Discrete Optimization in Early Vision - Model Tractability Versus Fidelity

Strandmark, Petter

2012

DISCRETE OPTIMIZATION IN EARLY VISION

MODEL TRACTABILITY VERSUS FIDELITY

PETTER STRANDMARK

LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118
SE-221 00 Lund
Sweden

http://www.maths.lth.se/

Doctoral Theses in Mathematical Sciences 2012:5
ISSN 1404-0034

ISBN 978-91-7473-407-2
LUTFMA-1046-2012

Printed in Sweden by Media-Tryck, Lund 2012

Contents

Preface

Early vision is the process occurring before any semantic interpretation of an image takes place. Motion estimation, object segmentation and detection are all parts of early vision, but recognition is not. Some models in early vision are easy to perform inference with—they are tractable. Others describe the reality well—they have high fidelity. This thesis improves the tractability-fidelity trade-off of the current state of the art by introducing new discrete methods for image segmentation and other problems of early vision.

The first part studies pseudo-boolean optimization, both from a theoretical perspective as well as a practical one by introducing new algorithms. The main result is the generalization of the roof duality concept to polynomials of higher degree than two. Another focus is parallelization; discrete optimization methods for multi-core processors, computer clusters, and graphical processing units are presented.

Remaining in an image segmentation context, the second part studies parametric problems where a set of model parameters and a segmentation are estimated simultaneously. For a small number of parameters these problems can still be optimally solved. One application is an optimal method for solving the two-phase Mumford-Shah functional.

The third part shifts the focus to curvature regularization—where the commonly used length and area penalization is replaced by curvature in two and three dimensions. These problems can be discretized over a mesh and special attention is given to the mesh geometry. Specifically, hexagonal meshes in the plane are compared to square ones and a method for generating adaptive meshes is introduced and evaluated. The framework is then extended to curvature regularization of surfaces.

Finally, the thesis is concluded by three applications to early vision problems: cardiac MRI segmentation, image registration, and cell classification.

❧ ❧ ❧

Without my colleagues at Lund University, this thesis could not have come into existence. First, I would like to thank my supervisor, Fredrik Kahl, for his help and collaboration on several papers. During my four years at the Centre for Mathematical Sciences, I have also had the great privilege of working with Fredrik Andersson, Johan Fredriksson, Yubin Kuang, Carl Olsson, Niels Christian Overgaard, Thomas Schoenemann, Linus Svärm, and Johannes Ulén. Everyone in the Mathematical Imaging Group and at the Centre for Mathematical Sciences has my thanks for providing great company, courses and seminars. Bo Bernhardsson was the examiner of my Licentiate thesis and provided very useful feedback. Emmy Nilsson has supported me the entire time and has my deepest thanks for reading through my drafts and offering valuable suggestions for improvement.

I would also like to acknowledge the funding that has made my graduate studies possible. My work has been funded by the Swedish Foundation for Strategic Research (FFL and VINST) and by the European Research Council (Global Vision, grant number 209480).

Lund, December 7, 2012

Chapter 1

Introduction

Eyes have evolved independently in at least 40 groups of animals (Coyne, 2009). Despite considerable complexities and evolutionary hurdles, the camera-type eye, with a lens focusing the image onto the retina, has evolved in very different groups: from vertebrates to crustaceans and jellyfish (Nilsson, 1999). This is a testament to the importance of vision for life on earth and a suggestion that artificial vision systems, if constructed properly, will be immensely useful to machines for navigation, interaction and measurement. The construction of such artificial vision systems may therefore be seen as a subfield of artificial intelligence, and is commonly referred to as *computer vision*. It is a relatively recent subfield of artificial intelligence; recent because it usually involves processing large quantity of data—even very small images could exhaust the best computers in the 70s. The history of artificial intelligence itself goes back to at least 1950, when Alan Turing concluded his seminal paper with the following paragraph, suggesting two lines of future research:

> We may hope that machines will eventually compete with men in all purely intellectual fields. But which are the best ones to start with? Even this is a difficult decision. Many people think that a very abstract activity, like the playing of chess would be best. It can also be maintained that it is best to provide the machine with the best sense organs that money can buy, and then teach it to understand and speak English. This process could follow the normal teaching of a child. Things would be pointed out and named, etc. Again I do not know what the right answer is, but I think both approaches should be tried.

Half a century later, the first possibility has turned out to be overwhelmingly successful, whereas the second has had little success. "The best sense organs that money can buy" are indeed stunningly powerful today, but any sense

organ, visual ones certainly not excluded, need a powerful computational system to process and interpret the data. One of the ultimate goals of computer vision is precisely this.

Of course, the goal of this thesis is much more modest. I will describe methods for solving certain optimization problems commonly arising in computer vision. One of the goals will be globality, that is, the solution obtained should be guaranteed to be the best possible. Two other goals will be parallelization to make use of modern microprocessors and memory efficiency, since solving computer vision problems can be prohibitively expensive in terms of memory requirements.

The state of the art of computer vision highly depends on what applications one has in mind. Computers are able to estimate the scene depth from a pair of images with great accuracy and to recover the 3D structure of an entire city just from arbitrarily downloaded images from the Internet (Agarwal et al., 2009). On the other hand, computers are still unable to perform tasks which humans perform effortlessly, such as determining whether an image contains a chair or not. Recognition is an example of high-level vision and is very difficult for computers. In contrast, my thesis mostly focuses on low-level vision, tasks which humans perform subconsciously in every waking second, but still provide challenges for designers of artificial vision systems. These tasks are also commonly referred to as tasks of *early vision*. Recognition will only be touched upon towards the end, where chapter 14 will show an example of a well-functioning recognition system—made possible by the very controlled setting in which the images were captured.

1.1 Tractability and Fidelity

Models in image analysis have two important desiderata: *tractability* and *fidelity* (McIntosh and Hamarneh, 2011). Tractability specifies how amenable the model is to optimization techniques. For example, models involving curvature and geometric shape priors tend to be harder to optimize than models only based on image intensities. On the other hand, these more complex models often describe the reality better and have the potential for better results—they have higher fidelity. There is a natural trade-off between tractability and fidelity.

Traditionally, most approaches in computer vision rely on local descent

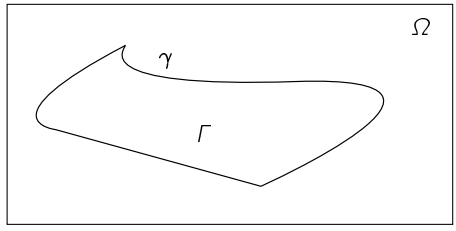


Figure 1.1: Segmentation of an image. The foreground region Γ has the curve γ as its boundary.

techniques, e.g. (Lorenzo-Valdés et al., 2004; Mitchell et al., 2002; Paragios, 2002), and may get stuck in local optima which make them less reliable. These models are not always tractable, but their fidelity can still be excellent. On the other hand, it is possible to apply global optimization techniques to make the solution more robust to poor initialization, e.g. (Boykov and Jolly, 2000; Lin et al., 2006). The models used then have great tractability, but are often less sophisticated.

The purpose of this thesis is to improve the tractability–fidelity trade-off. The first parts focus on optimization, allowing existing models to be optimized more efficiently. Towards the end, new models are developed that are still tractable.

1.2 Thesis Overview

A major part of my thesis is devoted to image segmentation and the problems surrounding it. A segment of an image is described by a simple closed curve γ enclosing an area $\Gamma \subseteq \Omega$ in such a way that the area represents something significant about the image, e.g. foreground or an interesting object, see figure 1.1. The segmentation task is to find the best (closed) curve γ given an image I, or more precisely,

$$\underset{\gamma}{\text{maximize}} \quad \mathrm{P}(\gamma \mid I), \tag{1.1}$$

where $\mathrm{P}(\gamma \mid I)$ is the posterior probability of γ given the image $I : \Omega \to \mathbf{R}^3$ (or $\mathbf{R}$ if the image is black and white). This thesis will discuss various aspects of solving (1.1). Using Bayes' rule, this expression becomes $\mathrm{P}(\gamma \mid I) =$

$\mathrm{P}(\gamma)\,\mathrm{P}(I \mid \gamma)/\mathrm{P}(I)$ and, if the log-likelihood is denoted $\ell(\gamma) = \log \mathrm{P}(\gamma)$,

$$\ell(\gamma \mid I) = \ell(\gamma) + \ell(I \mid \gamma) \underbrace{- \ell(I)}_{\substack{\text{independent} \\ \text{of } \gamma}} . \tag{1.2}$$

This is known as maximum a posteriori (MAP) estimation. In computer vision and image analysis the probabilistic interpretation is sometimes dropped and $\ell(\gamma \mid I)$ is simply referred to as the energy of γ. The reason for this is the influence from the calculus of variations used in physics. Consequently, image segmentation and other tasks are often referred to by computer vision researchers as energy minimization problems. The prior $\ell(\gamma)$ is often called the regularizing term, or smoothness term and $\ell(I \mid \gamma)$ is called the data term , a vocabulary I will use throughout the thesis. Letting $E_{\text{smooth}}(\gamma) = -\ell(\gamma)$ and $E_{\text{data}}(\gamma) = -\ell(I \mid \gamma)$, we have finally arrived at the optimization problem

smoothness term
data term

$$\underset{\gamma}{\text{minimize}} \quad E_{\text{smooth}}(\gamma) + E_{\text{data}}(\gamma). \tag{1.3}$$

It is equivalent to (1.1), but more similar to how these problems are commonly presented in the computer vision community. The data term is the likelihood of an image given a particular segmentation and is commonly written as

$$E_{\text{data}}(\gamma) = \int_{\Gamma} -\ell(\boldsymbol{x} \text{ is foreground} \mid I(\boldsymbol{x}))d\boldsymbol{x} + \int_{\Omega \setminus \Gamma} -\ell(\boldsymbol{x} \text{ is background} \mid I(\boldsymbol{x}))d\boldsymbol{x}. \tag{1.4}$$

Chapter 3 of my Licentiate thesis (Strandmark, 2010) briefly described how smoothness and regularizing terms arise in the context of conditional independence between image pixels (the Markov property). The data term requires likelihoods $\ell(\boldsymbol{x} \text{ is foreground} \mid I(\boldsymbol{x}))$ and $\ell(\boldsymbol{x} \text{ is background} \mid I(\boldsymbol{x}))$ for each position $\boldsymbol{x}$ in the image and these are highly application-dependent. A typical simple application uses some known or estimated color histogram to model the two regions. Mumford and Shah (1989) describe various forms of $\ell(\gamma)$ and $\ell(I \mid \gamma)$.

Maximizing (1.3) with data term (1.4) becomes very easy under the assumption that all possible curves γ are equally likely a priori, i.e. $\ell(\gamma)$ is

constant. Simply thresholding $\ell(\cdot, I(x))$ will give the optimal Γ. Regrettably, the resulting model is often not good enough. Each point in the image can under this assumption be classified independently of the others, which is unrealistic. This assumption can be removed and a more reasonable prior is then imposed on the curve γ. Several chapters in this thesis will discuss the aspects of the optimization problems arising for different classes of regularizing terms.

The thesis starts out with chapters 2 and 3 about optimization—continuous and discrete, respectively. They will introduce some required terminology and notation. After these introductory chapters, the thesis contains five parts, each containing one or several chapters, which can be read in any order. The rest of this section will briefly summarize the contributions of these chapters.

Chapter 4. The first contribution of the thesis is not directly concerned with image analysis and computer vision and studies the minimization of pseudo-boolean functions. Although this problem does arise in vision, the potential applications extend further than that. The major contribution of this chapter is an extension of "roof duality" to polynomials of higher degree than two. This framework is called "generalized roof duality."

Chapter 5. There are other possible approaches to solving pseudo-boolean optimization problems. Chapter 5 introduces a few other new methods and discusses their advantages and shortcomings.

Chapter 6. One problem used to evaluate discrete optimization methods is image denoising. This chapter shows that although the the problem is useful for benchmarking, continuous methods solve it faster and more accurately.

Chapter 7. Image segmentation with length regularity may also be converted into the graph theoretical problem of finding the minimum cut in a directed graph. Many other types of problems can also be formulated as minimum cut problems. Fast algorithms for solving these problems are therefore of the utmost importance in computer vision. Chapter 7 will introduce a method for solving the minimum cut problem in parallel, solving small problems faster and making larger problems tractable by distributing them

across multiple computers. This will be achieved by dual decomposition, a well-known optimization technique summarized in chapter 2.

Chapter 8. Devices capable of performing more than a trillion arithmetic operations per second are today available over the counter. These highly parallel graphical processing units (GPUs) have in the last decade found their way into computing, with manufacturers like Nvidia diverting more and more die space from gaming to computing. This chapter continues the search for fast algorithms for graph cuts and other more general labeling problems by exploring possible GPU algorithms. An algorithm based on dual decomposition and dynamic programming is introduced, similar to one presented by Komodakis et al. (2007).

Chapter 9. This chapter considers the case where the prior of the curve γ is a linear function of its length:

$$\ell(\gamma) = -\rho \,\text{length}(\gamma), \qquad \rho > 0. \tag{1.5}$$

The theory developed can also handle slight variations thereof, such as anisotropic length (Olsson et al., 2009). This optimization problem is well-known to be solvable and I will describe methods to simultaneously find the best possible *data term* under some conditions. As a simple example, one might wish to model an image as having foreground and background pixels drawn from two different Gaussian distributions with (unknown) means but equal variance. The techniques developed in chapter 9 enable the efficient computation of optimally estimated means simultaneously with an optimal segmentation.

Chapter 10. After covering length-based regularization, the focus is shifted to more general functionals involving curvature:

$$\ell(\gamma) = -\rho \,\text{length}(\gamma) - \sigma \int_\gamma |\kappa(s)|^d ds, \qquad \rho, \sigma \geq 0, \tag{1.6}$$

where the curvature $\kappa(s) = \|\boldsymbol{\gamma}''(s)\|$ for a curve γ parametrized by its arc length s. Functionals with such terms can be discretized with a grid and subsequently solved with linear programming, as was shown recently by Schoenemann et al. (2012). Chapter 10 will build upon their work and

extend the method in several ways. A new set of constraints will be introduced which fixes an issue in the original formulation. The framework is then extended in two directions: It is shown that the square grid naturally arising from the image pixels is not as efficient as hexagonal grids. Complementary to this, a method for adaptive grid generation is discussed and experimentally tested.

Chapter 11. Whereas the previous chapter expanded upon the existing framework for functionals involving the curvature of plane curves, chapter 11 will introduce a new framework for minimizing the curvature of surfaces. Applications include three-dimensional segmentation and surface completion, for example, within stereo vision.

Chapter 12. The first chapter in the applications part describes an application of dual decomposition to multi-region modes for medical 3D segmentation. A complete system for the human heart is presented and evaluated using publicly available data.

Chapter 13. The penultimate chapter considers an application of multi-label minimization to image registration. "Multi-label" refers to that fact that each variable has more than two possible values and registering a pair of images means to find pixel-to-pixel correspondences between them. With both the number of variables and the number of labels ranging in the millions, the size of the search space is too large for any guarantee of globality or approximations thereof.

Chapter 14. Finally, the last contribution chapter is a demonstration of what can be done if good segmentations are available. Sorting images of cells into clinically relevant categories is a difficult problem and this chapter introduces a new classification system that attempts to automate this task.

Recent work. The field of image processing and computer vision is rapidly changing. Some work performed by others build on the work in this thesis and the last chapter is a summary of some of these publications.

1.3 Previous Publications and Author Contributions

A doctoral thesis is not only an attempt to improve the state of the art in a research field; it is also the main requirement to obtain a degree. This section is an account of previous publications on which this thesis is based and my contributions to them. Some material presented in this thesis is not based on previous publications, but most parts of it are.

- Petter Strandmark, Fredrik Kahl and Niels Chr. Overgaard, "Optimizing Parametric Total Variation Models," *International Conference on Computer Vision* (ICCV), Kyoto, 2009.

  I and Fredrik developed the theory. I implemented the method and designed the experiments. Most of the paper was written by me and the remaining part by Fredrik. Niels Christian developed the numerical solver.

- Petter Strandmark and Fredrik Kahl, "Parallel and Distributed Graph Cuts by Dual Decomposition," *Conference on Computer Vision and Pattern Recognition* (CVPR), San Francisco, 2010.

  The work distribution between me and Fredrik was practically identical to the previous paper.

- Linus Svärm and Petter Strandmark, "Shift-map Image Registration," *International Conference on Pattern Recognition*, Istanbul, 2010.

  After ICCV 2009, Olof Enqvist suggested using shift-maps for image registration. I re-implemented the inpainting code and Linus Svärm developed and implemented the registration method. The paper was mostly written by me.

- Petter Strandmark, Fredrik Kahl and Thomas Schoenemann, "Parallel and Distributed Vision Algorithms Using Dual Decomposition." *Computer Vision and Image Understanding*, 2011.

  This is a journal paper whose first part is based on the CVPR 2010 paper. The remaining two parts were mostly developed and written by me. Fredrik assisted in writing the paper and Thomas contributed curvature source code and wrote section 6.1 in the paper..

- Petter Strandmark and Fredrik Kahl, "Curvature Regularization for Curves and Surfaces in a Global Optimization Framework," *Energy Minimization Methods in Computer Vision and Pattern Recognition*, St. Petersburg, 2011.

  The methods in this paper were developed and implemented by me, with ideas and suggestions by Fredrik. I wrote most of the paper. The implementation was an extension of the framework by Thomas Schoenemann.

- Johannes Ulén, Petter Strandmark and Fredrik Kahl, "Optimization for Multi-Region Segmentation of Cardiac MRI," *MICCAI Workshop on Statistical Atlases and Computational Models of the Heart: Imaging and Modelling Challenges*, Toronto, 2011.

  Johannes did most of the work for this paper. I took part in all stages of the work, especially the optimization. I wrote the initial C++ implementation of the optimization routine.

- Johannes Ulén, Petter Strandmark and Fredrik Kahl, "An Efficient Optimization Framework for Multi-Region Segmentation based on Lagrangian Duality," *IEEE Transactions on Medical Imaging*, to appear, 2013.

  The individual contributions were similar to the above conference version.

- Fredrik Kahl and Petter Strandmark, "Generalized Roof Duality for Pseudo-Boolean Optimization," *International Conference on Computer Vision* (ICCV), Barcelona, 2011.

  Fredrik came up with the idea for this paper while giving a course in discrete optimization. I implemented the method and designed the experiments. Consequently, Fredrik wrote most of the first half of the paper, while I wrote the experiments section.

- Fredrik Kahl and Petter Strandmark, "Generalized Roof Duality," *Discrete Applied Mathematics*, 2012.

  The journal version provided proofs of our claims in the conference version, some written by me and some by Fredrik.

- Petter Strandmark, Johannes Ulén and Fredrik Kahl, "HEp-2 Staining Pattern Classification," *International Conference on Pattern Recognition*, Tsukuba 2012.

  I wrote most of the classification framework, evaluation code and the initial features. Johannes wrote most of the feature extraction code together with me and Fredrik. I wrote most of the paper.

The material in chapters 5 and 6 has not been previously published. The convex shape prior in section 10.7 (page 157) and the refinement in section 10.8 (page 161) have also not appeared in previous publications. Much of the material in my Licentiate thesis (Strandmark, 2010) appears in revised form in this thesis, but some parts of it, the largest being chapter 3, have been completely omitted. The experiments in chapter 6 were conducted by me in cooperation with Sameer Agarwal during my stay at Google. Finally, I have presented selected parts of this work at the annual Swedish Symposium on Image Analysis (SSBA)[1]:

- Petter Strandmark and Fredrik Kahl, "Pseudo-Boolean Optimization: Theory and Applications in Vision," 2012.
- Petter Strandmark and Fredrik Kahl, "Mesh Types for Curvature Regularization," 2011.
- Petter Strandmark and Fredrik Kahl, "Parallel and Distributed Graph Cuts," 2010.
- Linus Svärm and Petter Strandmark, "Shift-map Image Registration," 2010.
- Petter Strandmark, Fredrik Kahl and Niels Christian Overgaard, "Optimal Levels for the Two-phase, Piecewise Constant Mumford-Shah Functional," 2009.

[1] This symposium is not peer-reviewed.

Chapter 2

Continuous Optimization

This chapter and the next will provide an overview of optimization techniques used throughout the thesis. My view of the field of convex optimization comes from the books by Boyd and Vandenberghe (2004), Bertsekas (1999), Nesterov (2004), and Nocedal and Wright (2006). Dual decomposition is a technique for splitting up an optimization problem into smaller and, hopefully, easier subproblems. These subproblems are solved iteratively to obtain a solution to the original problem. The next chapter will cover methods from combinatorial optimization, which are of utmost importance for modern low-level vision, and branch and bound, which solves hard optimization problems by searching the spaces of feasible solutions in a clever way, thereby eliminating the need for an exhaustive search.

2.1 Convex Functions

The minimization of a differentiable function $f \in \mathcal{C}^1(\mathbf{R}^n)$ is in general an intractable task. The class of differentiable functions is too large and has too little structure for there to exist any method to minimize them all in any reasonable amount of time. The best one can expect from gradient descent schemes is the convergence to a local minima in general, not the global minimum. One might then ask for a class of functions which are possible to minimize globally in a reasonable amount of time. What would be a suitable definition of such a class $\mathcal{F} \subset \mathcal{C}^1(\mathbf{R}^n)$?

We must be able to determine whether we have reached the global optimum. Furthermore, such a certificate would ideally be possible to compute locally. Therefore, since the functions in $\mathcal{F}$ are only assumed to be differentiable it is natural to require that

$$f \in \mathcal{F}, \quad \nabla f(\boldsymbol{x}^*) = \mathbf{0} \quad \Longrightarrow \quad f(\boldsymbol{x}^*) \text{ is the global minimum.} \quad \text{(A)}$$

Another useful requirement is that the class $\mathcal{F}$ is closed under addition:

$$f, h \in \mathcal{F} \quad \Longrightarrow \quad f + h \in \mathcal{F}. \tag{B}$$

Lastly, to ensure that the class is non-empty, we require that the affine functions are members of $\mathcal{F}$. They are certainly easy to minimize over $\mathbf{R}^n$.

$$\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{b} \in \mathcal{F}. \tag{C}$$

It turns out that the requirements A–C are enough to derive a simple characterization of the functions in $\mathcal{F}$. Let $f \in \mathcal{F}$ and $\boldsymbol{x}_0 \in \mathbf{R}^n$ be fixed. Consider the function h defined by

$$h(\boldsymbol{x}) = f(\boldsymbol{x}) - \nabla f(\boldsymbol{x}_0)^{\mathrm{T}}\boldsymbol{x}. \tag{2.1}$$

Assumptions B and C tells us that $h \in \mathcal{F}$. We have that $\nabla h(\boldsymbol{x}_0) = \nabla f(\boldsymbol{x}_0) - \nabla f(\boldsymbol{x}_0) = \mathbf{0}$. Therefore, $\boldsymbol{x}_0$ is the global minimizer to h according to assumption A, which means that for any $\boldsymbol{x} \in \mathbf{R}^n$,

$$h(\boldsymbol{x}) \geq h(\boldsymbol{x}_0) = f(\boldsymbol{x}_0) - \nabla f(\boldsymbol{x}_0)^{\mathrm{T}}\boldsymbol{x}_0 \tag{2.2}$$

and thus,

$$f(\boldsymbol{x}) \geq f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}_0). \tag{2.3}$$

convex functions

This is the definition of differentiable convex functions on $\mathbf{R}^n$. Conversely, it is easily seen that A and B hold for convex functions. This argument (Nesterov, 2004) shows the importance of convex functions in optimization. In general, convex functions are defined by the inequality

$$f(t\boldsymbol{x} + (1-t)\boldsymbol{y}) \leq tf(\boldsymbol{x}) + (1-t)f(\boldsymbol{y}), \qquad t \in [0, 1]. \tag{2.4}$$

concave

If instead the reverse inequality holds for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{R}^n$, the function is said to be concave.

Convex functions need not be differentiable. Still, it is always possible to find a tangent plane completely below the function graph at any point. More precisely, for any $\boldsymbol{x}_0$ it is always possible to find a vector $\boldsymbol{g} \in \mathbf{R}^n$ such that

$$f(\boldsymbol{x}) \geq f(\boldsymbol{x}_0) + \boldsymbol{g}^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}_0), \qquad \text{for all } \boldsymbol{x}. \tag{2.5}$$

subgradient
supergradient

The vector $\boldsymbol{g}$ is called a subgradient of f at $\boldsymbol{x}_0$. The analogue for concave functions with a reversed inequality are supergradients, shown in Figure 2.1. If f is differentiable, $\boldsymbol{g} = \nabla f(\boldsymbol{x}_0)$ is the only subgradient to f at $\boldsymbol{x}_0$. If not, the set of all $\boldsymbol{g}$ satisfying (2.5) is denoted $\nabla f(\boldsymbol{x}_0)$. This set is convex. One can observe that $\mathbf{0} \in \nabla f(\boldsymbol{x}_0)$ if and only if $\boldsymbol{x}_0$ is the global optimum.

2.2 Dual Decomposition

This section introduces dual decomposition (Bertsekas, 1999), a general technique in optimization to split a large problem into two or more smaller, better manageable ones. The technique of decomposing problems with dual variables was introduced by Everett (1963) and it has been used in many different contexts, e.g. in control (Rantzer, 2009). The application within computer vision that bears the most resemblance to that of this thesis is the work of Komodakis et al. (2007) where dual decomposition is used for computing approximate solutions to general Markov Random Field (MRF) problems. Consider the following optimization problem:

$$\inf_{\boldsymbol{x} \in X} E(\boldsymbol{x}), \tag{P}$$

where X and E are arbitrary. Sometimes the function E can be split up into two parts: $E(\boldsymbol{x}) = E_1(\boldsymbol{x}) + E_2(\boldsymbol{x})$, where the functions E_1 and E_2 are much easier to minimize. We will see later that this is the case with many objective functions associated with MRFs. Now let us consider the equivalent optimization problem

$$\begin{aligned} &\inf_{\boldsymbol{x}, \boldsymbol{y} \in X} E_1(\boldsymbol{x}) + E_2(\boldsymbol{y}) \\ &\text{subject to} \quad \boldsymbol{x} = \boldsymbol{y}. \end{aligned} \tag{2.6}$$

The Lagrangian dual function (Bertsekas, 1999; Boyd and Vandenberghe, 2004) of this optimization problem is

$$\begin{aligned} d(\boldsymbol{\lambda}) &= \inf_{\boldsymbol{x}, \boldsymbol{y} \in X} \Big(E_1(\boldsymbol{x}) + E_2(\boldsymbol{y}) + \boldsymbol{\lambda}^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{y}) \Big) \\ &= \inf_{\boldsymbol{x} \in X} \Big(E_1(\boldsymbol{x}) + \boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{x} \Big) \\ &+ \inf_{\boldsymbol{y} \in X} \Big(E_2(\boldsymbol{y}) - \boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{y} \Big). \end{aligned} \tag{2.7}$$

The last two rows show that evaluating the dual function is equivalent to solving two independent minimization problems. If minimizing E_1 and E_2 is much easier than minimizing E, the dual function can be evaluated quite efficiently. Since the value of the dual function $d(\boldsymbol{\lambda})$ is a lower bound on the solution to (P) for every $\boldsymbol{\lambda}$, it is of great interest to compute the largest

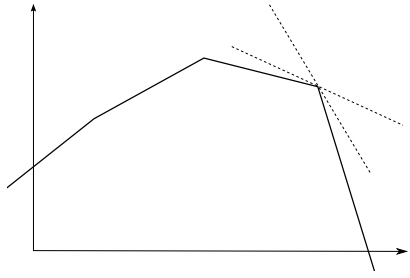


Figure 2.1: Two different supergradients of a concave function represented as two tangent planes.

possible lower bound, i.e. to solve the dual problem:

$$\sup_{\boldsymbol{\lambda}} d(\boldsymbol{\lambda}). \tag{D}$$

The dual function is concave, since it is the infimum over a set of concave (affine) functions of $\boldsymbol{\lambda}$. Furthermore, it is also easy to find a supergradient to d, as stated by the following lemma:

LEMMA 2.1 *Given a $\boldsymbol{\lambda}_0$, let $\boldsymbol{x}^*$ be an optimal solution to*

$$d(\boldsymbol{\lambda}_0) = \min_{\boldsymbol{x}} \Big(f(\boldsymbol{x}) + \boldsymbol{\lambda}_0^{\mathrm{T}} \boldsymbol{g}(\boldsymbol{x}) \Big). \tag{2.8}$$

Then $\boldsymbol{g}(\boldsymbol{x}^*)$ is a supergradient to d at $\boldsymbol{\lambda}_0$.

Proof. The following inequality holds for any $\boldsymbol{\lambda}$:

$$\begin{aligned} d(\boldsymbol{\lambda}) &\leq f(\boldsymbol{x}^*) + \boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{g}(\boldsymbol{x}^*) \\ &= f(\boldsymbol{x}^*) + \boldsymbol{\lambda}_0^{\mathrm{T}} \boldsymbol{g}(\boldsymbol{x}^*) + (\boldsymbol{\lambda} - \boldsymbol{\lambda}_0)^{\mathrm{T}} \boldsymbol{g}(\boldsymbol{x}^*) \\ &= \min_{\boldsymbol{x}} \Big(f(\boldsymbol{x}) + \boldsymbol{\lambda}_0^{\mathrm{T}} \boldsymbol{g}(\boldsymbol{x}) \Big) + (\boldsymbol{\lambda} - \boldsymbol{\lambda}_0)^{\mathrm{T}} \boldsymbol{g}(\boldsymbol{x}^*) \\ &= d(\boldsymbol{\lambda}_0) + (\boldsymbol{\lambda} - \boldsymbol{\lambda}_0)^{\mathrm{T}} \boldsymbol{g}(\boldsymbol{x}^*), \end{aligned} \tag{2.9}$$

which is the definition of a supergradient; see (2.5). □

Maximizing the dual function d in (2.7) can then be done with subgradient methods as described by Bertsekas (1999). Lemma 2.1 states that a supergradient to d is $\boldsymbol{x} - \boldsymbol{y}$. Given a $\boldsymbol{\lambda}$, one takes a step in that direction:

Start with $\boldsymbol{\lambda} = \mathbf{0}$
repeat
  Find $\boldsymbol{x}$ and $\boldsymbol{y}$ by evaluating $d(\boldsymbol{\lambda})$
  $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \tau(\boldsymbol{x} - \boldsymbol{y})$, for some step size τ
until $\boldsymbol{x} = \boldsymbol{y}$

During the maximization of the dual function, we have access to a lower bound $d(\boldsymbol{\lambda})$ of the optimal function value, and also to two feasible solutions $\boldsymbol{x}$ and $\boldsymbol{y}$ of the original problem. An upper bound on the optimal function value is then given by $\min(E(\boldsymbol{x}), E(\boldsymbol{y}))$. We can compute the relative duality gap which gives an indication of how close we are to the optimum:

duality gap

$$r = \frac{\min(E(\boldsymbol{x}), E(\boldsymbol{y})) - d(\boldsymbol{\lambda})}{\min(E(\boldsymbol{x}), E(\boldsymbol{y}))}. \tag{2.10}$$

If X is a convex set and E a convex function, the optimal duality gap is generally zero. However, subsequent chapters consider non-convex sets consisting of a finite set of points with integral coordinates. In chapter 7, an optimal gap of zero is nonetheless guaranteed as it corresponds to solving an integer linear program whose linear programming relaxation is tight.

2.2.1 Choices of Step Sizes and Their Limitations

A step size τ_k needs to be chosen in each iteration k. One of the simplest ways of doing this and still ensuring convergence is to pick $\tau_k = T/k$, with T a constant (Bertsekas, 1999). In fact, all step sizes chosen such that $\tau_k \to 0$ and $\sum_{k=1}^{\infty} \tau_k = \infty$ guarantee convergence. The step sizes for subgradient minimization can be chosen in more sophisticated ways, but minimizing a non-differentiable convex function is still much harder than minimizing a differentiable convex function. This somewhat disappointing result is a consequence of the following theorem which describes a worst-case scenario:

THEOREM 2.2 (Nesterov, 2004, p. 138) *For any $k > 0$ and $L, R > 0$, there exists a convex function f, Lipschitz continuous with constant L with a minimizer $\boldsymbol{x}^*$ such that*

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}^*) \geq \frac{LR}{2(1 + \sqrt{k+1})}, \tag{2.11}$$

for any optimization scheme with $\|\boldsymbol{x}_0 - \boldsymbol{x}^*\| \leq R$ *generating its points from subgradients* $\boldsymbol{g}$ *of* f *in the following way:*

$$\boldsymbol{x}_k \in \boldsymbol{x}_0 + \operatorname{span}\{\boldsymbol{g}(\boldsymbol{x}_0), \ldots, \boldsymbol{g}(\boldsymbol{x}_{k-1})\}. \tag{2.12}$$

The consequence of this theorem is that $O\left(\frac{1}{\varepsilon^2}\right)$ iterations are needed to get within ε of the optimal function value for any optimization method generating its iterates as (2.12). This is much worse than for differentiable convex functions, since there are first-order methods requiring $O\left(\frac{1}{\sqrt{\varepsilon}}\right)$ iterations.[1]

Despite this, we will sometimes see fast convergence for the applications in this thesis. The problems in chapter 7 converge quickly, but the convergence in chapter 8 is much slower.

2.2.2 Projected Supergradient Method

Let us instead consider a minimization problem with an inequality constraint:

$$\begin{aligned} &\inf_{\boldsymbol{x} \in X} E(\boldsymbol{x}) \\ &\text{subject to} \quad \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0}. \end{aligned} \tag{2.13}$$

The dual function of this problem is $d(\boldsymbol{\lambda}) = \inf_{\boldsymbol{x} \in X}(E(\boldsymbol{x}) + \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{g}(\boldsymbol{x}))$, but its domain is now $\{\boldsymbol{\lambda} \geq \boldsymbol{0}\}$, which turns the dual problem into a constrained maximization problem. The updating of $\boldsymbol{\lambda}$ in the algorithm on page 15 is changed into:

$$\boldsymbol{\lambda} \leftarrow \left[\boldsymbol{\lambda} + \tau \boldsymbol{g}(\boldsymbol{x})\right]^+, \tag{2.14}$$

where $[\cdot]^+$ is the orthogonal projection onto the feasible set $\{\boldsymbol{\lambda} \geq \boldsymbol{0}\}$.

[1] Close to the optimal point, first-order methods may display linear convergence, which require $O\left(\log \frac{C}{\varepsilon}\right)$ iterations. Newton's method require $O\left(\log\log \frac{C}{\varepsilon}\right)$ iterations once sufficiently close to the optimal point (Nesterov, 2004, p. 36).

2.3 Non-Linear Least Squares Problems

A non-linear least squares problem is a minimization problem with an objective function $f : \mathbf{R}^n \to \mathbf{R}^+$ of the following form:

$$f(\boldsymbol{x}) = \frac{1}{2}\sum_{j=1}^{m} \big(r_j(\boldsymbol{x})\big)^2, \tag{2.15}$$

where $m > n$ and each r_j is a function from $\mathbf{R}^n$ to $\mathbf{R}$ and is called a residual. Collect all residuals in a vector $\boldsymbol{r}(\boldsymbol{x}) = (r_1(\boldsymbol{x}), \dots, r_m(\boldsymbol{x}))^{\mathrm{T}}$ and define the Jacobian as the matrix

$$\mathsf{J}(\boldsymbol{x}) = \left(\frac{\partial r_j}{\partial x_i}(\boldsymbol{x})\right)_{\substack{i=1,\dots,n \\ j=1,\dots,m}}. \tag{2.16}$$

The gradient and Hessian of f can then be written as

$$\nabla f(\boldsymbol{x}) = \mathsf{J}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{r}(\boldsymbol{x}), \tag{2.17a}$$

$$\nabla^2 f(\boldsymbol{x}) = \mathsf{J}(\boldsymbol{x})^{\mathrm{T}}\mathsf{J}(\boldsymbol{x}) + \sum_{j=1}^{m} r_j(\boldsymbol{x})\nabla^2 r_j(\boldsymbol{x}). \tag{2.17b}$$

A useful property is that the Hessian can be approximated well by $\mathsf{J}(\boldsymbol{x})^{\mathrm{T}}\mathsf{J}(\boldsymbol{x})$ when the residuals are small. When $r_j(\boldsymbol{x}) \approx 0$, the last term in (2.17b) will be relatively small. With this approximation, a second-order model of f is

$$\begin{aligned} f(\boldsymbol{x}+\boldsymbol{h}) \approx m(\boldsymbol{h}) &= \frac{1}{2}\|\boldsymbol{r}(\boldsymbol{x})\|^2 + \boldsymbol{h}^{\mathrm{T}}\mathsf{J}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{r}(\boldsymbol{x}) + \frac{1}{2}\boldsymbol{h}^{\mathrm{T}}\mathsf{J}(\boldsymbol{x})^{\mathrm{T}}\mathsf{J}(\boldsymbol{x})\boldsymbol{h} \\ &= \frac{1}{2}\|\mathsf{J}(\boldsymbol{x})\boldsymbol{h} + \boldsymbol{r}(\boldsymbol{x})\|^2. \end{aligned} \tag{2.18}$$

The Gauss-Newton method is Newton's method applied to a least-squares problem using this approximation. It enjoys similar convergence properties. The Gauss-Newton step $\boldsymbol{h}_{\text{G-N}}$ is obtained by solving (2.18) in closed form:

$$\mathsf{J}(\boldsymbol{x})^{\mathrm{T}}\mathsf{J}(\boldsymbol{x})\boldsymbol{h}_{\text{G-N}} = -\mathsf{J}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{r}(\boldsymbol{x}). \tag{2.19}$$

The Gauss-Newton method can run into problems when $\mathsf{J}^{\mathrm{T}}\mathsf{J}$ is singular or nearly singular. The Levenberg-Marquardt method avoids this by adding a

damping factor μ when computing the step:

$$\Big(\mathsf{J}(\boldsymbol{x})^{\mathrm{T}}\mathsf{J}(\boldsymbol{x}) + \mu\mathsf{I}\Big)\,\boldsymbol{h}_{\text{L-M}} = -\mathsf{J}^{\mathrm{T}}\boldsymbol{r}(\boldsymbol{x}). \tag{2.20}$$

To make the algorithm invariant to diagonal scaling of the parameters $\boldsymbol{x}$, the matrix $\operatorname{diag}(\mathsf{J}(\boldsymbol{x})^{\mathrm{T}}\mathsf{J}(\boldsymbol{x}))$ may be used instead of the identity matrix. If the damping factor μ is very large, the method reduces to steepest descent, which is slow but robust. If μ is very small, the method is close to Gauss-Newton, which is faster. Therefore, depending on the quality of the step, λ should be increased or decreased. This is typically done via a quantity known as the gain factor:

$$g = \frac{f(\boldsymbol{x}) - f(\boldsymbol{x} + \boldsymbol{h})}{m(\mathbf{0}) - m(\boldsymbol{h})}. \tag{2.21}$$

The following procedure has been shown to perform well (Nielsen, 1999):

> If $g > g_{\min}$, set $\mu \leftarrow \mu \max\{\frac{1}{3}, 1 - (2g-1)^3\}$ and $k \leftarrow 2$.
>
> Otherwise, set $\mu \leftarrow \mu k$ and $k \leftarrow 2k$.

This procedure is what chapter 6 will use when solving image denoising problems. For many more details about the Levenberg-Marquardt method and other approaches to non-linear least squares, see the books by Nocedal and Wright (2006), and Madsen et al. (1999).

Chapter 3

Discrete Optimization

Let $\mathbf{B} = \{0, 1\}$. A function from $\mathbf{B}^n$ to $\mathbf{B}$ is called a boolean function and a function $f : \mathbf{B}^n \to \mathbf{R}$ is a pseudo-boolean function. It can be written uniquely as a polynomial in its n variables:

$$f(\boldsymbol{x}) = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{1 \le i < j \le n} a_{ij} x_i x_j + \sum_{1 \le i < j < k \le n} a_{ijk} x_i x_j x_k + \dots \tag{3.1}$$

The degree of f is m. This chapter will describe the current state of the art of minimizing pseudo-boolean functions. That is, we will study the following problem:

$$\underset{\boldsymbol{x} \in \mathbf{B}^n}{\text{minimize}} \quad f(\boldsymbol{x}). \tag{3.2}$$

This problem is well-known to be NP-hard, so for large n we have to settle for non-optimal solutions.

There are numerous application problems that can be cast in this framework, ranging from portfolio problems in operations research to the minimization of the Ising model in physics. Many graph-theoretic problems can also be turned into pseudo-boolean optimization problems, for example, maximum satisfiability and vertex cover (Boros and Hammer, 2002; Glover et al., 2002). The work on pseudo-boolean optimization in this thesis is motivated by the many applications in computer vision and machine learning. State-of-the-art methods for stereo, segmentation and image denoising are often formulated as the inference of the maximum a posteriori estimate, which can be cast as a minimization problem where the objective function is given by a pseudo-boolean function (Kolmogorov and Rother, 2007).

Naturally, there have been many approaches for solving such optimization problems, especially for quadratic ($m = 2$) pseudo-boolean problems.

One of the most successful bounds in terms of computational efficiency is the "roof dual" of a quadratic pseudo-boolean optimization problem, introduced by Hammer et al. (1984). The idea is to relax the original problem and then compute a bound on the optimal value with a polynomial time algorithm. More specifically, they showed that several different types of linear programming relaxations of quadratic pseudo-boolean problems yield the same bound—the roof dual. Chapter 4 will demonstrate another way of attaining the same bound with submodular relaxations. Further, partial solutions can be extracted from the solutions to the relaxations, a property known as persistency. Subsequent studies have refined the technique and roof duality has been shown to produce state-of-the-art results for a variety of application problems compared to other bounding techniques based on linear programming and semidefinite relaxations; see the works by Billionnet and Sutter (1992); Boros and Hammer (2002); Boros et al. (2008); and Kolmogorov and Rother (2007). A key advantage is that max-flow/min-cut computations can be applied to an appropriately constructed graph for quadratic pseudo-boolean polynomials (Boros and Hammer, 2002). I will briefly review roof duality in section 3.4.

In recent years, there has been an increasing interest in higher-order models and approaches for minimizing the corresponding objective functions in computer vision and machine learning. For example, Lan et al. (2006*b*) use approximate belief propagation with a learned higher-order model for image denoising. Similarly, Cremers and Grady (2006) learn a higher-order model for texture restoration, but the model is restricted to submodular functions which can be optimized exactly in polynomial time. Curvature regularization requires higher-order models and will be studied extensively in chapters 10 and 11.

Even global terms defined over all variables have been considered, for example, to ensure connectedness (Nowozin and Lampert, 2010) and to model co-occurrence statistics of objects (Delong et al., 2010; Ladicky et al., 2010). Another state-of-the-art example is the work by Woodford et al. (2009), where second-order surface priors are used for stereo reconstruction. The optimization strategies rely on dual decomposition (Komodakis and Paragios 2009; and also chapter 8), fusion (Kohli et al., 2009; Lempitsky et al., 2010), linear programming (Werner, 2008), belief propagation (Lan et al., 2006*b*) and, of course, max-flow/min-cut.

3.1 Submodular Functions

Define $\vee$ and $\wedge$ to be the element-wise maximum and minimum respectively. The function f in (3.1) is submodular if

submodular

$$f(\boldsymbol{x} \vee \boldsymbol{y}) + f(\boldsymbol{x} \wedge \boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y}), \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbf{B}^n. \tag{3.3}$$

Evidently, if f is a sum of submodular terms, f itself is also submodular. An equivalent (Nemhauser et al., 1978) definition is that f is submodular if and only if

$$\frac{\partial^2 f}{\partial x_i \partial x_j}(\boldsymbol{x}) \leq 0 \quad \text{for all } 0 \leq i < j \leq n \text{ and } \boldsymbol{x} \in \mathbf{B}^n. \tag{3.4}$$

The derivative is symbolical on the polynomial representation (3.1). This has a simple consequence for quadratic functions: a quadratic f is submodular if and only if all coefficients for the quadratic terms are nonpositive.

Submodular functions are in some sense the discrete analogue to convex functions (Lovász, 1983). In particular, one can define the set of subgradients at a point $\boldsymbol{x}_0$ as all vectors $\boldsymbol{g} \in \mathbf{R}^n$ satisfying

$$f(\boldsymbol{x}) \geq f(\boldsymbol{x}_0) + \boldsymbol{g}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{g}^{\mathrm{T}}\boldsymbol{x}_0, \qquad \text{for all } \boldsymbol{x} \in \mathbf{B}^n. \tag{3.5}$$

The set of discrete subgradients is also denoted $\nabla f(\boldsymbol{x}_0)$ (there should be no possibility of confusion). See the book by Fujishige (2005, chap. 4) for properties of subgradients to submodular functions and similarities to convex analysis. With this definition, submodular functions satisfy the analogous properties of convex function from the last chapter (page 11):

$$\mathbf{0} \in \nabla f(\boldsymbol{x}^*) \quad \Longleftrightarrow \quad f(\boldsymbol{x}^*) \text{ is the global minimum} \tag{A}$$

$$f, g \text{ are submodular} \quad \Longrightarrow \quad f + g \text{ is submodular} \tag{B}$$

$$\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{b} \text{ is submodular for all } \boldsymbol{a}, \boldsymbol{b} \in \mathbf{R}^n. \tag{C}$$

It is always possible to minimize a submodular f in polynomial time, but the best known algorithm is not very efficient. The algorithm by Iwata (2001) runs in $O((n^6\gamma + n^7)\log n)$ time, where γ is the time required to evaluate f. Iwata (2002) has also developed a fully combinatorial algorithm (using additions, multiplication, comparisons and rounding), which runs in $O(n^9 \log^2 n)$ function evaluations and operations. If the degree $m \leq 3$,

f can be minimized via graph cuts, but Živný et al. (2009) showed that this is not possible in general when $m \geq 4$. In fact, even deciding whether f is submodular is co-NP-complete for $m \geq 4$ (Crama, 1989). This fact will be important later in my thesis and since the proof is quite short I am reproducing a variant of it here:

THEOREM 3.3 (Crama 1989) *Recognizing whether a quartic pseudo-boolean function f is submodular is co-NP-complete.*

Proof. Let $c_1, \ldots, c_n$ be a set of integers and define

$$d(\boldsymbol{x}) = \sum_{i=1}^{n} c_i x_i \qquad \text{and} \qquad C = \sum_{i=1}^{n} |c_i|. \tag{3.6}$$

It is now enough to study the function

$$f(\boldsymbol{x}, x_{n+1}, x_{n+2}) = (1 - d(\boldsymbol{x})^2) x_{n+1} x_{n+2} - 100C^2 \sum_{i=1}^{n} \sum_{j=i+1}^{n+2} x_i x_j. \tag{3.7}$$

All quadratic monomials except $x_{n+1}x_{n+2}$ are present in the second sum with negative coefficients of large magnitude. Therefore, f is nonsubmodular if and only if $\frac{\partial^2 f}{\partial x_{n+1} \partial x_{n+2}}(\boldsymbol{x}) > 0 \iff d(\boldsymbol{x}) = 0$ for some $\boldsymbol{x}$. This is equivalent to the existence of a subset of $c_1, \ldots, c_n$ whose sum is zero, which is the well-known NP-complete subset-sum problem. □

The method in chapter 4 works by optimizing over the set of submodular functions of a fixed degree. This is easy when $m \leq 3$, but for $m = 4$ the above theorem is one of the obstacles encountered (the other being the fact that using graph cuts for the minimization is no longer possible).

3.2 Submodularity and Minimum Cuts

Let G be a graph with n nodes (or vertices) in addition to two special nodes s and t. A non-negative weight w_{ij} is associated to each pair (i, j) of nodes, which is greater than zero if there is an edge between i and j. Because no self-loops are allowed in the graph, $w_{i,i} = 0$ is true for all i.

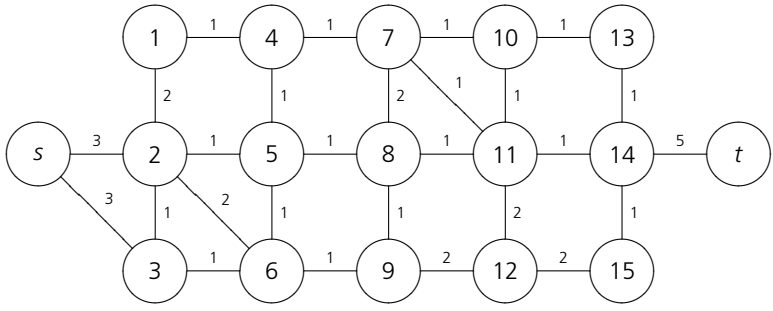


Figure 3.1: Example of an undirected graph. One minimum cut in this graph is $S = \{s, 1 \dots 6\}$ and $T = \{t, 7 \dots 15\}$. Another is $S' = \{s, 1 \dots 12, 15\}$ and $T' = \{t, 13, 14\}$. Both have the value 3.

A cut is a partition of the nodes of G into two sets S and T, such that $s \in S$ and $t \in T$. The value of a cut is the sum of all weights of edges leading from S to T. If all these edges are removed, there is no path from s to t, hence the name "cut." The minimum cut of G is the cut with the smallest possible value. Figure 3.1 shows an example of an undirected graph ($w_{ij} = w_{ji}$) and its minimum cuts. cut minimum cut

Now introduce indicator variables $\boldsymbol{x} = (x_1, \dots x_n)$, where x_i is equal to 0 if node i is in S and 1 otherwise. For convenience, let $\mathcal{N}_i$ denote the neighborhood of i, that is, the set of all nodes $j \neq s, t$ such that there is an edge between i and j. For example, $\mathcal{N}_5 = \{2, 4, 6, 8\}$ in figure 3.1. With this notation the problem of finding the minimum cut is to

$$\underset{\boldsymbol{x} \in \{0,1\}^n}{\text{minimize}} \quad \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i} w_{ij}(1 - x_i)x_j + \sum_{i=1}^{n} w_{si}(1 - x_i) + \sum_{i=1}^{n} w_{it}x_i. \tag{3.8}$$

There are efficient methods for solving this optimization problem if $w_{ij} \geq 0$ for all $i, j \in \{1 \dots n\}$. The algorithm by Boykov and Kolmogorov (2004) is used extensively in the computer vision community.

