model. First, the ROF model suffers from slight blurring and contrast loss. Additionally, it removes structures of the same size depending on its local contrast. This effect is much smaller when using the TV-L1 model in Figure 2.3c, which seems to be almost contrast invariant. Objects of a certain size are removed, regardless of their local contrast. When using an L2 norm in the data term, small deviations of u and f cost significantly less than large deviations. As a result, structures with low contrast are removed first. While this observation is still true when using an L1 norm, the dependence is not quadratic anymore, but linear. The effect of contrast dependence almost vanishes.

(a) Input (b) ROF (c) TV-L1

Figure 2.3: Demonstration of the contrast invariant denoising properties of the TV-L1 model (2.11), compared to the ROF model (2.9).

2.2.2 Shape denoising

In the previous section, we already noted that TV (2.10) counts the sum of all absolute gradients. This relationship between the level-sets of an image with bounded variation and TV was first expressed by the so called co-area formula [Fleming and Rishel, 1960]:

$$\int_{\Omega} |\nabla u| = \int_{-\infty}^{\infty} Per\left(\{x : u(x) > \gamma\}\right) d\gamma \ , \tag{2.12}$$

where $Per(\Sigma)$ denotes the perimeter of a set Σ. According to the co-area formula, TV can be decomposed into a sum of the length of all level-sets of u. As a result TV as in the ROF model (2.9), not only minimizes the jumps in the image, but also the length of the level-sets. This effect can be seen in Figure 2.3b, where the edges of the squares are noticeably rounded. A thorough study on this relationship can also be found in [Chambolle, 2005].

We now assume that we deal with a binary input image $f : \Omega \to \{0, 1\}$. This binary image can also be expressed by its characteristic function for a region Σ

$$f(x) = 1_{\Sigma}(x) \ , \tag{2.13}$$

with the characteristic function given as

$$\mathbf{1}_{\Sigma}(\boldsymbol{x}) = \begin{cases} 1 & \text{if} \quad \boldsymbol{x} \in \Sigma \\ 0 & \text{else} \qquad . \end{cases} \tag{2.14}$$

Recalling the coarea formula (2.12), it is obvious that the TV of the characteristic function

of a set, is exactly the perimeter of the set [Chan et al., 2006; Evans and Gariepy, 1992; Giusti, 1984].

$$Per(\Sigma) = \int_{\Omega} 1_{\Sigma}(x)dx. \tag{2.15}$$

We further assume that the binary input image f is the corrupted correspondence to another binary function $u : \Omega \to \{0, 1\}$. We can easily add the constraint of a binary u to the ROF model (2.9)

$$\min_{u\in\{0,1\}} \left\{ \int_{\Omega} |\nabla u| + \frac{\lambda}{2} \int_{\Omega} (u - f)^2 dx \right\}, \tag{2.16}$$

as well as the TV-L1 model (2.11)

$$\min_{u\in\{0,1\}} \left\{ \int_{\Omega} |\nabla u| + \lambda \int_{\Omega} |u - f| dx \right\}. \tag{2.17}$$

As a consequence, both models now minimize the contour length of the solution u. [Chan et al., 2006] used the ROF formulation (2.16) to solve the following equivalent shape denoising problem:

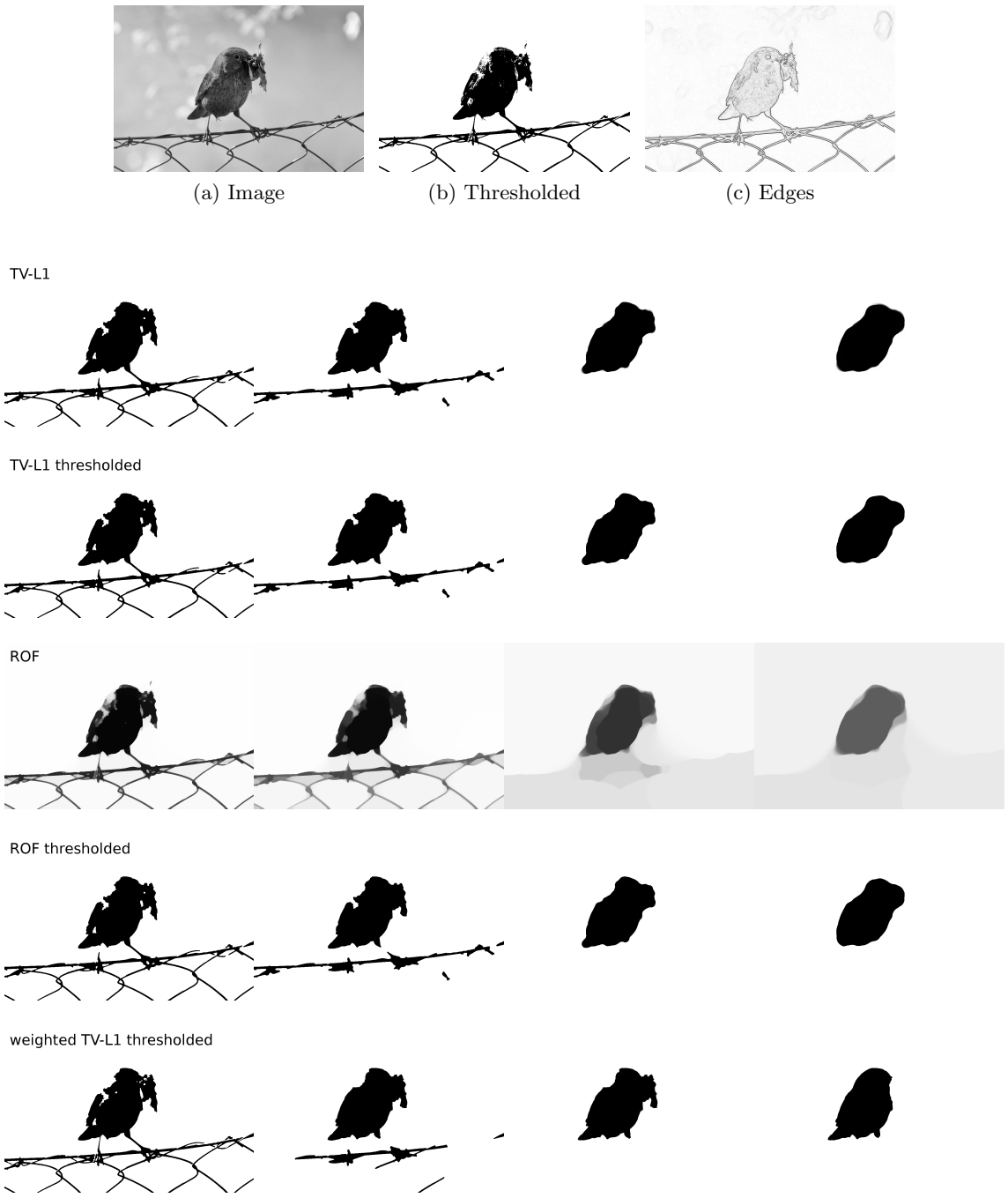
$$\min_{\Sigma} \{ Per(\Sigma) + \lambda |\Sigma \ominus S| \}, \tag{2.18}$$

for a given set S, and $\ominus$ the symmetric difference operator. As shown in [Chan and Esedoglu, 2005], the data term of the TV-L1 model can also be rewritten using the layer cake formula:

$$\int_{\Omega} |u(x) - f(x)| dx = \int_{-\infty}^{\infty} |\{x : u(x) > \gamma\} \ominus \{x : f(x) > \gamma\}| \, d\gamma. \tag{2.19}$$

Unfortunately, both (2.16) and (2.17) are non-convex due to the binary functions. In [Chan and Esedoglu, 2005; Chan et al., 2006], the authors proposed to rewrite (2.17) in terms of the original convex TV-L1 formulation (2.11). They proved that if $u(x)$ is a minimizer of the TV-L1 energy (2.11), for almost every $\gamma \in [0, 1]$, the binary function $1_{\Sigma}(x)$ with $\Sigma = \{x : u(x) > \gamma\}$ is a minimizer of the shape denoising problem (2.17). This allows to transform the non-convex shape denoising problem into a convex problem by making use of its level set formulation. Any threshold gives a globally optimal solution to the shape denoising problem. But as the energy is not strictly convex, this solution is not unique.

In Figure 2.4, we applied the shape denoising model (2.17) to a real world example. Figure 2.4a shows the input image I and Figure 2.4b a thresholded version that is used as input to the shape denoising problem $f = 1_{\{x:I(x)>0.5\}}$. The first row shows the result



(a) Image (b) Thresholded (c) Edges

Figure 2.4: Shape denoising of (b). The first two rows show the results of the TV-L1 model and the thresholded result. Row 3 and 4 show the output when using the ROF model and the thresholded result. While the ROF model causes a strong contrast reduction, the TV-L1 model delivers a nearly binary result. When thresholding the output at 0.5, both models deliver almost the same shape denoising result. In the bottom row the TV in the regularization term was weighted with the edge weighting function in (c). As a result the boundary is drawn towards the edges.

u of the TV-L1 model (2.11) with decreasing λ from left to right, and the second row the thresholded version $1_{\{x:u(x)>0.5\}}$. We used a threshold of $\gamma = 0.5$, but any threshold $\gamma \in [0, 1]$ is possible. First note that with decreasing λ, the regularization term becomes more important and the contour length is efficiently minimized. As one can see that the output of the TV-L1 model is almost binary, with only small blurred regions. This means that all globally optimal solutions to the shape denoising problem are very close to each other.

In the third row of Figure 2.4 we used the ROF model (2.9) with the same input as in the previous example. One can immediately notice the typical effects of blurring and contrast loss. But when we look at the thresholded version, the result is the same as for the TV-L1 model. This time the threshold $\gamma = 0.5$ is important. We will discuss this relationship in detail in Section 3.3.3.

The examples in Figure 2.4 show that TV can be used to minimize the contour length in a shape denoising framework. This already suggests that TV is well suited for image segmentation tasks. But it is obvious that minimizing the contour length also results in blobs where the boundaries might not align well with object boundaries in the image. A circle has the lowest possible ratio between length and area, and will therefore be favored in a shape denoising framework. Again note the rounded edges in Figure 2.3c. A way to overcome this problem is to add a weighting function to the regularization. [Bresson et al., 2007] suggested to use the weighted TV $\int_\Omega g|\nabla u|$ to minimize the GAC energy [Caselles et al., 1997a]. The shape denoising problem in (2.17) thus becomes

$$\min_{u\in\{0,1\}} \left\{ \int_\Omega g|\nabla u| + \lambda \int_\Omega |u - f| dx \right\}. \tag{2.20}$$

We can weight the boundary costs by setting g to an edge weighting function that has low values for strong edges and high values in flat regions. As a result the region boundaries are drawn towards the edges in the image. We depicted the results of (2.20) in the bottom row of Figure 2.4. One can note that the shape denoising results are much better aligned to object boundaries of the input image. All segmentation models presented in Chapter 3 and the rest of this thesis will rely on the weighted TV.

2.2.3 Mumford-Shah model

The Mumford-Shah segmentation model is a well known optimization problem introduced in [Mumford and Shah, 1989]. It determines a piecewise smooth approximation of the

input image f. The Mumford-Shah functional is given by the following minimization problem

$$\min_{\Lambda,u} \left\{ \nu \mathcal{H}^{c-1}(\Lambda) + \frac{\alpha}{2} \int_{\Omega} (u-f)^2 dx + \frac{\beta}{2} \int_{\Omega \setminus \Lambda} |\nabla u|^2 dx \right\}. \tag{2.21}$$

Here Λ denotes the edge set, and $\mathcal{H}^{c-1}(\Lambda)$ the $c-1$ dimensional Hausdorff measure. The first term therefore minimizes the length of the edges. With the second term, the quadratic distance of the solution u to the input image f is minimized. This data fidelity term is equivalent to the one used in the ROF model (2.9). The last term of (2.21) applies smoothing inside the region $\Omega \setminus \Lambda$. The free positive constants ν, α and β are used to weight the single terms.

A well known variation of the model in (2.21), is the piecewise constant Mumford-Shah functional defined by

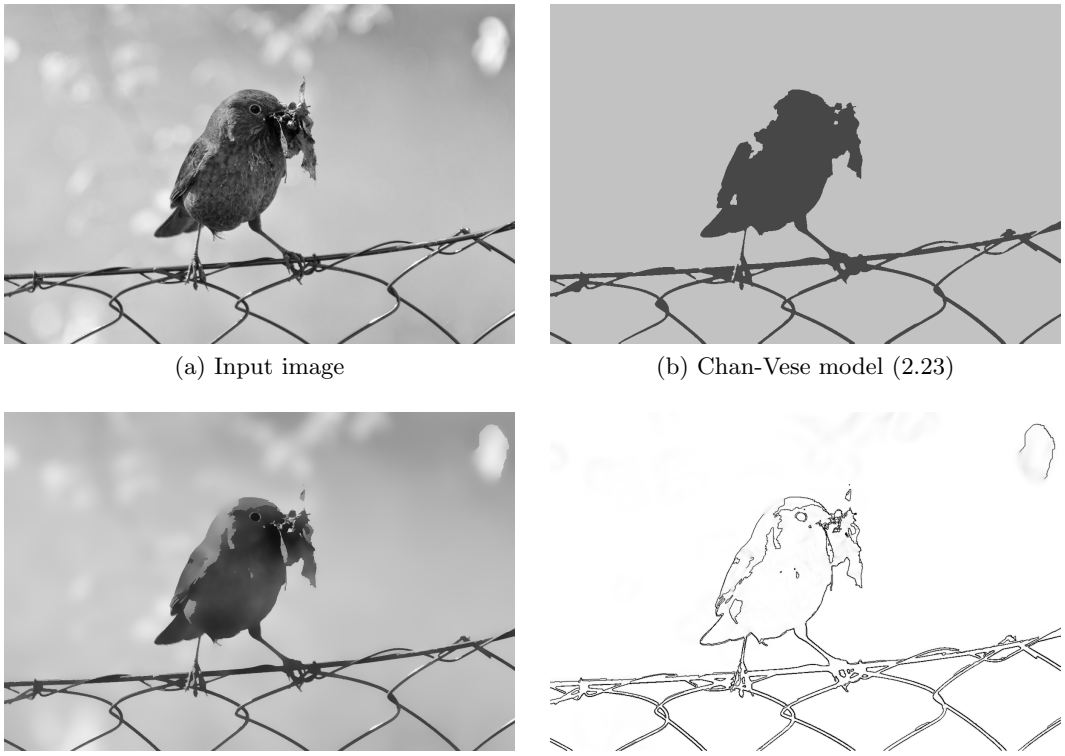
$$\min_{u} \left\{ \nu \mathcal{H}^{c-1}(\Gamma_u) + \frac{\alpha}{2} \int_{\Omega} (u-f)^2 dx \right\}, \tag{2.22}$$

where Γ_u now denotes the jump set of u. This problem is closely related to the Potts/Ising model [Potts, 1952]. A good convex relaxation to the piecewise constant Mumford-Shah model was e.g. presented in [Pock et al., 2009a]. We will discuss these approaches in Section 3.2.

An exact minimization of the original energy (2.21) is not a trivial task. The Mumford-Shah model is often solved using the level set framework [Osher and Sethian, 1988; Tsai et al., 2000; Vese and Chan, 2002]. But also graph based methods have become popular [Grady and Alvino, 2009] to improve on local minima. Unfortunately, graph based methods do not allow for open boundaries. In [Pock et al., 2009b], a convex relaxation approach is presented that uses functional lifting. In contrast to most other approaches, the convex approximation ensures a globally optimal solution, and thus independence of initialization. Additionally open boundaries pose no problem to the algorithm.

There is a wide range of applications of the Mumford-Shah model, that includes segmentation, image denoising, matching and motion estimation. See for example [Berkels, 2010] as an overview on the wide applicability of the Mumford-Shah functional. Figure 2.5(c,d), depicts an example of the piecewise smooth Mumford-Shah model (2.21), for which we used the implementation of [Pock et al., 2009b].

An important contribution is the active contours without edges (ACWE) model [Chan and Vese, 1999, 2001] that we also refer to as the Chan-Vese model. The Chan-Vese model minimizes the active contour energy (see Section 1.2.1), by reducing the Mumford-Shah



(a) Input image (b) Chan-Vese model (2.23)

(c) Piecewise smooth Mumford-Shah (2.21) (d) Edge set Λ (2.21)

Figure 2.5: Demonstration of different approximations to the Mumford-Shah energy (2.21). (b) depicts the two-phase Chan-Vese model. (c,d) were obtained using the convex relaxation of [Pock et al., 2009b].

model to the following two-phase segmentation problem;

$$\min_{\Sigma, c_1, c_2} \left\{ \nu \mathrm{Per}(\Sigma) + \int_{\Sigma} (c_1 - f)^2 dx + \int_{\Omega \setminus \Sigma} (c_2 - f)^2 dx \right\}. \tag{2.23}$$

The minimization problem minimizes not only for the region Σ, but also c_1 and c_2, making the optimization problem non-convex. We will see in Section 3.1.3, that the model is also closely related to the maximum flow problem. When fixing c_1 and c_2, the globally optimal solution of the Chan-Vese model can be computed. The Chan-Vese model does not allow for open boundaries. While this is not in the spirit of the original Mumford-Shah functional (2.21), closed boundaries are an requirement for image segmentation as defined in (1.1).

In Figure 2.5(b), the result of the Chan-Vese model is depicted. We used the implementation of [Getreuer, 2010]. When comparing to the shape denoising results in Figure 2.4,

the close connection becomes obvious.

The Chan-Vese model (2.23) was extended to a multiphase segmentation model in [Vese and Chan, 2002]. This so called Vese-Chan model uses two level set functions to model four different regions. Using the 4-color theorem, an arbitrary number of regions can be modeled. Unfortunately this requires the region to be spatially connected. The Potts model presented in Section 3.2 does not suffer from this topological constraints.

The Mumford-Shah model was also extended to vector-valued images [Chan, 2000]. Other approximations of the Mumford-Shah functional include the model by [Ambrosio and Tortorelli, 1990]. The proposed phase field functions result in a diffuse boundary representation. It also allows for open boundaries, making it a good approximation of the original formulation (2.21).

2.3 Convex optimization

In the following we introduce some basics of convex optimization. We will discuss the primal dual algorithm used throughout this thesis, and show how it can be applied to basic problems discussed in the previous section.

2.3.1 Preliminaries

This section gives a short mathematical background and introduces notations that we are going to use later on.

2.3.1.1 Discretization

We first consider the discretization of the continuous formulations. An image is usually given on a two dimensional regular Cartesian grid of the size $M \times N$:

$$\{(ih, jh) : 1 \leq i \leq M, 1 \leq j \leq N\}, \tag{2.24}$$

with the indices of the discrete locations given by (i, j) and the pixel size (or spacing) h. For this thesis, the pixels are considered isotropic and we therefore set $h = 1$. We make all considerations assuming the dimensionality of the image to be $d = 2$. The extension to higher order functions (e.g. 3D images) is straightforward.

We now define a finite dimensional vector space $X = \mathbb{R}^{MN}$ with a scalar product

$$\langle v, w \rangle_X = \sum_{i,j} v_{i,j} w_{i,j}, \quad v, w \in X \ . \tag{2.25}$$

Furthermore, we define a vector space $Y = \mathbb{R}^{MN} \times \mathbb{R}^{MN} = \mathbb{R}^{2MN}$, with the gradient operator as a linear mapping $\nabla : X \to Y$ using finite differences and Neumann boundary conditions:

$$(\nabla v)_{i,j} = \left((\delta_x^+ v)_{i,j}, (\delta_y^+ v)_{i,j} \right)^T \ , \tag{2.26}$$

where

$$\begin{aligned} (\delta_x^+ v)_{i,j} &= \begin{cases} v_{i+1,j} - v_{i,j} & \text{if} \quad i < M \\ 0 & \text{if} \quad i = M \end{cases} \ , \\ (\delta_y^+ v)_{i,j} &= \begin{cases} v_{i,j+1} - v_{i,j} & \text{if} \quad j < N \\ 0 & \text{if} \quad j = N \end{cases} \ . \end{aligned} \tag{2.27}$$

Given two vectors $\boldsymbol{p} = (p^x, p^y)^T$ and $\boldsymbol{q} = (q^x, q^y)^T \in Y$ we define the scalar product as follows:

$$\langle \boldsymbol{p}, \boldsymbol{q} \rangle_Y = \sum_{i,j} p_{i,j}^x q_{i,j}^x + p_{i,j}^y q_{i,j}^y \ . \tag{2.28}$$

Additionally we have to define a divergence operator $\mathrm{div} : Y \to X$ by choosing it to be adjoint to the gradient operator in (2.26), and thus fulfilling the equality

$$\langle \nabla u, \boldsymbol{p} \rangle_Y = - \langle u, \mathrm{div} \boldsymbol{p} \rangle_X \ . \tag{2.29}$$

Therefore, the discrete divergence operator is given as:

$$(\mathrm{div} \boldsymbol{p})_{i,j} = (\delta_x^- p^x)_{i,j} + (\delta_y^- p^y)_{i,j} \ , \tag{2.30}$$

with

$$\begin{aligned} (\delta_x^- p^x)_{i,j} &= \begin{cases} 0 & \text{if} \quad i = 0 \\ p_{i,j}^x - p_{i-1,j}^x & \text{if} \quad 0 < i < M \\ -p_{i-1,j}^x & \text{if} \quad i = M \end{cases} \ , \\ (\delta_y^- p^y)_{i,j} &= \begin{cases} 0 & \text{if} \quad j = 0 \\ p_{i,j}^y - p_{i,j-1}^y & \text{if} \quad 0 < j < N \\ -p_{i,j-1}^y & \text{if} \quad j = N \end{cases} \ . \end{aligned} \tag{2.31}$$

2.3.1.2 Vector norms

A norm on a vector space X is a function $||\cdot|| : X \to \mathbb{R}^+$, such that the following three conditions hold:

1. $||v|| = 0$ if $v = 0$ and $||v|| > 0$ if $v \neq 0$.
2. $||kv|| = |k| \, ||v||$.
3. $||v + w|| \leq ||v|| + ||w||$.

for all vectors $v, w \in X$ and the scalar k. In case the condition $||v|| > 0$ if $v \neq 0$ does not hold, one speaks of a seminorm.

We will make heavy use of the L^p-norm, that is defined as

$$||v||_p = \left(\sum_i |v_i|^p \right)^{1/p} , \tag{2.32}$$

with $p \geq 1$. There are three important cases of the L^p-norm:

- $p = 1$: The so called L^1-norm is often also referred to as the Manhattan or taxicab norm. It is computed as the sum over all absolute values

$$||v||_1 = \sum_i |v_i| . \tag{2.33}$$

- $p = 2$: The L^2-norm is also referred to as the Euclidean norm and computes the length of a vector. It can be written as

$$||v||_2 = \sqrt{|v_1|^2 + \ldots + |v_n|^2} = \sqrt{< v, v >_X} . \tag{2.34}$$

- $p = \infty$: The final case is the L^∞-norm also referred to as the Maximum norm, as it can be computed as

$$||v||_\infty = \max \left(|v_1|, \ldots, |v_n| \right) . \tag{2.35}$$

In case vector values are themself vector valued as e.g. in $\boldsymbol{p} = (p^x, p^y)^T, \in Y$, we have to take special care with the notation. We typically assume that the point wise or inner norm denotes the length of the vector computed as the L^2-norm (or absolute value $|\cdot|$ for scalars). To specify a different inner norm, we use the notation $||\cdot||_{q,p}$, that denotes an

inner L^q-norm, and an outer L^p-norm. Two examples are given in the following:

$$\begin{aligned} ||\boldsymbol{p}||_{2,1} &= \sum_i \sqrt{|p_i^x|^2 + |p_i^y|^2}\ , \\ ||\boldsymbol{p}||_{1,1} &= \sum_i \left(|p_i^x| + |p_i^y|\right)\ . \end{aligned} \tag{2.36}$$

As the $||\cdot||_{2,1}$-norm will be the most common used norm, we simplify our notation by writing $||\cdot||_p$ instead of $||\cdot||_{2,p}$ and simply $||\cdot||$ instead of $||\cdot||_{2,1}$.

2.3.1.3 Convexity

Convexity has an important role in optimization. Convex optimization deals with the minimization of convex functions over convex sets. As there are no local, but only global minima, convex optimization is much easier than the general case. We therefore shortly recap the basic principles of convexity.

Convex sets: Given a set C in the vector space V, we call C a convex set if the following condition holds

$$t\boldsymbol{x} + (1-t)\boldsymbol{y} \in C,\ \ \forall \boldsymbol{x}, \boldsymbol{y} \in C \text{ and } t \in [0,1]. \tag{2.37}$$

This means, that each point that lies on the line connecting two points $\boldsymbol{x}, \boldsymbol{y} \in C$, is again in the set C.

There are several operations that preserve the convexity of a set, with the most important as following: Given a convex set C, and an affine function f, then $\{f(\boldsymbol{x}) : \boldsymbol{x} \in C\}$ is again convex. The sum of two convex sets C, D is again convex $C + D = \{\boldsymbol{x} + \boldsymbol{y} : \boldsymbol{x} \in C,\ \boldsymbol{y} \in D\}$. The same holds for the intersection $C \cap D = \{\boldsymbol{x} : \boldsymbol{x} \in C \text{ and } \boldsymbol{x} \in D\}$.

Convex functions: A function $f : X \to \mathbb{R}$ with its domain a convex set C, is called convex if

$$f(t\boldsymbol{x} + (1-t)\boldsymbol{y}) \leq tf(\boldsymbol{x}) + (1-t)f(\boldsymbol{y})\ , \tag{2.38}$$

with $\boldsymbol{x}, \boldsymbol{y} \in X$ and $t \in [0,1]$. If further the condition

$$f(t\boldsymbol{x} + (1-t)\boldsymbol{y}) < tf(\boldsymbol{x}) + (1-t)f(\boldsymbol{y}) \tag{2.39}$$

holds, the function f is called strictly convex. While a convex function can have multiple global minima, a strictly convex function has only one global minimum. The existence of only global minima allows to base minimization on the local variations or derivatives, as one cannot get stuck in local minima. As we will see in Section 2.3.2, this allows for a wide range of algorithms to solve convex optimization problems.

(2.38) can be geometrically interpreted, that all points of the line connecting $(f(\boldsymbol{x}), \boldsymbol{x})$ and $(f(\boldsymbol{y}), \boldsymbol{y})$ lie above the function, the so called epigraph. The epigraph epi of f is the set containing everything on or above the function f.

$$\text{epi} f = \{(\boldsymbol{x}, \mu) : \boldsymbol{x} \in X, \mu \in \mathbb{R} \text{ and } \mu \geq f(\boldsymbol{x})\}. \tag{2.40}$$

In other words, f is convex if epi$f$ is a convex set.

The convexity of a function is preserved under several operations. If $f_1, \ldots, f_n$ are convex functions, then also the sum $h(\boldsymbol{x}) = \sum_i f_i(\boldsymbol{x})$, as well as the maximum $h(\boldsymbol{x}) = \max\{f_1(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})\}$ are convex. Further, any affine mapping $f(A\boldsymbol{x} + b)$ preserves convexity.

2.3.1.4 Duality

In the following we introduce basic principles of duality. We will make use of the Legendre-Fenchel (LF) transformation or convex conjugate [Rockafellar, 1970]. The LF transformation for a function $f(x)$ is given as

$$f^*(y) = \sup_{x \in \mathbb{R}} \{yx - f(x)\}. \tag{2.41}$$

We can also write $f^*(k) = (f(x))^*$ or simply $f^* = (f)^*$ for the LF transformation. The LF transformation of $f^*(k)$ is also called the double LF transformation (or biconjugate) of $f(x)$, and is given as

$$f^{**}(x) = \sup_{y \in \mathbb{R}} \{yx - f^*(y)\}. \tag{2.42}$$

For a good introduction to the LF transformation, we refer the interested reader to [Touchette, 2007] and summarize the following properties of the LF transformation:

- The LF transformation f^* is always a convex function.
- The double LF transformation f^{**} is the convex envelope of f. For a convex function f we get $f^{**} = f$.

- Points in f correspond to slopes in f^*, and slopes in f are transformed into points in f^*.
- Non-differential points in f result in an affine slope in f^*. Affine as well as non-convex segments in f correspond to a non-differentiable point in f^*.

We now derive the LF transformation for TV (2.10), that is a convex function with one non-differential point. We have $f(\nabla u) = |\nabla u|$ and its LF transformation with respect to ∇u given as

$$f^*(\boldsymbol{p}) = \sup_{\nabla u \in \mathbb{R}^d} \{\boldsymbol{p} \cdot \nabla u - |\nabla u|\} = I_{\{||\boldsymbol{p}||_\infty \leq 1\}}(\boldsymbol{p}). \tag{2.43}$$

With the indicator function $I_\Sigma(x)$ for the set Σ given as

$$I_\Sigma(x) = \begin{cases} 0 & \text{if} \quad x \in \Sigma\,, \\ \infty & \text{else}\,. \end{cases} \tag{2.44}$$

We call $\boldsymbol{p} : \Omega \to \mathbb{R}^d$ the dual variable. Note that the ∇ operator performs a mapping from $\mathbb{R} \to \mathbb{R}^d$ for d-dimensional images. As a consequence the dual variable $\boldsymbol{p}$ will be of dimensionality d. See Section 2.3.1.1 for more details.

Now the LF transformation of $f^*(\boldsymbol{p})$ can be computed as

$$f^{**}(\nabla u) = \sup_{\boldsymbol{p}} \left\{\boldsymbol{p} \cdot \nabla u - I_{\{||\boldsymbol{p}||_\infty \leq 1\}}(\boldsymbol{p})\right\} = \sup_{||\boldsymbol{p}||_\infty \leq 1} \{\boldsymbol{p} \cdot \nabla u\}\,. \tag{2.45}$$

The above equation gives two equivalent notations for the constraint $||\boldsymbol{p}(x)||_\infty \leq 1$, $\forall x \in \Omega$.

2.3.2 An overview on minimization algorithms

In the following we give a brief overview on convex optimization methods. As we will use a single primal dual algorithm throughout this thesis (the algorithm is detailed in the next section), we only introduce the most relevant ideas. For an in depth introduction to convex optimization there are excellent books such as [Boyd and Vandenberghe, 2004].

We first look at the following unconstrained minimization problem:

$$\min_u E(u), \tag{2.46}$$

with $E : \mathbb{R}^n \to \mathbb{R}$ a convex energy with $E \in \mathcal{C}^1$ and $u \in \mathbb{R}^n$. As the class of energies E that allow to solve (2.46) directly is very small, one generally uses iterative approaches to

solve the problem.

First order methods are based on the first derivative of E. The gradient $\frac{d}{du}E(u)$ gives the direction in which to move in order to minimize E. A critical point is reached when the Euler-Lagrange equation is fulfilled:

$$\frac{d}{du}E(u) = 0. \tag{2.47}$$

Using gradient descend methods will converge to the globally optimal solution if the optimization problem is convex.

A gradient descend approach was already proposed for the ROF model [Rudin et al., 1992]. Unfortunately gradient descend methods tend to be very slow in flat regions (typically close to the optimum), and often do not move towards the global optimum directly. To tackle this problem over-relaxation is used by well known approaches such as Gauss-Seidel iterations or successive over relaxation (SOR). Another example for a gradient descend method is the interior point method of [Vogel and Oman, 1996]. There, a linearization of the Euler-Lagrange equation is used. Still other problems of gradient descend methods as discontinuities in the energy E and additional constraints, still remain.

Interior point methods solve a problem with linear equality as well as inequality constraints by reduction to a sequence of linear equality constrained problems. According to [Boyd and Vandenberghe, 2004], they can be used to solve problems of the form

$$\begin{aligned} &\min_u E(u), \\ \text{s.t. } &f_i(u) \leq 0,\ i = 1, \ldots, M \\ &Au = b, \end{aligned} \tag{2.48}$$

with $f_i : \mathbb{R}^n \to \mathbb{R}$ convex functions in $\mathcal{C}^2$, and $A \in \mathbb{R}^{p \times n}$ and $\text{rank}A = p < n$.

Another well known concept is the one of proximal point methods. While interior point methods work inside the convex feasible domain, the proximal point method is a simplex method that works through the border of the convex feasible domain. They can be applied if the energy can be split up into a sum of convex functions. First the gradient based updates are done to obtain $\tilde{u}$. Then a proximity operator projects the update $\tilde{u}$ back to the convex set by solving

$$\arg\min_u \left\{ E_i(u) + \frac{||u - \tilde{u}||^2}{2} \right\}. \tag{2.49}$$

A famous proximal point algorithm is the Douglas-Rachford splitting [Douglas and Rachford, 1956]. Other well known proximal algorithms are the iterative shrinkage thresholding algorithm (ISTA) algorithm [Nesterov, 1983], extended in [Nesterov, 2004] and adapted by [Beck and Teboulle, 2009] as fast iterative shrinkage thresholding algorithm (FISTA).

Using duality as introduced in Section 2.3.1.4, discontinuities can be easily handled. Examples are the fixed-point algorithm in [Chambolle, 2004] or the projected gradient descent [Chambolle, 2005]. Other examples are [Kunisch and Hintermüller, 2004] and [Zhu and Chan, 2008]. In the next section, we will discuss the primal dual algorithm of [Chambolle and Pock, 2010], that we use throughout the rest of this thesis.

2.3.3 A general primal dual algorithm

In [Chambolle and Pock, 2010], a fast first order primal-dual algorithm for non-smooth convex saddle point problems was presented. For the general algorithm a convergence rate $O(1/N)$ is proved. The general saddle point problem that can be solved using the primal dual algorithm is of the form

$$\min_{\alpha \in A} \max_{\beta \in B} \left\{ \langle D\alpha, \beta \rangle + \Phi(\alpha) - \Psi^*(\beta) \right\}. \tag{2.50}$$

Where A and B are finite-dimensional real vector spaces, the linear operator $D : A \to B$, and two proper, convex, lower-semicontinuous functions $\Phi : A \to \mathbb{R} \cup \{\infty\}$ and $\Psi^* : B \to \mathbb{R} \cup \{\infty\}$. Here Ψ^* is the convex conjugate of a convex lower-semicontinuous function Ψ. The general algorithm to solve the saddle point problem in (2.50) is given in Algorithm 1. The iterative algorithm consists of a gradient ascent step on the dual variable β with an additional resolvent operator. For the primal variable α a gradient descent step with an additional resolvent operator is performed. Additionally an extra-gradient step is applied to the primal variable α. The time-steps τ and σ are chosen such that $\tau\sigma L^2 < 1$ with the Lipschitz constant $L^2 = ||D||^2$. We chose $\theta = 1$ for the rest of this paper. The newly introduced variable $\bar{\alpha} \in A$ represents the leading point for the primal variable α. The subgradients of the convex functions Φ and Ψ^* are denoted as $\partial\Phi$ and $\partial\Psi^*$. The resolvent

Algorithm 1 General primal dual algorithm to solve (2.50)

$\tau > 0,\ \sigma > 0,\ \theta \in [0,1]$
$\alpha^0 \in A,\ \beta^0 \in B$ and $\bar{\alpha}^0 = \alpha^0$
for $j = 1$ to J **do**
$\beta^j = (I + \sigma\partial\Psi^*)^{-1}\left(\beta^{j-1} + \sigma D\bar{\alpha}^{j-1}\right)$
$\alpha^j = (I + \tau\partial\Phi)^{-1}\left(\alpha^{j-1} - \tau D^*\beta^j\right)$
$\bar{\alpha}^j = \alpha^j + \theta\left(\alpha^j - \alpha^{j-1}\right)$
end for

Algorithm 2 Accelerated primal dual algorithm to solve (2.50) if Φ or Ψ^* are uniformly convex.

$\tau_0 > 0,\ \sigma_0 > 0,\ \tau_0\sigma_0 L^2 < 1,\ \gamma > 0,\ \theta \in [0,1]$
$\alpha^0 \in A,\ \beta^0 \in B$ and $\bar{\alpha}^0 = \alpha^0$
for $j = 1$ to J **do**
$\beta^j = \left(I + \sigma^{j-1}\partial\Psi^*\right)^{-1}\left(\beta^{j-1} + \sigma^{j-1}D\bar{\alpha}^{j-1}\right)$
$\alpha^j = \left(I + \tau^{j-1}\partial\Phi\right)^{-1}\left(\alpha^{j-1} - \tau^{j-1}D^*\beta^j\right)$
$\theta^j = 1/\sqrt{1+2\gamma\tau^{j-1}}$, $\tau^j = \theta^j\tau^{j-1}$ and $\sigma^j = \sigma^{j-1}/\theta^j$
$\bar{\alpha}^j = \alpha^j + \theta^j\left(\alpha^j - \alpha^{j-1}\right)$
end for

operators can be calculated as

$$\begin{aligned}\beta = (I + \sigma\partial\Psi^*)^{-1}(\tilde{\beta}) &= \arg\min_{\beta}\left\{\frac{||\beta - \tilde{\beta}||^2}{2\sigma} + \Psi^*(\beta)\right\},\\ \alpha = (I + \tau\partial\Phi)^{-1}(\tilde{\alpha}) &= \arg\min_{\alpha}\left\{\frac{||\alpha - \tilde{\alpha}||^2}{2\tau} + \Phi(\alpha)\right\}.\end{aligned} \tag{2.51}$$

[Chambolle and Pock, 2010] also showed that Algorithm 1 can be further accelerated if either Φ or Ψ^* are uniformly convex, meaning that they have a Lipschitz continuous gradient. In this case, the they can prove a $O(1/N^2)$ convergence rate, using the modified algorithm in Algorithm 2. In contrast to Algorithm 1, additionally the time steps τ and σ, as well as θ are modified depending on the number of iterations. We will see in Section 2.3.4, that this algorithm can efficiently solve the ROF model (2.9). As demonstrated in [Chambolle and Pock, 2010], this results in a significantly faster convergence. They also showed that the algorithm can be further accelerated to linear convergence, if both Φ and Ψ^* are uniformly convex. As we are dealing with TV in all our algorithms, this case is not relevant in our context. A uniformly convex regularization term is e.g. the Huber norm [Huber, 1973].

Primal-dual formulations in continuous optimization have the advantage that the gap

between primal energy and dual energy provides a meaningful convergence measure. The saddle point problem in (2.50) has a corresponding primal problem

$$E_p = \min_{\alpha \in A} \left\{ \Psi(K\alpha) + \Phi(\alpha) \right\}, \tag{2.52}$$

as well as a pure dual problem

$$E_d = \max_{\beta \in B} \left\{ -\Psi^*(\beta) - \Phi^*(-K^*\beta) \right\}. \tag{2.53}$$

The partial primal-dual gap is defined as

$$\begin{aligned} G(\alpha', \beta') = &\max_{\beta' \in X_2} \left\{ \left\langle D\alpha, \beta' \right\rangle + \Phi(\alpha) - \Psi^*(\beta') \right\} \\ &- \min_{\alpha' \in X_1} \left\{ \left\langle D\alpha', \beta \right\rangle + \Phi(\alpha') - \Psi^*(\beta) \right\}. \end{aligned} \tag{2.54}$$

In the case of $E_p(\alpha) = E_d(\beta)$, the global optimum is reached, and (α, β) are a saddle point. Though the primal-dual gap provides an optimality measure, it is not guaranteed to decrease continuously.

In [Pock and Chambolle, 2011], a diagonal preconditioning for the above algorithms was introduced. The preconditioning makes the computation of the time steps unnecessary. They showed, that especially for problems with irregular structure, the preconditioned algorithm outperforms [Chambolle and Pock, 2010].

2.3.4 Practical application

This section describes the optimization of the ROF and TV-L1 model using the principles introduced in the previous sections, and the algorithms from Section 2.3.3.

2.3.4.1 ROF model

We now make use of the Legendre-Fenchel transformation to convert the ROF model (2.9) into a suitable primal-dual problem (2.50). For a convex function f we know that $f^{**} = f$. Therefore the ROF model can be rewritten using the double LF transformation of TV in

(2.45) as the following saddle-point problem

$$\min_u \max_{||\boldsymbol{p}||_\infty \leq 1} \left\{ \int_\Omega \boldsymbol{p} \cdot \nabla u + \frac{\lambda}{2} \int_\Omega (u - f)^2 dx \right\}. \tag{2.55}$$

The corresponding discretized version reads

$$\min_u \max_{\boldsymbol{p}} \left\{ \langle \boldsymbol{p}, \nabla u \rangle + \frac{\lambda}{2} ||u - f||^2 - I_{\{||\boldsymbol{p}||_\infty \leq 1\}}(\boldsymbol{p}) \right\}. \tag{2.56}$$

Let us recall that $f \in \mathbb{R}^{MN}$, $u \in \mathbb{R}^{MN}$ and $\boldsymbol{p} \in \mathbb{R}^{dMN}$ with $d = 2$ for 2D images. Consequently the linear operator ∇ is of size $MN \times 2MN$.

Analyzing (2.56), we see that the primal variable $\alpha = u$ and the dual variable $\beta = \boldsymbol{p}$. The linear operator $D = \nabla$ results in a Lipschitz constant $L^2 = 8$. Finally we have the non-linear functions $\Phi(u) = \frac{\lambda}{2}||u - f||^2$ and $\Psi^*(\boldsymbol{p}) = I_{\{||\boldsymbol{p}||_\infty \leq 1\}}(\boldsymbol{p})$.

The resolvent operator for the primal update can be computed according to (2.49) as

$$u = (I + \tau \partial \Phi)^{-1}(\tilde{u}) = \arg\min_u \left\{ \frac{||u - \tilde{u}||^2}{2\tau} + \frac{\lambda}{2} ||u - f||^2 \right\}. \tag{2.57}$$

To solve this minimization problem we look at the corresponding Euler-Lagrange equation:

$$\frac{1}{\tau}(u - \tilde{u}) + \lambda(u - f) = 0. \tag{2.58}$$

Thus, the solution to the resolvent operator is given as the following point-wise update

$$u_{i,j} = \frac{\tilde{u}_{i,j} + \tau\lambda f_{i,j}}{1 + \tau\lambda}. \tag{2.59}$$

We finally have to compute the resolvent operator for the dual update as

$$\boldsymbol{p} = (I + \sigma \partial \Psi^*)^{-1}(\tilde{\boldsymbol{p}}) = \arg\min_{\boldsymbol{p}} \left\{ \frac{||\boldsymbol{p} - \tilde{\boldsymbol{p}}||^2}{2\sigma} + I_{\{||\boldsymbol{p}||_\infty \leq 1\}}(\boldsymbol{p}) \right\}. \tag{2.60}$$

It is easy to see, that as $\Psi^*(\boldsymbol{p})$ is the indicator function of a convex set, the resolvent operator is given as simple point-wise Euclidean projections on the unit ball. We can write this as

$$\boldsymbol{p}_{i,j} = \frac{\tilde{\boldsymbol{p}}_{i,j}}{\max(1, |\tilde{\boldsymbol{p}}_{i,j}|)}. \tag{2.61}$$

It is now trivial to apply the algorithms from Section 2.3.3 to (2.56). We summarized the general primal dual algorithm for the ROF model in Algorithm 3. As the data term in

Algorithm 3 General primal dual algorithm to solve the ROF model (2.9)

$\tau = \sigma = \frac{1}{L} = \frac{1}{\sqrt{8}}$, $\theta = 1$
$u^0 = f$, $\boldsymbol{p}^0 = 0$ and $\bar{u}^0 = u^0$ // Initialization
for $j = 1$ to J **do**
  $\tilde{\boldsymbol{p}} = \boldsymbol{p}^{j-1} + \sigma \nabla \bar{u}^{j-1}$ // Update dual variable
  $\boldsymbol{p}^j = \frac{\tilde{\boldsymbol{p}}}{\max(1, |\tilde{\boldsymbol{p}}|)}$
  $u = (1 + \tau\lambda)^{-1}(u^{j-1} + \tau \mathrm{div} \boldsymbol{p}^j + \tau\lambda f)$ // Update primal variable
  $\bar{u}^j = 2u^j - u^{j-1}$ // Extra-gradient step
end for

Algorithm 4 Accelerated primal dual algorithm to solve the ROF model (2.9)

$\tau = \sigma = \frac{1}{L} = \frac{1}{\sqrt{8}}$
$u^0 = f$, $\boldsymbol{p}^0 = 0$ and $\bar{u}^0 = u^0$ // Initialization
for $j = 1$ to J **do**
  $\tilde{\boldsymbol{p}} = \boldsymbol{p}^{j-1} + \sigma \nabla \bar{u}^{j-1}$ // Update dual variable
  $\boldsymbol{p}^j = \frac{\tilde{\boldsymbol{p}}}{\max(1, |\tilde{\boldsymbol{p}}|)}$
  $u = (1 + \tau\lambda)^{-1}(u^{j-1} + \tau \mathrm{div} \boldsymbol{p}^j + \tau\lambda f)$ // Update primal variable
  $\theta^j = 1/\sqrt{1 + 0.7\tau^{j-1}}$
  $\tau^j = \theta^j \tau^{j-1}$ and $\sigma^j = \sigma^{j-1}/\theta^j$ // Update time steps
  $\bar{u}^j = u^j + \theta^j \left(u^j - u^{j-1}\right)$ // Extra-gradient step
end for

the ROF model is uniformly convex, we can apply Algorithm 2 as a solver. The resulting accelerated algorithm is summarized in Algorithm 4.