Some pseudo-boolean functions can be minimized by computing the minimal cut of a graph. Consider a function of two boolean variables, which is often called a clique of size 2. It can be written as a polynomial of degree clique

two:[1]

$$\begin{aligned} E(x_1, x_2) = E_{11}x_1x_2 + E_{10}x_1(1 - x_2) \\ + E_{01}(1 - x_1)x_2 + E_{00}(1 - x_1)(1 - x_2), \end{aligned} \tag{3.9}$$

or, ignoring constant terms:

$$\begin{aligned} E(x_1, x_2) = \big(E_{11} - E_{10} - E_{01} + E_{00}\big)x_1x_2 \\ + E_{10}x_1 + E_{01}x_2 - E_{00}(x_1 + x_2). \end{aligned} \tag{3.10}$$

This expression can be written on the form (3.8) if and only if the coefficient in front of x_1x_2 is non-positive (since w_{ij} is required to be non-negative). Written out, this requirement is

$$E_{00} + E_{11} \leq E_{01} + E_{10}. \tag{3.11}$$

This property of the clique E is submodularity. The fact that submodular functions can be efficiently minimized by computing a minimum cut was shown by Ivănescu (1965) for quadratic functions and Billionnet and Minoux (1985) for cubic functions.

3.3 Linearization

Take a monomial $cx_{i_1}x_{i_2}\cdots x_{i_d}$ appearing in f of degree d larger than 1. With the goal of computing a linear programming relaxation of (3.2), this monomial can be associated with a new variable $y \in \{0, 1\}$. Forcing the equality $y = x_{i_1}x_{i_2}\cdots x_{i_d}$ is possible with the following linear constraints:

$$y \leq x, \quad \forall x \in \{x_{i_1}, x_{i_2} \dots x_{i_d}\} \tag{3.12a}$$

$$y \geq x_{i_1} + x_{i_2} + \cdots + x_{i_d} - d + 1 \tag{3.12b}$$

These linear inequalities can easily be seen to uniquely determine y. In fact, if $c < 0$, only constraints (3.12a) are needed and only (3.12b) is needed

[1] $E(0, 0)$ is abbreviated as E_{00} etc. E is defined by the four values in its range: $E_{00}, \dots E_{11}$.

when $c > 0$ (because of the minimization of f). Problem (3.2) is then equivalent to the following integer program:

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{y}}{\text{minimize}} \quad & a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{1 \le i < j \le n} a_{ij} y_{ij} + \sum_{1 \le i < j < k \le n} a_{ijk} y_{ijk} + \dots \\ \text{subject to} \quad & \text{all } y \text{ satisfy (3.12a) and (3.12b),} \\ & \text{all } x \text{ and } y \in \{0, 1\}. \end{aligned} \tag{3.13}$$

For quadratic polynomials, this linear program is related to the discrete Rhys form of f (Rhys, 1970), which was used by Hammer et al. (1984) to introduce roof duality. It is natural to relax the domain of the variables from $\{0, 1\}$ to $[0, 1]$, thus obtaining a linear program. This will always give a lower bound to problem (3.2).

3.4 Roof Duality

The problem of minimizing a quadratic pseudo-boolean function is already a hard problem. It covers, as will be discussed in the next section, all pseudo-boolean functions by possible introduction of auxiliary variables. Therefore, it is in some sense enough to consider

$$\underset{\boldsymbol{x} \in \mathbf{B}^n}{\text{minimize}} \quad a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{1 \le i < j \le n} a_{ij} x_i x_j. \tag{3.14}$$

This is known as a quadratic pseudo-boolean optimization problem or QPBO. It can be reformulated into (3.13) according to the previous section. Relaxing the constraint $\boldsymbol{x} \in \{0, 1\}^n$ to $\boldsymbol{x} \in [0, 1]^n$ results in a linear program. It can be shown by looking at the determinant of all submatrices of the linear constraints and employing Cramer's rule that there always exists an optimal solution $\hat{\boldsymbol{x}} \in \left\{0, 1, \frac{1}{2}\right\}^n$ to the relaxation of (3.13) (Hammer et al., 1984; Adams and Dearing, 1994). The optimal objective value of the linear program is obviously a lower bound to the minimum of f and is one way of computing the roof dual of f.

QPBO

roof dual

Given the solution of the linear programming relaxation, one can often say a lot about the solution to the original problem (3.14). This is due to the following persistency result:

persistency

THEOREM 3.4 *Let $\hat{\boldsymbol{x}} \in \left\{0, 1, \frac{1}{2}\right\}^n$ be a solution to (3.13) and $L = \left\{i \,|\, x_i \neq \frac{1}{2}\right\}$. Then there exists a solution $\boldsymbol{x}^*$ to (3.14) with $x_i^* = \hat{x}_i$ for all $i \in L$.*

assigned variable

Theorem 3.4 motivates the following terminology: If $i \neq \frac{1}{2}$ the linear programming relaxation is said to have assigned variable variable i, otherwise x_i is said to be unassigned, which is sometimes written $x_i = \varnothing$. Thus, the solution of the relaxation is in $\{0, 1, \varnothing\}^n$. Linear programs can be solved in polynomial time, but there are even more efficient algorithms. A solution satisfying Theorem 3.4 with the same lower bound of the optimal value as in (3.13) can be computed as a minimum cut in a graph with $2n$ nodes (Boros and Hammer, 2002).

probing

If the solution contains several unassigned variables, it can sometimes be improved with a technique called probing. The idea is to pick an index i with $x_i = \varnothing$. Then two problems are solved, with x_i fixed to 0 and 1, respectively. If a previously unlabeled variable x_j is equal to the same value (not $\varnothing$) in both of the new solutions, then x_j is permanently fixed to this value, as it is known to be optimal. The process is then repeated. Probing is sometimes very effective in finding globally optimal solutions.

The persistency results of a linear programming relaxation were introduced by Hammer et al. (1984). They also gave an efficient method of calculating the lower bound using a minimum cut in a graph. The graph construction by Boros et al. (1991) is what is commonly used today. The probing technique is due to Boros et al. (2006). Their work was made popular in the computer vision community by an efficient implementation by Rother et al. (2007*a*; 2007*b*), which is available for download. In this context, "QPBO" sometimes refers to solving the linear programming relaxation with a minimum cut. Blake et al. (2011) call it the "BHS" method after the authors Boros, Hammer, and Sun.

3.5 Reductions

The standard way to handle problems where $m > 2$ is to employ reductions. By introduction of new variables, the problem of minimizing $f(\boldsymbol{x})$ can always be reduced to minimizing another quadratic pseudo-boolean function $g(\boldsymbol{x}, \boldsymbol{z})$ with more variables. The function g is equivalent to f in the sense that $\min_{\boldsymbol{z}} g(\boldsymbol{x}, \boldsymbol{z}) = f(\boldsymbol{x})$ and is in general not unique.

One way (Boros and Hammer, 2002) of reducing higher-order monomials to quadratic monomials (sometimes called higher-order clique reduction, or HOCR) is to observe that if

$$g(x_1, x_2, z) = x_1x_2 - 2x_1z - 2x_2z + 3z, \tag{3.15}$$

then

$$\begin{aligned} x_1x_2 = z &\iff g(x_1, x_2, z) = 0 \quad \text{and} \\ x_1x_2 \neq z &\iff g(x_1, x_2, z) > 0. \end{aligned} \tag{3.16}$$

Higher-order monomials like $x_1x_2x_3$ can then be replaced by the expression $zx_3 + Mg(x_1, x_2, z)$ in any minimization problem, where M is a constant larger than any value of f. This procedure can be iterated until no monomials of higher degree than 2 remain.

However, this reduction does not perform well in practice, since the resulting optimization problems tend to be very hard to solve, due to the introduction of the large coefficients. Ishikawa (2011, Sec. 3.3) discusses this. There are, however, alternative reductions which do not introduce large coefficients. If the coefficient in front of a monomial is negative, one can use the following identity:

$$-x_1x_2 \cdots x_d = \min_z z(d - 1 - x_1 - x_2 \ldots - x_d). \tag{3.17}$$

This eliminates the higher-order monomial at the cost of one introduced variable z and d new monomials of degree 2. If the coefficient is positive, the situation is slightly more complicated. For example,

$$\begin{aligned} x_1x_2x_3 = \min_z z(x_1 + x_2 + x_3 - 1) \\ + x_1x_2 + x_1x_3 + x_2x_3 - (x_1 + x_2 + x_3) + 1 \end{aligned} \tag{3.18}$$

and

$$\begin{aligned} x_1x_2x_3x_4 = \min_z z(3 - 2x_1 - 2x_2 - 2x_3 - 2x_4) \\ + x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4. \end{aligned} \tag{3.19}$$

The latter was introduced by Ishikawa (2011) and there are many other alternatives. The following theorem gives a general method:

THEOREM 3.5 (Ishikawa, 2011) *For $x_1, \ldots, x_d \in \mathbf{B}$,*

$$x_1 x_2 \cdots x_d = \min_{\boldsymbol{z} \in \mathbf{B}^d} \sum_{i=1}^{n_d} z_i \Big(c_{i,d}(2i - S_1) - 1 \Big) + S_2, \tag{3.20}$$

where

$$S_1 = \sum_{1 \le i \le k} x_i, \quad S_2 = \sum_{1 \le i < j \le k} x_i x_j, \quad n_d = \left\lfloor \frac{d-1}{2} \right\rfloor \tag{3.21}$$

and

$$c_{i,d} = \begin{cases} 1 & \text{if } d \text{ is odd and } i = d, \\ 2 & \text{otherwise.} \end{cases} \tag{3.22}$$

Combining (3.17) and Theorem 3.5 gives a complete procedure of reducing any polynomial to a quadratic one without introducing very large coefficients. This reduction method works reasonably well in practice, but a major part of this thesis is devoted to minimizing the higher-order polynomials directly, instead of first converting them to quadratic ones.

Reductions do not have to be applied for each term individually; there are more economical reductions available:

THEOREM 3.6 (Fix et al., 2011) *Consider a set $\mathcal{H}$ of terms each of which contains a common subset of variables C, whose hyperedges H have positive weights $\alpha_H > 0$. For any assignment of the boolean variables $x_1, \ldots, x_n$,*

$$\sum_{H \in \mathcal{H}} \alpha_H \prod_{j \in H} x_j = \min_{y \in \mathbf{B}} \left(\sum_{H \in \mathcal{H}} \alpha_H \right) y \prod_{j \in C} x_j + \sum_{H \in \mathcal{H}} \alpha_H \bar{y} \prod_{j \in H \setminus C} x_j.$$

For the remainder of this thesis, C will only contain a single variable, that is $C = \{x_k\}$ in all experiments using these reductions. This is consistent with the experiments performed by Fix et al. (2011).

It is not obvious which reduction one should pick. As an example, consider the problem of minimizing the following cubic polynomial f over $\mathbf{B}^3$:

$$f(\boldsymbol{x}) = -2x_1 + x_2 - x_3 + 4x_1x_2 + 4x_1x_3 - 2x_2x_3 - 2x_1x_2x_3. \tag{3.23}$$

The standard reduction scheme (3.17) can be applied to reduce this to a quadratic function. Roof duality gives a lower bound of $f_{\min} \geq -3$, but it does not reveal how to assign any of the variables in $\boldsymbol{x}$. However, there are many possible reduction schemes from which one can choose. Another possible reduction is $-x_1x_2x_3 = \min_{z\in\mathbf{B}} z(-x_1 + x_2 + x_3) - x_1x_2 - x_1x_3 + x_1$. For this reduction, the roof dual bound is tight and the optimal solution $\boldsymbol{x}^* = (0, 1, 1)$ is obtained. This simple example illustrates two facts: (i) different reductions lead to different lower bounds and (ii) it is not an obvious matter how to choose the optimal reduction. I will return to this example in chapter 4.

3.6 Beyond Boolean Variables

For many low-level problems in computer vision, a Markov random field is used for modeling and the resulting inference problem consists of estimating some unknown quantity (e.g., depths) from one or several observations. The objective function for such a minimization problem is typically of the form

$$E(\boldsymbol{w}) = \underbrace{\sum_{i=1}^{n} E_i(w_i)}_{E_{\text{data}}(\boldsymbol{w})} + \underbrace{\sum_{i<j} E_{ij}(w_i, w_j) + \sum_{i<j<k} E_{ijk}(w_i, w_j, w_k) + \dots}_{E_{\text{smooth}}(\boldsymbol{w})} \tag{3.24}$$

As was explained in the first chapter, the data term E_{data} specifies the agreement between $\boldsymbol{w}$ and the observations. The smoothness term E_{smooth} measures how well $\boldsymbol{w}$ agrees with prior information such as smoothness and noise levels in the image.

In many applications, $\boldsymbol{w}$ is not a boolean vector but instead $\boldsymbol{w} \in \mathbf{L}^n$, where, for example, $\mathbf{L} = \{1, \dots, K\}$ or $\mathbf{L} = \mathbf{R}$. The number of discrete values K is sometimes called the number of labels. Although optimization problems with a finite number of possible values for each variable can be converted into boolean problems by binary encoding, this is normally not the preferred method as the resulting optimization problems tend to be very hard to solve. In practice, the multi-label problem is reduced to the boolean case by iterative algorithms (Boykov et al., 2001; Lempitsky et al., 2010): Given a current solution $\boldsymbol{w}^{(t)}$ and a proposal $\boldsymbol{y}^{(t)}$, a new and better solution

labels

proposal

is constructed by combining them in an optimal way. The combination is determined by a solution $\boldsymbol{x}^*$ to a pseudo-boolean minimization problem:

$$w_i^{(t+1)} = \begin{cases} y_i^{(t)} & \text{if } x_i^* = 1 \\ w_i^{(t)} & \text{if } x_i^* = 0 \text{ or unassigned.} \end{cases} \tag{3.25}$$

This local optimization method is sometimes called fusion and can be seen as a generating set search (or direct search) method (Kolda et al., 2003). However, the number of points considered in each iteration is *exponential* in the number of dimensions, whereas the number of points evaluated in other direct search methods normally is linear. The fact that solving a pseudo-boolean problem effectively explores an exponential set of points efficiently is one way of explaining the success of fusions.

The pseudo-boolean objective function $F^{(t)}(\boldsymbol{x})$ at iteration t seeks to merge $\boldsymbol{y}$ and $\boldsymbol{w}$ optimally. The data terms are given by

$$F_i^{(t)}(x_i) = E_i(y_i)x_i + E_i(w_i)\bar{x}_i, \tag{3.26}$$

where $\bar{x} = 1 - x$. Similarly,

$$F_{ij}^{(t)}(x_i, x_j) = E_{ij}(y_i, y_j)x_i x_j + E_{ij}(y_i, w_j)x_i\bar{x}_j + E_{ij}(w_i, y_j)\bar{x}_i x_j + E_{ij}(w_i, w_j)\bar{x}_i\bar{x}_j \tag{3.27}$$

and so forth. Because of this, the maximum number of variables appearing in a single term of E_{smooth} can be referred to as its degree. The partial solution obtained via roof duality has a property called autarky, which ensures that the objective function value will never increase for any proposed solution if w_i is left unmodified whenever $x_i = \varnothing$. Autarky will be discussed further in chapter 4.

autarky

A special case of the fusion algorithm is when $y_i = \alpha$ for all i. This variant is referred to as α-expansion. An expansion for every pixel value is performed until the function value does not decrease for any value of α. This usually happens within a few loops through the set of pixel values. While reaching the global optimum is not guaranteed, it is possible to prove (Boykov et al., 2001, Theorem 6.1) that the α-expansion algorithm will end up within a constant of the optimal function value. The constant depends on the type of pairwise interactions between the variables and is under favorable circumstances equal to 2.

expansion

Computing the optimal α-expansion can often be done exactly. With the correct assumptions on the interactions between the variables, all pseudo-boolean problems will be submodular and can be solved readily as minimum cut problems. The graph construction by Boykov et al. (2001, sec. 5) uses additional nodes where $\alpha \neq w_i^{(t)} \neq w_j^{(t)} \neq \alpha$ and $j \in \mathcal{N}_i$, although the construction can be done without extra nodes (Kolmogorov and Zabih, 2004).

In addition to α-expansion the related α/β-swap (Boykov et al., 2001) may also be used. Instead of considering a single value α and expanding it as much as possible, the swap considers two possible labels $\alpha, \beta \in \{1 \dots K\}$ and computes the optimal swap between the two. It can handle more general functions than expansion while still ensuring submodularity. The downsides are the lack of any guarantee to end up close to the optimum and the fact that each full iteration now consists of solving $\binom{K}{2} = (K^2 - K)/2$ boolean problems instead of K.

swap

Geometric Constraints. Several recent papers have extended the framework for multi-label minimization for which global solutions are tractable. If there is a geometric relationship between the labels (e.g. the image region number 2 is contained in region number 1) the energy can sometimes still be minimized exactly (Delong and Boykov, 2009). This framework will be put to use in chapter 12.

3.7 Branch and Bound

Let us now once again consider the problem of minimizing a function $E(\boldsymbol{x})$, where $\boldsymbol{x}$ takes values in an arbitrary set X. If X is partitioned into $\mathcal{X} = \{S_1, \dots, S_n\}$, we clearly have:

$$\inf_{\boldsymbol{x} \in X} E(\boldsymbol{x}) = \inf_{i=1\dots n} \inf_{\boldsymbol{x} \in S_i} E(\boldsymbol{x}). \tag{3.28}$$

Let E_S^* be the optimal value of E on the set S. In general, E_S^* is hard to compute. Now assume that we can compute bounds $\underline{E_S^*} \leq E_S^* \leq \overline{E_S^*}$ much more efficiently than computing E_S^* itself. With a method to compute $\underline{E_S^*}$ and $\overline{E_S^*}$ we can compare the set S to other parts of the search space $\overline{X}$. To see this, assume we have computed these bounds for two sets S and T. If $\overline{E_S^*} < \underline{E_T^*}$, we can disregard the set T from further processing,

$\mathcal{X} = \{X\}$
repeat
  $\overline{E^*} \leftarrow \min_{S \in \mathcal{X}} \overline{E_S^*}$
  $\underline{E^*} \leftarrow \max_{S \in \mathcal{X}} \underline{E_S^*}$
  foreach $S \in \mathcal{X}$ **do**
    if $\overline{E^*} < \underline{E_S^*}$ **then**
      Remove S from $\mathcal{X}$
    else
      Replace S in $\mathcal{X}$ by a partition of S
    end
  end
until $\overline{E^*} - \underline{E^*}$ *is small enough*

Figure 3.2: Branch and bound algorithm to minimize E over the set X.

since it is guaranteed to not contain the global minimum of E. An upper bound is usually easy to compute, since it is possible to choose $\overline{E_S^*} = E(\boldsymbol{x})$ where $\boldsymbol{x}$ is any point in S. Finding good ways to compute lower bounds is harder and problem-specific. As an example of a lower bound, consider the minimum cut problem. If there is a path from s to t, the lowest arc cost in that path is a lower bound of the minimum cut value.

The process of minimizing E consists of two steps which are iterated. First, bounds are computed for every set in the partition after which sets known not to contain the global minimum are removed. Second, any remaining sets are further subdivided. This is the branch step. Formally, the algorithm starts with $\mathcal{X} = \{X\}$ and proceeds as shown in figure 3.2.

As an example, consider the following pseudo-boolean optimization problem:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \boldsymbol{x}^{\mathrm{T}} \mathsf{A} \boldsymbol{x} + \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{x} \in \{0,1\}^n, \end{aligned} \tag{3.29}$$

where A is a symmetric positive definite matrix. In the terminology of the previous section, $E(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}} \mathsf{A} \boldsymbol{x} + \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x}$ and

$$X = \{0,1\}^n. \tag{3.30}$$

One simple way of branching (dividing the search space) is to use the following partition $\{S_0, S_1\}$ of X:

$$\begin{aligned} S_0 &= X \cap \{\boldsymbol{x} \,|\, x_1 = 0\} \\ S_1 &= X \cap \{\boldsymbol{x} \,|\, x_1 = 1\}. \end{aligned} \tag{3.31}$$

This can be continued for more variables:

$$\begin{aligned} S_{00} &= X \cap \{\boldsymbol{x} \,|\, x_1 = 0, x_2 = 0\} \\ S_{01} &= X \cap \{\boldsymbol{x} \,|\, x_1 = 0, x_2 = 1\}. \\ &\vdots \end{aligned} \tag{3.32}$$

Both $\{S_0, S_1\}$ and $\{S_{00}, S_{01}, S_{10}, S_{11}\}$ are partitions of X, and so on for higher numbers of fixed elements of $\boldsymbol{x}$. A lower bound can be computed by replacing the set X with $\tilde{X} = [0, 1]^n$ and so on for $\tilde{S}_0, \tilde{S}_1$ etc. We clearly have $X \subset \tilde{X}$, $S_0 \subset \tilde{S}_0$ etc. which means that minimizing E over these larger sets will give a lower bound of the optimum values of the boolean sets. For an upper bound, we can round the real-valued solution vector and evaluate $E(\boldsymbol{x})$ for the resulting boolean vector. The following numerical example will further explain the procedure. Let us take

$$\mathsf{A} = \begin{bmatrix} 46 & 3 & 16 & -15 & -12 \\ 3 & 59 & -10 & -8 & -12 \\ 16 & -10 & 39 & -10 & 16 \\ -15 & -8 & -10 & 27 & 8 \\ -12 & -12 & 16 & 8 & 46 \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} -6 \\ -24 \\ -47 \\ -18 \\ -16 \end{bmatrix}. \tag{3.33}$$

The complete optimization procedure is shown as a tree in figure 3.3. It starts by computing bounds for S_0 and S_1. This immediately results in S_1 to be discarded. S_0 is split into S_{00} and S_{01}. For S_{01} the optimal value is known since the relaxed and the rounded values are equal. Further branching down the tree shows that it is in fact globally optimal for X.

Branch and bound is a very general method and its efficiency highly depends on the quality of the computed bounds. The next chapter uses branch and bound to evaluate lower bounds to pseudo-boolean problems and chapter 9 uses it to compute optimal solutions to a special class of segmentation problems.

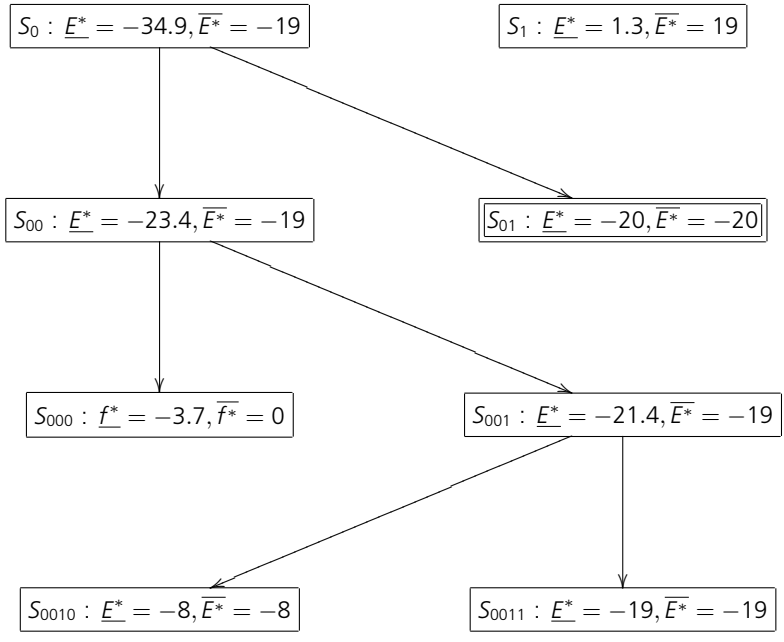


Figure 3.3: The branch and bound tree for the boolean optimization problem (3.29), (3.33). The optimal value is -20. Each line is one outer iteration in figure 3.2.

Part I

Pseudo-Boolean Optimization

Chapter 4

Generalized Roof Duality

Roof duality and pseudo-boolean minimization were introduced in chapter 3. The purpose of this chapter is to derive a generalization of roof duality and explore its properties. Just as in the previous chapter, a pseudo-boolean function $f\colon \mathbf{B}^n \to \mathbf{R}$ where $\mathbf{B} = \{0, 1\}$ is represented by a multilinear polynomial of the form

$$f(\boldsymbol{x}) = \sum_i a_i x_i + \sum_{i<j} a_{ij} x_i x_j + \sum_{i<j<k} a_{ijk} x_i x_j x_k + \ldots, \qquad (4.1)$$

of degree$(f) = m$. Without loss of generality, we have assumed that $f(\mathbf{0}) = 0$. This chapter considers the following optimization problem:

$$\underset{\boldsymbol{x}\in\mathbf{B}^n}{\text{minimize}} \quad f(\boldsymbol{x}). \qquad (4.2)$$

The goal is to provide means to compute effective bounds on the optimal value for large-scale problems (that is, when n is of the order of several thousands of variables). Submodular function minimization has polynomial time complexity (Lovász, 1983), a fact that makes submodular relaxations practical. The framework applies for arbitrary degree problems, but I will only briefly discuss $m \geq 5$ and the experiments will focus solely on $m \leq 4$. The main contributions are (i) how one can define a general bound for any order (for which the quadratic case is a special case) and (ii) how one can efficiently compute solutions that attain this bound in polynomial time. These contributions are of course coupled—it makes little sense to define a bound that is not tractable.

The inspiration for this work comes primarily from three different sources. First of all, as max-flow/min-cut computations are considered to be state-of-the-art for quadratic pseudo-boolean polynomials (Boros et al., 2008; Kolmogorov and Rother, 2007), reduction techniques of higher-order

polynomials ($m > 2$) have been explored, for example, by Lu and Williams (1987), Freedman and Drineas (2005), Rother et al. (2009), Ishikawa (2011), Gallagher et al. (2011) and Fix et al. (2011), which have been summarized in chapter 3. However, all of these approaches choose suboptimally between a fixed set of possible reductions. Then, there exist several suggestions for generalizations of roof duality for higher-order polynomials. Lu and Williams (1987) present a roof duality framework based on reduction, but at the same time the authors note that their roof duality bound depends on which reductions are applied. Kolmogorov (2010) proposes submodular relaxations as a generalization for roof duality, but no method is given for constructing or minimizing such relaxations. The framework of this chapter also builds on using submodular relaxations. Finally, the complete characterization of submodular functions up to degree $m = 4$ is instrumental to this work; see the works by Billionet and Minoux (1985), Promislow and Young (2005) and Živný et al. (2009).

Outline. The next section introduces the concept of submodular relaxations and formulate the problem of finding relaxations that attain the maximum lower bound. Then, section 4.2 shows how to construct such relaxations in closed-form for quadratic pseudo-boolean functions. This construction turns out to be equivalent to the quadratic roof dual relaxation. For higher-order functions, things are more complicated. The generalized roof dual bound is analyzed in section 4.3 and a polynomial-time algorithm is derived to compute the roof dual bound. Sections 4.4 and 4.5 analyze cubic and quartic relaxations in more detail. A faster, but non-optimal heuristic method for constructing the relaxations is also proposed in Section 4.6.

4.1 Submodular Relaxations

I will follow the framework of submodular (and bisubmodular) relaxations introduced by Kolmogorov (2010). Consider the optimization problem in (4.2) where f has n variables. The idea in this chapter is to instead study the following tractable problem over a larger domain:

$$\min_{(\boldsymbol{x},\boldsymbol{y})\in\mathbf{B}^{2n}} g(\boldsymbol{x},\boldsymbol{y}), \tag{4.3}$$

where $g\colon \mathbf{B}^{2n} \to \mathbf{R}$ is a pseudo-boolean function that satisfies the three conditions

$$g(\boldsymbol{x}, \bar{\boldsymbol{x}}) = f(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \mathbf{B}^n, \tag{A}$$

$$g \text{ submodular}, \tag{B}$$

$$g(\boldsymbol{x}, \boldsymbol{y}) = g(\bar{\boldsymbol{y}}, \bar{\boldsymbol{x}}), \quad \forall (\boldsymbol{x}, \boldsymbol{y}) \in \mathbf{B}^{2n} \text{ (symmetry)}. \tag{C}$$

Here, $\bar{\boldsymbol{x}} = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n) = (1 - x_1, 1 - x_2, \ldots, 1 - x_n)$. The reason for requirement (A) is that if the range of f is included in the range of g then the minimum of g is a lower bound to the minimum of f. If the computed minimizer $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ of the relaxation g happens to fulfill $\boldsymbol{x}^* = \bar{\boldsymbol{y}}^*$ then, of course, $\boldsymbol{x}^*$ is a minimizer of f as well. Even if it is not the case that $\boldsymbol{x}^* = \bar{\boldsymbol{y}}^*$, we still obtain a lower bound on f and, as we shall see, it is possible to extract a partial solution for a minimizer of f.

Requirement (B) is also fairly obvious. Since minimizing g must be feasible, requiring that g is submodular is natural. The last requirement will be motivated below.

4.1.1 Problem Formulation

Let $f_{\min}$ denote the unknown minimum value of f, that is, $f_{\min} = \min f(\boldsymbol{x})$. Ideally, $g(\boldsymbol{x}, \boldsymbol{y}) \geq f_{\min}$ for all points $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbf{B}^{2n}$. This is evidently not possible in general. However, one could try to find the g that gives the largest possible bound, $\max \min_{\boldsymbol{x},\boldsymbol{y}} g(\boldsymbol{x}, \boldsymbol{y})$, that is,

$$\begin{aligned} \max_{g,\ell} \quad & \ell \\ \text{subject to} \quad & g(\boldsymbol{x}, \boldsymbol{y}) \geq \ell, \quad \forall (\boldsymbol{x}, \boldsymbol{y}) \in \mathbf{B}^{2n}, \\ & g \text{ satisfies (A)–(C)}. \end{aligned} \tag{4.4}$$

A relaxation g that provides the maximum lower bound will be called optimal. Note that the problem involves exponentially many constraints on g and therefore may seem like an intractable problem. In the quadratic case, the lower bound coincides with the roof duality bound (Kolmogorov, 2010) and therefore this bound will be referred to as *generalized roof duality*. The general case will be analyzed in section 4.3, which shows how to compute the solution when the maximum is taken over a restricted set of submodular functions in spite of exponentially many constraints on g.

Symmetry. The last requirement, (C), which specifies symmetry, is perhaps not so obvious. Its motivation is two-fold:

- Restricting ourselves to this class of symmetric functions does not affect the obtained lower bound, i.e. there is a symmetric optimal relaxation.
- For a symmetric g it is possible to prove persistency—recall that this means a possibility of extracting a partial solution even though computing the complete, globally optimal solution is intractable.

A pseudo-boolean function g can be decomposed into a symmetric and an antisymmetric part, $g(\boldsymbol{x}, \boldsymbol{y}) = g_{\text{sym}}(\boldsymbol{x}, \boldsymbol{y}) + g_{\text{asym}}(\boldsymbol{x}, \boldsymbol{y})$, where the symmetric part is defined by $g_{\text{sym}}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}(g(\boldsymbol{x}, \boldsymbol{y}) + g(\bar{\boldsymbol{y}}, \bar{\boldsymbol{x}}))$ and the antisymmetric part by $g_{\text{asym}}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}(g(\boldsymbol{x}, \boldsymbol{y}) - g(\bar{\boldsymbol{y}}, \bar{\boldsymbol{x}}))$. Note that $g_{\text{sym}}(\boldsymbol{x}, \boldsymbol{y}) = g_{\text{sym}}(\bar{\boldsymbol{y}}, \bar{\boldsymbol{x}})$ and $g_{\text{asym}}(\boldsymbol{x}, \boldsymbol{y}) = -g_{\text{asym}}(\bar{\boldsymbol{y}}, \bar{\boldsymbol{x}})$. If g satisfies requirements (A) and (B), then so does g_{sym}.

Consider the function g evaluated at the two points $(\boldsymbol{x}, \boldsymbol{y})$ and $(\bar{\boldsymbol{y}}, \bar{\boldsymbol{x}})$. The function values should be larger than some lower bound ℓ, hence $g(\boldsymbol{x}, \boldsymbol{y}) = g_{\text{sym}}(\boldsymbol{x}, \boldsymbol{y}) + g_{\text{asym}}(\boldsymbol{x}, \boldsymbol{y}) \geq \ell$ and $g(\bar{\boldsymbol{y}}, \bar{\boldsymbol{x}}) = g_{\text{sym}}(\boldsymbol{x}, \boldsymbol{y}) - g_{\text{asym}}(\boldsymbol{x}, \boldsymbol{y}) \geq \ell$. To achieve a maximum lower bound, it follows that $g_{\text{asym}}(\boldsymbol{x}, \boldsymbol{y}) = 0$. Thus, to solve (4.4), restricting the attention to symmetric pseudo-boolean functions is enough.

Existence. The existence of feasible solutions for the optimization problem (4.4) can be seen from the following explicit example:

$$g(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} f(\boldsymbol{x}) & \text{if } \boldsymbol{y} = \bar{\boldsymbol{x}}, \\ -M \cdot \#\{i \mid x_i = y_i\} & \text{otherwise,} \end{cases} \tag{4.5}$$

where M is a sufficiently large constant. Because $\min g = -Mn$, this is, in some sense, the worst possible choice of g. Conditions (A) and (C) are satisfied by construction and submodularity (B) can be easily verified.

Linearity. Provided that the relaxation g is represented by a multilinear polynomial, constraint (A) is a linear equality constraint in the coefficients of g, as is constraint (C). The submodularity constraint can be expressed via linear inequality constraints; see sections 4.4 and 4.5. Therefore, the

optimization problem (4.4) is a linear program where the variables are the coefficients of g (and ℓ). As there always exists a feasible solution and the objective function is bounded from above, the concept of an optimal relaxation is well-defined.

Notation. Just as in the previous chapter, $\boldsymbol{x} \wedge \boldsymbol{y}$ and $\boldsymbol{x} \vee \boldsymbol{y}$ mean elementwise minimum and maximum, respectively. Let

$$\mathbf{S}^n = \{(\boldsymbol{x}, \boldsymbol{y}) \in \mathbf{B}^{2n} \mid (x_i, y_i) \neq (1,1), i = 1, \dots, n\}. \tag{4.6}$$

For $(x_1, y_1) \in \mathbf{S}^1$ and $(x_2, y_2) \in \mathbf{S}^1$, the operators $\sqcap$ and $\sqcup$ are defined by

$$\begin{aligned}(x_1, y_1) \sqcap (x_2, y_2) &= (x_1 \wedge x_2, y_1 \wedge y_2)\\ (x_1, y_1) \sqcup (x_2, y_2) &= \begin{cases}(0,0) & \text{if } (x_1 \vee x_2, y_1 \vee y_2) = (1,1)\\ (x_1 \vee x_2, y_1 \vee y_2) & \text{otherwise.}\end{cases}\end{aligned} \tag{4.7}$$

For $(\boldsymbol{x}_1, \boldsymbol{y}_1) \in \mathbf{S}^n$ and $(\boldsymbol{x}_2, \boldsymbol{y}_2) \in \mathbf{S}^n$, these operators extends elementwise. Note that the resulting points still belong to $\mathbf{S}^n$. Further, for a scalar a, its positive and negative parts will be denoted by a^+ and a^-, where $a^+ = \max(a, 0)$ and $a^- = -\min(a, 0)$, respectively and hence $a = a^+ - a^-$. The conventions $a^+_{ij\bullet} = \sum_k a^+_{ijk}$ and $|a|_{ij\bullet\bullet} = \sum_{k<l} |a_{ijkl}|$ are also used for ease of notation.

Relationship to bisubmodular functions. For any point $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbf{B}^{2n}$, it follows from the submodularity and symmetry of g that

$$\begin{aligned} g(\boldsymbol{x}, \boldsymbol{y}) &= \frac{1}{2}\left(g(\boldsymbol{x}, \boldsymbol{y}) + g(\bar{\boldsymbol{y}}, \bar{\boldsymbol{x}})\right)\\ &\geq \frac{1}{2}\left(g(\boldsymbol{x} \wedge \bar{\boldsymbol{y}}, \boldsymbol{y} \wedge \bar{\boldsymbol{x}}) + g(\boldsymbol{x} \vee \bar{\boldsymbol{y}}, \boldsymbol{y} \vee \bar{\boldsymbol{x}})\right)\\ &= g(\boldsymbol{x} \wedge \bar{\boldsymbol{y}}, \boldsymbol{y} \wedge \bar{\boldsymbol{x}}),\end{aligned} \tag{4.8}$$

where $(\boldsymbol{x} \wedge \bar{\boldsymbol{y}}, \boldsymbol{y} \wedge \bar{\boldsymbol{x}}) \in \mathbf{S}^n$. So, when analyzing $\min_{(\boldsymbol{x},\boldsymbol{y})} g(\boldsymbol{x}, \boldsymbol{y})$, considering the points in $\mathbf{S}^n$ is enough. Also, for any two points $(\boldsymbol{x}_1, \boldsymbol{y}_1) \in \mathbf{S}^n$ and $(\boldsymbol{x}_2, \boldsymbol{y}_2) \in \mathbf{S}^n$,

$$\begin{aligned} g(\boldsymbol{x}_1, \boldsymbol{y}_1) + g(\boldsymbol{x}_2, \boldsymbol{y}_2) &\geq g(\boldsymbol{x}_1 \wedge \boldsymbol{x}_2, \boldsymbol{y}_1 \wedge \boldsymbol{y}_2) + g(\boldsymbol{x}_1 \vee \boldsymbol{x}_2, \boldsymbol{y}_1 \vee \boldsymbol{y}_2)\\ &\geq g(\boldsymbol{x}_1 \wedge \boldsymbol{x}_2, \boldsymbol{y}_1 \wedge \boldsymbol{y}_2) + g((\boldsymbol{x}_1 \vee \boldsymbol{x}_2) \wedge (\overline{\boldsymbol{y}_1 \vee \boldsymbol{y}_2}), (\boldsymbol{y}_1 \vee \boldsymbol{y}_2) \wedge (\overline{\boldsymbol{x}_1 \vee \boldsymbol{x}_2}))\\ &= g((\boldsymbol{x}_1, \boldsymbol{y}_1) \sqcap (\boldsymbol{x}_2, \boldsymbol{y}_2)) + g((\boldsymbol{x}_1, \boldsymbol{y}_1) \sqcup (\boldsymbol{x}_2, \boldsymbol{y}_2)).\end{aligned} \tag{4.9}$$

The first inequality is submodularity and the second is obtained by applying (4.8). The last equality can be easily checked. By definition, a function satisfying

$$g(\boldsymbol{x}_1, \boldsymbol{y}_1) + g(\boldsymbol{x}_2, \boldsymbol{y}_2) \geq g((\boldsymbol{x}_1, \boldsymbol{y}_1) \sqcap (\boldsymbol{x}_2, \boldsymbol{y}_2)) + g((\boldsymbol{x}_1, \boldsymbol{y}_1) \sqcup (\boldsymbol{x}_2, \boldsymbol{y}_2)) \quad (4.10)$$

bisubmodular

is called bisubmodular (Fujishige and Iwata, 2006) and hence, the restriction $g\colon \mathbf{S}^n \to \mathbf{R}$ is indeed a bisubmodular function. The class of bisubmodular functions $\mathbf{S}^n \to \mathbf{R}$ is strictly larger than the class of submodular functions defined on the same domain.

4.1.2 Persistency

The notion of persistency in this section is no different from section 3.4 on page 25, but is presented differently. I will start by defining the "overwrite" operator.

DEFINITION 4.1 *For any point $\boldsymbol{x} \in \mathbf{B}^n$ and $(\boldsymbol{x}^*, \boldsymbol{y}^*) \in \mathbf{S}^n$, the operator $\mathbf{B}^n \times \mathbf{S}^n \to \mathbf{B}^n$ denoted $\boldsymbol{x} \leftarrow (\boldsymbol{x}^*, \boldsymbol{y}^*)$ is defined by*

$$\boldsymbol{x} \leftarrow (\boldsymbol{x}^*, \boldsymbol{y}^*) = \boldsymbol{u} \quad \textit{where} \quad (\boldsymbol{u}, \bar{\boldsymbol{u}}) = ((\boldsymbol{x}, \bar{\boldsymbol{x}}) \sqcup (\boldsymbol{x}^*, \boldsymbol{y}^*)) \sqcup (\boldsymbol{x}^*, \boldsymbol{y}^*).$$

One can check that this is well-defined and that

$$u_i = \begin{cases} x_i^*, & \text{if } (x_i^*, y_i^*) \neq (0, 0) \\ x_i, & \text{otherwise} \end{cases} \qquad \text{for } i = 1, \ldots, n. \quad (4.11)$$

So, $\boldsymbol{x} \leftarrow (\boldsymbol{x}^*, \boldsymbol{y}^*)$ can be thought of as the result of replacing elements of $\boldsymbol{x}$ by elements of $\boldsymbol{x}^*$ provided the corresponding element pairs in $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ are non-zero.

LEMMA 4.2 (Autarky; Kolmogorov, 2010) *Let g be a function satisfying (A)–(C) and $(\boldsymbol{x}^*, \boldsymbol{y}^*) \in \operatorname{argmin} g$. Then $f(\boldsymbol{x} \leftarrow (\boldsymbol{x}^*, \boldsymbol{y}^*)) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x}$.*

Proof. From bisubmodularity, it follows that, for any $\boldsymbol{v} \in \mathbf{B}^n$,

$$\begin{aligned} g((\boldsymbol{v}, \bar{\boldsymbol{v}}) \sqcup (\boldsymbol{x}^*, \boldsymbol{y}^*)) &\leq g(\boldsymbol{v}, \bar{\boldsymbol{v}}) + \Big(g(\boldsymbol{x}^*, \boldsymbol{y}^*) - g((\boldsymbol{v}, \bar{\boldsymbol{v}}) \sqcap (\boldsymbol{x}^*, \boldsymbol{y}^*))\Big) \\ &\leq g(\boldsymbol{v}, \bar{\boldsymbol{v}}), \end{aligned} \quad (4.12)$$

which implies that, for any $\boldsymbol{x} \in \mathbf{B}^n$,

$$\begin{aligned} f(\boldsymbol{x}) &= g(\boldsymbol{x}, \bar{\boldsymbol{x}}) \\ &\geq g((\boldsymbol{x}, \bar{\boldsymbol{x}}) \sqcup (\boldsymbol{x}^*, \boldsymbol{y}^*)) \\ &\geq g(((\boldsymbol{x}, \bar{\boldsymbol{x}}) \sqcup (\boldsymbol{x}^*, \boldsymbol{y}^*)) \sqcup (\boldsymbol{x}^*, \boldsymbol{y}^*)) \\ &= f(\boldsymbol{x} \leftarrow (\boldsymbol{x}^*, \boldsymbol{y}^*)). \end{aligned}$$ □

Autarky is needed to ensure that the objective function does not increase when generalized roof duality is used in a fusion framework (see section 4.7.2). An arguably more important consequence is persistency: if $\boldsymbol{x} \in \operatorname{argmin}(f)$, then $\boldsymbol{x} \leftarrow (\boldsymbol{x}^*, \boldsymbol{y}^*) \in \operatorname{argmin}(f)$. Hence, autarky also implies the following special case:

LEMMA 4.3 (Persistency) *Let g satisfy (A)–(C) and $(\boldsymbol{x}^*, \boldsymbol{y}^*) \in \operatorname{argmin} g$. If $\boldsymbol{x} \in \operatorname{argmin}(f)$, then $\boldsymbol{x} \leftarrow (\boldsymbol{x}^*, \boldsymbol{y}^*) \in \operatorname{argmin}(f)$.*

In other words, all elements (x_i^*, y_i^*) not equal to $(0, 0)$ of a minimizer of g give us the corresponding elements x_i^* of a minimizer of f. Compare this statement to Theorem 3.4 on page 26.

REMARK 4.4 Lemmas 4.2 and 4.3 hold for *any* feasible relaxation g, not just the optimal one. This fact will be used later on.

4.2 Standard Roof Duality

I will start by analyzing quadratic submodular relaxations g of a quadratic pseudo-boolean function f. As we shall see, this is no restriction—for a quadratic f, there are optimal relaxations of degree two.

A symmetric polynomial $g\colon \mathbf{B}^{2n} \to \mathbf{R}$ with $\operatorname{degree}(g) = 2$ can be represented by

$$\begin{aligned} g(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2} \sum_i b_i(x_i + \bar{y}_i) + \sum_i b_{ii} x_i \bar{y}_i \\ + \frac{1}{2} \sum_{i<j} \Big(b_{ij}(x_i x_j + \bar{y}_i \bar{y}_j) + c_{ij}(x_i \bar{y}_j + \bar{y}_i x_j) \Big). \end{aligned} \tag{4.13}$$

The above expression contains all monomials of degree two or less. The symmetry requirement forces some of them to have the same coefficients.

LEMMA 4.5 *If the quadratic pseudo-boolean function f is represented by a multilinear polynomial (4.1) and the symmetric function g by (4.13), then the constraint $g(\boldsymbol{x}, \bar{\boldsymbol{x}}) = f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbf{B}^n$ implies that*

$$b_i + b_{ii} = a_i \text{ for } 1 \leq i \leq n \quad \text{and} \quad b_{ij} + c_{ij} = a_{ij} \text{ for } 1 \leq i < j \leq n.$$

Proof. For a given pseudo-boolean function, the multilinear polynomial representation (4.1) is unique (Boros and Hammer, 2002). Evaluate $g(\boldsymbol{x}, \bar{\boldsymbol{x}})$ and set the corresponding coefficients equal in the multilinear representations of $g(\boldsymbol{x}, \bar{\boldsymbol{x}})$ and $f(\boldsymbol{x})$. □

Recall the necessary and sufficient requirement for submodularity (3.4) on page 21. For a submodular, symmetric polynomial g in the form (4.13), this is equivalent to

$$b_{ii} \geq 0, \quad b_{ij} \leq 0 \quad \text{and} \quad c_{ij} \geq 0. \tag{4.14}$$

It follows that $b_{ij} = a_{ij} - c_{ij} \leq a_{ij}$ and therefore $b_{ij} \leq \min(a_{ij}, 0) = -a_{ij}^-$ and $c_{ij} \geq \max(a_{ij}, 0) = a_{ij}^+$.

The roof dual construction given by Boros and Hammer (2002) proposes to set $b_{ij} = -a_{ij}^-$ and $c_{ij} = a_{ij}^+$ so it is in fact a quadratic submodular relaxation. A stronger statement can be proven, namely that this relaxation g dominates any other bisubmodular relaxation $\tilde{g}$ of arbitrary degree, that is, $g(\boldsymbol{x}, \boldsymbol{y}) \geq \tilde{g}(\boldsymbol{x}, \boldsymbol{y})$ for all $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbf{S}^n$.

THEOREM 4.6 (Kolmogorov, 2012) *An optimal submodular relaxation g of a quadratic pseudo-boolean function f is obtained through roof duality:*

1. *Set $b_i = a_i$ and $b_{ii} = 0$ for $1 \leq i \leq n$ in (4.13),*
2. *Set $b_{ij} = -a_{ij}^-$ and $c_{ij} = a_{ij}^+$ for $1 \leq i < j \leq n$ in (4.13).*

Further, the relaxation g is optimal among all possible bisubmodular relaxations.

Let $\boldsymbol{e}_i$ be a vector with zeros everywhere except at position i. The proof of Theorem 4.6 requires the following lemma:

LEMMA 4.7 *Let $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbf{S}^n$ and $x_i = y_i = 0$. Then $\tilde{g}(\boldsymbol{x}, \boldsymbol{y}) - \tilde{g}(\boldsymbol{x} \vee \boldsymbol{e}_i, \boldsymbol{y}) \leq g(\boldsymbol{x}, \boldsymbol{y}) - g(\boldsymbol{x} \vee \boldsymbol{e}_i, \boldsymbol{y})$.*

Proof. The proof is by induction over $N = |\{k \mid x_k = y_k = 0\}|$, the number of $(0,0)$-elements in $(\boldsymbol{x}, \boldsymbol{y})$.

To establish the base case $N = 1$, note that $\tilde{g}(\boldsymbol{x} \vee \boldsymbol{e}_i, \boldsymbol{y}) = g(\boldsymbol{x} \vee \boldsymbol{e}_i, \boldsymbol{y})$ because of requirement (A). For the other term,

$$\begin{aligned} 2\tilde{g}(\boldsymbol{x}, \boldsymbol{y}) \leq & \quad \text{(bisubmodularity)} \\ \tilde{g}(\boldsymbol{x} \vee \boldsymbol{e}_i, \boldsymbol{y}) + \tilde{g}(\boldsymbol{x}, \boldsymbol{y} \vee \boldsymbol{e}_i) = & \quad \text{(requirement (A))} \\ g(\boldsymbol{x} \vee \boldsymbol{e}_i, \boldsymbol{y}) + g(\boldsymbol{x}, \boldsymbol{y} \vee \boldsymbol{e}_i) = & \quad (b_{ii} = 0) \\ g(\boldsymbol{x}, \boldsymbol{y}) + g(\boldsymbol{x} \vee \boldsymbol{e}_i, \boldsymbol{y} \vee \boldsymbol{e}_i) = & \quad \text{(by construction of } g). \\ 2g(\boldsymbol{x}, \boldsymbol{y}). & \end{aligned}$$

For the inductive step, pick $j \neq i$ such that $x_j = y_j = 0$. There are two cases: $a_{ij} \geq 0$ and $a_{ij} < 0$.