2.3.4.2 TV-L1 model

Using the knowledge from the previous sections, we can transform the TV-L1 model (2.11) into the following discrete primal-dual saddle point problem

$$\min_u \max_{\boldsymbol{p}} \left\{ \langle \boldsymbol{p}, \nabla u \rangle + \lambda ||u - f|| - I_{\{||\boldsymbol{p}||_\infty \leq 1\}}(\boldsymbol{p}) \right\}. \tag{2.62}$$

This problem is very similar to the ROF formulation in (2.56), but now we have an L1 data term and thus $\Phi(u) = \lambda ||u - f||$.

The resolvent operator can again be computed as

$$u = (I + \tau \partial \Phi)^{-1}(\tilde{u}) = \arg\min_u \left\{ \frac{||u - \tilde{u}||^2}{2\tau} + \lambda ||u - f|| \right\}, \tag{2.63}$$

Algorithm 5 Algorithm to solve the TV-L1 model (2.11) using thresholding scheme.

$\tau = \sigma = \frac{1}{\sqrt{9}},\ \theta = 1$
$u^0 = f,\ \boldsymbol{p}^0 = 0$ and $\bar{u}^0 = u^0$ // Initialization
for $j = 1$ to J **do**
  $\tilde{\boldsymbol{p}} = \boldsymbol{p}^{j-1} + \sigma \nabla \bar{u}^{j-1}$ // Update dual variable
  $\boldsymbol{p}^j = \frac{\tilde{\boldsymbol{p}}}{\max(1, |\tilde{\boldsymbol{p}}|)}$
  $\tilde{u} = u^{j-1} + \tau \mathrm{div} \boldsymbol{p}^j f$ // Update primal variable
  $u^j = \begin{cases} \tilde{u} - \tau\lambda & \text{if} \quad \tilde{u} - f > \tau\lambda \\ \tilde{u} + \tau\lambda & \text{if} \quad \tilde{u} - f < -\tau\lambda \\ f & \text{if} \quad |\tilde{u} - f| \leq \tau\lambda \end{cases}$
  $\bar{u}^j = 2u^j - u^{j-1}$ // Extra-gradient step
end for

with the corresponding Euler-Lagrange equation:

$$\frac{1}{\tau}(u - \tilde{u}) + \lambda \frac{u - f}{|u - f|} = 0. \tag{2.64}$$

As a result, we arrive at the following soft thresholding schema:

$$u_{i,j} = \begin{cases} \tilde{u}_{i,j} - \tau\lambda & \text{if} \quad \tilde{u}_{i,j} - f_{i,j} > \tau\lambda \\ \tilde{u}_{i,j} + \tau\lambda & \text{if} \quad \tilde{u}_{i,j} - f_{i,j} < -\tau\lambda \\ f_{i,j} & \text{if} \quad |\tilde{u}_{i,j} - f_{i,j}| \leq \tau\lambda \end{cases}. \tag{2.65}$$

As the TV-L1 model is a non-smooth optimization problem, we can apply only the general primal-dual algorithm. We summarized the algorithm in Algorithm 5.

While the soft thresholding schema in (2.65) works very well, we could also provide an alternative optimization approach by dualizing the data term. The Legendre Fenchel transformation for the L1 norm was already discussed for TV. Thus we can rewrite the primal-dual TV-L1 model with dualized data term as

$$\min_u \max_{\boldsymbol{p},q} \left\{ \langle \boldsymbol{p}, \nabla u \rangle + \langle q, u - f \rangle - I_{\{||\boldsymbol{p}||_\infty \leq 1\}}(\boldsymbol{p}) - I_{\{||q||_\infty \leq \lambda\}}(q) \right\}. \tag{2.66}$$

Since we can rewrite $\langle \boldsymbol{p}, \nabla u \rangle + \langle q, u - f \rangle$ as $\left\langle \begin{pmatrix} \boldsymbol{p} \\ q \end{pmatrix}, \begin{pmatrix} \nabla \\ I \end{pmatrix} u \right\rangle - qf$, the reformulation is again trivial. We now have the primal variable $\alpha = u$ and the dual variable $\beta = \begin{pmatrix} \boldsymbol{p} & q \end{pmatrix}^T$. The linear operator $D = \begin{pmatrix} \nabla \\ I \end{pmatrix}$, with I the identity matrix of size $MN \times MN$. This

Algorithm 6 Algorithm to solve the TV-L1 model (2.11) with dualized data term.

$\tau = \sigma = \frac{1}{\sqrt{9}},\ \theta = 1$
$u^0 = f,\ \boldsymbol{p}^0 = 0,\ q^0 = 0$ and $\bar{u}^0 = u^0$ // Initialization
for $j = 1$ to J **do**
  $\tilde{\boldsymbol{p}} = \boldsymbol{p}^{j-1} + \sigma \nabla \bar{u}^{j-1}$ // Update dual variables
  $\boldsymbol{p}^j = \frac{\tilde{\boldsymbol{p}}}{\max(1, |\tilde{\boldsymbol{p}}|)}$
  $q^j = \left[q^{j-1} + \sigma(\bar{u}^{j-1} - f)\right]_{-\lambda}^{\lambda}$
  $u^j = u^{j-1} - \tau(-\mathrm{div}\boldsymbol{p}^j + q^j)$ // Update primal variable
  $\bar{u}^j = 2u^j - u^{j-1}$ // Extra-gradient step
end for

results in a Lipschitz constant $L^2 = 9$. We have no non-linear parts any more $\Phi(u) = 0$ and $\Psi^*(\boldsymbol{p}, q) = qf + I_{\{||\boldsymbol{p}||_\infty \leq 1\}}(\boldsymbol{p}) + I_{\{||q||_\infty \leq \lambda\}}(q)$. The resolvent operator can then be computed by solving the following constrained optimization problem

$$\arg\min_{\beta} \left\{ \frac{||(\ \boldsymbol{p}\quad q\ )^T - (\ \tilde{\boldsymbol{p}}\quad \tilde{q}\ )^T||^2}{2\sigma} + qf \right\}, \quad s.t.\ ||\boldsymbol{p}||_\infty \leq 1,\ ||q||_\infty \leq \lambda\ . \tag{2.67}$$

Solving the Euler-Lagrange equation $\frac{1}{\sigma}(q - \tilde{q}) + f = 0$ we get

$$q = \tilde{q} - \sigma f\ , \tag{2.68}$$

with the additional clamping of q to the interval $[-\lambda, \lambda]$. We denote this point wise truncation as $[\ \cdot\ ]_{-\lambda}^{\lambda}$. The resolvent operator for $\boldsymbol{p}$ is again a point-wise Euclidean projection on the unit ball as stated in (2.61).

The resulting algorithm can be found in Algorithm 6.

Chapter 3

Variational Image Segmentation

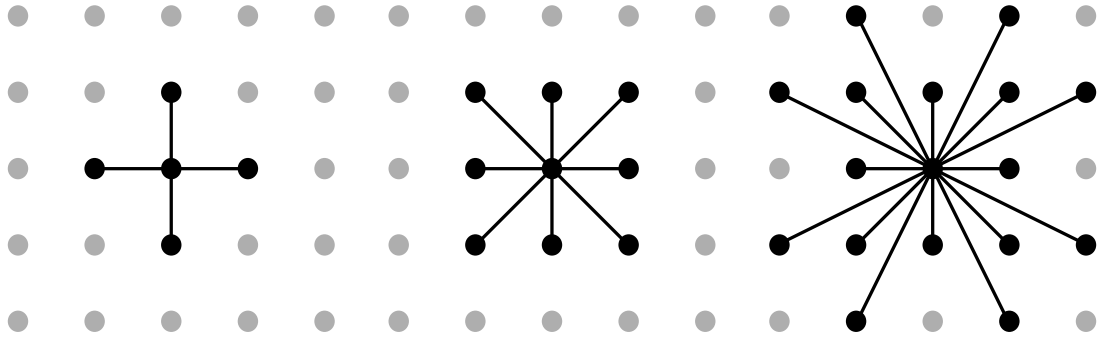
3.1 Binary image segmentation - Continuous max flow

Binary image segmentation is probably the most important segmentation task. A lot of problems can be formulated as a simple foreground/background segmentation. Although it is a special case of the more general multi label problem, treating binary image segmentation separately allows for a lot of improvements and optimizations. It is thus more efficient to solve for the specific rather than the general problem.

In this Section we focus on the continuous minimum cut / maximum flow problem. We show how the problem can be solved efficiently using the convex optimization methods introduced in the last chapter. Connections to the discrete minimum cut / maximum flow approach as well as to the Mumford Shah functional and TV-L1 shape denoising are discussed.

3.1.1 Recap of the discrete min cut / max flow

In Section 1.2.2.1, we already introduced the graph cut or discrete min cut / max flow problem. We now reformulate the problem in (1.9) using the characteristic function $u \in \mathbb{R}$. Additionally we construct the vectors $w_s, w_t \in \mathbb{R}$ using the costs $C(e)$ at the corresponding positions of the edges linked with source s or sink t. We refer to these terms as the unary terms. The same is done for the spatial edge costs by constructing the vector $w_b \in \mathbb{R}^K$ (the binary terms). Here K indicates the neighborhood in the graph. Different neighborhoods are depicted in Figure 3.1.



(a) 4-connected (b) 8-connected (c) 16-connected

Figure 3.1: Different connectivity for the graph of the discrete min cut / max flow model.

We can rewrite (1.9) as the following minimization problem

$$\begin{aligned} \min_u \left\{ \|W_b \nabla u\|_{1,1} + \langle 1-u, w_s \rangle + \langle u, w_t \rangle \right\}, \\ \text{s.t. } u \in \{0,1\}, \end{aligned} \tag{3.1}$$

with $W_b = \operatorname{diag}(w_b)$. To solve the above energy, we have to relax the variable $u \in [0,1]$ to vary continuously between 0 and 1. It is well known [Chambolle, 2005] that the resulting convex relaxation will provide the globally optimal solution for the original problem in (3.1). We can rewrite the terms $\langle 1-u, w_s \rangle + \langle u, w_t \rangle$ as $\langle u, w_t - w_s \rangle + \langle 1, w_s \rangle$. The last term is the sum over all sink costs w_s (note that $w_s \geq 0$). As this term is constant, we can neglect it during optimization. To further simplify the notation, we define the unary terms as $w_u = w_t - w_s$. Therefore we can rewrite the relaxed version of (3.1) as

$$\begin{aligned} \min_u \left\{ \|W_b \nabla u\|_{1,1} + \langle u, w_u \rangle \right\}, \\ \text{s.t. } u \in [0,1], \end{aligned} \tag{3.2}$$

We again use Legendre-Fenchel duality to obtain the convex conjugate of the L^1 norm in (3.2). We then arrive at the following primal-dual saddle point formulation of the graph cut energy

$$\begin{aligned} \min_u \max_{\boldsymbol{p}} \left\{ \langle \nabla u, \boldsymbol{p} \rangle + \langle u, w_u \rangle \right\}, \\ \text{s.t. } u \in [0,1],\ \boldsymbol{p} \in [-w_b, w_b]. \end{aligned} \tag{3.3}$$

With the dual variable $\boldsymbol{p} \in \mathbb{R}^K$.

Algorithm 7 Primal-dual algorithm to solve graph cuts (3.3).

$\tau = \sigma = \frac{1}{\sqrt{8}},\ \theta = 1$
$u^0 = 0.5,\ \boldsymbol{p}^0 = 0,\ q^0 = 0$ and $\bar{u}^0 = u^0$ // Initialization
for $j = 1$ to J **do**
  $\boldsymbol{p}^j = \left[\boldsymbol{p}^{j-1} + \sigma\nabla\bar{u}^{j-1}\right]_{-w_b}^{w_b}$ // Update dual variable
  $u^j = \left[u^{j-1} - \tau(-\mathrm{div}\boldsymbol{p}^j + q^j)\right]_0^1$ // Update primal variable
  $\bar{u}^j = 2u^j - u^{j-1}$ // Extra-gradient step
end for

In comparison to the continuous optimization approach (3.9), the gradient operator ∇ is already defined on the discrete grid. We will limit ourselves to 4-connected ($K = 2$) and 8-connected ($K = 4$) graphs. For 4-connected graphs, the finite differences with Neumann boundary condition as defined in (2.26) are used. In the case of 8-connected graphs, we have

$$(\nabla v)_{i,j} = \left((\delta_x^+ v)_{i,j}, (\delta_y^+ v)_{i,j}, (\delta_{xy}^{++} v)_{i,j}, (\delta_{xy}^{+-} v)_{i,j}\right)^T , \tag{3.4}$$

with the forward gradients defined in (2.27)the additional diagonal gradients

$$\begin{aligned}(\delta_{xy}^{++} v)_{i,j} &= \begin{cases} v_{i+1,j+1} - v_{i,j} & \text{if} \quad i < M,\ j < N \\ 0 & \text{else} \end{cases}, \\ (\delta_{xy}^{+-} v)_{i,j} &= \begin{cases} v_{i+1,j-1} - v_{i,j} & \text{if} \quad i < M,\ j > 0 \\ 0 & \text{else} \end{cases}. \end{aligned} \tag{3.5}$$

∇ can be simply represented as a $MNK \times MN$ matrix.

Applying the primal dual algorithm in Algorithm 1 to the discrete max flow problem (3.3) is trivial, as we again have the primal variable $\alpha = u$ and the dual variable $\beta = \boldsymbol{p}$. The linear operator $D = \nabla$ with a Lipschitz constant $L^2 = 8$, and $\Phi(u) = \langle u, w_u\rangle + I_{\{u\in[0,1]\}}(u)$ and $\Psi^*(\boldsymbol{p}) = I_{\{|p^x|\leq w_b\}}(p^x) + I_{\{|p^y|\leq w_b\}}(p^y)$. We already solved a very similar problem in Section 2.11, and therefore directly summarize the resulting algorithm in Algorithm 7.

We showed that it is possible to solve the discrete graph cut problem by means of continuous convex optimization. An evaluation, and results can be found in Section 3.3.

3.1.2 The continuous formulation

The max-flow/min-cut problem is not restricted to the discrete setting only. In the continuous setting, the max-flow problem was first studied by [Strang, 1983], but remains challenging [Strang, 2009] up to now (e.g. the continuous equivalence of directed graphs).

[Klodt et al., 2008], showed an experimental comparison of continuous and discrete formulations. They showed that the well known metrication errors of discrete graph cuts, caused by the definition of the grid, can be easily overcome in a continuous formulation. As we will see later, this is especially important when using no or only slight edge information.

While the discrete and continuous approaches are defined in different settings, the problems are very closely related. [Sinop and Grady, 2007], presented a common segmentation model that evaluated different norms in the regularization term. Namely, the L^1 norm that corresponds to the discrete graph cut model, the squared L^2 norm and the L^∞ norm. The random walker framework of [Grady, 2006] is a special case of this segmentation model based on the squared L^2 norm, and thus is closely related to the Dirichlet problem. In [Couprie et al., 2011a], this connections were further extended to the watershed segmentation. A good overview on this topic is given in the thesis of [Couprie, 2011]. While closely related to the continuous maximum flow, the combinatorial continuous max-flow (CCMF) [Couprie et al., 2011b] solves the true analogous discrete formulation. The advantage of the CCMF lies in working on a discrete graph while preserving the advantages of the continuous formulation like no metrication errors.

[Appleton and Talbot, 2006], implemented the continuous max-flow equations using a finite-differences scheme to find a globally optimal solution of the continuous max-flow problem. [Chan et al., 2006] investigated convex minimization problems in computer vision, and established relations between image segmentation and denoising. A wide range of fast optimization methods for the max flow problem was also introduced in [Yuan, 2011]. All these methods have in common that they use the Total Variation as a smoothness term.

In the continuous setting there is a lot of work that was also influenced by the GAC formulation of [Caselles et al., 1997a]. This includes e.g. the work of [Leung and Osher, 2005] and [Unger et al., 2008a,b]. These methods are all based on the weighted Total Variation first introduced in [Bresson et al., 2007].

3.1.2.1 The optimization problem

The continuous equivalent to the weighted graph in the previous section, is a Riemannian space R, that consists of domain Ω and an associated metric $c_b : \Omega \to \mathbb{R}^+$. If we assume that $u, c_s, c_t, c_b : \Omega \to \mathbb{R}$ are now continuous functions, we can write the continuous min-

cut/max-flow problem as

$$\min_u \left\{ \int_\Omega c_b |\nabla u|_{2,1} + \int_\Omega c_t u \; d\boldsymbol{x} + \int_\Omega c_s (1-u) \; d\boldsymbol{x} \right\}, \quad \text{s.t. } u \in \{0,1\}. \tag{3.6}$$

We can again simplify the above model with $c_u = c_t - c_s$ and $\int_\Omega c_t u + c_s(1-u) \; d\boldsymbol{x} = \int_\Omega c_u u \; d\boldsymbol{x} + \int_\Omega c_s \; d\boldsymbol{x}$, neglecting the constant term. To make the above optimization problem convex, we relax the binary constraint to a continuous one $u(\boldsymbol{x}) = [0,1]$. As a result, the optimum is no longer guaranteed to be binary. The well known thresholding theorem [Strang, 1983], [Klodt et al., 2008] states that all upper level sets $\{x \in \Omega \mid u^*(\boldsymbol{x}) > \theta\}$, $\theta \in [0,1)$ of the optimal solution u^* of the relaxed problem provide a globally optimal solution to the binary labeling problem in (3.6). We therefore arrive at the following relaxed max flow formulation:

$$\min_u \left\{ \int_\Omega c_b |\nabla u|_{2,1} + \int_\Omega c_u u \; d\boldsymbol{x} \right\}, \quad \text{s.t. } u \in [0,1]. \tag{3.7}$$

As we have a discretized input image, we have to optimize a discrete version of (3.6). In a discrete setting we can again use vectors w_u for the unary and w_b for the binary terms exactly as used for the graph cut model in the previous section. The discrete minimization problem of the continuous maximum flow (3.6) thus becomes

$$\min_u \{ \|W_b \nabla u\|_{2,1} + \langle u, w_u \rangle \}, \quad \text{s.t. } u \in [0,1]. \tag{3.8}$$

Note that the only difference to the graph cut energy in (3.2) is the point-wise L^2 norm for the regularization term. The primal dual formulation of (3.8) is then given as

$$\min_u \max_{\boldsymbol{p}} \{ \langle \nabla u, \boldsymbol{p} \rangle + \langle u, w_u \rangle \}, \quad \text{s.t. } u \in [0,1], \; ||\boldsymbol{p}||_\infty \leq w_b. \tag{3.9}$$

Unfortunately, the discrete functional in (3.8) no longer guarantees a binary solution since the thresholding theorem holds only in the continuous formulation.

As everything necessary has already been introduced, the general primal dual algorithm (Algorithm 1), applied to the saddle point formulation in (3.9), is summarized in Algorithm 8.

Algorithm 8 Primal-dual algorithm to solve the continuous max flow problem (3.9).

$\tau = \sigma = \frac{1}{\sqrt{8}}$
$u^0 = 0.5$, $\boldsymbol{p}^0 = 0$, and $\bar{u}^0 = u^0$ // Initialization
for $j = 1$ to J **do**
  $\tilde{\boldsymbol{p}} = \boldsymbol{p}^{j-1} + \sigma \nabla \bar{u}^{j-1}$ // Update dual variable
  $\boldsymbol{p}^j = \frac{\tilde{\boldsymbol{p}}}{\max(w_b, |\tilde{\boldsymbol{p}}|)}$
  $u^j = \left[u^{j-1} - \tau(-\mathrm{div}\boldsymbol{p}^j + w_u)\right]_0^1$ // Update primal variable
  $\bar{u}^j = 2u^j - u^{j-1}$ // Extra-gradient step
end for

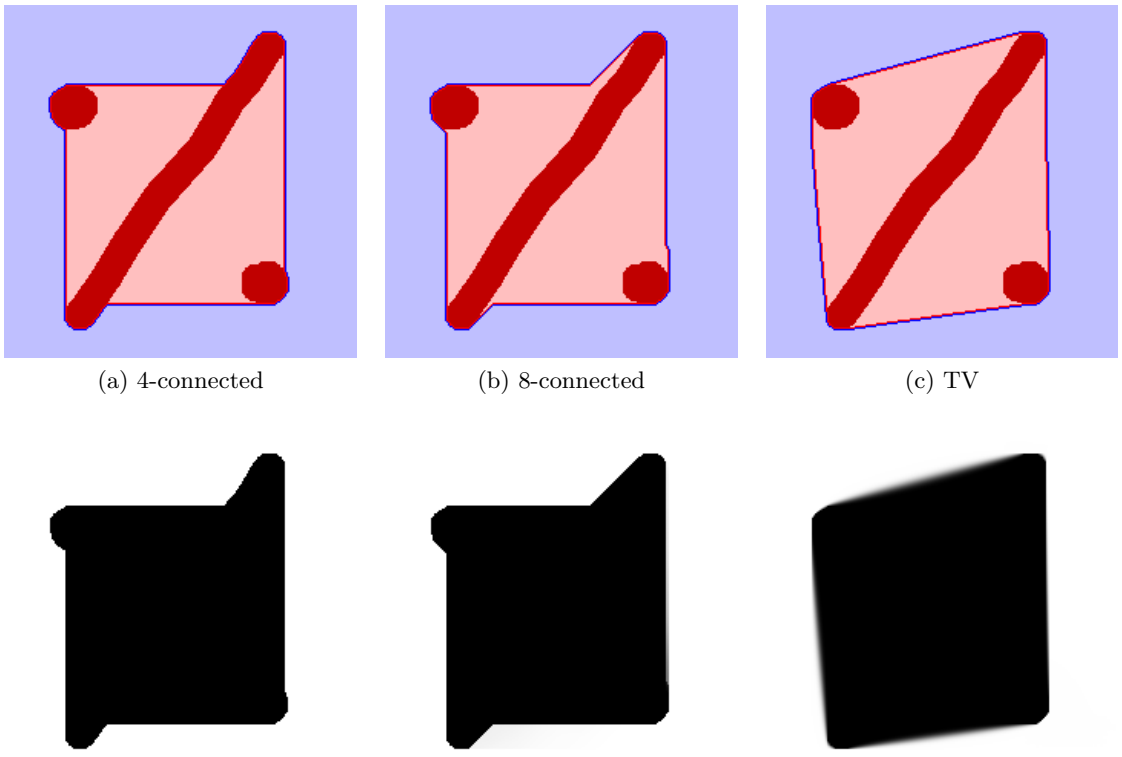
At this point we want to make some considerations on applying input data to the segmentation algorithm. In Section 4.1, a more comprehensive view is given in the context of interactive segmentation. Note that an additional parameter λ can be added similar to the ROF or TV-L1 model, to model the tradeoff between data fidelity and regularization. We neglected it here, as we can simply set the unary potentials to $w_u = \lambda w'_u$. We can differentiate between different ranges for w_u:

- $(w_u)_{i,j} = -\infty$: This is a hard foreground constraint that will enforce $u_{i,j} = 1$.
- $-\infty < (w_u)_{i,j} < 0$: A negative binary potential will vote for the foreground. The more negative the value, the more likely the final pixel will be $u_{i,j} = 1$. As there is also the influence of the regularization term, the final state of the pixel cannot be predicted.
- $(w_u)_{i,j} = 0$: For this pixel the data term will be 0. The final state of $u_{i,j}$ is solely determined by the regularization term.
- $0 < (w_u)_{i,j} < \infty$: A positive binary potential will vote for background.
- $(w_u)_{i,j} = \infty$: This is a hard background constraint that will enforce $u_{i,j} = 0$.

Theoretically, setting w_u to either $+/-\infty$ is sufficient to apply a hard constraint on certain pixels, but dealing with ∞ during implementation is not a trivial task. Additionally, there are problems when computing the energy for the primal dual gap we will introduce in Section 3.1.2.3. Instead we can apply simple additional constraints that enforce $u(x) = 0$ for background and $u(x) = 1$ for foreground. The addition to Algorithm 8 is trivial: First the pixels are initialized according to the constraints, and then never updated.

3.1.2.2 Discretization artifacts

An advantage of a TV based formulation is that in contrast to the graph cut energy, it does not suffer from metrication errors [Klodt et al., 2008]. In Figure 3.2, the difference between the discrete and continuous max flow formulation becomes obvious. Here, the dark red areas are constrained to be foreground pixels, and the border was set to background. We set the binary potentials to a constant value $w_b = 1$, so no edge information is used. The continuous (TV based) formulation efficiently minimizes the contour length (that equals the convex hull for this special set of constraints). On the other hand the discrete methods do not measure distances correctly, as they are based on a pre-defined grid. This grid structure (also see Figure 3.1) becomes obvious when comparing 4-connected and 8-connected graph neighborhoods. As shown by [Boykov and Kolmogorov, 2003], the finer



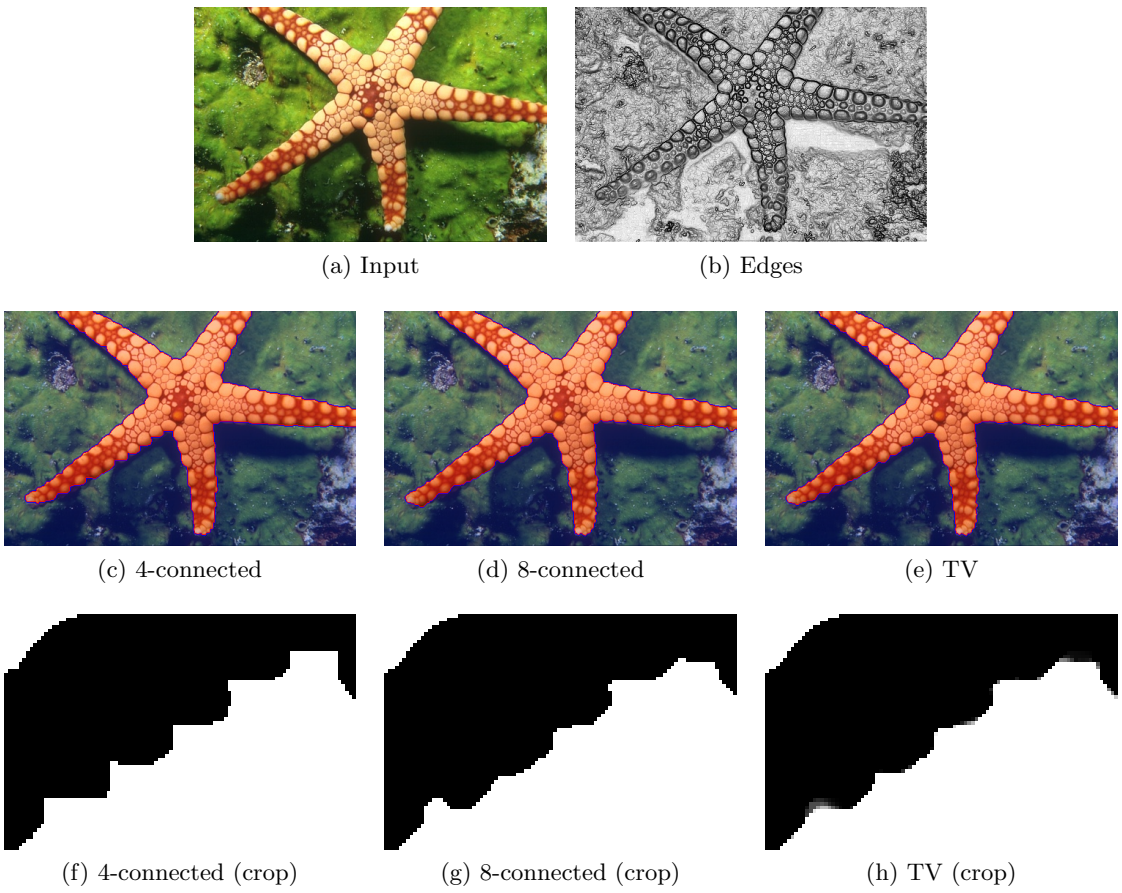
(a) 4-connected (b) 8-connected (c) TV

Figure 3.2: Comparison of discrete to continuous min cut/max flow problem with constant binary terms. The top row depicts the input with the segmentation as an overlay. The bottom row shows the output of the relaxed problems without thresholding. The graph based methods are biased by the underlying grid, while TV based continuous max flow minimizes the true Euclidean length.

the grid the closer the discrete min cut will be to the Euclidean length. To exactly model the Euclidean length, the angle between two edges has to become 0.

Note that the segmentation results in the bottom row of Figure 3.2 are not binary. This is due to the fact, that the according energies are not strictly convex. Thus each problem might have several global minima. By thresholding u in the range $[0, 1]$, each of these solutions can be obtained. Typically the solutions are very close to each other (note the slight blur of some edges).

In practice, often 8-connected graphs deliver acceptable results. Especially when considering edge information in the binary terms. This effect is demonstrated in Figure 3.3, where a strong edge weighting function was used. While for the 4-connected graph, the



(a) Input (b) Edges

(c) 4-connected (d) 8-connected (e) TV

(f) 4-connected (crop) (g) 8-connected (crop) (h) TV (crop)

Figure 3.3: Comparison of discrete to continuous min cut/max flow problem with strong binary terms incorporating edge information. While for 4-connected graphs the grid structure is very prominent, the 8-connected graph delivers results very similar to the continuous version.

discretization artifacts are still prominent, the 8-connected graph already gives a result close to the TV based result. See also Figure 3.12 for more comparisons on discrete and continuous segmentation results.

3.1.2.3 Convergence criterion

As we have iterative algorithms to solve the discrete or continuous segmentation problem, a convergence criterion is of great importance. The primal-dual gap as discussed in Section 2.3.3 offers such a criterion. In the following, we derive the normalized primal dual gap, which we will use as a convergence criterion. We start with the primal energy for the graph cut model and the TV model

$$\begin{aligned} E_p^{GC}(u) &= \|W_b \nabla u\|_{1,1} + \langle u, w_u \rangle + \langle u, w_s \rangle \,, \\ E_p^{TV}(u) &= \|W_b \nabla u\|_{2,1} + \langle u, w_u \rangle + \langle u, w_s \rangle \,. \end{aligned} \tag{3.10}$$

Note that we have to use the term $\langle u, w_s \rangle$ (neglected during optimization) for the energy calculation to ensure that $E_p > 0$. Otherwise the normalization in (3.14) would not make sense.

The primal-dual energies for the discrete (3.3) and continuous (3.9) model are the same with only the constraints on $\boldsymbol{p}$ differing. To find a dual only formulation, we have to obtain the optimal u for a given $\boldsymbol{p}$ as

$$\begin{aligned} \hat{u} = \arg\min_u \left\langle u, \nabla^T \boldsymbol{p} + w_u \right\rangle , \\ \text{s.t. } u \in [0,1]. \end{aligned} \tag{3.11}$$

It is trivial to obtain the optimal $\hat{u}$ as

$$\hat{u}_{i,j} = \begin{cases} 1 & \text{if} \quad (\nabla^T \boldsymbol{p})_{i,j} + (w_u)_{i,j} < 0 \\ 0 & \text{else} \end{cases} . \tag{3.12}$$

Therefore, the dual energy E_d can be written as

$$E_d(\boldsymbol{p}) = \left\langle \hat{u}, \nabla^T \boldsymbol{p} + w_u \right\rangle + \langle u, w_s \rangle \,. \tag{3.13}$$

The normalized primal dual gap is then given as

$$G(u, \boldsymbol{p}) = \frac{E_p(u) - E_d(\boldsymbol{p})}{E_p(u)}. \tag{3.14}$$

The gap will become 0 if u reaches the globally optimal solution. Thus it provides an excellent optimality measure [Chambolle and Pock, 2010].

3.1.3 Connections to other segmentation models

Connection to Mumford Shah: In Section 2.2.3, we already introduced the Mumford Shah segmentation model. Using the binary labeling function u and TV, we can reformulate the Chan-Vese model defined in (2.23) as

$$\min_{u,c_1,c_2} \left\{ \nu \int_\Omega |\nabla u|_2 + \int_\Omega u(c_1 - f)^2 d\boldsymbol{x} + \int_\Omega (1-u)(c_2 - f)^2 d\boldsymbol{x} \right\}, \quad \text{s.t. } u \in \{0,1\} \ . \tag{3.15}$$

We denoted the input image as $I : \Omega \to \mathbb{R}$. If we further assume that c_1 and c_2 are no longer subject to optimization, but given in advance, we can reformulate the problem further as

$$\min_u \left\{ \int_\Omega |\nabla u|_2 + \int_\Omega c_t u \ d\boldsymbol{x} + \int_\Omega c_s (1-u) \ d\boldsymbol{x} \right\} \ , \tag{3.16}$$

where we set $c_t = \frac{(c_1 - f)^2}{\nu}$ and $c_s = \frac{(c_2 - f)^2}{\nu}$. Up to the binary potentials, this formulation corresponds exactly to the continuous maximum flow formulation in (3.6). We conclude that in the case of fixed c_1 and c_2 the Chan-Vese model equals the continuous maximum flow formulation.

Connection to shape denoising: Section 2.2.2 already revealed that the weighted TV-L1 model (2.20) can be used for shape denoising. (2.20) can be rewritten as

$$\min_u \left\{ \int_\Omega c_b |\nabla u|_2 + \lambda \int_\Omega |u - f| d\boldsymbol{x} \right\}, \quad \text{s.t. } u \in \{0,1\} \ . \tag{3.17}$$

We already used a similar formulation for image segmentation in [Unger et al., 2008a,b], and referred to it as TVSeg. Additionally, we showed in [Unger et al., 2008a] that from an algorithmic point of view this approach is equivalent to the continuous maximum flow algorithm of [Appleton and Talbot, 2006]. Note that we used an L^2 data term in [Unger et al., 2008a].

Although, the formulation in (3.17) looks quite similar to the continuous maximum flow in (3.7) the data term is quite different. The L^1 data term allows to set f to either foreground ($f = 1$) or background ($f = 0$). In order to be comparable to the maximum

flow approach, we can limit the unary potentials in (3.7) to $\{1, -1\}$. An $f = \{0, 1\}$ for the weighted TV-L1 model would then correspond to an $c_u = \{1, -1\}$ for the maximum flow model.

For [Unger et al., 2008b], we made the weighting parameter λ spatially dependent

$$\min_u \left\{ \int_\Omega c_b |\nabla u|_2 + \int_\Omega \lambda(\boldsymbol{x}) |u - f| d\boldsymbol{x} \right\}, \quad \text{s.t. } u \in \{0, 1\} . \tag{3.18}$$

with $\lambda : \Omega \to \mathbb{R}^+$. The spatially varying $\lambda(\boldsymbol{x})$ efficiently circumvents the aforementioned problem of discrete unary potentials, but was never used in this sense. While still limiting $f(\boldsymbol{x})$ to the interval $\{0, 1\}$ the pixel wise $\lambda(\boldsymbol{x})$ can be used to model pixel wise probabilities. Unfortunately a direct correspondence to the max flow problem cannot be established. In case of the max flow data term $\int_\Omega c_u u \, d\boldsymbol{x}$, the actual value of the unary potential $c_u(\boldsymbol{x})$ has no influence as soon as $u(\boldsymbol{x}) = 0$. In case of (3.18), there is still a linear dependence.

3.2 Multi-label image segmentation

In the previous section the segmentation model was restricted to foreground and background. We will now focus on the general minimal partition problem for an arbitrary number of labels.

$$\begin{aligned} &\min_{\Omega_i} \left\{ \frac{1}{2} \sum_{i=1}^{K} \mathrm{Per}(\Omega_i) + \sum_{i=1}^{K} \int_{\Omega_i} f_i(\boldsymbol{x}) d\boldsymbol{x} \right\}, \\ &s.t. \quad \Omega = \bigcup_{i=1}^{K} \Omega_i , \quad \Omega_i \cap \Omega_j = \emptyset \quad \forall i \neq j . \end{aligned} \tag{3.19}$$

Here f_i are now the unary potentials, that are integrated inside the region Ω_i. The regularization term penalizes the contour length, and is weighted with $\frac{1}{2}$ to account for double counting. For the rest of this thesis we neglect this scale factor, as we can apply arbitrary weighting by scaling f_i.

Note, that there is a very close connection to the piecewise constant Mumford-Shah functional discussed in Section 2.2.3. We can simply set

$$f_i(\boldsymbol{x}) = \lambda (c_i - I((x)))^2 , \tag{3.20}$$

with a given input image I and the mean intensities c_i, similar as done in the binary case

in Section 3.1.3.

In the discrete setting the model in (3.19) is usually referred to as the Potts model, and is known to be an NP-hard problem. The Potts model was first introduced in the context of physics and statistical mechanics in [Potts, 1952]. It generalizes the Ising model [Ising, 1925] to multiple labels. Important contributions were made by [Ishikawa, 2003], and the family of methods discussed in Section 1.2.2.2 can be used to approximately solve the discrete multi label problem.

In the following we focus on the convex approximations of the continuous minimal partition problem. More precisely we present a fast relaxation approach based on the work of [Zach et al., 2008]. We chose this straightforward approach for its speed and ease of use. While the relaxation is not the tightest, it delivers very good results in practical applications. Other approaches are discussed in the subsequent section.

3.2.1 Fast relaxation

As stated by the co-area formula (2.12), the length of a contour can be expressed by the TV. This requires that we represent the single labels by their characteristic functions $u_i : \Omega \to \mathbb{R}$. We therefore rewrite the minimal partition problem (3.19) as

$$\min_{u_i} \left\{ \sum_{i=1}^{K} \int_\Omega |\nabla u_i| + \sum_{i=1}^{K} \int_\Omega u_i(\boldsymbol{x}) f_i(\boldsymbol{x}) d\boldsymbol{x} \right\}, \tag{3.21}$$

with the additional segmentation constraints of non overlapping regions as in (1.1) and $u \in \{0, 1\}$. [Zach et al., 2008] proposed to relax the constraints in (1.1) to $\sum_{i=1}^{K} u_i = 1$, and the characteristic functions to $u \in [0, 1]$. We can additionally add the edge information by adding binary terms to TV with $c_b : \Omega \to \mathbb{R}$. The resulting optimization problem is then given as

$$\begin{aligned} &\min_{u_i} \left\{ \sum_{i=1}^{K} \int_\Omega c_b |\nabla u_i| + \sum_{i=1}^{K} \int_\Omega u_i f_i d\boldsymbol{x} \right\}, \\ &s.t. \sum_{i=1}^{K} u_i = 1,\ u_i \geq 0\ \forall i = 1, \ldots, K\ . \end{aligned} \tag{3.22}$$

We transform the relaxed version of (3.22) into the following discretized primal-dual

saddle point formulation

$$\min_{u_i} \max_{\boldsymbol{p}_i, r} \left\{ \sum_{i=1}^{K} \langle \boldsymbol{p}_i, \nabla u_i \rangle + \sum_{i=1}^{K} \langle u_i, f_i \rangle + \left\langle r, \sum_{i=1}^{K} u_i - 1 \right\rangle \right\}, \quad s.t.\ u \geq 0,\ ||\boldsymbol{p}_i||_\infty \leq w_b. \tag{3.23}$$

Here we introduced the Lagrange multiplier $r \in \mathbb{R}$ to obtain a primal-dual formulation of the sum constraint.

(3.23) can be written in the form of (2.50) with

$$\alpha = \begin{pmatrix} (u_1)^T \dots (u_K)^T \end{pmatrix}^T, \quad D = \begin{pmatrix} \nabla & & \\ & \ddots & \\ & & \nabla \\ \hline I & \dots & I \end{pmatrix} \text{ and } \beta = \begin{pmatrix} \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_K \\ \hline r \end{pmatrix}. \tag{3.24}$$

This results in a Lipschitz constant $L^2 = 9$. We further get $\Phi(u) = \sum_{i=1}^{K} \left(\langle u_i, f_i \rangle + I_{\{u_i \geq 0\}}(u) \right)$ and $\Psi^*(\boldsymbol{p}, r) = r + \sum_{i=1}^{K} I_{\{||\boldsymbol{p}_i||_\infty \leq w_b\}}(\boldsymbol{p})$. The resulting algorithm is summarized in Algorithm 9.

While the relaxation of [Zach et al., 2008] is very fast, it might give inaccurate results. Especially the constraint $\sum_{i=1}^{K} u_i = 1$ does not guarantee binary labeling. Instead multiple labels might have a value $0 < u_i < 1$. In practice this approach delivers very good results. Due to the low memory consumption and speed, this algorithm is recommended for interactive segmentation as well as problems with a large number of labels.

3.2.2 Other relaxations

A variant of (3.21) was introduced by [Lellmann et al., 2009b]. They used the vectorial TV and relaxed the minimum partitioning problem to

$$\min_{\boldsymbol{u}} \left\{ \int_\Omega \sqrt{||\nabla u_1||^2 + \dots + ||\nabla u_K||^2} d\boldsymbol{x} + \sum_{i=1}^{K} \int_\Omega u_i(\boldsymbol{x}) f_i(\boldsymbol{x}) d\boldsymbol{x} \right\}. \tag{3.25}$$

They solve the above problem using Douglas-Rachford splitting [Douglas and Rachford, 1956]. The resulting algorithm should deliver similar results to the approach of [Zach et al., 2008].

Algorithm 9 Algorithm for solving the fast multi-label segmentation approximation by [Zach et al., 2008] in (3.23).

// Initialization
$\tau = \sigma = \frac{1}{\sqrt{9}}$
$\boldsymbol{p}_i^0 = (0,0)^T$, $r^0 = 0$ and $\bar{u}_i^0 = u_i^0 = \frac{1}{K}$

for $j = 1$ to J **do**
  // Update dual variables
  for $i = 1$ to K **do**
    $\tilde{\boldsymbol{p}}_i = \boldsymbol{p}_i^{j-1} + \sigma \nabla \bar{u}_i^{j-1}$
    $\boldsymbol{p}_i^j = \frac{\tilde{\boldsymbol{p}}_i}{\max(w_b, |\tilde{\boldsymbol{p}}_i|)}$
  end for
  $r^j = r^{j-1} + \sigma \left(\sum_{i=1}^K \bar{u}_i^{j-1} - 1 \right)$

  // Update primal variables
  for $i = 1$ to K **do**
    $u_i^j = \max \left(0, u_i^{j-1} - \tau \left(-\mathrm{div} \boldsymbol{p_i}^j + r^j + f_i \right) \right)$
    $\bar{u}_i^j = 2u_i^j - \bar{u}_i^{j-1}$
  end for
end for

In [Lellmann et al., 2009a], the approaches of [Lellmann et al., 2009b] and [Zach et al., 2008] are generalized to arbitrary label distances. They achieve this by extending the vector valued TV by an embedding matrix. Nestorov's algorithm [Nesterov, 2004] is used to solve the resulting convex optimization problem.

A tight relaxation of the minimal partition problem 3.19 was introduced in [Pock et al., 2009a] for the Potts regularization. Later in [Chambolle et al., 2008, 2012] this approach was extended to general label distance functions. They use a labeling function $v : \Omega \to \{0, \ldots, K\}$, that is then represented as

$$v(\boldsymbol{x}) = \sum_{i=1}^{K} \theta_i(\boldsymbol{x}), \tag{3.26}$$

with the K binary functions $\boldsymbol{\theta(x)} = (\theta_1(\boldsymbol{x}), \ldots \theta_K(\boldsymbol{x}))$ such that

$$\theta_i(\boldsymbol{x}) = \begin{cases} 1 & \text{if} \quad v(\boldsymbol{x}) \geq i \\ 0 & \text{else} \end{cases} . \tag{3.27}$$

They then express the perimeter with the following constrained primal dual formulation

$$\sum_{i=1}^{K} \mathrm{Per}(\Omega_i) = \sup_{\boldsymbol{\xi} \in \mathcal{K}} \left\{ \sum_{i=1}^{K} - \int_{\Omega} \theta_i \mathrm{div} \xi_i \right\} , \tag{3.28}$$

with the dual variables $\boldsymbol{\xi}$ constrained to lie in the set

$$\mathcal{K} = \left\{ \boldsymbol{\xi} = (\xi_1, \ldots, \xi_K) : \Omega \to \mathbb{R}^{dK}, \left| \sum_{i_1 \leq i \leq i_2} \xi_i(\boldsymbol{x}) \right| \leq 1, \ \forall \boldsymbol{x} \in \Omega, 1 \leq i_1 \leq i_2 \leq K \right\} , \tag{3.29}$$

While this approach delivers slightly superior results, the computational complexity and memory consumption gets intractable for a large number of labels. Although, up to a number of at least 10 labels this could be used for interactive segmentation, we stick to the approach in Section 3.2.1.

3.2.3 Label costs

We extend the minimum partition problem in (3.21) with an additional label cost term, as proposed by [Yuan and Boykov, 2010].

$$\begin{aligned} \min_{\Omega_i} \left\{ \sum_{i=1}^{K} \mathrm{Per}(\Omega_i) + \sum_{i=1}^{K} \int_{\Omega_i} f_i(\boldsymbol{x}) d\boldsymbol{x} + \gamma ||\mathbf{1}_{\Omega_i}||_{\infty} \right\} , \\ s.t. \quad \Omega = \bigcup_{i=1}^{K} \Omega_i , \quad \Omega_i \cap \Omega_j = \emptyset \quad \forall i \neq j . \end{aligned} \tag{3.30}$$

The infinity norm penalizes the maximum value of the characteristic function $\mathbf{1}_{\Omega_i}$, and minimizes the number of non-empty segments. We again use the characteristic functions $u_i : \Omega \to \mathbb{R}$ to represent the regions Ω_i. The label cost term then becomes $\sum_{i=1}^{K} \gamma ||u_i||_{\infty}$.

In order to optimize the label cost term, we follow the ideas proposed in [Yuan and Boykov, 2010]. To replace the L^{∞}-norm in $||u||_{\infty}$, the scalar t is introduced. We then arrive at the equivalent representation

$$t, \quad s.t. \ u_{k,l} \leq t, \ \forall k, l. \tag{3.31}$$

The pixel wise constraints can be written in a single term when using a primal-dual formulation $\left\langle q, P \begin{pmatrix} u \\ t \end{pmatrix} \right\rangle$. Where the Lagrange multiplier $q \in \mathbb{R}^{MN}$ has to be positive to

account for the inequality constraint. We define the $(MN+1)\times MN$ matrix $P = \left(I, -\mathbf{1}^T\right)$, with the $MN \times MN$ identity matrix I and the row vector $\mathbf{1}$ of size MN with all entries equal to 1.

When combined with the fast relaxation in (3.23), the multi-label segmentation problem with label costs can be written as the following saddle point problem

$$\min_{u_i,t_i} \max_{\boldsymbol{p}_i,r,q_i} \left\{ \sum_{i=1}^{K} \langle \boldsymbol{p}_i, \nabla u_i \rangle + \sum_{i=1}^{K} \langle u_i, f_i \rangle + \left\langle r, \sum_{i=1}^{K} u_i - 1 \right\rangle + \gamma t_i + \sum_{i=1}^{K} \left\langle q_i, P \begin{pmatrix} u_i \\ t \end{pmatrix} \right\rangle \right\}, \quad s.t.\ u_i \geq 0,\ ||\boldsymbol{p}_i||_\infty \leq w_b,\ q_i \geq 0. \tag{3.32}$$

The above saddle point problem can be brought to the form of (2.50):

$$\begin{aligned}
\alpha &= \left(\begin{array}{c|c} (u_1)^T \dots (u_K)^T & (t_1)^T \dots (t_K)^T \end{array} \right)^T, \\
D &= \left(\begin{array}{ccc|ccc} \nabla & & & & & \\ & \ddots & & & & \\ & & \nabla & & & \\ \hline I & & & -\mathbf{1}^T & & \\ & \ddots & & & \ddots & \\ & & I & & & -\mathbf{1}^T \\ \hline I & \dots & I & & & \end{array} \right) \text{ and } \beta = \left(\begin{array}{c} \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_K \\ \hline q_1 \\ \vdots \\ q_K \\ \hline r \end{array} \right). \\
\Phi(\alpha) &= \sum_{i=1}^{K} \left(\langle u_i, f_i \rangle + \gamma t_i + I_{\{u_i \geq 0\}}(u_i) \right), \\
\Psi^*(\beta) &= r - \sum_{i=1}^{K} \left(I_{\{||\boldsymbol{p}||_\infty \leq w_b\}}(\boldsymbol{p_i}) + I_{\{q_i \geq 0\}}(q_i) \right).
\end{aligned} \tag{3.33}$$

The resulting algorithm is summarized in Algorithm 10.

While regularization and dataterm sum up over the whole image, the variable t_i is in the range $[0,\ 1]$. To account for this imbalance, we introduce a normalized γ_n and set $\gamma = \gamma_n MN$ by scaling γ_n with the number of pixels. In Figure 3.4, the effect of the label cost term is demonstrated. With increasing γ the number of non empty labels is reduced. If γ gets to high, the segmentation reduces to a single label as can be seen in Figure 3.4(d). Meaningful values are typically in the range $\gamma_n \in [0,\ 0.015]$.

The label cost term poses only small benefit in interactive segmentation. But, we demonstrate in Section 5, that the label cost term helps to automatically determine the

Algorithm 10 Algorithm for solving the fast multi-label segmentation approximation with label cost term (3.23).

// Initialization
$\tau = \sigma = \frac{1}{L}$
$\boldsymbol{p}_i^0 = (0,0)^T$, $q_i^0 = 0$, $\bar{u}_i^0 = u_i^0 = \frac{1}{K}$, $\bar{t}_i^0 = t_i^0 = 0$ and $r^0 = s^0 = 0$

for $j = 1$ to J **do**
  // Update dual variables
  for $i = 1$ to K **do**
    $\tilde{\boldsymbol{p}} = \boldsymbol{p}_i^{j-1} + \sigma \nabla \bar{u}_i^{j-1}$
    $\boldsymbol{p}_i^j = \frac{\tilde{\boldsymbol{p}}}{\max(w_b, |\tilde{\boldsymbol{p}}|)}$
    $q_i^j = \max\left(0, q_i^{j-1} + \sigma\left(P\begin{pmatrix}\bar{u}_i^{j-1}\\ \bar{t}_i^{j-1}\end{pmatrix}\right)\right)$
  end for
  $r^j = r^{j-1} + \sigma\left(\sum_{i=1}^K \bar{u}_i^{j-1} - 1\right)$

  // Update primal variables
  for $i = 1$ to K **do**
    $t_i^j = t_i^{j-1} - \tau\left(\gamma - ||q_i||\right)$
    $\bar{t}_i^j = 2t_i^j - \bar{t}_i^{j-1}$
    $u_i^j = \max\left(0, u_i^{j-1} - \tau\left(-\mathrm{div}\boldsymbol{p_i}^j + q_i^j + r^j + f_i\right)\right)$
    $\bar{u}_i^j = 2u_i^j - \bar{u}_i^{j-1}$
  end for
end for

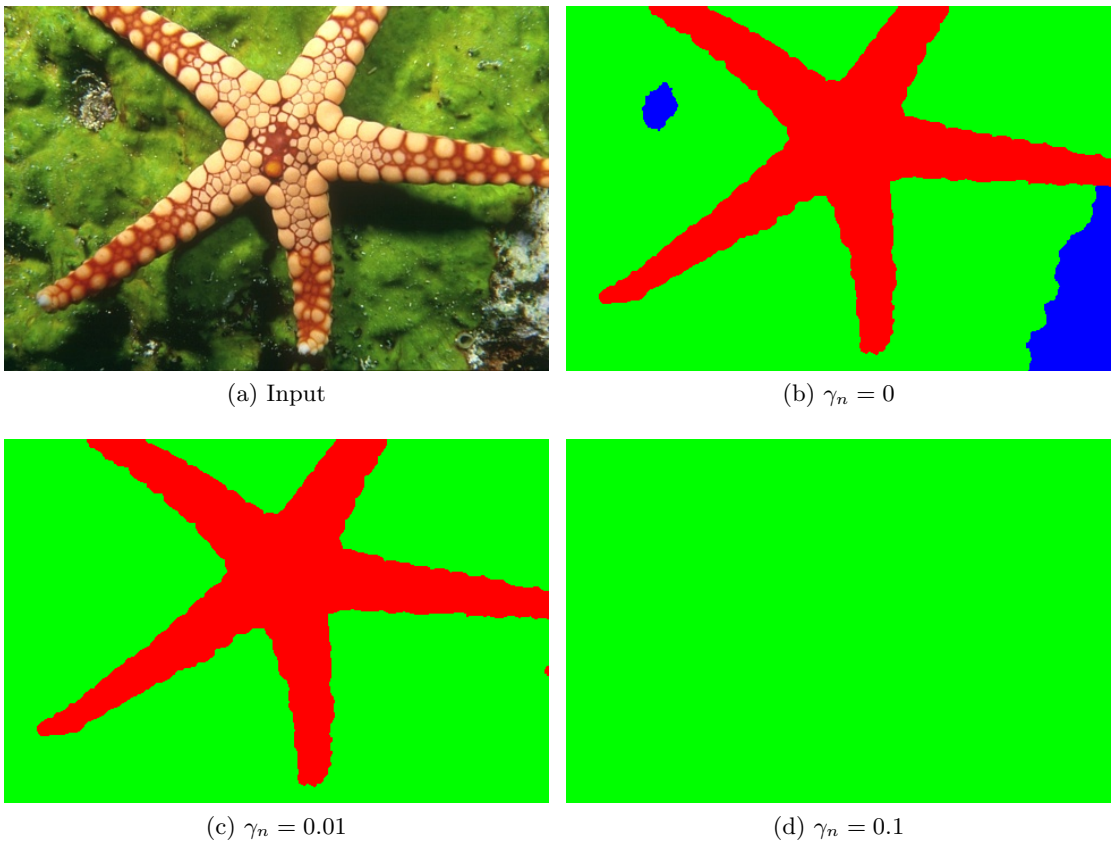
number of necessary labels in unsupervised segmentation.

3.3 Fast Optimization

After we have discussed variational approaches to binary as well as multi label segmentation, we further investigate fast optimization of these models. We therefore first make considerations on the implementation of the algorithms. Later we show algorithmic improvements for the binary segmentation task, and present an evaluation in terms of speed and convergence rates.

3.3.1 Thoughts on the implementation

While the algorithms presented in this thesis are computationally expensive, they are in general inherently parallel. When we look at Algorithm 1, the iterative updates are usually



(a) Input (b) $\gamma_n = 0$

(c) $\gamma_n = 0.01$ (d) $\gamma_n = 0.1$

Figure 3.4: Demonstration of the label cost term for multi label image segmentation. With increasing label costs, the number of non empty segmentation labels decreases.

pixel wise. Note, that this is not the case for all algorithms (e.g. the sum in Algorithm 9). In the ideal case, this would allow for independent threads for each pixel. Of course current hardware offers only a limited amount of processors. Modern graphics processing units (GPUs) are equipped with several hundred to thousands of processors, and are therefore perfectly suited for implementing variational methods. While the central processing unit (CPU) generally relies on a single instruction, single data (SISD) architecture, the GPU uses a single instruction, multiple data (SIMD) architecture. Again, the SIMD architecture perfectly suites our algorithms, where we can apply the same operations to each pixel in parallel. Additionally the computational power in terms of GFlops and memory bandwidth of modern GPUs has already significantly surpassed the computational power of CPUs, as can be seen in Figure 3.5.

As a result, general-purpose computing on graphics processing units (GPGPU) has

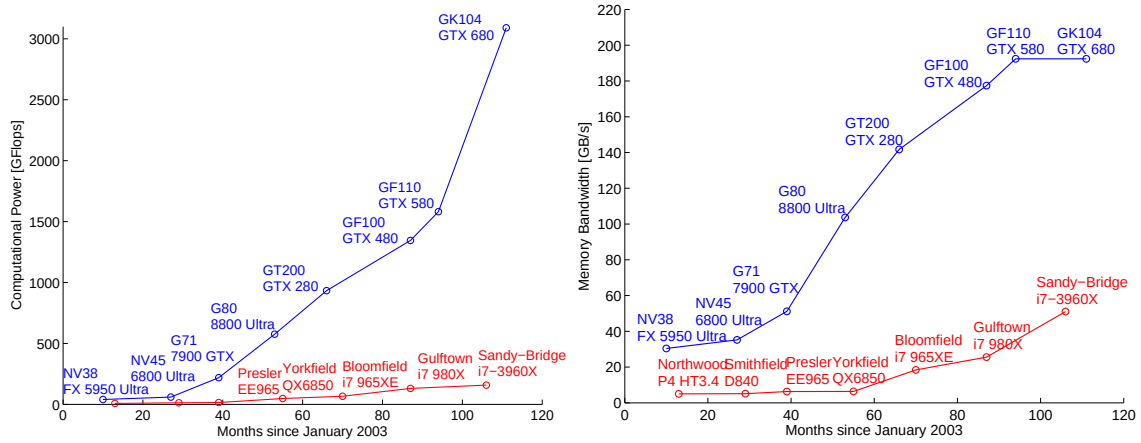


Figure 3.5: Comparison of GPU (blue) and CPU (red) performance: On the left, the theoretical number of floating point operations is depicted over time. The right hand side, shows the evolution of the maximum memory bandwidth. In both numbers GPUs significantly exceed the computational power of CPUs.

become increasingly important over the last years. Especially the introduction of the compute unified device architecture (CUDA) [NVidia, 2011b] at the end of 2006, gave a big boost to GPU implementations in computer vision. With the NVidia GeForce 8800 being the first GPU featuring the new "unified shader architecture" specifically designed for GPGPU applications [Glaskowsky, 2009]. The CUDA framework* allows to write C and C++ code directly for the GPU. A similar approach was taken by the open computing language (OpenCL) †. While the proprietary CUDA framework is only designed for NVidia GPUs, OpenCL is an open standard that can be implemented for any GPU or CPU. We use the CUDA framework for all implementations presented in this thesis, as it allows better control of the hardware and special features. Though, we believe that OpenCL will become increasingly important in the future as more and more implementations occur, allowing to produce code that can be executed on different platforms. The parallelization potential makes the algorithms not only suitable for GPUs, but also for other highly parallel devices such as an FPGA [Akin et al., 2011].

In the following we give some details on parallel GPU as well as CPU implementations and do some performance evaluations based on the hardware setup listed in Table 3.1.

*`http://www.nvidia.com/object/cuda_home_new.html` (as of 29.09.2012)
†`http://www.khronos.org/opencl/` (as of 29.09.2012)

Type	Product Name	Cores	Clock	Memory
GPU	NVidia GTX 580	512	1,56 GHz	3 GB
Desktop CPU	Intel i7 960	4 (8)	3,2 GHz	12 GB
Server CPU	2x Intel Xeon X5650	12	2,67 GHz	96 GB

Table 3.1: Used setups for performance evaluations.

GPU implementations: Most of the algorithms presented in this thesis consist exclusively of iterative point wise updates that need only their neighboring pixel information e.g. for the gradient operator ∇ in Algorithm 8. We will therefore give a few details on the implementation of these basic algorithms with CUDA as these are the main part of all presented algorithms. Special cases like e.g. the sum over an image, require special implementations and are not the focus of this thesis.

We first introduce basic concepts of CUDA enabled GPUs. For a full introduction we refer the interested reader to [Nguyen, 2007; NVidia, 2011a,b; Sanders and Kandrot, 2010]. The following assumptions are all based on the NVidia GPUs using the Fermi architecture, but generally hold for all CUDA enabled GPUs (including the Kepler architecture). All numbers we give as examples are optimal values for a NVidia GTX 580 GPU. Details on the architecture can be found in [Glaskowsky, 2009].

Processors on a GPU are organized in groups, so called multi-processors. The number of threads computed simultaneously by a multi-processor (e.g. 32) is called a warp, and due to pipelining usually double the number of processors (e.g. 16). All threads within a warp execute the same command according to the principles of the SIMD architecture. Therefore it is important to avoid diverging threads, as they have to be serialized. A program that is executed on the GPU is referred to as a kernel.

Threads in CUDA are organized in blocks and grids. While blocks are restricted to a two-dimensional layout, threads in a grid are organized in a three-dimensional layout. The first fragmentation occurs at the block level. Each block will be calculated on a multi-processor and consists of a grid of threads. While all threads inside a block can be easily synchronized, the execution order of the blocks is arbitrary. As a consequence global synchronization can only be performed after a kernel finished. For our algorithms the most obvious fragmentation is therefore to make a thread for each pixel and organize the threads in local blocks.

Each multi-processor has access to local memory, namely registers and shared memory that provides very fast access to a small (64K) amount of memory. The main memory

of the graphics card (DRAM) is usually referred to as global memory, and is significantly slower than the local memory. As a result the most important design principle is to reduce the amount of memory reads and writes. Additionally memory access should be coalesced (consecutive access of aligned memory), as segments (of e.g. 128 bytes) are read in a single read operation. Another prominent advantage of GPUs are textures. Textures not only provide cached memory access, but also free bilinear interpolation and border handling. With the introduction of the Fermi architecture the shared memory can partly be used as an L1 cache, eliminating texture usage for caching only.

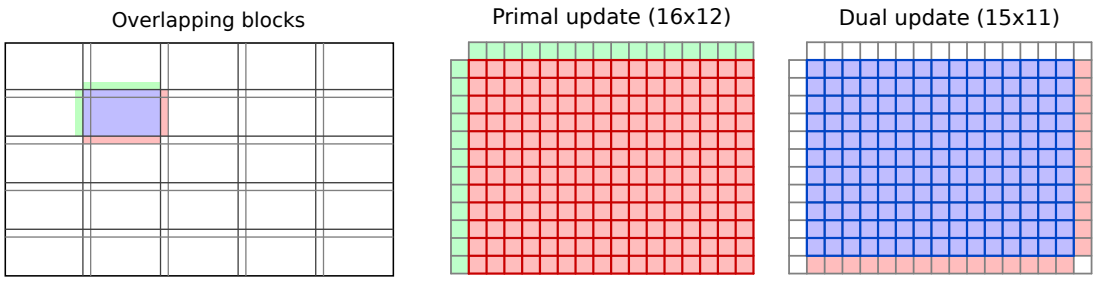
Using this basic knowledge on the GPU architecture, we can implement our algorithms using the following principles:

- A fragmentation into blocks (of width 16) results in an optimal memory and warp alignment (also see Figure 3.6a).

- Blocks should be as big as possible to be able to synchronize threads within a block. The maximum size is usually limited by the amount of local memory (registers, shared memory) needed. Additionally, there is a maximum number of threads in a block (1024). Further, at least three times more blocks than multi-processors are required to get optimal occupancy (pipelining).

- Images are stored in an aligned memory layout to provide fast access using caching in 2D as well as 3D.

- A single kernel for a complete iteration avoids unnecessary memory transfers, even if this means that some calculations have to be done twice. In Figure 3.6, we depicted an example fragmentation required for a complete iteration inside a single kernel. First the grid is defined with one pixel overlap at the right hand side and the bottom. For the primal update, the divergence operator needs its left and top neighbors. Using texture fetches (and caching) the divergence can be efficiently computed using global memory reads. We can therefore update all pixels handled by the block. As the information of the primal update is required by the dual update, we store the result in the shared memory and make sure that all threads are synchronized. In the dual update, the ∇ operator requires the right and bottom neighbors. While we can efficiently use the data from the shared memory, it is not possible to access the neighbors outside the block as a global synchronization cannot be performed inside a kernel. Consequently, the right and bottom borders are not updated and only the results inside the blue area (Figure 3.6b) are stored back to global memory.



(a) Fragmentation of blocks

(b) Grid for primal and dual updates (one block)

Figure 3.6: The blocks of the grid are defined with one pixel overlap as can be seen on the left hand side. First the primal update is calculated for the full block. Here the div operator needs the left and top neighbors. For the dual update the ∇ operator needs the right and bottom neighbors. Therefore, we only update the block without its right and bottom border, using the data from the primal update.

Unfortunately, a single kernel is not always possible. This might be the case for too large memory requirements (Algorithm 9), or separate calculations as the sparse matrix multiplication (e.g. in Algorithm 15). When splitting kernels, excessive memory transfers should be avoided.

CPU implementations: Current CPUs follow the trend of parallelization and increase the number of processors on a single chip with almost every generation. Compared to GPUs the number of processors is still significantly lower, as can be seen in Table 3.1. On the other hand CPUs offer a more powerful instruction set, larger cache sizes and higher clock rates. We used the OpenMP[‡] framework for parallelization as well as Intel IPP[§] functions for an efficient implementation.

Similar to GPUs, the memory transfers are very costly. Therefore, good memory management and efficient caching strategies are important. Interestingly, the same block wise strategy used on the GPU also proved to be the most successful on the CPU. We thus again parallelized with the same block structure as in Figure 3.6, but this time using a block size of 32×32 as there is more local memory available. Each block corresponds to a thread and first loads the necessary data into an array. The updates are then the same as on the GPU. Only the blue area (Figure 3.6b) is then stored back to the main memory.

While on the Xeon server, a simple row wise parallelization is as fast as the block wise

[‡] `http://openmp.org` (as of 29.09.2012)

[§] `http://software.intel.com/en-us/intel-ipp` (as of 29.09.2012)

Size	32×32	50×50	100×100	128×128	256×256	481×312
Num. pixels	1024	2500	10000	16384	65536	150072
GPU	164924	164516	163978	163738	52878	25808
Desktop CPU	57624	24658	6726	4104	1004	435
Server CPU	68609	47519	14503	10480	2859	1100

512×512	640×480	1024×768	1200×1200	2000×2000	3160×3160
262144	307200	786432	1440000	4000000	9985600
15544	13695	5426	3016	1094	462
255	218	85	47.4	17	7
673	606	247	143	55.7	22.2

Table 3.2: Evaluation results depicting the number of iterations per second of Algorithm 8 on different architectures. A visualization of the data can be found in Figure 3.7.

parallelization, this is not the case on the i7 desktop CPU. We believe that this is mainly due to better caching strategies of the server CPU.

Performance comparison: To make a quantitative performance comparison, we optimized the binary segmentation Algorithm 8 both on the GPU and the CPU using the principles from above. An evaluation was done using the three setups presented in Table 3.1. As a measure, we report the number of iterations calculated per second. Additionally, we varied the image size to demonstrate the dependence on the number of input pixels. The measurements are summarized in Table 3.2 and visualized in Figure 3.7.

From the left hand side in Figure 3.7 we can note that the number of iterations per second is in general inverse proportional to the number of pixels. We also included very small images in our evaluation, to reveal an interesting effect: When going below an image width of 128 using the GPU implementation, the number of iterations per second saturates at approximately 160000. This effect is caused by the memory alignment in global GPU memory. As mentioned the GPU reads memory in big segments at once. Once the image size gets below this segment size, memory transfer still cost the same. As a result only the actual computation, that contributes only a very small part to the overall time, will get faster. Also note, that the server multi-CPU setup shows a slight decrease from the inverse proportionality for small images. This is due to the fact, that the communication overhead between the two separate CPUs becomes apparent.

On the right hand side of Figure 3.7 the speedup of the GPU implementation compared to the CPU implementations is depicted. This shows that due to the previous mentioned

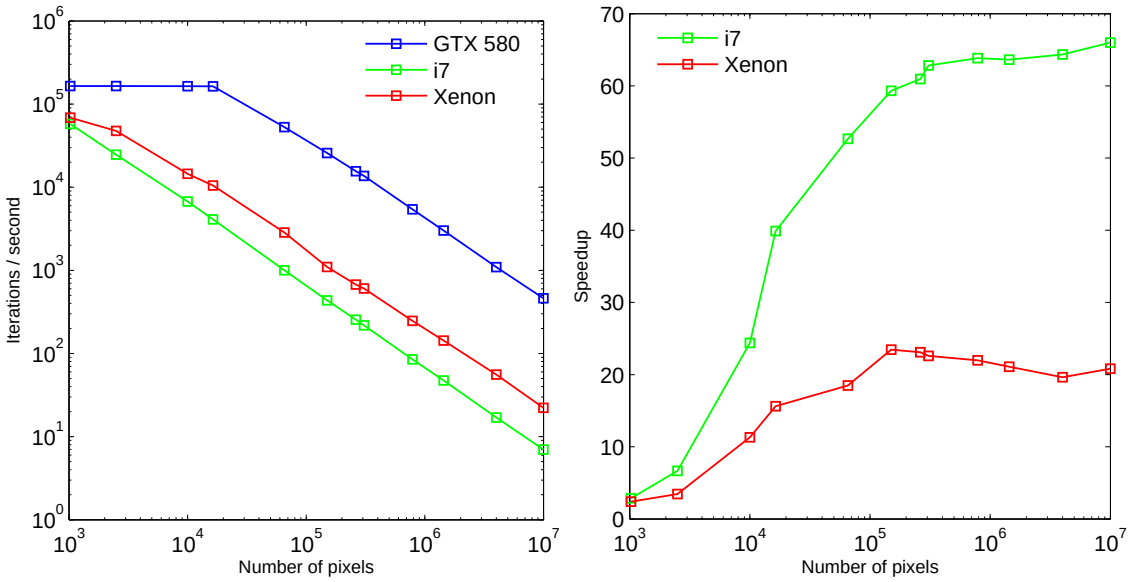


Figure 3.7: Comparison of number of iterations per second (on the left) and the speedup of the GPU (on the right) over the number of pixels. Note that the significant part of this plots approximately start at 10^5 pixels.

restrictions, the speedup for small images decreases. When considering images with a size of at least 128×128, we get a speedup of approximately 60 when comparing to the desktop CPU and approximately 20 when comparing to the server multi-CPU setup. As most practical imaging and segmentation problems easily exceed this size, we consider this numbers as the relevant speedup for the segmentation problem in Algorithm 8. This is especially true for interactive image segmentation.

3.3.2 Global relabeling for continuous optimization

In this section, we present an algorithm to solve the discrete as well as the continuous min cut/max flow problem. We make use of the properties of (near) binary labeling problems and introduce additional global relabeling steps. We show that using global relabeling significantly speeds up the segmentation problem in the discrete setting. Additionally, we also obtain speedups for the continuous segmentation that has a near binary solution. Although we restrict ourselves to the 2-label segmentation problem, the basic idea could be extended to the multi label case.

3.3.2.1 Motivation

Most algorithms that solve the discrete graph cut problem in Section 3.1.1 are highly specialized algorithms (e.g. [Boykov and Kolmogorov, 2004]). Instead, we showed that the general primal-dual algorithm of [Chambolle and Pock, 2010] can be used to solve the graph cut problem in Algorithm 7. This algorithm can easily deal with a wide range of non-smooth convex problems. We have shown with Algorithm 8 that the algorithm can also be used for the continuous max flow problem. When comparing the algorithms, the differences are only marginal.

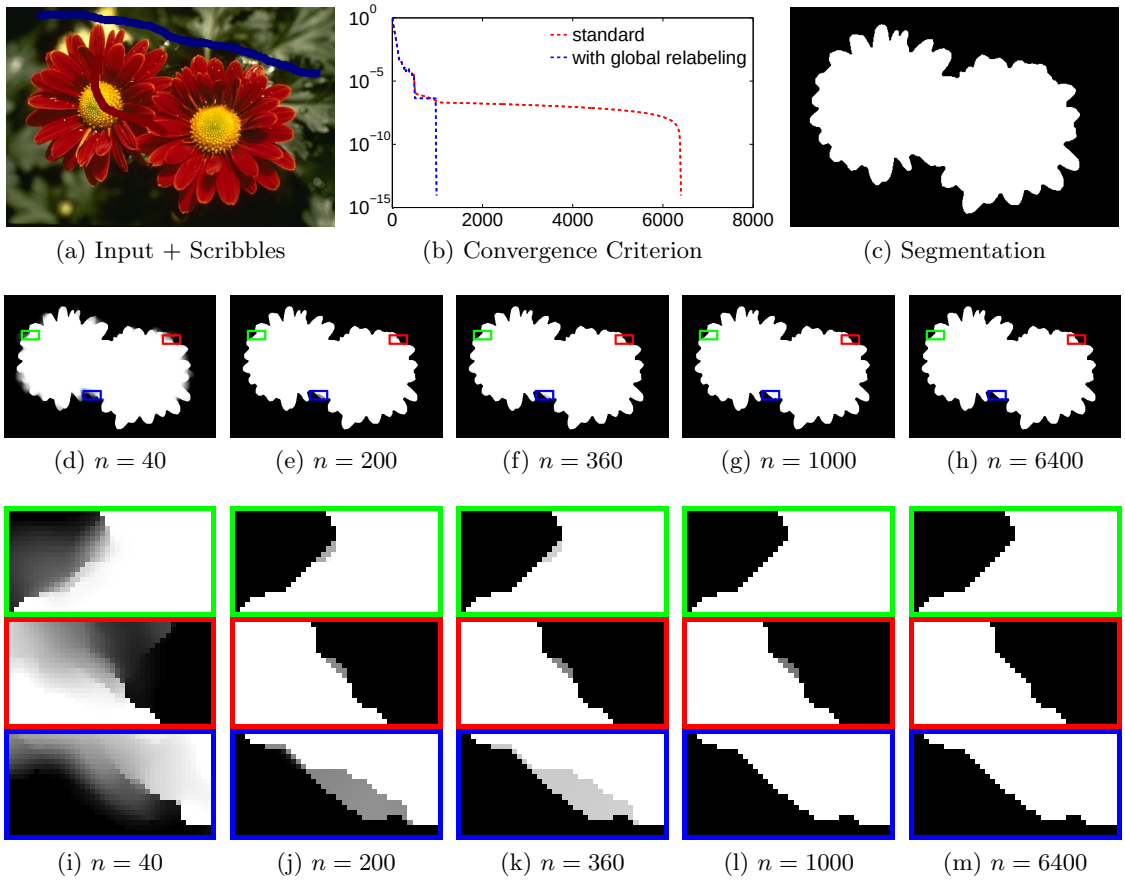
However, continuous optimization methods also have disadvantages. We observed that continuous methods are very fast in the beginning, but slow down as they get closer to the globally optimal solution. We illustrate this problem in Fig. 3.8. The key observation when watching the algorithm during optimization is as following: While the segmentation variable gets very close to the final segmentation in a few iterations, usually some small areas change their value very slowly over time. We therefore introduce global relabeling steps that speed up the convergence process by evaluating thresholded versions of the current segmentation variable. Instead of changing the value of an area only slowly, the global relabeling step results in a discrete labeling with lower energy in just a single iteration.

3.3.2.2 Algorithm

As the example in Figure 3.8 shows, the primal dual (pd) steps described in the previous section are very fast in the beginning, but often slow down as the result gets closer to the optimal solution. The main problem are small areas that change their value very slowly.

With the global relabeling (grl) step we want to assign this regions either the value 1 or 0. It is easy to identify this regions e.g. by over-segmentation or region labeling. One can then apply global relabeling to each region separately, all possible combinations or all regions at once. With the normalized primal dual gap (3.14) we have a meaningful optimality measure. We then simply take the global relabeling that minimizes the primal dual gap $G(u, \boldsymbol{p})$. Obviously this could lead to very complex and computationally expensive algorithms even when computed on the GPU.

To keep the grl step fast and efficient on parallel hardware, we simply compute the upper level sets of u by thresholding u several times in the range $(0, 1)$. We then compute



(a) Input + Scribbles (b) Convergence Criterion (c) Segmentation

(d) $n = 40$ (e) $n = 200$ (f) $n = 360$ (g) $n = 1000$ (h) $n = 6400$

(i) $n = 40$ (j) $n = 200$ (k) $n = 360$ (l) $n = 1000$ (m) $n = 6400$

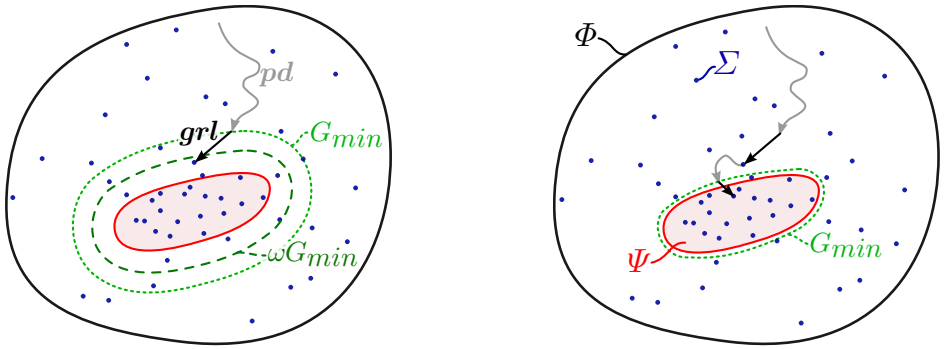
Figure 3.8: Standard continuous optimization methods are usually very fast in the beginning, but slow down later on. Note that the segmentation looks already good after 200 iterations (e, j). Nevertheless there are some small regions that are changing their value very slowly. It takes the standard algorithm 6400 iterations to fully converge (h,m). We propose to use global relabeling steps to speed up convergence. As can be seen in (b) the proposed algorithm converges significantly faster.

the best thresholded version $\tilde{u}$ and corresponding $\tilde{\boldsymbol{p}}$ as

$$(\tilde{u}, \tilde{\boldsymbol{p}}) = \arg \min_{\theta \in (0,1)} \{G(u_\theta, \boldsymbol{p}_\theta)\}, \tag{3.34}$$

where

$$(u_\theta)_{i,j} = \begin{cases} 1 & \text{if} \quad u_{i,j} > \theta \\ 0 & \text{else} \end{cases}, \tag{3.35}$$
$$\boldsymbol{p}_\theta = \Pi_{w_b} \left(\boldsymbol{p} + c \nabla u_\theta \right).$$



(a) First *grl* step (b) Second *grl* step

Figure 3.9: Illustration of the optimization schema. While the *pd* steps can move freely through the space of solutions, the *grl* step only allows jumps to binary solutions that are closer to the global optimum than the best solution so far.

Hence, u_θ represent thresholded versions of u. To obtain $\boldsymbol{p}_\theta$ the update equation is evaluated with a very large time step $c \gg 1$. If the solution $\tilde{u}$ is significantly closer to the global optimum than the current solution u, we accept the *grl* step, otherwise we continue optimization from the current solution u. Experiments showed that $\theta \in \{0, 0.01, 0.05, 0.1, 0.5, 0.9, 0.95, 0.99\}$ provides enough different thresholds and is reasonably fast.

We need a non-binary u for the thresholding to work. The iterative primal dual algorithm only takes into account its direct neighborhood with the ∇ operator. As a result information is propagated only slowly as a wavefront. We compute the *grl* step only every $J = \max(M, N)$ iterations. This ensures that information travels at least once through the image. We will investigate this choice in Section 3.3.2.3.

In Figure 3.9, we illustrated the overall primal dual algorithm with global relabeling *pdgrl*. We denote by $\phi = [0, 1]^{MN}$ the feasible set of the relaxed labeling vector and by $\Psi \subseteq \Phi$ the set of solutions (note that in general, graph cuts could have multiple solutions). Furthermore, let $\Sigma = \{0, 1\}^{MN}$ be the set of binary labeling vectors and hence $\Psi \cap \Sigma$ is the set of binary solutions. Starting from an arbitrary initialization, the *pd* steps will change the labeling vector according to (primal and dual) gradient information. This could also result in a temporary increase of the gap.

The *grl* step is only considered if the resulting gap is smaller than ωG_{min}. We added an additional multiplicative parameter to take into account only significant updates. The parameter $0 < \omega < 1$ allows a global relabeling only if G is significantly reduced. We

Algorithm 11 Primal dual algorithm with global relabeling for binary image segmentation

repeat
  for $1, \dots, J$ **do**
    $\boldsymbol{p}^{n+1} = \Pi_{w_b}\left(\boldsymbol{p}^n + \sigma\nabla\left(2u^n - \bar{u}^{n-1}\right)\right)$ // Primal Update
    $u^{n+1} = \left[u^n - \tau\left(\nabla^T\boldsymbol{p}^{n+1} + w_u\right)\right]_0^1$ // Dual Update
    $\bar{u}^j = 2u^j - u^{j-1}$ // Extra-gradient step
    $n = n + 1$
  end for
  $G_{min} = \min\left\{G_{min}, G(u^n, \boldsymbol{p}^n)\right\}$
  $(\tilde{u}, \tilde{\boldsymbol{p}}) = \arg\min_{\theta\in(0,1)}\left\{G(u_\theta, \boldsymbol{p}_\theta)\right\}$ // Thresholding
  if $G(\tilde{u}, \tilde{\boldsymbol{p}}) \leq \omega G_{min}$ **then**
    $(u^n, \boldsymbol{p}^n) = (\tilde{u}, \tilde{\boldsymbol{p}})$ // Global Relabeling
  end if
until $G(u^n, \boldsymbol{p}^n) \leq tol$

used $\omega = 0.5$ throughout the thesis. Algorithm 11 summarizes the proposed primal dual algorithm with global relabeling. Note that for the graph cut the re-projection $\Pi_{w_b}(\boldsymbol{p})$ is a simple clamping to the interval $[-w_b, w_b]$, and in case of the TV formulation, an orthogonal projection to a L^2-ball of radius w_b.

$$\begin{aligned}\Pi_{w_b}^{GC}(\boldsymbol{p}) &= [\boldsymbol{p}]_{-w_b}^{w_b}\,,\\ \Pi_{w_b}^{TV}(\boldsymbol{p}) &= \frac{\boldsymbol{p}}{\max\left\{w_b, |\boldsymbol{p}|\right\}}\,.\end{aligned} \tag{3.36}$$

Convergence of the algorithm follows from the fact that both steps, the primal-dual optimization *pd* and the global relabeling *grl* are guaranteed to decrease the gap. In fact, we allow a *grl* step only if the new gap G after *grl* is smaller than the minimal gap G_{min} obtained by *pd* so far. In [Chambolle and Pock, 2010], it is shown that the *pd* algorithm decreases the primal-dual gap with a sublinear rate of $O(1/N)$ where N is the total number of iterations. Although the proposed global relabeling does not change this estimate, it empirically gives a super-linear convergence close to the optimal solution.

The convergence criterion *tol* was chosen as following: When using 64bit double precision (Matlab) numerical accuracy is reached with a normalized primal dual gap $G = 10^{-14}$. For the 32bit float precision (CUDA) numerical accuracy is already reached with $G = 10^{-7}$. In case of the TV segmentation model we set $tol = 5 \cdot 10^{-4}$ as the segmentation did not show any visible changes.

3.3.2.3 Experimental results

Experiments were conducted on a Intel Core i7 960 with 12 GB available memory and a NVidia GeForce GTX 480 with 1.5 GB available memory. The segmentation framework was implemented in Matlab. The actual algorithms were additionally implemented on the GPU using the CUDA framework using the principles detailed in Section 3.3.1. Measured

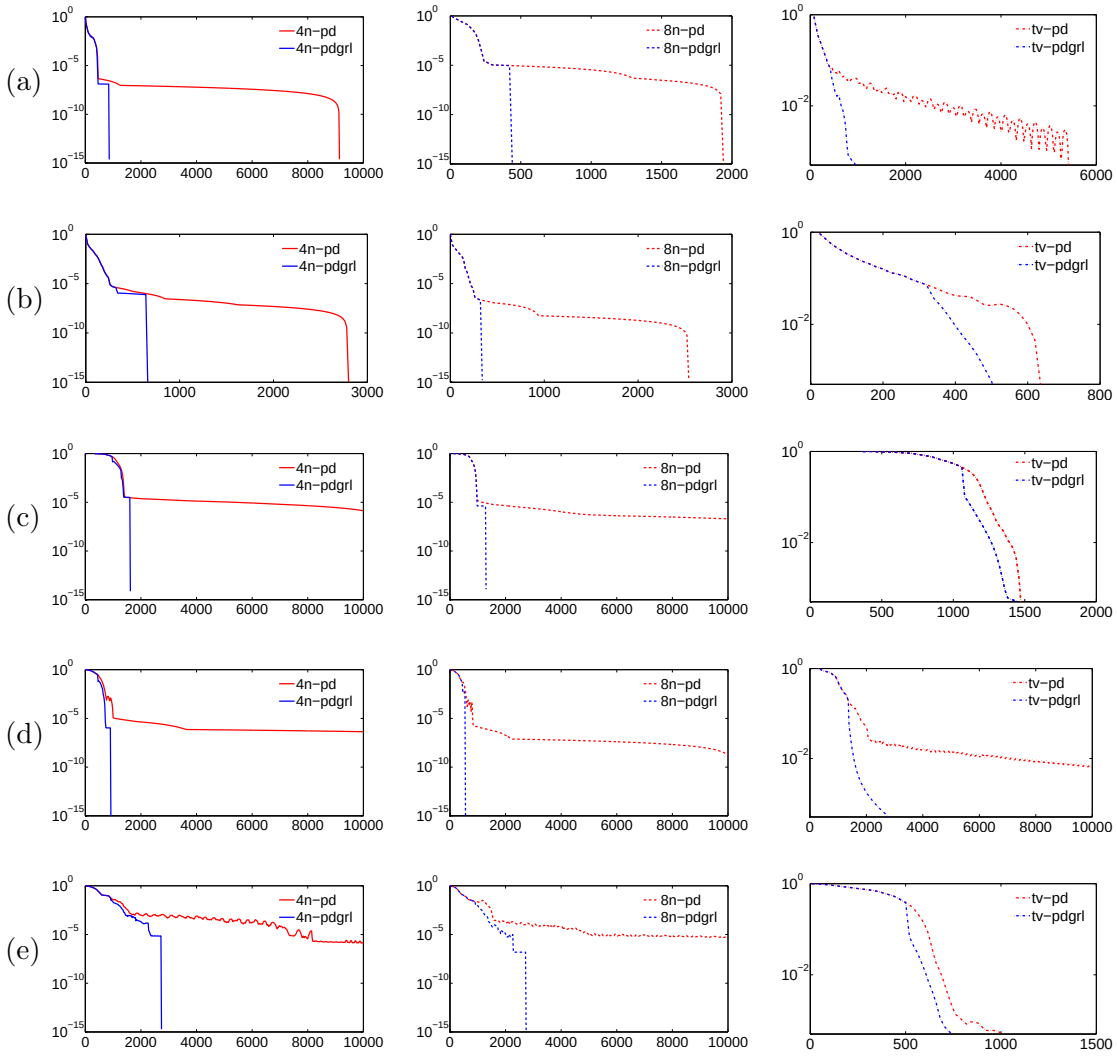
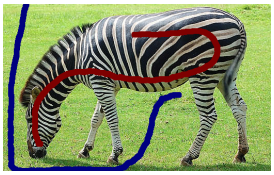
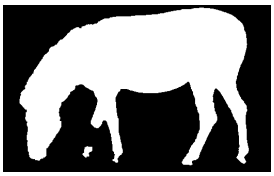


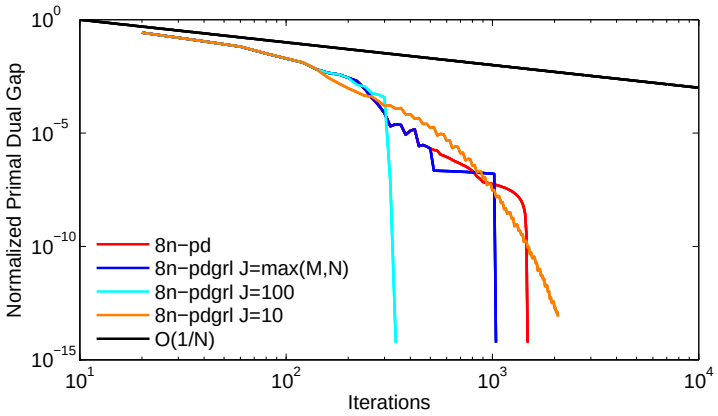
Figure 3.10: Comparison of the *pd* algorithm and the primal-dual algorithm with global relabeling (*pdgrl*) for different segmentation problems. From left to right: 4-connected graph cut, 8-connected graph cut, TV based segmentation. For the corresponding input images see Figure 3.12.



(a) Input



(b) Segmentation



(c) Convergence depending on J

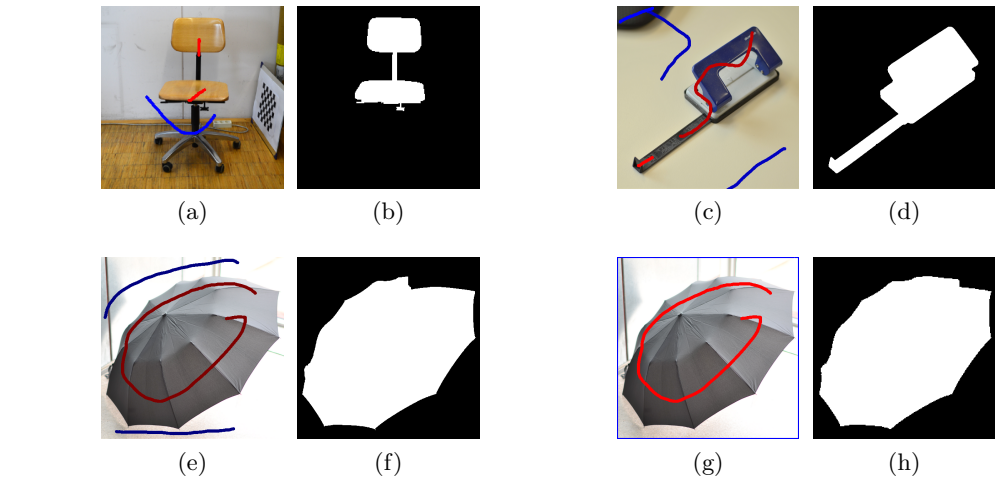
Figure 3.11: Demonstration on the effect of different global relabeling intervals J. While the chosen intervals (blue) are faster than the standard *pd* algorithm, $J = 100$ (cyan) would speed up convergence even more. If global relabeling is done too often, e.g. $J = 10$ (orange), the algorithm might become slower.

times do not include any transfer times between CPU and GPU (the approximate overhead ranges from 10 ms for images of size 256×256 to 700 ms for images of size 3200×3200).