$\boxed{a_{ij} \geq 0}$ The function g then contains the term $\frac{1}{2}a_{ij}(x_i\bar{y}_j + \bar{y}_i x_j)$ and $b_{ij} = 0$. A calculation gives

$$\begin{aligned} \tilde{g}(\boldsymbol{x}, \boldsymbol{y}) - \tilde{g}(\boldsymbol{x} \vee \boldsymbol{e}_i, \boldsymbol{y}) \leq & \quad \text{(bisubmodularity)} \\ \tilde{g}(\boldsymbol{x} \vee \boldsymbol{e}_j, \boldsymbol{y}) - \tilde{g}(\boldsymbol{x} \vee \boldsymbol{e}_i \vee \boldsymbol{e}_j, \boldsymbol{y}) \leq & \quad \text{(induction hypothesis)} \\ g(\boldsymbol{x} \vee \boldsymbol{e}_j, \boldsymbol{y}) - g(\boldsymbol{x} \vee \boldsymbol{e}_i \vee \boldsymbol{e}_j, \boldsymbol{y}) = & \quad \text{(by construction of } g) \\ -\tfrac{1}{2}a_i - \tfrac{1}{2}a_{ij} = & \quad \text{〃} \\ g(\boldsymbol{x}, \boldsymbol{y}) - g(\boldsymbol{x} \vee \boldsymbol{e}_i, \boldsymbol{y}), & \end{aligned}$$

which is what we want to prove.

$\boxed{a_{ij} < 0}$ This case is similar to the previous one—g now contains $\frac{1}{2}a_{ij}(x_i x_j + \bar{y}_i\bar{y}_j)$:

$$\begin{aligned} \tilde{g}(\boldsymbol{x}, \boldsymbol{y}) - \tilde{g}(\boldsymbol{x} \vee \boldsymbol{e}_i, \boldsymbol{y}) \leq & \quad \text{(bisubmodularity)} \\ \tilde{g}(\boldsymbol{x}, \boldsymbol{y} \vee \boldsymbol{e}_j) - \tilde{g}(\boldsymbol{x} \vee \boldsymbol{e}_i, \boldsymbol{y} \vee \boldsymbol{e}_j) \leq & \quad \text{(induction hypothesis)} \\ g(\boldsymbol{x}, \boldsymbol{y} \vee \boldsymbol{e}_j) - g(\boldsymbol{x} \vee \boldsymbol{e}_i, \boldsymbol{y} \vee \boldsymbol{e}_j) = & \quad \text{(by construction of } g) \\ -\tfrac{1}{2}a_i = & \quad \text{〃} \\ g(\boldsymbol{x}, \boldsymbol{y}) - g(\boldsymbol{x} \vee \boldsymbol{e}_i, \boldsymbol{y}). & \end{aligned}$$

□

Proof of Theorem 4.6. The proof is by induction over n.

For $n = 1$, we have $f(x_1) = g(x_1, \bar{x}_1) = \tilde{g}(x_1, \bar{x}_1)$. If $f(x_1) = a_1 x_1$ then $g(x_1, y_1) = \frac{1}{2}a_1(x_1 + \bar{y}_1)$ and

$$g(0,0) = \frac{1}{2}a_1 = \frac{1}{2}\left(\tilde{g}(0,1) + \tilde{g}(1,0)\right) \geq \tilde{g}(0,0),$$

which follows from symmetry and bisubmodularity of $\tilde{g}$.

For $n > 1$, assume that the statement holds for $n - 1$ variables. Then, for any $i = 1, \ldots, n$, note that $g(\boldsymbol{x} \wedge \bar{\boldsymbol{e}}_i, \boldsymbol{y} \vee \boldsymbol{e}_i)$ is an optimal relaxation of $f(\boldsymbol{x} \wedge \bar{\boldsymbol{e}}_i)$ and hence

$$g(\boldsymbol{x} \wedge \bar{\boldsymbol{e}}_i, \boldsymbol{y} \vee \boldsymbol{e}_i) \geq \tilde{g}(\boldsymbol{x} \wedge \bar{\boldsymbol{e}}_i, \boldsymbol{y} \vee \boldsymbol{e}_i) \text{ for all } (\boldsymbol{x}, \boldsymbol{y}) \in \mathbf{S}^n.$$

In a similar manner,

$$g(\boldsymbol{x} \vee \boldsymbol{e}_i, \boldsymbol{y} \wedge \bar{\boldsymbol{e}}_i) \geq \tilde{g}(\boldsymbol{x} \vee \boldsymbol{e}_i, \boldsymbol{y} \wedge \bar{\boldsymbol{e}}_i) \text{ for all } (\boldsymbol{x}, \boldsymbol{y}) \in \mathbf{S}^n.$$

The only point not checked in $\mathbf{S}^n$ is $(\mathbf{0}, \mathbf{0})$. Lemma 4.7 establishes that $g(\mathbf{0}, \mathbf{0}) - g(\mathbf{0}, \boldsymbol{e}_1) \geq \tilde{g}(\mathbf{0}, \mathbf{0}) - \tilde{g}(\mathbf{0}, \boldsymbol{e}_1)$ and from this it follows that

$$g(\mathbf{0}, \mathbf{0}) \geq \tilde{g}(\mathbf{0}, \mathbf{0}) + \underbrace{(g(\mathbf{0}, \boldsymbol{e}_1) - \tilde{g}(\mathbf{0}, \boldsymbol{e}_1))}_{\geq 0} \geq \tilde{g}(\mathbf{0}, \mathbf{0}). \quad \square \qquad (4.15)$$

The roof dual bound is also known to be the tightest bound for several different linear programming relaxations (Hammer et al., 1984).

4.3 Generalized Roof Duality

For a pseudo-boolean function f in n variables with degree$(f) > 2$, directly solving (4.4) is not tractable since the required number of constraints is exponential in n. Two obvious heuristic alternatives are:

1. Decompose f into a sum $f(\boldsymbol{x}) = \sum_{i<j<k\ldots} f_{ijk\ldots}(x_i, x_j, x_k, \ldots)$ and compute an optimal relaxation for each term $f_{ijk\ldots}$. However, the sum of optimal relaxations is generally not optimal.

2. Use a subset of the points in $\mathbf{S}^n$ for $g(\boldsymbol{x}, \boldsymbol{y}) \geq \ell$ to get an approximate optimal relaxation.

Neither of these approaches are satisfactory. One may even wonder: is the optimal relaxation g polynomial time computable at all?

One important issue is to make sure that the set of submodular relaxations can be expressed in an easy manner, and in the end, be minimized by max-flow/min-cut. For this purpose, the notation of expressibility by Živný et al. (2009) comes in handy.

DEFINITION 4.8 *A function* $h\colon \mathbf{B}^n \to \mathbf{R}$ *is called* expressible *if it can be expressed as* $h(\boldsymbol{x}) = \min_{\boldsymbol{x}' \in \mathbf{B}^k} h'(\boldsymbol{x}, \boldsymbol{x}')$ *for some* k*, where* $h'(\boldsymbol{x}, \boldsymbol{x}')$ *is a quadratic submodular function. The variables in* $\boldsymbol{x}'$ *are called* auxiliary variables.

An expressible function is always submodular. From the definition of submodularity, it follows that a submodular function should satisfy exponentially many inequality constraints, but this is intractable. On the other hand, we have seen that polynomially many constraints are enough for quadratic submodular functions (page 21).

DEFINITION 4.9 *Consider a set of pseudo-boolean functions (up to a fixed degree) in* n *variables parametrized by a coefficient vector* $\boldsymbol{a} \in \mathbf{R}^d$*. A subset of submodular functions is called* recognizable *if the submodularity condition can be expressed by polynomially many linear inequality constraints in* $\boldsymbol{a}$ *with respect to* n*.*

All cubic submodular functions are expressible, and the set of cubic submodular functions is recognizable among the set of cubic functions. Unfortunately, all quartic submodular functions are not expressible (Živný et al., 2009) and whether the subset of expressible functions is recognizable is an open problem. This makes the quartic and higher-order generalization of the roof dual much harder to handle than the quadratic and cubic cases. Section 4.5 will define recognizable sets of quartic expressible functions in two different ways, and investigate their properties.

Requirement (B) is from now on replaced by the following extended condition:

$$g \in \mathcal{G}, \text{ where } \mathcal{G} \text{ is a recognizable set of expressible functions.} \qquad \text{(B')}$$

The precise definition of roof duality is then as follows.

DEFINITION 4.10 (Generalized Roof Duality) *The* generalized roof duality *bound over a recognizable set of expressible functions is the optimal value of* (4.4) *with constraints (A), (B') and (C). The optimal value will be denoted by* g^*_{GRD}*.*

Note that computing the optimal value g^*_{GRD} directly via (4.4) still involves exponentially many constraints due to the requirements that $g(\boldsymbol{x}, \boldsymbol{y}) \geq \ell$ for all $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbf{S}^n$.

4.3.1 Main Result

Lemma 4.3 states that persistency holds for any bisubmodular relaxation g—optimal or not. From the example in (3.23) on page 28, it is clear, however, that not all relaxations are equally powerful. Instead of solving (4.4), which, although possible, may require a large number of constraints, one can consider a simpler problem:

$$\begin{aligned} \max_{g} \quad & g(\mathbf{0}, \mathbf{0}) \\ \text{subject to} \quad & g \text{ satisfies (A), (B') and (C).} \end{aligned} \tag{4.16}$$

Instead of maximizing $\min g(\boldsymbol{x}, \boldsymbol{y})$, this problem only maximizes $g(\mathbf{0}, \mathbf{0})$. This problem is considerably less arduous and can be solved in polynomial time.[1] Consider the minimizer $(\boldsymbol{x}^*, \boldsymbol{y}^*) \in \arg\min(g)$, which is also polynomial time computable since g is submodular:

- If $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is non-zero, persistency will reduce the number of variables in f and make the problem smaller.
- Otherwise, as the optimum is indeed the trivial solution, and as $g(\mathbf{0}, \mathbf{0})$ is maximized in the construction of g, it must be an optimal relaxation and the generalized roof duality bound has been obtained.

These observations lead to the following algorithm that computes the generalized roof duality bound:

1. Construct g by solving (4.16).
2. Compute $(\boldsymbol{x}^*, \boldsymbol{y}^*) \in \arg\min(g)$.
3. If $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is non-zero, use persistency to simplify f and start over from 1. Otherwise, stop.

THEOREM 4.11 *A lower bound on* $\min f(\boldsymbol{x})$ *which is greater than or equal to the generalized roof duality bound* g^*_{GRD} *over a recognizable set of expressible pseudo-boolean functions can be computed in polynomial time.*

[1] The condition $g(\boldsymbol{x}, \bar{\boldsymbol{x}}) = f(\boldsymbol{x})$ for all $\boldsymbol{x}$ does involve exponentially many constraints, but as g is required to be of fixed degree (independent of n), then its polynomial representation has only polynomially many terms.

Proof. If $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is non-zero, persistency can be used to simplify the original function f. This is equivalent to adding constraints of the type $x_i = \bar{y}_i = c$ to $\max_g \min_{\boldsymbol{x},\boldsymbol{y}} g(\boldsymbol{x}, \boldsymbol{y})$. This can only increase the computed value. If on the other hand $(\boldsymbol{x}^*, \boldsymbol{y}^*) = (\boldsymbol{0}, \boldsymbol{0})$, then the best possible lower bound is obtained by construction of g. Therefore, the final bound is at least equal to the optimal value g^*_{GRD} of (4.4).

The algorithm can obviously not run for more than n iterations, since in each iteration either persistencies are found and f is simplified or the algorithm terminates. With all steps being solvable in polynomial time, the algorithm itself is polynomial. □

For the cubic case $m = 3$, the theorem can be simplified.

COROLLARY 4.12 *A lower bound on* $\min f(\boldsymbol{x})$ *which is greater than or equal to the generalized roof duality bound* g^*_{GRD} *over cubic submodular functions can be computed in polynomial time.*

REMARK 4.13 Note that an optimal relaxation g is not explicitly constructed by the above procedure, but rather a sequence of relaxations g_1, $g_2, \ldots, g_k$, such that the final relaxation g_k fulfills $\min_{\boldsymbol{x},\boldsymbol{y}} g_k(\boldsymbol{x}, \boldsymbol{y}) \geq g^*_{\text{GRD}}$.

REMARK 4.14 As suggested by the proof of Theorem 4.11, the iterative approach can obtain a better bound than the "optimal" value g^*_{GRD} in Definition 4.10. This can be observed in practice for small problems where directly solving (4.4) is feasible. One example is

$$f(\boldsymbol{x}) = 14x_1 + 15x_2 - 6x_3 + 9x_4 - 5x_1x_2 + 6x_1x_3 + 3x_1x_4 + 13x_2x_3 + 13x_2x_4 - 6x_3x_4 + 20x_1x_2x_3 + 9x_1x_2x_4 + 17x_1x_3x_4 + 2x_2x_3x_4, \tag{4.17}$$

for which (4.4) gives the lower bound $g^*_{\text{GRD}} = -8$ and the iterative method above gives $-6 > g^*_{\text{GRD}}$, which is tight.

4.4 Cubic Relaxations

This section will analyze the properties of relaxations of degree three with respect to symmetry and submodularity in detail.

A cubic symmetric polynomial $g\colon \mathbf{B}^{2n} \to \mathbf{R}$ can be written as

$$\begin{aligned} g(\boldsymbol{x},\boldsymbol{y}) = L + Q + \frac{1}{2}\sum_{i<j}\Big(b_{iij}x_i\bar{y}_i(x_j+\bar{y}_j) + b_{ijj}(x_i+\bar{y}_i)x_j\bar{y}_j\Big) \\ + \frac{1}{2}\sum_{i<j<k}\Big(b_{ijk}(x_ix_jx_k+\bar{y}_i\bar{y}_j\bar{y}_k) + c_{ijk}(x_ix_j\bar{y}_k+\bar{y}_i\bar{y}_jx_k) \\ + d_{ijk}(x_i\bar{y}_jx_k+\bar{y}_ix_j\bar{y}_k) + e_{ijk}(\bar{y}_ix_jx_k+x_i\bar{y}_j\bar{y}_k)\Big), \end{aligned} \tag{4.18}$$

where L and Q denote linear and quadratic terms as in section 4.2. The objective function in (4.16) is then simply equal to

$$g(\mathbf{0},\mathbf{0}) = \frac{1}{2}\Big(\sum_i b_i + \sum_{i<j} b_{ij} + \sum_{i<j<k} b_{ijk}\Big). \tag{4.19}$$

LEMMA 4.15 *If the cubic pseudo-boolean function f is represented by a multilinear polynomial (4.1) and the symmetric function g by (4.18), then the constraint $g(\boldsymbol{x},\bar{\boldsymbol{x}}) = f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbf{B}^n$ implies that*

$$\begin{aligned} b_i + b_{ii} &= a_i \text{ for } 1 \le i \le n \\ b_{ij} + c_{ij} + b_{iij} + b_{ijj} &= a_{ij} \text{ for } 1 \le i < j \le n \\ b_{ijk} + c_{ijk} + d_{ijk} + e_{ijk} &= a_{ijk} \text{ for } 1 \le i < j < k \le n. \end{aligned} \tag{4.20}$$

Proof. See the proof of Lemma 4.5. □

The necessary and sufficient conditions for a cubic polynomial to be submodular were given by Billionet and Minoux (1985). The characterization is slightly more complicated than the quadratic case: A cubic multilinear polynomial f in (4.1) is submodular if and only if, for every $i < j$,

$$a_{ij} + a^+_{ij\bullet} + a^+_{i\bullet j} + a^+_{\bullet ij} \le 0. \tag{4.21}$$

The set of symmetric, submodular functions of degree 3 will be denoted by $\Gamma_{\text{sym},3}$. It is possible to derive corresponding inequality constraints for this class:

LEMMA 4.16 *A cubic symmetric polynomial g represented by (4.18) is submodular if and only if*

$$b_{ij} + b^+_{iij} + b^+_{ijj} \le -b^+_{ij\bullet} - b^+_{i\bullet j} - b^+_{\bullet ij} - c^+_{ij\bullet} - d^+_{i\bullet j} - e^+_{\bullet ij} \quad (4.22a)$$

$$-c_{ij} + b^-_{iij} + b^-_{ijj} \le -c^-_{i\bullet j} - c^-_{\bullet ij} - d^-_{ij\bullet} - d^-_{\bullet ij} - e^-_{ij\bullet} - e^-_{i\bullet j} \quad (4.22b)$$

$$b_{ii} \ge b^-_{ii\bullet} + b^-_{\bullet ii}, \quad (4.22c)$$

for $1 \le i < j \le n$.

Proof. Both (4.21) and this lemma can be proved using the second derivative requirement (3.4) on page 21. For the variables x_I and x_J this requirement is

$$\begin{aligned} b_{IJ} &+ \sum_{J<k} b_{IJk}x_k + \sum_{I<k<J} b_{IkJ}x_k + \sum_{k<I} b_{kIJ}x_k \\ &+ \sum_{J<k} c_{IJk}\bar{y}_k + \sum_{I<k<J} d_{IkJ}\bar{y}_k + \sum_{k<I} e_{kIJ}\bar{y}_k \\ &+ b_{IIJ}\bar{y}_I + b_{IJJ}\bar{y}_J \le 0, \end{aligned} \quad (4.23)$$

where the sums are taken over all valid indices. Since this expression has to hold for all $(\boldsymbol{x}, \boldsymbol{y})$, only the worst case is interesting, where the left hand side is maximized:

$$b_{IJ} + b^+_{IJ\bullet} + b^+_{I\bullet J} + b^+_{\bullet IJ} + c^+_{IJ\bullet} + d^+_{I\bullet J} + e^+_{\bullet IJ} + b^+_{IIJ} + b^+_{IJJ} \le 0. \quad (4.24)$$

This gives (4.22a). To obtain (4.22b), one can observe that $\frac{\partial^2 g}{\partial u \partial v}(\boldsymbol{x}, \boldsymbol{y}) = -\frac{\partial^2 g}{\partial u \partial \bar{v}}(\boldsymbol{x}, \boldsymbol{y})$, repeat the procedure for the variables x_I and y_J and use the requirement that $b_{IJ} + c_{IJ} + b_{IIJ} + b_{IJJ} = a_{IJ}$. □

LEMMA 4.17 *There is a solution g represented by (4.18) to the optimization problem (4.16) such that*

(i) $b_{ii} = 0$ *for* $1 \le i \le n$, $b_{iij} = b_{ijj} = 0$ *for* $1 \le i < j \le n$,

(ii) if $a_{ijk} \ge 0$ *then* $b^-_{ijk} = c^-_{ijk} = d^-_{ijk} = e^-_{ijk} = 0$ *for* $1 \le i < j < k \le n$,

(iii) if $a_{ijk} \le 0$ *then* $b^+_{ijk} = c^+_{ijk} = d^+_{ijk} = e^+_{ijk} = 0$ *for* $1 \le i < j < k \le n$.

Proof. (i) Suppose that $b_{iij} > 0$ for the optimal g. Then (4.20) and (4.22a) show that setting b_{iij} to 0 and increasing b_{ij} by the same amount will still be feasible. This operation increases the objective function $g(\mathbf{0}, \mathbf{0})$. If, on the other hand, $b_{iij} < 0$, then b_{iij} can also be set to 0 and decreasing c_{ij} by the same amount will give a feasible solution; see (4.20) and (4.22b), with no change to the objective function. The same argument holds for b_{ijj}. Finally, setting $b_{ii} = 0$ and increasing b_i by the same amount will always be feasible and increase the objective function.

(ii) Suppose $b^-_{ijk} > 0$. Then setting $b^-_{ijk} = 0$ and decreasing c^+_{ijk}, d^+_{ijk} and e^+_{ijk} such that the sum $c^+_{ijk} + d^+_{ijk} + e^+_{ijk}$ is decreased by the same amount will still be a feasible solution with higher objective function value. Other variables are similarly handled.

(iii) The proof is analogous to (ii). □

The above lemma simplifies matters. If, say $a_{ijk} > 0$, then we can set $b_{ijk} = b^+_{ijk}$, $c_{ijk} = c^+_{ijk}$, $d_{ijk} = d^+_{ijk}$ and $e_{ijk} = e^+_{ijk}$. Further, the submodularity conditions (4.22a) and (4.22b) become linear inequality constraints in the unknowns, condition (4.22c) becomes obsolete and the optimization problem (4.16) is turned into an instance of linear programming.

EXAMPLE 4.18 Consider again the example of a cubic pseudo-boolean function f in (3.23) on page 28. Finding a $g(\boldsymbol{x}, \boldsymbol{y})$ of the form (4.18) by solving (4.16) results in

$$g(\boldsymbol{x}, \boldsymbol{y}) = -(x_1 + \bar{y}_1) + \frac{1}{2}(x_2 + \bar{y}_2) - \frac{1}{2}(x_3 + \bar{y}_3) + 2(x_1\bar{y}_2 + \bar{y}_1 x_2) + 2(x_1\bar{y}_3 + \bar{y}_1 x_3) - (x_2 x_3 + \bar{y}_2\bar{y}_3) - (\bar{y}_1 x_2 x_3 + x_1\bar{y}_2\bar{y}_3). \quad (4.25)$$

Minimizing this submodular relaxation gives $g_{\min} = -2$ for $\boldsymbol{x}^* = (0, 1, 1)$ and $\boldsymbol{y}^* = (1, 0, 0)$. Since $\boldsymbol{x}^* = \bar{\boldsymbol{y}}^*$, it follows that $\boldsymbol{x}^*$ is the global minimizer for f as well.

4.5 Quartic Relaxations

Determining whether a given quartic polynomial is submodular or not is known to be co-NP-complete (Theorem 3.3 on page 22), and not all submodular quartic polynomials are expressible by quadratic submodular

functions (Živný et al., 2009). Therefore, a compromise is required. This section defines and analyzes two different proposals of recognizable sets for quartic functions.

4.5.1 Approach I: The Set $\Gamma_{\text{sym},4}$

One possible approach is to work with the quartic polynomials in (4.1) that satisfy, for every $i < j$,

$$a_{ij} + a^+_{ij\bullet} + a^+_{i\bullet j} + a^+_{\bullet ij} + a^+_{ij\bullet\bullet} + a^+_{i\bullet j\bullet} + \ldots + a^+_{\bullet\bullet ij} \leq 0. \quad (4.26)$$

This choice can be seen as a natural generalization of the cubic case; see (4.21). The set has a number of advantageous properties. First, it is a rich set of submodular functions; Živný and Jeavons (2010) denote the set by $\Gamma_{\text{suff},4}$ and analyze it in more detail. The set of cubic submodular functions is a subset of these functions. Second, each quartic term only needs one auxiliary variable for expressibility. Finally, only $O(n^2)$ inequalities are sufficient to make sure the relaxation is submodular. Thus, the set is recognizable among all quartic pseudo-boolean functions.

Similar to previous derivations, a large class of quartic symmetric polynomials can be written

$$\begin{aligned} g(\boldsymbol{x},\boldsymbol{y}) = L + Q + C + \frac{1}{2}\sum_{i<j<k<l} \Big(\\ b_{ijkl}(x_ix_jx_kx_l + \bar{y}_i\bar{y}_j\bar{y}_k\bar{y}_l) + c_{ijkl}(x_ix_jx_k\bar{y}_l + \bar{y}_i\bar{y}_j\bar{y}_kx_l) + \\ d_{ijkl}(x_ix_j\bar{y}_kx_l + \bar{y}_i\bar{y}_jx_k\bar{y}_l) + e_{ijkl}(x_i\bar{y}_jx_kx_l + \bar{y}_ix_j\bar{y}_k\bar{y}_l) + \\ p_{ijkl}(\bar{y}_ix_jx_kx_l + x_i\bar{y}_j\bar{y}_k\bar{y}_l) + q_{ijkl}(x_ix_j\bar{y}_k\bar{y}_l + \bar{y}_i\bar{y}_jx_kx_l) + \\ r_{ijkl}(x_i\bar{y}_jx_k\bar{y}_l + \bar{y}_ix_j\bar{y}_kx_l) + s_{ijkl}(x_i\bar{y}_j\bar{y}_kx_l + \bar{y}_ix_jx_k\bar{y}_l) \Big), \end{aligned} \quad (4.27)$$

where L, Q and C denote lower-order terms, and $b_{ijkl}+c_{ijkl}+\ldots+s_{ijkl} = a_{ijkl}$. Just as for the cubic case, $b_{ii} = b_{iij} = b_{ijj} = 0$.

By first expanding all conjugate factors in the symmetric form (4.27) to a multilinear polynomial and then applying the sufficient condition for recognizability (4.26), one obtains the following constraints.[2]

[2] These constraints are preferably verified using e.g. Maple

LEMMA 4.19 *A quartic symmetric polynomial g represented by (4.27) is submodular and expressible by a quadratic submodular polynomial if*

$$\begin{aligned} b_{ij} \leq (4.22a) &- b^+_{ij\bullet\bullet} - b^+_{i\bullet j\bullet} - b^+_{i\bullet\bullet j} - b^+_{\bullet ij\bullet} - b^+_{\bullet i\bullet j} - b^+_{\bullet\bullet ij} \\ &- |c|_{ij\bullet\bullet} - |c|_{i\bullet j\bullet} - |c|_{\bullet ij\bullet} - |d|_{ij\bullet\bullet} - |d|_{i\bullet\bullet j} - |d|_{\bullet i\bullet j} \\ &- |e|_{i\bullet j\bullet} - |e|_{i\bullet\bullet j} - |e|_{\bullet\bullet ij} - |p|_{\bullet ij\bullet} - |p|_{\bullet i\bullet j} - |p|_{\bullet\bullet ij} \\ &- q^+_{ij\bullet\bullet} - |q|_{ij\bullet\bullet} - q^+_{\bullet\bullet ij} - r^+_{i\bullet j\bullet} - |r|_{i\bullet j\bullet} - r^+_{\bullet i\bullet j} \\ &- s^+_{i\bullet\bullet j} - |s|_{i\bullet\bullet j} - s^+_{\bullet ij\bullet} \end{aligned} \tag{4.28a}$$

and

$$\begin{aligned} -c_{ij} \leq (4.22b) &- c^-_{i\bullet\bullet j} - c^-_{\bullet i\bullet j} - c^-_{\bullet\bullet ij} - d^-_{i\bullet j\bullet} - d^-_{\bullet ij\bullet} - d^-_{\bullet\bullet ij} \\ &- e^-_{ij\bullet\bullet} - e^-_{\bullet ij\bullet} - e^-_{\bullet i\bullet j} - p^-_{ij\bullet\bullet} - p^-_{i\bullet j\bullet} - p^-_{i\bullet\bullet j} \\ &- |q|_{i\bullet j\bullet} - |q|_{i\bullet\bullet j} - |q|_{\bullet ij\bullet} - |q|_{\bullet i\bullet j} \\ &- |r|_{ij\bullet\bullet} - |r|_{i\bullet\bullet j} - |r|_{\bullet ij\bullet} - |r|_{\bullet\bullet ij} \\ &- |s|_{ij\bullet\bullet} - |s|_{i\bullet j\bullet} - |s|_{\bullet i\bullet j} - |s|_{\bullet\bullet ij}, \end{aligned} \tag{4.28b}$$

for $1 \leq i < j \leq n$, where (4.22a) and (4.22b) denote the right-hand sides of those inequalities, respectively.

This subset of symmetric functions is denoted by $\Gamma_{\text{sym},4}$, and, naturally, $\Gamma_{\text{sym},3} \subset \Gamma_{\text{sym},4}$.

EXAMPLE 4.20 Ishikawa (2011) proposed the following reduction identity:

$$\begin{aligned} x_1x_2x_3x_4 = &\min_{z\in\mathbf{B}} z(3 - 2x_1 - 2x_2 - 2x_3 - 2x_4) \\ &+ x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4. \end{aligned} \tag{4.29}$$

This can be used to express

$$f(\boldsymbol{x}) = x_1 + x_3 - x_4 + 2x_1x_4 + 2x_2x_3 - x_3x_4 + x_1x_2x_3x_4 \tag{4.30}$$

as a quadratic polynomial with one auxiliary variable z. The quadratic roof duality bound gives $f_{\min} \geq -2$ and does not find an assignment for any of

the four variables. On the other hand, solving the linear program (4.16), one obtains the relaxation

$$\begin{aligned} g(\boldsymbol{x},\boldsymbol{y}) = &\frac{1}{2}(x_1+\bar{y}_1)+\frac{1}{2}(x_3+\bar{y}_3)-\frac{1}{2}(x_4+\bar{y}_4)\\ &-\frac{1}{2}(x_1x_3+\bar{y}_1\bar{y}_3)+\frac{1}{2}(x_1\bar{y}_3+\bar{y}_1x_3)-\frac{1}{2}(x_1x_4+\bar{y}_1\bar{y}_4)\\ &+\frac{3}{2}(x_1\bar{y}_4+x_1\bar{y}_4)+(x_2\bar{y}_3+\bar{y}_2x_3)-\frac{1}{2}(x_3x_4+\bar{y}_3\bar{y}_4)\\ &+\frac{1}{2}(x_1\bar{y}_2x_3x_4+\bar{y}_1x_2\bar{y}_3\bar{y}_4). \end{aligned} \tag{4.31}$$

Solving the submodular problem $\min g(\boldsymbol{x},\boldsymbol{y})$ via max-flow yields $\boldsymbol{x}^* = (0,0,0,1)$ and $\boldsymbol{y}^* = (1,1,1,0)$. Again, since $\boldsymbol{x}^* = \bar{\boldsymbol{y}}^*$, it follows that $\boldsymbol{x}^*$ is the global minimizer for f, that is, $f_{\min} = g^*_{\text{GRD}} = -1$.

4.5.2 Approach II: Generators of Expressible Functions

Submodular pseudo-boolean functions form a convex cone in $\mathbf{R}^d$ (Promislow and Young, 2005). Recall that a cone in a vector space is a set $\mathcal{C}$ such that $\mathbf{0} \in \mathcal{C}$ and $\lambda\boldsymbol{x} \in \mathcal{C}$ for every $\lambda \geq 0$ and every $\boldsymbol{x} \in \mathcal{C}$.

One way to work with a cone of expressible pseudo-boolean functions is to find a finite set of generators for the cone, that is, a set of pseudo-boolean functions $\{e_1,\ldots,e_k\}$ such that every function f in the cone can be written $f(\boldsymbol{x}) = \sum_{i=1}^{k} \alpha_i e_i(\boldsymbol{x})$ for $\alpha_i \geq 0$, $i = 1,\ldots,k$. For $n = 4$ variables, generators of the submodular cone have been derived (Promislow and Young, 2005). Apart from the linear functions, there are 10 generator classes (1 quadratic, 2 cubic and 7 quartic generators); see Figure 2 in the paper by Živný and Jeavons (2008) for a complete list. Out of these 10 generators, one (quartic) generator is *not* expressible. This gives a convenient way to represent all expressible functions in 4 variables.

We are of course interested in the cone of symmetric, expressible functions in $2n$ variables. However, it is an open problem to determine this cone's set of generators and even if the generators were known, working with the full set of generators is likely to be intractable.

The generators $\{e_1,\ldots,e_k\}$ for the expressible cone of 4 variables will explicitly define a cone of symmetric and expressible functions over $\mathbf{S}^n$. Every combination of a non-zero quartic coefficient a_{ijkl} of $f(\boldsymbol{x})$ and a

quartic generator e_s, where $s = 1, \ldots, k$, gives a symmetric generator[3]

$$e_s(x_i, x_j, x_k, x_l) + e_s(\bar{y}_i, \bar{y}_j, \bar{y}_k, \bar{y}_l). \tag{4.32}$$

Such a generator will not be able to generate functions with monomials consisting of both $\boldsymbol{x}$ and $\boldsymbol{y}$ variables. Therefore, x_i and y_i need to be exchanged as well:

$$e_s(y_i, x_j, x_k, x_l) + e_s(\bar{x}_i, \bar{y}_j, \bar{y}_k, \bar{y}_l), \tag{4.33}$$

and similarly for the other variables. There are up to $2^4/2 = 8$ (and not 16 due to symmetry) such combinations for every e_s. In an analogous manner, quadratic and cubic generators are constructed for each pair and triplet of indices, respectively. This procedure creates, not counting duplicates:

- 2 quadratic generators for every combination i, j:

$$-x_i x_j - \bar{y}_i \bar{y}_j \text{ and } - x_i y_j - \bar{y}_i \bar{x}_j, \tag{4.34}$$

- 8 cubic generators for every combination i, j, k:

$$\begin{aligned} &-x_i x_j x_k - \bar{y}_i \bar{y}_j \bar{y}_k, \quad -y_i x_j x_k - \bar{x}_i \bar{y}_j \bar{y}_k, \quad \ldots \\ &\ldots, -x_i y_j y_k - \bar{y}_i \bar{x}_j \bar{x}_k, \quad -y_i y_j y_k - \bar{x}_i \bar{x}_j \bar{x}_k. \end{aligned} \tag{4.35}$$

- 132 quartic generators for every combination i, j, k, l, for example:

$$-x_i x_j x_k x_l - \bar{y}_i \bar{y}_j \bar{y}_k \bar{y}_l. \tag{4.36}$$

It can be shown that (i) the set of all submodular and symmetric cubic functions $\Gamma_{\text{sym},3}$ is generated by the quadratic and cubic generators above (modulo linear terms), (ii) the set of symmetric quartic polynomials $\Gamma_{\text{sym},4}$ is a subcone of the cone generated by the generators above (not considering the linear terms).

Problem (4.16) has to be tractable to compute the generalized roof dual bound. The submodular relaxation g is described by the set of generators above as $g(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{K} \alpha_i e_i(\boldsymbol{x}, \boldsymbol{y})$. Constraint (A), that is, $g(\boldsymbol{x}, \bar{\boldsymbol{x}}) =$

[3] $f(\boldsymbol{x})$ is submodular iff $f(\bar{\boldsymbol{x}})$ is submodular. This is because $\overline{\min(\boldsymbol{x}, \boldsymbol{y})} = \max(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ and $\overline{\max(\boldsymbol{x}, \boldsymbol{y})} = \min(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$.

$f(\boldsymbol{x})$ can be written as $\mathsf{A}\boldsymbol{\alpha} = \boldsymbol{a}$, where $\boldsymbol{\alpha}$ is a vector of length K and $\boldsymbol{a}$ is a vector with all polynomial coefficients of f. Constraint (B') is satisfied by $\boldsymbol{\alpha} \geq \mathbf{0}$ and (C) is automatically satisfied by the construction of the set of generators. The objective function $g(\mathbf{0}, \mathbf{0})$ can be written $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{\alpha}$ where $\boldsymbol{c} \in \mathbf{R}^K$. In summary, the maximization problem in (4.16) using generators can be cast as the linear programming problem

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{\alpha} \\ \text{subject to} \quad & \mathsf{A}\boldsymbol{\alpha} = \boldsymbol{a} \\ & \boldsymbol{\alpha} \geq 0. \end{aligned} \tag{4.37}$$

4.6 Heuristics

In many cases, the optimization problem (4.16) does not need to be solved exactly. Minimizing g amounts to solving a maximum flow problem, which is considerably faster than solving a linear program to create g. Simpler, heuristic methods which approximately maximize $g(\mathbf{0}, \mathbf{0})$ are therefore of interest.

Since $c_{ij} = a_{ij} - b_{ij}$, Lemma 4.19 (and similarly Lemma 4.16) can be written on the form

$$b_{ij} \leq \text{(4.28a)} \tag{4.38a}$$
$$b_{ij} \leq a_{ij} + \text{(4.28b)}. \tag{4.38b}$$

Because b_{ij} appears in the objective function, taking the minimum of (4.38a) and (4.38b) will give the optimal value. Therefore, if all higher-order coefficients are fixed, the optimal value of b_{ij} (and, consequently, c_{ij}) is known. A heuristic method tries to quickly choose the higher-order coefficients as well as possible. Then, the quadratic coefficients are chosen optimally. While this does not give the best relaxation, all submodularity constraints will at least be tight.

Naturally, any heuristic can be combined with the generalized roof duality method. The following procedure can be used to compute the roof dual bound:

1. Use heuristics or any type of relaxations to obtain persistencies and simplify f.

2. Apply the generalized roof duality procedure from section 4.3.

The end result will still attain the generalized roof dual bound g^*_{GRD} for the original function f, but much faster for some problems due to the fact that much smaller linear programs are solved.

Cubic Case. For the cubic case the following heuristics is natural: let f be written as the sum of functions $f_1 + f_2 + \dots f_N$, where each f_i is a function of three variables only. Computing the optimal g_i for each of these functions is possible to do very quickly. The sum of optimal relaxations is in general not optimal, as noted before, but the approximation might give a reasonable heuristic.

Quartic Case. For the quartic case, the following, even simpler, approach is effective: Use the procedure from the cubic case and set

$$d_{ijkl} = a^+_{ijkl} \text{ and } b_{ijkl} = -a^-_{ijkl}. \tag{4.39}$$

Even this simple method performs surprisingly well for some application problems, as the experimental section will show (see Figures 4.4 and 4.5). Presumably this is due to the fact that setting b_{ij} to the minimum of (4.22a) and (4.22b) is optimal given that the higher-order coefficients are determined. The submodularity inequalities will always be tight.

4.7 Experiments

This section evaluates generalized roof duality experimentally. When computing the generalized roof duality, every step used linear programming, that is, combinations of linear programming and heuristics were not used, as mentioned in section 4.6. The exception is section 4.7.2, where the problems were preprocessed with heuristic relaxations and simplified, after which the linear programming relaxation was computed. In practice however, one would typically always use a combination of heuristics and linear programming, or heuristics only. Table 4.1 lists the abbreviations of the different relaxation methods.

All linear programming problems were solved using Clp[4], which is a free open-source solver. I have tested the implementation thoroughly;

[4] http://www.coin-or.org/Clp

GRD	Generalized Roof Duality using $\Gamma_{\text{sym},m}$ (sections 4.4 and 4.5.1).
GRD-gen.	GRD using generators for $m = 4$ (section 4.5.2).
GRD-heuristic	The heuristic relaxations (section 4.6).
Fix et al.	The reductions proposed by Fix et al. (2011).
HOCR	The reductions proposed by Ishikawa (2011).

Table 4.1: Abbreviations used in the experimental section.

persistency and lower bounds for each method has been verified on tens of thousands of small polynomials for which the global optimum could be calculated via exhaustive search. The implementation is freely available.[5]

4.7.1 Random Polynomials

The first experiment used synthetically generated polynomials with random coefficients:

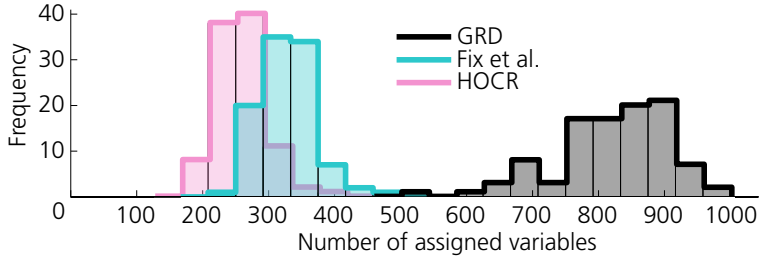
$$f(\boldsymbol{x}) = \sum_{(i,j,k)\in T} f_{ijk}(x_i, x_j, x_k), \tag{4.40}$$

where $T \subseteq \{1 \dots n\}^3$ is a random set of triplets and each f_{ijk} is a cubic polynomial in x_i, x_j and x_k with all its coefficients picked uniformly in $\{-100, \dots, 100\}$. The set of coefficients T was drawn uniformly after making sure that all variables were used once.[6]

Each f was minimized with the different methods listed in table 4.1. After each algorithm finishes, the number of persistencies (also called the number of assignments) is a measure of how well the algorithm performed. The results from 100 problem instances can be seen in Figure 4.1. For this type of polynomials, generalized roof duality significantly outperforms the previous state of the art for every problem. The time required to solve the linear program (4.16) was longer, but in combination with heuristics this time may be shortened significantly. The minimum and maximum number of iterations required was 3 and 12, respectively, with 93% of the problem instances requiring 6 or less. The figures also report the relative difference $(\ell_{\text{GRD}} - \ell)/|\ell_{\text{GRD}}|$.

[5] https://github.com/PetterS/submodular

[6] That is, first the triplets $(1, 2, 3), (4, 5, 6), \dots, (n-2, n-1, n)$ were added to T, after which the remaining triplets were picked uniformly at random.



	Rel. bounds			Time (ms)		
	Min	Med.	Max	Min.	Med.	Max
GRD	0	0	0	422	514	749
GRD-heur.	0.00	0.00	0.00	31	63	125
Fix et al.	0.05	0.09	0.13	15	16	32
HOCR	0.10	0.14	0.20	0	15	31

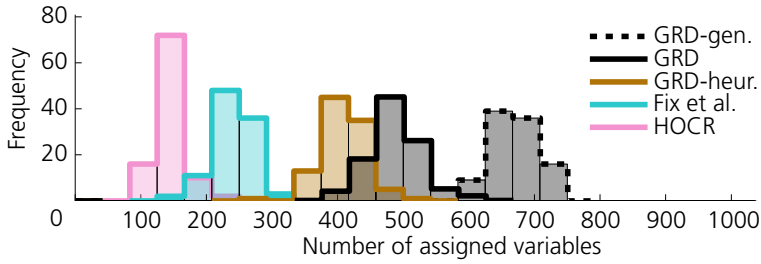
Figure 4.1: Number of assigned variables, relative bounds and running time for 100 random cubic polynomials with $n = 1000$ and $|T| = 1000$. GRD-heuristic is not shown in the histogram because it is almost indistinguishable from GRD.

I also generated random quartic polynomials in the same manner; see figure 4.2. This experiment also resulted in a large separation, and the relative lower bound differences were much larger. The best performing method in terms of lower bounds is, not surprisingly, the GRD method based on generators. Perhaps somewhat surprisingly, it is also faster than GRD based on $\Gamma_{\text{sym},4}$. Even though the linear program for the generator method is much larger, it is easier to solve. Note that this experiment only used $|T| = 300$; with $|T| = 1000$, generalized roof duality only obtained a median of 14 persistencies while HOCR obtained 3.

4.7.2 Applications in Computer Vision

Section 3.6 on page 29 discussed how more general multi-label problems can be solved with fusions. Lemma 4.2 guarantees that the objective function value will never increase when performing such moves. This subsection applies generalized roof duality to a few problems in computer vision.

Image Denoising. Ishikawa (2011) used image denoising as a benchmark problem for higher-order pseudo-boolean minimization. In each iteration



	Rel. bounds			Time (ms)		
	Min	Med.	Max	Min.	Med.	Max
GRD-gen.	−0.10	−0.07	−0.05	360	438	560
GRD	0	0	0	599	810	1478
GRD-heur.	0.04	0.06	0.09	60	100	160
Fix et al.	0.13	0.16	0.20	0	10	20
HOCR	0.42	0.48	0.55	0	10	17

Figure 4.2: Number of assigned variables, relative bounds and running time for 100 random quartic polynomials with $n = 1000$ and $|T| = 300$.

the proposals are generated in two possible ways which are alternated: by blurring the current image or picking all pixels at random. The smoothness term consists of a Fields of Experts model using patches of size 2×2. Thus, quartic polynomials are needed to formulate the image restoration task as a pseudo-boolean minimization problem (see page 30). See chapter 6 for a complete formulation and discussion of the Fields of Experts denoising problem.

Figures 4.4 and 4.5 show a comparison between the different methods for this problem. Generalized roof duality performed very well, often assigning very close to 100% of the problem variables. The plots displaying the number of persistencies in each iteration show the average over the two types of proposals generated, just as Fix et al. (2011) does. Otherwise, the oscillating graphs overlap and the plot becomes hard to read.

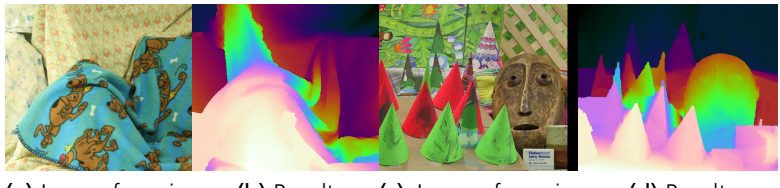
If we instead look at the objective function values versus the time spent computing, the heuristic method still outperforms HOCR, but the difference is smaller. This is due to the fact that GRD has to solve multiple graph cut problems in each iteration while HOCR only has to solve one. The results are shown in figure 4.6. The best performing method is the one by Fix et al., which also solves just one graph cut problem in each iteration, but has better

reductions than HOCR in general. In this application it does not pay off to iterate and compute the best possible solution; it is better to just generate a new proposal.

See chapter 2 for another method for Fields of Experts denoising. As it turns out, continuous methods can solve this problem significantly faster than discrete methods.

Stereo Reconstruction. In dense stereo reconstruction, second order surface priors have recently been used to obtain very good results (Woodford et al., 2009); see Figure 4.3. The problem is to estimate a depth $d(x, y)$ for each pixel in a specific view. A reasonable model is to assign zero cost to all planar surfaces and non-zero cost to all non-planar ones. This requires the smoothness term to contain three types of terms: $\frac{\partial^2 d}{\partial x^2}$, $\frac{\partial^2 d}{\partial y^2}$ and $\frac{\partial^2 d}{\partial x \partial y}$. The first two derivatives require polynomials of degree three, since three points are necessary for estimating non-mixed second derivatives. The mixed derivative, however, requires four points for estimation. Woodford et al. (2009) did not include the mixed derivative in their framework, since no good methods for minimizing degree 4 polynomials were known at the time.[7]

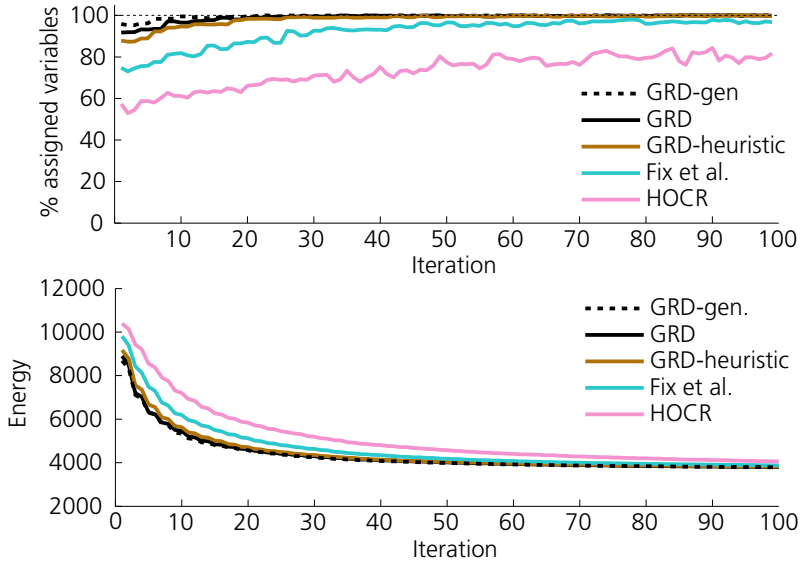
Since the framework uses a heuristic to obtain a complete non-optimal solution (i.e. an upper bound), we instead compare to HOCR. Table 4.2 shows the result for a few image sets. For this problem type, the simple heuristics does not perform better than HOCR.



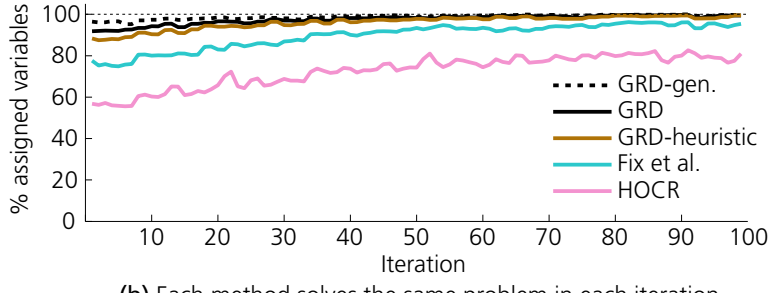
(a) Image from input sequence **(b)** Result **(c)** Image from input sequence **(d)** Result

Figure 4.3: From a calibrated sequence of images, stereo reconstruction can be used to recover a dense depth map (Woodford et al., 2009).

[7] Woodford et al. (2009) claimed that the result of not including $\frac{\partial^2 d}{\partial x \partial y}$ resulted in all harmonic functions having zero cost. This is incorrect, but there are of course non-planar harmonic functions which satisfy $\frac{\partial^2 d}{\partial x^2} = \frac{\partial^2 d}{\partial y^2} = 0$, for example $d(x, y) = xy$.

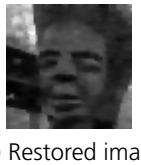


(a) Each method progresses independently



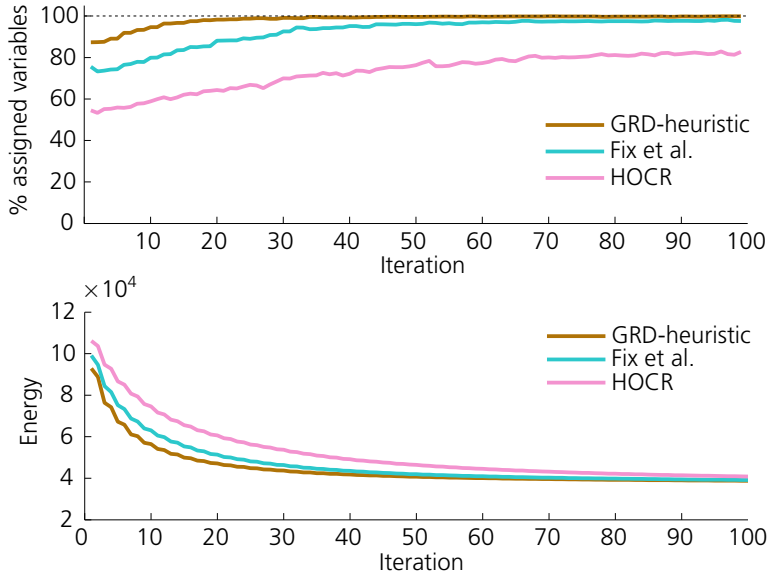
(b) Each method solves the same problem in each iteration

(c) Noisy image

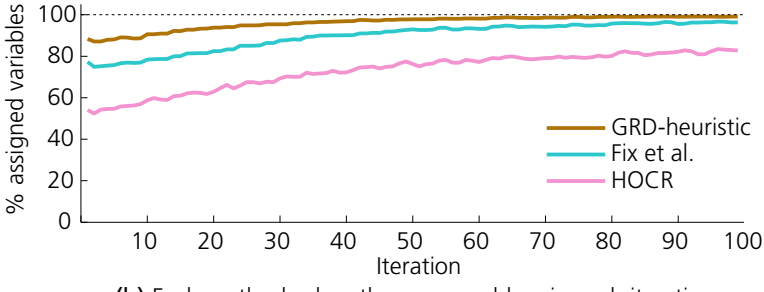


(d) Restored image

Figure 4.4: Restoring a small image. In each iteration a proposal is generated and each pixel can either stay the same or switch to the proposal. A quartic smoothness function is used.



(a) Each method progresses independently

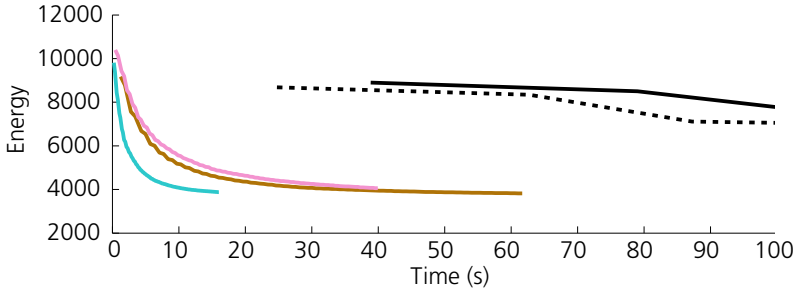


(b) Each method solves the same problem in each iteration

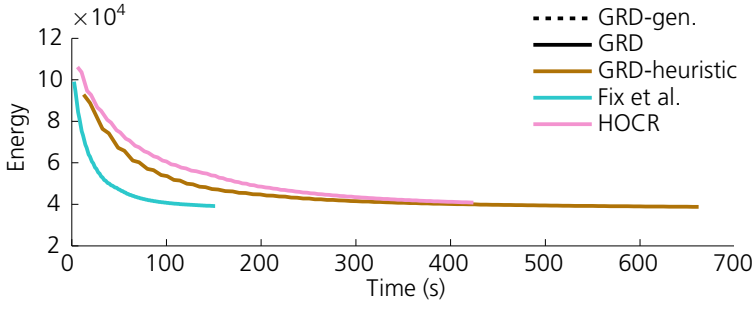
(c) Noisy image

(d) Restored image

Figure 4.5: Restoring a larger image.



(a) Problem in figure 4.4



(b) Problem in figure 4.5

Figure 4.6: Objective function value vs. time for the denoising experiments. The method by Fix et al. (Fix et al., 2011) wins since it performs a single graph cut computation per iteration while having better reductions than HOCR.

Set	Problems	n	Lower bounds ($\cdot 10^{10}$)		
			GRD	GRD-heuristic	HOCR
Cones	259	506250	1.22974	1.22972	1.22945
Cloth3	392	462870	0.833185	0.833176	0.833110

Persistencies ($\cdot 10^5$)		
GRD	GRD-heuristic	HOCR
4.998	4.917	4.986
4.60	4.55	4.59

Table 4.2: Comparison between HOCR and generalized roof duality for stereo reconstruction. The computed numbers are the sample means over all problem instances.

Set	Problems	n	Lower bounds		
			GRD	GRD-heur.	HOCR
CBS	1000	100	$-173_{\pm 2.90}$	$-181_{\pm 2.63}$	$-251_{\pm 4.25}$
RTI	500	100	$-183_{\pm 2.87}$	$-192_{\pm 2.65}$	$-268_{\pm 4.77}$
uf20-91	1000	20	$-26_{\pm 1.39}$	$-31_{\pm 1.51}$	$-54_{\pm 2.55}$
			Times (ms)		
			GRD	GRD-heur.	HOCR
			$77.54_{\pm 14.88}$	$8.93_{\pm 10.34}$	$2.20_{\pm 5.62}$
			$86.80_{\pm 16.73}$	$8.26_{\pm 8.77}$	$2.47_{\pm 5.60}$
			$6.78_{\pm 8.15}$	$1.21_{\pm 4.11}$	$0.53_{\pm 2.83}$

Table 4.3: Lower bounds on some 3-SAT problems from (Hoos and Stützle, 2000). The instances are all satisfiable, so the optimal value is always 0. The computed numbers are the sample means $\pm$ one standard deviation.

4.7.3 3-SAT

If "true" and "false" are encoded as 1 and 0, respectively, a clause "x_i or not x_j or not x_k" is encoded as the term $\bar{x}_i x_j x_k$. The satisfiability problem (SAT) seeks to find an assignment of n variables such that many such clauses are simultaneously satisfied. In other words, SAT amounts to minimize

$$f(\boldsymbol{x}) = \bar{x}_1 x_2 x_3 + x_1 \bar{x}_3 x_6 + \ldots + x_2 \bar{x}_3 \bar{x}_4. \tag{4.41}$$

If the problem is satisfiable, then the optimal value is 0. Table 4.3 shows the performance on some publicly available SAT databases (Hoos and Stützle, 2000). Since persistencies are almost never found for these problems, only lower bounds are interesting. Knowing how the problems are generated, it is trivial to prove a lower bound of 0. This information about the problem structure is not available to any method in Table 4.3, though.

4.7.4 Branch and bound

The experiments so far have mostly focused on the number of assigned variables and the obtained lower bounds. The practical usefulness of better lower bounds has not been discussed. To show that better lower bounds can actually matter in practice, I let each method compute the globally optimal solution of randomly generated polynomials using branch and bound (section 3.7 on page 31).

	$n = 100$, $\|T\| = 30$		$n = 300$, $\|T\| = 80$	
	iterations	total time (s)	iterations	total time (s)
GRD-gen	61	0.7	8,565	92
GRD	515	2.4	$\geq$ 42,280,410	$\geq$ several days
GRD-heur.	2,521	2.3	—	—
Fix et al.	16,195	7.2	—	—
HOCR	1,848,373	1,324	—	—

Table 4.4: Branch and bound on two randomly generated polynomials of degree 4.

Table 4.4 shows that lower bounds can indeed matter—by a lot. The values of $|T|$ have been carefully chosen though; decreasing $|T|$ a little would make the problem easier and the reduction methods would have an advantage. On the other hand, increasing $|T|$ a little would render the problem hopelessly hard for all methods.

This experiment used the simple branching strategy of always splitting the node with the highest bound and replacing it with two new nodes with the first available variable fixed; see section 3.7 on page 32. A good problem-specific branching strategy can be very important in practice.

4.8 Concluding Discussion

This chapter has shown how the roof duality bound for unconstrained quadratic pseudo-boolean functions can be generalized for higher-order functions. The bound is defined as the maximum lower bound over a set of submodular relaxations. The main result is that a solution that attains this bound can be computed in polynomial time.

The main focus of the analysis is on cubic and quartic submodular relaxations, which are the most interesting from an application point of view. The cubic case is more straight-forward and the solution is more elegant than the quartic case, mainly due to the fact that all cubic submodular functions are expressible and the functions form a set which is recognizable. For $m \geq 5$, one can work with any set of higher-order, expressible generators. The generators can be problem-specific and the number of them can depend on the computational resources available.

The experimental results demonstrate that much better lower bounds, and many more assignments can be determined with the generalized roof

dual bound compared to the state of the art. The price to pay is the computational effort due to the time spent on (i) constructing the relaxations, and (ii) the iterative improvements. The method is still very attractive in terms of speed, particularly for large-scale problems involving several thousands of variables. For $m = 4$, the roof dual based on generators is preferable to the approach using relaxations in $\Gamma_{\text{sym},4}$, both in terms of execution times and bounding performance.

Linearization revisited. Recall the integer programming formulation (3.13) on page 25 for minimizing a pseudo-boolean function $f(\boldsymbol{x})$ (3.1). For quadratic f, the linear programming relaxation of (3.13) is the roof duality bound and persistency holds for this relaxation (Hammer et al., 1984; Adams and Dearing, 1994). Another natural way of extending roof duality to functions of higher degree could be to instead call the linear programming relaxation a "generalized roof duality" bound. However, this alternative method does not generally possess the persistency property, as the explicit examples in this subsection will show.

Call the lower bound obtained from the linear programming relaxation $R(f)$ and the generalized roof duality bound $g^*_{\text{GRD}}(f)$. The first example will show that generalized roof duality can give a better bound than linearization; let

$$\begin{aligned} f_1(\boldsymbol{x}) = {} & 80x_0 + 59x_1 + 16x_2 - 36x_3 \\ & - 12x_0x_1 + 114x_0x_2 + 3x_0x_3 - 55x_1x_2 - 33x_1x_3 + 72x_2x_3 \\ & - 15x_0x_1x_2 - 86x_0x_1x_3 - 63x_0x_2x_3 + 71x_1x_2x_3. \end{aligned} \tag{4.42}$$

The minimum of f_1 is $f_1(0, 0, 0, 1) = -36$. The lower bounds are $R(f_1) = -72.5$ and $g^*_{\text{GRD}}(f_1) = -41$ (no assignments were obtained in the first maximization of $g(\mathbf{0}, \mathbf{0})$). As a second example, consider

$$\begin{aligned} f_2(\boldsymbol{x}) = {} & -116x_0 - 27x_1 + 20x_2 - 140x_3 \\ & - 11x_0x_1 + 106x_0x_2 + 104x_0x_3 + 69x_1x_2 + 28x_1x_3 + 4x_2x_3 \\ & + 58x_0x_1x_2 + 79x_0x_1x_3 + 79x_0x_2x_3 - 94x_1x_2x_3. \end{aligned} \tag{4.43}$$

The unique minimizer to f_2 is $(1, 1, 0, 0)$, with an optimal value of -154. A solution to the LP relaxation, however, is $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 1\right)$, with $R(f_2) = -186$. This disproves persistency for the LP relaxation of degrees > 2. On the other

hand, generalized roof duality gives $g^*_{\text{GRD}}(f_2) = -239.5$, showing that the LP relaxation can indeed give better lower bounds.

I should add that during these experiments the integral variables in the relaxed solution almost always took the correct value. In fact, many polynomials were tested before the counterexample f_2 was found. Therefore, keeping the integer-valued variables would probably be a good heuristic in e.g. a branch and bound framework.

Linearization hierarchies. Adams et al. (1998) have proposed a hierarchy of linear programming relaxations LP(d, n), where degree$(f) = m \leq d \leq n$. When $d = m = 2$, standard roof duality is obtained (Adams and Dearing, 1994) and the relaxation becomes stronger as d increases. When $d = n$, the relaxation is always tight. Unfortunately, persistency only holds in two cases: $d = 2$ and $d = n - 1$. However, Adams et al. (1998) proves that persistency can be extracted anyway in some cases: either using duality or when $n - d$ components of the solutions are boolean. I think investigating this hierarchy for difficult problems in computer vision might be interesting.

Beyond the boolean case. Our first results on generalized roof duality appeared in ICCV 2011. Windheuser et al. (2012) have subsequently generalized some of the theorems in this chapters to ordered sets with more than two elements. Chapter 15 contains a brief account of this work.

4.9 Open Problems

Generators for higher degrees. It is an open problem to characterize the symmetric submodular cone $\Gamma_{\text{sym},m} \subset \Gamma_{\text{suff},m}$ and to derive generators for the cone of expressible pseudo-boolean functions when $m \geq 5$. Živný et al. (2009) conjecture that the generators are given by so-called *upper and lower fans*. These issues are left for future work.

Cubic relaxations. Another open problem is whether the cubic submodular relaxations can be improved by enlarging the set of relaxations to include bisubmodular and higher-order submodular relaxations. Theorem 4.6 tells us that when degree$(f) = 2$, considering relaxations of the same degree is enough. On the other hand, Kolmogorov (2010) gives an example with

degree$(f) = 4$ where a bisubmodular relaxation strictly dominates any submodular relaxation.

CONJECTURE 4.21 *When degree$(f) = 3$, the tightest cubic submodular relaxation dominates all other submodular relaxations of arbitrary degree.*

Generalized roof duality and reductions. A natural question to ask is whether the generalized roof duality bound always is at least as good as the bound obtained from any reduction. A reduction is a polynomial $\tilde{f}(\boldsymbol{x}, \boldsymbol{z})$ such that $f(\boldsymbol{x}) = \min_{\boldsymbol{z}} \tilde{f}(\boldsymbol{x}, \boldsymbol{z})$. When roof duality is applied to the reduction, one obtains $\tilde{g}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}, \boldsymbol{w})$, where $\boldsymbol{y}$ and $\boldsymbol{w}$ correspond to $\boldsymbol{x}$ and $\boldsymbol{z}$, respectively. These variables can then be minimized out from $\tilde{g}$:

$$h(\boldsymbol{x}, \boldsymbol{y}) = \min_{\boldsymbol{z}, \boldsymbol{w}} \tilde{g}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}, \boldsymbol{w}). \tag{4.44}$$

The function h is almost like a submodular relaxation of f. It is submodular, since submodularity is preserved when minimizing out variables.[8] It is symmetric, since

$$h(\boldsymbol{x}, \boldsymbol{y}) = \min_{\boldsymbol{z}, \boldsymbol{w}} \tilde{g}(\bar{\boldsymbol{y}}, \bar{\boldsymbol{w}}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}) = \min_{\boldsymbol{z}, \boldsymbol{w}} \tilde{g}(\bar{\boldsymbol{y}}, \boldsymbol{w}, \bar{\boldsymbol{x}}, \boldsymbol{z}) = h(\bar{\boldsymbol{y}}, \bar{\boldsymbol{x}}). \tag{4.45}$$

However, it does not exactly satisfy requirement (A) (on page 39). But

$$\begin{aligned} h(\boldsymbol{x}, \bar{\boldsymbol{x}}) = \min_{\boldsymbol{z}, \boldsymbol{w}} \tilde{g}(\boldsymbol{x}, \boldsymbol{z}, \bar{\boldsymbol{x}}, \boldsymbol{w}) &\leq \min_{\boldsymbol{z}} \tilde{g}(\boldsymbol{x}, \boldsymbol{z}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}) \\ &= \min_{\boldsymbol{z}} \tilde{f}(\boldsymbol{x}, \boldsymbol{z}) = f(\boldsymbol{x}). \end{aligned} \tag{4.46}$$

Although I do not have a proof, I think the above motivation and the extensive experimental results make the following conjecture reasonable:

CONJECTURE 4.22 *Let $f : \mathbf{B}^n \to \mathbf{R}$ be of degree 3 and $\tilde{f} : \mathbf{B}^{n+n'} \to \mathbf{R}$ be of degree 2 such that $\tilde{f}(\boldsymbol{x}) = \min_{\boldsymbol{z}} \tilde{f}(\boldsymbol{x}, \boldsymbol{z})$ for all $\boldsymbol{x} \in \mathbf{B}^n$. Then the generalized roof duality bound of f is greater than or equal to the roof duality bound of $\tilde{f}$.*

[8] To see this, let $f(\boldsymbol{x}) = \min_{\boldsymbol{z}} g(\boldsymbol{x}, \boldsymbol{z})$ where g is submodular. Then $f(\boldsymbol{x} \vee \boldsymbol{y}) + f(\boldsymbol{x} \wedge \boldsymbol{y}) \leq g(\boldsymbol{x} \vee \boldsymbol{y}, \boldsymbol{z} \vee \boldsymbol{w}) + g(\boldsymbol{x} \wedge \boldsymbol{y}, \boldsymbol{z} \wedge \boldsymbol{w}) = g((\boldsymbol{x}, \boldsymbol{z}) \vee (\boldsymbol{y}, \boldsymbol{w})) + g((\boldsymbol{x}, \boldsymbol{z}) \wedge (\boldsymbol{y}, \boldsymbol{w})) \leq g(\boldsymbol{x}, \boldsymbol{z}) + g(\boldsymbol{y}, \boldsymbol{w})$ for any $\boldsymbol{z}$ and $\boldsymbol{w}$. Minimizing over them proves submodularity for f.

Chapter 5

Other Approaches to Pseudo-Boolean Optimization

The discussion in the previous chapter mentioned an alternative definition of "generalized roof duality." That definition amounted to solving a linear programming relaxation and could, in some cases, give a better lower bound than generalized roof duality. Unfortunately, it lacked the important property of persistency. In this short chapter, I will introduce two new optimization methods for pseudo-boolean functions. The first one works for functions of any degree and has persistency. The second method is an idea on how quadratic functions can be optimized better than using roof duality if the variables are arranged in a grid.

5.1 Using Bipartite Vertex Packing

Hammer et al. (1984) used the vertex packing problem to introduce persistency to quadratic pseudo-boolean functions. The approach readily carries over to higher degree polynomials, although I am not aware of anyone actually trying it. The purpose of this section is to explore this alternate approach to "generalized roof duality."

The vertex packing problem is the following integer linear program:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{maximize}} \quad & \sum_{i=1}^{n} \alpha_i x_i \\ \text{subject to} \quad & x_i + x_j \leq 1, \quad (i,j) \in E, \\ & \text{and } x_i \in \{0,1\}, \end{aligned} \tag{5.1}$$

where the weights $\{\alpha_i\}$ are non-negative and E is a set of index pairs. Nemhauser and Trotter (1975) showed that when relaxing the integer con-

straint to $x_i \in [0,1]$, there is an optimal solution $\hat{\boldsymbol{x}}$ for which $\hat{x}_i \in \left\{0, \frac{1}{2}, 1\right\}$. More importantly, persistency holds for this solution, in precisely the same sense as I have discussed in earlier chapters (e.g. Theorem 3.4 on page 26). Finally, the relaxation of (5.1) can be efficiently solved as another vertex packing problem in a bipartite graph. This is a crucial fact, as it allows a generalization of roof duality which never uses the relatively costly linear programming at any step.

I should note that the (weighted) vertex packing problem is also referred to as the problem of finding the maximum (weighted) independent set. Also, seeing that it is, informally, the complement of finding the (weighted) minimum vertex cover is not hard.

The vertex packing problem is a standard problem formulated as maximization. Therefore, this section will also discuss maximization. Write f as a posiform

$$f(\boldsymbol{x}) = \phi(\boldsymbol{x}) = \sum_i \alpha_i u_i + \sum_{(i,j)\in\mathcal{Q}} \alpha_{i,j} u_i u_j + \sum_{(i,j,k)\in\mathcal{C}} \alpha_{i,j,k} u_i u_j u_k + \dots$$

or, more succinctly,

$$\phi(\boldsymbol{x}) = \sum_{T\in\mathcal{T}} \alpha_T \prod_{i\in T} u_{T,i}, \tag{5.2}$$

where all $\alpha_T > 0$. The symbol $u_{T,i}$ should be thought of as a placeholder which is equal to either x_i or $\bar{x}_i$. Now we introduce the following integer program:

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}}{\text{maximize}} \quad & M\sum_i (x_i + y_i - 1) + \sum_{T\in\mathcal{T}} \alpha_T z_T \\ \text{subject to} \quad & x_i + y_i \leq 1, \\ & z_T + \bar{u}_{T,i} \leq 1, \quad \forall i \in T, \quad \forall T \in \mathcal{T}, \\ & \text{and all variables in } \{0,1\}. \end{aligned} \tag{5.3}$$

In this formulation, y_i is a variable representing $\bar{x}_i$. The placeholder $u_{T,i}$ is therefore replaced with x_i or y_i and $\bar{u}_{T,i}$ with the opposite. M is a sufficiently large constant (i.e. guaranteed to be larger than the maximum of f). Note that no extra variables are needed whenever $|T| = 1$.

Example. The function $f(\boldsymbol{x}) = x_1x_2 - x_1x_2x_3$ can be reformulated as the posiform $\phi(\boldsymbol{x}) = x_1x_2\bar{x}_3$. The objective function in (5.3) becomes $M\sum_{i=1}^{3}(x_i + y_i - 1) + z$ with the constraints $x_i + y_i \leq 1$ for all i, $z + \bar{x}_1 \leq 1$, $z + \bar{x}_2 \leq 1$, and $z + x_3 \leq 1$.

LEMMA 5.1 *If $\boldsymbol{x}^*$ is part of a maximizer $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$ to (5.3) with optimal value r, then $r = \max_{\boldsymbol{x}} f(\boldsymbol{x}) = f(\boldsymbol{x}^*)$.*

Proof. The linear inequalities along with the first M-sum in the objective function guarantees that $\boldsymbol{x} = \bar{\boldsymbol{y}}$ in the optimum. The variables z_T represent each monomial and since all $\alpha_T > 0$, $z_T = 1$ in the solution if and only if $\prod_{i\in T} u_{T,i} = 1$. □

Problem (5.3) is a vertex packing problem (5.1). I used a maximum flow solver (Goldberg et al., 2011) to solve the resulting bipartite vertex packing problem as a minimum cut problem. Figure 5.1 shows the same experiment as in figure 4.2 on page 61. Although using this approach almost always gives worse results than GRD from the previous chapter, there are counterexamples. For a degree-three polynomial with five variables the optimal submodular relaxation gave a lower bound of -413, while the vertex packing approach found the global optimum of -359.

Solving the vertex packing problem as a general minimum cut problem is not that efficient. The time required for the vertex packing method was about the same as for the heuristic method from the previous chapter. The main advantage of this approach is that it offers an alternate generalization of roof duality that immediately works for polynomials of any degree. The downsides are that the result depends on how the conversion to a posiform is done and that the performance is not better than the state of the art for low-degree polynomials. Whether custom maximum flow methods for bipartite graphs can improve the performance considerably is left as future work.

5.2 Elimination on a Grid

This section will describe an idea for minimizing quadratic pseudo-boolean functions defined on a grid similar to the work by Carr and Hartley (2009). Let us take figure 5.2a as an example. The variable x_8 is a point in a grid

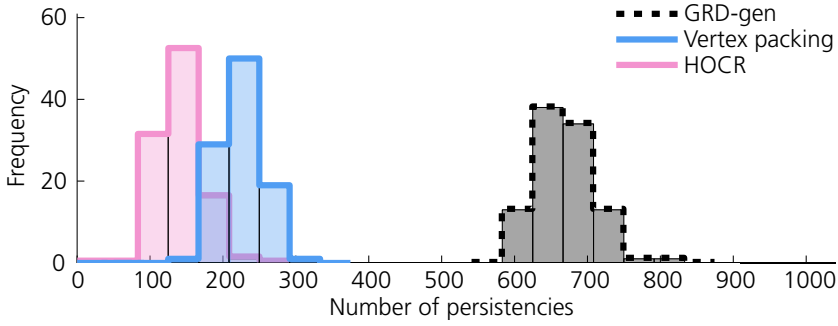


Figure 5.1: Vertex packing experiment with the same types of polynomials as in figure 4.2 on page 61. The approach in this section is slightly better than HOCR, but not by much.

and x_3, x_7, x_9 and x_{13} are its four neighbors. Let the terms involving x_8 be denoted

$$E_1(x_3, x_8) + E_2(x_7, x_8) + E_3(x_8, x_9) + E_4(x_8, x_{13}). \tag{5.4}$$

Minimizing the sum of these four terms is equivalent to minimizing the new function

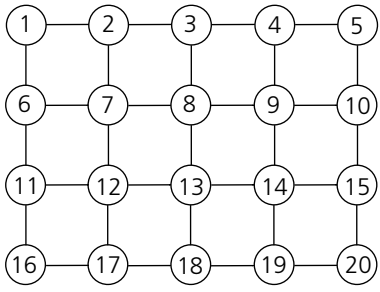
$$E_8(x_3, x_7, x_9, x_{13}) = \min_{x_8} \left(E_1(x_3, x_8) + E_2(x_7, x_8) + E_3(x_8, x_9) + E_4(x_8, x_{13})\right). \tag{5.5}$$

E_8 is a function of the four remaining variables and its 16 values can be computed quickly (by trying both possibilities for x_8). In the end, x_8 and the terms containing it are replaced by

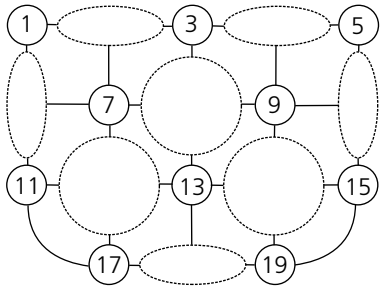
$$\bar{x}_3\bar{x}_7\bar{x}_9x_{13}E_8(0,0,0,1) + \bar{x}_3\bar{x}_7\bar{x}_9\bar{x}_{13}E_8(0,0,0,0) + \ldots + x_3x_7x_9x_{13}E_8(1,1,1,1). \tag{5.6}$$

This gives us a method of replacing every other variable in the grid with cliques of degree four in the remaining variables. Figure 5.2b shows the resulting grid after all eliminations of this type have been performed.

Why would this be a better idea than using quadratic roof duality? One way of solving the quartic problem is to use a special set of reductions



(a) Original grid. Edges denote pairwise terms between variables.



(b) Grid after reduction. Higher-order terms are represented by dashed ovals.

Figure 5.2: Half the number of variables in a 4-connected grid can be eliminated and replaced by degree-4 cliques.

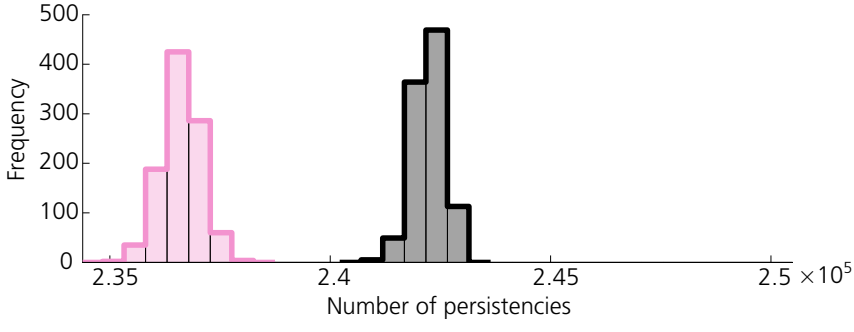


Figure 5.3: Experiments on 500 × 500 grids with randomly generated pairwise terms. Pink is quadratic roof duality and black is the method described in this section. Note the scale on the x-axis! There is not much separation between the two methods.

that recovers the original quadratic problem. Because the previous chapter showed that generalized roof duality often performed better than a fixed set of reductions, this is potentially a better way than using quadratic roof duality.

To evaluate this method, I generated 500×500 grids with random pairwise terms. The number of persistencies is shown in figure 5.3. The baseline method is standard quadratic roof duality, which is compared to quadratic roof duality alternated with eliminations. By itself, elimination and quartic roof duality do not perform better than standard roof duality (recall that the quartic relaxation is computed for a subset of all submodular functions), but combined with standard roof duality more persistencies are always found. When computing the generalized roof duality of the quartic functions, the generators from the previous chapter were used. Although using eliminations always found more persistencies, I did not find the difference great enough to motivate further testing.

Chapter 6

Continuous Fields of Experts Denoising

For many optimization problems, a local model built using derivatives simply does not give any useful information about the global structure, making it hard for gradient descent methods to find good solutions. For some of these problems, the discrete optimization methods discussed in chapter 3 are able to avoid undesired local minima.

Fields of Experts (FOE) is a sophisticated prior on the statistics of natural images (Roth and Black, 2009). It has a larger clique structure that is capable of capturing higher order interactions around each image pixel than models solely based on pairwise interactions. One area where the FOE priors have been used to great success is image denoising. For example, the recent state-of-the-art results in image denoising (Jancsary et al., 2012) use FOE as one part in a more complicated machine learning system.

There seems to be a general sense that MAP inference in problems arising from the use of FOE models is hard and that continuous optimization methods may not be suitable for it. Thus an increasingly sophisticated (and expensive) array of discrete optimization methods have been developed to solve them (Lan et al., 2006*a*; Potetz, 2007; Ishikawa, 2009*a*,*b*, 2011; Fix et al., 2011). For this reason, chapter 4 used image denoising with an FOE prior as a benchmark problem for generalized roof duality. This short chapter will explore another approach. In fact, a relatively simple continuous optimization method can be used to solve the FOE denoising problem cheaply and effectively.

6.1 Denoising using Fields of Experts

Given a noisy image $\boldsymbol{u}$, the negative log-likelihood for an image $\boldsymbol{x} \in \mathbf{R}^n$ using the FOE prior is

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \frac{(x_i - u_i)^2}{2\sigma^2} + \sum_{P \in \mathcal{P}} \sum_{k=1}^{K} \alpha_k \log\left(1 + \frac{1}{2}\left(\boldsymbol{b}_k^{\mathrm{T}} \boldsymbol{x}_P\right)^2\right), \quad (6.1)$$

where P is an image patch in the set of all $m \times m$ patches $\mathcal{P}$ of $\boldsymbol{x}$ and σ is the standard deviation of the Gaussian noise in $\boldsymbol{u}$. The coefficients α_k and the filters $\boldsymbol{b}_k$ for $k = 1, \ldots, K$ are estimated from a database of natural images (Roth and Black, 2009). Thus, finding the maximum likelihood image given a noisy image $\boldsymbol{u}$ can be formulated as finding the image $\boldsymbol{x}$ that minimizes (6.1).

6.2 Non-linear Least Squares Formulation

A robustified non-linear least squares problem is the minimization of a function of the form (Agarwal and Mierle, 2012):

$$\sum_{i=1}^{N} \rho_i\left(\|f_i(\boldsymbol{x}_{P_i})\|^2\right). \quad (6.2)$$

Here, ρ_i is a *loss function*. If it is the identity function, the problem is an ordinary non-linear least squares problem; see section 2.3 on page 17. Otherwise, under some mild conditions on ρ_i, (6.2) can still be solved (locally) using a non-linear least squares algorithm after appropriate modifications to the residual vector and the Jacobian matrix (Triggs et al., 2000).

Observe that the first sum in (6.1) can be written in the form of (6.2) by setting

$$f_i(x) = \frac{x - u_i}{\sqrt{2}\sigma} \qquad \text{and} \qquad \rho_i(s) = s, \quad (6.3)$$

and the second sum by setting

$$f_k(\boldsymbol{x}_P) = \boldsymbol{b}_k^{\mathrm{T}} \boldsymbol{x}_P \quad \text{and} \quad \rho_k(s) = \alpha_k \log\left(1 + \frac{s}{2}\right). \quad (6.4)$$

Thus, minimizing (6.1) is a robustified non-linear least squares problem.

Method		test001		test002	
		obj.	time	obj.	time
Reported by Ishikawa	(2011; 2009*a*)	37769	1326 s.	25030	1330 s.
	(2009*b*)	38132	71 s.	24831	81 s.
Own computer	(2011; 2009*a*)	37691	625 s.	24997	631 s.
	(2009*b*)	37686	16 s.	25129	24 s.
	Levenberg-Marquardt	37374	2 s.	24556	2 s.

test003		test004	
obj.	time	obj.	time
29805	1305 s.	27356	1290 s.
29683	67 s.	27354	79 s
29762	623 s.	27330	604 s.
29734	22 s.	27219	18 s.
29434	2 s.	27088	2 s.

Table 6.1: Denoising the four 160 × 240 test images used by Ishikawa (2009*a*,*b*, 2011) (shown in figure 6.1) with 2 × 2 filters, $K = 3$ and $\sigma = 20$. The table reports final objective function values and running times.

Figure 6.1: The four denoised test images using a 2 × 2 FoE model. Table 6.1 shows the quantitative results.

Method	test001		test002	
	obj.	time	obj.	time
Levenberg-Marquardt, 3×3, $K = 8$	55186	25 s.	39750	24 s.
Levenberg-Marquardt, 5×5, $K = 24$	61304	132 s.	42518	139 s.

test003		test004	
obj.	time	obj.	time
44798	25 s.	42367	20 s.
47093	149 s.	44820	113 s.

Table 6.2: Denoising with higher-order FoE models using the noisy images in figure 6.1. The objective function values are from different optimization problems and are therefore not comparable.

6.3 Experiments

The continuous experiments used Ceres Solver (Agarwal and Mierle, 2012) with the Levenberg-Marquardt algorithm (Nocedal and Wright, 2006) in combination with the CHOLMOD sparse Cholesky factorization library (Chen et al., 2008) to minimize the robustified non-linear least squares formulation of (6.1). The two state-of-the-art discrete optimization methods are based on proposals (see section 3.6 on page 29) generated in two different ways. An implementation of the method using random and blurred proposals (Ishikawa, 2011) is publicly available. Higher-order gradient descent (Ishikawa, 2009*b*) is not, but was easy to implement.

All experiments used the noisy image as the initial point for the solver. Since the discrete methods use integer images, the continuous solution using the 2×2 filters was rounded to the nearest integer in $\{0, \ldots, 255\}$ to get a fair comparison. This increased the final objective function value a little bit ($< 0.2\%$). All of the experiments were performed on a 3.47 GHz Intel Xeon and did not use any multi-threaded capabilities.

The first experiment processed four test images that has been commonly used for evaluating discrete methods using the 2×2 FOE model. Table 6.1 shows the results, both as reported by Ishikawa (2009*b*) and of my own experiments. Figure 6.1 illustrates the minima found by the continuous method; all methods produce visually indistinguishable results. The non-linear least squares solver finds a lower objective function value in a fraction of the time used by the discrete methods.

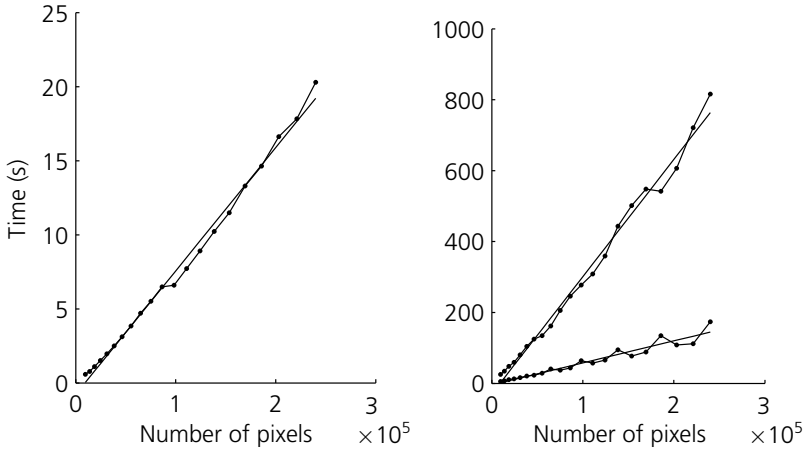


Figure 6.2: The time required to denoise an image is approximately linear in the number of pixels. The figure shows graphs for 2×2 (left), 3×3 and 5×5 (right) filter sizes.

Apart from the four test images used for benchmarking in previous publications, the next experiment added noise to the 100 test images in the Berkeley Segmentation Database (Martin et al., 2001). The non-linear least squares solver always found a better objective value than the method in (Ishikawa, 2009*b*) (1% on average) and was much faster (11 and 100 seconds, respectively, on average). The choice of initial point does not seem important; an all-zero image worked just as well.

Because of limitations of the methods used for pseudo-boolean optimization, discrete methods for FoE inference have focused on the 2×2 case. The methods by Ishikawa (2011), Fix et al. (2011), and chapter 4 are not capable of handling the degree-9 polynomials that would be required for 3×3 inference. Even representing an arbitrary degree-9 term requires 512 coefficients and each one of them has to be reduced to quadratic terms. In contrast, nothing is preventing a non-linear least squares solver from using 3×3 or 5×5 filters. Table 6.2 contains the running times and final objective function values for the four test images.

Finally, the last experiment processed the same image in many different sizes with the non-linear solver. Figure 6.2 shows that the running time is approximately linear in the number of pixels.

6.4 Discussion

Non-linear least squares performs very well when applied to MAP Fields of Experts denoising. It is several times faster than the fastest method based on discrete optimization and it is immediately applicable to problems with larger filter sizes.

As pointed out in chapter 4 (page 65), the more efficient reductions by Fix et al. (2011) do improve the speed, but only by at most 2 seconds when used with higher-order gradient descent (Ishikawa, 2009*b*). Generalized roof duality (chapter 4) also solves the individual pseudo-boolean problems better, but is slower. Two other approaches exist, but they are both significantly slower (Lan et al., 2006*a*; Potetz, 2007).

While FOE denoising is a useful benchmark problem for discrete optimization, we should keep in mind that continuous methods can solve these problems much more efficiently.

Chapter 7

Parallel and Distributed Graph Cuts

Chapter 3 briefly explained that many problems in low-level computer vision can be formulated as labeling problems using Markov random fields. Among the examples are image segmentation, image restoration, dense stereo estimation and shape estimation (Greig et al., 1989; Boykov and Kolmogorov, 2004; Boykov et al., 1998; Rother et al., 2004; Lempitsky and Boykov, 2007). For problems with three or more labels, α-expansion and other approximation methods (Komodakis et al., 2007; Komodakis and Tziritas, 2007; Carr and Hartley, 2009) are available.

This and the next chapter will describe how the technique of dual decomposition (see chapter 2, page 13) applies to these and related optimization methods. The focus for this chapter will be graph cuts. We will see that dual decomposition is a valuable tool to parallelize existing algorithms to make them run faster on modern multi-core processors. We will also see that prohibitively large memory requirements can be made tractable on both desktop computers and supercomputer clusters. Examples of optimization problems with huge memory requirements are segmentation problems in three or more dimensions and curvature regularization problems. Finally, we will see that dual decomposition also provides an interesting way of implementing algorithms on parallel hardware, such as graphical processing units (GPUs).

Decomposition of graph cuts. The contribution of this chapter is a parallel version of the graph cut method by Boykov and Kolmogorov (2004), which is henceforth referred to as "BK." The approach has a number of advantages compared to other parallelization methods:

1. The BK-method has been shown to have superior performance compared to competing methods for a number of applications (Boykov and Kolmogorov, 2004), most notably sparse 2D graphs and moderately sized 3D graphs.

2. It is possible to reuse the search trees in the BK-method (Kohli and Torr, 2005, 2007), which makes the dual decomposition approach attractive.

3. Perhaps most importantly, experiments demonstrate good empirical performance with significant speed-ups compared to single-threaded computations, both on multi-core platforms and multi-computer networks.

Naturally, there are also some disadvantages:

1. There is no theoretical guarantee that the parallelization will be faster for every problem instance. Already for the BK-method no polynomial time guarantee is known, and I do not give one for the number of iterations, either. In practice this matters little.

2. The current implementation is only effective for graphs for which the BK-method is effective. The underlying dual decomposition principle can however be applied in combination with any graph cut algorithm.

7.1 Previous Approaches to Graph Cuts in Vision

The work in this chapter builds on the following two trends: the ubiquity of maximum flow computations in computer vision and the tendency of modern microprocessor manufacturers to increase the number of cores in mass-market processors. This implies that an efficient way of parallelizing maximum flow algorithms would be of great use to the community. Due to a result from Goldschlager et al. (1982), there is little hope in finding a general algorithm for parallel maximum flow with guaranteed performance gains. However, the graphs encountered in computer vision problems are often sparse with much fewer edges than the maximum $n^2 - n$ in a graph with n vertices. The susceptibility to parallelization depends on the structure and costs of the graph.

There are essentially three types of approaches used in computer vision for solving the maximum flow/minimum cut problem:

Augmenting paths. The most popular method due to its computational efficiency for 2D problems and moderately sized 3D problems with low connectivity (i.e., sparse graphs) is the BK-method using augmenting paths (Boykov and Kolmogorov, 2004). However, as augmenting path algorithms use non-local operations, they have not been considered as a viable candidate for parallelization. One way of making multiple threads cooperate is to divide the graph into disjoint parts. This is the approach taken by Liu et al. (2009), in which the graph is split, solved and then split differently in an iterative fashion until no augmenting paths can be found. The key observation is that the search trees of the subgraphs can be merged relatively fast. The more recent work by Liu and Sun (2010) splits the graph into many pieces which, in turn, multiple threads solve and merge until only one remains and all augmenting paths have been found. The method in this chapter also splits the graph into multiple pieces, but differs in that it does not require a shared-memory model, which makes distributed computation possible.

Push-relabel. The push-relabel algorithm (Goldberg and Tarjan, 1986) is an algorithm suitable for parallelization. The implementation by Delong and Boykov (2008) has been tested for up to 8 processors with good results. There have been attempts to implement this method on a GPU, the latest being CUDA cuts by Vineet and Narayanan (2008; 2009), but my tests of the (freely available) implementation only gave the correct result for graphs with low regularization. Another attempt was made by Hussein et al. (2007), which performed all experiments on generated images with a very low amount of regularization. Solving such graphs essentially reduces to trivial thresholding of the data term. The earliest reference I was able to find was the paper by Dixit et al. (2005) which does not report any speed-up compared to sequential push-relabel.

Convex optimization. Another approach to parallel graph cuts is to formulate the problem as a linear program. Under the assumption that all edges are bidirectional, the problem can then be reformulated as an ℓ_1 minimization problem. The work by Bhusnurmath and Taylor (2008) attempts to solve this problem with Newton iterations using the conjugate gradient method with a suitable preconditioner. Matrix-vector multiplications can be highly parallelized, but this approach has not proven to be significantly

faster than the single-threaded BK algorithm for any type of graph, even though Bhusnurmath and Taylor used a GPU in their implementation.

Convex optimization based on a GPU has also been used to solve continuous versions of graph cuts, e.g. (Klodt et al., 2008). However, the primary advantage of continuous cuts is the reduction of metrication errors due to discretization.

Graph cuts is also a popular method for multi-label problems using, e.g., iterated α-expansions. Such local optimization methods can naturally be parallelized by performing two different expansions in parallel and then trying to fuse the solutions, as done in (Lempitsky et al., 2010).

7.2 Decomposition of Graphs

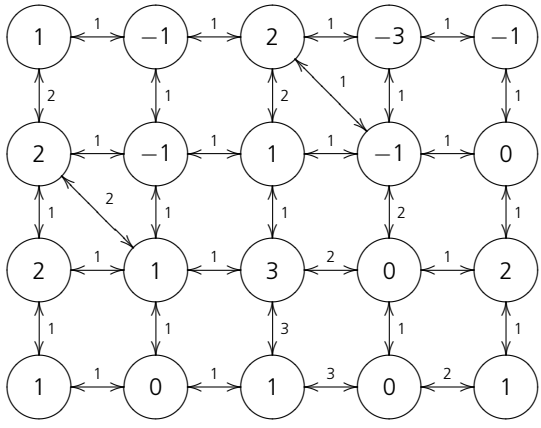
Section 2.2 on page 13 gave an introduction to dual decomposition. This section describes how the graph is split and how the dual variables enter the two subgraphs. The next two sections provide extensive experiments for graphs in 2, 3 and 4 dimensions, both multi-threaded and distributed across many computational nodes in a supercomputer.

7.2.1 Graph Cuts as a Linear Program

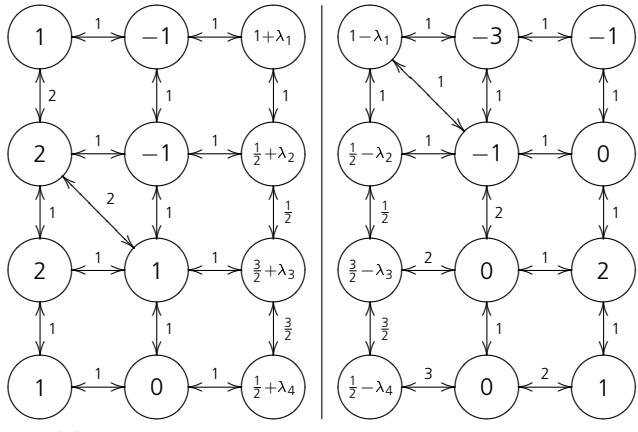
Finding the maximum flow, or, by duality, the minimum cut in a graph can be formulated as a linear program. Let $G = (V, \boldsymbol{c})$ be a graph where $V = \{s, t\} \cup \{1, 2, \ldots, n\}$ are the source, sink and vertices, respectively, and $\boldsymbol{c}$ the edge costs. A cut is a partition S, T of V such that $s \in S$ and $t \in T$. The minimum cut problem is finding the partition where the sum of all costs of edges between the two sets is minimal. It can be formulated as

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \sum_{i,j \in V} c_{ij} x_{ij} \\ \text{subject to} \quad & x_{ij} + x_i - x_j \geq 0, \quad i, j \in V \\ & x_s = 0, \quad x_t = 1, \quad \boldsymbol{x} \geq 0. \end{aligned} \tag{7.1}$$

The variable x_i indicates whether node i is part of S or T ($x_i = 0$ or 1, respectively) and x_{ij} indicates whether the edge (i, j) is cut or not. The variables are not constrained to be 0 or 1, but there always exists one such solution, according to the duality between maximum flow and minimum



(a) Original graph. Numbers inside the nodes indicate s/t connections, positive for *s*, negative for *t*.



(b) Subproblems with vertices in M and N, respectively.

Figure 7.1: The graph decomposition into sets M and N. The pairwise functions in $M \cap N$ are part of both E_M and E_N and has to be weighted by 1/2. Four dual variables $\lambda_1 \ldots \lambda_4$ are introduced as s/t connections.

cut. Let $\mathcal{D}_V$ denote the convex set defined by the constraints in (7.1) for a node set V.

7.2.2 Splitting the Graph

Now pick two sets M and N such that $M \cup N = V$ and $\{s, t\} \subset M \cap N$ and assume that when $i \in M \setminus N$ and $j \in N \setminus M$, $c_{ij} = c_{ji} = 0$. That is, every edge is either within M or N, or within both; see figure 7.1. The objective function in (7.1) can be rewritten as:

$$\sum_{i,j \in V} c_{ij} x_{ij} = \sum_{i,j \in M} c_{ij} x_{ij} + \sum_{i,j \in N} c_{ij} x_{ij} - \sum_{i,j \in M \cap N} c_{ij} x_{ij}. \tag{7.2}$$

Define two objective functions for M and N, respectively:

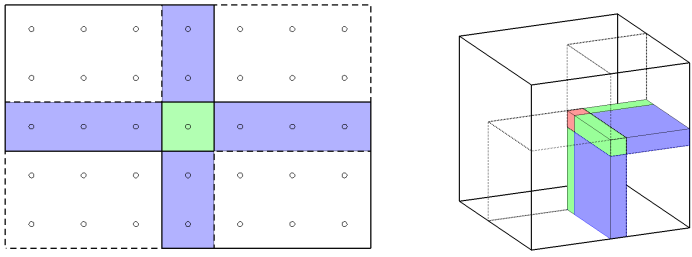
$$\begin{aligned} E_M(\boldsymbol{x}) &= \sum_{i,j \in M} c_{ij} x_{ij} - \frac{1}{2} \sum_{i,j \in M \cap N} c_{ij} x_{ij} \\ E_N(\boldsymbol{y}) &= \sum_{i,j \in N} c_{ij} y_{ij} - \frac{1}{2} \sum_{i,j \in M \cap N} c_{ij} y_{ij}. \end{aligned} \tag{7.3}$$

This leads to the following equivalent linear program:

$$\begin{aligned} \underset{\substack{\boldsymbol{x} \in \mathcal{D}_M \\ \boldsymbol{y} \in \mathcal{D}_N}}{\text{minimize}} \quad & E_M(\boldsymbol{x}) + E_N(\boldsymbol{y}) \\ \text{subject to} \quad & x_i = y_i, \quad i \in M \cap N. \end{aligned} \tag{7.4}$$

Here $\boldsymbol{x}$ is the variable belonging to the set M (left in figure 7.1b) and $\boldsymbol{y}$ belongs to N. The two variables $\boldsymbol{x}$ and $\boldsymbol{y}$ are constrained to be equal in the overlap. The dual function of this optimization problem is:

$$\begin{aligned} d(\boldsymbol{\lambda}) &= \min_{\substack{\boldsymbol{x} \in \mathcal{D}_M \\ \boldsymbol{y} \in \mathcal{D}_N}} \left(E_M(\boldsymbol{x}) + E_N(\boldsymbol{y}) + \sum_{i \in M \cap N} \lambda_i (x_i - y_i) \right) \\ &= \min_{\boldsymbol{x} \in \mathcal{D}_M} \left(E_M(\boldsymbol{x}) + \sum_{i \in M \cap N} \lambda_i x_i \right) \\ &+ \min_{\boldsymbol{y} \in \mathcal{D}_N} \left(E_N(\boldsymbol{y}) - \sum_{i \in M \cap N} \lambda_i y_i \right). \end{aligned} \tag{7.5}$$



(a) 2 × 2 (b) 2 × 2 × 2

Figure 7.2: Splitting a graph into several components. The blue, green and red parts are weighted by ½, ¼ and ⅛, respectively.

Just as in section 2.2 on page 13, this shows that evaluating the dual function d amounts to solving two independent minimum cut problems. The extra unary terms $\lambda_i x_i$ are shown in figure 7.1b. Let $\boldsymbol{x}^*, \boldsymbol{y}^*$ be the solution to (7.4) and let $\boldsymbol{\lambda}^*$ maximize d. Because strong duality holds, $d(\boldsymbol{\lambda}^*) = E_M(\boldsymbol{x}^*) + E_N(\boldsymbol{y}^*)$ (Bertsekas, 1999). Each subproblem may in general have multiple solutions—a unique solution can be chosen by setting the optimal $\boldsymbol{x}^*$ and $\boldsymbol{y}^*$ equal to 1 wherever possible.

Splitting a graph into more than two components can be achieved with the same approach. The objective functions analogous to (7.3) might then contain terms weighted by 1/4 and 1/8, depending on the geometry of the split; see figure 7.2.

7.2.3 Implementation

Solving the original problem (7.4) amounts to finding the maximum value of the dual function. It follows from Lemma 2.1 that $x_i - y_i$, for $i \in M \cap N$, is a supergradient to d. The iterative scheme described in section 2.2 can be used to maximize d. This scheme requires the dual function to be evaluated many times. This is done efficiently by reusing the search trees as described by Kohli and Torr (2007). Only a small part of the cost coefficients is changed between iterations and my experiments show that the subsequent max-flow computations can be completed within microseconds; see table 7.1.