To calculate the unary terms w_u, we use two kind of scribbles as detailed in Section 4.1. First, we can directly draw sparse source and sink seeds. Second, we can draw scribbles that will be used to build color histograms. The binary terms w_b are calculated according to Section 4.1.3.2.

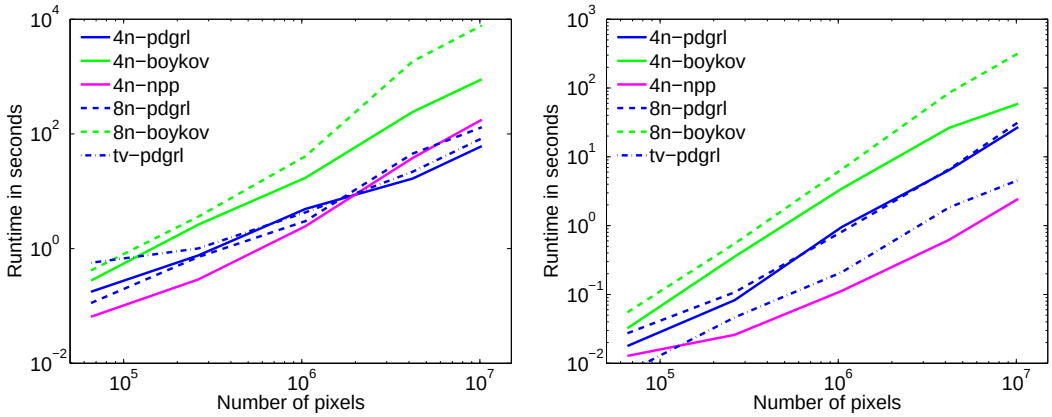
In Figure 3.10, we show experiments on typical segmentation problems using the Matlab implementation. The corresponding input images and resulting segmentations can be found in Figure 3.12. For (a,b) we used color information while for (c,d,e) the algorithm relies solely on seed regions (only edge information is used). It shows that the global relabeling steps (the *pdgrl* algorithm is depicted in blue) converges significantly faster than the *pd* algorithm alone. This is true for the graph cut model as well as the TV model. Note that for the *pd* algorithm, some experiments did not converge within the 10000 iterations allowed for this experiment.

Figure 3.12: Input images and segmentation results used in the comparison of the *pd* algorithm and the primal-dual algorithm with global relabeling (*pdgrl*). See Figure 3.10 for the algorithmic results. From left to right: The input image, 4-connected graph cut, 8-connected graph cut, TV based segmentation.



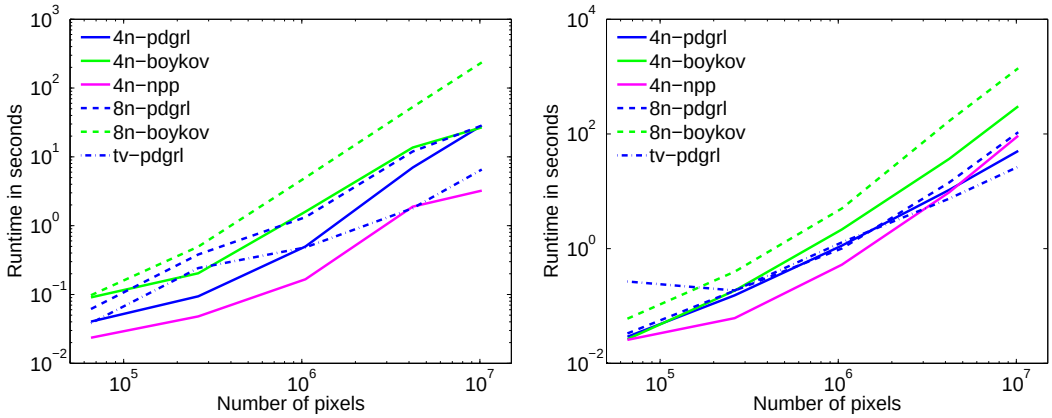
(a) (b) (c) (d)

(e) (f) (g) (h)



(i) Image (a) with seeds only

(j) Image (c) with seeds and color



(k) Image (e) using color only

(l) Image (g) with seeds only

Figure 3.13: Evaluation of the influence of the image size to the runtime of different algorithms. A full table with runtimes can be found in Appendix A.1.

To motivate the choice of J (the iteration interval after which global relabeling is performed), we conducted experiments with varying J in Figure 3.11 on an image of size 519×324. In Section 3.3.2.2, we choose $J = \max(M, N)$. As one can see from Figure 3.11, that this choice is rather conservative, as e.g. $J = 100$ would result in a much faster convergence of the algorithm. On the other hand $J = 10$ would result in much slower convergence, as the primal dual steps need some time to provide a meaningful direction. Setting $J = \max(M, N)$ gives the primal dual steps the chance to propagate information through the whole image before the next global relabeling is performed. Note that when using $J = \max(M, N)$, during our large number of experiments, the global relabeling steps never resulted into slower convergence. On the other hand, Figure 3.11 shows that there might be better strategies on when to perform global relabeling.

In Figure 3.13, we compared the proposed *pdgrl* algorithm with the *pd* algorithm on the GPU and a CPU implementation of [Boykov and Kolmogorov, 2004] (denoted as *boykov*), that is one of the most used graph cut implementations to date. Additionally we compare to the NPP library [NVidia, 2010] graph cut implementation (*npp*), that is to our knowledge currently the fastest graph cut implementation on a GPU. Note that the *npp* implementation only works for $4n$. ¶ We conducted the experiment for 4 different quadratic images that were scaled to 256, 512, 1024, 2048 and 3200 edge length, thus ranging from approximately $6 \cdot 10^4$ to 10^7 pixels. All algorithms have an approximately linear runtime behaviour. With the proposed algorithm most of the time a bit slower than the *npp* implementation for *4n*. The slowest algorithm is always the *boykov* CPU implementation with *8n*. Note that for the runtime of the *pdgrl* there is not much difference for the graph cut with *4n* and *8n*, and the *tv* model.

3.3.3 Binary segmentation using the ROF model

Let us now reconsider the ROF model introduced in Section 2.2.1.2 (2.9)

$$\min_u \left\{ \int_\Omega |\nabla u| + \frac{\lambda}{2} \int_\Omega (u - f)^2 dx \right\}. \tag{3.37}$$

As shown in [Chambolle, 2005; Chambolle and Darbon, 2009], with a solution u of (3.37) and $z \in \mathbb{R}$, the super-level set $E_z = \{u \geq z\}$ is a minimizer of

$$\min_{u \subseteq \Omega} \left\{ \int_\Omega |\nabla u| + \lambda \int_E (z - f) dx \right\}. \tag{3.38}$$

¶As of 29.09.2012 there is also a $8n$ version available.

Figure 3.14: Example of image segmentation using the weighted ROF model. From left to right: The segmentation result using the continuous maximum flow algorithm (Algorithm 8). The unary potentials $-c_u$ depicted in the interval $[-0.05,\ 0.05]$. The binary potentials c_b. The denoising result u. Finally, the segmentation obtained using the ROF model $u > 0$.

As a consequence the zero super-level set $E = \{u > 0\}$ of the minimizer of the ROF model is also a minimizer of maximum flow segmentation model as in (3.7). This connection was also discussed by [Berkels, 2010], in the context of Mumford Shah segmentation. A first example was already given in Figure 2.4, where we showed that the thresholded ROF model is visually equivalent to the shape denoising problem.

Of course the equivalence of (3.37) and (3.38) also holds when dealing with the weighted TV. The fully equivalent segmentation problem using the ROF model can then be written as

$$\min_u \left\{ \int_\Omega c_b |\nabla u| + \frac{1}{2} \int_\Omega (u - c_u)^2 dx \right\}. \tag{3.39}$$

The advantage of treating segmentation as ROF denoising, is the possibility to apply the accelerated algorithm as done in Algorithm 4. As we will see in the evaluation of the next Section this speedup is significant, making the weighted ROF model perfectly suited for realtime image segmentation. Using the accelerated primal dual algorithm of [Chambolle and Pock, 2010] on the weighted ROF model in (3.39), results in Algorithm 12.

In Figure 3.14, we show an additional example that demonstrates the segmentation using the weighted ROF model. The solution u is simply a denoisied version of the unary constraints. We see that the zero super-level set delivers exactly the same solution as the continuous maximum flow algorithm.

Algorithm 12 Accelerated primal dual algorithm to solve the weighted ROF model (3.39) for image segmentation.

$\tau = \sigma = \frac{1}{\sqrt{8}}$
$u^0 = c_u$, $\boldsymbol{p}^0 = 0$ and $\bar{u}^0 = u^0$ // Initialization
for $j = 1$ to J **do**
  $\tilde{\boldsymbol{p}} = \boldsymbol{p}^{j-1} + \sigma \nabla \bar{u}^{j-1}$ // Update dual variable
  $\boldsymbol{p}^j = \frac{\tilde{\boldsymbol{p}}}{\max(w_b, |\tilde{\boldsymbol{p}}|)}$
  $u = (1+\tau)^{-1}(u^{j-1} + \tau \mathrm{div}\boldsymbol{p}^j + \tau w_u)$ // Update primal variable
  $\theta^j = 1/\sqrt{1+0.7\tau^{j-1}}$
  $\tau^j = \theta^j \tau^{j-1}$ and $\sigma^j = \sigma^{j-1}/\theta^j$ // Update time steps
  $\bar{u}^j = u^j + \theta^j \left(u^j - u^{j-1}\right)$ // Extra-gradient step
end for

3.3.4 Comparison of continuous binary image segmentation algorithms

In this section, we compare the algorithms to solve the continuous max flow problem introduced earlier in this chapter in terms of performance. As the primal-dual gap is a problem specific measure, it is not suited for comparing different problem formulations. Moreover, the segmentation problems presented are not strictly convex. As a consequence results might slightly differ when using different algorithms or initializations.

We overcome this problems, by first letting each algorithm run several hundred thousand iterations to make sure it is converged. This first run is considered the ground truth for the current segmentation problem. In a second step, the algorithm is run once more, with the thresholded result compared to the ground truth every 20 iterations. As soon as the results are identical, the algorithm is considered to be converged.

In order to be able to include the weighted TV-L1 shape denoising model as described in Section 2.2.2, we have to make some considerations on the unary terms. While the maximum flow approaches allow continuous constraints, the weighted TV-L1 model measures the difference of the segmentation $u \in \{0, 1\}$ to some input f. In order to make comparison possible, we simply restricted ourselves to unary terms that either vote for foreground, background or nothing at all. We therefore refer to this experiment as the one with discrete unary terms. See Section 3.1.3 for more details on the relationship between shape denoising and continuous max flows.

Using the discrete constraints, we have an experimental setup as following. We compare the shape denoising methods using the weighted TV-L1 model. Algorithm 6 with dualized data term is denoted by 'TV-L1' and Algorithm 5 with thresholding schema is denoted as 'TV-L1 threshold'. Additionally, we use the continuous maximum flow algo-

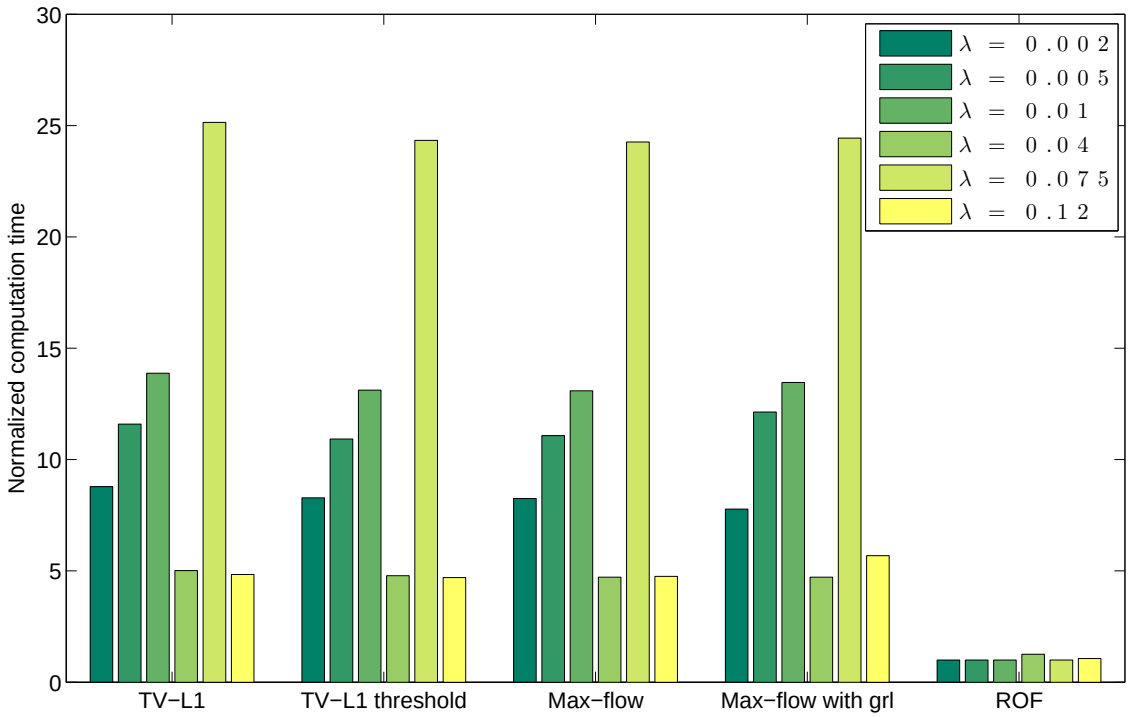


Figure 3.15: Comparison of the runtime of different algorithms for two label image segmentation, with discrete unary terms. The results are averaged over 8 different images, and normalized by the fastest performing method.

rithm in Algorithm 8 denoted as 'Max-flow' as well as the primal dual algorithm with global relabeling Algorithm 11 denoted as 'Max-flow with grl'. Finally, the weighted ROF model in Algorithm 12 is denoted as 'ROF'.

We evaluated the algorithm on 8 different images and for 6 different values of λ using the comparison to a precalculated ground truth as stated above. Each experiment was then normalized by the fastest approach, and the average over all images was computed to obtain a mean computation time. The result is depicted in Figure 3.15. We clearly note, that the weighted ROF model is the fastest approach in this comparison. All other approaches are approximately 10 times slower. There is hardly any speedup by the global relabeling step for the continuous maximum flow algorithm. We made this observation already in Section 3.3.2.3, where we saw a significant speedup for the discrete but not for the continuous model. The weighted TV-L1 model with dualized dataterm is the slowest one in this test, with the algorithm using the thresholding schema slightly faster.

Another observation we can make from Figure 3.15, is the fact that the speedup of the ROF model generally increases with increasing λ. Note that a higher λ means less

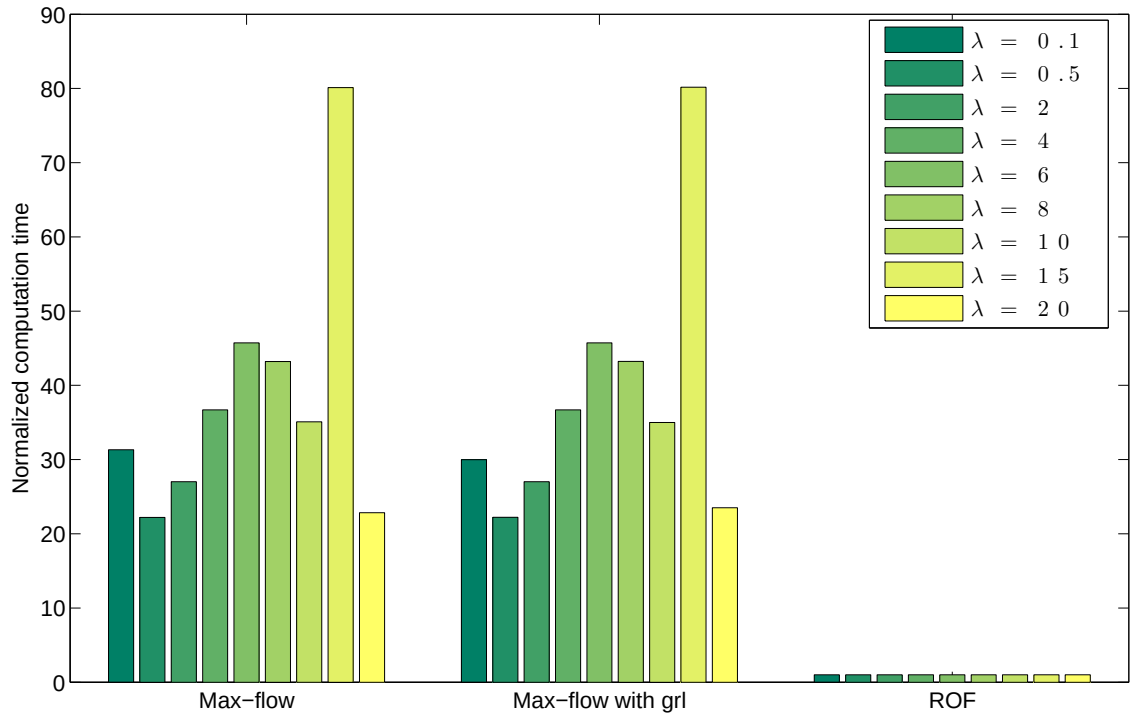


Figure 3.16: Comparison of the runtime of different algorithms for two label image segmentation, with continuous unary terms. The results are averaged over 13 different images, and normalized by the fastest performing method.

regularization and shorter runtimes. When looking at the full data in Table A.2, a factor of 10 or more is typical when varying λ in the range $[0.002, 0.12]$. We also observed some outliers from this trend ($\lambda = 0.04$ and $\lambda = 0.12$), for which we have no explanation.

In case of using continuously varying unary potentials (the normal case), we skipped the models based on the weighted TV-L1 model. We evaluated the remaining algorithms on 10 different images and for 9 different values of λ. The result is depicted in Figure 3.16. Again, the weighted ROF model is significantly faster than the continuous maximum flow algorithms. Generally a speedup of 30 can be achieved.

This makes the algorithm using the weighted ROF model (Algorithm 12) typically the best choice for interactive segmentation methods. The fast minimization of the ROF model is owed to the time step scheme of the accelerated primal dual algorithm of [Chambolle and Pock, 2010] (Algorithm 2).

Full tables with all experiments conducted in this Section can be found in Appendix A.2.

Practical considerations: While comparing to a ground truth result is a valid choice for evaluation purposes, one does not have this information in practical applications. In this case we again have to rely on the primal-dual gap as a convergence criterion as detailed in the previous sections.

All of the above methods perform optimization until we reached the globally optimum up to machine precision. While this is generally desirable, in practice a solution close to the global optimum may suffice. As long as we know (e.g. by the primal-dual gap) that we are in a reasonable range to the global optimum, the algorithm can be terminated much earlier with only minor impact. We can further justify an early termination of the algorithm, as most of the time, the proposed model together with unary and pairwise potentials cannot accurately describe the desired solution. This is especially true in the case of interactive segmentation, that we tackle in the next chapter.

Chapter 4

Supervised Segmentation

4.1 Interactive image segmentation

In the previous chapter, we introduced segmentation models for the binary as well as the multi-label case. We now focus on the application of these models to the actual task of interactive image segmentation.

4.1.1 Introduction

In Section 1.1, we demonstrated that image segmentation is a highly ambiguous task. As a consequence the actual algorithm has to be tailored for a very specific task using as much prior knowledge as possible. This can be done by completely unsupervised algorithms (examples will be given in Section 5), or using some form of supervision or interaction.

Ideally, an algorithm would precalculate all meaningful segmentations and present the most likely ones to the user for selection. Unfortunately, this approach is impractical and we have to rely on more guidance by the user. We first make out some important requirements for interactive segmentation:

- Speed:
  As discussed in the introduction we need fast algorithms to get immediate response. Using the algorithms presented in the previous section, we are able to accomplish this point.

- Efficiency:
  User interaction should be reduced to a minimum. This can be achieved both by the type of constraints as well as the models used to create unary and binary potentials.

Both of this points will be tackled in this Section.

- Predictability:
  Interacting with the application should be predictable, to allow for precise user input. Using convex methods, one is well aware of the desired objective function, and they are guaranteed to find the globally optimum. As a consequence the proposed methods are very predictable.

4.1.2 Related work

Providing information to the algorithm by direct user input has lead to a wide range of different approaches. Commercial software such as Adobe Photoshop or Gimp provide a wide range of different interaction tools. They typically fulfill all of the requirements mentioned earlier.

A still common way to obtain an outline of an object, is to simply draw the contour. This can be either done by exactly following the cursor, or by drawing line segments. While this method allows to perform very precise segmentations, it is quite cumbersome and time-consuming. One of the most frequently used improvements it to guide the border towards nearby edges. This idea was first suggested by [Mortensen and Barrett, 1995] and called "intelligent scissors". A similar approach in Photoshop is called the "magnetic lasso". Also the snake model [Kass et al., 1988] can be used to refine an approximately drawn contour. As a consequence, the user does not have to draw very precisely, resulting in a significant speedup.

Often gray value and color information is utilized to obtain a segmentation. Selecting a simple gray value or color model and thresholding is often used to select a number of pixels simultaneously. The threshold as well as the seed regions can be interactively defined by the user. When using seed regions the resulting segmentation can also be constrained to be spatially connected. Modern implementation, such as the "magic wand" in Photoshop, allow to further modify the contour and apply smoothing, dilation, matting and alpha blending.

The concept of seed regions was already used by [Adams and Bischof, 1994]. They suggested to use seeded region growing. In region growing approaches, the seeds are expanded as long as some homogeneity criterion (e.g. color consistency) is fulfilled. Apart from being fast, these methods have the advantage of working locally allowing to segment complex objects by successive brushes.

Drawing scribbles with different brushes can be used as input for a wide range of

subsequent algorithms. As we will see later, this is our preferred method of interaction. But also approximate outlines of the object can be used to learn models for more complex algorithms. In the "GrabCut" approach [Rother et al., 2004], a simple bounding box is drawn around the object. The learned color model is then used in a graph cut.

4.1.3 Creating potentials for segmentation

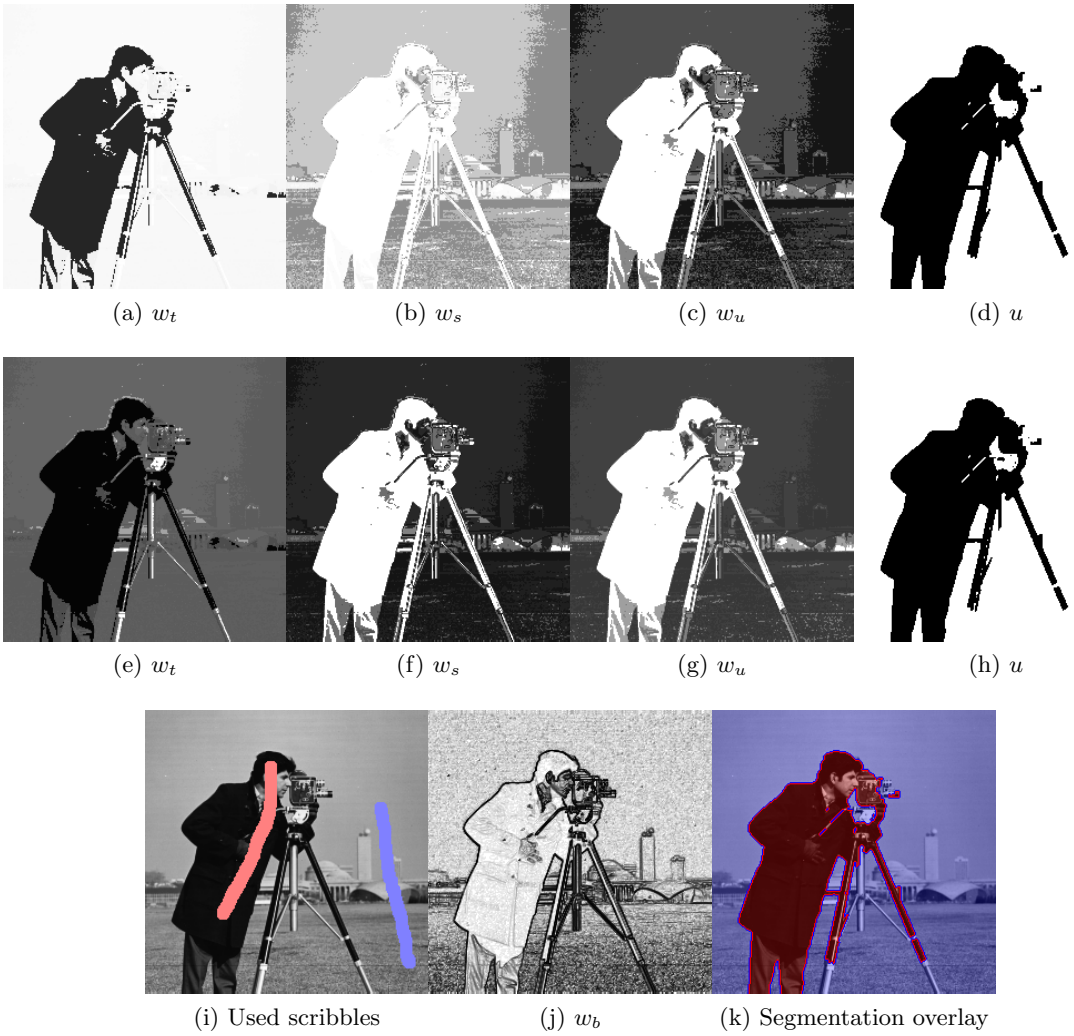
In the following we are going to discuss how the image as well as user input can be incorporated to the models presented in Section 3. We first review the unary potentials that are used for pixel wise (or feature based) information, and then binary potentials that contain edge information.

4.1.3.1 Unary potentials

The multi label approach in Section 3.2, uses f_i for the unary potentials assigned to label i. When recalling the definition of the multi label segmentation problem in (3.19), a small value f_i corresponds to a probability of the pixel belonging to label i. For the two-label model we denoted the discrete unary potentials for foreground and background with w_t and w_s. Additionally we simplified the model in (3.7) by using $w_u = w_t - w_s$, and discussed the influence of w_u in Section 3.1.2.1.

Both models are perfectly suited to use a pixel wise likelihood to create the unary potentials. Given some label probabilities p_i, we can simply set $f_i = 1 - p_i$. For the two label case we have to simply use $w_t = p_t$ and $w_s = p_s$. Another option is to use log-likelihoods and set $f_i = -\log(p_i)$. In Figure 4.1, unary potentials for both cases are depicted using a two-label example. Note, that although for both choices different values of λ had to be used, the resulting segmentations are very similar. In practice we observed, that both types of constructing the unary potentials deliver good results.

A very important task is to determine the pixel wise likelihoods p_i. This is typically evaluating a feature distribution over some user marked pixels. The choice of the right features has a great impact on the final segmentation quality. This reaches from simple gray values and color information to complex features like patches or HoG descriptors. Unfortunately, the more complex the features, the more difficult it is to efficiently learn and model distributions. Not every feature might work well for all images. Therefore, the algorithms to generate the unary terms, have to be sensitive to pick the right features. Learning methods, like random forests or a support vector machine (SVM), can be used model the label data. This is a very complex topic on its own. [Santner, 2010;



(a) w_t (b) w_s (c) w_u (d) u

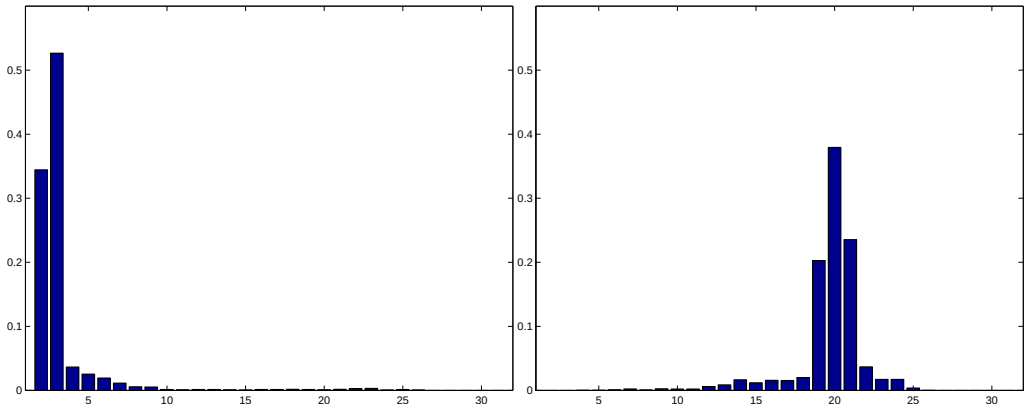
(e) w_t (f) w_s (g) w_u (h) u

(i) Used scribbles (j) w_b (k) Segmentation overlay

Figure 4.1: The top row depicts unary potentials and segmentation for a two-label problem using probabilities p_i directly ($\lambda = 0.1$). (a,b) are depicted in the range $[0, 1]$ while (c) is depicted in the range $[-0.5, 0.5]$. The middle row, shows the same results using log-likelihoods ($\lambda = 0.01$). (e,f) are depicted in the range $[0, 10]$ while (g) is depicted in the range $[-5, 5]$. We also used binary potentials depicted in (j), and user scribbles in (i).

Santner et al., 2011, 2009] provides extensive work on using various features for texture segmentation based on the models introduced in Section 3.

Instead of dealing with this rather complex methods, we focus on the low dimensional feature color to demonstrate the usage of the proposed segmentation algorithms. Color or gray value information is a strong feature to describe objects locally. Thus, it can be used to segment a wide range of natural images.



(a) Foreground histogram (b) Background histogram

Figure 4.2: Gray value histograms corresponding to the segmentation problem in Figure 4.1.

Based on some labeled pixels, a probability model has to be created for each label. Two common methods for color features, are gaussian mixture models (GMMs) and histograms. A gaussian mixture model (GMM) fits several Gaussians to the data, and thus allows for a compact and smoothed representation of the feature distribution. It was e.g. used by [Rother et al., 2004] for the GrabCut segmentation algorithm. While this works well for three dimensional data like color, it becomes increasingly costly in higher dimensions. The same holds for histograms, that we already used in [Unger et al., 2008b]. Smoothing is inherently provided by the bins, but can additionally be increased by Gaussian smoothing. An example of the corresponding gray value histograms for Figure 4.1 with 32 bins is given in Figure 4.2.

As already mentioned in Section 3.1.2.1, constraining pixels to a label can be achieved by setting $w_b = \infty$ or $w_b = -\infty$ in the two label case. Instead of ∞ a very large value can be used during implementation. As also mentioned earlier, it is better to simply fix the values of u in these areas.

4.1.3.2 Binary potentials

In the discrete setting, the weighted TV is defined as $\|W_b \nabla u\|_{2,1}$. The binary potentials (w_b or g), model the relationship between neighboring pixels. In a lot of cases, gray value or color discontinuities coincide with object boundaries. The binary potentials allow to incorporate this gradient information to the segmentation formulation. Note, that

at object boundaries we have $\nabla u = 1$ and $\nabla u = 0$ otherwise. As we are dealing with a minimization problem, a low value (or weight) of w_b will favor jumps in the labeling function u. We therefore have to design the binary potentials in a way that strong gradients (edges) result in a low value of w_b, and low gradients (flat areas) in a high value.

We compute the edge image $\boldsymbol{g} = (g_1, g_2, g_3, g_4)$ as the color gradient of the image similar to (3.4):

$$\boldsymbol{g} = \left(({}^{c+}_{x}v), (\delta^{c+}_{y}v), (\delta^{c++}_{xy}v), (\delta^{c+-}_{xy}v) \right)^T \ . \tag{4.1}$$

The color gradients simply compute the L^2-norm of the single channel gradients, that are otherwise computed as in (2.27) and (3.5). For 4-connected graphs we can then set the binary terms w_b as

$$(w_b)_{i,j} = \beta \left(e^{-\alpha|(g_1)_{i,j}|}, e^{-\alpha|(g_2)_{i,j}|} \right) \ , \tag{4.2}$$

with $\alpha > 0$, and for 8-connected graphs the binary terms become

$$(w_b)_{i,j} = \beta \left(e^{-\alpha|(g_1)_{i,j}|}, \ldots, e^{-\alpha|(g_4)_{i,j}|} \right) \ . \tag{4.3}$$

In case of the weighted TV, we set

$$(w_b)_{i,j} = (\bar{g}_{i,j}, \bar{g}_{i,j}) \, , \ \bar{g}_{i,j} = \beta e^{-\alpha\sqrt{(g_1)^2_{i,j} + (g_2)^2_{i,j}}} \ . \tag{4.4}$$

Meaningful values for α are in the range 0.6 - 1.2 and for β int the range 0 - 50. We typically use $\alpha = 0.8$ and $\alpha = 10$ as default values. Figure 4.1(j), shows an example of binary potentials computed with (4.4).

While simple gradients are typically sufficient, slight preprocessing can further enhance the binary potentials. This can either be a simple Gaussian filtering of the input image, or more complex steps like TV-L1 denoising. Figure 4.3, shows an example where these two methods are compared with no preprocessing. Note that the binary potentials are very noisy if no preprocessing is applied. In case of the Gaussian filtering boundaries are enhanced, but the edges are blurred. The TV-L1 denoising gives very precise boundaries, while eliminating small noise. While results are very similar, the preprocessing using the TV-L1 model results in the most exact segmentation. We therefore use slight TV-L1 denoising before calculating the binary potentials, for most practical applications.

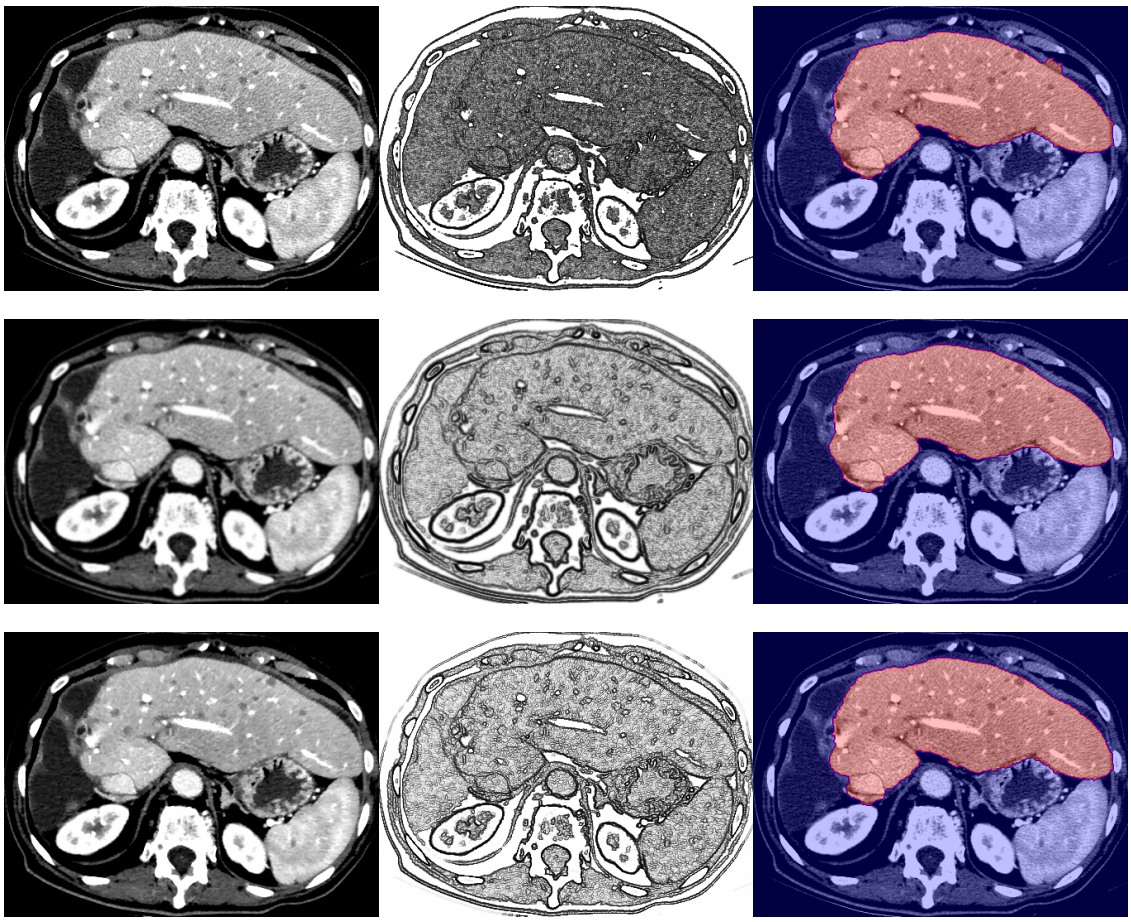


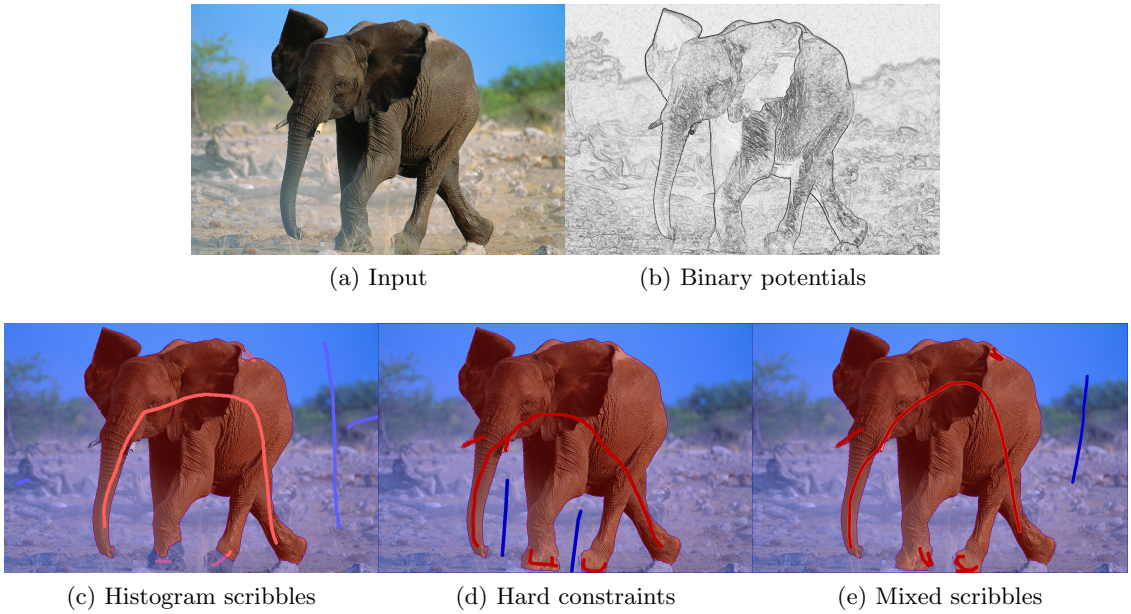
Figure 4.3: From left to right: The preprocessed input image, the corresponding binary potentials and the resulting segmentation overlay. From top to bottom: No preprocessing, Gaussian filtering and TV-L1 denoising.

4.1.4 Interaction

In order to generate the unary potentials as described in Section 4.1.3.1, user input or some other kind of prior knowledge is required. We suggest to use scribbles as a user input. They are easy to understand, allow great flexibility and are simple to implement. We provide two kinds of scribbles:

- Histogram scribbles: Pixels assigned to this brush are used to build the color histogram for each label.

- Hard constraints: Pixels assigned to this brush are set to the assigned label.



(a) Input (b) Binary potentials

(c) Histogram scribbles (d) Hard constraints (e) Mixed scribbles

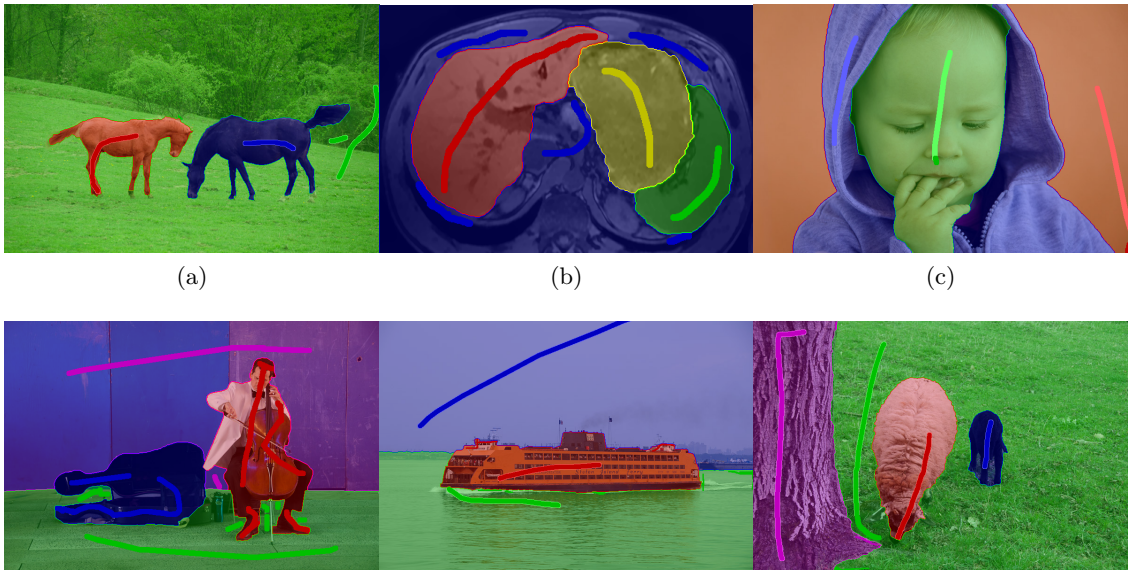
Figure 4.4: Different kinds of scribbles.

Of course both scribbles can be combined and drawn simultaneously. In addition to the scribbles we allow to set the border of the image to a certain label.

Figure 4.4, shows a comparison in the usage of the different scribbles. Histogram scribbles are shown in a bright color, while hard constraints are shown in a dark color. Although the histogram scribbles alone can perform well, they are sometimes not sufficient. This is the case if color alone is not a meaningful feature to describe the layers (e.g. the legs of the elephant). A great advantage of the histogram based scribbles is that the user does not need to draw very exact, as the scribbles are only used to build the color model, and can thus be assigned different labels by the algorithm.

Using hard constraints, as in Figure 4.4(d), we can specifically assign certain pixels to a label, which allows to obtain any desired segmentation. In this case the border was set to the blue label. Using only hard constraint scribbles will not take into account any color information and the final result will only depend on the binary potentials. This is useful when objects can't be described by a simple feature distribution, as e.g. is the case in magnetic resonance imaging (see Figure 4.5(b)).

Finally, as shown in Figure 4.4(e), both scribbles can be combined by drawing both constraints simultaneously or simply mixing them. This is the most common usage scenario. Typically, we first draw a few scribbles with both types of scribbles simultaneously,



(a) (b) (c)

(d) (e) (f)

Figure 4.5: Examples of interactive multi-label segmentation utilizing user input by scribbles.

and then fix the remaining errors by using hard constraints. In Figure 4.5, we show several additional examples using the scribbles for interactive multi label segmentation.

Of course other type of user input can be incorporated. Instead of scribbles, one could use a simple bounding box to mark the desired region. An example is shown in Figure 4.6. Here both approaches deliver nearly the same result. For the bounding box approach, the region inside the rectangle was used to learn the foreground model, and the whole image was used to learn the background model. Additionally, everything outside the rectangle was set as background.

The bounding box approach might sometimes be more efficient, as it only involves to draw one rectangle around the desired segmentation region. On the other hand it has several disadvantages: First, it does not offer the flexibility of the scribbles approach, and the resulting color models are not as exact. Second, it is typically only usable in the two-label case, and cannot be directly applied to multi-label segmentation problems.

Another way of user input is to directly modify the binary potentials. In [Unger et al., 2008a], we showed how edge information can be modified by an additional brush. There are some cases, where deleting edges may lead to the desired segmentation immediately. Typically, the same effect can be obtained using the hard constraint scribbles, why we

(a) Mixed scribbles (b) Bounding box

Figure 4.6: Comparison of segmentation by scribbles and a simple bounding box.

suggest only to provide brushes to modify the unary potentials.

4.2 Tracking by segmentation

In this Section, we present an application of the continuous maximum flow algorithms presented in Section 3.1. We will do this by extending the segmentation model to the spatio-temporal domain and apply it to tracking objects in videos.

4.2.1 Introduction

Robust visual object tracking is a vital topic in computer vision. The need for handling variations of the objects appearance, changes in shape and occlusions makes it a challenging task. Additionally, robust tracking algorithms should be able to deal with cluttered and varying background and illumination variations. We formulate the tracking problem as globally optimal segmentation of an object in the spatio-temporal volume. Under the assumption that an object undergoes only small geometric and appearance changes between two consecutive frames, the object is represented as a connected volume containing similar content. Applying the segmentation algorithm on a volume instead of single frames, enhances robustness in the case of partial occlusions and similar background. Furthermore, no explicit shape model has to be learned in advance. Instead spatial and temporal consistency is enforced by a single regularization term.

4.2.1.1 Previous work on tracking

In the following we give a very brief review on common tracking approaches. We start with simple frame based trackers.

The kernel-based method of [Avidan, 2005] considered tracking as a binary classification problem on the pixel level. An ensemble of weak classifiers is trained online to distinguish between object and background, while a subsequent mean-shift procedure [Comaniciu et al., 2000] obtains the exact object localization. The patch-based method of [Grabner et al., 2006] proposed online AdaBoost for feature selection, where the object representation is trained online with respect to the current background. Although these methods have shown their robust tracking behavior in several applications, they lack an explicit representation of the objects shape, due to their representation by a simple rectangular or elliptical region.

Shape-based [Donoser and Bischof, 2006] or contour based [Isard and Blake, 1996] tracking methods deliver additional information about the object state or enhance the tracking performance on cluttered background. While [Donoser and Bischof, 2006] used MSER regions for tracking, [Isard and Blake, 1996] applied the CONDENSATION algorithm on edge information. Therefore feature extraction or segmentation were independent from the tracking framework.

In contrast, especially level-set methods support the unified approach of tracking and segmentation in one system. A detailed review on the use of level set segmentation is given in [Cremers et al., 2007]. [Yilmaz et al., 2004] modeled object appearance using color and texture information while a shape prior is given by level sets. [Fussenegger et al., 2009] incorporated an active shape model based on an incremental principal component analysis (PCA), which allowed the online adoption of the shape models. [Yilmaz, 2007] extended the mean-shift procedure by [Comaniciu et al., 2000], by applying fixed asymmetric kernels to estimate translation, scale and rotation. [Bibby and Reid, 2008] proposed an approach, where they used pixel-wise posterior instead of likelihoods in a narrow band level set framework for robust visual tracking. The use of pixel-wise posterior led to sharper extrema of the cost function, while the GPU based narrow band level set implementation achieved real-time performance.

All of the above approaches work on single frames. [Mansouri et al., 2003] proposed a joint space-time segmentation algorithm based on level sets. The main idea of interpreting tracking as segmentation in a spatio-temporal volume is closely related to the approach presented in this paper. In contrary to our approach level set methods are used, that can

easily get stuck in local minima.

The idea of adding temporal information was already used by [Porikli, 2001], [Brox et al., 2010] and [Cremers, 2003, 2007]. Often this is done using optical flow, as optical flow already provides tracking on a pixel level. In the recent work of [Ochs and Brox, 2011, 2012] flow trajectories are estimated. Based on these trajectories, clustering is performed. The results is a fully automatic segmentation of the whole spatio-temporal volume (or video).

4.2.1.2 Tracking as segmentation in a spatio-temporal volume

In the following we motivate this Section by providing details on the concept of interpreting tracking as the segmentation of a 3D volume. As already stated, a color image I is defined in the 2D image domain Ω as $I : \Omega \to \mathbb{R}^3$. The 2D frames of a video sequence can be viewed as a volume by interpreting the temporal domain T as the third dimension. Thus the volume is defined as $V : (\Omega \times T) \to \mathbb{R}^3$. This makes it possible to incorporate spatial and temporal regularization in an unified framework. If we assume a high enough sampling rate, adjoining frames will contain similar content. The 2D objects of a single frame I correspond to cuts of planes with the 3D object defined in the volume V. Inherently this approach extends the forward propagation of information through time by additional backward propagation. Objects that are represented as disjoint regions in a single frame, correspond to a single volume, and are therefore tracked robustly.

This concept is illustrated with an artificial example in Figure 4.7. Our tracking approach is compared to an MSER tracker [Donoser and Bischof, 2006], that cannot handle multiple disjoint regions. The volumetric approach does not suffer from such a shortcoming, as the regions are connected in the volume.

4.2.2 Algorithm

We now formulate the basic idea of spatio-temporal segmentation, by extending the continuous max flow problem (3.7). Therefore, we first discuss the new segmentation model and its optimization. The complete tracking algorithm is then discussed in Section 4.2.3.

The segmentation model: While the segmentation problem in (3.7), easily generalizes to higher dimensions, we will take a look at an anisotropic formulation. We propose to

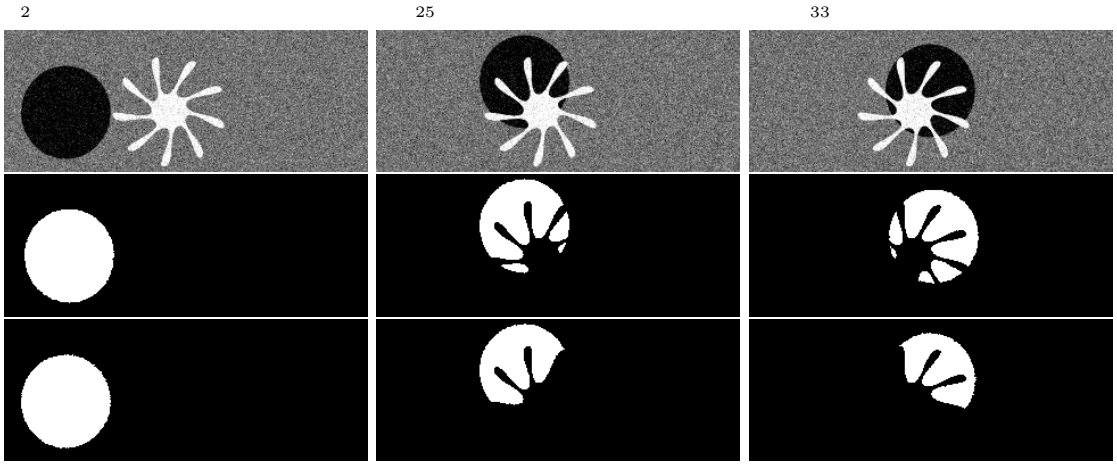


Figure 4.7: Tracking of an artificial object. The first row depicts frames of the input video with frame numbers at the top. The second row shows the segmentation result using the volumetric approach. The third row shows the result of an MSER tracker implementation [Donoser and Bischof, 2006].

use the following spatio-temporal extension:

$$\min_u \left\{ E_p = \int_{\Omega \times T} c_{bx} |\nabla_{\boldsymbol{x}} u| + c_{bt} |\nabla_t u| + \lambda \int_{\Omega \times T} c_u u \; d\boldsymbol{x} dt \right\} . \tag{4.5}$$

The first term now features the anisotropic TV. While in the spatial domain Ω the TV computation stays the same $|\nabla_{\boldsymbol{x}} u| = \sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2}$, an additional $|\nabla_t u| = \left|\frac{\partial u}{\partial z}\right|$ in the temporal domain T is added. Edge information is incorporated using two separate binary terms $c_{bx} : (\Omega \times T) \to \mathbb{R}$ and $c_{bt} : (\Omega \times T) \to \mathbb{R}$ that subsequently represent edges in the current frame and edges from one to the next frame. The binary potential is computed as $c_{bx} = \exp\left(-a \left|\nabla_{\boldsymbol{x}} V\right|^b\right)$. Likewise one can compute $c_{bt} = \exp\left(-a \left|\nabla_t V\right|^b\right)$.

The usage of an anisotropic weighted TV norm for regularization has the advantage that discontinuities in the spatial domain V and in the time domain T are separated. This allows for a more accurate segmentation of small and fast moving objects. To illustrate this, Figure 4.8 shows a comparison of the regularization term as used in (4.5), and the standard weighted TV as in (3.7). It shows that the anisotropic regularization delivers finer details during fast moving parts of the video.

Solving the minimization problem: To solve the energy defined in (4.5), we again use duality by introducing the dual variable $\boldsymbol{p} : \Omega \times T \to \mathbb{R}^3$. The dual variable can be separated into a spatial and a temporal component $\boldsymbol{p} = (\boldsymbol{p_x}, p_t)^T$. Similar to (3.9), we get

Figure 4.8: Comparison of anisotropic TV and standard TV regularization. The first row shows frames of the original video where the left player is tracked. In the second row the anisotropic regularization term shows a better segmentation of fast moving details than the standard weighted TV regularization in the third row.

the following discretized primal-dual formulation of the spatio-temporal segmentation:

$$\begin{aligned} &\min_{u} \max_{\boldsymbol{p}} \left\{ \langle \nabla u, \boldsymbol{p} \rangle + \langle u, w_u \rangle \right\}, \\ \text{s.t. } u \in [0,1],\ &||\boldsymbol{p_x}||_\infty \leq w_{bx},\ ||\boldsymbol{p_t}||_\infty \leq w_{bt}\ . \end{aligned} \tag{4.6}$$

The application of the primal dual algorithm in Algorithm 1 is straightforward, as it directly follows the application to the continuous max flow problem. Only the reprojection of the dual variables changes. We can formulate this by using the convex set $C = \left\{ \boldsymbol{q} = (\boldsymbol{q_x}, q_t)^T : |\boldsymbol{q_x}| \leq c_{bx}, |q_t| \leq c_{bt} \right\}$ denoting a cylinder centered at the origin with the radius c_{bx} and height $2c_{bt}$. The re-projection onto C can then be formulated as

$$\Pi_C(\boldsymbol{q}) = \left(\frac{\boldsymbol{q_x}}{\max\left\{ c_{bx}, |\boldsymbol{q_x}| \right\}}, [q_t]_{-c_{bt}}^{c_{bt}} \right)^T \tag{4.7}$$

We summarized the resulting algorithm in Algorithm 13.

For a convergence criterion based on the primal-dual gap, we have to find a pure dual

Algorithm 13 Primal-dual algorithm to solve the spatio-temporal segmentation problem (4.6) for tracking and video segmentation.

$\tau = \sigma = \frac{1}{\sqrt{6}}$
$u^0 = 0.5$, $\boldsymbol{p}^0 = 0$ and $\bar{u}^0 = u^0$ // Initialization
for $j = 1$ to J **do**
  $\boldsymbol{p}^j = \Pi_C\left(\boldsymbol{p}^{j-1} + \sigma\nabla\bar{u}^{j-1}\right)$ // Update dual variable
  $u^j = \left[u^{j-1} - \tau(-\mathrm{div}\boldsymbol{p}^j + w_u)\right]_0^1$ // Update primal variable
  $\bar{u}^j = 2u^j - u^{j-1}$ // Extra-gradient step
end for

formulation of (4.6). Similar to (3.13), we obtain the pure dual energy as

$$E_d(\boldsymbol{p}) = \left\langle \hat{u}, \nabla^T\boldsymbol{p} + w_u \right\rangle, \tag{4.8}$$

with $\hat{u}$ as in (3.12). We can therefore use the normalized primal dual gap defined in (3.14) as a convergence criterion.

4.2.3 Implementation

The anisotropic segmentation model can be directly used for video segmentation using the tools already introduced in Section 4.1. Instead of working on a single image, the user can draw constraints to arbitrary frames. The desired segmentation is then obtained in an interactive manner.

In the context of tracking we are not focused on the perfect segmentation, but in minimizing the user input. Small errors are accepted as long as user interaction is kept to a minimum. We now take a closer look at how the above segmentation algorithm can be used for tracking.

4.2.3.1 The tracking framework

Due to limitations in computer hardware such as memory, the size of volumes that can be computed at once is limited. Although modern computing hardware can handle volumes with several thousand frames, the necessity of working on the complete sequence at once restricts tracking to offline data. Multiple similar objects, or disjoint regions belonging to the same object (e.g. by occlusions) make additional information necessary. When attempting a general framework with objects of arbitrary size and shape, this becomes a difficult task.

To tackle these problems, we propose to use an incremental approach. Only n frames

are segmented at once. The algorithm is initialized on the first n frames, e.g. by drawing a rectangle around the desired object. See Section 4.2.3.2 for details of the feature based segmentation approach. If multiple objects are segmented, the user can select the desired object manually by editing additional scribbles. After convergence of the segmentation algorithm (Section 4.2.2) foreground and background models are updated.

Next, the oldest $m < n$ frames are discarded, and m new frames are added to the volume V. To speed up the tracking process we compute the segmentation only on small areas around the current object. To prevent the algorithm from segmenting similar nearby objects, only regions that overlap with the segmentation mask of the last step are selected. In case of occlusions, the volumetric representation of an object might be separated into several disjoint regions. Our overlap constraint might for some cases cause the tracker to discard the new region if they are not connected in the volume.

To handle complete occlusions in general we use the following strategy: We keep track of the average region size. If the segmentation gets smaller than a certain percentage of this average region size, the object is assumed to be occluded. In case of an occlusion the working region starts to grow slowly, and no updates of the foreground and background model are performed. If a region is segmented that is big enough to be considered as the object, tracking is continued on this region.

Any slice $k \in [1, n]$ of the volume V can be selected as the tracking result. The number of frames the tracker looks into the future is defined by $n - k$. Thus the smaller k and the bigger n, the more robust disjoint regions are tracked.

Implementation of the tracker was done mainly on the GPU. The volume depth was fixed for all experiments to $n = 8$, while slice $k = 4$ was used for the segmentation result.

4.2.3.2 Color tracking

Object appearance is represented in RGB color space using a foreground histogram $H_F : \mathbb{R}^3 \rightarrow [0, 1]$, and a background histogram $H_B : \mathbb{R}^3 \rightarrow [0, 1]$. Following the ideas presented in [Bibby and Reid, 2008], we are using the pixel-wise posterior. We define $M = M_F, M_B$ as the model parameter that is either foreground F or background B. From the initialization, we obtain the foreground and background likelihoods $P(H_F|M_F)$ and $P(H_B|M_B)$. Applying Bayesian rule we can estimate the posterior $P(M_F|H_F)$ of a pixel being foreground in the context of the actual background given by $P(H_B|M_B)$ and a region-prior $P(M_j)$ with $j \in F, B$ by:

$$P(M_F|H_F) = \frac{P(H_F|M_F)P(M_F)}{\sum_{j=F,B} P(H_j|M_j)P(M_j)} \tag{4.9}$$

We keep track of foreground and background models by updating them online using an adaption rate α with likelihoods estimated from the current frame $P_{new}(H_j|M_j)$ as:

$$P(H_j|M_j) = (1-\alpha)P_{old}(H_j|M_j) + \alpha P_{new}(H_j|M_j) \quad with \quad j \in F, B \tag{4.10}$$

In contrast to [Bibby and Reid, 2008] we do not apply marginalization. Instead we simply set the segmentation cue $f(\boldsymbol{x}, t) = 0.5 - P(M_F|H_F(V((\boldsymbol{x}, t))))$.

4.2.4 Experimental results

In Figure 4.9, a white cat is successfully tracked and segmented. The first row shows the input video with different overlays. The rectangle is indicating the current working region. The blue color of the rectangle indicates that the tracker is working normally. If the object is believed to be occluded in some slice, the rectangle becomes orange. The current object is indicated by an orange overlay. If parts of the image get segmented, but do not belong to the object, these areas are indicated in red. Note that some regions are segmented that do not belong to the object, but most of the incorrect regions are removed. The second row shows the segmentation cue f where the value -0.5 is mapped to black and indicates foreground, the value 0.5 is mapped to white and indicates background. Frame 409 shows a segment where a cross-fade occurs. The tracker detects the loss of the object, starts

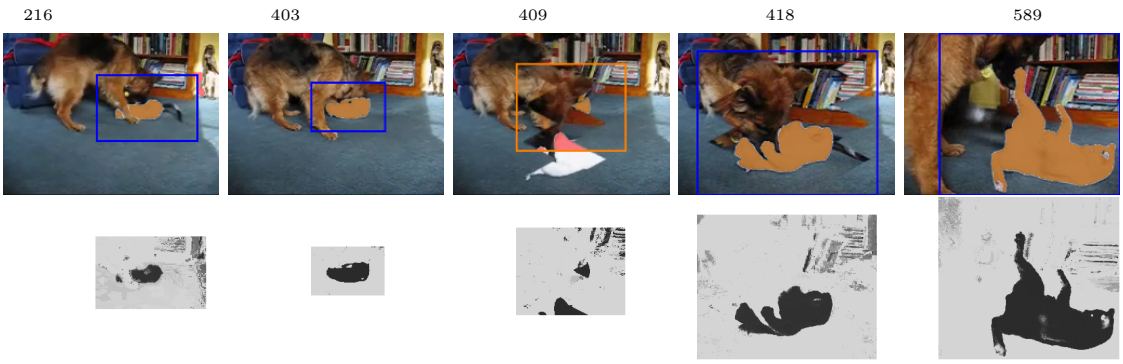


Figure 4.9: Tracking example of a cat. The first row depicts the tracked object with the current segmentation and the working region as overlays to the original input image. In the second row the segmentation cue f is depicted in the range $[-0.5,\ 0.5]$.

growing the search region and begins to search for the object. Frame 418 shows that the object was found correctly. Also note that the algorithm always correctly tracks the object despite large scale changes, as our tracking approach makes no restrictions on the region size.

The second example presented here shows the tracking of a fish in an aquarium. In the top row of Figure 4.11, the input video is shown, while the bottom row shows the extracted fish. Note that although several partially and complete occlusions occur, the tracker does not lose the object throughout the video. In case of partial occlusions the fish is still correctly segmented, as can be seen in frames 438 and 487. Also note that large shape changes do cause tracking failures, as we make no assumption on shape. In Figure 4.10, the video is displayed as a volume. The region corresponding to the fish is rendered using iso-surface rendering based on the segmentation mask as obtained by the tracker.

Naturally a color based tracker without any restrictions on shape and scale has its limitations. In Figure 4.12 a player in a volleyball game is tracked. In the beginning the tracker starts promisingly by separating skin tones from the very similar sand. Around frame 337 the skin tones of other players appear in the working region, and are learned as background. As one can see in frame 377 the tracker looses the legs and arms, but still

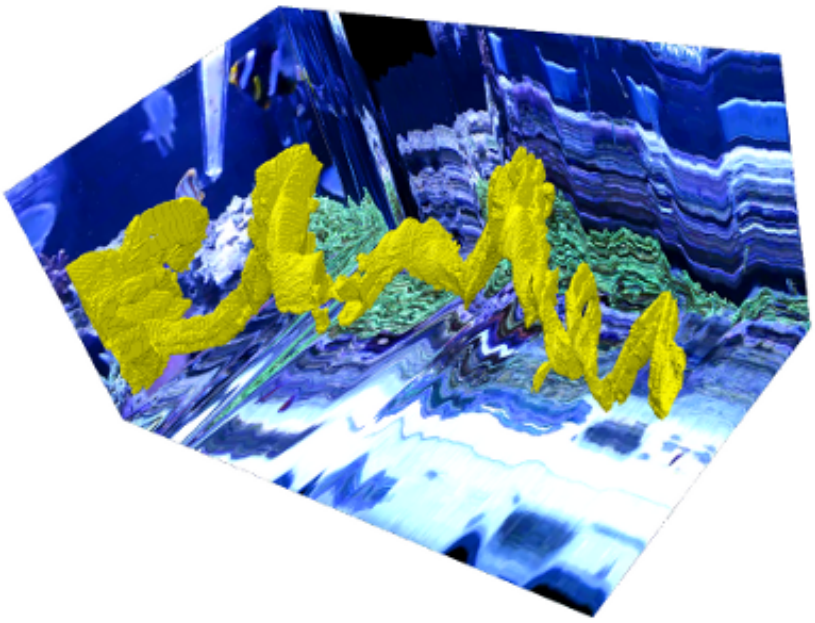


Figure 4.10: A schematic 3D rendering of the fish tracking sequence from Figure 4.11. The tracking result is rendered in yellow.

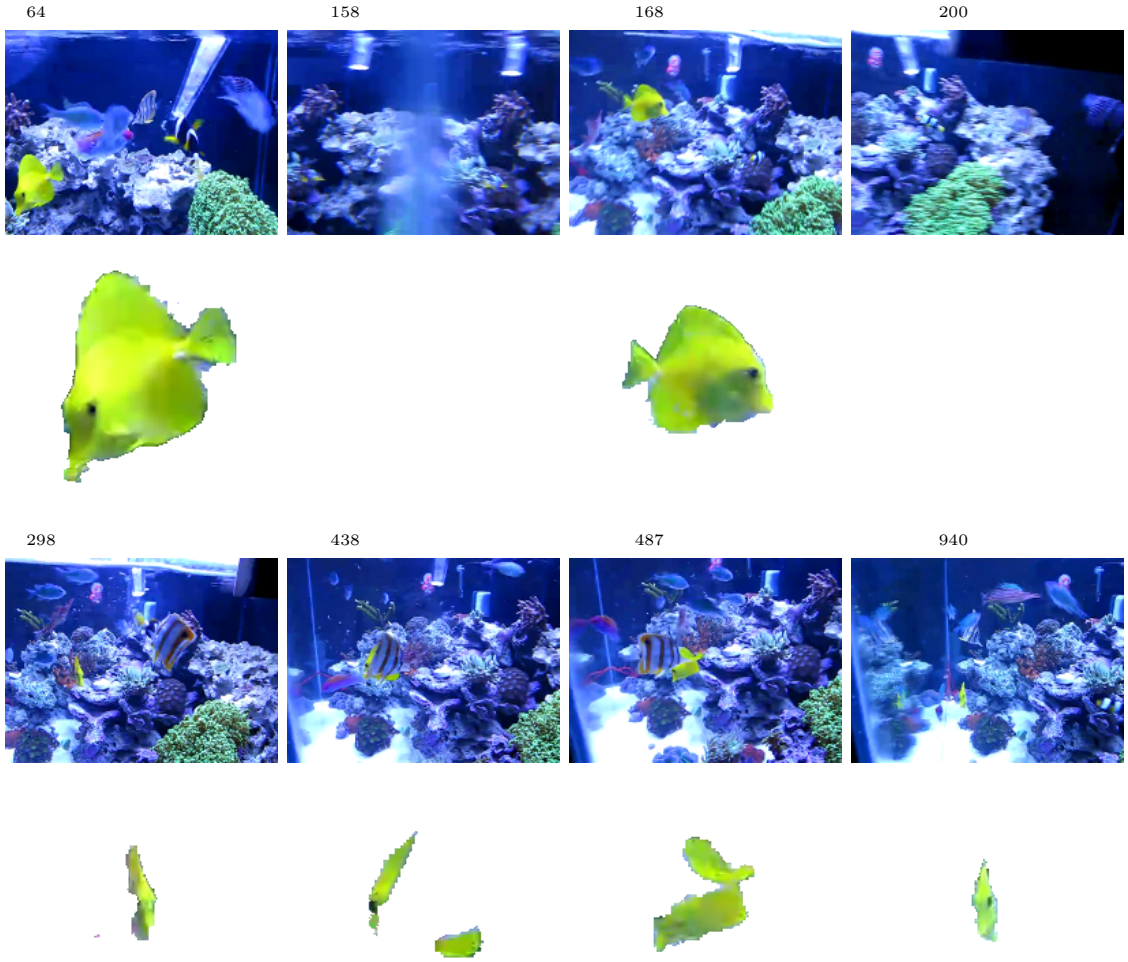


Figure 4.11: Tracking sequence of a fish in an aquarium, showing the ability of the tracker to handle large changes in shape, and various kinds of occlusions. The first row depicts the input video and the second row the extracted object.

tracks the very characteristic green shirt. In frame 551 the player gets occluded by his team member, with a very similar appearance. As no additional high level information is available, both players are tracked.

In Figure 4.13, another video sequence is shown where the tracker fails. We tried to track the skin of the person. Due to the many occlusions the volume corresponding to skin is separated into several disjoint regions, causing problems for the tracker. Though the tracker can recover several times, the object is permanently lost in frame 276. Other reasons for the failure in this video is the bad discrimination of foreground and background

Figure 4.12: Tracking of volleyball sequence, where tracker fails due to highly similar object and colors in the background. The first row shows the input video and the bottom row shows the extracted player.

by using color.

Experimental results showed that a simple color tracker benefits from interpreting tracking as segmentation in 3D. The tracker successfully handled large variations in scale and shape. The examples show, that the tracker can deal with partial occlusions. Due to the incremental approach also long complete occlusions do not pose any problem to the tracker. Figure 4.12 shows an important characteristic of the tracker to adapt foreground and background models to the most characteristic color values. This has the advantage of making the tracking of the object more robust, but also decreases segmentation performance. It also showed that multiple objects with similar appearance cannot be kept apart if occlusions occur. Here clearly high level information could help, e.g. in the volleyball example restrictions on the region size could be made, and shape information would

Figure 4.13: Video example where tracking and segmentation fail, due to too many occlusions, and bad discrimination of the color histograms.

definitely improve results.

4.2.5 Summary

We presented a tracking approach that tracks objects by segmenting them in a spatio-temporal volume. By using the segmentation result a pixel wise classification into foreground and background is achieved. The volumetric tracker presented in this paper, shows promising results for the examples provided in Section 4.2.4. An incremental tracking approach was presented and implemented, that works only on a small volume at a time, eliminating memory problems and allowing tracking of videos of arbitrary length. Due to the segmentation in a 3D volume, information is also propagated back through time if the regions are connected in 3D, showing improvements for tracking disjoint regions. As we make no assumptions on shape or scale even large variations cause no problems to the tracker. The tracker is able to handle partial as well as complete occlusions. It was shown that a pure color based foreground and background description is sometimes not sufficient, and leaves room for further improvement.

Future work might include optical flow into the segmentation process. This could e.g.

be done by directing the temporal gradient in the direction of the optical flow. Work that does this already in the context of image denoising is presented in [Werlberger et al., 2011a].

Chapter 5

Unsupervised Segmentation

5.1 Depth image segmentation

In this Section, we tackle the problem of depth (or height) image segmentation. We do this in the context of aerial imagery. Segments in depth images typically cannot be described by simple gray values. Instead, man made structures are typically composed out of several planes (e.g. rooftops or roads). In the following, we develop an approach that jointly optimizes a segmentation of the scene and a plane parametrization for each segment.

5.1.1 Depth segmentation

Depth images play an important role in aerial computer vision and photogrammetry. They depict the geometry of the earth as seen from above using an orthographic projection. As this 2.5D data represents the surface of the earth it is also referred to the as digital surface model (DSM). Typical ways to obtain such depth images is either by using aerial imagery together with 3D reconstruction and stereo matching, or by using laser scanners (Lidar).

The DSM gives a lot of information on the world, and therefore gives rise to extract semantic information from it. Fully understanding the depth images is a difficult task, and we will only focus on a very small sub task in this chapter. We want to explain 2.5D footprints of buildings by a set of 3D planes. This makes a lot of sense, as most man made structures like buildings, streets or parking lots are composed of simple planes.

Figure 5.1 illustrates the task we try to solve. The given depth image cannot be successfully segmented using just edges of gray value information. But when looking at the 3D visualization, it is obvious, that the scene consists of piecewise planar regions. We

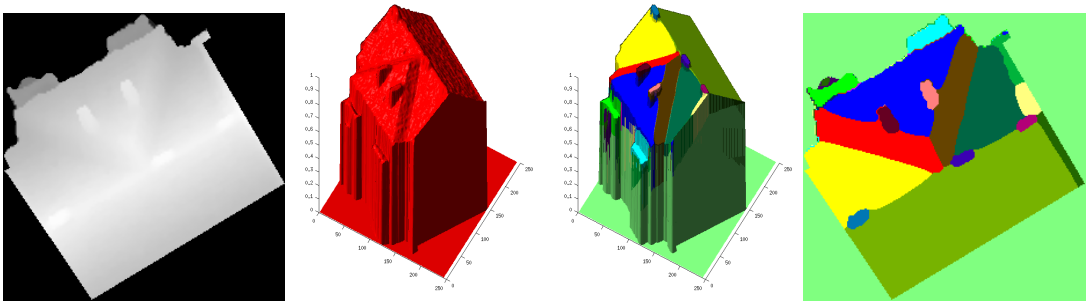


Figure 5.1: Illustration of the depth segmentation problem: On the left, the 2.5D input image, and the corresponding 3D visualization. The right hand side depicts the resulting segmentation into planes. The 3D reconstruction show the accuracy of the result.

now want to segment the input into this consistent regions such that each segment can be represented using simple plane parameters. The corresponding reconstruction not only shows a meaningful segmentation of the building, but also successfully removed noise and small artifacts.

Image segmentation has already proved very useful during stereo estimation. Early works (e.g. [Bleyer and Gelautz, 2004] and [Zitnick and Kang, 2007]) used an over-segmentation of the input images to help the estimation process. In [Sinha et al., 2009], piecewise planar stereo reconstructions were proposed in order to do image based rendering mainly on side views of buildings. First structure from motion is used to generate a point cloud, to which then planes are fitted. The final stereo result is then obtained by a MRF optimization problem. In [Bleyer et al., 2010] and [Bleyer et al., 2011], models for joint stereo matching and object segmentation were introduced. They propose a model that segments the scene based on color and 3D connectivity. Therefore each segment is represented as a plane or B-spline. The segmentation not only improves the final result, but also gives rise to better occlusion handling. Everything is formulated as one joint MRF. We present a similar approach in Section 5.2 for joint motion estimation and segmentation. For now, we assume a given stereo image that has to be segmented.

5.1.2 Model and algorithm

We start with an input image $I : \Omega \to \mathbb{R}$ that contains the depth information. As a model for the label representations, we use planes that are represented by a linear operator $h = (a, b, c)$. The plane is then computed as $y = h\tilde{\boldsymbol{x}}$. Instead of standard coordinates $\boldsymbol{x} = (x, y)^T$ we use homogeneous coordinates $\tilde{\boldsymbol{x}} = (x, y, 1)^T$

Given a segmentation into K regions Ω_i and the according plane parameters h_i it is possible to reconstruct the depth image as

$$u(\boldsymbol{x}) = \sum_{i=1}^{K} (h_i \tilde{\boldsymbol{x}})\, \mathbf{1}_{\Omega_i}(\boldsymbol{x}) \; . \tag{5.1}$$

As a data term for the segmentation, we minimize the quadratic distance of the reconstruction u to the input image I by setting

$$f_i(\boldsymbol{x}) = \lambda |u_i(\boldsymbol{x}) - I(\boldsymbol{x})|^2 \; . \tag{5.2}$$

We then formulate the multi label segmentation problem with label costs in the manner of (3.30) as

$$\begin{aligned} \min_{\Omega_i, h_i} \left\{ \sum_{i=1}^{K} \left(Per(\Omega_i) + \lambda \int_{\Omega_i} |h_i \tilde{\boldsymbol{x}} - I(\boldsymbol{x})|^2 d\boldsymbol{x} + \gamma ||\mathbf{1}_{\Omega_i}||_\infty \right) \right\}, \\ s.t. \quad \Omega = \bigcup_{i=1}^{K} \Omega_i \; , \quad \Omega_i \cap \Omega_j = \emptyset \quad \forall i \neq j \; . \end{aligned} \tag{5.3}$$

The optimization problem (5.3) is non-convex due to the data term. We therefore propose to do an alternating minimization with respect to Ω_i and h_i.

Optimization for Ω_i: For the optimization in the variable Ω_i, we assume that the plane parameters h_i are constant. We can therefore precompute f_i as in (5.2). As a result (5.3) reduces to

$$\begin{aligned} \min_{\Omega_i} \left\{ \sum_{i=1}^{K} \left(Per(\Omega_i) + \int_{\Omega_i} f_i(\boldsymbol{x}) d\boldsymbol{x} + \gamma ||\mathbf{1}_{\Omega_i}||_\infty \right) \right\}, \\ s.t. \quad \Omega = \bigcup_{i=1}^{K} \Omega_i \; , \quad \Omega_i \cap \Omega_j = \emptyset \quad \forall i \neq j \; . \end{aligned} \tag{5.4}$$

This is exactly the multi-label formulation in (3.30) and can be solved by Algorithm 10.

Optimization for h_i: Optimization for the plane parameters can be done by setting Ω_i to be constant. (5.3) then reduces to

$$\min_{h_i} \left\{ \int_{\Omega_i} |h_i \tilde{\boldsymbol{x}} - I(\boldsymbol{x})|^2 d\boldsymbol{x} \right\}. \tag{5.5}$$

Algorithm 14 Algorithm for depth image segmentation (5.3).

Initialize Ω_i by splitting the region into 10×10 patches
for $j = 1$ to J **do**
  Update plane parameters h_i using (5.8)
  Update segmentation Ω_i using Algorithm 10
  Discard empty labels and split spatially disconnected regions
end for

This is a simple label-wise least squares problem, that can be written as

$$X_i h_i^T = y_i, \tag{5.6}$$

with

$$X_i = \begin{pmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_K & y_K & 1 \end{pmatrix} \text{ and } y_i = \begin{pmatrix} I(\boldsymbol{x}_1) \\ \vdots \\ I(\boldsymbol{x}_K) \end{pmatrix} : \boldsymbol{x}_k \in \Omega_i \, . \tag{5.7}$$

As this is a linear least squares problem, it can be solved using the pseudo inverse:

$$h_i^T = (X_i^T X_i)^{-1} X_i^T y_i. \tag{5.8}$$

Initialization and algorithm: For the above two optimization steps, an initialization of either the segmentation Ω_i or the plane parameters H_i is necessary. We start with a 'patch' segmentation by splitting the image in 10×10 regions. For each of these initial regions, we then update the plane parameters h_i according to (5.8). We then use the new plane parameters to compute the segmentation according to Algorithm 10. The resulting segmentation will give a first segmentation, that is further refined by iterating the two optimization steps. In order to allow for new plane parameters, we split up regions of labels that are not spatially connected. This allows to also capture and adapt planes that were not found with the initialization. Empty labels are deleted during the optimization process. The algorithm is summarized in Algorithm 14.

Determining convergence of the algorithm is not trivial. One can either monitor changes in the parameters h_i or the segmentation Ω_i. Both approaches require a manually selected convergence criterion. For practical application a fixed number of iterations seems to be sufficient. We selected $J = 10$ for all examples presented in this thesis.

5.1.3 Experimental results

In the following we show experimental results of the depth image segmentation model 5.3 in the context of height model generation. As input we use $2^1/_2$D orthographic images where single buildings were already preselected. Figure 5.2, shows different segmentation examples. On the left hand side, the input images with preselected buildings and the

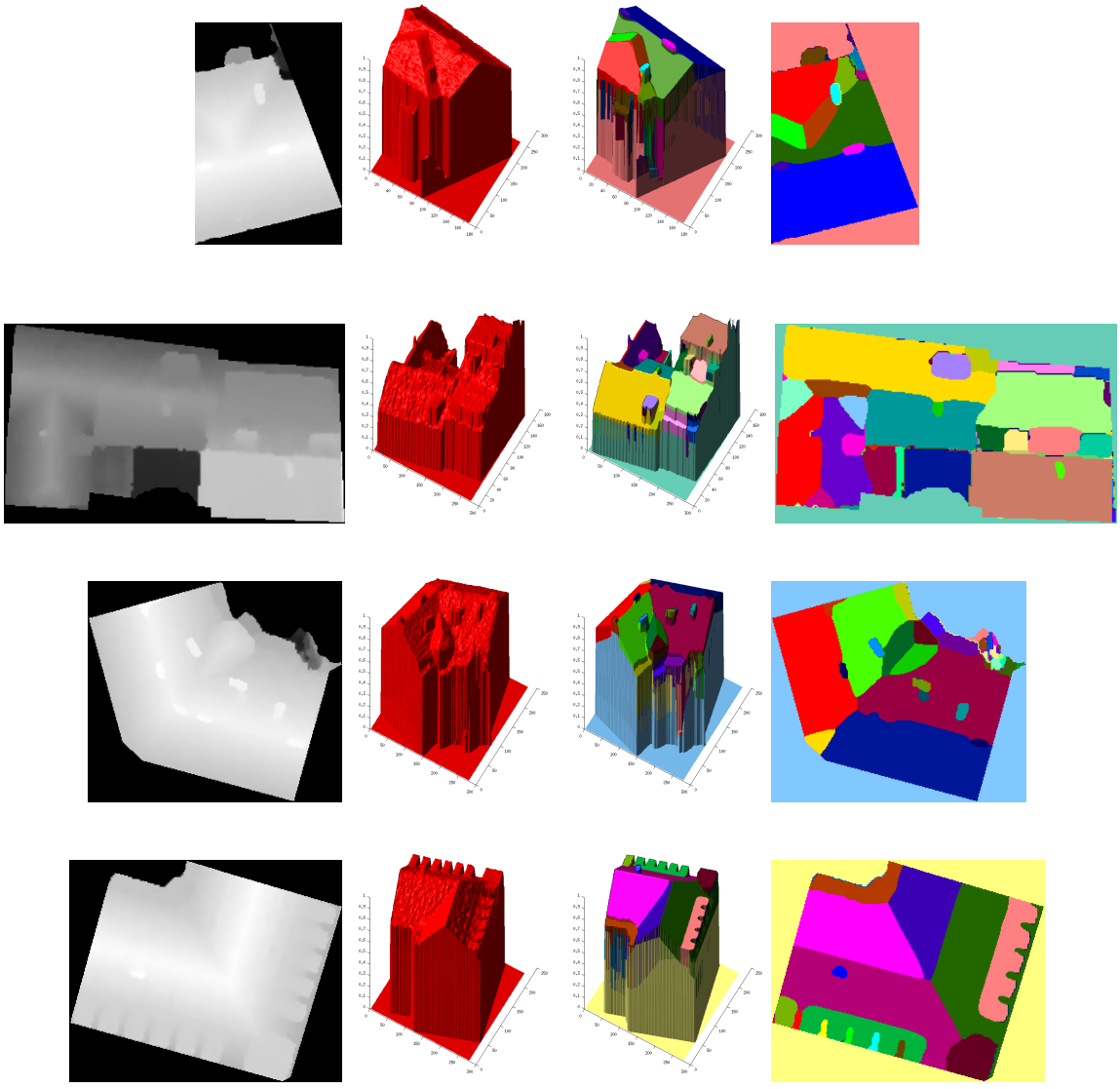


Figure 5.2: Different examples of the height model segmentation, using the same parameters. From left to right: Input depth map I, input rendering, reconstruction rendering and segmentation.

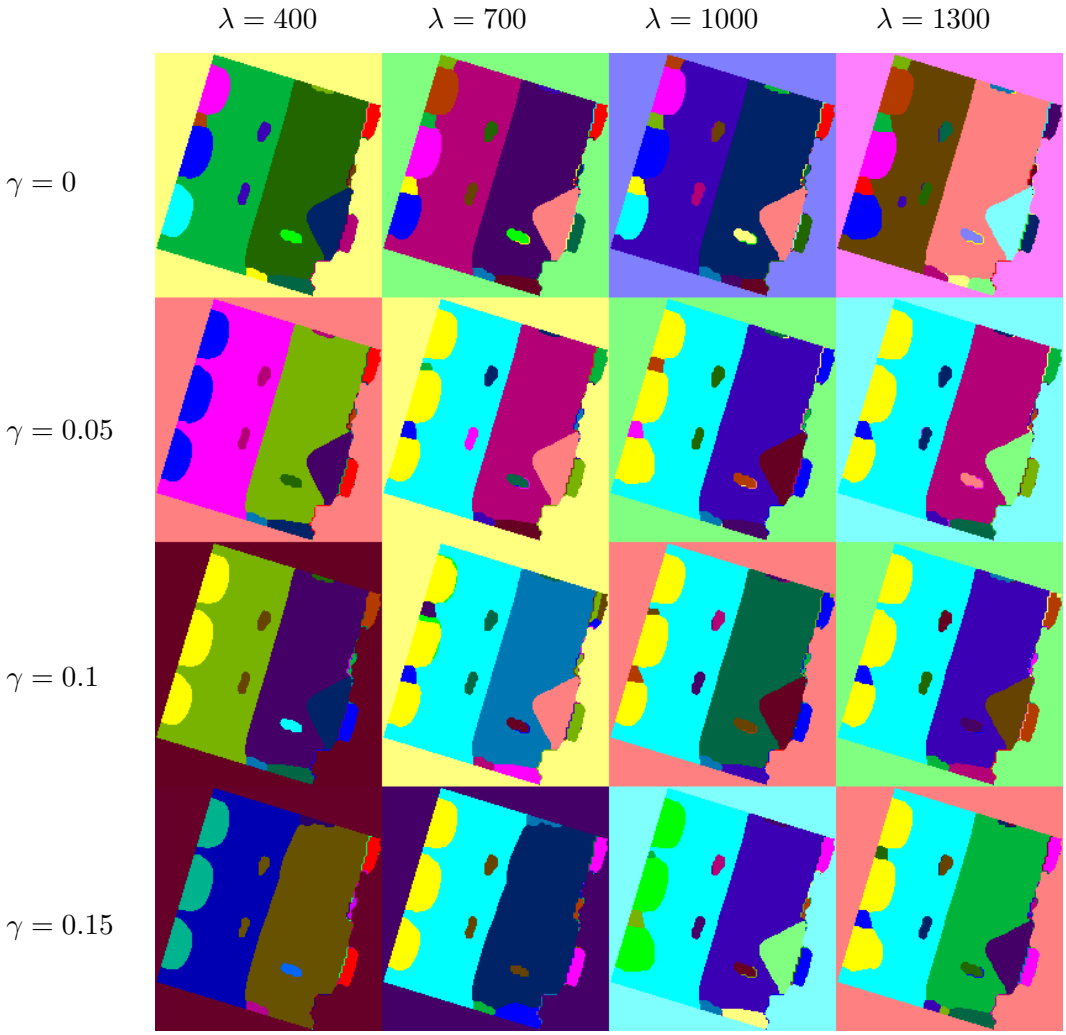


Figure 5.3: Variation of the parameters λ and γ. Within a reasonable range, the segmentation result is quite robust to the choice of parameters.

corresponding 3D visualization. The right hand side shows the reconstruction and the segmentation result. Parameters were kept constant at $\lambda = 700$ and $\gamma = 0.05$. The resulting segmentation provides a meaningful partitioning into the underlying structure of the building. Note that the reconstruction (5.1) provides a denoised version of the input data. The output of the proposed segmentation algorithm is therefore perfectly suited to build a full 3D mesh.

While Figure 5.2 already shows, that a single parametrization can be used for a wide range of input data, we further investigate the influence of parameter choice. In Figure 5.3, the same scene was segmented with varying parameters λ and γ. The segmentation results

show only minor differences, demonstrating that the Algorithm 14 is robust to parameter variations in a reasonable range. When strong regularization is combined with large label costs (bottom left) some regions already start to disappear.

In Figure 5.4, we also demonstrate negative examples where the approach using planes as a model to describe a label are not appropriate. A common structure found in most aerial images are trees. While in the top row, the structure of the house is successfully segmented, strong over-segmentation occurs in areas of with plants. This effect is even more visible in the second row of Figure 5.4. In order to deal with such structures, other models have to be used. We could for instance use quadratic or higher order parametrization, or non-parametric models. In Section 5.2, we will investigate such models in the context of joint motion estimation and segmentation.

Finally, the bottom two rows of Figure 5.4, depict segmentation results of stereo images. The stereo images are ground truth images of the Middlebury stereo benchmark [Scharstein et al., 2002]. While most parts of the images are successfully segmented, some regions are flattened or approximated with several regions. Nevertheless results show that a plane prior might improve regularization in stereo matching.

Algorithm 14 was implemented in Matlab with the exception of the multi-label segmentation problem in Algorithm 10. The segmentation was computed on the GPU, as it poses the most expensive part of the computation. Experiments presented in this section ranged from approximately 10 seconds to 2 minutes per iteration, where we calculated 10 iterations for all examples. The size of the input images ranged form approximately 0.1 to 1.5 million pixels.

5.1.4 Summary

We have demonstrated that the multi label segmentation formulation from Section 3.2 can be extended to segment aerial depth images and stereo data. The data term no longer corresponds to some precalculated binary potentials, but is used to optimize for a parametric representation of each label. Thus, simple feature based models are replaced by parametric models. In addition, we do not longer need any user input, but can use a fully unsupervised segmentation algorithm.