The step size τ needs to be chosen in each iteration. One possible choice is $\tau = 1/k$, where k is the current iteration number. For this particular application, this scheme and others appearing in the literature (Bertsekas,

1999; Komodakis et al., 2007) can be a bit too conservative. Instead of using a single step length τ, each node in the overlap has its own step length τ_i. The reason for this is because different parts of the graph behave in different ways. In each iteration, λ_i should ideally force $x_i = y_i$; therefore, if $x_i - y_i$ changed sign, then the step length was too large and the next step should go in the opposite direction with a reduced length.

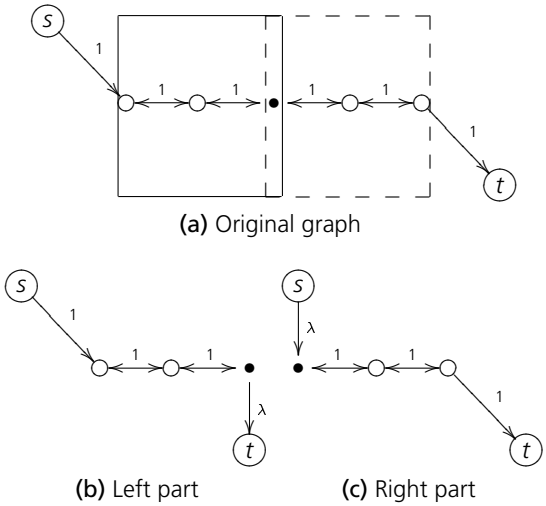
foreach $i \in M \cap N$ **do**
  if $x_i - y_i \neq 0$ **then**
    $\lambda_i \leftarrow \lambda_i + \tau_i(x_i - y_i)$
    if $x_i - y_i \neq$ *previous difference* **then**
      $\tau_i \leftarrow \tau_i/2$
    end
  end
end

To handle cases like the one shown in figure 7.8, the step length should increase if nothing happens between iterations. Empirical tests show that keeping an individual step length improves convergence speed for all graphs. The extra memory requirements are insignificant.

Convergence. Some graphs may have problems converging to the optimal solution. This can occur for graphs admitting multiple solutions. Figure 7.3 shows an illustrative example. While the proposed scheme will converge toward $\lambda = 1$ for this graph, it will not reach it in a finite number of iterations. If $\lambda \neq 1$, the two partial solutions will not agree for the node marked black. In practice, I observed this phenomenon for a few pixels when processing large graphs with integer costs.

One possible solution is to add small, positive, random numbers to the edge costs of the graph. If the graph only has integer costs, this is not a problem. Increasing edge costs only increases the maximum flow, so the global maximum flow in the original graph is the integer part of the flow in the modified graph provided the sum of all values added is less than 1. However, there is an alternative way of handling graphs with integer costs.

Floating point arithmetic is not always desirable in discrete optimization. Round-off and cancellation errors may cause a suboptimal cut to be generated. An interesting question is then whether d has an integer maximizer.



(a) Original graph

(b) Left part (c) Right part

Figure 7.3: Convergence problem. The original graph (a) has multiple solutions, i.e., multiple minimum cuts. If $\lambda \neq 1$, the solutions for the two graphs (b) and (c) will not agree.

THEOREM 7.1 *If all the edge costs* $\mathbf{c}$ *are* even *integers, then there is an integer vector* $\boldsymbol{\lambda}$ *maximizing the dual function* (7.5).

Proof. The constraint sets $\mathcal{D}_M$ and $\mathcal{D}_N$ in (7.4) can be described by $\mathsf{A}\boldsymbol{z} \geq \mathbf{b}$, where A is an (integer) matrix, $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y})$ and $\mathbf{b}$ is an (integer) vector. Therefore, the optimization problem (7.4) can be written as:

$$\begin{aligned} \underset{\boldsymbol{z}}{\text{minimize}} \quad & \mathbf{a}^{\mathrm{T}}\mathbf{z} \\ \text{subject to} \quad & \mathsf{A}\mathbf{z} \geq \mathbf{b} \\ & x_i - y_i = 0, \quad i \in M \cap N. \\ & \boldsymbol{z} \geq 0. \end{aligned} \tag{7.6}$$

Here, $\mathbf{a}$ is an integral vector describing the objective function in (7.4). Write $\mathsf{B}\boldsymbol{z} = 0$ for the equality constraint. Then, the dual function to this optimization problem is

$$\tilde{d}(\boldsymbol{w}, \boldsymbol{\lambda}) = \min_{\boldsymbol{z}} \left\{ \mathbf{a}^{\mathrm{T}}\boldsymbol{z} + \boldsymbol{w}^{\mathrm{T}}(\mathbf{b} - \mathsf{A}\mathbf{z}) + \boldsymbol{\lambda}^{\mathrm{T}}\mathsf{B}\boldsymbol{z} \right\}.$$

The dual is also a linear program,

$$\begin{aligned} \underset{\boldsymbol{w},\boldsymbol{\lambda}}{\text{maximize}} \quad & \mathbf{b}^{\mathsf{T}}\boldsymbol{w} \\ \text{subject to} \quad & \mathsf{A}^{\mathsf{T}}\boldsymbol{w} + \mathsf{B}^{\mathsf{T}}\boldsymbol{\lambda} \leq \mathbf{a} \\ & \boldsymbol{w} \geq 0. \end{aligned} \tag{7.7}$$

Since A is totally unimodular (t.um.) (see section 13.2 in the book by Papadimitriou and Steiglitz (1998)), so is A^{T}. Also, B is t.um. and $\mathbf{a}$ is integral, and thus the dual function $\tilde{d}$ has an integer optimum $(\boldsymbol{w}^*, \boldsymbol{\lambda}^*)$. Since $d(\boldsymbol{\lambda}) = \max_{\boldsymbol{w}\geq 0} \tilde{d}(\boldsymbol{w}, \boldsymbol{\lambda})$, the proof is complete. □

Remark. The theorem does not necessarily hold if the costs contain odd integers. The graph $s \xrightarrow{\infty} \circ \xrightarrow{1} \circ \xrightarrow{1} t$, split at the second node, provides a counterexample. The subproblems are $s \xrightarrow{\infty} \circ \xrightarrow{1} \circ \xrightarrow{1/2+\lambda} t$ and $s \xrightarrow{\lambda} \circ \xrightarrow{1/2} t$. Then $d(0) = d(1) = 1/2$, but $d(1/2) = 1$.

For a general graph with integer costs split into two pieces, multiplying each edge by 2 results in an equivalent problem with an integer maximizer $\boldsymbol{\lambda}$. The graph may be split in more than two pieces in such a way that smaller costs than $1/2$ are used, as in figure 7.2. These problems can be set up by multiplying every cost by 4 and 8, respectively, to ensure integer maximizers.

7.3 Experiments on a Single Machine

This section describes experiments performed in parallel on a single machine executed across multiple threads. All experiments used the BK-method (v3.0) both for the single-threaded baseline and for solving the subproblems. The timings of the multi-threaded runs include any overhead associated with starting and stopping threads, allocation of extra memory etc. The time required to construct the graphs is not taken into account. However, graph construction trivially benefits from parallel processing.

Image segmentation. The first experiment used the 301 images in the Berkeley segmentation database (Martin et al., 2001); see figure 7.4 for examples. The segmentation model was a piecewise constant model (the pixel values

were normalized to $[0, 1]$) with the boundary length as a regulating term:

$$E(\boldsymbol{x}) = \rho \sum_i \sum_{j \in \mathcal{N}(i)} w_{ij} |x_i - x_j| + \sum_i x_i \Big((I_i - c_1)^2 - (I_i - c_0)^2 \Big). \tag{7.8}$$

The boundary length is here approximated using a neighborhood $\mathcal{N}(i)$ of edges around each pixel, usually of sizes 4, 8 or 16 in the two-dimensional case. See (Boykov and Kolmogorov, 2003; Kolmogorov and Boykov, 2005) for details on how to choose w_{ij}. The overall influence of the boundary length is specified with the parameter ρ, where larger values usually correspond to harder optimization problems (longer s/t paths). The two constants c_1 and c_0 specify the intensity of the foreground and background. They were estimated in an alternating EM fashion for each image separately before the experiments.

The relative times ($t_{\text{multi-thread}}/t_{\text{single}}$) using two computational threads are shown in figures 7.5 and 7.6. Since the images in the database are quite small, the total processing time for a single image is around 10 milliseconds. Even with the overhead of creating threads and iterating to find the global minimum, almost all images saw a significant speed improvement.

Table 7.1 shows how the processing time varies with each iteration. In the last steps, very few vertices change and solving the maximum flow problems can therefore be done very quickly within microseconds.

It is very important to note that the problem complexity depends heavily on the amount of regularization used. That is, segmentation problems with low penalties on boundary length are easy to solve and parallelize. In the extreme case where no regularization is used, the problem reduces to simple thresholding, which of course is trivial to parallelize. Therefore, it is relevant to investigate how the algorithm performs with different amounts of regularization. The graph decomposition scheme in this chapter performs well for a wide range of different settings; see figure 7.7. The relative improvement in speed remains roughly constant over a large interval of different regularizations, whereas the absolute processing times vary by an order of magnitude.

When the number of computational threads increase, the computation times decrease as shown in figure 7.5.

Cameraman (256 × 256)

Iteration	1	2	3	4	5
Number of Differences	36	17	3	3	0
Time (ms)	6.8	0.141	0.078	0.071	0.447

Tree (1152 × 1536, shown in figure 7.7)

Iteration	1	2	3	4	···	8	9	10	11
Differences	108	105	30	33	···	16	9	9	0
Time (ms)	245	1.5	1.2	0.1	···	0.15	0.06	0.07	0.47

Table 7.1: The processing time for each iteration for two example images. The number of overlapping pixels ($M \cap N$) was 256 and 1536, respectively (one column). Deallocating memory and terminating threads is the cause of the processing time increase in the last iteration. Note the short processing times after the first iteration due to the reuse of search trees.

Figure 7.4: Examples from the Berkeley database (Martin et al., 2001).

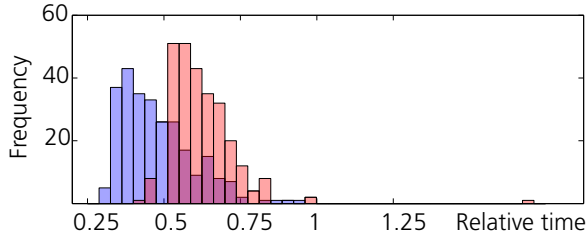


Figure 7.5: Relative times with 2 (red) and 4 (dotted blue) computational threads for the 301 images in the Berkeley segmentation database, using 4-connectivity. The medians are 0.596 and 0.455.

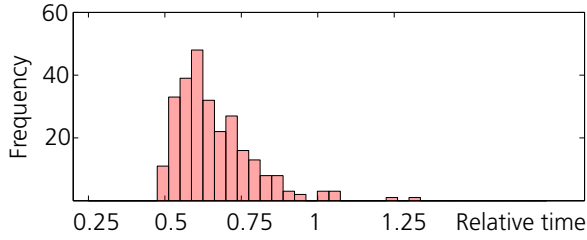


Figure 7.6: Relative times using 8-connectivity and 2 computational threads. The median is 0.628.

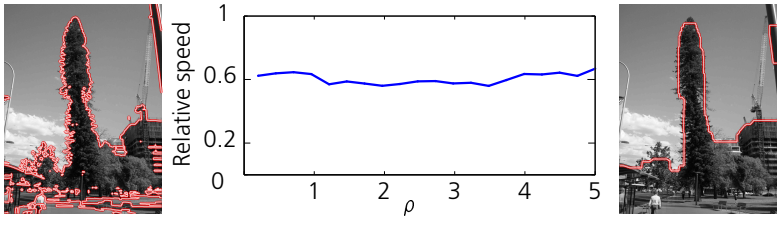


Figure 7.7: Relative improvement in speed with two computational threads when the regularization parameter changes. Although the processing time ranged from 230 ms to 4 seconds, the relative improvement was not affected.

Stereo problems. The "Tsukuba" data set, which can be obtained from University of Western Ontario (UWO), consists of a sequence of max-flow instances corresponding to the first iteration of α-expansion. See section 3.6 on page 29 and also the paper by Boykov et al. (2001) for a more thorough explanation. I solved the 16 problems first without any parallelization and then, using two computational threads. The relative times ranged from 0.51 to 0.72, with the average being 0.61.

Three-dimensional graphs. Lempitsky and Boykov (2007) use a 3D graph construction to fit a surface to a point cloud. The graphs are regular, 6-connected and are suitable tests for the algorithm in three dimensions. Data is available for download at UWO. For the "bunny" data set the relative time was 0.67 with two computational threads.

Limitations. It is interesting to see how the algorithm performs when the choice of split is very poor. Figure 7.8 shows an image split it in half from top to bottom. The leftmost pixel column is attached to the source and the

Figure 7.8: "Worst-case" test. The left and right side of the image is connected to the source and sink, respectively. The edge costs are determined by the image gradient. The entire flow must be communicated between the two computational threads when splitting the graph vertically.

rightmost to the sink. Splitting horizontally would have been much more preferable, since splitting vertically severs every possible s-t path and all flow has to be communicated between the threads. Still, the parallel approach finished processing the graph 30% *faster* than the single-threaded approach. This is a good indication that the choice of the split is not crucial.

Figures 7.5 and 7.6 contain a few examples ($< 1\%$) where the multi-threaded algorithm actually performs slower or almost the same as the single-threaded algorithm. The single example in figure 7.5 is interesting, because solving one of the subgraphs *once* takes significantly *longer* than solving the entire original graph. This can happen for the BK algorithm, but is very uncommon in practice. I have noted that slightly perturbing any of the problem parameters (regularization, image model, split position etc.) makes the multi-threaded algorithm faster also for this example.

The other slow examples have a simpler explanation: there is simply nothing interesting going on in one of the two halves of the graph, see e.g. the first image in figure 7.4. Therefore, the overhead of creating and deallocating the threads and extra memory gives the multi-threaded algorithm a slight disadvantage. The approach by Liu and Sun (2010) (using smaller pieces) is better suitable for these graphs.

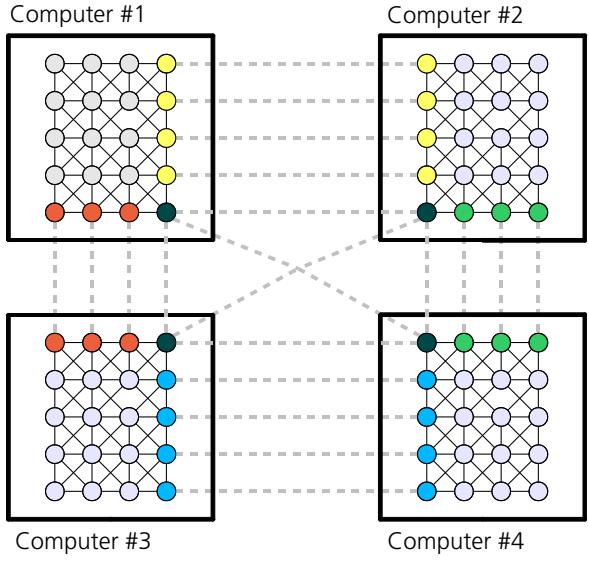


Figure 7.9: Splitting a graph across many computers. Colors indicate nodes that should be equal.

7.4 Splitting across Different Machines

There exist other, large-scale, applications of graph decomposition. Instead of assigning each part of the graph to a computational thread, one may assign each subgraph to a different machine and let the machines communicate the flow over a network. Figure 7.9 shows a diagram of the setup.

Memory is often a limiting factor for maximum flow calculations. Splitting the graph made segmenting 4-dimensional (space + time) MRI heart data with $95 \times 98 \times 30 \times 19 = 5.3$ M voxels possible. The connectivity used was 80, requiring 12.3 GB memory for the graph representation. By dividing this graph among four (two by two) different machines and using MPI for communication (Snir and Otto, 1998), the graph was solved in 1,980 seconds. Since only a small amount of data (54 kB in this case) needs to be transmitted between machines each iteration, this is an efficient way of processing large graphs. On the experimental system[1], the communication time was about 7–10 ms per iteration, for a total of 68 iterations until convergence.

[1] LUNARC Iris, http://www.lunarc.lu.se/Systems/IrisDetails

I have also evaluated the algorithms for some of the big problems available from the University of Western Ontario. The largest version of the "bunny" data set is $401 \times 396 \times 312 = 50$ M with 300 M edges was solved in 7 seconds across 4 machines. As a reference, a slightly larger version of the same data set (not publicly available) was solved in over a minute with an (iterative) touch-and-expand approach in (Lempitsky and Boykov, 2007).

The largest data set was a $512 \times 512 \times 2{,}317 = 607$ M voxel CT scan with 6-connectivity. Storing this graph required 131 GB of memory divided among 36 ($3 \times 3 \times 4$) machines. The regularization used was low, which ensured convergence in 38 seconds with fairly even load distribution. Even with low regularization, the computation required 327 iterations.

Splitting graphs across multiple machines also saves computation time, even though the MPI introduces some overhead. For the small version of the "bunny" data set, a single machine solved the problem in 268 milliseconds, while two machines used 152 ms. Four machines (two by two) required 105 ms. For the medium sized version the elapsed times were 2.3, 1.34 and 0.84 seconds, respectively.

It should be noted that in many cases the BK algorithm is not the fastest possible choice, especially for graphs with higher dimensionality than 2 and connectivity greater than the minimum (Boykov and Kolmogorov, 2004). However, the method described in this chapter could just as easily be combined with a push-relabel algorithm better suited for graphs with 3 or 4 dimensions. Using a method optimized for grid graphs with fixed connectivity instead of the general BK would also reduce memory requirements significantly.

7.5 Conclusion

This chapter has shown that it is possible to split a graph and obtain the global maximum flow by iteratively solving subproblems in parallel. Two applications of this technique were demonstrated:

- Faster maximum flow computations when multiple CPU cores are available (section 7.3).

- The ability to handle graphs which are too big to fit in the computer's RAM, by splitting the graph across multiple machines (section 7.4).

> This is in contrast to other approaches (Liu and Sun, 2010; Liu et al., 2009) where shared memory is required.

Methods based on push-relabel generally perform better than BK for large, high dimensional and highly connected graphs as discussed in (Boykov and Kolmogorov, 2004). Therefore, using this approach with push-relabel should be investigated in the future.

The work in this chapter is based on a publication from 2010 (Strandmark and Kahl, 2010*b*). Since then, others have continued the work on parallelization of discrete optimization algorithms in computer vision. Chapter 15 contains a brief summary of these developments.

Chapter 8

Parallel Inference on a GPU

I will now shift the focus from two-label problems to multi-label segmentation problems. The goal is to use dual decomposition for a highly parallel algorithm which can be run on hardware such as a field-programmable gate array (FPGA) or a graphics processing unit (GPU). The objective function to be minimized is

$$E(\boldsymbol{x}) = \sum_{i=1}^{n} T_i(x_i) + \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i} E_{ij}(x_i, x_j), \tag{8.1}$$

where $\boldsymbol{x} \in \mathbf{L} = \{1 \dots L\}^n$ and the functions T_i and E_{ij} are arbitrary. The set $\mathcal{N}_i$ is the neighborhood of i, that is, all nodes that are directly dependent on i. This chapter focuses on 4-connectivity, but the ideas are able to handle higher connectivities.

While this chapter does not offer any new theoretical insights, I will offer a practical way to parallelize previous approaches with good performance. The method uses dual decomposition and dynamic programming similar to Komodakis et al. (2007), but the subproblems are simpler and therefore more easily solved on massively parallel architectures. The potential for parallelization was not explored by Komodakis et al. and this chapter will show that these methods are useful to greatly increase the speed of algorithms for which other fast methods already are available, such as minimum cut problems.

Graphical processing units (GPUs) have been used extensively to facilitate the often very computationally expensive tasks in low-level vision. Segmentation with total variation models (e.g. problem (Q) in chapter 9 on page 120) has successfully been implemented and can be performed in real time (Pock et al., 2008; Unger et al., 2008). Computing the optical flow between two images (Werlberger et al., 2009) and stereo estimation have also benefited greatly from the parallelization offered by a multi-processor GPU.

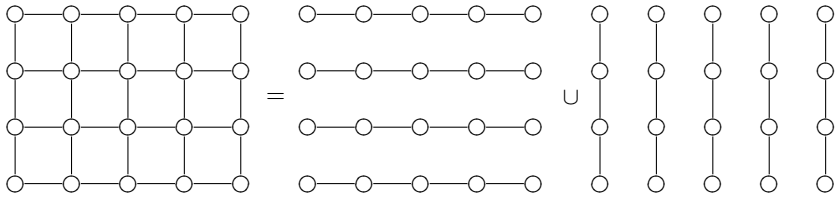


Figure 8.1: The graph decomposition. An edge between two nodes i and j indicate that $j \in \mathcal{N}_i$ so that the two nodes are directly dependent on each other. Note that the two subproblems each consists of many independent one-dimensional problems.

Solving discrete labeling problems such as minimum cuts and its generalizations on a GPU has proven to be harder. Often the speed-up is not great compared to algorithms that run on a CPU or the method might work well for only a restricted amount of regularization. In my experience, existing approaches do not work well, as I mentioned in the previous chapter on page 85 about push-relabel. Figure 8.3 shows one of my experiments.

8.1 Splitting the Graph

The first thing to do is to split the objective function E in (8.1). The approach is simple: instead of considering a single graph, I consider two graphs, one of which contains all vertical connections and one contains all horizontal connections. The two new objective functions are

$$\begin{aligned} E_1(\boldsymbol{x}) &= \frac{1}{2}\sum_{i=1}^{n} T_i(x_i) + \sum_{i=1}^{n}\sum_{j\in\mathcal{H}_i} E_{ij}(x_i, x_j), \\ E_2(\boldsymbol{x}) &= \frac{1}{2}\sum_{i=1}^{n} T_i(x_i) + \sum_{i=1}^{n}\sum_{j\in\mathcal{V}_i} E_{ij}(x_i, x_j), \end{aligned} \tag{8.2}$$

where $\mathcal{H}_i$ and $\mathcal{V}_i$ denotes the horizontal and vertical neighbors of i, respectively. The factor ½ is needed to have $E(\boldsymbol{x}) = E_1(\boldsymbol{x}) + E_2(\boldsymbol{x})$. Figure 8.1 illustrates this split of the objective function.

After splitting the graph, the next step is to solve the dual problem (D) on page 14. Not only are the two subproblems independent, but each also

consists of many one-dimensional subproblems, completely independent of each other. Such (acyclic) one-dimensional problems can always be solved exactly in polynomial time using dynamic programming.

8.2 Dynamic Programming

It is well-known that one-dimensional functions of the type (8.1) can be minimized exactly. The one-dimensional version of (8.1) is

$$E(\boldsymbol{x}) = \sum_{i=1}^{n} T_i(x_i) + \sum_{i=1}^{n-1} E_i(x_i, x_{i+1}). \tag{8.3}$$

To solve this problem, define $C_k(y)$ to be the lowest objective value possible when assigning $x_k = y$, and only counting the variables up to k. That is:

$$C_k(y) \equiv \min_{x_1 \ldots x_k = y} \left(\sum_{i=1}^{k} T_i(x_i) + \sum_{i=1}^{k-1} E_i(x_i, x_{i+1}) \right). \tag{8.4}$$

The following lemma describes how to compute $C_k(y)$ recursively over k:

LEMMA 8.2

$$C_k(y) = \begin{cases} T_1(y) & k = 1 \\ T_k(y) + \min_{z \in \mathbf{L}} \Big(C_{k-1}(z) + E_{k-1}(z, y) \Big) & k > 1. \end{cases}$$

Proof. The first case $C_1(y) = T_1(y)$ is trivial. For any $k > 1$,

$$\begin{aligned} C_k(y) &= \min_{x_1 \ldots x_k = y} \left(T_k(y) + E_{k-1}(x_{k-1}, y) + \sum_{i=1}^{k-1} T_i(x_i) + \sum_{i=1}^{k-2} E_i(x_i, x_{i+1}) \right) \\ &= T_k(y) + \min_{x_1 \ldots x_{k-1} = z} \left(E_{k-1}(z, y) + \sum_{i=1}^{k-1} T_i(x_i) + \sum_{i=1}^{k-2} E_i(x_i, x_{i+1}) \right) \\ &= T_k(y) + \min_{z \in \mathbf{L}} \Big(C_{k-1}(z) + E_{k-1}(z, y) \Big). \end{aligned}$$

□

Lemma 8.2 gives an algorithm to efficiently compute $C_k(y)$ for all nodes k and labels y. An optimal labeling $\boldsymbol{x}^*$ can be extracted from this information. The value of x_k can be computed recursively given x^*_{k+1}.

LEMMA 8.3 *The optimum* $\boldsymbol{x}^*$ *to* (8.3) *satisfies*

$$\boldsymbol{x}_k^* = \begin{cases} \underset{y \in \mathbf{L}}{\arg\min}\ C_n(y) & k = n \\ \underset{y \in \mathbf{L}}{\arg\min}\ C_k(y) + E_k(y, \boldsymbol{x}_{k+1}^*). & k < n \end{cases}$$

Proof. The case $k = n$ is the definition of $C_n(y)$. If $k < n$,

$$\begin{aligned}
\boldsymbol{x}_k^* &= \underset{y \in \mathbf{L}}{\arg\min} \quad \min_{\boldsymbol{x} \in \mathbf{L},\, x_k = y} E(\boldsymbol{x}) \\
&= \underset{y \in \mathbf{L}}{\arg\min} \quad \min_{\boldsymbol{x} \in \mathbf{L},\, x_k = y} \left(\sum_{i=1}^{n} T_i(x_i) + \sum_{i=1}^{n-1} E_i(x_i, x_{i+1}) \right) \\
&= \underset{y \in \mathbf{L}}{\arg\min} \quad \min_{\boldsymbol{x} \in \mathbf{L},\, x_k = y} \Bigg(\sum_{i=1}^{k} T_i(x_i) + \sum_{i=1}^{k-1} E_i(x_i, x_{i+1}) \\
&\quad + \sum_{i=k+1}^{n} T_i(\boldsymbol{x}_i^*) + \sum_{i=k}^{n-1} E_i(\boldsymbol{x}_i^*, \boldsymbol{x}_{i+1}^*) \Bigg) \\
&= \underset{y \in \mathbf{L}}{\arg\min} \quad \Bigg(C_k(y) + \sum_{i=k+1}^{n} T_i(\boldsymbol{x}_i^*) \\
&\qquad\qquad + E_k(y, \boldsymbol{x}_{k+1}^*) + \sum_{i=k+1}^{n-1} E_i(\boldsymbol{x}_i^*, \boldsymbol{x}_{i+1}^*) \Bigg) \\
&= \underset{y \in \mathbf{L}}{\arg\min}\ C_k(y) + E_k(y, \boldsymbol{x}_{k+1}^*).
\end{aligned}$$

□

Lemmas 8.2 and 8.3 together give an efficient method of minimizing $E(\boldsymbol{x})$. The time required for arbitrary objective functions will be quadratic in the number of labels, which makes the method described in this chapter prohibitively slow for problems with a huge number of labels, such as problems arising in (Pritch et al., 2009).

8.3 Boolean Formulation and Updating of Weights

The problem formulation of minimizing (8.1) such that $\boldsymbol{x} \in \mathcal{L}$ imposes an ordering on the labels. This is because a supergradient to d is $\boldsymbol{x}^* - \boldsymbol{y}^*$

(see Lemma 2.1 on page 14) and this difference depends on the numerical values of the labels. To avoid this dependence, reformulate the optimization problem as a boolean problem:

$$\hat{E}(\hat{\boldsymbol{x}}) = \sum_{i=1}^{n} \sum_{\ell \in \mathbf{L}} \hat{x}_{\ell,i} \cdot T_i(\ell) + \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i} \sum_{\ell_1 \in \mathbf{L}} \sum_{\ell_2 \in \mathbf{L}} \hat{x}_{\ell_1,i} \cdot \hat{x}_{\ell_2,j} \cdot E_{ij}(\ell_1, \ell_2), \tag{8.5}$$

where now $\hat{x}_{\ell,i} \in \{0,1\}$ and $\hat{x}_{\ell,i} = 1 \iff x_i = \ell$. Minimizing $\hat{E}$ is clearly equivalent to minimizing E. The constraint $x = y$ is reformulated as $\hat{x}_\ell = \hat{y}_\ell$, $\ell \in \mathbf{L}$ and each of these is associated with a dual variable $\hat{\lambda}_\ell$. The supergradient is according to Lemma 2.1 on page 14

$$(\nabla \hat{d}(\hat{\lambda}_1, \ldots, \hat{\lambda}_\ell))_{\ell,i} = \hat{x}_{\ell,i} - \hat{y}_{\ell,i}. \tag{8.6}$$

If the two solutions disagree for node i, the dual variables will then have to be updated accordingly:

$$\hat{\lambda}_{\ell,i} \leftarrow \hat{\lambda}_{\ell,i} + \tau(\hat{x}_{\ell,i} - \hat{y}_{\ell,i}), \tag{8.7}$$

where τ is the step length for the current iteration. Since the dual variables enter as multiplicative constants in front of $\{0,1\}$-variables, adding a number c to $\lambda_{\ell,i}$ is equivalent to adding c to $T_i(\ell)$. Therefore, the dual variables need not be stored explicitly and the data terms are instead modified directly.

8.3.1 Step Lengths

The step length rule I found to work best in practice was simply $\tau = C/k$, where k is the iteration number. To allow for data terms of different magnitudes, the step lengths can be normalized to $\tau = m/(3k)$, where m is the maximum value of the data terms. If the data terms contain hard constraints with infinite cost weights, these will have to be excluded. I have also tried the primal-dual based rule used by Komodakis et al. (2007), but in this context it was slightly inferior. I discussed different step size schemes in section 2.2.1 on page 15 and mentioned that no method working with supergradients alone can be very effective in general, and the problems in this chapter are very general indeed.

8.4 Linear Programming Relaxation

A very relevant question is whether the decomposed problem will solve the original problem. In general, this is too much to hope for, but how good will the solution be? This is answered by Komodakis et al. (2011). The combinatorial minimization problem of minimizing E can be relaxed to a linear programming problem. Problem (3.13) on page 25 is such a relaxation for the case $\mathcal{L} = \{0, 1\}$. For the one-dimensional subproblems the relaxation is always tight, and this fact can be used to show that the optimal value of the decomposed problem is equal to the value of the relaxed linear program. This is very similar to what was used in the previous chapter to prove that dual decomposition always converges to the global optimum in the submodular boolean case.

8.5 Experiments

All CPU experiments were performed with an Intel Core2 Quad 2.5 GHz processor (using a single core only) and the GPU experiments used an Nvidia Tesla 2050. The iterations continued until the computed relative duality gap was smaller than 0.001. Results are given for the standard Potts model for which the regularizer is simply

$$E_{ij}(\ell_1, \ell_2) = \begin{cases} \rho & \ell_1 \neq \ell_2 \\ 0 & \ell_1 = \ell_2 \end{cases}. \tag{8.8}$$

If the number of labels is $L = 2$, the exact solution can be computed as the minimum cut in an appropriate graph. If $L \geq 3$, two approximate methods provide good comparisons: α-expansion (section 3.6 on page 29) and FastPD (Komodakis and Tziritas, 2007). The results are given in tables 8.1–8.3 as well as Figures 8.2–8.5. Comparing to CUDA cuts (Vineet and Narayanan, 2008) was not possible—the publicly available algorithm did not give the correct solution for any of the problems in figure 8.2. For $\rho = 10^5$ and 10^6 the algorithm produced a constant image. There are other methods available, e.g. TRW-S which are not included, but other researchers (Alahari et al., 2010; Carr and Hartley, 2009) have made comparisons to these methods.

The data terms for the three-class experiments (table 8.3 and figure 8.5) were created by estimating a Gaussian mixture model using the standard

ρ	BK time	CPU time	GPU time	Rel. duality gap	# iter.
10^4	0.0108s	0.0691s	0.042s	0.00024	71
10^5	0.0204s	0.0166s	0.01s	0.00017	16
10^6	0.4973s	0.0330s	0.017s	0.00046	26

Table 8.1: Boolean segmentation results for the "cameraman" image in figure 8.2. This table compares the BK algorithm (Boykov and Kolmogorov, 2004, v.3.01) to a CPU and GPU implementation of the algorithm described in this chapter. The dual decomposition algorithm seems to perform best using high regularizations.

ρ	Images	CPU Relative speed			CPU Absolute time (s)
		Min.	Median	Max.	Median
10^4	300	0.45	3.99	9.13	0.0120
$5 \cdot 10^4$	293	0.43	2.50	8.69	0.0240
10^5	279	0.35	1.55	4.18	0.0373
$5 \cdot 10^5$	163	0.07	0.31	1.77	0.1548
10^6	95	0.04	0.13	0.77	0.3483

Table 8.2: Binary segmentation results using (7.8) on page 93 for the images in the Berkeley segmentation data set (CPU, single thread). If the result was a trivial (constant) segmentation, that image was excluded; therefore, the lower number of images for higher regularization. Two facts can be seen from these numbers: (i) that higher regularization results in far more difficult problems and that (ii) the row/column decomposition performs much better for those problems and those problems only.

expectation maximization algorithm. However, the figures in tables 8.2 and 8.3 do not exploit the possibility of massive parallelization, making dual decomposition a very attractive alternative, provided that the architecture can support the execution of many dynamic programming threads. Unlike in the previous chapter (fig. 7.7 on page 95), a speed-up is present mainly for problems where the amount of regularization is high.

When the number of labels increases, the time until convergence increases drastically. This is illustrated in figure 8.4. Furthermore, for general objective functions, the processing time required for a single iteration increases quadratically with the number of labels. These two facts make the method unattractive for a large number of labels and inferior to e.g. α-expansion.

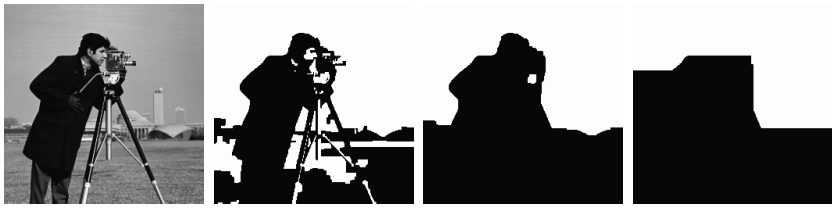


Figure 8.2: Pseudo-boolean segmentation with different regularizations $\rho \in \{10^4, 10^5, 10^6\}$. See table 8.1.

Figure 8.3: Result using CUDA cuts (Vineet and Narayanan, 2008) with $\rho = 10^4$. The correct solution is shown in figure 8.2. For $\rho = 10^5$ and 10^6 the algorithm produces a constant image.

Figure 8.4: Result using the Potts model with 64 labels. To the left the solution obtained by α-expansion is shown, followed by the row and column solutions of figure 8.1 after 500 iterations. Running 10000 iterations did not improve the solutions significantly.

ρ	Decomposition (s)			FastPD (s)			α-expansion (s)		
	Min.	Med.	Max.	Min.	Med.	Max.	Min.	Med.	Max.
2	0.02	0.27	3.06	0.13	0.15	0.33	0.22	0.34	0.76
10	0.04	0.27	2.93	0.13	0.16	0.39	0.22	0.38	1.08
100	0.02	0.23	4.15	0.14	0.32	1.46	0.16	0.64	2.67

Table 8.3: Multi-label segmentation with the Potts model for the 300 images in the Berkeley data set. All three methods produced virtually indistinguishable results. The decomposition method was run single-threadedly, i.e. the possibility of massive parallelization was not used.

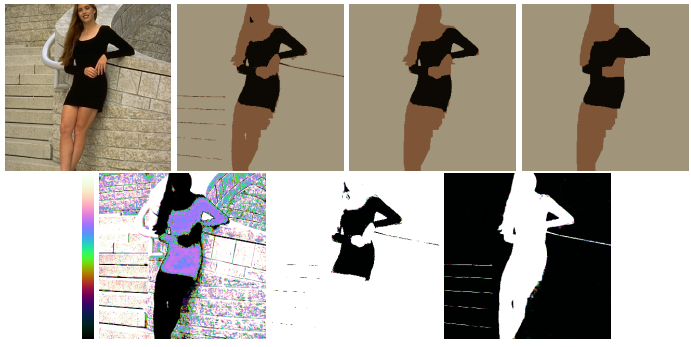


Figure 8.5: Three-class example. Top row: Original image and segmentations with the Potts model for different regularizations $\rho \in \{2, 10, 100\}$, respectively. Bottom row: data terms. See table 8.3.

8.6 Coordinate Ascent

I will end this chapter with a comment on the shape of the dual function d in the boolean case where $\boldsymbol{x} \in \{0,1\}^n$. The goal is to maximize d. For this purpose, let us consider the dual function as a function of only one component of $\boldsymbol{\lambda}$, say corresponding to x_j, and keep all other components fixed: $\lambda_i = \mu_i$ if $i \neq j$. We can then drop the subscript and write $\lambda_j = \lambda$. It is natural to consider the two parts of the dual function separately by introducing the two functions

$$\begin{aligned} d_1(\boldsymbol{x}, \lambda) &= E_1(\boldsymbol{x}) + \sum_{i \neq j} \mu_i x_i + \lambda x_j \\ d_2(\boldsymbol{y}, \lambda) &= E_2(\boldsymbol{y}) - \sum_{i \neq j} \mu_i y_i - \lambda y_j. \end{aligned} \tag{8.9}$$

The dual function is then the sum of the two functions

$$\begin{aligned} d_1(\lambda) &= \min_{\boldsymbol{x} \in \{0,1\}^n} d_1(\boldsymbol{x}, \lambda) \\ d_2(\lambda) &= \min_{\boldsymbol{y} \in \{0,1\}^n} d_2(\boldsymbol{y}, \lambda). \end{aligned} \tag{8.10}$$

What do d_1 and d_2 look like? In fact, they are quite simple: Figures 8.6a and 8.6b show the shape of these functions and the following lemmas prove that this is the case.

LEMMA 8.4 *d_1 is increasing.*

Proof. Let $\lambda' \geq \lambda$. Then $d_1(\lambda) \leq d_1(\boldsymbol{x}', \lambda) \leq d_1(\boldsymbol{x}', \lambda')$ for any $\boldsymbol{x}'$. Taking the minimum gives $d_1(\lambda) \leq d_1(\lambda')$. □

LEMMA 8.5 *Let $\lambda' > \lambda$. If $x_j = 0$ for* some *solution $\boldsymbol{x}$ to the minimization problem $d_1(\lambda)$, then*

(a) $x'_j = 0$ for all *solutions $\boldsymbol{x}'$ to the problem $d_1(\lambda')$ and*

(b) $d_1(\lambda) = d_1(\lambda')$.

Proof. Let $\boldsymbol{x}$ be a solution to $d_1(\lambda)$ with $x_j = 0$. To prove statement (a), assume that there is a solution $\boldsymbol{x}'$ to $d_1(\lambda')$ with $x'_j = 1$. Then $d_1(\boldsymbol{x}, \lambda) = d_1(\boldsymbol{x}, \lambda') \geq d_1(\boldsymbol{x}', \lambda') > d_1(\boldsymbol{x}', \lambda)$, which is a contradiction to the optimality of $\boldsymbol{x}$.

To prove (b), observe that $d_1(\lambda) = d_1(\boldsymbol{x}, \lambda) = d_1(\boldsymbol{x}, \lambda') \geq d_1(\lambda')$. Since d_1 is increasing this establishes that $d_1(\lambda) = d_1(\lambda')$. □

LEMMA 8.6 *Let $\lambda' < \lambda$. If $x_j = 1$ for* some *solution $\boldsymbol{x}$ to $d_1(\lambda)$, then*

(a) $x'_j = 1$ for all *solutions $\boldsymbol{x}'$ to $d_1(\lambda')$ and*

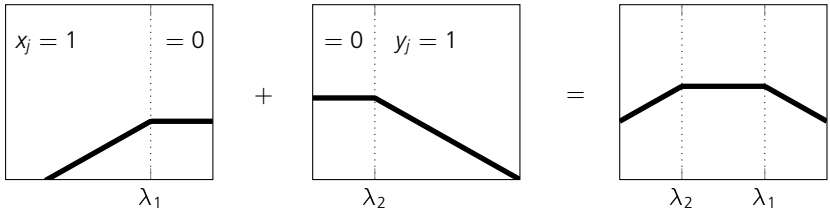
(b) $d_1(\lambda') = d_1(\lambda) - (\lambda - \lambda')$.

Proof. The proof proceeds in the same manner as in the previous lemma. Let the solutions to $d_1(\lambda)$ and $d_1(\lambda')$ be $\boldsymbol{x}$ and $\boldsymbol{x}'$, with $x_j = 1$ and $x'_j = 0$. Then $d_1(\boldsymbol{x}', \lambda) = d_1(\boldsymbol{x}', \lambda') \leq d_1(\boldsymbol{x}, \lambda') < d_1(\boldsymbol{x}, \lambda)$, which is a contradiction to the optimality of $\boldsymbol{x}$.

For (b), $d_1(\lambda) - (\lambda - \lambda') = d_1(\boldsymbol{x}, \lambda) - (\lambda - \lambda') = d_1(\boldsymbol{x}, \lambda') \geq d_1(\lambda')$. On the other hand, $d_1(\lambda) - (\lambda - \lambda') \leq d_1(\boldsymbol{x}', \lambda) - (\lambda - \lambda') \leq d_1(\boldsymbol{x}', \lambda')$ for any $\boldsymbol{x}' \in \{0, 1\}^n$. Taking the minimum gives the reverse inequality. □

The previous lemmas show that d_1 has at most one breakpoint and is constant for values larger than the breakpoint and increasing with slope 1 for values smaller. It is now easy to see that there must be exactly one breakpoint, by letting $\lambda \to \pm\infty$. Figure 8.6a shows the function $d_1(\lambda)$. The function d_2 is of course analogous, as shown in figure 8.6b. In this case:

LEMMA 8.7 *d_2 is decreasing.*



(a) $d_1(\lambda)$ (b) $d_2(\lambda)$ (c) $d_1(\lambda) + d_2(\lambda)$

Figure 8.6: The dual function value as a function of one of the components in $\boldsymbol{\lambda}$.

LEMMA 8.8 *Let $\lambda' < \lambda$. If $y_j = 0$ for* some *solution to $d_2(\lambda)$, then $y'_j = 0$ for* all *solutions to $d_2(\lambda')$.*

LEMMA 8.9 *Let $\lambda' > \lambda$. If $y_j = 1$ for* some *solution to $d_2(\lambda)$, then $y'_j = 1$ for* all *solutions to $d_2(\lambda')$.*

LEMMA 8.10 *Let $\lambda' \leq \lambda$. If $\boldsymbol{y}$ is a solution to $d_2(\lambda)$ with $y_j = 0$, then $d_2(\lambda) = d_2(\lambda')$.*

LEMMA 8.11 *Let $\lambda' \geq \lambda$. If $\boldsymbol{y}$ is a solution to $d_2(\lambda)$ with $y_j = 1$, then $d_2(\lambda') = d_2(\lambda) - (\lambda' - \lambda)$.*

The full dual function $d(\lambda) = d_1(\lambda) + d_2(\lambda)$ must be constant between the two breakpoints λ_1 and λ_2 and have slope ± 1 outside this interval; see figure 8.6c. This holds both if $\lambda_1 > \lambda_2$ or the other way around. This gives an ascent method for the dual function. The breakpoints of d are easily obtained from the dynamic programming scheme seen previously in this chapter.

8.7 Conclusion

The method of splitting the graph used in the previous chapter is not suitable for massive parallelization. If a lot of parallelism is desired, the graph needs to be split in another way, preserving some of the long paths of the original graph. This chapter has proposed such a decomposition, with the additional benefit of being able to find approximate solutions for arbitrary objective functions and number of labels. While the idea of tree decompositions is not new, the possibilities for parallelism and GPU implementation have not

been previously investigated. However, GPU parallelization of minimum cut algorithms for vision is still an unsolved problem in general. Implementing the dynamic programming schemes on a massively parallel architecture is non-trivial.

Part II

Parametric Models

Chapter 9

Optimizing Parametric Total Variation Models

In the introduction, I presented the segmentation problem as finding an optimal curve γ enclosing a region $\Gamma \subseteq \Omega$. Alternatively, this can be formulated as finding a function $\theta : \Omega \to \{0, 1\}$, such that $\theta(\boldsymbol{x}) = 1 \iff \boldsymbol{x} \in \Gamma$. The optimal function then defines the region of interest directly.

The data term (1.4) on page 4 is constructed from an observation model, which normally contains one or several unknown parameters that either must be known a priori or estimated together with the segmentation itself. This estimation is a minimization problem, where the optimal parameters depend on the proposed segmentation $\hat{\theta}$. This chapter investigates special cases of these functionals, where the image to be estimated is binary and the number of unknown parameters is relatively small.

9.1 The Mumford-Shah Functional

The functional introduced by Mumford and Shah (1989) is a widely used functional for image segmentation. As a special case, Chan and Vese (2001) proposed a segmentation method where an image is approximated with a function taking only two values. By minimizing an objective function consisting of a smoothness term added to the squared distance between the original and the approximation, a large variety of images can be segmented correctly. However, the exact minimization of this functional is a difficult problem and this chapter will describe new results on this topic obtained by using recent works by Chambolle (2005) and Chan et al. (2006).

Without loss of generality the image $I : \mathbf{R}^2 \supset \Omega \to \mathbf{R}$ is assumed to

take values in $[0, 1]$. The main contribution of this chapter is a method to evaluate real-valued functions of the following type:

$$m(\boldsymbol{t}) = \min_{\theta, s} E(\theta, s, \boldsymbol{t}), \tag{9.1}$$

where E is a functional depending on the binary valued function $\theta : \Omega \to \{0, 1\}$, the one-dimensional parameter $s \in \mathbf{R}$ as well as some additional vector of real parameters $\boldsymbol{t}$. The ability to evaluate such functions allows us to efficiently optimize parametric, binary total variation models including several variants of the Mumford-Shah functional. The standard way of solving such problems is by alternating optimization:

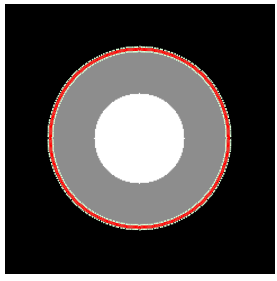
1. Keep the real parameters $s, \boldsymbol{t}$ fixed and solve for θ.
2. Keep θ fixed and solve for the real parameters $s, \boldsymbol{t}$.

By including one additional parameter in the first step, the neighborhood search is enlarged and the risk of getting trapped in local minima is reduced.

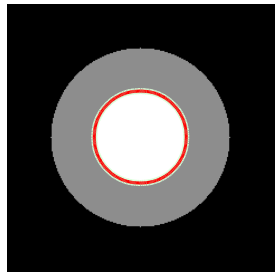
Another consequence of (9.1) is the possibility of obtaining globally optimal solutions to low-order parametric total variation models. One of the primary problems in this class of segmentation models is the Chan-Vese model, in which the image is approximated with a function taking only two values, μ_0 and μ_1, by solving the following optimization problem:

$$\begin{aligned} \underset{\theta, \mu_0, \mu_1}{\text{minimize}} \quad & \rho J(\theta) \\ & + \int_\Omega (1 - \theta(\boldsymbol{x}))(I(\boldsymbol{x}) - \mu_0)^2 + \theta(\boldsymbol{x})(I(\boldsymbol{x}) - \mu_1)^2 \, d\boldsymbol{x} \\ \text{subject to} \quad & \theta(\boldsymbol{x}) \text{ binary} \\ & 0 \leq \mu_0 < \mu_1 \leq 1. \end{aligned} \tag{9.2a}$$

Here $J(\theta)$ is the total variation of θ, $J(\theta) = \int_\Omega |\nabla \theta| \, d\boldsymbol{x}$. When θ is binary, the total variation is the length of the boundary between the two regions defined by θ. The weight $\rho > 0$ controls how important a short boundary is. The assumption that $\mu_1 > \mu_0$ is without loss of generality and it prevents (9.2a) from inherently having two optima. Section 9.3 will show how to find the optimal segmentation as well as the optimal values of the two parameters μ_0 and μ_1 by a simple branch and bound search over a single dimension.



(a) $\mu_1 = 0.7$, $\mu_0 = 0$, $E = 615$



(b) $\mu_1 = 1$, $\mu_0 = 0.2$, $E = 600$

Figure 9.1: Segmenting a simple image. The result shown in (a) was obtained after setting $\mu_0 = 0$, $\mu_1 = 1$ and alternating between minimizing θ and updating μ_0, μ_1. In (b), the global minimum is shown.

9.1.1 Related Work

If μ_0 and μ_1 are fixed, problem (9.2a) becomes equivalent to

$$\underset{\theta(\boldsymbol{x}) \text{ boolean}}{\text{minimize}} \quad \rho J(\theta) + \int_\Omega \theta(\boldsymbol{x}) \Big((I(\boldsymbol{x}) - \mu_1)^2 - (I(\boldsymbol{x}) - \mu_0)^2 \Big) \, d\boldsymbol{x} \,. \tag{9.2b}$$

This problem is still non-convex, because the discrete set $\{0, 1\}$ is non-convex. Chan et al. (2006) showed that globally optimal solutions can still be obtained by relaxing θ to the interval $[0, 1]$, solving the resulting *convex* problem and then thresholding the result. Several algorithms have been developed to solve this convex minimization problem, e.g. by Bresson et al. (2007). If the image is discretized, optimal solutions can also be obtained via graph-cuts, with a suitable J.

On the other hand, if one wants to also optimize over μ_0 and μ_1 simultaneously the problem is no longer convex. In practice, this is solved by alternating between minimizing over θ with μ_0, μ_1 fixed and minimizing μ_0, μ_1 with θ fixed (Bresson et al., 2007; Chan et al., 2006; Chan and Vese, 2001). The latter step is very simple, it just consists of taking the means of the two regions defined by θ (Mumford and Shah, 1989). This procedure does not guarantee that the final solution obtained is globally optimal. Indeed, figure 9.1 shows an image where this procedure fails. The result with initial values of $\mu_0 = 0$ and $\mu_1 = 1$ is shown in figure 9.1a,

which is only a local optimum, because the segmentation in figure 9.1b has a lower objective function value.