The resulting segmentation and reconstruction provides a meaningful representation of the scene, that can be used for further analysis and extraction of scene semantics. While the planar parametrization is sufficient for most buildings, more complex models would be required to segment vegetation. Of course the approach could also be used directly during

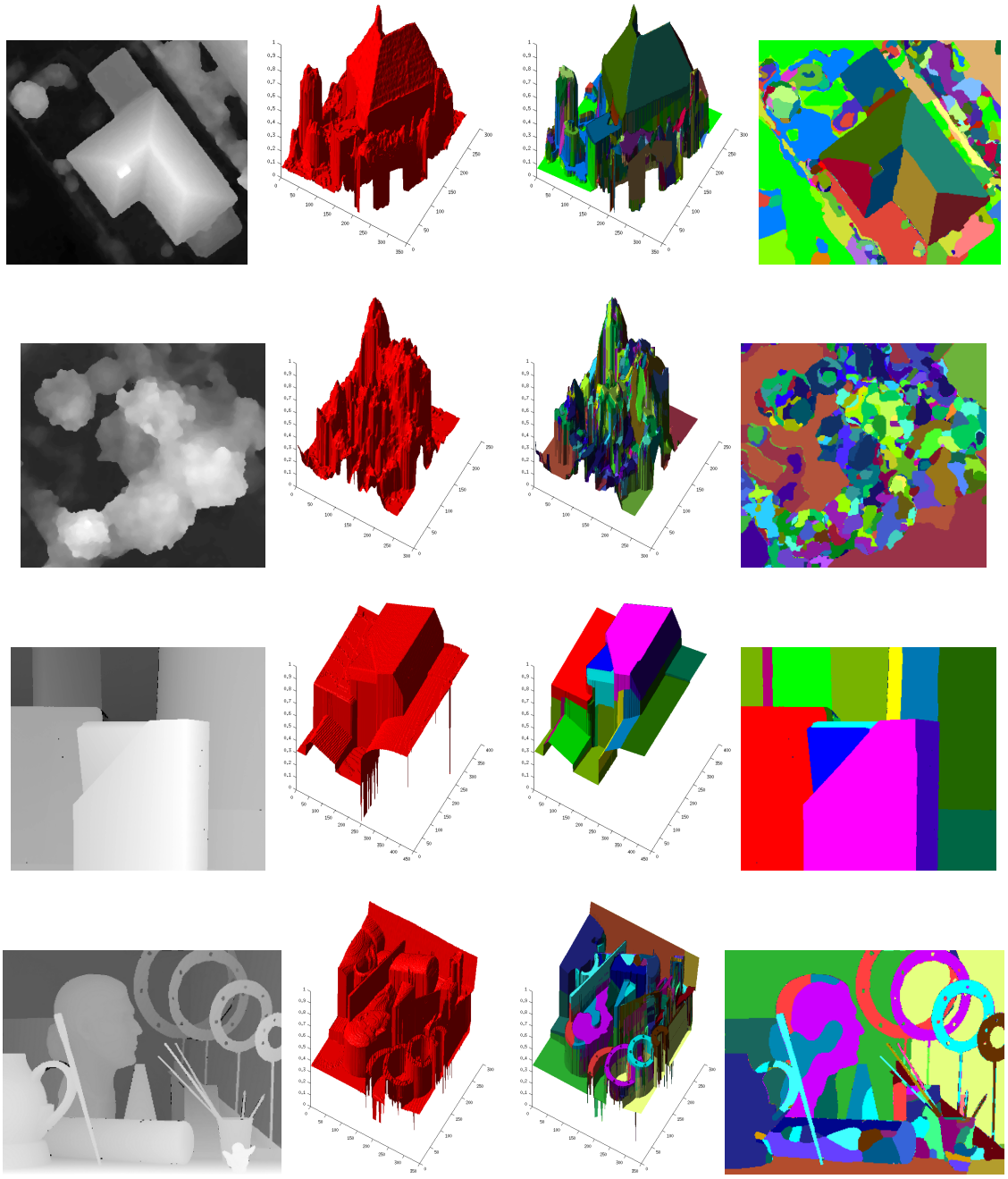


Figure 5.4: The top two rows show negative examples for the depth image segmentation. The bottom tow rows show the algorithm applied to stereo results. From left to right: Input depth map I, input rendering, reconstruction rendering and segmentation.

stereo matching or 3D reconstruction as a regularization term. We will use this idea in the more general case of motion estimation in the next Section.

5.2 Motion Segmentation

This chapter deals with the problem of fully automatic motion segmentation. We present an energy minimization formulation for joint motion estimation, segmentation and occlusion handling. Motion estimation is used as a basic source of information for numerous computer vision applications including tracking, 3D reconstruction, video processing and navigation. Additional motion segmentation gives rise to advanced algorithms for video decomposition, superresolution, compression and others. Motion segments can also be used as additional cues in video segmentation. Note that video segmentation is a very different task as it has to discriminate objects even if there is no motion. In the case of motion segmentation the aim is not to segment objects in the first, but regions with coherent motion. We will demonstrate that for correct occlusion handling and motion segmentation of complex scenes a large number of regions is required. Before we present our method that can deal with all this problems, we first give a brief overview on motion estimation techniques.

5.2.1 A short introduction to motion estimation

We first like to repeat the definition of motion estimation and optical flow, as these are different but closely related problems, but are often used as synonyms. Optical flow refers to the apparent motion of intensity patterns in image sequences. In case of lightning changes this definition might not be intuitive. As an example, think of a torchlight moving over a static scene. Optical flow estimates the motion of the resulting spotlight as it moves through the scene. On the other hand, motion estimation corresponds only to the actual motion of the underlying objects. Motion estimation thus has to be robust against lightning changes and moving shadows. In the following we will use the term *flow* also for the actual motion field, and the term *optical flow* only if we explicitly mean optical flow.

5.2.1.1 Classical optical flow

Starting from the basic assumption, that the intensities do not change between consecutive frames $I_2(\boldsymbol{x} + \boldsymbol{v}(\boldsymbol{x})) = I_1(\boldsymbol{x})$ and applying a first-order Taylor expansion, one obtains the

well-known optical flow constraint (OFC) [Horn and Schunck, 1981]

$$\varrho(\boldsymbol{v}(\boldsymbol{x})) = (\boldsymbol{v}(\boldsymbol{x}) - \boldsymbol{v}_0(\boldsymbol{x}))\nabla I_2(\boldsymbol{x}) + I_2(\boldsymbol{x} + \boldsymbol{v}_0(\boldsymbol{x})) - I_1(\boldsymbol{x}) \ , \tag{5.9}$$

where I_1 and I_2 are the image frames and $\boldsymbol{v}$ is the flow vector. Note that this equation is under-determined, as there is only one equation for each unknown 2-dimensional flow vector. This problem is also called the aperture problem. To overcome this problem, an additional assumption (e.g. spatial coherence) has to be incorporated into the formulation. The seminal work of [Horn and Schunck, 1981] used a variational formulation incorporating a regularization term to circumvent the aperture problem:

$$\min_{\boldsymbol{v}} \left\{ \int_\Omega |\nabla \boldsymbol{v}(\boldsymbol{x})|^2 + \lambda |\varrho(\boldsymbol{v}(\boldsymbol{x}))|^2 d\boldsymbol{x} \right\} \ , \tag{5.10}$$

where the free parameter λ defines the trade-off between smoothness and the residual of the OFC. In subsequent work, the basic formulation of Horn and Schunck has been modified and improved in different ways. For example robust functions [Black, 1991] have been used instead of the quadratic functions in (5.10), to deal with motion discontinuities and outliers in the OFC.

An important and more recent contribution was the realtime optical flow in [Zach et al., 2007], that is often also referred to as *TV-L1 optical flow*:

$$\min_{\boldsymbol{v}} \left\{ \int_\Omega |\nabla \boldsymbol{v}(\boldsymbol{x})| + \lambda |\varrho(\boldsymbol{v}(\boldsymbol{x}))| d\boldsymbol{x} \right\} \ . \tag{5.11}$$

Instead of the quadratic terms of the Horn and Schunck model (5.10) robust $L1$ norms were used. As we have already seen in Section 2.2 the TV in the regularization works edge preserving. It was first used for optical flow in the work of [Shulman and Hervé, 1989]. The $L1$ data term gives additional robustness. While the problem in (5.11) is more difficult than the one in (5.10), using duality the resulting saddle point problem can be solved efficiently. [Zach et al., 2007] showed that the resulting algorithm can be easily parallelized on a GPU making it applicable for realtime applications. Variational methods are well suited for motion estimation and are thus always found among the top performers on the Middlebury evaluation benchmark [Baker et al., 2010]. While the data term using the OFC is highly non-convex, optimization is usually embedded in a coarse-to-fine approach. As a result larger displacements can be handled.

5.2.1.2 Improving optical flow

There is a wide range of possible improvements to the classical optical flow approach [Sun et al., 2010a]. They typically focus either on the data term or the regularization. In the following, we pick a few relevant contributions. For an extensive overview and evaluation of recent optical flow methods, we refer the interested reader to [Werlberger, 2012].

To improve the data fidelity term, more robust norms or features can be used. In [Steinbruecker et al., 2009], a truncated L_1 norm as well as a normalized cross correlation (NCC) based data term was used. Also working with gradients or structure-texture decomposition creates some additional robustness. Especially for large displacements, gray values are subject to strong variations. In this case descriptor matching, as demonstrated in [Brox and Malik, 2011], is an effective method to find correspondences of subsequent frames.

The regularization for optical flow estimation has been heavily researched [Nagel and Enkelmann, 1986]. As simple yet very efficient regularization is the Huber norm [Huber, 1973], that is well known from robust statistics. While still allowing for discontinuities, also smooth transitions are allowed for small gradients. As shown in [Werlberger et al., 2009] for optical flow, this already reduces the typical stair-casing artifacts caused by the TV. To enforce affine regions in the flow field, it is also possible to use 2nd order Total Generalized Variation [Bredies et al., 2010].

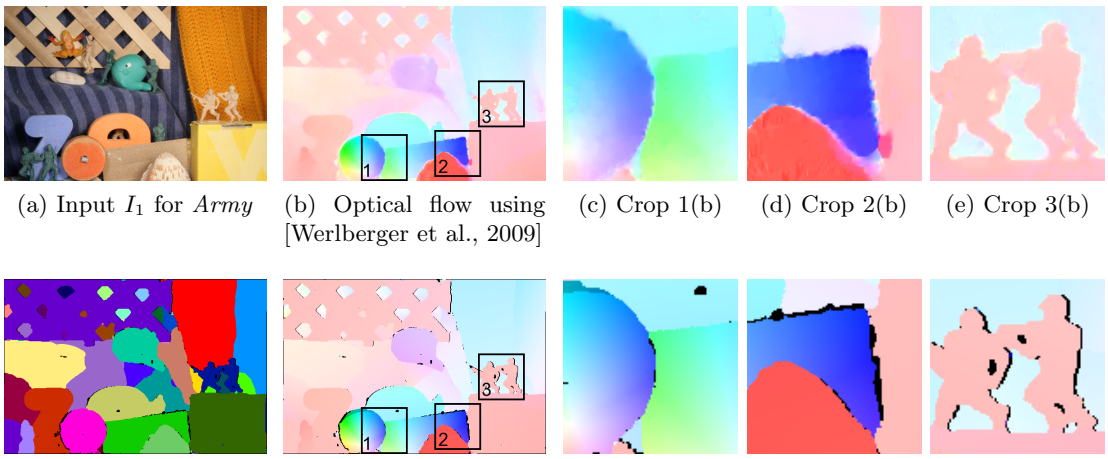
Regularization using non-local TV as presented in [Werlberger et al., 2010] is very efficient at preserving fine details of the scene. The non-local weights (e.g. based on color similarity) can already be seen as a presegmentation.

Another form of regularization is to take into account longer image sequences. As shown in [Ochs and Brox, 2011], the additional information creates more robustness. The trajectory based approach of [Brox and Malik, 2010; Ochs and Brox, 2011] is also a good input for subsequent motion segmentation and clustering.

5.2.1.3 Drawbacks of classical motion estimation

In Figure 5.5, the most common drawbacks are depicted:

First, the regularization of the flow field can lead to artifacts in the flow field. This is especially true around object boundaries. Gradient based regularization penalizes jumps in the motion field. While this is a good prior inside an object, this assumption is violated at object boundaries. Imagine two objects with different motions intersecting: Along the boundary of these two objects the motion gradient is exactly the relative motion difference.



(a) Input I_1 for *Army* (b) Optical flow using [Werlberger et al., 2009] (c) Crop 1(b) (d) Crop 2(b) (e) Crop 3(b)

(f) Motion segmentation (g) Improved flow (h) Crop 1(g) (i) Crop 2(g) (j) Crop 3(g)

Figure 5.5: Comparison of our approach (bottom row) to a classical optical flow approach (top row). We are able to deal with a very large number of labels that are necessary for complex scenes. Our joint motion estimation and segmentation approach is able to improve problems caused by a gradient-based smoothness term (stair-casing artifacts, edge smoothing). Additionally our implicit occlusion handling improves the optical flow at motion borders.

The higher the relative speed, the higher the costs in the regularization term. This will eventually cause artifacts and smooth transitions at strong motion boundaries. Obviously it is desirable to treat regularization inside an object independently from the regularization along object borders. Therefore, an additional segmentation step is required. This problem motivates us to introduce the joint motion estimation and segmentation model in Section 5.2.3.

Second, an optimal regularization of the flow field is not a trivial task. While in general a smoothness assumption inside an object makes sense, this might not be true for all transformations e.g. rotations. Especially TV-regularization is well known for stair-casing artifacts. In many natural scenes the motion field consists of piecewise homogeneous motion (e.g. affine, quadratic). Hence, we will see that modeling the motion field as a piecewise parametric field gives a stronger prior to the motion estimation problem.

Finally, occlusions and dis-occlusions are necessary to identify regions where the OFC cannot be fulfilled. Most classical motion estimation approaches do not consider occlusions during optimization, but rather calculate occlusions as a post processing step. As we will see, a piecewise parametric representation of the flow field allows to simultaneously solve for occlusions and the flow field.

5.2.2 Related work on motion segmentation

Opposed to video segmentation, where a video is decomposed into single objects (even if there is no motion), motion segmentation aims to partition videos into regions with coherent motion. While there are interactive methods as e.g. [Nieuwenhuis et al., 2010], we will focus on unsupervised methods only. Motion segmentation can be done by using the estimated motion vectors in a clustering algorithm to obtain regions with similar motion vectors [Wang and Adelson, 1994]. The approach of [Reina et al., 2010] groups regions together by oversegmentation. Then trajectories of those superpixels are estimated. Those hypothesis then compete against each other to form a reasonable segmentation. A similar superpixel based approach is taken in [Zitnick et al., 2005]. In [Bleyer et al., 2011], the benefit of a joint segmentation is presented in the context of stereo estimation.

Drawbacks of classical optical flow make it clear that joint segmentation and motion estimation circumvents problems otherwise caused by treating segmentation as a post processing step. In [Cremers, 2003; Cremers and Soatto, 2005; Schoenemann and Cremers, 2006], this is done by modeling motion with affine parameters. It showed that these methods can be used to discriminate between a background layer and a moving foreground. As an application the decomposed image was used to generate high resolution motion layers [Schoenemann and Cremers, 2008]. In [Kumar et al., 2008], more complex models are learned and applied to motion segmentation.

Occlusions and dis-occlusions are a crucial component for motion segmentation [Ogale et al., 2005], but are often neglected by classical motion estimation approaches. In [Kolmogorov and Zabih, 2001], a map uniqueness constraint is embedded in a graph cut framework for stereo and motion estimation. For stereo estimation the depth ordering (and therefore the knowledge of occlusion presence) is essential, occlusion handling is often incorporated in such approaches [Bleyer et al., 2010; Woodford et al., 2009]. More complex approaches that jointly try to optimize for segmentation, camera motion, optical flow, depth ordering and occlusions were recently proposed by Sun et al. [Sun et al., 2010b] for multiple layers or Zhang et al. [Zhang et al., 2011] for two layers.

Most of the above mentioned methods are defined in a discrete setting. In contrary the proposed method will be defined in a continuous setting. Additionally, we are able to use a large number of labels in the segmentation approach.

5.2.3 A model for joint parametric motion estimation and segmentation

As we have seen in Section 5.2.1, classical regularization in motion estimation may lead to artifacts. To overcome this problems, we suggest to do joint motion estimation, segmentation and occlusion handling, as already seen in Figure 5.5. In the following we introduce a novel model to solve this task for a large number of labels.

5.2.3.1 The basic model

We start with two given input images $I_1 : \Omega \to \mathbb{R}^D$ and $I_2 : \Omega \to \mathbb{R}^D$. Note that the input images do not necessarily have to be gray value or RGB images. We therefore denoted with D the feature dimension of the input images: e.g. $D = 1$ for gray value images, $D = 2$ for gradients, $D = 3$ for RGB or HSV. Of course also higher dimensional features can also be used.

We define optical flow $\boldsymbol{v} : \Omega \to \mathbb{R}^2$ as the motion from image I_1 to image I_2. The proposed motion segmentation model partitions the image domain Ω into K pairwise disjoint sets Ω_i. Where the motion inside a region is described by a set of parameters integrated into a linear operator H_i. The complete flow field $\boldsymbol{v}$ is therefore given as

$$\boldsymbol{v}(\boldsymbol{x}) = \sum_{i=1}^{K} (H_i \boldsymbol{x} - \boldsymbol{x})\, \mathbf{1}_{\Omega_i}(\boldsymbol{x}), \tag{5.12}$$

with the characteristic function as in (2.14) identified with the set Ω_i. For modeling occlusions, we introduce an additional label $i = K+1$ as the occlusion label. Of course, we could easily add more complex motion models as separate labels. It is also straightforward to add a precalculated optical flow as an additional label, as we will demonstrate in Section 5.2.5.2.

We can now define our joint motion estimation and segmentation model as the following energy minimization problem:

$$\begin{aligned} \min_{\Omega_i, H_i} \left\{ \sum_{i=1}^{K+1} \left(Per(\Omega_i) + \lambda \int_{\Omega_i} f_i(\boldsymbol{x}) d\boldsymbol{x} + \gamma ||\mathbf{1}_{\Omega_i}||_\infty \right) \right\}, \\ s.t. \quad \Omega = \bigcup_{i=1}^{K+1} \Omega_i \ , \quad \Omega_i \cap \Omega_j = \emptyset \quad \forall i \neq j \ . \end{aligned} \tag{5.13}$$

The first part is a regularization term that minimizes the perimeter of the regions Ω_i, and the second term models the data fidelity. The number of used segments should be as small

as necessary. Therefore, the last part of (5.13) is again the label cost term as discussed in Section 3.2.3. λ and γ are free parameters to weight the single terms. The unary functions $f_i : \Omega \to \mathbb{R}$ are given by a similarity measure $\Phi\left(I_1(\boldsymbol{x}), I_2(H_i\boldsymbol{x})\right)$:

$$f_i(\boldsymbol{x}) = \begin{cases} \Phi\left(I_1(\boldsymbol{x}), I_2(H_i\boldsymbol{x})\right) & \text{if} \quad 1 \leq i \leq K \\ \theta & \text{if} \quad i = K+1 \end{cases} , \tag{5.14}$$

with θ being a fixed cost for the occlusion label. If we want to use the OFC as in (5.9), the brightness constancy assumption $I_1(\boldsymbol{x}) = I_2(H_i\boldsymbol{x})$ can be enforced with

$$\Phi\left(I_1(\boldsymbol{x}), I_2(H_i\boldsymbol{x})\right) = \left|I_2(H_i\boldsymbol{x}) - I_1(\boldsymbol{x})\right| . \tag{5.15}$$

For gray scale or RGB images, (5.15) will result in the classical OFC. But we already mentioned that I_1 and I_2 could also be arbitrary feature vectors. In case of image gradients (e.g. $D = 2$ for x- and y-gradients) the data fidelity term becomes invariant to lightning changes.

But we can also think of more complex data terms as suggested in [Werlberger et al., 2010]. The similarity measure for the NCC is given as:

$$\begin{aligned} \Phi\left(I_1(\boldsymbol{x}), I_2(H_i\boldsymbol{x})\right) &= -\mathrm{NCC}\left(I_1(\boldsymbol{x}), I_2(H_i\boldsymbol{x})\right) \\ &= -\frac{\int_\Omega \left(I_1(\boldsymbol{x}) - \mu_1(\boldsymbol{x})\right)\left(I_2(H_i\boldsymbol{x}) - \mu_2(H_i\boldsymbol{x})B_\Sigma(\boldsymbol{x}-\boldsymbol{y})d\boldsymbol{y}\right)}{\sigma_1(\boldsymbol{x})\sigma_2(H_i\boldsymbol{x})} . \end{aligned} \tag{5.16}$$

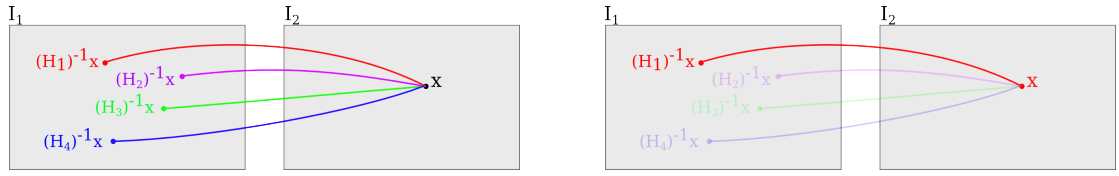
where B_Σ is a box filter with normalization $\int B_\Sigma(\boldsymbol{x})d\boldsymbol{x} = 1$ a local neighborhood Σ around $\boldsymbol{x}$. Thus we can write mean and standard deviation as

$$\begin{aligned} \mu(\boldsymbol{x}) &= \int_\Omega I(\boldsymbol{x}) B_\Sigma\left(\boldsymbol{x}-\boldsymbol{y}\right) d\boldsymbol{y} , \\ \sigma(\boldsymbol{x}) &= \sqrt{\int_\Omega \left(I(\boldsymbol{x}) - \mu(\boldsymbol{x})\right)^2 B_\Sigma\left(\boldsymbol{x}-\boldsymbol{y}\right) d\boldsymbol{y}} . \end{aligned} \tag{5.17}$$

5.2.3.2 Occlusions constraints

As we now have a parametrized version of the motion, it is easy to include different occlusion constraints into the minimization problem in (5.13). We will investigate two different occlusion constraints. First, the map uniqueness constraint that is defined as

$$\sum_{i=1}^{K} \mathbf{1}_{\Omega_i}\left(H_i^{-1}\boldsymbol{x}\right) \leq 1 \; \forall \boldsymbol{x} \in \Omega. \tag{5.18}$$



(a) Map uniqueness constraint (b) Backmatch constraint

Figure 5.6: For the map uniqueness constraint only pixels in I_1 have a label assigned. But only one label that maps to position $\boldsymbol{x}$ in I_2 is allowed to be 1. For the backmatch constraint pixel in I_1 and I_2 have assigned labels. If label i is assigned to pixel $H_i\boldsymbol{x}$ in image I_1 the same label has to be assigned to $\boldsymbol{x}$ in image I_2. In other words, matched pixels must have the same values in their indicator functions $\mathbf{1}_{\Omega_i}$ for all labels.

Only one or no proposal is allowed to map to a single pixel in I_2. In the map uniqueness constraints disocclusions are not handled explicitly. For an illustration of the constraints see Figure 5.6.

The second occlusion constraint is the backmatch constraint and also allows to handle disocclusions. It requires an additional segmentation of image I_2 into disjoint regions Ω_i', resulting in more computational complexity. On the other hand, one automatically obtains the inverse optical flow $\boldsymbol{v}' : \Omega \to \mathbb{R}^2$ that describes the motion for I_2 to image I_1. See Section 5.2.4.3 for more details. The backmatch constraint is defined as

$$\mathbf{1}_{\Omega_i}\left(H_i^{-1}\boldsymbol{x}\right) = \mathbf{1}_{\Omega_i'}(\boldsymbol{x}),\ \mathbf{1}_{\Omega_i}(\boldsymbol{x}) = \mathbf{1}_{\Omega_i'}\left(H_i\boldsymbol{x}\right)\ \forall \boldsymbol{x} \in \Omega. \tag{5.19}$$

5.2.3.3 Parametrization

We also need a meaningful representation for the parametrization H_i. As a basic parametrization, we suggest to use affine parameters:

$$H_i = \begin{pmatrix} h_{i,0} & h_{i,1} & h_{i,2} \\ h_{i,3} & h_{i,4} & h_{i,5} \end{pmatrix}. \tag{5.20}$$

We therefore have to replace the coordinates $\boldsymbol{x} = (x, y)^T$ with homogeneous coordinates $\tilde{\boldsymbol{x}} = (x, y, 1)^T$. Using affine transformations the motion can consist of translation, rotation, scaling and shearing. Of course this is well suited for planar and rigid objects but will cause problems for more complex motions as we will demonstrate in Section 5.2.5.

A slightly more flexible parametrization can also include quadratic terms:

$$H_i = \begin{pmatrix} h_{i,0} & h_{i,1} & h_{i,2} & h_{i,3} & h_{i,4} \\ h_{i,5} & h_{i,6} & h_{i,7} & h_{i,8} & h_{i,9} \end{pmatrix} , \tag{5.21}$$

with

$$\tilde{\boldsymbol{x}} = (x, y, x^2, y^2, 1)^T . \tag{5.22}$$

This additional degrees of freedom will allow for more smoothness when dealing with non-rigid transformations. While the computational complexity increases only marginally, we will restrict the following derivations to the affine parametrization (5.20) for simplicity. The extensions to the quadratic terms (5.21) are straightforward.

In this Section, we have constructed a joint motion estimation and segmentation model in (5.13) with additional map uniqueness (5.18) and backmatch (5.19) constraints. In the following we will give more details on the parametrization and optimization.

5.2.4 Optimization

The optimization problem in (5.13) with the occlusion constraints, poses a difficult non-convex optimization problem. But if we take a closer look, the model is convex in the segmentation Ω_i. Unfortunately the model is not convex in the motion parameters H_i. But we can easily apply linearization methods well known from standard optical flow calculation [Zach et al., 2007] to obtain a convex approximation of the energy around some current parameters $H_{i,0}$. Hence, we can split up the optimization into two steps. This results in an iterative algorithm, where we first create initial parameters and then iteratively optimize for Ω_i and H_i. In the following, the single steps of the algorithm are described in detail.

5.2.4.1 Parameters H_i

Initialization: As an initialization $H_{0,i}$, we can simply set the parameters to constant values that approximately sample the expected flow range. Alternatively we can initialize parameters by RANSAC style sampling on a precalculated optical flow $\boldsymbol{v}_0$. We therefore use some regions $\Omega_{0,i}$. Thus we can find $H_{0,i}$ by solving $H_{0,i}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{x}} + \boldsymbol{v}_0, \forall \tilde{\boldsymbol{x}} \in \Omega_{0,i}$. The

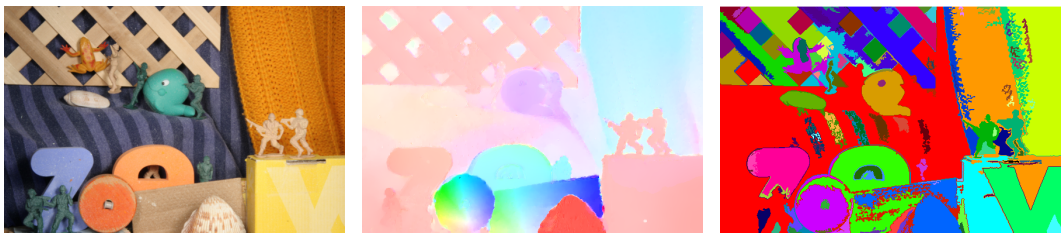


Figure 5.7: Input for initialization when using precalculated flow. From left to right: The input image (*Army*). The precalculated optical flow $\boldsymbol{v}_0$ by [Werlberger et al., 2009]. Superpixels $\Omega_{0,i}$ created with [Felzenszwalb and Huttenlocher, 2004].

point-wise problem can be stated as

$$\begin{pmatrix} h_{i,0} & h_{i,1} & h_{i,2} \\ h_{i,3} & h_{i,4} & h_{i,5} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + v_{i,x} \\ y + v_{i,y} \end{pmatrix} . \tag{5.23}$$

To obtain the regions $\Omega_{0,i}$, the image can be split into any form superpixels. While a simple grid (e.g. 20×20 patches) is sufficient, an meaningful oversegmentation will obviously give even better results. We therefore used the fast graph-based segmentation approach of [Felzenszwalb and Huttenlocher, 2004] to obtain superpixels. Figure 5.7 shows exemplary input used for creating initial proposals based on precalculated optical flow.

Optimization of L1 data term: We first focus on the L1 data term as proposed in (5.15), as the overall optimization shows to be the most robust. A more general optimization scheme is discussed later.

As already stated in (5.14) we only have to consider the first K labels, where we have for each label $\Phi\left(I_1(\boldsymbol{x}), I_2(H_i\boldsymbol{x})\right) = |I_2(H_i\boldsymbol{x}) - I_1(\boldsymbol{x})|$. To optimize for H_i one has to linearize I_2 around the current parameters $H_{i,0}$:

$$I_2(H_i\tilde{\boldsymbol{x}}) = I_2(H_{i,0}\tilde{\boldsymbol{x}}) + \langle \nabla I_2, H_i\tilde{\boldsymbol{x}} - H_{i,0}\tilde{\boldsymbol{x}} \rangle . \tag{5.24}$$

We can thus write the linearized optical flow constraint as

$$f_i(\boldsymbol{x}) = |I_t(\boldsymbol{x}) - (h_{i,0}x + h_{i,1}y + h_{i,2} - v_{x0})I^x(\boldsymbol{x}) - (h_{i,3}x + h_{i,4}y + h_{i,5} - v_{y0})I^y(\boldsymbol{x})| , \tag{5.25}$$

with $I_t(\boldsymbol{x}) = I_2(H_{i,0}\tilde{\boldsymbol{x}}) - I_1(\boldsymbol{x})$, I^x the derivation of I_2 in x-direction, and I^y the derivation

in y-direction and the current flow $\begin{pmatrix} v_{x0} \\ v_{y0} \end{pmatrix} = H_{i,0}\tilde{\boldsymbol{x}}$. For label i, we get the following optimization problem:

$$\min_{h_i} ||C_i h_i - \boldsymbol{d}_i|| \ , \tag{5.26}$$

with

$$h_i = (h_{i,0}, h_{i,1}, h_{i,2}, h_{i,3}, h_{i,4}, h_{i,5})^T \ , \tag{5.27}$$

the $|H_i| \times |\Omega_i|$ matrix

$$C_i = \left(-c_1, -c_j, \ldots, -c_{|\Omega_i|}\right)^T \forall j | \boldsymbol{x}_j \in \Omega_i \ , \tag{5.28}$$

where $|H_i|$ being the number of parameters, $|\Omega_i|$ the number of pixels inside region Ω_i and

$$c_j = \begin{pmatrix} x_j I^x_{\boldsymbol{x}_j} & y_j I^x_{\boldsymbol{x}_j} & I^x_{\boldsymbol{x}_1} & x_j I^y_{\boldsymbol{x}_j} & y_j I^y_{\boldsymbol{x}_j} & I^y_{\boldsymbol{x}_j} \end{pmatrix}^T \ . \tag{5.29}$$

Additionally, we define the $|\Omega_i|$-dimensional vector

$$\boldsymbol{d}_i = \left(d_1, d_j, \ldots, d_{|\Omega_i|}\right)^T \forall j | \boldsymbol{x}_j \in \Omega_i \ , \tag{5.30}$$

with

$$d_j = I_t(\boldsymbol{x}_j) + v_{x_{0,j}} I^x(\boldsymbol{x}_j) + v_{y_{0,j}} I^y(\boldsymbol{x}_j) \ . \tag{5.31}$$

By introducing the vector $\boldsymbol{t}$ of size $|\Omega_i|$, the problem (5.26) can be transformed into the following linear program (LP) that can be solved using any available LP-solver

$$\begin{aligned} &\min_{h_i, \boldsymbol{t}} (\boldsymbol{0}, \boldsymbol{1}) \begin{pmatrix} h_i \\ \boldsymbol{t} \end{pmatrix}, \\ &s.t. \begin{pmatrix} C_i & -I \\ -C_i & -I \end{pmatrix} \begin{pmatrix} h_i \\ \boldsymbol{t} \end{pmatrix} \leq \begin{pmatrix} \boldsymbol{d}_i \\ -\boldsymbol{d}_i \end{pmatrix}. \end{aligned} \tag{5.32}$$

LP-solvers typically require a lot of memory, especially when it comes to large problems like we are dealing with in (5.32). Alternatively, one could apply the primal-dual algorithm from Section 2.3.3 directly to the problem in (5.26), where usually a few hundred iterations will be enough. While this does not guarantee an optimal solution like the LP in (5.32), results are usually sufficient. Additionally, direct variational optimization of (5.26) significantly reduces the amount of memory needed and the GPU implementation is significantly faster.

Optimization of arbitrary data terms: In [Werlberger et al., 2010], a second order approximation of the complete data term was used to handle complex data terms like the truncated normalized cross correlation (TNCC). The NCC is invariant against multiplicative illumination changes. Also patch based data terms like the sum of squared differences (SSD), or more complex data terms become possible. Of course this second order approximation of the data term can also be used for the L1 data terms as in (5.15).

We start with a second order Taylor expansion of the point-wise data term (5.16) around some current parameters $h_{i,0}$, with h_i defined in (5.27). To simplify the notation we replace $\Phi\left(I_1(\boldsymbol{x}), I_2(H_i\boldsymbol{x})\right)$ by $\Phi\left(\boldsymbol{x}, h_i\right)$, and obtain the second order approximation as

$$\begin{aligned}\Phi\left(\boldsymbol{x}, h_i\right) \approx \Phi\left(\boldsymbol{x}, h_{i,0}\right) &+ \nabla\Phi\left(\boldsymbol{x}, h_{i,0}\right)^T \left(h_i - h_{i,0}\right) \\ &+ \frac{1}{2}\left(h_i - h_{i,0}\right)^T \nabla^2\Phi\left(\boldsymbol{x}, h_{i,0}\right)\left(h_i - h_{i,0}\right) \ .\end{aligned} \tag{5.33}$$

Here $\nabla\Phi\left(\boldsymbol{x}, h_{i,0}\right)$ are the first order derivatives of the data term computed by finite differences. We therefore have to choose a Δ_s as a step width for the central differences such that the local variation of the data term is well captured. A good choice of Δ_s is such that the resulting maximum variation of the flow $\boldsymbol{v}_i$ is 0.5, as empirical studies showed. To ensure convexity of the Hessian matrix $\nabla^2\Phi\left(\boldsymbol{x}, h_{i,0}\right)$, we neglect all mixed derivatives according to [Werlberger et al., 2010]. Thus the Hessian is a diagonal matrix containing only the second order derivatives of the single parameters. Note that this data term (5.33) is defined point-wise for all $\boldsymbol{x} \in \Omega_i$. The corresponding Euler-Lagrange equation is then given as

$$\nabla\Phi\left(\boldsymbol{x}, h_{i,0}\right) + \nabla^2\Phi\left(\boldsymbol{x}, h_{i,0}\right)\left(h_i - h_{i,0}\right) = 0 \ . \tag{5.34}$$

As a result we get a very simple update equation for the parameters h_i as

$$h_i = h_{i,0} - \frac{\nabla\Phi\left(\boldsymbol{x}, h_{i,0}\right)}{\nabla^2\Phi\left(\boldsymbol{x}, h_{i,0}\right)} \ . \tag{5.35}$$

The second order derivation of (5.33) is only valid in a local neighborhood. To prevent the algorithm from drifting into a wrong direction, we clamp h_i to the interval $[h_i - \Delta_s, h_i + \Delta_s]$. Additionally, we accept updates only if the energy of the original data term $\Phi\left(\boldsymbol{x}, h_i\right)$ is lower than before the update. Although one could iterate this update (5.35) and do several subsequent approximation steps, usually a single step is sufficient.

Splitting and Merging: In order to allow new proposals to be generated, we split each label into spatially connected regions. For each of this regions the parameters H_i are

optimized separately. As we demonstrate in Figure 5.11 this process allows to create and adapt new proposals throughout the iteration process. To prevent over fitting of small (possibly incorrect) regions and due to memory limitations, we skip regions with less than 10 pixels. We do not have to account for merging labels, as the segmentation model with label costs will do this implicitly.

5.2.4.2 Segmentation Ω_i with map uniqueness constraint

To solve the Potts based energy with label cost (5.13) and occlusion term with map uniqueness constraint (5.18), we first have to apply some relaxations. We again assume that images are given on a discrete Cartesian grid of size $M \times N$: $\{(k,l) : 1 \leq k \leq M, 1 \leq l \leq N\}$, with the indices of the discrete locations given by (k,l) and pixels of size 1. For reasons of clarity we stick to the notation of $\boldsymbol{x}$ for the discrete location and use function arguments for indexing a vector. We use the multi label segmentation presented in Section 3.2.1. The continuous variable $u_i \in [0,1]^{MN}$ replaces the characteristic function $\mathbf{1}_{\Omega_i}$. The unary terms becomes the vector $f_i \in \mathbb{R}^{MN}$. Thus we can formulate the Potts model with label costs (5.13) and map uniqueness constraint as

$$\begin{aligned} &\min_{u_i} \left\{ \sum_{i=1}^{K+1} \left(||W_b \nabla u_i|| + \lambda \langle f_i, u_i \rangle + \gamma ||u_i||_\infty \right) \right\}, \\ s.t. \sum_{i=1}^{K+1} u_i(\boldsymbol{x}) = 1, &\sum_{i=1}^{K} u_i(H_i^{-1}\tilde{\boldsymbol{x}}) \leq 1, u_i \geq 0, \ \forall i = 1, \ldots, K+1 \, . \end{aligned} \tag{5.36}$$

Discretization: For the map uniqueness constraint in (5.36), we need to index the segmentation at position $\boldsymbol{y} = H_i^{-1}\tilde{\boldsymbol{x}}$. For $\tilde{\boldsymbol{x}} = (k,l,1)^T$, $\boldsymbol{y}$ will not necessarily point to a location on the Cartesian grid. Therefore, one either has to do interpolation or rounding of the coordinates. We will use a simple rounding scheme to obtain indices on the Cartesian grid. With the inverse affine parameters denoted as $b_0, \ldots, b_5$, we get

$$H_i^{-1}\tilde{\boldsymbol{x}} = \begin{pmatrix} \lfloor b_0 k + b_1 l + b_2 + 0.5 \rfloor \\ \lfloor b_3 k + b_4 l + b_5 + 0.5 \rfloor \end{pmatrix} \, . \tag{5.37}$$

This enables us to define a linear operator $G \in \mathbb{R}^{MN \times MN}$ with elements

$$g_{m,n,k,l} = \begin{cases} 1 & \text{if} \quad m = \lfloor b_0 k + b_1 l + b_2 + 0.5 \rfloor \text{ and} \\ & \quad\ \ n = \lfloor b_3 k + b_4 l + b_5 + 0.5 \rfloor \\ 0 & \text{else} \end{cases} \tag{5.38}$$

such that the discretized version of the neighboring pixels for label $i = 1, \dots, K$ can be written as $G_i u_i$. Note that $\lfloor x \rfloor$ denotes the floor of x. In other words u_i is warped using the inverse affine transformation with an implicit rounding schema. To further simplify the notation, we add G_{K+1} with elements $g_{m,n,k,l} = 0,\ \forall m, n, k, l$ for the occlusion label.

With the discretized relationship of the map uniqueness constraint, (5.36) can be written as

$$\begin{aligned} &\min_{u_i} \left\{ \sum_{i=1}^{K+1} ||W_b \nabla u_i|| + \lambda \langle u_i, f_i \rangle + \gamma ||u_i||_\infty \right\} , \\ s.t. \sum_{i=1}^{K+1} u_i = \mathbf{1},\ &\sum_{i=1}^{K+1} G_i u_i \leq \mathbf{1},\ u_i \geq \mathbf{0},\ \forall i = 1, \dots, K+1 \ . \end{aligned} \tag{5.39}$$

We will use the primal-dual algorithm of [Chambolle and Pock, 2010] as described in Section 2.3.3 as a solver for the above problem. Therefore we first have to transform (5.36) into a primal-dual saddle point problem.

The saddle point problem: We have already presented a primal dual formulation of the multi label segmentation model with label costs in (3.32). In the same manner the inequality constraint of the map uniqueness constraint $\sum_{i=1}^{K+1} G_i u_i - \mathbf{1} \leq \mathbf{0}$ can be treated. Therefore, we introduce the Lagrange multiplier $s \in \mathbb{R}^{MN},\ s.t.\ s \geq \mathbf{0}$ to obtain the primal-dual formulation $\left\langle s, \sum_{i=1}^{K+1} G_i u_i - \mathbf{1} \right\rangle$. The primal-dual formulation of the sum-constraint is straightforward, by using the Lagrange multiplier $r \in \mathbb{R}^{MN}$. Finally, we arrive at the following primal-dual formulation of (5.39):

$$\begin{aligned} \min_{u_i, t_i} \max_{\boldsymbol{p_i}, q_i, r, s} \Bigg\{ \sum_{i=1}^{K+1} \Bigg(&\langle \boldsymbol{p_i}, \nabla u_i \rangle + \lambda \langle u_i, f_i \rangle + \gamma t_i + \left\langle q_i, P \begin{pmatrix} u_i \\ t_i \end{pmatrix} \right\rangle + I_\Gamma(u_i) \\ &- I_{\Lambda_g}(\boldsymbol{p_i}) - I_\Gamma(q_i) \Bigg) + \left\langle r, \sum_{i=1}^{K+1} u_i - \mathbf{1} \right\rangle + \left\langle s, \sum_{i=1}^{K+1} G_i u_i - \mathbf{1} \right\rangle - I_\Gamma(s) \Bigg\}, \end{aligned} \tag{5.40}$$

with the sets

$$\begin{aligned} \Gamma &= \left\{ z \in \mathbb{R}^{MN} : z_{k,l} \geq 0\ \forall k, l \right\} , \\ \Lambda_g &= \left\{ \boldsymbol{z} = (z^1, z^2)^T \in \mathbb{R}^{2MN} : |\boldsymbol{z}| \leq w_{b_{k,l}},\ \forall k, l \right\} . \end{aligned} \tag{5.41}$$

The saddle point problem (5.40) can be easily brought to the form of (2.50):

$$\alpha = \left(\begin{array}{c|c} (u_1)^T \dots (u_{K+1})^T & (t_1)^T \dots (t_{K+1})^T \end{array} \right)^T ,$$

$$D = \left(\begin{array}{ccc|ccc} \nabla & & & & & \\ & \ddots & & & & \\ & & \nabla & & & \\ \hline I & & & -\mathbf{1}^T & & \\ & \ddots & & & \ddots & \\ & & I & & & -\mathbf{1}^T \\ \hline I & \dots & I & & & \\ \hline G_1 & \dots & G_{K+1} & & & \end{array} \right) \text{ and } \beta = \left(\begin{array}{c} \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_{K+1} \\ \hline q_1 \\ \vdots \\ q_{K+1} \\ \hline r \\ \hline s \end{array} \right) . \tag{5.42}$$

Further we get

$$\Phi(\alpha) = \sum_{i=1}^{K+1} \left(\lambda \left\langle u_i, f_i \right\rangle + \gamma t_i + I_\Gamma(u_i) \right) , \tag{5.43}$$

and

$$\Psi^*(\beta) = r + s - I_\Gamma(s) - \sum_{i=1}^{K+1} \left(I_{\Lambda_g}(\boldsymbol{p_i}) + I_\Gamma(q_i) \right) . \tag{5.44}$$

We can directly apply the primal dual algorithm [Chambolle and Pock, 2010] from Algorithm 1 . The resulting algorithm is summarized in Algorithm 15.

Algorithm 15 is perfectly suited for implementation on parallel hardware. The gradient operator ∇ can be efficiently implemented using local operations, and the adjoint operator div is also simple to compute (see (2.30)). On the other hand a manual inversion of the linear operators G_i is not trivial. Therefore G_i was implemented as a sparse matrix, making it easy to compute G_i^T.

5.2.4.3 Segmentation Ω_i with backmatch constraint

In order to use the backmatch constraint we have to further optimize for the segmentation $u_i' \in [0,1]^{MN}$ of image I_2. Additionally, we need to introduce $f_i' \in \mathbb{R}^{MN}$ similar to (5.14), with a similarity measure $\Phi'\left(I_1(H_i^{-1}\boldsymbol{x}), I_2(\boldsymbol{x})\right)$. The same way as done in (5.38), we can construct a sparse matrix $G' \in \mathbb{R}^{MN \times MN}$ that models the discretized neighbourhoods. Using the backmatch constraint as defined in (5.19) we can write the full discretized joint

Algorithm 15 Algorithm for solving the joint motion estimation and segmentation with map uniqueness constraint (5.40)

// Initialization:
$\tau = \sigma = \frac{1}{L}$
for $i = 1$ to K+1 **do**
  $\boldsymbol{p}_i^0 = (0,0)^T$
  $q_i^0 = 0$
  $\bar{u}_i^0 = u_i^0 = \frac{1}{K+1}$
  $\bar{t}_i^0 = t_i^0 = 0$
end for
$r^0 = s^0 = 0$

for $j = 1$ to J **do**
  // Update dual variables:
  for $i = 1$ to K+1 **do**
    $\boldsymbol{p}_i^{j-1/2} = \boldsymbol{p}_i^{j-1} + \sigma \nabla \bar{u}_i^{j-1}$
    $\boldsymbol{p}_i^j = \frac{\boldsymbol{p}_i^{j-1/2}}{\max\left(w_b, |\boldsymbol{p}_i^{j-1/2}|\right)}$
    $q_i^j = \max\left(0, q_i^{j-1} + \sigma\left(P\begin{pmatrix}\bar{u}_i^{j-1}\\ \bar{t}_i^{j-1}\end{pmatrix}\right)\right)$
  end for
  $r^j = r^{j-1} + \sigma\left(\sum_{i=1}^{K+1} \bar{u}_i^{j-1} - 1\right)$
  $s^j = \max\left(0, s^{j-1} + \sigma\left(\sum_{i=1}^{K} G_i \bar{u}_i^{j-1} - 1\right)\right)$

  // Update primal variables:
  for $i = 1$ to K+1 **do**
    $t_i^j = t_i^{j-1} - \tau\left(\gamma - ||q_i||\right)$
    $\bar{t}_i^j = 2t_i^j - \bar{t}_i^{j-1}$
    $u_i^j = \max\left(0, u_i^{j-1} - \tau\left(-\mathrm{div}\boldsymbol{p_i}^j + q_i^j + r^j + G_i^T s^j + \lambda f_i\right)\right)$
    $\bar{u}_i^j = 2u_i^j - \bar{u}_i^{j-1}$
  end for
end for

motion estimation and segmentation model with label costs and backmatch constraint as

$$\begin{aligned}\min_{u_i, u_i'} \left\{ \sum_i^{K+1} \left(||W_b \nabla u_i|| + \lambda \langle f_i, u_i \rangle + ||W_b' \nabla u_i'|| + \lambda \langle f_i', u_i' \rangle + \gamma ||u_i + u_i'||_\infty \right) \right\}, \\ s.t. \ \sum_{i=1}^{K+1} u_i = \mathbf{1}, \ \sum_{i=1}^{K+1} u_i' = \mathbf{1}, \text{ and } u_i \geq \mathbf{0}, \ u_i' \geq \mathbf{0}, \ \forall i = 1, \ldots, K+1, \\ \text{and } G_i' u_i' = u_i, \ G_i u_i = u_i', \ \forall i = 1, \ldots, K.\end{aligned} \tag{5.45}$$

To solve (5.45) we again transform the minimization problem into a primal-dual saddle point formulation. All necessary conversions were already introduced in Section 5.2.4.2. Therefore the saddle point formulation of (5.45) can be directly written as

$$\min_{u_i,u_i',t_i} \max_{\boldsymbol{p_i},q_i,r,s_i,\boldsymbol{p_i}',r',s_i'} \left\{ \sum_{i=1}^{K+1} \left(\langle \boldsymbol{p_i}, \nabla u_i \rangle + \lambda \langle u_i, f_i \rangle + \langle \boldsymbol{p_i}', \nabla u_i' \rangle + \lambda \langle u_i', f_i' \rangle + \gamma t_i + \right. \right.$$
$$\left. \left\langle q_i, P \begin{pmatrix} u_i + u_i' \\ t_i \end{pmatrix} \right\rangle + I_\Gamma(u_i) + I_\Gamma(u_i') - I_{\Lambda_g}(\boldsymbol{p_i}) - I_{\Lambda_{g'}}(\boldsymbol{p_i'}) - I_\Gamma(q_i) \right) +$$
$$\left. \sum_{i=1}^{K} \left(\langle s_i, G_i u_i - u_i' \rangle + \langle s_i', G_i' u_i' - u_i \rangle \right) + \left\langle r, \sum_{i=1}^{K+1} u_i - \mathbf{1} \right\rangle + \left\langle r', \sum_{i=1}^{K+1} u_i' - \mathbf{1} \right\rangle \right\}. \tag{5.46}$$

Where we additionally introduced the dual variable $\boldsymbol{p'}_i \in Y$, and the Lagrange multipliers $r \in X$ and $s_i' \in X$. Rewriting the primal-dual saddle point problem (5.46) in the form of (2.50), we get

$$\alpha = \left(\; (u_1)^T \dots (u_{K+1})^T \;\middle|\; (u_1')^T \dots (u_{K+1}')^T \;\middle|\; (t_1)^T \dots (t_{K+1})^T \; \right)^T,$$

$$D = \left(\begin{array}{ccc|ccc|ccc}
\nabla & & & & & & & & \\
& \ddots & & & & & & & \\
& & \nabla & & & & & & \\
\hline
& & & \nabla & & & & & \\
& & & & \ddots & & & & \\
& & & & & \nabla & & & \\
\hline
I & & & I & & & -\mathbf{1}^T & & \\
& \ddots & & & \ddots & & & \ddots & \\
& & I & & & I & & & -\mathbf{1}^T \\
\hline
I & \dots & I & & & & & & \\
\hline
& & & I & \dots & I & & & \\
\hline
G_1 & & \mathbf{0} & -I & & \mathbf{0} & & & \\
& \ddots & \vdots & & \ddots & \vdots & & & \\
& G_K & \mathbf{0} & & -I & \mathbf{0} & & & \\
\hline
-I & & \mathbf{0} & G_1' & & \mathbf{0} & & & \\
& \ddots & \vdots & & \ddots & \vdots & & & \\
& -I & \mathbf{0} & & G_K' & \mathbf{0} & & &
\end{array}\right) \text{ and } \beta = \left(\begin{array}{c}
\boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_{K+1} \\ \hline
\boldsymbol{p'}_1 \\ \vdots \\ \boldsymbol{p'}_{K+1} \\ \hline
q_1 \\ \vdots \\ q_{K+1} \\ \hline
r \\ \hline
r' \\ \hline
s_1 \\ \vdots \\ s_K \\ \hline
s_1' \\ \vdots \\ s_K'
\end{array}\right). \tag{5.47}$$

Further we get

$$\Phi(\alpha) = \sum_{i=1}^{K+1} \Big(\lambda \langle u_i, f_i \rangle + \lambda \langle u_i', f_i' \rangle + \gamma t_i + I_\Gamma(u_i) + I_\Gamma(u_i') \Big). \tag{5.48}$$

and

$$\Psi^*(\beta) = r + r' + \sum_{i=1}^{K+1} \left(I_{\Lambda_g}(\boldsymbol{p}_i) + I_{\Lambda_g'}(\boldsymbol{p'}_i) + I_\Gamma(q_i) \right). \tag{5.49}$$

The full algorithm to solve (5.45) is summarized in Algorithm 16. Again, the algorithm is easy to parallelize, and computations involving G_i and G_i' are carried out using sparse matrix multiplications.

5.2.5 Experimental Results

The algorithm was implemented in Matlab with the exception that the segmentation algorithms are solved on the GPU using the CUDA framework. The number of labels is only limited by memory. On the used Tesla C2050 we have 2688MB memory available. For an image size of 640×480 this currently limits our approach to 305 labels for the map uniqueness constraint and approximately the half for the backmatch constraint. The current implementation is still quite slow and ranges from a few seconds (for small images, few labels and flow initialization) up to an hour (for large images, hundreds of labels and initialization with constant values). The computationally most expensive part, is the update of the segmentation. While the algorithm is generally easy to parallelize on a pixel level, the sparse matrix multiplications with G_i and G_i^T consume more than 90% of computation time.

In Figure 5.8, we show the effects of occlusion handling on synthetic data with known transformation. It shows that the occlusion label combined with the map uniqueness or backmatch constraint delivers superior results. In some cases (e.g. top right) the backmatch constraint improves the map uniqueness constraint as it offers additional information on disocclusions.

Figure 5.9 shows a real world example comparing the map uniqueness constraint with the backmatch constraint. We can observe that there is a slight improvement when using the backmatch constraint. Additionally, we get the inverse optical flow and a segmentation for both input images.

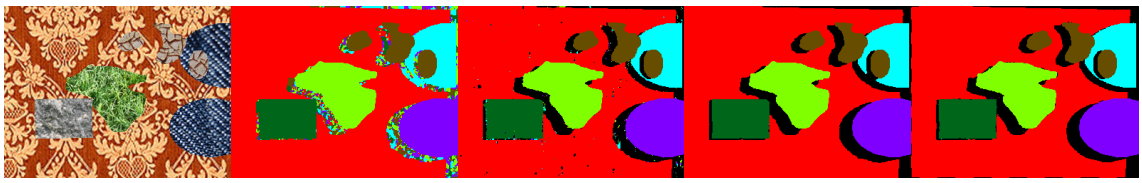
Another example (*Army*) using the backmatch constraint is depicted in Figure 5.10. See also Figure 5.15 for the results with map uniqueness constraint.

Algorithm 16 Algorithm for solving the joint motion estimation and segmentation with backmatch constraint (5.46).

// Initialization:
$\tau = \sigma = \frac{1}{L}$
for $i = 1$ to K+1 **do**
  $\boldsymbol{p}_i^0 = \boldsymbol{p'}_i^0 = (0,0)^T$
  $q_i^0 = \bar{t}_i^0 = t_i^0 = 0$
  $\bar{u}_i^0 = u_i^0 = \bar{u'}_i^0 = u'_i{}^0 = \frac{1}{K+1}$
  if $i \leq K$ **then**
    $s_i^0 = 0$
  end if
end for
$r^0 = r'^0 = 0$

for $j = 1$ to J **do**
  // Update dual variables:
  for $i = 1$ to K+1 **do**
    $\boldsymbol{p}_i^{j-1/2} = \boldsymbol{p}_i^{j-1} + \sigma \nabla \bar{u}_i^{j-1}$ and $\boldsymbol{p'}_i^{j-1/2} = \boldsymbol{p'}_i^{j-1} + \sigma \nabla \bar{u'}_i^{j-1}$
    $\boldsymbol{p}_i^j = \frac{\boldsymbol{p}_i^{j-1/2}}{\max\left(w_b, |\boldsymbol{p}_i^{j-1/2}|\right)}$ and $\boldsymbol{p'}_i^j = \frac{\boldsymbol{p'}_i^{j-1/2}}{\max\left(w'_b, |\boldsymbol{p'}_i^{j-1/2}|\right)}$
    $q_i^j = \max\left(0, q_i^{j-1} + \sigma\left(P\begin{pmatrix}\bar{u}_i^{j-1} + \bar{u'}_i^{j-1} \\ \bar{t}_i^{j-1}\end{pmatrix}\right)\right)$
  end for
  $r^j = r^{j-1} + \sigma\left(\sum_{i=1}^{K+1} \bar{u}_i^{j-1} - 1\right)$ and $r'^j = r'^{j-1} + \sigma\left(\sum_{i=1}^{K+1} \bar{u'}_i^{j-1} - 1\right)$
  for $i = 1$ to K **do**
    $s^j = s^{j-1} + \sigma\left(G_i \bar{u}_i^{j-1} - \bar{u'}_i^{j-1}\right)$
    $s'^j = s'^{j-1} + \sigma\left(\bar{u}_i^{j-1} - G'_i \bar{u'}_i^{j-1}\right)$
  end for

  // Update primal variables:
  for $i = 1$ to K+1 **do**
    $t_i^j = t_i^{j-1} - \tau\left(\gamma - ||q_i||\right)$
    $\bar{t}_i^j = 2t_i^j - \bar{t}_i^{j-1}$
    $u_i^j = \max\left(0, u_i^{j-1} - \tau\left(-\mathrm{div}\boldsymbol{p}_i^j + q_i^j + r^j + G_i^T s^j - s'^j + \lambda f_i\right)\right)$
    $u'_i{}^j = \max\left(0, u'_i{}^{j-1} - \tau\left(-\mathrm{div}\boldsymbol{p'}_i^j + q'_i{}^j + r'^j - s^j + G'_i{}^T s'^j + \lambda f'_i\right)\right)$
    $\bar{u}_i^j = 2u_i^j - \bar{u}_i^{j-1}$ and $\bar{u'}_i^j = 2u'_i{}^j - \bar{u'}_i^{j-1}$
  end for
end for



(a) Input I_1 (b) no occlusion (c) occl. label (d) map unique (e) backmatch

Figure 5.8: Comparing different types of occlusion handling: (b) Neglecting occlusions reveals noise in the segmentation within occluded regions. Adding the occlusion label (c) without an additional constraint yields nice result whereas an additional map uniqueness constraint (d) improves the result significantly.

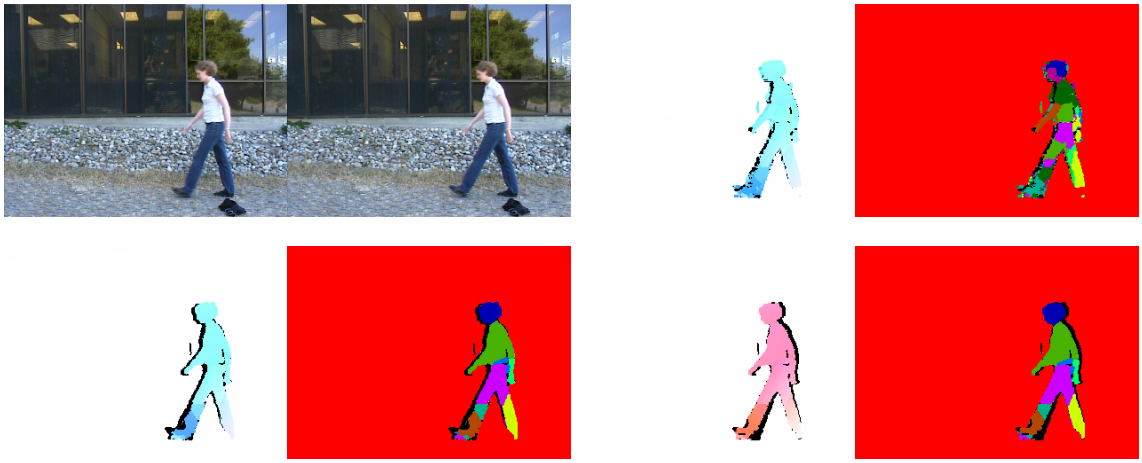


Figure 5.9: The top left images depict the input, and the top right images flow and segmentation using the map uniqueness constraint. In the bottom row, the backmatch constraint was used. We additionally obtain a segmentation and optical flow in the inverse direction.

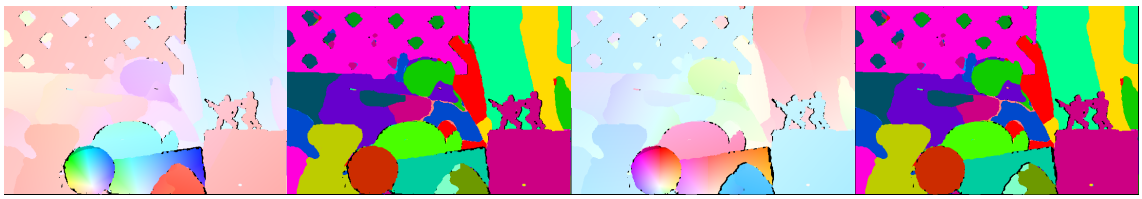


Figure 5.10: The *Army* example with backmatch constraint.

In the following we will show more experimental results on the single aspects of the presented approach.

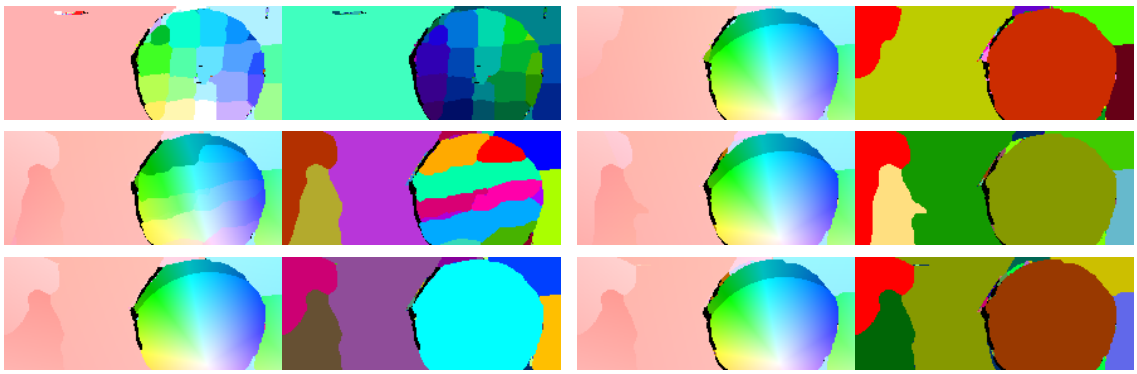


Figure 5.11: Demonstration of different initializations for the case of L1 data terms as proposed in (5.15). We used a crop of the *Army* example at iterations 1, 7 and 16: On the left, constant proposals were used as initialization. On the right, the proposals were initialized using estimates based on a precalculated optical flow. Both initializations converge against very similar results, affirming the robustness of the proposed algorithm.

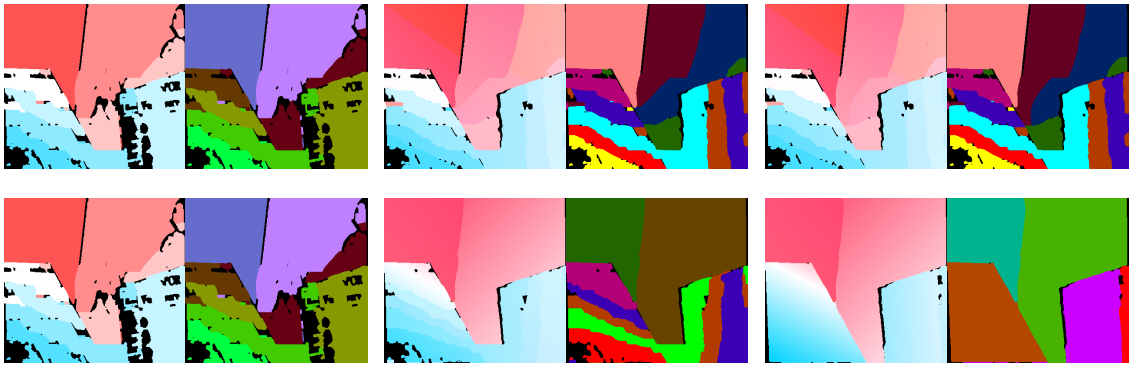
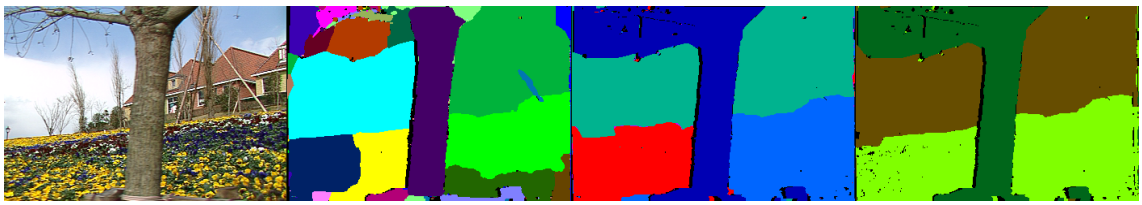


Figure 5.12: Different optimization schemes when using initialization with constant proposals. Iterations from left to right are 1, 8 and 30. The quad-fit optimization in the first row cannot handle parameters that are completely off, while this is no problem for the $L1$ optimization in the bottom row.

5.2.5.1 General evaluation with map uniqueness constraint

We compare different initialization methods in Figure 5.11. As discussed in Section 5.2.4.1, we use either constant values or a RANSAC based estimation on a precalculated flow field. Note that, after 16 iterations both initializations result in a very similar result. The more sophisticated initialization by a precalculated flow is not mandatory, but helps to speed up the overall convergence process. Therefore we will use it throughout the rest of the thesis.



(a) Input I_1 (b) $\gamma = 0$ (c) $\gamma = 0.06$ (d) $\gamma = 0.18$

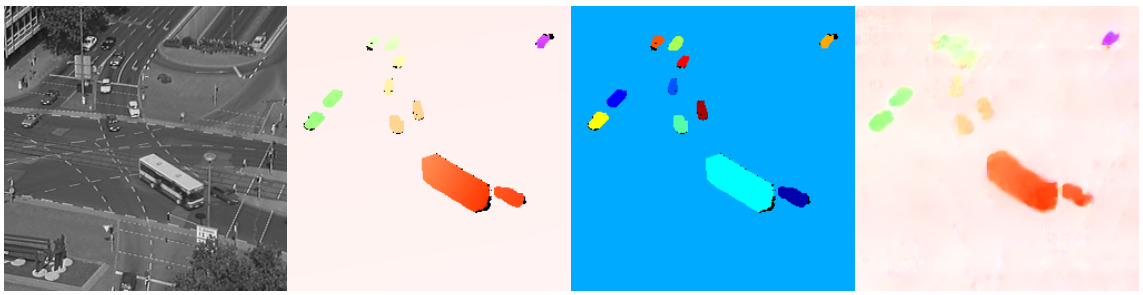
Figure 5.13: Different weights for the label cost term. With increasing γ the number of labels used to explain the scene decreases.

In Figure 5.12, we show that the $L1$ optimization scheme can successfully optimize parameters inside a region. This is especially important when using proposals with constant motions as an initialization. On the other hand, the quad-fit optimization cannot recover the correct motion estimation / segmentation. The quad-fit approach suffers from the drawback, that the dataterm is approximated locally around some current parameters. As a result only local changes can be made, and the optimization gets stuck in a local minima. Therefore the quad-fit approach always needs an initialization that is already reasonable close to the optimum. On the other hand, the $L1$ optimization scheme does not suffer from this problem and finds the globally optimum in every iteration without initialization.

In Figure 5.13, the effectiveness of the label cost term is demonstrated on the *flower-garden* sequence. With increasing γ, fewer labels are used resulting in a simplified segmentation. While we already demonstrated this effect in the previous sections, the label cost term is especially important for our motion segmentation approach. Figure 5.13(d) provides a very simplistic representation of the scene. In contrast, the segmentation with $\gamma = 0$ splits up regions due to perspective distortion of the scene.

Our method is well suited for applications like traffic analysis (see Figure 5.14). We not only get a segmentation of all moving vehicles, but also the according motion. In comparison, the gradient based continuous flow in Figure 5.14(d) is more noisy and subsequent clustering/segmentation algorithms would have problems discriminating the cars on the top left.

A large number of labels helps to make occlusion handling more efficient. We demonstrate this in Figure 5.15 by comparing our approach to the work of Sun et al. [Sun et al., 2010b]. One can clearly note that a large number of labels results in a more meaningful segmentation of the motion. As for both approaches occlusions only occur at region boundaries, a more detailed segmentation also improves the occlusion detection. More



(a) Input I_1 (b) Flow $\boldsymbol{v}$ (c) Seg. u (d) Flow by [Werlberger et al., 2009]

Figure 5.14: Motion segmentation of a traffic scene (*Bad*). We not only obtain a good segmentation of the single cars, but also their motion field.

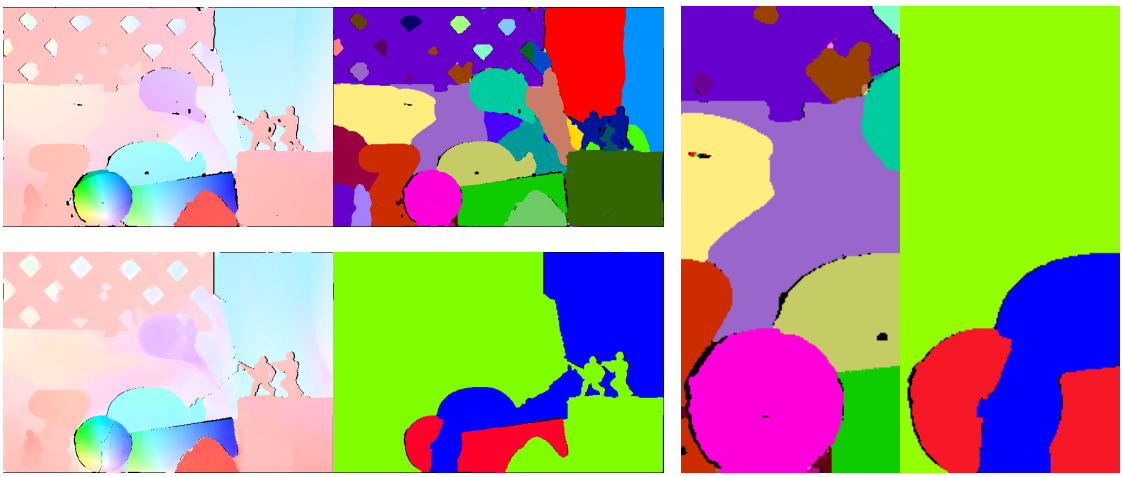
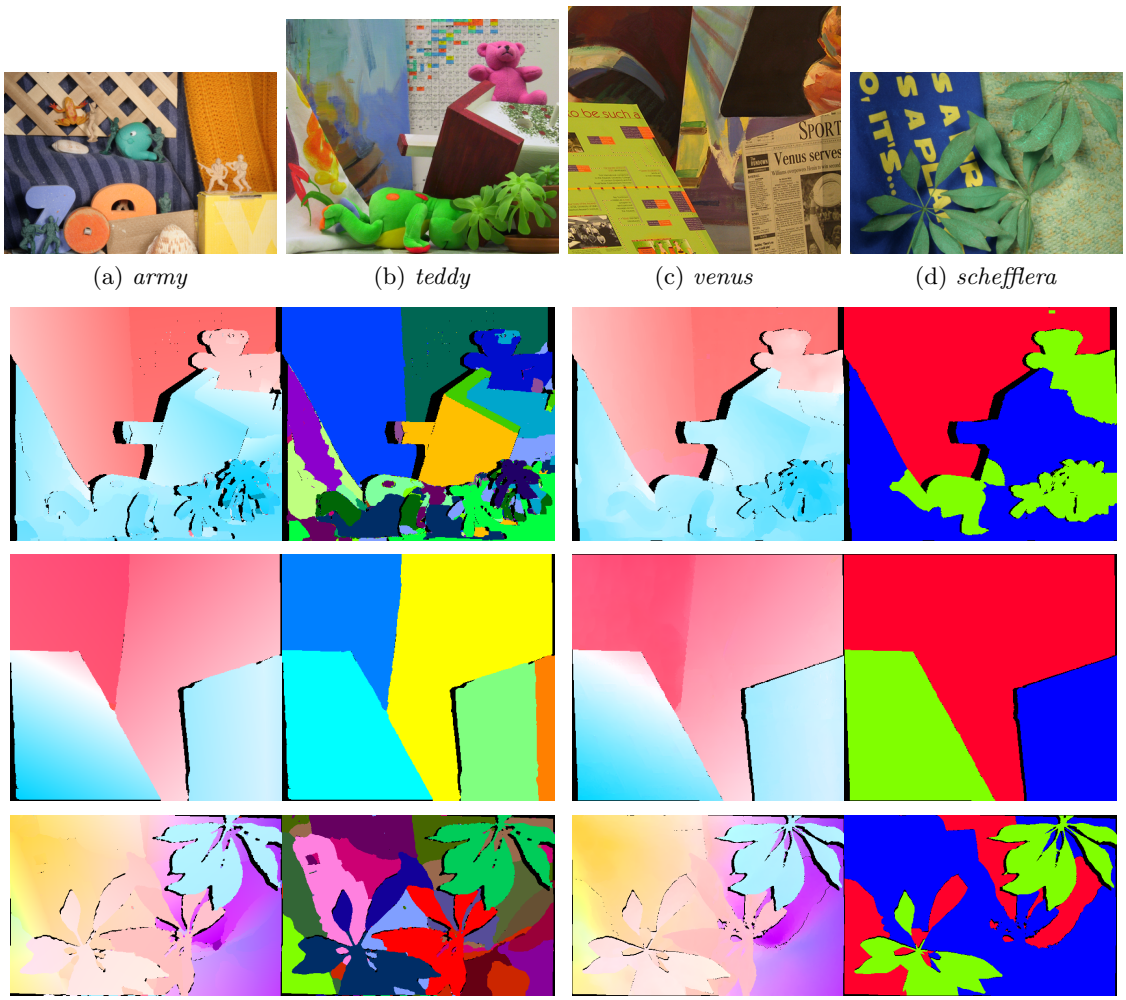


Figure 5.15: This example demonstrates that a large number of labels improves occlusion detection. Top left: our approach, Bottom left: [Sun et al., 2010b]. Right: Crops for better visualization.

comparison results can be found in Figure 5.16 for visual comparison.

The simple affine model is sometimes not enough to sufficiently describe non-rigid deformations. In Figure 5.17, we show two more examples where the proposed algorithms delivers very good results. But as seen in the *dogdance* example, complex non-rigid motions cannot be described with affine transformations efficiently. As a result, we get strong oversegmentation. Motion of small pixels is more error prone, resulting into wrong flow and wrong occlusions. In the following, we show how the proposed method can also deal with non-rigid motion.



(a) *army* (b) *teddy* (c) *venus* (d) *schefflera*

Figure 5.16: Comparison of our approach (on the left) to the approach of [Sun et al., 2010b] (on the right) using images from the Middlebury benchmark database. Our approach delivers meaningful segmentations of the scenes. Occlusion detection benefits from a large number of labels.

5.2.5.2 Continuous flow label

Looking again at the *Army* example in Figure 5.18(b,e), reveals that non-rigid deformations of the fabric cause problems. Several labels are required to explain complex motion, resulting in inaccurate optical flow and incorrect borders. While we could easily use more complex models, the simplest solution is to add an additional label that contains a precalculated optical flow. We therefore used the continuous optical flow with Huber-norm regularization from [Werlberger et al., 2009] as shown in Figure 5.18(d). In Figure 5.18(c), the motion segmentation with the additional precalculated flow label is shown. Obviously

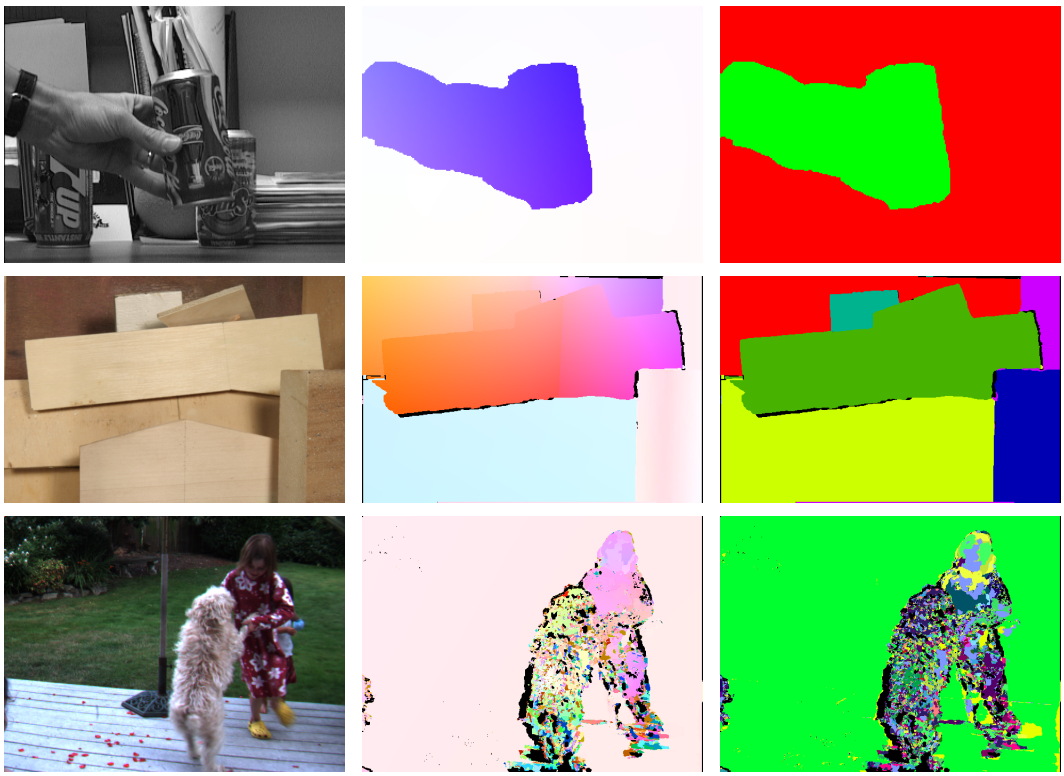


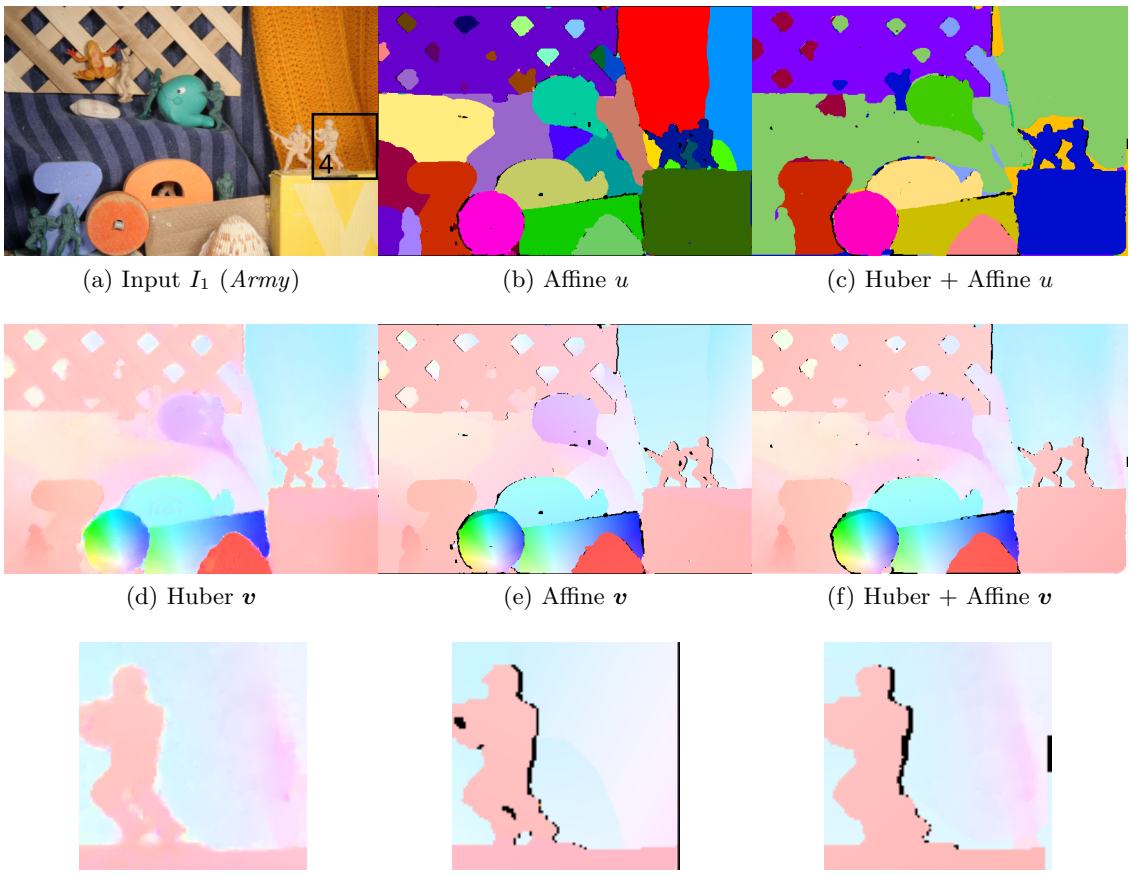
Figure 5.17: The first two scenes (*coke* and *wooden*) are perfectly suited for the proposed algorithm. The third scene (*dogdance*), cannot be described with affine transformations.

		avg.	Army			Mequon			Schefflera			Wooden		
		rank	all	disc	untex	all	disc	untex	all	disc	untex	all	disc	untex
Endpoint	**H.+Affine**	**22.2**	**0.09**	**0.24**	0.09	0.27	**0.74**	0.26	**0.28**	**0.58**	0.31	**0.18**	**1.21**	**0.08**
	Huber	28.4	**0.09**	0.28	**0.07**	**0.21**	0.81	**0.16**	0.54	1.22	**0.22**	0.25	1.38	0.14
Angular	**H.+Affine**	**23.4**	3.73	**9.43**	3.61	3.51	**8.70**	3.70	**4.11**	**9.52**	5.28	**3.14**	**19.6**	**1.53**
	Huber	31.0	**3.57**	10.5	**2.89**	**2.80**	11.4	**2.18**	7.23	17.8	**3.14**	4.68	23.8	2.54

	Grove			Urban			Yosemite			Teddy		
	all	disc	untex	all	disc	untex	all	disc	untex	all	disc	untex
	0.88	**1.32**	**0.48**	1.79	1.97	0.99	**0.11**	**0.15**	**0.15**	**0.74**	**1.52**	**1.02**
	1.15	1.50	1.35	**0.45**	**1.69**	**0.32**	0.20	0.19	0.26	0.88	1.88	1.03
	3.58	**4.66**	**2.49**	5.98	17.7	5.96	**2.31**	**4.34**	**1.36**	**3.77**	**8.10**	5.09
	4.05	5.20	3.28	**4.59**	**16.7**	**3.47**	4.37	5.56	2.84	5.32	12.4	**3.83**

Table 5.1: Evaluation of the proposed joint motion estimation / segmentation approach with the precalculated optical flow (Huber + Affine) on the Middlebury benchmark. The second row provides a comparison to the precalculated optical flow alone (Huber).

the new label is used to explain the non-rigid motions of the fabric while regions where the affine model fits well are segmented with separate labels. Around borders and occlusions new labels are introduced, because in standard gradient based optical flow object borders and occlusions cause artifacts.



(a) Input I_1 (*Army*) (b) Affine u (c) Huber + Affine u

(d) Huber $\boldsymbol{v}$ (e) Affine $\boldsymbol{v}$ (f) Huber + Affine $\boldsymbol{v}$

Figure 5.18: Comparison of continuous Huber regularized optical flow [Werlberger et al., 2009] (d), with the proposed affine motion segmentation model (b,e) and the motion segmentation with precalculated flow label (c,f).

In Figure 5.19, we show the improvement for the *dogdance* example, that suffered from oversegmentation in Figure 5.17. While this is still not a very good segmentation, we do not suffer from nearly as much oversegmentation. As the Huber regularized optical flow is not perfect either, pixels where no model fits are labeled as occlusion.

By using the combined motion estimation and segmentation model with a precalculated flow label, we can improve the flow in regions where the affine models better fits the data. We evaluated this method on the Middlebury benchmark [Baker et al., 2010]. As the benchmark does not take into account occlusions, we inpainted occlusions using the ROF model [Rudin et al., 1992]. Table Table 5.1 shows that the combination of the proposed joint motion estimation / segmentation approach with flow label significantly improves the the precalculated optical flow alone.

Figure 5.19: The (*dogdance*) with the Huber regularized optical flow on the left, and the proposed model on the right.

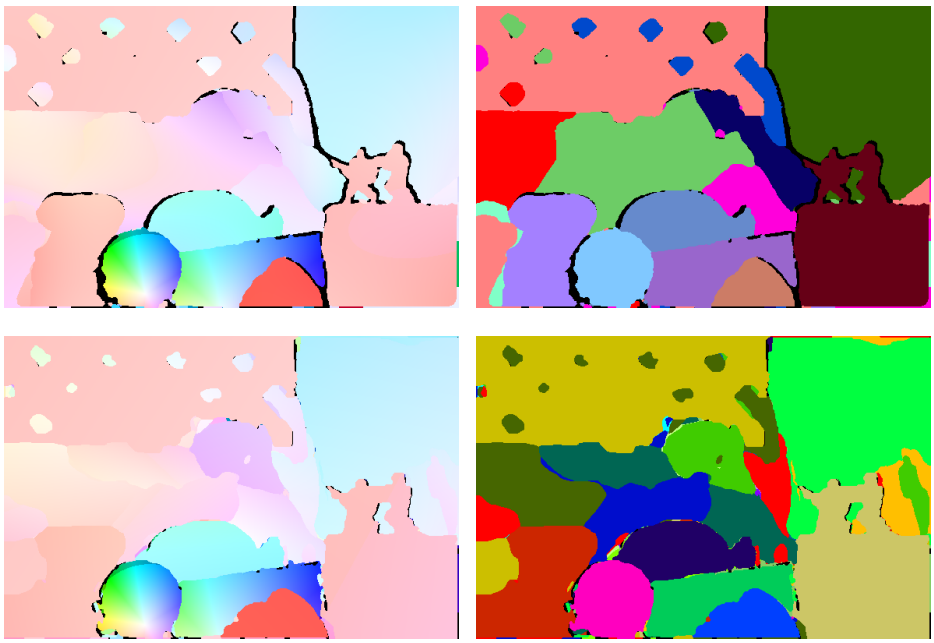


Figure 5.20: The *Army* example with map uniqueness constraint. Quadratic parametrization is depicted in the top row. The NCC dataterm is depicted in the bottom row.

5.2.5.3 Extensions and variants

In the following we are going to show two more experimental results. First, the model with map uniqueness constraint is used together with quadratic parametrization as in (5.21). The result using the *Army* example is depicted in the top row of Figure 5.20. One can notice a slight improvement in non-planar regions. In contrast to the continuous flow label as shown in Section 5.2.5.2, there is no clear benefit. Although the quadratic terms could improve over simple affine models, we recommend the use of continuous flow label that simply offers more flexibility.

As a second experiment the bottom row of Figure 5.20 depicts results when using the NCC dataterm as described in (5.16). The NCC implicitly performs some form of regularization (we used a 5×5 kernel). As a result fine details are often lost in the final result. This makes us confident that a pixel based data fidelity term is sufficient for a wide range of input data. Especially as the affine parametrization of the optical flow already provides a very strong prior that can deal with a noisy data fidelity term.

5.2.6 Summary

In this section, we presented an approach for joint motion estimation and segmentation based on a continuous Potts like model (see Section 3.2.1) with occlusion handling and label costs. The approach can be used for a very large number of layers resulting in a meaningful segmentation of the scene. The large number of layers clearly improves occlusion handling, while the actual number of labels is determined automatically by the algorithm. Parameter optimization can be solved with an off-the-shelf linear program. We also eliminate classical problems of optical flow calculation such as local boundary blurring and artifacts caused by occlusions. Experiments show that our approach is robust to different initializations and delivers good results for a wide range of images.

However, the presented approach also has some drawbacks. First, the method is still too slow for many practical purposes. To remedy this, on the fly calculation of the map uniqueness or backmatch constraint could eliminate the costly sparse matrix multiplication and speed up the algorithm significantly. Second, the strong prior imposed by the parametric flow is not the optimal choice for many natural scenes with non-rigid deformations. As we have shown with the additional label with precalculated gradient based flow, this problem can be circumvented at the cost of a mixed representation.

Chapter 6

Conclusion

6.1 Summary

This thesis shows, that continuous convex optimization offers a powerful framework for segmentation related tasks and image processing in general. This strength comes from several properties:

First, energy minimization methods in general provide a very structured way of tackling a problem. Designing the energy already defines the final properties of the solution, thus making the result predictable. As long as the energy is convex, we are able to find the globally optimal solution in a reasonable time. Unfortunately, non-convex energies are very hard to deal with when doing large scale optimization problems typically found in computer vision. Although, non-convex energies allow for more flexible problem formulations, they are no longer guaranteed to find the globally optimal solution in a tractable time. As a result, there is always the tradeoff between using an inferior energy formulation and finding the exact solution, or using a superior energy and have no guarantee to come close to the solution. We have seen that for a lot of non-convex problems we can find good convex relaxations, that either give use the real globally optimal solution of the original formulation, or at least come very close to the solution. If we are able to find such a good convex relaxation this offers a good tradeoff between constraints on the energy design, and finding the optimal solution.

Another advantage of continuous convex optimization is the idea of continuous functions itself. For the implementation, we cannot work directly with continuous functions, but have to work with a discretized formulation. As a result, there is no fundamental difference between continuous and discrete optimization formulations. The real changes

comes from thinking either in continuous functions or a discrete grid. We have seen that the only difference between the discrete maximum flow problem and the continuous maximum flow is a point-wise L^1 or L^2 norm. The L^2 norm computes the real Euclidean distance and thus does not suffer from any grid bias. However this renders the problem more difficult.

A big advantage of continuous convex optimization approaches is the availability of powerful optimization algorithms like the primal-dual algorithm we used. This allows not only to solve problems like the discrete graph cut problem, but also the continuous maximum flow formulation. The primal-dual formulation has the big advantage, that it is able to deal with non-smooth functions like the Total Variation. As demonstrated we can efficiently optimize various segmentation problems based on the TV, even if the optimization problems are very complex. The resulting algorithms proved to be perfectly suited for parallel hardware, as they can be parallelized on a pixel level and are therefore perfectly suited for modern GPUs. The combination of fast algorithms and fast implementations makes the proposed approaches very well suited for practical application.

We have introduced several different segmentation models. These ranged from the simple two label image segmentation task, to multi-label segmentation with label costs and more complex applications. The tracking or video segmentation approach shows that segmentation can also be used in currently rather unconventional ways. Using the algorithms for supervised segmentation demonstrated that the algorithms are fast enough to be used in an interactive fashion. We also showed that the models can be extended to unsupervised segmentation models as seen for the depth image segmentation or the joint motion estimation and segmentation.

6.2 Outlook

Although, the presented algorithms already show some promising results, there is still a lot of room for improvement and future work.