Another method is of course to perform an exhaustive search over the parameters μ_0 and μ_1, solving (9.2b) for each possible pair. This is done by Darbon (2007), where a maximum-flow formulation of (9.2b) is solved for every pair of the two levels. The size of the graphs is reduced with a method that bears some resemblance to the one described here. An alternative approach was pursued by Lempitsky et al. (2008) where branch and bound was applied over μ_0 and μ_1 for a discretized version of (9.2a).

9.1.2 Example: Existence of Local Minima

Constructing an explicit example of an image for which (9.2a) has a non-optimal local minimum is not hard. Let $\Omega = [0, 4] \times [0, 1]$. Let $I(\boldsymbol{x})$ consist of three regions separated by the two vertical lines described by $x = 1$ and $x = 2$, respectively. The three regions have gray values 1, 0.5 and 0. The areas of the three regions are:

$$A_1 = 1, \quad A_2 = 1 \quad \text{and} \quad A_3 = 2. \tag{9.3}$$

Because of symmetry, an optimal segmentation will always be a vertical line. Consider the simplest case where $\rho = 0$. If the line $x = 1$ is chosen the optimal values of the mean values are $\mu_0 = \frac{0.5A_2+0A_3}{A_2+A_3} = \frac{1}{6}$ and $\mu_1 = 1$, for a total objective function value of

$$A_1(1-1)^2 + A_2\left(0.5 - \frac{1}{6}\right)^2 + A_3\left(0 - \frac{1}{6}\right)^2 = \frac{1}{6}. \tag{9.4}$$

And for these values of μ_0 and μ_1 the segmentation is optimal since $|0.5 - \mu_0| < |0.5 - \mu_1|$ makes the pixels valued 0.5 be optimally assigned to the background (0). If the outer boundary is chosen the optimal mean values are $\mu_0 = 0$ and $\mu_1 = \frac{1A_1+0.5A_2}{A_1+A_2} = \frac{3}{4}$, for a total function value of

$$A_1\left(1 - \frac{3}{4}\right)^2 + A_2\left(0.5 - \frac{3}{4}\right)^2 + A_3(0-0)^2 = \frac{1}{8}. \tag{9.5}$$

Because $|0.5 - \mu_0| > |0.5 - \mu_1|$ the gray (0.5) pixels are optimally assigned to the foreground (μ_1). Since $1/8 < 1/6$, the first segmentation is a non-optimal local minimum with respect to alternating minimization. Figure 9.1 shows how this can occur in practice.

9.2 Parametric Binary Problems

For any function v, let $v^{(t)}$ denote the function thresholded at t; that is, $v^{(t)}(\boldsymbol{x}) = 1$ if $v(\boldsymbol{x}) > t$ and 0 otherwise. From now on, the smoothness function J satisfies the following requirements:

1. $J(v)$ is convex and $J(v) \geq 0$.
2. $J(tv) = tJ(v)$ for every $t > 0$.
3. $J(v) = \int_{-\infty}^{\infty} J(v^{(t)})dt$ (general co-area formula).

For example, the total variation $\int_\Omega |\nabla v| \, d\boldsymbol{x}$ satisfies these three conditions.

We will now define two optimization problems and show that thresholding the solution to one gives a solution to the other. Let $f(\boldsymbol{x}, s)$ be a real-valued function such that $f(\boldsymbol{x}, \cdot)$ is continuously strictly increasing for each fixed $\boldsymbol{x} \in \Omega$ and $f(\boldsymbol{x}, z(\boldsymbol{x})) = 0$ for all $\boldsymbol{x}$ and some bounded function z. Let F be any function such that $\partial F / \partial s(\boldsymbol{x}, s) = f(\boldsymbol{x}, s)$ for all $(\boldsymbol{x}, s)$. Consider the following discrete problem, defined for a real parameter s:

$$\underset{\theta(\boldsymbol{x}) \text{ boolean}}{\text{minimize}} \quad \rho J(\theta) + \int_\Omega \theta(\boldsymbol{x}) f(\boldsymbol{x}, s) \, d\boldsymbol{x}. \tag{P_s}$$

We will need the following property of the solutions to (P_s):

LEMMA 9.1 *Let $s_1 > s_2$. Then the solutions θ_1 and θ_2 to (P_{s_1}) and (P_{s_2}), respectively, satisfy $\theta_1(\boldsymbol{x}) \leq \theta_2(\boldsymbol{x})$ (a.e.).*

Proof. Define operators $\wedge$ and $\vee$ by:

$$\begin{aligned} (\eta \wedge \theta)(\boldsymbol{x}) &= \min(\eta(\boldsymbol{x}), \theta(\boldsymbol{x})) \\ (\eta \vee \theta)(\boldsymbol{x}) &= \max(\eta(\boldsymbol{x}), \theta(\boldsymbol{x})). \end{aligned} \tag{9.6}$$

Let $E_s(\theta)$ denote the functional to be minimized in (P_s). Since θ_1 and θ_2 are optimal, $E_{s_1}(\theta_1) \leq E_{s_1}(\theta_1 \wedge \theta_2)$ and $E_{s_2}(\theta_2) \leq E_{s_2}(\theta_1 \vee \theta_2)$. Summing these inequalities and using the fact that $J(\theta_1 \wedge \theta_2) + J(\theta_1 \vee \theta_2) \leq J(\theta_1) + J(\theta_2)$ (Chambolle, 2005, Lemma 2.3) results in

$$\int_\Omega \Big(f(\boldsymbol{x}, s_1)(\theta_1(\boldsymbol{x}) - (\theta_1 \wedge \theta_2)(\boldsymbol{x})) + f(\boldsymbol{x}, s_2)(\theta_2(\boldsymbol{x}) - (\theta_1 \vee \theta_2)(\boldsymbol{x})) \Big) \, d\boldsymbol{x} \leq 0. \tag{9.7}$$

But $\theta_1(\boldsymbol{x}) - (\theta_1 \wedge \theta_2)(\boldsymbol{x}) = -(\theta_2(\boldsymbol{x}) - (\theta_1 \vee \theta_2)(\boldsymbol{x}))$, so this is equivalent to

$$\int_\Omega (f(\boldsymbol{x}, s_1) - f(\boldsymbol{x}, s_2))(\theta_1(\boldsymbol{x}) - (\theta_1 \wedge \theta_2)(\boldsymbol{x}))d\boldsymbol{x} \leq 0. \tag{9.8}$$

Since $f(\boldsymbol{x}, s_1) - f(\boldsymbol{x}, s_2) > 0$, $\theta_1(\boldsymbol{x}) - (\theta_1 \wedge \theta_2)(\boldsymbol{x})$ must be equal to 0. $\theta_1(\boldsymbol{x}) \leq \theta_2(\boldsymbol{x})$ a.e. follows. □

The corresponding convex variational problem to (P_s) is:

$$\underset{w(\boldsymbol{x})\in\mathbf{R}}{\text{minimize}} \quad \rho J(w) + \int_\Omega F(\boldsymbol{x}, w(\boldsymbol{x}))\, d\boldsymbol{x}\,. \tag{Q}$$

Problems (P_s) and (Q) are related, as stated by the following theorem:

THEOREM 9.2 *A function w solves (Q) if and only if $w^{(s)}$ solves (P_s) for any $s \in \mathbf{R}$.*

Proof. Define w, for every $\boldsymbol{x} \in \Omega$, as follows:

$$w(\boldsymbol{x}) = \sup\{s \mid \exists\theta \text{ solving } (\mathrm{P}_s) \text{ with } \theta(\boldsymbol{x}) = 1\}.$$

The first task is to show that w is a well-defined real-valued function. If $f(\boldsymbol{x}, s) \leq 0$ for all $\boldsymbol{x}$ it follows that $\theta \equiv 1$ solves (P_s). Similarly, $f(\boldsymbol{x}, s) \geq 0$ implies that $\theta \equiv 0$ is a solution. This means that w is bounded, more precisely that $\inf_{\boldsymbol{y}} z(\boldsymbol{y}) \leq w(\boldsymbol{x}) \leq \sup_{\boldsymbol{y}} z(\boldsymbol{y})$. To see this, choose $s' < \inf_{\boldsymbol{y}} z(\boldsymbol{y})$. By definition we have $w(\boldsymbol{x}) \geq s'$ for all $\boldsymbol{x}$.

The second task is to show that the thresholded function $w^{(s)}$ solves (P_s). Let s be fixed. If $s < w(\boldsymbol{x})$, any solution θ of (P_s) must satisfy $\theta(\boldsymbol{x}) = 1$, while if $s > w(\boldsymbol{x})$ we must have $\theta(\boldsymbol{x}) = 0$ (Lemma 9.1). Since $w^{(s)}$ satisfies these requirements, we see that $w^{(s)}$ is a solution to (P_s). If the set where $w(\boldsymbol{x}) = s$ is not a null set, we can consider s' arbitrary close to s.

Finally, to show that w is the solution to (Q), start out by letting $s_* < \inf_{\boldsymbol{y}} v(\boldsymbol{y})$ for some v and observe that

$$\int_{s_*}^{\infty} v^{(s)}(\boldsymbol{x}) f(\boldsymbol{x}, s)\, ds \;= \int_{s_*}^{v(\boldsymbol{x})} f(\boldsymbol{x}, s)\, ds = F(\boldsymbol{x}, v(\boldsymbol{x})) - F(\boldsymbol{x}, s_*). \tag{9.9}$$

Now integrate over problem (P_s) for all s:

$$\int_{s_*}^{\infty} \left(\rho J(v^{(s)}) + \int_\Omega v^{(s)}(\boldsymbol{x}) f(\boldsymbol{x}, s)\, d\boldsymbol{x} \right) ds \\ = \rho J(v) + \int_\Omega F(\boldsymbol{x}, v(\boldsymbol{x}))\, d\boldsymbol{x} - \int_\Omega F(\boldsymbol{x}, s_*)\, d\boldsymbol{x}\,.$$

But $w^{(s)}$ minimizes the integrand for every s. This means that for any function v,

$$\rho J(v) + \int_\Omega F(\boldsymbol{x}, v(\boldsymbol{x}))\, d\boldsymbol{x} \;\geq\; \rho J(w) + \int_\Omega F(\boldsymbol{x}, w(\boldsymbol{x}))\, d\boldsymbol{x}\,. \quad (9.10)$$

This shows that w is the unique solution of the strictly convex functional in (Q). □

REMARK 9.3 Problem (Q) with $F(\boldsymbol{x}, w(\boldsymbol{x})) = \frac{1}{2}(w(\boldsymbol{x}) - I(\boldsymbol{x}))^2$ is a well-studied convex functional for image restoration, (Rudin et al., 1992). I will refer to it as an ROF problem.

REMARK 9.4 Chambolle (2005) proved Theorem 9.2 for the special case $f(\boldsymbol{x}, s) = s - G(\boldsymbol{x})$ and used the result to approximate the solution to problem (Q) with a series of discrete solutions to (P_s). This chapter uses the result in the other direction, for solving a one-parameter family of discrete problems by thresholding a single ROF solution.

REMARK 9.5 If $f(\boldsymbol{x}, s) = H(\boldsymbol{x})s - G(\boldsymbol{x})$ with $H(\boldsymbol{x}) > 0$, then

$$F(\boldsymbol{x}, w(\boldsymbol{x})) = \frac{1}{2}\left(\sqrt{H(\boldsymbol{x})} w(\boldsymbol{x}) - \frac{G(\boldsymbol{x})}{\sqrt{H(\boldsymbol{x})}} \right)^2 . \quad (9.11)$$

This is the result the rest of this chapter will use. I will call problem (Q) with this data term a weighted ROF problem.

9.2.1 Numerical Method

All numerical experiments in this chapter use the following smoothness function:

$$J(\theta) = \sum_{i=0}^{M} \sum_{j=0}^{N} \sqrt{(\theta_{i+1,j} - \theta_{i,j})^2 + (\theta_{i,j+1} - \theta_{i,j})^2}, \quad (9.12)$$

which can be seen as a discretization of the "true" length of the boundary. Problem (Q) with (9.11) can be written as

$$\underset{w}{\text{minimize}} \qquad J(w) + \frac{1}{2\rho}\int_{\Omega} \left(D(\boldsymbol{x})w(\boldsymbol{x}) - B(\boldsymbol{x})\right)^2 \, d\boldsymbol{x}\,, \tag{9.13}$$

and can be solved by adapting a method by Chambolle (2004, 2005). It involves a projection onto

$$K = \{\operatorname{div}\xi \,|\, \xi \in \mathcal{C}^1(\Omega, \mathbf{R}^2), |\xi(\boldsymbol{x})| \leq 1\}. \tag{9.14}$$

A solution can be found by iterating the following scheme:

$$\begin{cases} w_{i,j}^{(n)} = \left(B_{i,j} + \rho\left(\operatorname{div}\xi^{(n)}\right)_{i,j}/D_{i,j}\right)/D_{i,j} \\ \xi_{i,j}^{(n+1)} = \dfrac{\xi_{i,j}^{(n)} + (\tau/\rho)(\nabla w^{(n)})_{i,j}}{\max\{1, |\xi_{i,j}^{(n)} + (\tau/\rho)(\nabla w^{(n)})_{i,j}|\}} \end{cases}, \tag{9.15}$$

where τ is the step-length. The initial condition can be set to $\xi^{(0)} = \mathbf{0}$ and $w^{(0)} = B$. A suitable stopping criterion when $D \equiv 1$ is also derived by Chambolle (2005): if w^* is the true solution, the error is bounded by

$$\|w^n - w^*\|^2 \leq \rho J(w) - \rho \sum_{i,j} \xi_{i,j}^n \cdot (\nabla w^n)_{i,j}\,. \tag{9.16}$$

This error bound can be divided by the number of pixels to get a bound independent of the size of the image being processed.

9.3 Two-Phase Mumford-Shah Functional

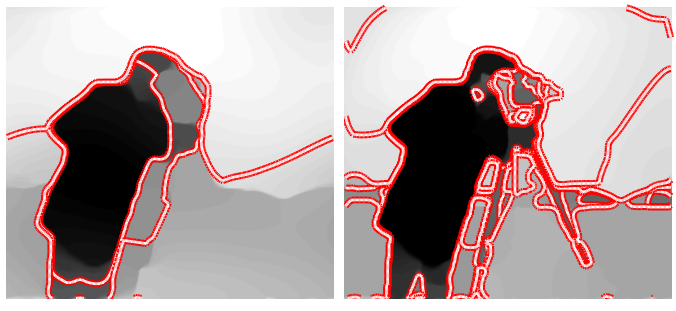
Recall the original problem (9.2a). The following change of variables is required to proceed:

$$\begin{cases} \delta = \mu_1 - \mu_0 \\ \nu = \mu_1^2 - \mu_0^2 \end{cases} \iff \begin{cases} \mu_1 = \dfrac{\nu + \delta^2}{2\delta} \\ \mu_0 = \dfrac{\nu - \delta^2}{2\delta}\,. \end{cases} \tag{9.17}$$

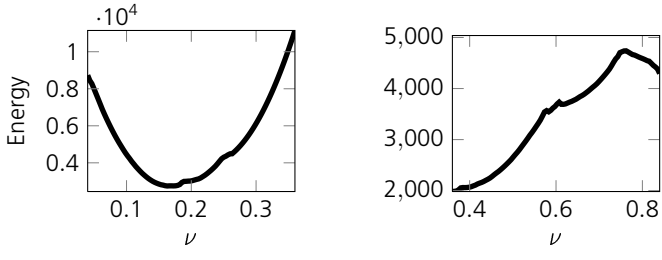
In these new variables, the objective function in (9.2a) is

$$E(\theta, \nu, \delta) = \rho J(\theta) + \int_{\Omega} \theta(\boldsymbol{x})(\nu - 2\delta I(\boldsymbol{x})) + \left(\frac{\nu - \delta^2}{2\delta} - I(\boldsymbol{x})\right)^2 d\boldsymbol{x}\,. \tag{9.18}$$

(a) Solutions to (Q)



(b) Thresholded solutions



(c) Objective function values vs. ν

Figure 9.2: Finding the globally optimal objective function value for a fixed δ by thresholding the solution to problem (Q). The two columns correspond to different values of δ: *left*, 0.2, and *right*, 0.6. Note the completely different non-convex profiles for the two different values of δ.

Let the function $m(\delta)$ previously introduced in (9.1) denote the minimum objective function value possible given a fixed δ, $m(\delta) = \min_{\theta,\nu} E(\theta, \nu, \delta)$. The set of parameters (μ_0, μ_1) has two degrees of freedom and evaluating $m(\delta)$ means optimizing over one degree of freedom while keeping the other one fixed.

9.3.1 Optimization with Fixed Difference

The result in Theorem 9.2 implies that $m(\delta)$ can be evaluated by thresholding the solution to the real-valued problem (Q) for all ν and evaluating the objective function. This is because evaluating $m(\delta)$ amounts to solving

$$\min_{\nu} \left(\min_{\theta} \; E(\theta, \nu, \delta) \right) . \tag{9.19}$$

The inner problem is a special case of (P_s) with $f(\boldsymbol{x}, s) = s - 2\delta I(\boldsymbol{x})$. To see this, recall that the last term of $E(\theta, \nu, \delta)$ in (9.18) does not depend on θ. Remark 9.5 states that it can be solved with (9.15). After the solution to (Q) is obtained, the solution is thresholded and E evaluated for all ν. To summarize, computing $m(\delta)$ consists of the following steps:

1. Solve problem (Q) with $F(\boldsymbol{x}, s) = \frac{1}{2}(s - 2\delta I(\boldsymbol{x}))^2$.
2. For each ν, threshold the solution w at ν and evaluate the resulting objective function value (9.18).
3. The pair (ν^*, θ^*) with the lowest objective value is the global solution to problem (9.2a) with δ fixed.

Step one is a standard ROF problem, for which there exist fast minimization methods; see (Pock, 2008) for an overview and (Pock et al., 2008) for a GPU implementation. The simple MATLAB implementation I used performed one (9.15)-iteration in about 27 ms for $\rho = \delta = 0.5$. The number of iterations required until convergence is strongly dependent on ρ and δ. The second step does not need as much attention as it is a very fast procedure and can trivially be parallelized.

Figure 9.2 shows an example where δ has been fixed to 0.2 and 0.6, respectively. The graphs show that the objective function has a lot of local minima as ν varies. The thresholding process finds the global minimum quickly. It is also interesting to note that the graph of the objective function

looks entirely different for different δ, which suggest that the minimum objective function value is a complicated function with respect to (δ, ν), and therefore nontrivial to minimize.

Figure 9.3 shows $m(\delta)$ evaluated on the entire interval $[0, 1]$ for five images. Note that $m(\delta)$ is often very flat around the global optimum, which has two consequences: (i) it will be difficult to find the optimum δ^* with certainty, but (ii) one evaluation of $m(0.5)$ is often enough to find a good solution, close to the global solution.

9.3.2 Optimization with Varying Difference

It is also possible to solve problem (9.2a) along another direction. The analogous function to $m(\delta)$ is

$$\hat{m}(\nu) = \min_{\theta,\delta} E(\theta, \nu, \delta), \quad \theta(\boldsymbol{x}) \text{ boolean.} \tag{9.20}$$

If we let $s = -\delta$ computing this function means solving

$$\min_{s} \left(\min_{\theta(\boldsymbol{x})} \rho J(\theta) + \int_{\Omega} \theta(\boldsymbol{x})(2I(\boldsymbol{x})s + \nu)\, d\boldsymbol{x} \right) . \tag{9.21}$$

The procedure for calculating $\hat{m}(\nu)$ is the same as the one described in the previous section, with the first step replaced by:

1'. Solve problem (Q) with $F(\boldsymbol{x}, s) = \frac{1}{2}(2I(\boldsymbol{x})s + \nu)^2$.

The resulting minimization problem can be written in the form (9.13). Therefore, this step can be performed with the method described in section 9.2.1.

Figure 9.4 shows an example with the "camera man" image. In the experiment ν was fixed to 0.55 and 0.75. This resulted in two very different curves for the same image.

9.3.3 Obtaining a Lower Bound

We have a method to compute $m(\delta)$; the next logical step is to minimize it. To be able to prove a lower bound, a way to obtain a bound for m on an interval $[\delta_1, \delta_2]$ is required. A good lower bound is an essential part of the branch and bound framework; see section 3.7 on page 31.

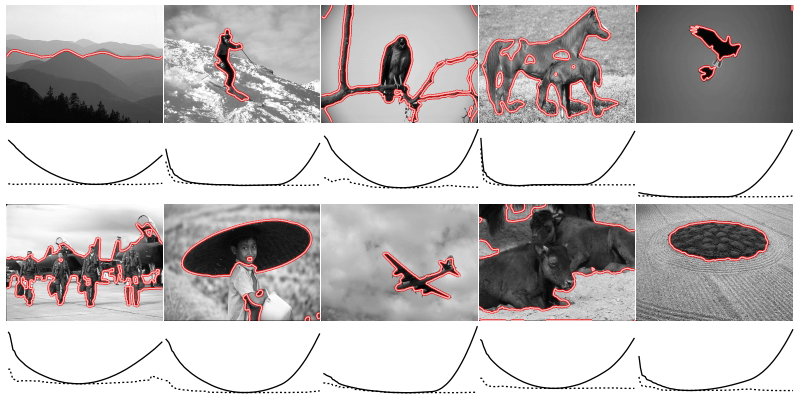


Figure 9.3: The function $m(\delta)$ for 5 images with $\rho = 0.5$. Note that m is very flat near the optimum. The dashed line shows the objective function value after subsequent alternating optimization of μ_0 and μ_1. Image credit: Martin et al. (2001).

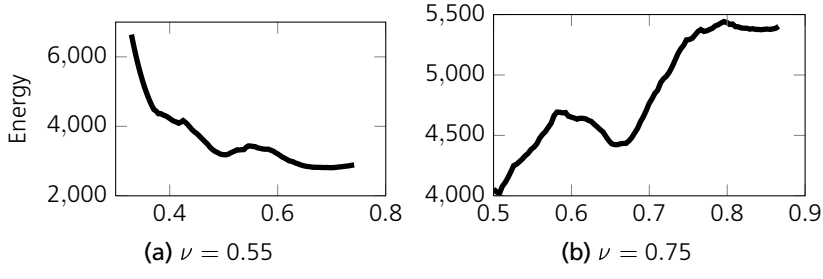


Figure 9.4: "Camera man" image. Objective function values (y-axis) for problem (9.2a) on the curve $\mu_1^2 - \mu_0^2 = \nu$, each obtained by thresholding all possible δ (x-axis).

Finding a lower bound for $m(\delta)$ requires finding one for the objective function $E(\theta, \nu, \delta)$ defined in (9.18) for any $\delta \in [\delta_1, \delta_2]$. This function can be bounded from below on the interval by:

$$E^{\text{bound}}_{\delta_1,\delta_2}(\theta,\nu) = \rho J(\theta) + \int_\Omega \theta(\boldsymbol{x})(\nu - 2\delta_2 I(\boldsymbol{x}))\, d\boldsymbol{x} + \min_{\delta\in[\delta_1,\delta_2]} \int_\Omega \left(\frac{\nu-\delta^2}{2\delta} - I(\boldsymbol{x})\right)^2 d\boldsymbol{x}\,. \quad (9.22)$$

It follows that the minimum of $E^{\text{bound}}_{\delta_1,\delta_2}$ is a lower bound to the minimum of m on $[\delta_1, \delta_2]$. The last term does not depend on θ and can be computed by choosing δ such that $\frac{\nu-\delta^2}{2\delta}$ is as close to the mean of the image as possible. Finding the lower bound therefore amounts to solving (P_s) with $f(\boldsymbol{x}, s) = s - 2\delta_2 I(\boldsymbol{x})$ for every s and computing the minimum of the resulting objective function values. Just like before, every solution can be obtained by thresholding the solution to (Q). Denote the obtained lower bound $m_{\text{bound}}(\delta_1, \delta_2)$.

9.3.4 Global Optimization

The lower bound can be used to perform a branch and bound search on the interval $[0, 1]$, splitting each subinterval until it can either be discarded or contains the optimum. However, obtaining a useful bound for even moderately large subintervals is hard because m is flat (see figure 9.3). Since every calculated bound and every evaluation of m require a solution to (Q), it is essential that previous solutions can be reused. The number of (9.15)-iterations can then be kept to a minimum.

The following method to search for the optimum δ^* satisfies these requirements: A *feasible region* $[\delta_L, \delta_H]$ is maintained, known to contain the optimal value. This region is initially set to $[0, 1]$. The goal is to shrink the feasible region from both ends, i.e. to provide new regions $[\delta_L^{(n+1)}, \delta_H^{(n+1)}] \subset [\delta_L^{(n)}, \delta_H^{(n)}]$ containing δ^*, with the limit of the lengths equal to 0. The algorithm consists of three main steps: two for shrinking the interval from both endpoints using lower bounds and one to search the remaining feasible interval after good candidates to the optimal value E^*. Good candidates are necessary for the bounds to be useful; fortunately, good candidates are found very quickly in practice.

The algorithm iterates three main steps, each associated with a cached dual field ξ for speeding up the (9.15)-iterations. The two bounding steps also store step lengths t_L, t_H which controls the size of the interval to be removed. The steps are detailed in the following list:

1. Try to shrink the interval from above
   - Using the cached dual field ξ_H, solve problem (9.11) with $G(\boldsymbol{x}) = 2(\delta_H + t_H)I(\boldsymbol{x})$.
   - Evaluate $m_{\text{bound}}(\delta_H, \delta_H + t_H)$ by thresholding the solution.
   - If the bound is greater than the currently best function value, discard the interval by setting $\delta_H \leftarrow \delta_H + t_H$. Otherwise, replace t_H by a smaller step; I used $0.8t_H$.
2. Similarly, try to shrink the interval from below.
3. Choose δ inside the feasible interval from $\left\langle \frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{7}{8}, \frac{5}{8}, \frac{3}{8}, \frac{1}{8}, \frac{1}{16}, \ldots \right\rangle$ and evaluate $m(\delta)$.

Because the sequence of evaluated δ is dense in $[0, 1]$, m will eventually be evaluated arbitrarily close to the optimal value. Also, $m_{\text{bound}}(\delta, \delta + t) \to m(\delta)$ as $t \to 0$. From these observations, it is not hard to show that the algorithm is convergent.

9.3.5 Results

Because $m(\delta)$ typically is very flat (figure 9.3), the interval cannot be made very small without substantial computational effort. But an approximate localization of the global optimum can be computed and proved in reasonable time. Figure 9.5 shows the cumulative number of (9.15)-iterations required to localize the global optimum for the "camera man" image. The computation of the first bound for $m(\delta)$ required 401 iterations, while the total number of iterations required to compute the bounds for every subinterval was 1151. The search for the optimal point within the feasible interval required 302 iterations.

It should be noted that even a single evaluation of m at e.g. $\delta = 0.5$ is enough for most images in practice, due to the flatness of m and the fact that the solution will be optimal in the s-direction, which typically has lots

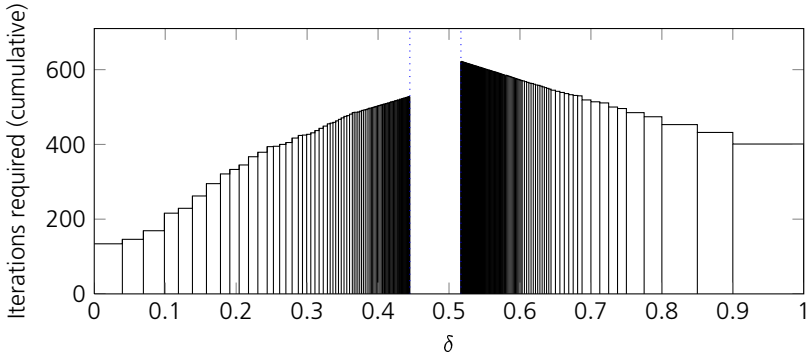


Figure 9.5: Iterations required to shrink the interval for the "camera man" image. A precision of 0.001 was used for the ROF problems. As the intervals grow small, the cached dual field ξ can be reused, allowing the total number of iterations to stay reasonable.

of local minima as shown in figure 9.2. Also, after the optimal solution for a particular δ is obtained, μ_0 and μ_1 are updated before evaluating the function. In fact, the first evaluation of $m(0.5)$ during the test in figure 9.5 resulted in a solution that could not be further improved.

9.4 Ratio Minimization

Problem (P_s) appears in (Kolmogorov et al., 2007) as "parametric max-flow", where it is used, among other things, to minimize a ratio of two functionals. A similar method is used in (Kolev and Cremers, 2009), where instead a sequence of convex problems is solved. Some problems of the same type can be optimized by solving a single convex minimization problem. Pick two functionals P and Q with $Q(\theta) > 0$ and consider the ratio

$$R(\theta) = \frac{P(\theta)}{Q(\theta)}. \tag{9.23}$$

Let $s^* = R(\theta^*)$ be the optimal value of $R(\theta)$. Then $P(\theta^*) - s^*Q(\theta^*) = 0$ and

$$\begin{aligned} \min_\theta P(\theta) - sQ(\theta) \geq 0 \quad &\iff \quad s \leq s^* \\ \min_\theta P(\theta) - sQ(\theta) \leq 0 \quad &\iff \quad s \geq s^*. \end{aligned} \tag{9.24}$$

This means that repeatedly solving problems for different values of s converges to the optimal value s^* via bisection. This is done by Kolmogorov et al. (2007) with repeated max-flow problems.

The result in Theorem 9.2 provides a method to minimize functionals of the following form:

$$\frac{P(\theta)}{Q(\theta)} = \frac{\rho J(\theta) + \int_\Omega \theta(\boldsymbol{x}) g(\boldsymbol{x})\, d\boldsymbol{x}}{\int_\Omega \theta(\boldsymbol{x}) h(\boldsymbol{x})\, d\boldsymbol{x} + K}, \quad h(x) > 0. \tag{9.25}$$

Finding the minimizer of $P(\theta) - sQ(\theta)$ amounts to solving

$$\underset{\theta(\boldsymbol{x}) \text{ boolean}}{\text{minimize}} \quad \rho J(\theta) + \int_\Omega \theta(\boldsymbol{x}) \left(h(\boldsymbol{x})(-s) + g(\boldsymbol{x})\right)\, d\boldsymbol{x}. \tag{9.26}$$

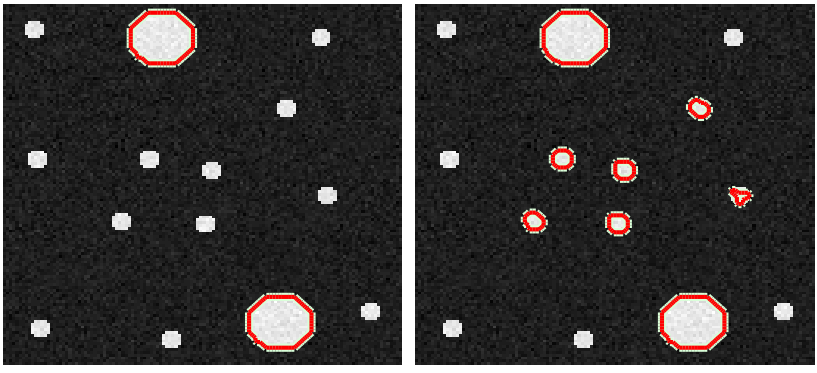
Solving (Q) once and thresholding the result at $-s$ finds minimizers to $P(\theta) - sQ(\theta)$. Searching for s^* is now reduced to thresholding the solution w at different levels and evaluating a function, which can be performed very fast. Define $E_s(\theta) = P(\theta) - sQ(\theta)$ and

1. Start with $s_{\text{min}}, s_{\text{max}}$
2. $s \leftarrow (s_{\text{min}} + s_{\text{max}})/2$.
3. $\theta \leftarrow w^{(-s)}$.
4. If $E_s(\theta) > 0$ set $s_{\text{max}} \leftarrow s$. Otherwise, set $s_{\text{min}} \leftarrow s$.
5. Repeat from step 2.

This scheme will rapidly converge to s^*. Figure 9.6 shows an example. More examples of ratio minimization are found in the paper by Kolmogorov et al. (2007).

9.4.1 Constrained Optimization

It is interesting to note that minimizing a ratio of two functionals bears some similarities to constrained minimization. Consider the following problem, where in addition to a functional, the area of the resulting foreground (where



(a) Minimizing $P(\theta)$ **(b)** Minimizing $P(\theta)/Q(\theta)$

Figure 9.6: Minimizing a ratio of two functions. $Q(\theta) = \int_\Omega \theta(\boldsymbol{x})h(\boldsymbol{x})\,d\boldsymbol{x}$, where $h(\boldsymbol{x})$ is chosen to be larger in the center of the image domain, which makes $\theta(\boldsymbol{x}) = 1$ more favorable.

$\theta(\boldsymbol{x}) = 1$) is also required to be larger than a predetermined minimum value:

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & E(\theta, \mu_0, \mu_1) \\ \text{subject to} \quad & \theta(\boldsymbol{x}) \text{ boolean} \\ & \int_\Omega \theta(\boldsymbol{x})\,d\boldsymbol{x} \geq A\,. \end{aligned} \tag{9.27}$$

The dual function $d(s)$ of this problem is

$$\min_\theta \left(E(\theta, \mu_0, \mu_1) + s\left(A - \int_\Omega \theta(\boldsymbol{x})\,d\boldsymbol{x} \right) \right). \tag{9.28}$$

For any $s \geq 0$, $d(s) \leq E^*$, where E^* is the optimum for (9.27). The best lower bound is given by $d^* = \max_{s \geq 0} d(s)$, which can be computed by thresholding a solution to (Q), since computing $d(s)$ is equivalent to solving

$$\underset{\theta}{\text{minimize}} \quad E(\theta, \mu_0, \mu_1) + \int_\Omega \theta(\boldsymbol{x})\, s\,d\boldsymbol{x}\,, \tag{9.29}$$

followed by an evaluation of (9.28). However, since θ is constrained to be boolean, strong duality does not generally hold (Boyd and Vandenberghe, 2004), that is, $d^* < E^*$ in general.

9.5 Gaussian Distributions

Many variants of the two-phase Mumford-Shah functional have been used for image segmentation. For example, ultrasound images can be segmented using a maximum-likelihood formulation with the assumption that the image pixels are Rayleigh distributed (Sarti et al., 2004). A model emphasizing the difference to (9.2a) is to assume that all pixels have equal expected values. The resulting problem has previously been treated by Rousson and Deriche (2002) with local methods. Consider the following observation model, were the image pixels comes from two Gaussian distributions with zero mean and different variance:

$$I(\boldsymbol{x}) \sim \begin{cases} \mathrm{N}(0, \sigma_1^2), & \theta(\boldsymbol{x}) = 1 \\ \mathrm{N}(0, \sigma_0^2), & \theta(\boldsymbol{x}) = 0\,. \end{cases} \tag{9.30}$$

Given that $\theta(\boldsymbol{x}) = i$, the log-likelihood for the image pixel is

$$\ell_i(I(\boldsymbol{x})) = \log\left(\frac{1}{\sigma_i\sqrt{2\pi}}\exp\left(\frac{I(\boldsymbol{x})^2}{2\sigma_i^2}\right)\right)\,. \tag{9.31}$$

Given an observed image, recovering θ means solving the following minimization problem:

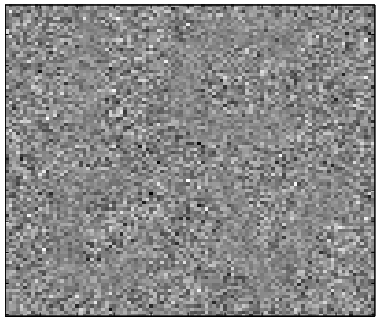
$$\begin{aligned}\underset{\theta,\sigma_0,\sigma_1}{\text{minimize}} \qquad & \rho J(\theta) + \int_\Omega \theta(\boldsymbol{x})\left[-\ell_1(I(\boldsymbol{x}))\right] \\ & + \big(1-\theta(\boldsymbol{x})\big)\big[-\ell_0(I(\boldsymbol{x}))\big]\,d\boldsymbol{x}\,. \end{aligned} \tag{9.32}$$

Following the same approach as in section 9.3.1, we remove the term which does not depend on θ. After rearranging the factors inside the logarithms of the functional, we obtain:

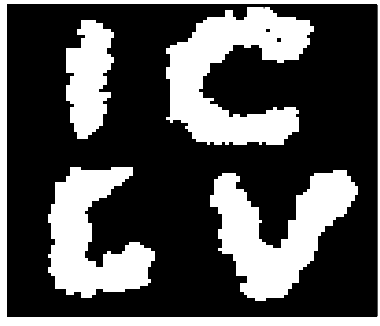
$$\rho J(\theta) + \int_\Omega \theta(\boldsymbol{x})\left(\log\frac{\sigma_1}{\sigma_0} + I(\boldsymbol{x})^2\left(\frac{1}{2\sigma_1^2} - \frac{1}{2\sigma_0^2}\right)\right)\,d\boldsymbol{x}\,.$$

This suggests the following change of variables:

$$\begin{cases} r = \log\frac{\sigma_1}{\sigma_0} \\ t = \frac{1}{2\sigma_1^2} - \frac{1}{2\sigma_0^2} \end{cases} \quad\Longleftrightarrow\quad \begin{cases} \sigma_1 = \frac{\sqrt{-2t(e^{2r}-1)}}{2t} \\ \sigma_0 = \frac{\sqrt{-2t(e^{2r}-1)}}{2te^r}\,. \end{cases}$$



(a) Original image with $\sigma_1 = 6$ and $\sigma_0 = 10$



(b) Resulting segmentation. Recovered σ: 6.07 and 10.35

Figure 9.7: Segmentation of an image composed of two Gaussian distributions with zero mean.

Problem (9.32) is now tractable for $t =$ constant. The first step is solving an ROF problem with $(w(\boldsymbol{x}) + I(\boldsymbol{x})^2 t)^2$ as data term. Then, threshold the solution at all possible levels r, evaluating the function in (9.32) and choose the lowest value. A result with this segmentation model can be seen in figure 9.7.

9.6 Conclusion

This chapter has shown that the two-phase, binary Mumford-Shah functional can be effectively optimized by solving a continuous problem followed by thresholding. The method works if the difference between the two levels is fixed. A branch and bound-like method is required to solve the general problem. It is interesting to note that a single evaluation of $m(\delta)$, which is optimal along one dimension, often seems to be enough to find the global optimum.

The work in this chapter is based on a publication from 2009 (Strandmark et al., 2009*b*). Since then, "Little work has been devoted to global optimization over the regions and parameters simultaneously in the image segmentation models" (Bae et al., 2011), with a few exceptions briefly summarized in chapter 15.

Part III

Curvature Regularization

Chapter 10

Curvature Regularization in the Plane

The previous chapter involved finding the optimal curve with respect to its contents and its length. This chapter adds a curvature term, which requires radically different methods. The discussions I have had on this topic with Thomas Schoenemann have been very useful, and as this chapter builds upon the framework by Schoenemann et al. (2009, 2012), familiarity with their papers will be useful.

10.1 Problem Formulation

In this chapter and the next, the goal is to improve the computational methods for solving variational problems involving length and curvature regularization. This chapter starts by looking at the most basic setting in the plane; the next chapter will consider extensions:

$$\inf_{\gamma} \left(\int_{\text{Int}(\gamma)} g(\boldsymbol{x})\, d\boldsymbol{x} + \int_{\gamma} \left(\rho + \sigma \kappa(s)^d \right) ds \right), \tag{10.1}$$

where the infimum is taken over all $\gamma = \gamma^{(1)} + \ldots + \gamma^{(n)}$ and each $\gamma^{(k)}$ is a simple, closed, twice differentiable Jordan curve parametrized by arc length. The interiors, denoted by $\text{Int}(\gamma^{(k)})$, are required to be pairwise disjoint. The domain is a compact subset $\Omega \subset \mathbf{R}^2$. Note that the number of curves n is also considered to be an unknown. The function $g : \Omega \to \mathbf{R}$ is called the data term and is, because of the nature of image acquisition, piecewise constant over the pixels $[i, i+1) \times [j, j+1)$ for all integers i and j. The scalars $\rho \geq 0$ and $\sigma \geq 0$ are given weighting factors for controlling the amount of length and curvature regularization, respectively.

The curvature power d controls how sharp turns are penalized. If $d = 1$, sharp turns cost the same as slow turns, whereas if $d = 2$, sharp turns cost more than slow ones. Figure 10.16 shows the difference between the two; using $d = 2$ gives the letter 'R' rounded corners. Unless otherwise stated, I have used $d = 2$ for all experiments in this paper.

Problem (10.1) is an infinite-dimensional problem that is unlikely to be solved analytically (i.e. by giving a closed-form solution). It is hence natural to resort to a computational approach. Any such approach will eventually have to discretize the problem. This work follows a line of spatially discretizing the problem by only allowing polygonal region boundaries. Indeed, the work of Bruckstein et al. (2001) shows that the integrals in (10.1) can be approximated by polygonal curves, so that if one successively enlarges the set of considered curves one will eventually converge to the continuous problem.

Therefore, we consider the following problem instead:

$$\inf_{\Gamma=\Gamma^{(1)}+\ldots+\Gamma^{(n)}} \sum_{k=1}^{n} \Bigg(\int_{\mathrm{Int}(\Gamma^{(k)})} g(\boldsymbol{x})\, d\boldsymbol{x} + \sum_{(\vec{\ell}_i,\vec{\ell}_{i+1})\in\Gamma^{(k)}} \Big(\rho|\vec{\ell}_i| + \sigma\kappa_d(\vec{\ell}_i,\vec{\ell}_{i+1})\Big)\Bigg), \quad (10.2)$$

where $\Gamma^{(1)},\ldots,\Gamma^{(n)}$ are closed, simple and piecewise linear curves. The inner sum is taken over all pairs of consecutive segments $(\vec{\ell}_i,\vec{\ell}_{i+1})$ of the curve $\Gamma^{(k)}$. This chapter uses the discrete curvature $\kappa_d(\vec{\ell}_i,\vec{\ell}_{i+1}) = |\alpha_i|^d$, where α_i is the angular difference between the line segments $\vec{\ell}_i$ and $\vec{\ell}_{i+1}$.

Problem (10.2) is still not finite-dimensional. However, by restricting Γ to a finite mesh of allowable line segments, the problem can be turned into a combinatorial optimization problem. This is the approach pursued by Schoenemann et al. (2009). It is clear that finer meshes with shorter line segments and higher number of angular directions will yield lower objective functions values and thus approximate the original problem (10.2) more accurately. I will explore different types of meshes; see for example figure 10.3. It may not come as a surprise that hexagonal meshes are more economical than square ones in the sense that they achieve lower objective function values with fewer line segments.

10.2 Length-Based Regularization

The basis for this work is the discrete differential geometry framework developed by Sullivan (1990) and Grady (2010) for computing minimal surfaces and shortest paths. Schoenemann et al. (2009) recast the final combinatorial problem as an integer linear program and solved it via LP relaxation. This section limits the exposition to the standard case without the curvature term (corresponding to $\sigma = 0$). Interestingly, this integer linear program can be shown to be totally unimodular and hence the LP relaxation will be tight.

The method is based on tessellating the domain of interest into a so-called cell complex, a collection of non-overlapping basic regions whose union gives the original domain. Several different kinds of tessellations are possible. Some two-dimensional examples are given in figure 10.3. Typical choices are square meshes (2D), resulting in 4-connectivity. To mimic 8-connectivity, pixels are subdivided into four triangular regions each. This issue will be elaborated upon in section 10.5.

The boundaries of 2D regions are called edges. It is necessary to consider both possible orientations of each edge and facet. In the integer linear program, there are two sets of boolean variables, one reflecting regions and the other related to boundaries. For each basic region, a binary variable reflects whether the region belongs to the foreground or the background. Let x_i, $i = 1, \ldots, m$ denote these binary variables, where m is the number of basic regions. The region integral in (10.2) is now easily approximated by a linear objective function of the form $\sum_{i=1}^{m} g_i x_i$.

In this chapter x_i denote region variables and y_i boundary variables. The length term in (10.2) is then represented with $\rho \sum_i |\vec{\ell}_i| y_i$, where $|\vec{\ell}_i|$ denotes the length of edge i. To enforce consistency between the region and boundary variables, surface continuation constraints (Schoenemann et al., 2009) are used:

Surface continuation constraints. Assume that a basic region is part of the foreground. Then, each of its edges can have two valid configurations: either the associated basic region on the other side of the edge is also part of the foreground—or the foreground region terminates here in an appropriately oriented boundary element. These constraints, together with the cases where the considered basic region is background, can be phrased as a linear

equation system, with one constraint for each edge k:

$$\sum_i b_{k,i} x_i + \sum_i b_{k,i} y_i = 0, \tag{10.3}$$

where $b_{k,i}$ indicates whether region i is positive (1), negative (-1) or not incident (0) to edge k. Two terms in each sum will be nonzero.

10.2.1 Computation of the Data Term

The data term for the region R_i is the integral of g over that region:

$$g_i = \int_{R_i} g(\boldsymbol{x}) d\boldsymbol{x}. \tag{10.4}$$

That is, to determine the contribution of each region to the data term requires the computation of many integrals of the type $\int_{Int(\Gamma)} g(\boldsymbol{x})\, d\boldsymbol{x}$. A routine application of Green's theorem yields

$$\int_{Int(\Gamma)} g(\boldsymbol{x})\, d\boldsymbol{x} = \int_{\Gamma} G(x, y)\, dy, \tag{10.5}$$

where $G(x, y) = \int_0^x g(\hat{x}, y)\, d\hat{x}$ can be computed in advance. This technique is similar to the "summed area tables" (Crow, 1984) and "integral images" (Viola and Jones, 2004) commonly used in computer graphics and vision.

10.3 Incorporating Curvature

To be able to handle curvature regularization, *pairs* of boundary variables has to be introduced. Let us denote these pairs by y_{ij}. Schoenemann et al. (2009) described how to introduce boundary continuation constraints to ensure that an actual boundary curve is formed. Without this constraint, only straight line pairs would be used.

Boundary continuation constraints. If a pair of line segments (l_1, l_2) is part of the boundary line, there must be another pair of line segments (l_2, l_3) that is also part of the boundary line. Furthermore, there must be a pair

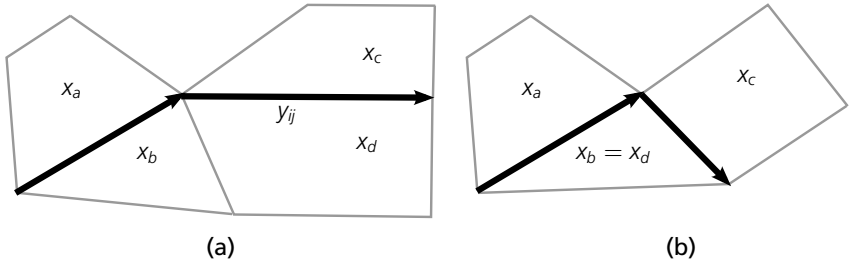


Figure 10.1: The line pair variable y_{ij} and its four incident region variables x_a, x_b, x_c and x_d. The four region variables may coincide for some edge pairs.

(l_0, l_1) that likewise belongs to the boundary line. Again, these constraints can be phrased as a linear equation system for each oriented edge ℓ:

$$\sum_{i,j} c_{\ell,ij} y_{ij} = 0, \tag{10.6}$$

where

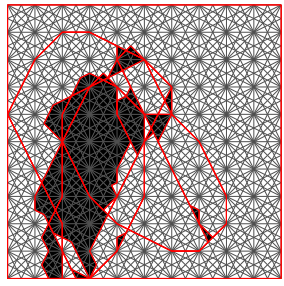
$$c_{\ell,ij} = \begin{cases} 1 & \text{if } \ell = i \\ -1 & \text{if } \ell = j \\ 0 & \text{otherwise.} \end{cases} \tag{10.7}$$

Having introduced the line pair variables, the last term in (10.2) may also be represented as a linear function: $\sigma \sum_{i,j} b_{ij} y_{ij}$.

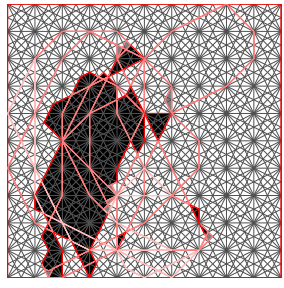
10.3.1 Avoiding Extraneous Arcs

The constraints introduced by Schoenemann et al. admit too many feasible solutions. This is illustrated in figure 10.2a, where sharp corners are avoided by introducing extra curves, which due to the nonexistent length penalty have low cost. This solution is integral and optimal in the original formulation, because along the spurious large arcs both y_{ij} and y_{ji} are active.

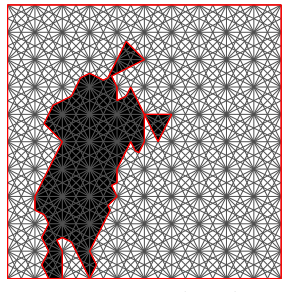
The solution seems simple: to add constraints $y_{ij} + y_{ji} \leq 1$. This would indeed solve the problem if the variables could be restricted to be integral, but in practice these constraints give a fractional solution with even more spurious arcs (fig. 10.2b). Therefore, the additional linear constraints proposed in this chapter are of a different type:



(a) Using the original constraints from (Schoenemann et al., 2009). The small black regions to the right have their boundary costs reduced by the large arcs of extra boundaries.



(b) Simply requiring that $y_{ij} + y_{j,i} \leq 1$ would work if the variables were integers, but causes the LP relaxation to output a fractional solution.



(c) Result using the additional constraints (10.9). The LP-relaxation output an integral solution.

Figure 10.2: Segmentation with and without region consistency constraints. A very crude mesh was used to make the visualization clearer. Gray scale polygons indicate region variables and red lines indicate edge pair variables. Gray lines show the mesh used. Parameters: $\rho = 0$, $\sigma = 300000$. The time to solve the problem decreased by adding the extra constraints: 0.251s vs. 3.402s for the original problem.