Using the Total Variation, the contour length of the segmentation is penalized. Combined with the edge weighting of the binary terms, the weighted TV is a very good segmentation prior. But for some occasions one could think of more complex priors, like using the non-local TV as in [Werlberger et al., 2011b]. There is also a lot of work going on in penalizing the curvature of the contour [Bredies et al., 2012; Schoenemann et al., 2012]. Globally optimal curvature regularization is still limited in terms of speed and discretization of directions. However this could represent a great potential of improving

segmentation quality.

As we focused on the segmentation models and optimization, we did not talk much about the optimal features for image segmentation. Although, color information is sufficient for a lot of images, it sometimes is not enough. As shown in [Santner et al., 2011, 2009], the proposed methods can be easily extended with complex features when combined with various learning methods. This also makes it possible to segment complex textures in images. One could also think of more complex models for the labels (e.g. as we did in the unsupervised examples). This could e.g. be color gradients, that were combined with GMMs in the inspiring work of [Delong et al., 2011]. They not only presented a MRF that combined different type of models for an label, but also showed a way to use superpixels combined with local models. Local models allow to use simple features like color in a reasonable spatial neighborhood to segment heavily textured and cluttered images. There is certainly a lot of potential for interactive segmentation in these methods.

On an algorithmic side, the ROF model used for solving the continuous maximum flow problem, showed excellent runtimes. The reason is that we are able to use algorithmic speedups if at least one part of the energy is a smooth function. It would be worth to investigate whether similar methods for multi-label segmentation are possible. Furthermore, the preconditioned primal-dual algorithm of [Pock and Chambolle, 2011] would speed up convergence when using strong weights in the binary terms.

For the applications presented in this thesis we already gave a summary and short outlook at each section. We would like to stress once more the idea of extending the spatio-temporal video segmentation approach by using flow guided gradients. This will certainly lead to improvements of the segmentation, and could be used for fast and precise video segmentation.

Appendix A

Segmentation comparison results

A.1 Global relabeling results

In Table A.1, we summarized the experimental results visualized in Figure 3.13. See Section 3.3.2.3 for a full description of the experimental settings.

Runtimes corresponding to Figure 3.13(i)						
		256×256	512×512	1024×1024	2048×2048	3200×3200
4N	pd	0.25986	0.8439	7.8765	18.0204	66.1176
	pdgrl	0.17647	0.7579	4.9137	16.6636	61.1402
	byokov	0.27618	2.6130	17.1329	241.5678	897.8244
	npp	0.06467	0.2887	2.4546	37.9883	176.8301
8N	pd	0.16892	0.7308	4.0466	45.4415	146.5739
	pdgrl	0.11213	0.7115	3.0100	45.3134	130.5741
	byokov	0.41579	3.6015	40.7945	1841.6567	7775.3371
TV	pd	1.80500	7.0344	34.7248	38.0857	83.5642
	pdgrl	0.56535	1.0043	4.2760	21.8553	83.6442

Runtimes corresponding to Figure 3.13(j)						
		256×256	512×512	1024×1024	2048×2048	3200×3200
4N	pd	0.0148	0.0956	1.3497	14.5545	45.7615
	pdgrl	0.0178	0.0826	0.9559	6.4206	27.0064
	byokov	0.0322	0.3544	3.4546	26.3678	59.0944
	npp	0.0128	0.0259	0.1127	0.6225	2.4497
8N	pd	0.0198	0.1081	0.8148	6.8776	51.9962
	pdgrl	0.0273	0.1077	0.8157	6.6190	31.7820
	byokov	0.0550	0.5484	6.5967	85.0985	319.0546
TV	pd	0.0073	0.0466	0.2107	1.9537	4.5053
	pdgrl	0.0074	0.0465	0.2106	1.8365	4.5925
Runtimes corresponding to Figure 3.13(k)						
		256×256	512×512	1024×1024	2048×2048	3200×3200
4N	pd	0.050043	0.21313	1.0455	7.2184	32.5055
	pdgrl	0.040377	0.09411	0.4986	6.9881	28.5708
	byokov	0.090662	0.20354	1.5960	13.6675	26.8015
	npp	0.023544	0.04798	0.1664	1.8905	3.2300
8N	pd	0.045491	0.94154	1.6263	8.0988	44.3078
	pdgrl	0.061445	0.38124	1.3292	11.9545	28.5320
	byokov	0.097347	0.49469	4.9993	52.9113	235.9746
TV	pd	0.025698	0.25456	0.4635	1.7904	6.5542
	pdgrl	0.038327	0.24185	0.4836	1.7904	6.5532
Runtimes corresponding to Figure 3.13(l)						
		256×256	512×512	1024×1024	2048×2048	3200×3200
4N	pd	0.037088	0.14210	1.0871	30.025	114.8381
	pdgrl	0.029001	0.15337	1.1293	10.5521	50.5208
	byokov	0.026275	0.18502	2.1537	36.6416	302.8063
	npp	0.025631	0.06132	0.5238	9.5882	92.8287
8N	pd	0.026056	0.17251	1.1477	13.6945	247.4028
	pdgrl	0.033199	0.18468	1.0040	14.1786	106.9219
	byokov	0.059929	0.39582	5.0149	167.842	1395.7276
TV	pd	0.213610	0.40766	1.2638	7.4865	26.6235
	pdgrl	0.267360	0.18590	1.2832	7.4143	26.7953

Table A.1: Runtimes (in seconds) for various segmentation algorithms. The top row depicts the size of the input images. The data of this table corresponds to Figure 3.13.

A.2 Continuous segmentation results

This Section provides the full data of the experiments of Section 3.3.4. Table A.2 list the results when using discrete constraints. Table A.3 presents the data when the constraints are allowed to vary continuously.

TV-L1						
	0.002	**0.005**	**0.010**	**0.040**	**0.075**	**0.120**
man	35180	5960	6660	1220	1320	760
starfish	47600	4080	19260	4880	4340	600
horses	22000	37640	10720	1460	6760	580
elephant	16720	12820	23920	1760	8560	2420
flowers	17900	7800	1360	900	3400	1020
rabbit	46320	36740	29540	4840	2900	2040
wulfenia	20860	22980	18560	2000	2420	1740
zebra	43880	5820	4360	5200	10380	1420
TV-L1 threshold						
	0.002	**0.005**	**0.010**	**0.040**	**0.075**	**0.120**
man	33200	5640	6300	1160	1260	700
starfish	44880	3840	18160	4600	4300	660
horses	20740	35320	10100	1380	6540	560
elephant	15760	12100	22840	1660	8080	2300
flowers	16900	7400	1280	860	3540	1000
rabbit	43680	34640	27880	4560	2740	1780
wulfenia	19660	21700	17800	1900	2280	1640
zebra	41360	5480	4120	5200	9920	1460

Max-flow						
	0.002	**0.005**	**0.010**	**0.040**	**0.075**	**0.120**
man	33160	5640	6300	1160	1260	700
starfish	44880	3840	18160	4460	4280	640
horses	20740	36800	10100	1380	6540	560
elephant	15200	12100	22840	1640	7800	2460
flowers	16620	7400	1300	860	3540	1000
rabbit	43760	34640	27580	4560	2740	1900
wulfenia	19660	20940	17800	1900	2280	1640
zebra	41360	5480	4120	5200	9920	1440
Max-flow with grl						
	0.002	**0.005**	**0.010**	**0.040**	**0.075**	**0.120**
man	4440	7520	5900	1160	1260	700
starfish	43920	3840	18160	4460	4420	1380
horses	20740	36800	10100	1380	6540	560
elephant	22680	12660	23700	1640	7800	2460
flowers	16620	7400	1300	860	3540	1020
rabbit	41200	35700	17660	4560	2740	1900
wulfenia	48020	33460	17800	1900	2280	1640
zebra	42120	7020	17140	5200	9920	1440
ROF						
	0.002	**0.005**	**0.010**	**0.040**	**0.075**	**0.120**
man	3160	700	700	1600	120	260
starfish	2860	1140	520	280	100	100
horses	3560	1040	580	620	900	80
elephant	3140	4000	2700	780	720	940
flowers	2760	780	1020	2300	260	320
rabbit	3540	1900	1160	780	500	220
wulfenia	7420	3280	2920	780	520	2480
zebra	5100	1180	1100	720	100	220

Table A.2: Segmentation comparison results using the discrete constraints. The top row depicts the varying λ value, and the left column the name of the current image. Numbers of the table state the number of iterations until all pixels equaled the ground truth. The data of this table corresponds to Figure 3.15

Max-flow									
	0.1	**0.5**	**2.0**	**4.0**	**6.0**	**8.0**	**10.0**	**15.0**	**20.0**
man	27680	6420	6840	4020	3120	4120	5180	9680	4520
starfish	36260	8760	16480	10880	12620	15040	18600	45940	1420
flowers	71720	12900	15400	2820	4800	17560	4860	860	1920
zebra	23020	11320	3000	2520	11700	1280	1700	300	580
horses	63320	4900	15340	3140	5020	1360	1600	6300	1740
elephant	57020	64320	35120	11400	46400	8140	6520	4000	50880
rabbit	57720	6200	2580	6800	14420	2160	11980	5100	1400
wulfenia	16300	8340	5640	16860	4200	4740	1640	760	1160
lady	13760	18740	4100	4220	18420	4560	1540	3860	1440
liver1	27240	5440	6300	4480	4760	12200	15540	4200	1460
Max-flow with grl									
	0.1	**0.5**	**2.0**	**4.0**	**6.0**	**8.0**	**10.0**	**15.0**	**20.0**
man	27700	6740	6840	4020	3120	4120	5180	9680	4520
starfish	36760	8760	16480	10880	12620	15040	18600	45940	1420
flowers	71720	12900	15400	2820	4800	17560	4860	880	1920
zebra	32640	11320	3000	2520	11700	1280	1700	300	600
horses	63740	4900	15340	3140	5020	1360	1600	6300	1740
elephant	58300	65640	35120	11400	46400	8140	6520	4000	50880
rabbit	60920	3680	2580	6800	14420	2160	11680	5100	1420
wulfenia	23720	8340	5640	16860	4200	4740	1640	760	1160
lady	14200	18740	4100	4220	18440	4580	1540	3880	1440
liver1	11900	5620	6300	4480	4760	12200	15540	4200	2780
ROF									
	0.1	**0.5**	**2.0**	**4.0**	**6.0**	**8.0**	**10.0**	**15.0**	**20.0**
man	500	340	360	160	360	160	100	120	120
starfish	1880	280	400	80	100	120	380	80	100
flowers	1760	820	320	200	160	160	140	60	60
zebra	1420	760	440	500	320	340	260	220	220
horses	1460	680	300	260	380	260	220	260	480
elephant	3200	1260	4900	1380	1080	1580	640	1500	480
rabbit	1500	1020	1040	600	340	680	320	240	420
wulfenia	2360	1260	1000	240	200	240	140	440	320
lady	520	380	160	80	140	140	120	100	80
liver1	560	260	100	140	1040	120	120	100	200

Table A.3: Segmentation comparison results using continuous constraints. The top row depicts the varying λ value, and the left column the name of the current image. Numbers of the table state the number of iterations until all pixels equaled the ground truth. The data of this table corresponds to Figure 3.16

Appendix B

Acronyms and Symbols

List of Acronyms

ACWE	active contours without edges
CCMF	combinatorial continuous max-flow
CPU	central processing unit
CRF	conditional random field
CT	computed tomography
CUDA	compute unified device architecture
DSM	digital surface model
FISTA	fast iterative shrinkage thresholding algorithm
FPGA	field programmable gate array
GAC	geodesic active contours
GMM	gaussian mixture model
GPU	graphics processing unit
GPGPU	general-purpose computing on graphics processing units
ISTA	iterative shrinkage thresholding algorithm
LF	Legendre-Fenchel
LP	linear program
MAP	maximum a posteriori
MCMC	Markov Chain Monte Carlo
MRF	Markov random field
MSER	maximally stable extremal region
NCC	normalized cross correlation

OFC	optical flow constraint
OpenCL	open computing language
PCA	principal component analysis
PDE	partial differential equation
ROF	Rudin, Osher and Fatemi
SIMD	single instruction, multiple data
SISD	single instruction, single data
SOR	successive over relaxation
SSD	sum of squared differences
SVM	support vector machine
TNCC	truncated normalized cross correlation
TV	Total Variation
TV-L1	Total Variation regularization with L1 dataterm

List of Symbols

$< x, y >$	Tensor product
$\mathbf{I}$	Identity matrix
$f^*(y)$	Conjugate
$f^{**}(x)$	Biconjugate
$\mathcal{C}^k$	The k derivatives exist and are continuous
∇	Nabla (gradient) operator
div	Divergence operator
$\triangle$	Laplace operator
δ_x^+	Forward differences in x direction
δ_y^-	Backward differences in y direction
$\lvert x \rvert$	Absolute value of x
$\lfloor x \rfloor$	Floor of x
$[x]_a^b$	Clamping of x to the interval $[a, b]$
$I_\Sigma(x)$	Indicator function for set Σ
$\lvert\lvert \cdot \rvert\rvert_{q,p}$	Inner L^q-norm and outer L^p-norm
$\lvert\lvert \cdot \rvert\rvert_p$	Inner L^2-norm and outer L^p-norm ($\lvert\lvert \cdot \rvert\rvert_{2,p}$)
$\lvert\lvert \cdot \rvert\rvert$	Inner L^2-norm and outer L^1-norm ($\lvert\lvert \cdot \rvert\rvert_{2,1}$)
Ω	Image domain

Bibliography

Adams, R. and Bischof, L. (1994). Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647.

Akin, A., Beretta, I., Nacci, A. A., Rana, V., Santambrogio, M. D., and Atienza, D. (2011). A High-Performance Parallel Implementation of the Chambolle Algorithm. In *IEEEACM 2011 Design Automation and Test in Europe Conference*, pages 7–12.

Alliney, S. (1992). Digital filters as absolute norm regularizers. *IEEE Transactions on Signal Processing*, 40(6):1548–1562.

Alliney, S. (1997). A property of the minimum vectors of a regularizing functional defined by means of the absolute norm. *IEEE Trans Signal Processing*, 45(4):913–917.

Ambrosio, L. and Tortorelli, V. M. (1990). Approximation of functionals depending on jumps by elliptic functionals via Γ-convergence. *Communications on Pure and Applied Mathematics*, XLIII(8):999–1036.

Appleton, B. and Talbot, H. (2005). Globally Optimal Geodesic Active Contours. *Journal of Mathematical Imaging and Vision*, 23(1):67–86.

Appleton, B. and Talbot, H. (2006). Globally minimal surfaces by continuous maximal flows. *IEEE transactions on pattern analysis and machine intelligence*, 28(1):106–18.

Aubert, G. and Blanc-Féraud, L. (1998). An elementary proof of the equivalence between 2D and 3D classical snakes and geodesic active contours.

Aubert, G. and Blanc-Féraud, L. (1999). Some Remarks on the Equivalence between 2D and 3D Classical Snakes and Geodesic Active Contours. *International Journal of Computer Vision*, 34(1):19–28.

Aujol, J. and Kang, S. (2006). Color image decomposition and restoration. *Journal of Visual Communication and Image Representation*, 17(4):916–928.

Aujol, J.-F., Gilboa, G., Chan, T., and Osher, S. (2006). Structure-Texture Image Decomposition - Modeling, Algorithms, and Parameter Selection. *International Journal of Computer Vision*, 67(1):111–136.

Avidan, S. (2005). Ensemble tracking. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 494–501.

Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M., and Szeliski, R. (2010). A Database and Evaluation Methodology for Optical Flow. *International Journal of Computer Vision*, pages 1–31.

Bayes, M. and Price, M. (1763). An Essay towards Solving a Problem in the Doctrine of Chances. By the Late Rev. Mr. Bayes, communicated by Mr. Price, in a letter to John Canton, M. A. and F. R. S. *Philosophical Transactions (1683-1775)*.

Beck, A. and Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183.

Berkels, B. (2010). *Joint methods in imaging based on diffuse image representations*. PhD thesis, Friedrich-Wilhelms-University Bonn.

Bertero, M. and Boccacci, P. (1998). *Introduction to Inverse Problems in Imaging*. Taylor & Francis.

Bhusnurmath, A. and Taylor, C. J. (2008). Graph cuts via l1 norm minimization. *IEEE transactions on pattern analysis and machine intelligence*, 30(10):1866–71.

Bibby, C. and Reid, I. (2008). Robust Real-Time Visual Tracking Using Pixel-Wise Posteriors. In *Proc. European Conference on Computer Vision*, volume 2, pages 831–844.

Black, M. J. (1991). Robust dynamic motion estimation over time. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Bleyer, M. and Gelautz, M. (2004). A layered stereo algorithm using image segmentation and global visibility constraints. In *IEEE International Conference on Image Processing*, pages 2997–3000.

Bleyer, M., Rother, C., and Kohli, P. (2010). Surface stereo with soft segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 1570–1577.

Bleyer, M., Rother, C., Kohli, P., Scharstein, D., and Sinha, S. (2011). Object Stereo - Joint Stereo Matching and Object Segmentation. In *Computer Vision and Pattern Recognition*, number 1, pages 3081–3088.

Blomgren, P. and Chan, T. F. (1998). Color TV: Total variation methods for restoration of vector valued images. *IEEE Trans Image Proc*, 7:304–309.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Boykov, Y. and Kolmogorov, V. (2003). Computing geodesics and minimal surfaces via graph cuts. In *Ninth IEEE International Conference on Computer Vision*, pages 26–33 vol.1.

Boykov, Y. and Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, 26(9):1124–37.

Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast Approximate Energy Minimization via Graph Cuts. *IEEE transactions on pattern analysis and machine intelligence*, 23(11):1222–1239.

Bredies, K., Kunisch, K., and Pock, T. (2010). Total Generalized Variation. *SIAM Journal on Imaging Sciences*, 3(3):492–526.

Bredies, K., Pock, T., and Wirth, B. (2012). Convex relaxation of a class of vertex penalizing functionals. *preprint (gpu4vision.org)*.

Bresson, X. and Chan, T. (2008). Fast dual minimization of the vectorial total variation norm and applications to color image processing. *Inverse Problems and Imaging*, 2(4):455–484.

Bresson, X., Esedoglu, S., Vandergheynst, P., Thiran, J.-P., and Osher, S. (2007). Fast Global Minimization of the Active Contour/Snake Model. *Journal of Mathematical Imaging and Vision*, 28(2):151–167.

Brox, T. and Malik, J. (2010). Object Segmentation by Long Term Analysis of Point Trajectories. *IEEE European conference on computer vision*.

Brox, T. and Malik, J. (2011). Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):500–513.

Brox, T., Rousson, M., Deriche, R., and Weickert, J. (2010). Colour, texture, and motion in level set based segmentation and tracking. *Image and Vision Computing*, 28(3):376–390.

Caselles, V., Kimmel, R., and Sapiro, G. (1997a). Geodesic active contours. *International Journal of Computer Vision*, 1(22):61–79.

Caselles, V., Kimmel, R., Sapiro, G., and Sbert, C. (1997b). Minimal surfaces : a geometric three dimensional segmentation approach. *Numerische Mathematik*, 77(4):423–425.

Chambolle, A. (2000). Inverse Problems in Image processing and Image segmentation: some mathematical and numerical aspects. Technical report, Lecture notes given at the School on Mathematical Problems in Image Processing.

Chambolle, A. (2004). An Algorithm for Total Variation Minimization and Applications. *Journal of Mathematical Imaging and Vision*, 20(1/2):89–97.

Chambolle, A. (2005). Total Variation Minimization and a Class of Binary MRF Models. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, number 1, pages 136–152.

Chambolle, A., Caselles, V., Novaga, M., Cremers, D., and Pock, T. (2009). An introduction to Total Variation for Image Analysis. Technical report, Summer School on "Theoretical Foundations and Numerical Methods for Sparse Recovery", Linz, Austria.

Chambolle, A., Cremers, D., and Pock, T. (2008). A convex approach for computing minimal partitions. Technical report, Ecole Polytechnique Centre De Mathematiques Appliquees UMR CNRS 7641.

Chambolle, A., Cremers, D., and Pock, T. (2012). A convex approach to minimal partitions (revised version). *Preprint (gpu4vision.org)*.

Chambolle, A. and Darbon, J. (2009). On Total Variation Minimization and Surface Evolution Using Parametric Maximum Flows. *International Journal of Computer Vision*, 84(3):288–307.

Chambolle, A. and Lions, P. L. (1997). Image recovery via total variation minimization and related problems. *Numerische Mathematik*, 76(2):167–188.

Chambolle, A. and Pock, T. (2010). A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging. *Journal of Mathematical Imaging and Vision*, pages 1–26–26.

Chan, T. (2000). Active Contours without Edges for Vector-Valued Images. *Journal of Visual Communication and Image Representation*, 11(2):130–141.

Chan, T. and Vese, L. (1999). An Active Contour Model without Edges. *Scale-Space Theories in Computer Vision*, 1682(1682):141–151.

Chan, T. F. and Esedoglu, S. (2005). Aspects of Total Variation Regularized L1 Function Approximation. *SIAM Journal on Applied Mathematics*, 65(5):1817.

Chan, T. F., Esedoglu, S., and Nikolova, M. (2006). Algorithms for Finding Global Minimizers of Image Segmentation and Denoising Models. *SIAM Journal on Applied Mathematics*, 66(5):1632.

Chan, T. F., Kang, S. H., and Shen, J. (2001). Total Variation Denoising and Enhancement of Color Images Based on the CB and HSV Color Models. *Journal of Visual Communication and Image Representation*, 12(4):422–435.

Chan, T. F. and Vese, L. A. (2001). Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277.

Cohen, I., Cohen, L. D., and Ayache, N. (1992). Using Deformable Surfaces to Segment 3-D Images and Infer Differential Structures. *Comput Vision Graphics Image Process Image Understand*, 56(2):242–263.

Cohen, L. D. (1991). On active contour models and balloons. *CVGIP Image Understanding*, 53(2):211–218.

Cohen, L. D. and Cohen, I. (1993). Finite-element methods for active contour models and balloons for 2-D and 3-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1131–1147.

Comaniciu, D., Meer, P., and Member, S. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619.

Comaniciu, D., V., R., and Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 142–149.

Couprie, C. (2011). *Graph-based variational optimization and applications in computer vision*. PhD thesis, Universitee de Marne-la-Vallee, ESIEE Paris.

Couprie, C., Grady, L., Najman, L., and Talbot, H. (2011a). Power Watershed : A Unifying Graph-Based Optimization Framework. *IEEE Trans. on Pattern Analysis and Machine Intelligence*.

Couprie, C., Grady, L., Talbot, H., and Najman, L. (2011b). Combinatorial continuous maximum flow. *SIAM Journal on Imaging Sciences*, 4(3).

Cremers, D. (2003). A variational framework for image segmentation combining motion estimation and shape regularization. In *Conference on Computer Vision and Pattern Recognition*, volume 1, pages 53–58.

Cremers, D. (2007). Bayesian Approaches to Motion-Based Image and Video Segmentation. *Complex Motion*, 3417:106–125.

Cremers, D., Rousson, M., and Deriche, R. (2007). A Review of Statistical Approaches to Level Set Segmentation: Integrating Color, Texture, Motion and Shape. *International Journal of Computer Vision*, 72:195–215.

Cremers, D. and Soatto, S. (2005). Motion Competition: A Variational Approach to Piecewise Parametric Motion Segmentation. *International Journal of Computer Vision*, 62(3):249–265.

Delong, A. (2011). *Advances in Graph-Cut Optimization: Multi-Surface Models, Label Costs, and Hierarchical Costs*. PhD thesis, University of Western Ontario.

Delong, A., Gorelick, L., Schmidt, F. R., Veksler, O., and Boykov, Y. (2011). Interactive Segmentation with Super-Labels. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*.

Dixit, N., Keriven, R., and Paragios, N. (2005). GPU-Cuts : Combinatorial Optimisation, Graphic Processing Units and Adaptive Object Extraction. Technical Report 05-07, Laboratoire Centre Enseignement Recherche Traitement Information Systemes (CERTIS), Ecole Nationale des Ponts et Chaussees (ENPC).

Donoser, M. and Bischof, H. (2006). Efficient Maximally Stable Extremal Region (MSER) Tracking. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 553–560.

Donoser, M., Bischof, H., and Wiltsche, M. (2006). Color blob segmentation by MSER analysis. *2006 International Conference on Image Processing*, 1(c):757–760.

Douglas, J. and Rachford, H. H. (1956). On the Numerical Solution of Heat Conduction Problems in Two and Three Space Variables. *Transactions of the AMS*, 82(2):421–439.

Edmonds, J. and Karp, R. M. (1972). Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *Journal of the ACM*, 19(2):248–264.

Evans, L. C. and Gariepy, R. F. (1992). *Measure Theory and Fine Properties of Functions*. CRC Press, Boca Raton, FL.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181.

Fitzpatrick, B. G. (1991). Bayesian analysis in inverse problems. *Inverse Problems*, 7(5):675–702.

Fleming, W. H. and Rishel, R. (1960). An integral formula for total gradient variation. *Archiv der Mathematik*, 11(1):218–222.

Ford, L. R. and Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian Journal of Mathematics*, pages 399–404.

Fussenegger, M., Roth, P., Bischof, H., Deriche, R., and Pinz, A. (2009). A level set framework using a new incremental, robust Active Shape Model for object segmentation and tracking. *Image and Vision Computing*.

Getreuer, P. (2010). tvreg: Variational Imaging Methods (http://www.getreuer.info/home/tvreg).

Gibbs, J. W. (1902). *Elementary principles in statistical mechanics*, volume 66. BiblioBazaar, LLC.

Giusti, E. (1984). *Minimal Surfaces and Functions of Bounded Variation*. Birkhäuser Basel.

Glaskowsky, P. N. (2009). NVIDIA's Fermi: The First Complete GPU Computing Architecture. *A white paper prepared under contract with NVIDIA Corporation*, (September):1–26.

Goldberg, A. V. and Tarjan, R. E. (1988). A new approach to the maximum-flow problem. *Journal of the ACM*, 35(4):921–940.

Goldluecke, B., Strekalovskiy, E., and Cremers, D. (2012). The Natural Vectorial Total Variation which arises from Geometric Measure Theory. *SIMS*, pages 1–27.

Grabner, H., Grabner, M., and Bischof, H. (2006). Real-Time Tracking via On-line Boosting. In *British Machine Vision Conference*, pages 47–56.

Grady, L. (2006). Random walks for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 28(11):1768–83.

Grady, L. and Alvino, C. V. (2009). The piecewise smooth Mumford-Shah functional on an arbitrary graph. *IEEE Transactions on Image Processing*, 18(11):2547–2561.

Greig, D. M., Porteous, B. T., and Seheult, A. H. (1989). Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society Series B*, 51(2):271–279.

Hadamard, J. (1902). Sur les problèmes aux dérivés partielles et leur signification physique. *Princeton University Bulletin*, 13:49–52.

Horn, B. K. P. and Schunck, B. G. (1981). Determining Optical Flow. *Art. Intell.*

Horowitz, S. L. and Pavlidis, T. (1974). Picture segmentation by a directed split-and-merge procedure. In *Proceedings of the 2nd Int Joint Conference on Pattern Recognition*, pages 424–433.

Huber, P. J. (1973). Robust Regression: Asymptotics, Conjectures and Monte Carlo. *The Annals of Statistics*, 1(5):799–821.

Huber, P. J. (1981). *Robust Statistics*. Wiley-Interscience, first edition.

Idier, J. (2008). *Bayesian Approach to Inverse Problems*. John Wiley and Sons.

Isard, M. and Blake, A. (1996). Contour Tracking by Stochastic Propagation of Conditional Density. In *Proc. European Conference on Computer Vision*, pages 343–356.

Ishikawa, H. (2003). Exact optimization for Markov random fields with convex priors. *IEEE transactions on pattern analysis and machine intelligence*, 25(10):1333–1336.

Ising, E. (1925). Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift Für Physik*, 31(1):253–258.

Jayadevappa, D., Kumar, S. S., and Murty, D. S. (2011). Medical Image Segmentation Algorithms using Deformable Models: A Review. *Iete Technical Review*, 28(3):248–255.

Kaipio, J. and Somersalo, E. (2005). *Statistical and Computational Inverse Problems*, volume 160 of *Applied Mathematical Sciences*. Springer.

Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331.

Keller, J. B. (1976). Inverse Problems. *The American Mathematical Monthly*, 83(2):107–118.

Kichenassamy, S., Kumar, A., Olver, P., Tannenbaum, A., and Yezzi, A. (1995). Gradient flows and geometric active contour models. *Proceedings of IEEE International Conference on Computer Vision*, 0:810–815.

Kichenassamy, S., Kumar, A., Olver, P., Tannenbaum, A., and Yezzi, A. (1996). Conformal curvature flows: From phase transitions to active vision. *Archive for Rational Mechanics and Analysis*, 134(3):275–301.

Kirsch, A. (2011). *An Introduction to the Mathematical Theory of Inverse Problems (Applied Mathematical Sciences)*. Springer, second edition.

Klodt, M., Schoenemann, T., Kolev, K., Schikora, M., and Cremers, D. (2008). An Experimental Comparison of Discrete and Continuous Shape Optimization Methods. In *10th European Conference on Computer Vision*, pages 332–345.

Knapik, B. T., Van Der Vaart, A. W., and Van Zanten, J. H. (2011). Bayesian Inverse Problems. *arXiv*, math.ST:1–35.

Kolmogorov, V. and Zabih, R. (2001). Computing Visual Correspondence with Occlusions via Graph Cuts. In *IEEE International Conference on Computer Vision*, pages 508–515.

Kumar, M. P., Torr, P. H. S., and Zisserman, A. (2008). Learning Layered Motion Segmentations of Video. *International Journal of Computer Vision*, 76(3):301–319.

Kunisch, K. and Hintermüller, M. (2004). Total Bounded Variation Regularization as a Bilaterally Constrained Optimization Problem. *SIAM Journal on Applied Mathematics*, 64(4):1311–1333.

Lellmann, J., Becker, F., and Schnörr, C. (2009a). Convex Optimization for Multi-Class Image Labeling with a Novel Family of Total Variation Based Regularizers. *IEEE 12th International Conference on Computer Vision*, (3):646–653.

Lellmann, J., Kappes, J., Yuan, J., Becker, F., and Schnörr, C. (2009b). Convex Multi-Class Image Labeling by Simplex-Constrained Total Variation. *Scale Space and Variational Methods in Computer Vision*, 5567:150–162.

Leung, S. and Osher, S. (2005). Global Minimization of the Active Contour Model with TV-Inpainting and Two-phase Denoising. In *3rd IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision*, pages 149–160.

Li, S. Z. (2009). *Markov Random Field Modeling in Image Analysis Third Edition*, volume i of *Advances in Pattern Recognition*. Springer London.

Mansouri, A.-R., Mitiche, A., and Aron, M. (2003). PDE-based region tracking without motion computation by joint space-time segmentation. In *Proc. International Conference on Image Processing*, pages III–113–16 vol.2.

Markov, A. (1971). Extension of the Limit Theorems of Probability Theory to a Sum of Variables Connected in a Chain. In Howard, R., editor, *Dynamic Probabilistic Systems (Volume I: Markov Models)*, chapter Appendix B, pages 552–577. John Wiley & Sons, Inc., New York City.

McInerney, T. and Terzopoulos, D. (1996). Deformable models in medical image analysis: a survey. *Medical Image Analysis*, 1(2):91–108.

Mortensen, E. N. and Barrett, W. A. (1995). Intelligent scissors for image composition. *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques SIGGRAPH 95*, 84602(801):191–198.

Mumford, D. and Shah, J. (1989). Optimal approximation by piecewise smooth functions and associated variational problems. *Comm Pure Appl Math*, 42:577–685.

Nagel, H. H. and Enkelmann, W. (1986). An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(5):565–593.

Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate. *Soviet Mathematics Doklady*.

Nesterov, Y. (2004). Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152.

Nguyen, H. (2007). *GPU Gems 3*. Addison-Wesley Professional, first edition.

Nieuwenhuis, C., Berkels, B., Rumpf, M., and Cremers, D. (2010). Interactive Motion Segmentation. In *DAGM, Annual Symposium of the German Association for Pattern Recognition*, pages 483–492.

Nikolova, M. (2002). Minimizers of cost-functions involving nonsmooth data-fidelity terms. Application to the processing of outliers. *SIAM Journal on Numerical Analysis*, 40(3):965–994.

Nikolova, M. (2004). A Variational Approach to Remove Outliers and Impulse Noise. *Journal of Mathematical Imaging and Vision*, 20(1/2):99–120.

NVidia (2010). NVIDIA Performance Primitives (NPP) Version 3.2.16. Technical report.

NVidia (2011a). CUDA C Best Practices Guide - DG-05603-001_v4.0. Technical report.

NVidia (2011b). NVIDIA CUDA C Programming Guide - Version 4.0. Technical report.

Ochs, P. and Brox, T. (2011). Object segmentation in video: A hierarchical variational approach for turning point trajectories into dense regions. In *International Conference on Computer Vision*, volume 21, pages 1583–1590. IEEE.

Ochs, P. and Brox, T. (2012). Higher Order Motion Models and Spectral Clustering. *Conference on Computer Vision and Pattern Recognition*, pages 614–621.

Ogale, A. S., Fermüller, C., and Aloimonos, Y. (2005). Motion segmentation using occlusions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):988–992.

Osher, S. and Sethian, J. A. (1988). Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49.

Plato (2008). *The Republic*, chapter Book VII. Project Gutenberg.

Pock, T. (2008). *Fast Total Variation for Computer Vision*. PhD thesis, Graz University of Technology.

Pock, T. and Chambolle, A. (2011). Diagonal preconditioning for first order primal-dual algorithms. *Convergence*, pages 1762–1769.

Pock, T., Chambolle, A., Cremers, D., and Bischof, H. (2009a). A Convex Relaxation Approach for Computing Minimal Partitions. In *Computer Vision and Pattern Recognition*.

Pock, T., Cremers, D., Bischof, H., and Chambolle, A. (2009b). An algorithm for minimizing the Mumford-Shah functional. *2009 IEEE 12th International Conference on Computer Vision*, (813396):1133–1140.

Porikli, F. M. (2001). *Video Object Segmentation*. PhD thesis, New York University, Brooklyn.

Potts, R. B. (1952). Some Generalized Order-Disorder Transformation. In *Transformations, Proceedings of the Cambridge Philosophical Society*, volume 48, pages 106–109.

Protter, M., Yavneh, I., and Elad, M. (2010). Closed-Form MMSE Estimation for Signal Denoising Under Sparse Representation Modeling Over a Unitary Dictionary. *IEEE Transactions on Signal Processing*, 58(7):3471–3484.

Ramsey, F. P. (1931). Truth and Probability. In *The Foundations of Mathematics and other Logical Essays*, volume 2, chapter 7. Routledge (Reprinted 2001).

Reina, A. V., Avidan, S., Pfister, H., and Miller, E. L. (2010). Multiple Hypothesis Video Segmentation from Superpixel Flows. In *IEEE European conference on computer vision*, volume 6315, pages 268–281. Springer.

Roberts, M., Jeong, W.-k., Amelio, V., and Unger, M. (2011). Neural Process Reconstruction from Sparse User Scribbles. In Fichtinger, G., Martel, A., and Peters, T., editors, *Medical Image Computing and Computer-Assisted Intervention*, volume 14 of *Lecture Notes in Computer Science*, pages 621–628. Springer.

Rockafellar, R. T. (1970). *Convex Analysis*, volume 28 of *Princeton Mathematical Series, No. 28*. Princeton University Press.

Rother, C., Kolmogorov, V., and Blake, A. (2004). "GrabCut" - Interactive Foreground Extraction using Iterated Graph Cuts. *ACM Transactions on Graphics (SIGGRAPH)*.

Rother, C., Kolmogorov, V., Lempitsky, V., and Szummer, M. (2007). Optimizing Binary MRFs via Extended Roof Duality. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Rudin, L., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268.

Sanders, J. and Kandrot, E. (2010). *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley Professional.

Santner, J. (2010). *Interactive Multi-Label Segmentation*. PhD thesis, Graz University of Technology.

Santner, J., Pock, T., and Bischof, H. (2011). Interactive Multi-Label Segmentation. *ACCV*, (813396):397–410.

Santner, J., Unger, M., Leistner, C., and Bischof, H. (2009). Interactive Texture Segmentation using Random Forests and Total Variation. In *British Machine Vision Conference*.

Scharstein, D., Szeliski, R., and Zabih, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Proceedings IEEE Workshop on Stereo and MultiBaseline Vision*, 47(1):131–140.

Schoenemann, T. and Cremers, D. (2006). Near Real-Time Motion Segmentation Using Graph Cuts. *DAGM, Annual Symposium of the German Association for Pattern Recognition*, pages 455–464.

Schoenemann, T. and Cremers, D. (2007). Introducing Curvature into Globally Optimal Image Segmentation: Minimum Ratio Cycles on Product Graphs. *IEEE 11th International Conference on Computer Vision (2007)*, pages 1–6.

Schoenemann, T. and Cremers, D. (2008). High Resolution Motion Layer Decomposition using Dual-space Graph Cuts. In *Conference on Computer Vision and Pattern Recognition*, Anchorage, Alaska.

Schoenemann, T., Kahl, F., Masnou, S., and Cremers, D. (2012). A linear framework for region-based image segmentation and inpainting involving curvature penalization. *International Journal of Computer Vision (to appear)*.

Sethian, J. A. (1999). *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision and Materials Science*, volume 11 of *Cambridge Monograph on Applied and Computational Mathematics*. Cambridge University Press.

Sezgin, M. and Sankur, B. (2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–168.

Shulman, D. and Hervé, J.-Y. (1989). Regularization of discontinuous flow fields. In *Proceedings Workshop on Visual Motion*, pages 81–86.

Sinha, S. N., Steedly, D., and Szeliski, R. (2009). Piecewise planar stereo for image-based rendering. In *ICCV*.

Sinop, A. K. and Grady, L. (2007). A Seeded Image Segmentation Framework Unifying Graph Cuts And Random Walker Which Yields A New Algorithm. *IEEE 11th International Conference on Computer Vision*.

Steinbruecker, F., Pock, T., and Cremers, D. (2009). Advanced Data Terms for Variational Optic Flow Estimation. In *Vision, Modeling and Visualization Workshop*, volume 1, Braunschweig, Germany.

Strang, G. (1983). Maximal flow through a domain. *Mathematical Programming*, 26(2):123–143.

Strang, G. (2009). Maximum flows and minimum cuts in the plane. *Journal of Global Optimization*, 47(3):527–535.

Stuart, A. M. (2010). Inverse Problems: A Bayesian Perspective. *Acta Numerica*, 19(1):1–107.

Sun, D., Roth, S., and Black, M. J. (2010a). Secrets of Optical Flow Estimation and Their Principles. *Baseline*, pages 2432–2439.

Sun, D., Sudderth, E., and Black, M. (2010b). Layered image motion with explicit occlusions, temporal consistency, and depth ordering. In *Advances in Neural Information Processing Systems 23*, pages 2226–2234.

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., and Rother, C. (2008). A comparative study of energy minimization methods for

Markov random fields with smoothness-based priors. *IEEE transactions on pattern analysis and machine intelligence*, 30(6):1068–80.

Terzopoulos, D., Witkin, A., and Kass, M. (1988). Constraints on deformable models:Recovering 3D shape and nonrigid motion. *Artificial Intelligence*, 36(1):91–123.

Tikhonov, A. N. (1963). Regularization of incorrectly posed problems. *Soviet Math Dokl*, 4(1):1624–1627.

Tikhonov, A. N. and Arsenin, V. Y. (1977). *Solutions of ill-posed problems*, volume 21 of *Scripta series in mathematics*. W. H. Winston.

Touchette, H. (2007). Legendre-Fenchel transform in a nutshell. Technical report, School of Mathematical Sciences, Queen Mary, University of London.

Tsai, A., Yezzi, A., and Willsky, A. S. (2000). A curve evolution approach to smoothing and segmentation using the Mumford-Shah functional. In *Conference on Computer Vision and Pattern Recognition*.

Unger, M., Mauthner, T., Pock, T., and Bischof, H. (2009). Tracking as Segmentation of Spatial-Temporal Volumes by Anisotropic Weighted TV. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 193–206.

Unger, M., Pock, T., and Bischof, H. (2008a). Continuous Globally Optimal Image Segmentation with Local Constraints. In *Computer Vision Winter Workshop*, Moravske Toplice, Slovenija.

Unger, M., Pock, T., and Bischof, H. (2011). Global Relabeling for Continuous Optimization in Binary Image Segmentation. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, Saint Petersburg, Russia.

Unger, M., Pock, T., Trobin, W., Cremers, D., and Bischof, H. (2008b). TVSeg - Interactive Total Variation Based Image Segmentation. In *British Machine Vision Conference*, Leeds, UK.

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 *Conference on Computer Vision and Pattern Recognition*, Providence, USA.

Veksler, O. (1999). *Efficient Graph-Based Energy Minimization Methods in Computer*. PhD thesis, Cornell University.

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(3):271–293.

Vincent, L. and Soille, P. (1991). Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598.

Vineet, V. and Narayanan, P. J. (2008). CUDA cuts: Fast graph cuts on the GPU. *IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

Vogel, C. R. and Oman, M. (1996). Iterative Methods For Total Variation Denoising. *SIAM J. Sci. Comput*, 17:227–238.

Wang, J. Y. A. and Adelson, E. H. (1994). Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638.

Werlberger, M. (2012). *Convex Approaches for High Performance Video Processing*. PhD thesis, Graz University of Technology, Austria.

Werlberger, M., Pock, T., and Bischof, H. (2010). Motion Estimation with Non-Local Total Variation Regularization. *Order - A Journal On The Theory Of Ordered Sets And Its Applications*, pages 2464–2471.

Werlberger, M., Pock, T., Unger, M., and Bischof, H. (2011a). Optical Flow Guided TV-L1 Video Interpolation and Restoration. In *Proceedings Energy Minimization Methods in Computer Vision and Pattern Recognition*, number 6819 in Lecture Notes in Computer Science, pages 273–286.

Werlberger, M., Trobin, W., Pock, T., Wedel, A., Cremers, D., and Bischof, H. (2009). Anisotropic Huber-L1 Optical Flow. In *Proceedings of the British Machine Vision Conference BMVC*, London, UK.

Werlberger, M., Unger, M., Pock, T., and Bischof, H. (2011b). Efficient Minimization of the Non-Local Potts Model. In *Proceedings International Conference on Scale Space and Variational Methods in Computer Vision*, Lecture Notes in Computer Science, pages 314–325. Springer.

Woodford, O., Torr, P., Reid, I., and Fitzgibbon, A. (2009). Global Stereo Reconstruction under Second-Order Smoothness Priors. *IEEE transactions on pattern analysis and machine intelligence*, 31:2115–2128.

Xu, C., Pham, D. L., and Prince, J. L. (2000). Medical Image Segmentation Using Deformable Models. In Fitzpatrick, J. M. and Sonka, M., editors, *Handbook of Medical Imaging*, volume 2, chapter 3, pages 129–174. SPIE Press.

Yilmaz, A. (2007). Object Tracking by Asymmetric Kernel Mean Shift with Automatic Scale and Orientation Selection. In *Conference on Computer Vision and Pattern Recognition*, pages 1–6.

Yilmaz, A., Li, X., and Shah, M. (2004). Contour Based Object Tracking with Occlusion Handling in Video Acquired Using Mobile Cameras. *IEEE transactions on pattern analysis and machine intelligence*, 26:1531–1536.

Yuan, J. (2011). *Convex Variational Approaches to Image Motion Estimation, Denoising and Segmentation*. PhD thesis, Ruprecht-Karls-University Heidelberg.

Yuan, J. and Boykov, Y. (2010). TV-Based Multi-Label Image Segmentation with Label Cost Prior. In *Procedings of the British Machine Vision Conference*, number 1, pages 101.1–101.12.

Zach, C., Gallup, D., Frahm, J.-m., and Niethammer, M. (2008). Fast Global Labeling for Real-Time Stereo Using Multiple Plane Sweeps. In *Vision, Modeling and Visualization Workshop*, pages 243–252.

Zach, C., Pock, T., and Bischof, H. (2007). A Duality Based Approach for Realtime TV-L1 Optical Flow. In *DAGM, Annual Symposium of the German Association for Pattern Recognition*.

Zhang, G., Jia, J., Hua, W., and Bao, H. (2011). Robust Bilayer Segmentation and Motion/Depth Estimation with a Handheld Camera. *IEEE transactions on pattern analysis and machine intelligence*, 33:603–617.

Zhu, M. and Chan, T. (2008). An Efficient Primal-Dual Hybrid Gradient Algorithm For Total Variation Image Restoration. *UCLA CAM Report 08-34*, (1):1–29.

Zitnick, C. L., Jojic, N., and Kang, S. B. (2005). Consistent Segmentation for Optical Flow Estimation. In *IEEE International Conference on Computer Vision*, pages II: 1308–1315.

Zitnick, C. L. and Kang, S. B. (2007). Stereo for Image-Based Rendering using Image Over-Segmentation. *International Journal of Computer Vision*, 75(1):49–65.