Region consistency constraints. Consider a line pair variable y_{ij} and call its four incident regions x_a, x_b, x_c and x_d, located as shown in figure 10.1a. If $x_a = x_b = 1$ or $x_a = x_b = 0$, the region pair should not be active. Similarly for x_c and x_d. This can be linearly encoded as

$$\begin{aligned} x_a + x_b + y_{ij} + y_{ji} &\leq 2 \\ -x_a - x_b + y_{ij} + y_{ji} &\leq 0. \end{aligned} \tag{10.8}$$

Similar constraints hold for x_c and x_d. All in all, four new constraints are introduced for each pair of edges. It might be the case that x_a and x_c or x_b and x_d coincide; see figure 10.1b. The constraint (10.8) still looks the same. To reduce the total number of constraints, the constraints can be combined into four constraints per edge k:

$$\begin{aligned} x_{k_1} + x_{k_2} + \sum_{(k,j) \text{ a pair}} y_{k,j} &\leq 2 \\ -x_{k_1} - x_{k_2} + \sum_{(k,j) \text{ a pair}} y_{k,j} &\leq 0 \\ x_{k_1} + x_{k_2} + \sum_{(j,k) \text{ a pair}} y_{j,k} &\leq 2 \\ -x_{k_1} - x_{k_2} + \sum_{(j,k) \text{ a pair}} y_{j,k} &\leq 0. \end{aligned} \tag{10.9}$$

Here x_{k_1} and x_{k_2} denote the two regions adjacent to edge k. The first two constraints sum over all line pairs starting with edge k and the last two sum over all pairs ending with edge k.

Figure 10.2c shows the result with these additional constraints where the boundary now is consistent with the region variables. As a bonus, the new constraints reduced the time required to solve the problem to about 7%. Figure 10.2 also shows that both before and after the additional constraints, the optimal solution has its region variables equal or very close to 0 or 1.

I have also run more quantitative experiments. Table 10.1 shows how the new constraints perform compared to no constraints and the simple fix. The new constraints always perform much faster and obtains a much higher quality solution when the amount of length regularization is low. Even when the correct global optimum is found without the new constraints, including them speeds up the computation by more than one order of magnitude.

Image	Overlapping arcs			Non-integral pairs			Runtime (s)		
	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)
	0	0	0	0	0	0	1.8	2.1	0.5
	221	625	0	680	1589	0	36.0	56.4	2.4
	930	1343	0	1952	3880	0	51.5	125.0	6.9
	0	0	0	0	0	0	38.0	42.3	1.0
	640	860	0	1123	2044	0	37.2	56.1	2.2

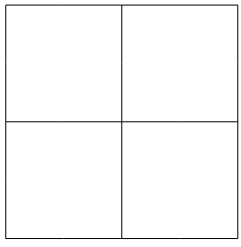
Table 10.1: Evaluation of the new constraints for a 16 × 16 mesh. (a) is no overlap prevention; (b) is the simple requirement that $y_{ij} + y_{ji} \leq 1$ and the (c) columns show the new constraints. Even for the cases where the solution is correct, the new constraints allows the solution to be computed faster.

For these reasons, all subsequent experiments in the paper use the new, additional constraints.

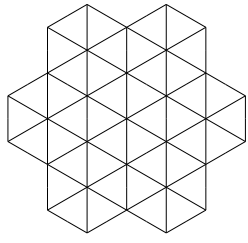
10.4 Pseudo-Boolean Optimization

Solving the discrete optimization problem does not have to be done using a linear program. It is also possible to use discrete optimization methods. Each edge pair is represented as a 3- or 4-clique in a pseudo-boolean objective function. This formulation has the advantage that it readily carries over to three dimensions. El-Zehiry and Grady (2010) used 3-cliques for minimizing curvature functionals and their formulation is equivalent to the LP formulation for 4-connected grids. This is because in a 4-connected grid, only configurations of the type in figure 10.1b are present. If one wants to go to higher connectivities, configurations as shown in figure 10.1a are present and 4-cliques are required. We will now see why.

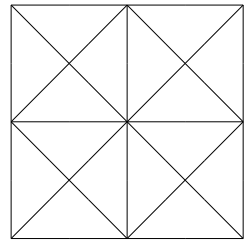
3-cliques. An edge pair connected to three region variables is shown in figure 10.1b. The edge pair adds a cost to the segmentation objective if



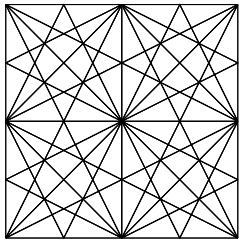
(a) Square mesh with 4-connectivity. Each cell has 1 region and 2 lines (on average).



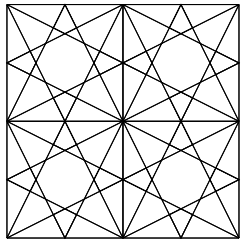
(b) Hexagonal mesh with 6-connectivity. Each cell has 6 regions and 9 lines.



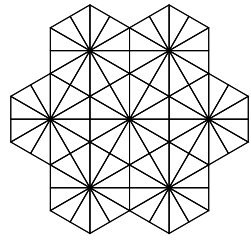
(c) Square mesh with 8-connectivity. Each cell has 4 regions and 6 lines.



(d) Square mesh with 16-connectivity. Each cell has 32 regions and 52 lines.



(e) Square mesh with 12-connectivity. Each cell has 25 regions and 44 lines.



(f) Hexagonal mesh with 12-connectivity. Each cell has 12 regions and 18 lines.

Figure 10.3: Different types of grids. The maximum angle between the possible straight lines is 90° in (a), 60° in (b) and 45° in (c). Meshes (d), (e) and (f) have about 27°, 37° and 30° as their maximum angle, respectively.

$x_a = x_c = 1$ and $x_b = 0$ or the opposite: $x_a = x_c = 0$ and $x_b = 1$:

$$b_{ij}\Big(x_a x_c(1 - x_b) + (1 - x_a)(1 - x_c)x_b\Big). \tag{10.10}$$

It is noted in (El-Zehiry and Grady, 2010) that this expression, when simplified, does not contain any monomials of degree 3. It is equal to

$$b_{ij}\Big(x_a x_c - x_a x_b - x_b x_c + x_b\Big), \tag{10.11}$$

which contains only pairwise interactions between the three variables.

4-cliques. For all connectivities higher than 4, many edge pairs are adjacent to 4 regions; see figure 10.1a. Just as before, this can be represented as

$$b_{ij}\Big(x_a x_c(1 - x_b)(1 - x_d) + (1 - x_a)(1 - x_c)x_b x_d\Big). \tag{10.12}$$

Representing this, however, requires an optimization method capable of handling higher degrees, because the above expression simplifies to

$$\begin{aligned} b_{ij}\Big(&2x_a x_b x_c x_d - x_a x_b x_c - x_a x_c x_b \\ &- x_a x_b x_d - x_b x_c x_d + x_b x_d + x_a x_c\Big), \end{aligned} \tag{10.13}$$

where the high-arity factors have not canceled each other out. How to handle terms like this was described in detail in the first part of this this thesis. Representing the above 4-clique with HOCR requires 10 extra nodes and 24 monomials of degree 2. This is much worse than the three clique which required no extra nodes and only 3 monomials.

10.4.1 Comparison Between the Two Approaches

In the case of 4-connectivity, El-Zehiry and Grady (2010) solve exactly the same problem as the LP relaxation. Since El-Zehiry and Grady do not use nor discuss higher-order cliques, comparisons in the 4-connected case are interesting even though such grids are far too coarse for any practical use.

σ	Time (s)			Unassigned		
	LP	HOCR	Probe	LP	HOCR	Probe
1000	23.9	0.1	0.7	0%	6%	3%
2000	30.6	0.7	5.2	0%	53%	39%
3000	41.4	1.7	14.4	≈0%	95%	56%
4000	43.5	2.0	6.7	≈0%	100%	66%
5000	49.1	1.9	5.1	≈0%	100%	≈100%
6000	54.6	2.0	5.1	≈0%	100%	≈100%
7000	60.5	2.1	5.1	≈0%	100%	≈100%
8000	64.1	2.2	5.0	0%	100%	≈100%
9000	70.5	2.2	5.1	0%	100%	≈100%
10000	76.6	2.9	5.6	0%	100%	100%
50000	553.3	3.1	5.2	0%	100%	100%
100000	1714.9	3.0	5.3	0%	100%	100%
1000000	12166.3	3.0	5.2	0%	100%	100%

Table 10.2: 4-connectivity results with $\rho = 100$. The LP formulation always found or ended up very close to the global optimum. For $\sigma = 1000$, the curvature part of the energy at the optimum was about 10 times larger than the length part.

4-connectivity. The first experiment used the standard image shown in figure 10.4 and the data term

$$g(x, y) = (I(x, y) - \mu_1)^2 - (I(x, y) - \mu_0)^2, \tag{10.14}$$

with $\mu_0 = 128$ and $\mu_1 = 0$. I used a small amount of length regularization ($\rho = 100$) to increase the likelihood of a unique solution. When σ varies, the segmentation changes as shown in figure 10.4.

The experimental results shown in table 10.2 are in strong contrast to the results reported by El-Zehiry and Grady (2010), who reported good performance for 4-connected meshes. For really simple problems (low curvature weight) the method works. Other research groups have also evaluated this method of solving discrete curvature problems without success (Andrew Delong, personal communication).

Higher Connectivities. Tables 10.3 and 10.4 show the result for 8- and 16-connected square meshes, respectively; table 10.5 uses a hexagonal mesh. Quadratic roof duality (using reductions) is not strong enough to solve the boolean problem, except for negligible amounts of regularization (see figures 10.5, 10.6 and 10.7).

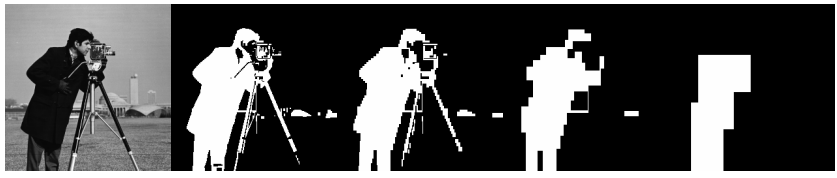


Figure 10.4: Input image and results with regularizations $\rho = 100$ and $\sigma \in \{1000, 10000, 100000, 1000000\}$. The mesh is 4-connected of size 256×256 (corresponding to the image pixels).

Figure 10.5: Input image and results with an 8-connected 128×128 mesh, using the same parameters as in Fig. 10.4.

Figure 10.6: Input image and results with a 16-connected 64×64 mesh, using the same parameters as in Fig. 10.4. The solution for $\sigma = 10^6$ is fractional, as shown by the gray regions.

Figure 10.7: Input image and results with a 64×64 hexagonal mesh, using the same parameters as in Fig. 10.4.

σ	Time (s)			Unassigned		
	LP	HOCR	Probe	LP	HOCR	Probe
1000	25.1	43.8	764.0	≈0%	≈100%	≈100%
2000	26.9	46.8	753.7	≈0%	100%	100%
3000	28.2	54.0	756.3	≈0%	100%	100%
4000	29.9	51.1	755.9	≈0%	100%	100%
5000	31.3	53.7	755.8	≈0%	100%	100%
6000	32.5	45.4	747.6	0%	100%	100%
7000	32.6	50.1	751.3	0%	100%	100%
8000	34.3	51.3	757.4	0%	100%	100%
9000	35.9	55.2	755.8	0%	100%	100%
10000	36.8	51.9	754.7	0%	100%	100%
50000	81.5	54.0	751.9	0%	100%	100%
100000	232.8	53.4	750.9	0%	100%	100%
1000000	10843.5	54.0	751.9	0%	100%	100%

Table 10.3: 8-connectivity results with $\rho = 100$. The LP formulation always found or ended up very close to the global optimum. Roof duality never worked.

σ	Time (s)			Unassigned		
	LP	HOCR	Probe	LP	HOCR	Probe
1000	100.5	58.3	2243.1	0%	100%	100%
2000	118.2	61.9	2238.7	≈0%	100%	100%
3000	130.9	63.4	2249.5	≈0%	100%	100%
4000	144.7	66.4	2250.5	≈0%	100%	100%
5000	158.5	67.1	2245.2	≈0%	100%	100%
6000	169.6	69.2	2242.6	≈0%	100%	100%
7000	177.6	70.2	2248.9	≈0%	100%	100%
8000	186.3	75.0	2258.6	≈0%	100%	100%
9000	211.6	69.2	2255.2	≈0%	100%	100%
10000	215.2	72.4	2293.5	≈0%	100%	100%
50000	791.3	69.0	2318.8	0%	100%	100%
100000	2688.9	70.0	2292.5	≈0%	100%	100%
1000000	362419.0	72.8	2359.7	≈0%	100%	100%

Table 10.4: 16-connectivity results with $\rho = 100$. The LP formulation always found or ended up very close to the global optimum. Roof duality never worked.

σ	Time (s)			Unassigned		
	LP	HOCR	Probe	LP	HOCR	Probe
1000	22.0	32.9	988.2	0%	100%	100%
2000	23.2	32.1	989.4	0%	100%	100%
3000	26.0	34.1	1003.4	≈0%	100%	100%
4000	26.9	36.5	997.1	≈0%	100%	100%
5000	28.3	33.0	996.3	≈0%	100%	100%
6000	29.6	34.9	997.7	≈0%	100%	100%
7000	30.7	36.3	996.9	0%	100%	100%
8000	30.8	34.5	1002.1	0%	100%	100%
9000	32.6	36.4	1001.5	≈0%	100%	100%
10000	33.5	36.2	1008.2	≈0%	100%	100%
50000	74.3	35.7	1000.6	0%	100%	100%
100000	188.2	37.7	1011.8	0%	100%	100%
1000000	6459.8	35.7	1000.6	0%	100%	100%

Table 10.5: Hexagonal 12-connectivity results with $\rho = 100$. The LP formulation always found or ended up very close to the global optimum. Roof duality never worked.

σ	Time (s)			Unassigned		
	GRD	HOCR	Probe	GRD	HOCR	Probe
1000	1.4	0.1	0.3	0%	1%	0%
2000	1.4	0.3	1.8	1%	6%	1%
3000	1.6	0.7	23.4	4%	30%	15%
4000	3.0	1.1	29.7	9%	52%	37%
5000	4.2	1.2	32.1	39%	56%	52%
6000	4.5	2.9	45.8	54%	78%	56%
7000	4.7	1.5	59.7	56%	97%	72%
8000	12.7	1.5	52.5	69%	98%	89%
9000	14.3	1.3	53.7	87%	99%	96%
10000	3.9	1.4	53.0	98%	99%	98%

Table 10.6: Comparison between generalized roof duality (GRD, chapter 4) and reductions using a small (32×32) 8-connected mesh.

Generalized Roof Duality. Chapter 4 proposed a new method of minimizing higher-order pseudo-Boolean functions. While generalized roof duality performs better than using reductions to handle the higher-order terms, it does not perform well enough to compete with the linear programming relaxation. See table 10.6. Computing the optimal submodular relaxation for each problem requires smaller meshes (section 4.6 introduced an efficient heuristic, though).

The experiments in this section show that formulating the discrete curvature problem (10.2) as a general pseudo-boolean problem and using current optimization methods is not a good approach. The linear programming approach is able to solve much harder problems exactly (but it might take a while).

10.5 Types of Meshes

The mesh used for segmentation can be created in a number of ways. The quality of the approximation depend on how many different possible straight lines that can be represented by the mesh, since a larger possible choice of line directions allows the mesh to approximate a continuous curve more closely. Figure 10.3 shows some possible meshes and the straight lines they admit. If a mesh allows n possible straight line directions, it is referred to as n-connected.

10.5.1 Hexagonal Meshes

Hexagonal meshes have long been studied for image processing (Middleton and Sivaswamy, 2005). One characterizing fact of hexagons is that they are the optimal way of subdividing a surface into regions of equal area while minimizing the sum of the boundary lengths (Hales, 2001). However, the property that is more important to us is the neighborhood structure. In a hexagonal lattice every region has 6 equidistant neighbors. When approximating curvature we would like to represent as many different straight lines as possible and we would like the maximum angle between them to be small, as that gives us a better approximation of a smooth curve (Bruckstein et al., 2001). The neighborhood structure of the hexagonal mesh allows for similar performance (number of lines and angle between them) while using fewer regions. This is illustrated in figure 10.3, where three crude meshes and three

finer meshes are shown. The meshes in figures 10.3d and 10.3f have similar maximal angle between the possible straight lines, but the hexagonal mesh achieves this with fewer regions due to the favorable intersection pattern of the lines. This suggests that hexagonal meshes can achieve the same accuracy as the mesh (d) used by Schoenemann et al. (2009) with a significantly smaller linear program.

With the introduction of the hexagonal grid, every region is no longer contained within a single pixel. Some regions will partly overlap more than one pixel. The contribution g_k to the data term of each region can be efficiently computed; see section 10.2.1 on page 140.

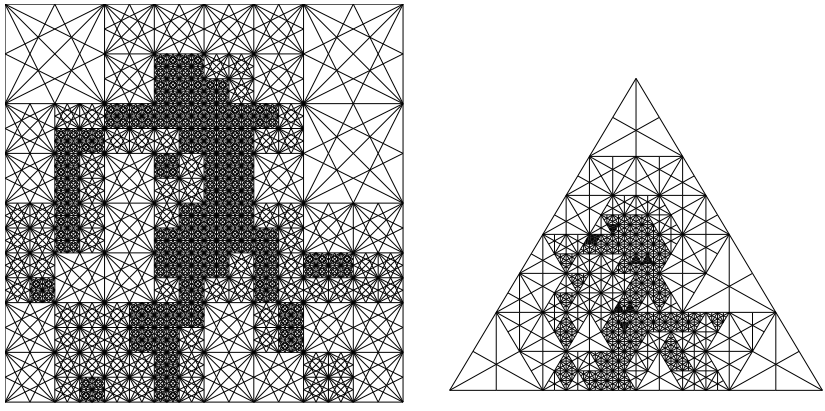
10.5.2 Adaptive Meshes

The memory requirements for solving the linear programs arising from the discretizations are very large. Each pair of connected edges introduce two variables. Linear programs are typically solved using the simplex method or interior point methods, both of which require a substantial amount of memory for these problems. As one example, a problem with 131,072 regions and 1,173,136 edge pairs required about 2.5 GB of memory to solve using the Clp solver.

For this reason, it is desirable to keep the size of the mesh small. However, a fine mesh is needed to be able to approximate every possible curve. The solution to this conflict of interest is to generate the mesh adaptively, to only give it high resolution where the segmentation boundary is likely to pass through. Adaptive meshes have previously been considered for image segmentation in the level-set framework (Xu et al., 2004) and in combinatorial optimization of continuous functionals (Kirsanov and Gortler, 2004).

The mesh is refined using an iterative process. First, a single region is put into a priority queue. Then regions are removed from the priority queue and subdivided into smaller regions which are put back into the queue. The region which most urgently needs to be split is removed first from the priority queue. This is determined by a score which is computed for each region as follows.

Start with q an empty priority queue
$R \leftarrow (0, 0, w, h)$
Add R to q with priority $=$ score(R)



(a) 16-connectivity by subdividing rectangles

(b) 12-connectivity by subdividing triangles

Figure 10.8: Adaptive meshes can be constructed by recursively subdividing basic shapes into several similar shapes and finally adding the extra connectivity.

while size(q) $< L$ **do**
  Remove R from q
  Split R into $R_1 \dots R_k$
  Add $R_1 \dots R_k$ to q with score(R_1) $\dots$ score(R_k)
end while

Both square and triangular basic shapes can be split up into four identical shapes similar to the original one. All adaptive meshes in this chapter use $k = 4$. The score function can be chosen in many different ways. One way is to use the squared deviation from the mean of each region, i.e.:

$$\text{score}(R) = \int_R (I(\boldsymbol{x}) - \mu(R))^2 d\boldsymbol{x}, \tag{10.15}$$

where $\mu(R) = \frac{1}{|R|} \int_R I(\boldsymbol{x}) d\boldsymbol{x}$. This way, regions where the data term varies a lot will be split before regions which have a uniform data term. The score is not normalized, because otherwise many very small regions would tend to have a big score. Once again, the integrals may be computed efficiently via (10.5) on page 140.

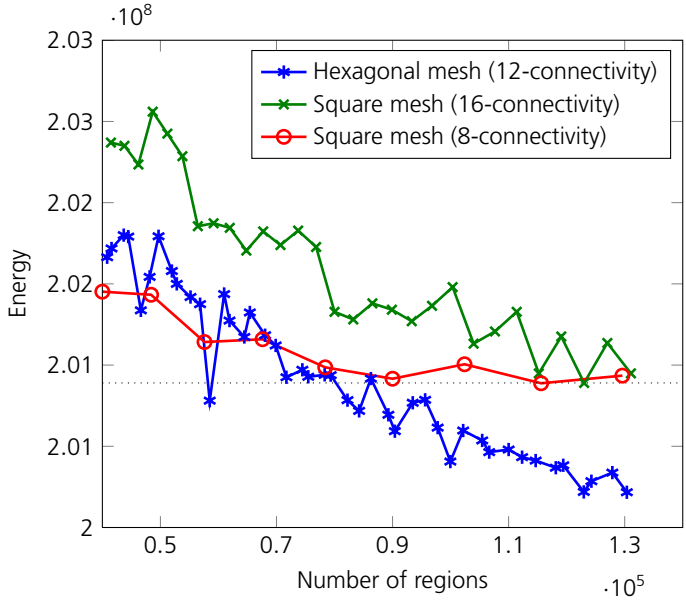


Figure 10.9: Optimal objective function value vs. the total number of regions. The best accuracy obtained by the square mesh was achieved by the hexagonal mesh with about half the number of regions. This experiment used $\rho = \sigma = 10000$. The objective function difference might seem small, but differences of these magnitudes often correspond to significant changes in segmentation; see figure 10.10.

10.5.3 Experiments

All experiments in this chapter use the simple, two-phase data term in (10.14) on page 147. The constants μ_0 and μ_1 were iteratively estimated without any regularization.

Hexagonal Meshes. The experiment evaluating hexagonal vs. square meshes compares three types of meshes, the 8- and 16-connected square mesh and the 12-connected hexagonal mesh, shown in figure 10.3c, d and f. The comparison uses a fixed data term of a 256×256 image and places meshes of various types and sizes on top of it and calculates the optimal objective function value.

The result is shown in figure 10.9, where the optimum is plotted as a function of the number of regions used. The number of regions is a good indicator of the total size of the linear program and plots using the

Image	Objective value		Number of regions		Runtime (s)	
	Square (16)	Hex	Square	Hex	Square	Hex
	$1.74239 \cdot 10^8$	$1.74536 \cdot 10^8$	131072	128856	45.0	48.6
	$2.02461 \cdot 10^8$	$2.01979 \cdot 10^8$	131072	128856	381.7	337.4
	$1.66482 \cdot 10^8$	$1.65938 \cdot 10^8$	131072	66528	303.5	71.8
	$3.32766 \cdot 10^7$	$3.34633 \cdot 10^7$	131072	128856	43.7	42.9
	$1.44697 \cdot 10^8$	$1.44208 \cdot 10^8$	131072	66528	179.9	39.6

Table 10.7: Evaluation of hexagonal meshes for image segmentation on the same set of images as in Table 10.1. Parameters: $\rho = 10^4$, $\sigma = 10^5$.

number of line pairs or edges look essentially the same. The 8-connected grid converges quickly, but to a suboptimal objective value. The hexagonal mesh consistently outperforms the 16-connected grid. With a very large number of regions, the 16-connected grid would probably achieve a lower objective value than the hexagonal, due to it having 2 more possible straight lines. I have not been able to observe this in practice, though, due to the large memory requirements.

Table 10.7 shows experiments for a couple of images. The difference is not great, but in some cases the hexagonal meshes can save a significant amount of computation time. This experiment used another very simple score to rank the regions: counting the number of pixels with positive and negative data term pixels and then taking the minimum of the two.

Adaptive Meshes. The effect of adaptive meshes can be evaluated in a number of ways. Firstly, figure 10.10 show the visual quality of the segmentation for regular and adaptive 16-connected meshes with the same number of regions. The fact that the adaptive mesh achieved a smoother curve is also reflected in the lower optimal objective function value. Figure 10.11 shows results for 8-connected meshes with similar visual improvement.

Solving the same segmentation problem a large number of times for different number of regions evaluates the performance more quantitatively. Figure 10.12 shows the optima for the different number of regions and the

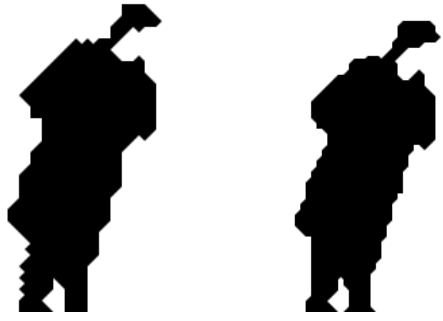
(a) Regular mesh **(b)** Adaptive mesh

Figure 10.10: Results with regular and adaptive 16-connected mesh. The number of regions used were 32,768 in both cases and the number of edge pairs were 291,664 and 285,056, respectively. The adaptive mesh gives a smoother curve and correctly includes the hand of the camera man. The optimum for the regular mesh was $2.470 \cdot 10^8$ and $2.458 \cdot 10^8$ for the adaptive. This experiment used $\rho = 30000$ and $\sigma = 1000$.

two types of meshes. The adaptive mesh converged to what probably is the optimum for that connectivity, while the regular mesh did not. The regular mesh would have required more than 20 times more regions to achieve the same objective value. In addition to this single-image experiment, table 10.8 shows results for five other images. It is not surprising to see simpler images benefit more from adaptive meshes than more complicated ones.

10.6 Dual Decomposition

Chapter 7 described a dual decomposition scheme, where the domain is split into two pieces which are solved separately and constrained to be equal on an overlap; see figure 7.1 on page 87. The same method can be used to split curvature linear programs to reduce the memory requirements of the linear solver. While splitting an adaptive mesh into two or more parts takes some care, splitting a regular square mesh is relatively straightforward. A specialized solver for the original problem must be able to solve the subproblems as well. Therefore, some care has to be taken to make sure all edge pairs are counted the correct number of times in and around the overlap. It is easy to test if the split has been made correctly—the sum of the



(a) Regular mesh **(b)** Adaptive mesh

Figure 10.11: Results with (a) regular and (b) adaptive 8-connected mesh. The optimum for the regular mesh was $2.517 \cdot 10^8$ and $2.481 \cdot 10^8$ for the adaptive. This experiment used $\rho = 30000$ and $\sigma = 1000$.

minima in the left and right part has to match the minimum of the original problem exactly. The solutions to the two problems are constrained to be equal for all overlapping regions. The updating scheme for graph cuts in section 7.2.3 on page 89 also works for curvature problems.

The difference in peak memory usage was evaluated by experiments using a 128×128 mesh. The memory reduction with dual decomposition was significant; see table 10.9. The achieved memory reduction was slightly less than 50% since the intermediate states of the solvers are stored between iterations to increase execution speed. The total time required by dual decomposition (single-threaded) was usually slightly shorter than solving the original problem, but the main benefit is the reduced memory consumption.

10.7 Convex Shape Priors

An interesting prior for image segmentation is to require the object to be convex. I find it interesting because it is a prior without any tuning parameters to choose. Raphael and Geman (1997) introduced a method based on dynamic programming. The method required the prior knowledge of a point within the object to be segmented. Curiously, their experiments do not produce convex objects and the authors do not comment on this. A subsequent publication (Raphael, 2001), however, has corrected this. The dynamic programming problem is prohibitively large and a coarse-to-fine

Image	Objective value		Number of regions		Runtime (s)	
	Regular	Adaptive	Regular	Adaptive	Regular	Adaptive
	$1.6846 \cdot 10^8$	$1.68423 \cdot 10^8$	131072	12800	16.7	4.4
	$1.48637 \cdot 10^8$	$1.4852 \cdot 10^8$	131072	80000	784.7	964.7
	$1.217 \cdot 10^8$	$1.21622 \cdot 10^8$	131072	72032	787.1	722.2
	$2.50306 \cdot 10^7$	$2.50257 \cdot 10^7$	131072	12800	54.8	5.6
	$8.96085 \cdot 10^7$	$8.8855 \cdot 10^7$	131072	48032	167.1	111.7

Table 10.8: Evaluation of adaptive 16-connected meshes for image segmentation on the same set of images as in Table 10.1. Parameters: $\rho = 100$, $\sigma = 10^4$.

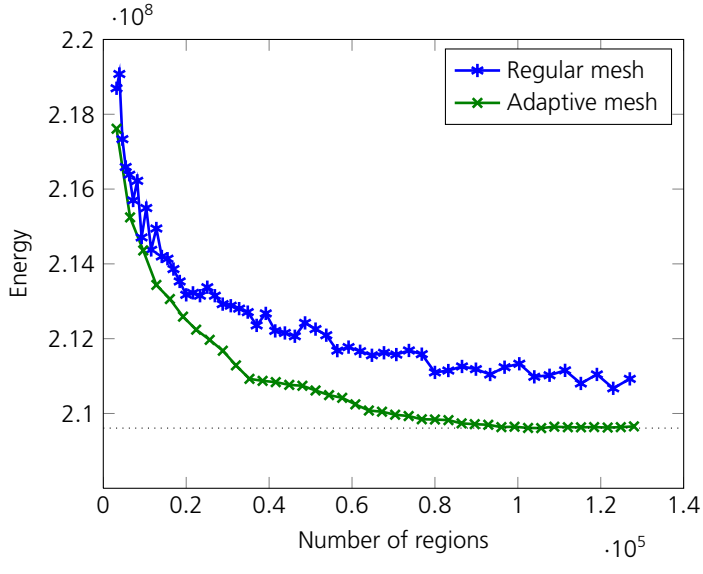
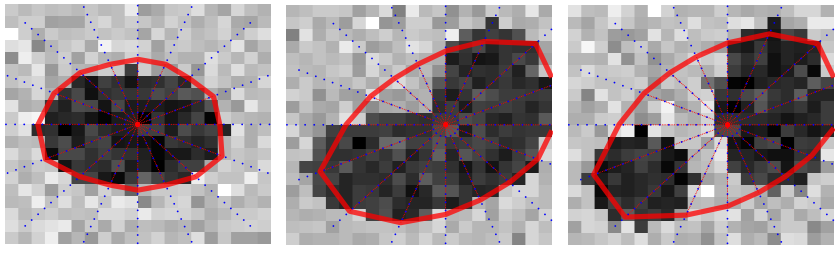


Figure 10.12: Optimum vs. the total number of regions for a square 16-connected mesh. To get the same accuracy as the finest parts of the adaptive mesh, the regular mesh would need $21 \cdot 10^5$ regions. In contrast, the adaptive mesh converged using about $1 \cdot 10^5$ regions. This experiment used $\rho = \sigma = 10000$.

	Peak memory usage (MB)
Original problem	468
Dual decomposition (2 parts)	279

Table 10.9: Peak memory usage for a 128 × 128 homogenous 8-connected mesh. The reduction is not as good as 50% because the solvers cache the solutions and system matrices between iterations.



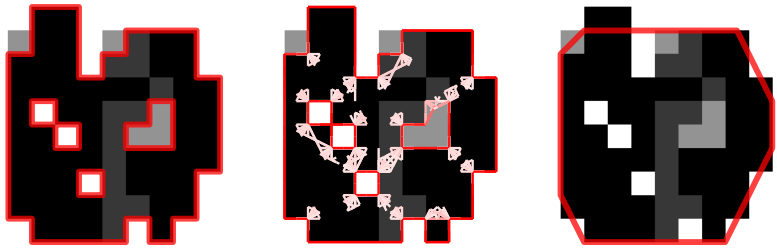
(a) 180 seconds **(b)** 425 seconds **(c)** 1393 seconds

Figure 10.13: Implementation of the method by Raphael (2001). The red dots are the specified center points and the blue dots are the discretized radii.

heuristic needs to be used to solve it (global optimality is preserved). Figure 10.13 shows my implementation of their method on very small images. The running time is dependent on how strongly the data term supports a convex object. Felzenszwalb and McAllester (2007) have improved the running times by using a more sophisticated search technique. The framework by Strekalovskiy and Cremers (2011) can handle a convex shape prior without any seed point. Their method is also very computationally expensive; using it requires a GPU to efficiently solve the optimization problem.

The framework described in this chapter seems very suitable for convex shape priors at a first glance. Indeed, when one walks along the positively oriented boundary of a convex set one will never turn left. Each non-straight pair of line segments has two variables associated with it: y_{ij} and y_{ji}. If all variables representing non-convex boundary segments are removed, only convex shapes will be possible.

Unfortunately, this approach has several issues. First, nothing guarantees that a *single* convex object is produced—a very noisy image can produce



(a) 0.28 seconds **(b)** 49 seconds **(c)** 371 seconds

Figure 10.14: Segmentation without any length or curvature regularization. **(a)** Without a convex prior (thresholding). **(b)** Removing all non-convex pairs makes many line pair variables in the linear programming relaxation fractional. **(c)** With the additional constraint that the number of objects equal one (see text).

hundreds of small objects. Second, figure 10.14b shows what happens when the non-convex pairs are removed. The linear programming relaxation is then no longer integral and the solution is very far from a convex object. While there is nothing wrong with the formulation (and, indeed, an integer programming solver will give the correct result), it is virtually useless in practice. Third, the time required to solve the linear program increased more than 100 times for the very small problem in figure 10.14.

The solution to the first two issues is to constrain the number of objects. The solution will consist of several closed curves, each of whose angles sum up to 2π. By adding a single constraint, the number of objects can be constrained to one:

$$\sum_{\text{all pairs } (i,j)} \alpha_{ij} b_{ij} = 2\pi. \tag{10.16}$$

Figure 10.14c shows the result for the small test image. In my experience, the relaxation is always tight. However, the third issue not only remains, but is amplified: the time to solve the relaxation with the new constraint added goes up by almost *another* order of magnitude. Adding the convex shape prior works, but the increased difficulty when removing almost half of the variables and adding a single constraint is surprising.

The use of a commercial solver can reduce the computation time significantly. To solve the linear program in figure 10.15, Gurobi's barrier method

Figure 10.15: Segmentation with a convex prior without any length or curvature regularization on a 16-connected 32 × 32 mesh. The time required to compute this (globally optimal) solution exceeded 4 hours.

required 2755 seconds to and Clp's dual simplex method required more than 4 hours. The barrier method is not suitable for all curvature problems, though, as it uses more memory.

10.8 Solution Refinement

Because the grids used for curvature optimization in the previous chapters have only a small number of directions, the result often looks jagged, even when a 16-connected grid is used. I have been asked several times when talking about this discrete approach to curvature whether local optimization would be able to fix this and produce a more visually pleasing solution. My answer has been that this would definitely be possible and that it would be interesting to try. Hence, it is almost fair to say that this short section has been written by popular demand.

10.8.1 Detaching the Mesh Points

So far, this chapter has been about solving (10.2) on page 138 over a fixed mesh. The linear programming procedure results in a number of simple,

closed curves $\boldsymbol{\Gamma}^{(1)}, \ldots \boldsymbol{\Gamma}^{(n)}$, which may be nested, depending on the topology of the foreground. The refinement stage post-processes each of these curves individually.

Each curve can be seen as a 2m-dimensional vector of its point coordinates: $\boldsymbol{\Gamma} = (\boldsymbol{p}^{(1)}, \ldots, \boldsymbol{p}^{(m)}) \in \mathbf{R}^{2m}$. The objective function for the curve can then be written as

$$E(\boldsymbol{\Gamma}) = \sum_{k=1}^{m} \tilde{g}\left(\boldsymbol{p}^{(k+1)}, \boldsymbol{p}^{(k)}\right) + \rho \sum_{k=1}^{m} \left\| \boldsymbol{p}^{(k+1)} - \boldsymbol{p}^{(k)} \right\| + \sigma \sum_{k=1}^{m} \tilde{\kappa}\left(\boldsymbol{p}^{(k+1)}, \boldsymbol{p}^{(k)}, \boldsymbol{p}^{(k-1)}\right), \quad (10.17)$$

where $\boldsymbol{p}^{(m+1)} = \boldsymbol{p}^{(1)}$ and $\boldsymbol{p}^{(0)} = \boldsymbol{p}^{(m)}$. The first sum is exactly equal to the integral in (10.2) by using the technique in section 10.2.1. The function $\tilde{g}$ is the computation of the line integral of G between two points. The function $\tilde{\kappa}$ is the computation of the discrete curvature for the line pair defined by the three points.

The sum of $E(\boldsymbol{\Gamma}^{(k)})$ for each k is exactly the objective function in (10.2), but instead formulated as a function of all corner points of the curves. The refinement step attempts to minimize this quantity locally. Since the data term g is discontinuous (piecewise constant over each image pixel), $\tilde{g}$ will not be differentiable everywhere. The same is true for the distance $\|\cdot\|$. Two simple approaches are possible:

- Coordinate descent for a single $\boldsymbol{p}^{(k)}$ in a specified direction while keeping all other points fixed.
- Minimizing $t \mapsto \boldsymbol{\Gamma} + t\boldsymbol{\Lambda}$ for some specified direction $\boldsymbol{\Lambda}$.

In practice, alternating between the two works better than using either exclusively. Computing a descent direction can be done in two ways:

- (easy way) Computing $\frac{E(\boldsymbol{\Gamma}+h\boldsymbol{e}^{(k)})-E(\boldsymbol{\Gamma})}{h}$ for small h for all coordinates k.
- (smart way) Using automatic differentiation to simultaneously compute a "gradient" when evaluating E (Nocedal and Wright, 2006). This procedure is often quite easy when working with object-oriented

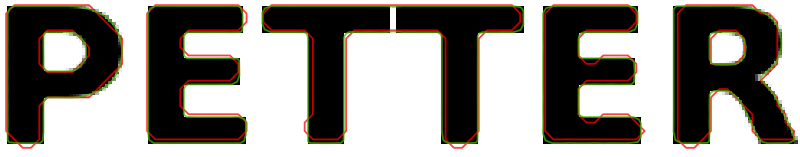
languages like C++. I used the implementation of the open-source library FADBAD++.

Both methods gave very similar results and the choice did not affect the final solution noticeably.

It is important that the topology of the interior of each curve stays the same; for example, self-intersections must be avoided. Currently I do not have a sophisticated way of doing this, other than preventing each curve point from moving too much from its initial position. Large movements are an indication of something gone wrong; after a self-intersection the outside becomes the inside. Self-intersections are also avoided by small enough steps in each iteration.

10.8.2 Experiments

The time required for the complete refinement process is negligible—less than a second for all problems in this chapter. Even if a very large mesh is used, the solution curves will only cover a small part of that mesh. Figures 10.16 and 10.17 show two examples. The large decrease in objective function value (from 8.1×10^7 to 4.7×10^7) is typical.

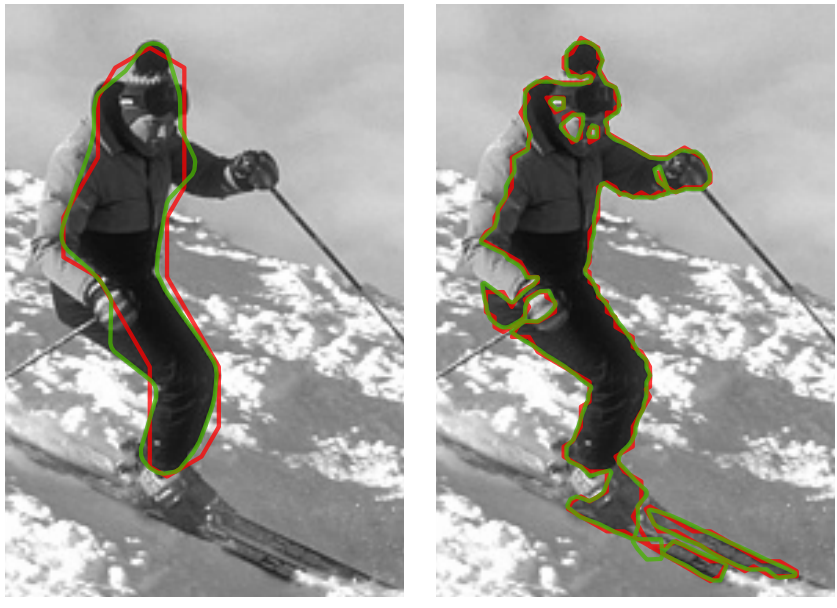


(a) Segmented image.

(b) Close-up on the "R."

(c) Same close-up, but with the curvature power equal to 1.

Figure 10.16: Refinement of a curvature segmentation on an 8-connected grid. The global discrete solution is shown in red and the refined solution in green. The objective function decreased from 8.1×10^7 to 4.7×10^7 during refinement.



(a) 32 × 32 hexagonal mesh and high regularization.

(b) 64 × 64 hexagonal mesh and low regularization.

Figure 10.17: Refinement of a curvature segmentation on a (12-connected) hexagonal grid. The global discrete solution is shown in red and the refined solution in green.

Chapter 11

Surface Completion and Segmentation with Curvature

The previous chapter showed that it is in many cases possible to minimize functionals involving the curvatures of plane curves. This chapter will extend the framework to surfaces in three dimensions. The task is to minimize the following function:

$$E(R) = \int_R g(\boldsymbol{x})\, d\boldsymbol{x} + \int_{\partial R} \left(\rho + \sigma\kappa(\boldsymbol{x})^2\right) dA(\boldsymbol{x}), \qquad (11.1)$$

where R is a volume with boundary ∂R. Here $\kappa(\boldsymbol{x})$ is the mean curvature of the surface ∂R at $\boldsymbol{x}$. $g(\boldsymbol{x})$ is the data term, which may take many different forms depending on the application. It is typically present in segmentation problems and not present in surface completion problems. ρ and σ controls the amount of area and curvature regularization, respectively.

In differential geometry, functions of the type in (11.1) have been studied for a long time. The functional is known as the Willmore energy (Willmore, 1965). It gives a quantitative measure of how much a given surface deviates from a round sphere. Local descent techniques have been derived for minimizing (11.1) (Hsu et al., 1992), but they are very dependent on a good initialization. This chapter will attempt to create a framework for minimizing these functions *globally*.

11.1 Curvature of Surfaces

Each facet in a 3D mesh is associated with a variable $\boldsymbol{y} = (y_1, \ldots, y_{2n})$ of areas $\boldsymbol{a} = (a_1, \ldots, a_{2n})$. There are twice as many variables as facets, because each facet is associated with two variables, one for each orientation.

The two are distinguished by (arbitrarily) assigning a normal to each face in the mesh. The optimization problem for surface completion with area regularization is

$$\begin{aligned} \underset{\boldsymbol{y}}{\text{minimize}} \quad & \rho \boldsymbol{a}^{\mathrm{T}} \boldsymbol{y} \\ \text{subject to} \quad & \mathsf{B}\boldsymbol{y} = 0 \\ & \boldsymbol{y} \in \{0,1\}^{2n} \\ & y_k = 1,\ k \in K. \end{aligned} \tag{11.2}$$

K is the set of facets that are supposed to be part of the minimal surface a priori. The matrix B is defined by Grady (2010) as:

$$\mathsf{B}_{e,y_i} = \begin{cases} +1, & \text{if edge e borders } y_i \text{ with coherent orientation} \\ -1, & \text{if edge e borders } y_i \text{ with incoherent orientation} \\ 0, & \text{otherwise.} \end{cases} \tag{11.3}$$

This section extends this formulation to support curvature by introducing face pairs. Each pair of facets in the mesh with an edge in common are associated with two variables $\{y_{ij}\}$ (one for each orientation). Enforcing consistency between the face variables and the variables corresponding to the pairs of faces can be done with linear constraints:

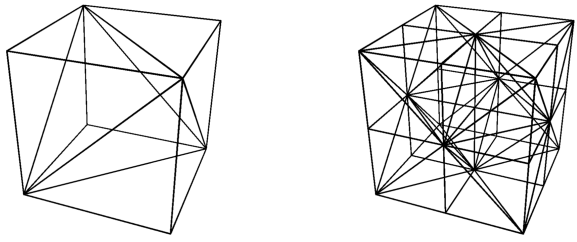
Surface continuation constraints. For each oriented facet k and each one of its edges e the following constraint prevents the surface from ending abruptly:

$$y_k = \sum_{(i,j) \text{ with edge } \boldsymbol{e}} d_{k,ij} y_{ij}. \tag{11.4}$$

The sum is over all pairs i, j with edge $\boldsymbol{e}$ in common. The indicator $d_{k,ij}$ is 1 if facet k is part of the pair (i, j).

Having introduced the facet pairs, we can follow Wardetzky et al. (2007) and associate them with a cost b_{ij}, approximating the squared mean curvature:

$$b_{ij} = \frac{3\|\boldsymbol{e}_{ij}\|^2}{2(a_i + a_j)} \left(2 \cos \frac{\theta_{ij}}{2}\right)^2, \tag{11.5}$$



(a) One unit cube.

(b) Eight unit cubes in a $2 \times 2 \times 2$ mesh.

Figure 11.1: Each unit cube is split into 5 tetrahedrons. This is the type of mesh used for our experiments in 3D. When stacking several, every other cube has to be mirrored in order to fit.

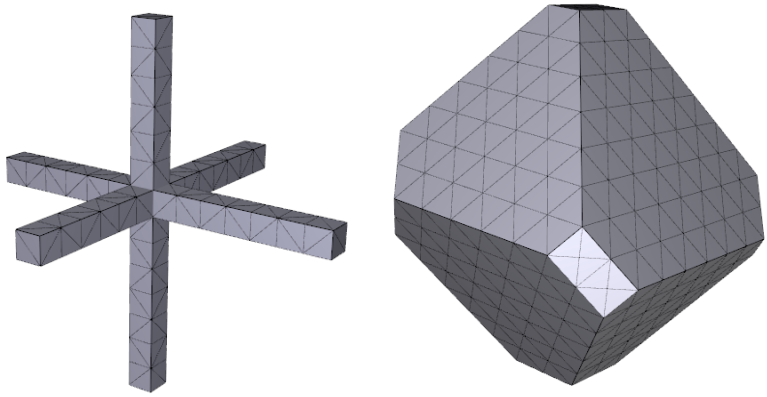
where θ_{ij} is the dihedral angle between the two facets in the pair. Here, $\|\boldsymbol{e}_{ij}\|$ is the length of their common edge. The objective function is then $\rho \sum_i a_i y_i + \sigma \sum_{i,j} b_{ij} y_{ij}$, subject to the constraints in (11.2) and (11.4). This approximation is far from perfect; for example, it will not give the correct approximation for saddle points. However, it measures how much the surface bends and fulfills a couple of requirements listed by Wardetzky et al. (2007): it is invariant under similarity transformations and vanishes for flat surfaces.

Segmentation, as opposed to surface completion, requires variables for each volume element to incorporate the data term. Let x_i be variables associated with the volume elements. Additional consistency constraints are then required:

Volume continuation constraints. For each facet k,

$$\sum_i \beta_{k,i} y_i + \sum_i g_{k,i} x_i = 0, \tag{11.6}$$

where $\beta_{k,i}$ indicates whether the facet y_i is positively or negatively incident w.r.t. the chosen face normal. The variable $g_{k,i}$ is 1 if the volume element x_i is positively incident (the face normal points towards its center), -1 if it is negatively incident and 0 otherwise. Both sums have two non-zero terms.



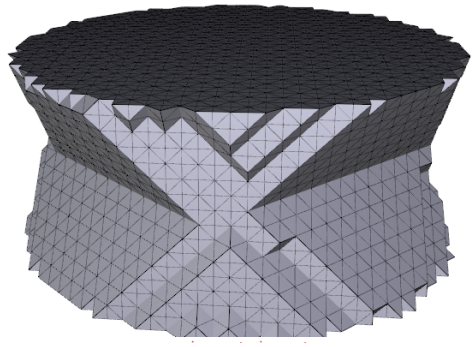
(a) Area regularization (b) Curvature regularization

Figure 11.2: Segmentation using a 16 × 16 × 16 mesh with area and curvature regularization and volume element variables. The data term and the optimal surface using area regularization coincide. Note that any sphere containing the cross is a minimizer for the continuous problem.

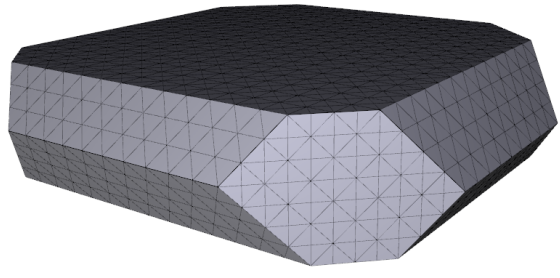
11.2 Experiments

The three-dimensional experiments used a mesh in which each unit cube is split into 5 tetrahedrons; see figure 11.1. The first experiment used a set K consisting of two circular surfaces at $z = 0$ and $z = z_{\max}$, with nothing in between. The analytic solution with area penalty is the catenoid (Pressley, 2010). Figure 11.3a shows the discrete version obtained with $\rho = 1$ and $\sigma = 0$. If instead the mean curvature is chosen as the regularizer, the optimal surface instead bends outwards. The solution to this problem is shown in figure 11.3b and is the global optimum, since all variables ended up integral in the LP relaxation of (11.2). Just as in the previous chapter, all linear programs were solved by Clp.

Another experiment used variables for the volume elements. The data term was a 3D 'cross' where the volume elements were forced to be equal to 1, whereas the volume elements at the boundary were forced to be 0. The optimal segmentation when the area was minimized coincided with the data term and is shown in figure 11.2a. When instead minimizing the curvature the optimal segmentation should resemble a sphere, which is observed in figure 11.2b.



(a) Area regularization on a 40 × 40 × 15 mesh (491k variables, 398k constraints and 447 seconds). A 25 × 25 × 7 mesh required 91k variables and 5 seconds.



(b) Curvature regularization on a 25 × 25 × 7 mesh (637k variables, 441k constraints and 178 seconds).

Figure 11.3: Surface completion with area and curvature regularization. Two flat, circular surfaces at the top and bottom were fixed to 1. One surface bends inwards to approximate a catenoid and the other correctly bends outwards to minimize the curvature.

11.3 Conclusion

This chapter has introduced constraints for 3D surface completion and segmentation. Experiments are encouraging with exclusively globally optimal solutions. To my knowledge, this is the first time the mean curvature of surfaces has been optimized globally. The next step would be to apply this method to e.g. the partial surfaces obtained by stereo estimation algorithms. Another line of further research is how to be able to cope with finer discretizations of the 3D volume and, of course, to speed up the computation in general. The adaptive meshes and and refinement from the previous chapter will surely have to play a part in this.

Part IV

Applications

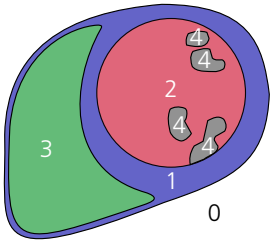
Chapter 12

Multiple Region Segmentation

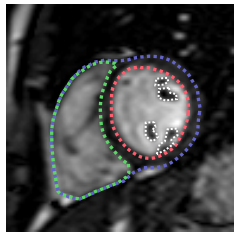
The field of medical imaging is full of challenging segmentation tasks. The aim of this chapter is to segment multiple regions with a model that encompasses both the underlying appearance and shape of the different regions as well as their geometric relationships. This is often overlooked in present methods. For example, many successful approaches to cardiac segmentation concentrate on segmenting the left ventricle (LV) as this part is the most interesting for diagnostic purposes. Still, quantifiable information about the cardiac function can be gained from segmenting the right ventricle (RV) as well. The framework presented in this chapter allows for the construction of a joint model of the whole heart, which is general enough to support other applications such as lung segmentation. The final result is improved compared to segmenting the parts independently.

The main contribution is a multi-region segmentation framework with good tractability. The framework builds on the multi-region scheme presented by Delong and Boykov (2009), who showed that geometric relationships, for example, when one object is included in another, can be modeled and globally optimized via graph cuts. The key property that makes this possible is that the resulting minimization problem is submodular. Not all geometric relationships, however, are submodular. Delong and Boykov used roof duality for these harder problems, but that becomes too memory intensive for large three-dimensional problems. Instead, I will describe a dual approach, which is very similar to what I used in chapter 7 for parallelization.

Another contribution is the evaluation of the optimization framework for medical segmentation problems. The cardiac segmentation model is applied to publicly available data and the optimization framework is compared to roof duality in terms of memory and speed.



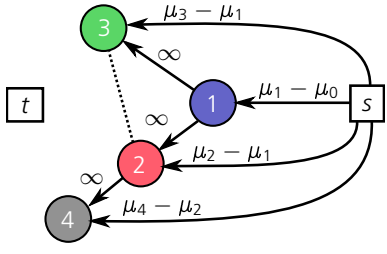
(a) Four-region model



(b) MR view

r_p	$\boldsymbol{x}_p$
0	(0, 0, 0, 0)
1	(1, 0, 0, 0)
2	(1, 1, 0, 0)
3	(1, 0, 1, 0)
4	(1, 1, 0, 1)

(c) Representation



(d) Graph

Figure 12.1: **(a)** A constructed short-axis view showing the model of the heart. Region 0 is the background, region 1 contains myocardium *and* the left and right ventricular cavities. Region 2 is the left ventricular cavity and region 3 the right ventricular cavity. Region 4 is the papillary muscles of the left ventricle. **(b)** An example of a slice from a short-axis image acquired with MRI where all four regions have been manually delineated. **(c)** The boolean representation of the four regions reflect their geometric relationships. **(d)** Graph construction for one voxel. The circled number corresponds to a node associated with the region number. The directed arrows are the directed edges in the graph.

12.1 Multi-Region Framework

Before introducing the general framework, figure 12.1 presents an example of a construction using the framework. The geometric inclusion constraints are encoded as edges in a graph with infinite weight.

Let $\mathcal{R}$ be the set of region labels (excluding the background) and let $\mathcal{P}$ be the set of voxel indices. Each voxel p should be assigned a region label $r \in \mathcal{R}$. The segmentation variables are $\boldsymbol{x} \in \mathbf{B}^{|\mathcal{R}|\times|\mathcal{P}|}$, where $\mathbf{B} = \{0, 1\}$ and $\boldsymbol{x}$ is indexed as x_p^r with $r \in \mathcal{R}$ and $p \in \mathcal{P}$. The partial indexing $\boldsymbol{x}^r$ represents all boolean variables associated with region r and $\boldsymbol{x}_p$ represents all boolean variables associated with voxel p. Each voxel in the image is represented by $|\mathcal{R}|$ boolean variables, which will make it possible to directly encode geometric relationships between regions, like inclusion and exclusion. Figure 12.1c shows the correspondence between r and $\boldsymbol{x}_p$ for the cardiac model. Here, the fact that regions 2 and 3 should be contained in region 1 is encoded in the boolean representation by the fact that the first boolean variable is set to one. Similarly, region 4 is contained in both region 1 and region 2 and consequently, the first two boolean variables are equal to one.

The objective function the framework minimizes is:

$$E(\boldsymbol{x}) = D(\boldsymbol{x}) + V(\boldsymbol{x}) + W(\boldsymbol{x}), \tag{12.1}$$

whose three components are, in order, the unary terms, the pairwise terms (regularization) and the geometric interaction terms. For every voxel p, the unary terms introduce a cost for each labeling of $\boldsymbol{x}_p$:

$$D(\boldsymbol{x}) = \sum_{p\in\mathcal{P}} \sum_{r\in\mathcal{R}} D_p^r\left(x_p^r\right). \tag{12.2}$$

The pairwise terms use a connectivity $\mathcal{N}$ to favor smooth and correctly located boundaries:

$$V(\boldsymbol{x}) = \sum_{p,q\in\mathcal{N}} \sum_{r\in\mathcal{R}} V_{p,q}^r\left(x_p^r, x_q^r\right). \tag{12.3}$$

The geometric interaction terms associate a cost with labeling voxel p in region i with different labelings for voxel q in region j. These terms are used either to attract or repel different regions to each other:

$$W(\boldsymbol{x}) = \sum_{p,q\in\mathcal{N}} \sum_{\substack{i,j\in\mathcal{R}\\ i\neq j}} W_{p,q}^{i,j}\left(x_p^i, x_q^j\right). \tag{12.4}$$

For any voxel p, the probability of this voxel to belong to region r is $\mathrm{P}\left(x_p^r = 1\right)$. This probability will have to be estimated from image data during a training phase.

12.1.1 Unary Terms

The unary terms are constructed from the probability of each voxel belonging to any of the regions. Define

$$\mu_r(p) = -\log\left(\mathrm{P}\left(x_p^r = 1\right)\right), \tag{12.5}$$

for voxel p and region r.

A region i is parent to a region j if region j is forced to be contained inside i directly. Regions not forced to be contained inside any specific regions have, by definition, the background (region 0) as parent.

Consider any region r and let $\mathcal{G}_r$ denote the set of all parents to r, then the unary term should be constructed as

$$D_p^r\left(x_p^r\right) = x_p^r\left(\mu_r\left(p\right) - \sum_{g\in\mathcal{G}_r}\mu_g\left(p\right)\right), \tag{12.6}$$

for all $p \in \mathcal{P}$ and $r \in \mathcal{R}$. Examples of these constructions are given in figures 12.1 and 12.6. The general case above is of limited interest, however, and the reason it works is most easily explained through an example:

EXAMPLE 12.1 Consider the cardiac model in figure 12.1. According to (12.6)

$$\begin{aligned}\sum_{r=1}^{4} D_p^r\left(x_p^r\right) = {} & x_p^4\left(\mu_4\left(p\right) - \mu_2\left(p\right)\right) + x_p^3\left(\mu_3\left(p\right) - \mu_1\left(p\right)\right) \\ & + x_p^2\left(\mu_2\left(p\right) - \mu_1\left(p\right)\right) + x_p^1\left(\mu_1\left(p\right) - \mu_0\left(p\right)\right).\end{aligned} \tag{12.7}$$

Now consider a voxel assigned to region 4 from the model. Then $x_p^4 = 1$, $x_p^2 = 1$, $x_p^1 = 1$ and $x_p^3 = 0$. It follows that

$$\begin{aligned}\sum_{r=1}^{4} D_p^r\left(x_p^r\right) &= \left(\mu_4\left(p\right) - \mu_2\left(p\right)\right) + \left(\mu_2\left(p\right) - \mu_1\left(p\right)\right) + \left(\mu_1\left(p\right) - \mu_0\left(p\right)\right) \\ &= \mu_4\left(p\right) - \mu_0\left(p\right).\end{aligned} \tag{12.8}$$

The reason this construction works is that boolean variables with parents are linked to their parents by the geometric interaction term. The final cost for assigning a voxel to a region is added up like a telescopic sum resulting in $\mu_r - \mu_0$ for each region r.

12.1.2 Pairwise Terms

The regularization weights are chosen differently for each region in a method related to the discussion by Grady and Jolly (2008). For each region i the pairwise terms are:

$$V_{p,q}^r\left(x_p^r, x_q^r\right) = \frac{w_{p,q}}{1 + \beta\left(\mathrm{P}\left(x_p^r = r\right) - \mathrm{P}\left(x_q^r = r\right)\right)^2}, \tag{12.9}$$

where β can be used to tune the regularization. The neighborhood $\mathcal{N}$ for the regularization is in all experiments 18-connectivity. The multipliers $w_{p,q}$ give different weights to different types of edges. One common choice is $w_{p,q} = 1/\operatorname{dist}(p,q)$; however, an arguably more correct way is described by Boykov and Kolmogorov (2003) and is based on solid angles.[1] The fact that MRI has anisotropic resolution is very important to take into consideration both when calculating the distance between voxels and when using the method by Boykov and Kolmogorov (2003).

12.1.3 Geometric Interaction Terms

Some regions should be contained inside other regions, while other regions should be forced apart. This is controlled by the geometric interaction terms. Recall the definition of submodularity on page 21 and that submodular functions can be transformed into minimum cut problems and are thus easily optimizable.

Submodular Interaction Terms. Suppose region j should be contained inside region i. This is accomplished by setting

$$W_{p,p}^{i,j}(0,1) = \infty \text{ for all } p \in \mathcal{P}. \tag{12.10}$$

[1] Computing the weights correctly is not completely trivial and requires a Voronoi tessellation of the unit sphere. I used the software package by Burkardt (2010) for this task. See also the book edited by Goodman and O'Rourke (1997) and the article by Renka (1997).

This term is clearly submodular. Enforcing a margin (minimum distance) between two regions is also possible by setting

$$W_{p,q}^{i,j}(0,1) = \infty \text{ for all } p \in \mathcal{P}, \tag{12.11}$$

where q is taken in some neighborhood $\mathcal{N}_p$ of p. As an example, let $\mathcal{N}_p$ be the 8-connected neighborhood of p. Now region j will not only be forced to be inside region i, it will be forced to be slightly smaller than region i.

Non-Submodular Interaction Terms. Similarly, if region i should be excluded from region j, set

$$W_{p,p}^{i,j}(1,1) = \infty \text{ for all } p \in \mathcal{P}. \tag{12.12}$$

This term is non-submodular. In some special cases the boolean variables can be transformed to allow for a submodular construction with exclusion constraints (Delong and Boykov, 2009). However, this is not possible for either model in this chapter.

12.2 Solving the Optimization Problem

The standard approach for minimizing non-submodular functions of this type is to use roof duality (Delong and Boykov, 2009). Using supergradients is also possible and results in a fast and memory-efficient method.

Let $E'(\boldsymbol{x})$ be the objective function without the non-submodular terms above. It will then be easy to minimize. The non-submodular terms can be written $\boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}$. Adding this constraint results in the new problem

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & E'(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}. \end{aligned} \tag{12.13}$$

It can be solved as an integer programming problem, but that is not tractable due to the large number of variables. Instead, look at the dual problem:

$$\begin{aligned} \underset{\boldsymbol{\lambda}}{\text{maximize}} \quad & d(\boldsymbol{\lambda}) = \max_{\boldsymbol{\lambda}} \left(\min_{\boldsymbol{x}} \left(E'(\boldsymbol{x}) + \boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{g}(\boldsymbol{x}) \right) \right) \\ \text{subject to} \quad & \boldsymbol{\lambda} \geq 0, \end{aligned} \tag{12.14}$$

where $d(\boldsymbol{\lambda})$ is the dual function. Let d^* denote the optimal value for (12.14) and p^* the optimal value for (12.13). Then $d^* \leq p^*$. The constraint on $\boldsymbol{\lambda}$ requires using the projected supergradient method (section 2.2.2 on page 16) to solve the dual problem. Just as in chapter 7, evaluating d consists of solving a minimum cut problem.

12.3 Cardiac Segmentation

The heart below the atrioventricular plane is modeled by four different regions as shown in figures 12.1a and b. The joint model describes both the geometry of the different regions and their appearances in the MR images. Region 1 (the myocardium) contains both region 2 and region 3—this is modeled by the use of geometric interaction terms as $W^{1,2}_{p,p}(0,1) = \infty$ and $W^{1,3}_{p,p}(0,1) = \infty$, for all $p \in \mathcal{P}$. Furthermore, the left ventricular papillary muscles must be inside the left ventricle. This is modeled as $W^{2,4}_{p,p}(0,1) = \infty$ for all $p \in \mathcal{P}$; see figure 12.1d. Excluding region 2 from 3 is taken care of by terms of the form $W^{2,3}_{p,p}(1,1) = \infty$. These terms are not submodular and they can be handled by either supergradient ascent or roof duality. For the supergradient approach, the (primal) optimization problem can be formulated as:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & E'(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{x}^2 + \boldsymbol{x}^3 \leq 1, \end{aligned} \tag{12.15}$$

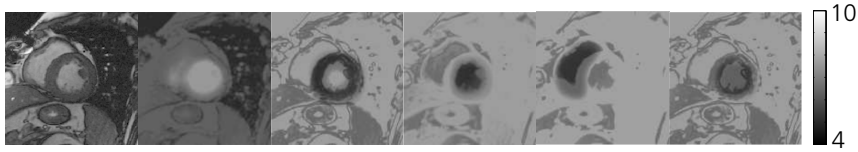
where $E'(\boldsymbol{x})$ is the objective function without the non-submodular terms.

The unary terms construction as given in figures 12.1c and d results in:

$$\begin{aligned} D^1_p(1) &= \mu_1(p) - \mu_0(p), & D^2_p(1) &= \mu_2(p) - \mu_1(p), \\ D^3_p(1) &= \mu_3(p) - \mu_1(p), & D^4_p(1) &= \mu_4(p) - \mu_2(p) \end{aligned} \tag{12.16}$$

and $D^r_p(0) = 0$ for all $r \in \mathcal{R}$ and $p \in \mathcal{P}$.

The spatial probability is split into four categories: left ventricle, right ventricle, myocardium and background. Similarly, the intensity is split into three categories: blood, muscle and background. The probability for each region is then calculated with the assumption that the spatial and intensity distributions are independent. An example of the final μ_r's can be found in figure 12.2. The spatial distribution is estimated by first resizing each image



(a) Slice (b) μ_0 (c) μ_1 (d) μ_2 (e) μ_3 (f) μ_4

Figure 12.2: Example of μ_r for the slice shown in (a). Recall that $\mu_r(p) = -\log\left(\mathrm{P}\left(x_p^r = 1\right)\right)$. A lower intensity corresponds to higher probability.

in the training data to the same size by linear interpolation. Then a binary mask is constructed for each category and and each image. The masks are enlarged and smoothed and then they are all added together to the final probability mask. The intensity distribution for each region is estimated by collecting all intensities from the examples in the training data.

The user selects which slices to be segmented and selects a center point of the right and left ventricle in *one* slice. The two center points are used to roughly align the hearts to get good spatial statistics. The algorithm can handle slices lacking some of the regions.

All ground truth data me and my colleagues have used only have delineations for the left ventricular epicardium. The method in this chapter does not have this restriction and segments the full myocardium. All myocardium which is not part of the left ventricular epicardium must be removed to compare the results to the ground truth. To do this, the thickness of the septum is approximated as the shortest distance between the left and right ventricles in the resulting segmentation. Then the outlying myocardium is removed based on this thickness approximation; see figure 12.1a. Since assuming the two ventricles to be convex is a reasonable model, the final segmentation is the convex hull in each slice.[2] The regularization can sometimes make the segmentation miss the apical slice, whose location is provided by the user. When this happens, either the segmentation from the same slice at another time step or, if it is not available, the segmentation from a more basal slice is shrunk and fitted at the bottom.

[2] See section 10.7 on page 157 for a general discussion about convex shape priors.

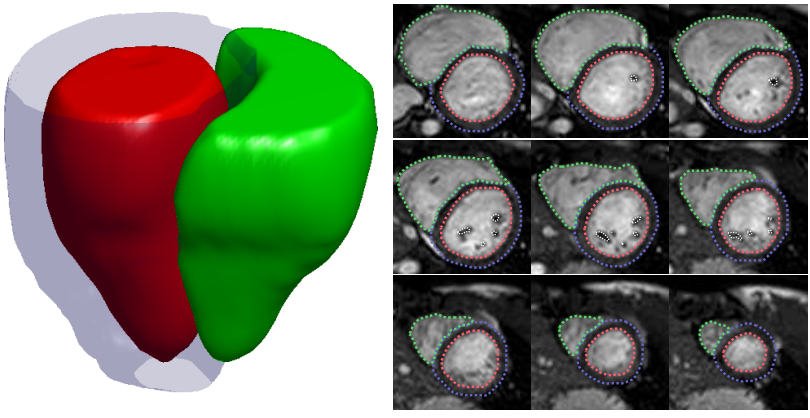


Figure 12.3: Example segmentation from Lund, 3D rendering and nine slices.

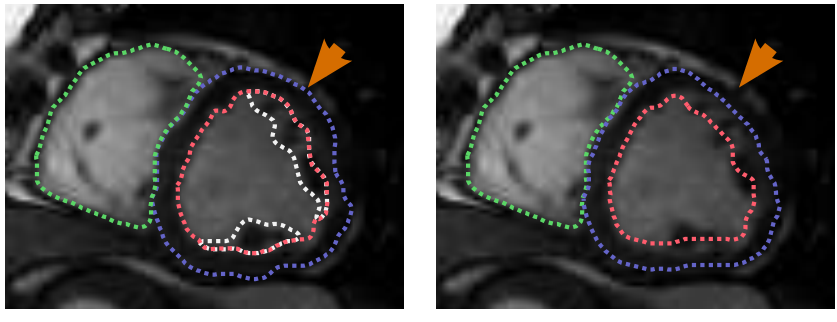
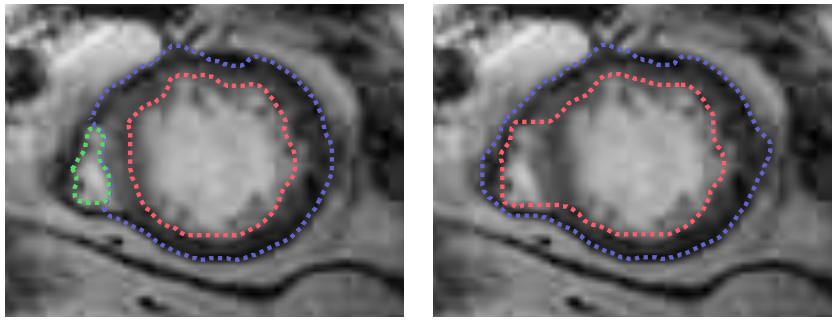
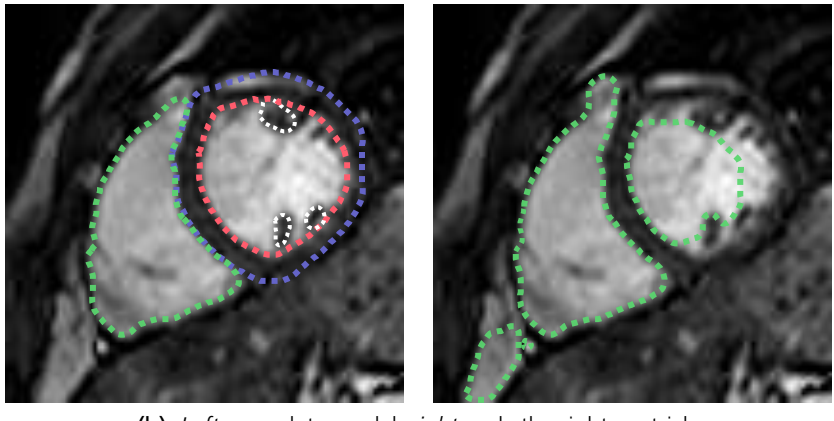


Figure 12.4: Example how modeling the papillary muscles improved both the segmentation of the left ventricle and the myocardium. *Left*, complete model; *right*, without modeling the papillary muscles.



(a) *Left*, complete model; *right*, without modeling the right ventricle.



(b) *Left*, complete model; *right*, only the right ventricle.

Figure 12.5: Examples of how modeling multiple regions improves the segmentation of the ventricular epi- and endocardium. The color scheme is the same as in figure 12.1b.

Method	Memory (MB)		Running time (s)	
	Sunnybrook	Lund	Sunnybrook	Lund
Supergradient	$2727_{\pm 680}$	$2103_{\pm 788}$	$46_{\pm 27}$	$30_{\pm 27}$
RD (improve)	$5038_{\pm 985}$	$3913_{\pm 1407}$	$135_{\pm 165}$	$80_{\pm 113}$
RD (probe)	$5041_{\pm 1014}$	$3949_{\pm 1402}$	$6109_{\pm 12451}$	$1934_{\pm 7984}$
	Relative duality gap		**Dice (average)**	
	Sunnybrook	Lund	Sunnybrook	Lund
	$0.00054_{\pm 0.0013}$	$0.00054_{\pm 0.0021}$	$0.888_{\pm 0.0484}$	$0.892_{\pm 0.0815}$
	$0.00016_{\pm 0.00034}$	$0.00049_{\pm 0.0021}$	$0.888_{\pm 0.0485}$	$0.892_{\pm 0.0816}$
	$0.0011_{\pm 0.0030}$	$0.00056_{\pm 0.0021}$	$0.888_{\pm 0.0484}$	$0.892_{\pm 0.0825}$

Table 12.1: Memory consumption of the optimization in megabytes and the relative duality gap. The data resolution was for Sunnybrook and Lund $146 \times 146 \times 10 \times 2$ and $126 \times 126 \times 10 \times 2$ voxels, respectively, on average. The average dice metric is the average over parts where ground truth was available.

	End systole			End diastole		
	LV endo.	LV epi.	RV	LV endo.	LV epi.	RV
Multi	$0.87_{\pm 0.05}$	$0.88_{\pm 0.05}$	$0.80_{\pm 0.11}$	$0.96_{\pm 0.02}$	$0.93_{\pm 0.03}$	$0.91_{\pm 0.07}$
Single	$0.47_{\pm 0.25}$	$0.86_{\pm 0.04}$	$0.42_{\pm 0.14}$	$0.62_{\pm 0.12}$	$0.90_{\pm 0.03}$	$0.57_{\pm 0.14}$

Table 12.2: Results in the dice metric for the Lund data set reported as mean $\pm$ one standard deviation. "Multi" is the full multi-region model and "single" is each region segmented separately (using the same data term and smoothness). Note that the multi-region model has a huge influence on the segmentation results.

Method	Dice		LV Mass (g)	LV ejection fraction (%)
	LV endo.	LV epi.		
This ch.	0.86±[0.05]	0.92±[0.02]	27.1±[28.3]	12.5±[8.7]
A	0.86±0.04	0.93±0.01	23±?	14±?
B	0.89±0.03	0.94±0.02	21.6±14.6	8.08±5.06
C	0.89±0.03	0.93±0.01	28.7±18.7	7.02±4.78
D	?	0.93±?	†	?
E	0.81±?	0.91±?	?	?
F	0.89±0.04	0.92±0.02	†	†
G	0.89±0.04	0.94±0.01	?	?
H	0.88±0.04	0.93±0.02	31.8±17.7	8.35±5.78

A Marak et al. (2009)
B Lu et al. (2009)
C Wijnhout et al. (2009)
D Casta et al. (2009)
E O'Brien et al. (2009)
F Constantinides et al. (2009)
G Huang et al. (2009)
H Jolly (2009)

Table 12.3: Results for *Sunnybrook*. "?" means not reported in the corresponding paper. "†" means that the result is not directly comparable. Mass and ejection fraction is reported as difference between manual and automatic value.

12.3.1 Experiments

The segmentation is only performed on the slices of the heart which are fully below the atrioventricular plane. The quality of the segmentation is measured by the dice metric, given by $2\,|A \cap B| \,/\, \big(|A| + |B|\big)$, where A and B are the ground truth and the computed segmentations, respectively.

The algorithm was evaluated on two data sets: Lund and Sunnybrook. Each data set was trained and evaluated separately. Lund consists of cine short-axis steady state free precession MR images of 62 healthy normals captured on a Philips Interera CV 1.5 T with a five channel cardiac synergy coil. Each heart has the left and right ventricular endocardium and the left ventricular epicardium manually delineated by an expert. The data set is split into two equally sized parts, one used for training and one used for evaluation. Results are given in table 12.2a and an example segmentation in figure 12.3. The experiments evaluated three clinical parameters: the left ventricular mass has an error of 15.6 ± 11.5 g, the left and right ventricular ejection fraction errors are $5.6 \pm 2.9\%$ and $7.1 \pm 5.2\%$, respectively.

A very important question is whether using multiple regions actually improves the segmentation performance. This is easy to answer by running a simplified version wherein the segmentation for each region is run separately; see table 12.2b. Without the complete multi-region model, the localization of the ventricles becomes very difficult and the blood pools are often overestimated. Figures 12.4 and 12.5 give two examples where the multi-region model improves the segmentation significantly.

Our method found a globally optimal solution for 52% of the hearts, and for the other hearts we can from the very small relative duality gap be certain that the method found a solution close to the global optimum. For 4 out of a total of 46 hearts the probing took more than 12 hours and we terminated the calculations after that time. This highlights the problem with probing - there is no real guarantee that the computations will be done within a reasonable time; on some problem instances we had probing running for several weeks without returning a complete solution.

The Sunnybrook set consists of 30 patients with different heart diseases and is split up into two equally sized parts, one for training and one for evaluation. The data set was used in the 2009 MICCAI cardiac MR left ventricle segmentation challenge. Sunnybrook lacks ground truth for the right ventricles, so Johannes Ulén drew this for all images. Skillful as Johannes is,

he is not a physician; therefore, this ground truth was only used for training and not for evaluation. The results given by the evaluation code used in the challenge are given in table 12.3 along with results from competing methods. The evaluation in the challenge calculates the dice metric per slice and averages over all slices.

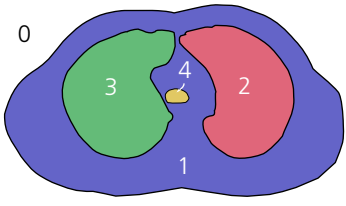
Another important question is whether the supergradient method performs worse or better than roof duality (RD). Using subgradients was consistently faster than RD and at the same time giving a very small relative duality gap. The small gaps gives us certificates that the solutions are very close to (and in many cases exactly) the global minimum; see table 12.1. This table reports results using the two variants of roof duality "probe" and "improve" by Rother et al. (2007*a*).

Extending the cardiac model to also include papillary muscles in the right ventricle is of course possible; one more variable per voxel is then required. Initial experiments gave worse results for both the right ventricle and myocardium segmentation with the added region. The new region had a tendency to overflow into the septum (the wall separating the ventricles) since this would give region 3 a rounder shape giving a lower regularization cost, and therefore it has not been incorporated in the model.

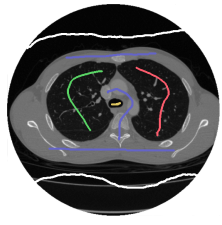
The Lund data set was manually delineated by experts using both short- and long-axis images. For a number of hearts the most basal slice for the short-axis images containing the left ventricular cavity also cut through to the atrium. For these slices it was hard or even impossible to even manually delineate the left ventricle solely based on information from the short-axis images. When the experts produced the ground truth, they used long-axis images to be able to correctly segment the short-axis images. It would be desirable for the segmentation method to incorporate information from long-axis images as well so these few slices could be handled correctly as well.

12.4 Lung Segmentation

The second application is the segmentation of lungs in a full-body X-ray CT scan. The model is shown in figure 12.6 and uses four regions: the body (region 1), the two lungs (regions 2 and 3) and the heart together with the throat (region 4). Regions 2, 3 and 4 are all forced to be contained inside region 1 by adding the terms $W_p^{1,2}(0,1) = \infty$, $W_p^{1,3}(0,1) = \infty$ and



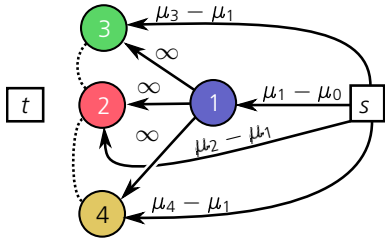
(a) Four-region model



(b) CT scan with seeds provided by the user

r_p	$\boldsymbol{x}_p$
0	(0, 0, 0, 0)
1	(1, 0, 0, 0)
2	(1, 1, 0, 0)
3	(1, 0, 1, 0)
4	(1, 0, 0, 1)

(c) Representation



(d) Graph

Figure 12.6: (a) A diagram showing the model used for lung segmentation. Region 0 is the background, region 1 the body, regions 2 and 3 are the right and left lungs, respectively, and region 4 is the throat. **(b)** The seeds in one slice used for the segmentation. In a clinical setting, these are provided by a physician. **(c)** The boolean representation of the four regions. **(d)** Graph construction for one voxel, showing the geometrical relationships.

$W_p^{1,4}(0,1) = \infty$ for all $p \in \mathcal{P}$. The major difference to the cardiac model is that more than one separation need to be enforced during optimization, that is,

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & E'(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{x}^2 + \boldsymbol{x}^3 + \boldsymbol{x}^4 \leq 1. \end{aligned} \tag{12.17}$$

Alternatively, the three-variable constraint could equivalently be replaced with three constraints of the same type as in the previous section:

$$\begin{aligned} \boldsymbol{x}^2 + \boldsymbol{x}^3 &\leq 1 \\ \boldsymbol{x}^2 + \boldsymbol{x}^4 &\leq 1 \\ \boldsymbol{x}^3 + \boldsymbol{x}^4 &\leq 1. \end{aligned} \tag{12.18}$$

These two sets of constraints perform almost identically in the experiments, but, obviously, the second set requires three times as many dual variables.

Figure 12.6d shows the unary terms construction—that is

$$\begin{aligned} D_p^1(1) &= \mu_1(p) - \mu_0(p), \qquad & D_p^2(1) &= \mu_2(p) - \mu_1(p), \\ D_p^3(1) &= \mu_3(p) - \mu_1(p), \qquad & D_p^4(1) &= \mu_4(p) - \mu_1(p) \end{aligned} \tag{12.19}$$

and $D_p^r(0) = 0$ for all $r \in \mathcal{R}$ and $p \in \mathcal{P}$.

The user gives ground truth seeds only in one slice of the data as shown in figure 12.6b. The background is removed by thresholding on an intensity level between the seeds given from the background and the body. The seeds are then used to build intensity histograms for the five regions which are used to estimate the intensity distribution. Figure 12.8 shows an example of the final unary terms using the estimated intensity distributions. The unary terms do not use any spatial component estimated from training data, but a gradient is present (figures 12.8d and e) to be able to distinguish between the left and the right lung.

12.4.1 Experiments

The experimental test of the algorithm used a full-body X-ray CT data set with seed as shown in figure 12.6. A sample result from a few slices can be seen in figure 12.7. The running time for roof duality is 39 seconds and

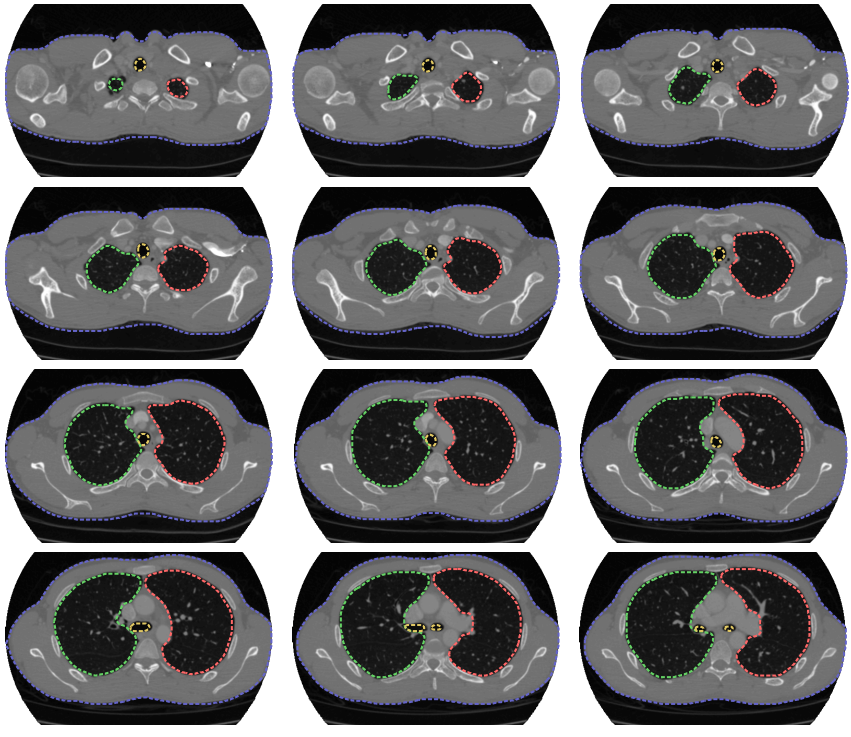
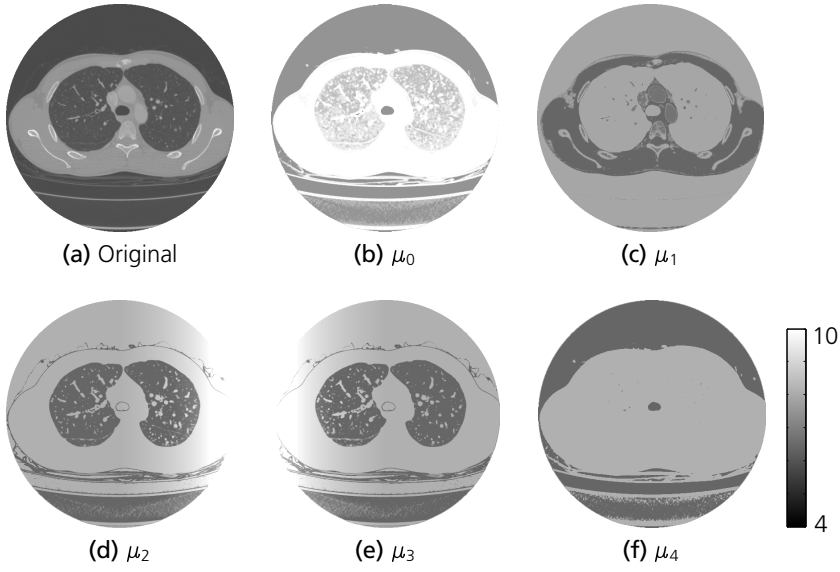


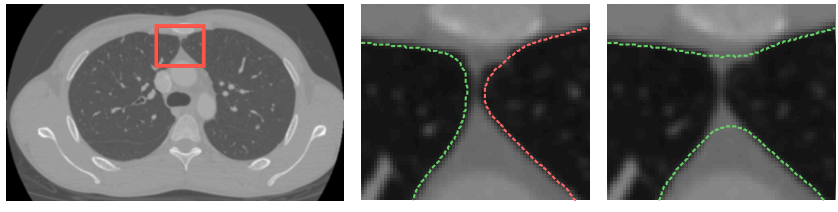
Figure 12.7: Sample result from the segmentation. The color coding is the same as in figure 12.6 (page 189).



(a) Original **(b)** μ_0 **(c)** μ_1

(d) μ_2 **(e)** μ_3 **(f)** μ_4

Figure 12.8: An example slice from the data set with the five calculated unary terms for this slice.



(a) Region of interest marked with a rectangle. **(b)** Forced separation. **(c)** No separation.

Figure 12.9: An advantage of the multi-region model. **(b)** The model have forced regions two and three apart. **(c)** The exclusion constraint between region 2 and 3 was removed, resulting in an over-smoothed boundary.

for supergradient ascent 29 seconds. Both methods give exactly the same solution.

The current implementation used two different general-purpose max-flow implementations (Boykov and Kolmogorov, 2004; Goldberg et al., 2011). The performance of the two algorithms was quite similar. One thing not taken into account is the fact that the structure of the graph is highly repetitive. For instance, all geometric interaction terms are equal and they need not be explicitly stored in the graph. A specialized-purpose solver for this problem could lead to a large reduction of memory requirements.

12.5 Conclusions

The following three points summarize the main observations from the experiments:

1. The experiments have shown that a multi-region model achieves significantly better results, all else being equal, than segmenting the regions one at a time (figures 12.5, 12.4 and 12.9). Enforcing geometric constraints, and more generally, incorporating prior information into the model, result in qualitative improvements. This is not always captured well by quantitative measures such as the dice metric.

2. The optimization method based on supergradients significantly outperforms roof duality, both in terms of speed, quality of solution and memory consumption.

3. Application of the multi-region framework for cardiac segmentation achieves results on par with dedicated LV methods on a publicly available data set. Although there are fine-tunings one can make to the model to improve performance, these results are encouraging.

The introduction to this thesis mentioned the trade-off between tractability and fidelity. Increasing the fidelity of models in medical image segmentation without sacrificing tractability has been the purpose of this chapter. The human heart is composed of several interacting geometric parts—this fact really should be reflected in the model.

Popular methods for segmenting lungs use multiple steps, where the first finds an initial segmentation and the subsequent second step separates

the left and right lungs. For example, Hu et al. (2001) and Armato III and Sensakovic (2004) separate the lungs in each slice individually, whereas Ukil and Reinhardt (2009) separate the whole volume at the same time. In contrast, the optimization problems in this chapter can be solved optimally without having to resort to multiple phases. This is what increased tractability without sacrificing fidelity means.

Chapter 13

Shift-Map Image Registration

This chapter will describe how multi-label optimization can be used for image registration. Shift-map image processing was recently introduced by Pritch et al. (2009) who applied their framework to image inpainting, content aware resizing, texture synthesis and image rearrangement. The purpose of this chapter is to extend the range of applications to image registration.

13.1 Problem Formulation

Registration can be performed using a parametric model, e.g. an affine or a projective transformation estimated from point correspondences between two images. This chapter considers a non-parametric model. We have a base image $B(i,j)$ and an input image $I(i,j)$. These two images need not have the same size. The goal is to register the pixels of the input image onto the base image using a shift-map $\boldsymbol{T}(i,j) = \Big(t_i(i,j) \;\; , \;\; t_j(i,j)\Big)$. The pixel $I(i,j)$ is registered onto $B\Big(i + t_i(i,j) \;\; , \;\; j + t_j(i,j)\Big)$. Figure 13.2 shows the input and base images and the resulting image obtained by moving all pixels in the input image as specified by the computed shift-map.

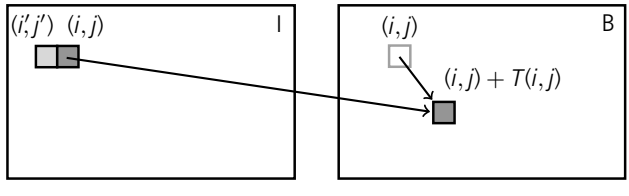


Figure 13.1: Shift-map between two images

(a) Input image I

(b) Base image B

(c) Shift-map result

(d) Final locations of pixels

(e) SIFT Flow result (Liu et al., 2008)

(f) SIFT Flow final locations

Figure 13.2: Registration of two images using a shift-map. Each pixel in the input image is placed on the base image as described by the shift-map.

Each possible shift-map $\boldsymbol{T}(i,j) \in \{-m,\dots,m\} \times \{-n,\dots,n\}$ is assigned a cost, based on a priori assumptions on what a good shift-map typically looks like and how well the two images match each other. The goal is then to find the optimal shift-map, that is, the shift-map with the lowest cost:

$$E(\boldsymbol{T}) = \rho \sum_{\substack{1\le i\le m\\ 1\le j\le n}} E_d^{ij}(\boldsymbol{T}(i,j)) + \sum_{\substack{1\le i\le m\\ 1\le j\le n}} \sum_{(i',j')} E_s^{ij}(\boldsymbol{T}(i,j),\boldsymbol{T}(i',j')), \tag{13.1}$$

where the last summation refers to summations over all (i',j') in a neighborhood $\mathcal{N}(i,j)$ of (i,j). Figure 13.1 shows one such neighbor. This chapter uses 4-connectivity exclusively. E_d^{ij} and E_s^{ij} are the data terms and smoothness terms, respectively. They will be described in separate sections below.

Comparison of pixels. A related problem to image registration is dense depth estimation from two images of the same object with known camera positions. This problem has been studied extensively; see for example the work by Kolmogorov and Zabih (2006). Recently a new descriptor, DAISY, was proposed by Tola et al. (2009), tailored to dense stereo estimation where the position of the two cameras differ by a large amount. This descriptor is shown to outperform other approaches (e.g. SIFT, SURF and pixel differences) in extensive experiments. Therefore, it seems relevant to try and apply this descriptor to the related problem of estimating a dense image registration.

Not unlike SIFT (Lowe, 2004), a DAISY descriptor samples the image derivative in different directions. Eight different directions and three different scales are used. By sampling these fields at different points around the feature location, a descriptor of dimensionality 200 is obtained. Since the same fields are used for all image locations, a dense field of descriptors can be computed in a couple of seconds. The main goal of the DAISY descriptor was efficient dense computation. The work by Winder et al. (2009) is helpful when facing the task of choosing the correct parameters.

Data terms. The data terms E_d^{ij} were previously used by Pritch et al. to enforce hard constraints on the shift-map. When inpainting an image, the

data term makes sure no pixels in the "hole" are used in the output image by assigning such shifts a cost of ∞.

In this chapter, where image registration is considered, more complex data terms are needed to incorporate the fact that we want to find a mapping between two images such that similar pixels are mapped to similar pixels. The data terms dictate that similar parts of the images should end up on top of each other. To measure similarity, dense DAISY is used.

It might only be possible to register parts of the input image, so shifting pixels outside the base image is permitted, at a constant cost P per pixel. The data terms are then given by

$$E_d^{ij}(\boldsymbol{T}) = \begin{cases} \left\| \hat{I}(i,j) - \hat{B}\Big((i,j) + \boldsymbol{T}(i,j)\Big) \right\|_2 \\ P \text{ when } (i,j) + \boldsymbol{T}(i,j) \text{ is outside B,} \end{cases} \tag{13.2}$$

where $\hat{I}(i,j)$ is the DAISY descriptor describing the image I at pixel location (i,j). If the shift takes pixel (i,j) outside the bounds of the base image, a constant cost is issued. Otherwise, dissimilarity of the pixels determines the cost of the assignment. Figure 13.6e shows a heat map of the distance from the circled feature in the first row to all locations in the image in row 2.

Smoothness terms. The smoothness terms are used to enforce global consistency to the shift-map, while allowing discontinuities at a limited number of places. Pritch, Kav-Venaki and Peleg's (2009) smoothness terms compared the color and gradient pixel-wise. Where a discontinuity in the shift-map occurs, the penalty is computed as the difference in color and gradients. The smoothness function takes the form of the Euclidean distance between the endpoints of the two shifts:

$$E_s^{ij}(\boldsymbol{T}(i,j), \boldsymbol{T}(i',j')) = \|(i',j') + \boldsymbol{T}(i',j') - (i,j) - \boldsymbol{T}(i,j)\|_2. \tag{13.3}$$

Here, (i,j) and (i',j') are neighboring pixels; see (13.1). Using the shift difference $\|\boldsymbol{T}(i',j') - \boldsymbol{T}(i,j)\|_2$ will penalize smoothly varying shift-maps too much, and hence it is important to compare the end points (as in (13.3)).

Color information. The DAISY descriptor does not use color information, yet intuitively it makes little sense to match pixels of very different colors.

Because of this, Linus Svärm and I also made experiments where the color information of the images is incorporated into the above data terms. The color model used assigned a cost of P to pixels with large difference in hue, given that the intensity and saturation allowed a reliable value of the hue. This model improved the result of the registration in figure 13.6. The experiment shown in figure 13.2 did not use color information, though.

13.2 Experiments

The objective function (13.1) can be minimized with α-expansion (section 3.6 on page 29). To make the optimization problem tractable, a Gaussian pyramid was used. The images in figure 13.6 used an initial size of 128×23. The size was then doubled 3 times until the final resolution of 1024×179 was reached. Each doubling of the image size is followed by a linear interpolation of the shift-map. This shift-map was used as a starting guess for the optimization at the larger level. At each level after the first, only 9 possible shifts then need to be considered: $\{-1, 0, 1\}$ in each direction.

To verify the implementation, I inpainted an example image used by Pritch et al. (2009); see figure 13.4. I tried to follow their implementation as closely as possible and got different, but qualitatively similar results. The pixels outside the area to be removed were not allowed to move at all, which is in contrast to the work by Pritch et al. (2009), in which all pixels except the border of the image were allowed to be shifted. Pritch et al. (2009) truncated every non-submodular term when calculating the moves. An arguably better approach is to use roof duality. I tried both and concluded that truncation seems to work well for these problems; figure 13.4 shows examples of both approaches.

Figures 13.2 and 13.3 show shift-map registration results. The bear image in figure 13.2 shows the same object from two different views and is from (Kushal and Ponce, 2006). The building images in figure 13.3 register correctly, except for the light pole, which is very thin and does not have a large enough data term. An experiment evaluating shift-maps quantitatively consisted of two images related by a known deformation. The results are displayed in figure 13.5.

During large-scale reconstruction of a city using images taken with a cylindrical camera (Hitta.se street view), my coauthor Linus Svärm has encountered many difficult image pairs where SIFT is unable to provide

(a) Input image I (b) Base image B (c) Final location of pixels

Figure 13.3: Registration of two images of a building.

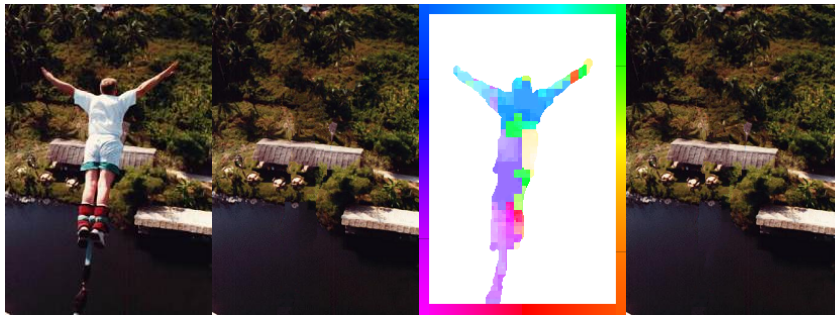
useful correspondences. The top two rows in figure 13.6 show one of the hardest. The computed SIFT features for the two images (794 and 1019 feature points, respectively) only gave 3 correct matches. The main reason for this was the image geometry and large, repetitive patterns. Computing a shift-map resulted in a dense, mostly correct map between the images. This was then used as an aid to compute SIFT correspondences, where features in one image were only allowed to match within a neighborhood of the location obtained by following the shift-map. This resulted in 28 matches, of which 12 were correct. The run time for this image was about 2 minutes.

I have also compared shift-map registration to the recent SIFT flow algorithm (Liu et al., 2008). Both algorithms worked well for simple distortions of small magnitudes, which can be seen in figure 13.5. However, for the other, more challenging experiments, I was not able to get any satisfactory results using SIFT flow. An example is shown in figure 13.2.

13.3 Conclusion

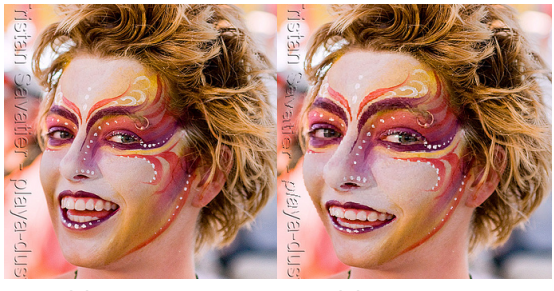
In conclusion, computing the smoothness term with color and gradient differences as done by Pritch et al. (2009) did not give satisfactory results when extended to image registration. Using the dense DAISY descriptor resulted in a great improvement.

Shift-maps perform better than SIFT flow for some problems because it is not (initially) restricted to a neighborhood and can find large shifts in every direction. A downside is that the first iteration at the coarsest resolution has to find a good initial map since all subsequent steps are refinements of the maps from the previous steps.

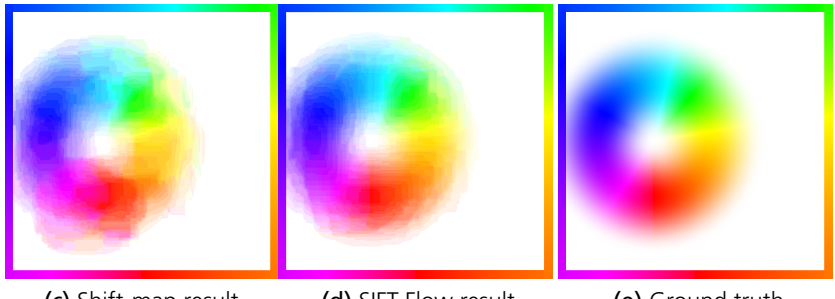


(a) Original (b) Shift-map inpainting (c) With roof duality

Figure 13.4: The inpainting algorithm by Pritch et al. (2009). The run times were 3.1415 seconds (pure coincidence!) for truncation (b) and 17.8 seconds for roof duality (c). Roof duality found a 5% lower objective value.



(a) Base image (b) Input image



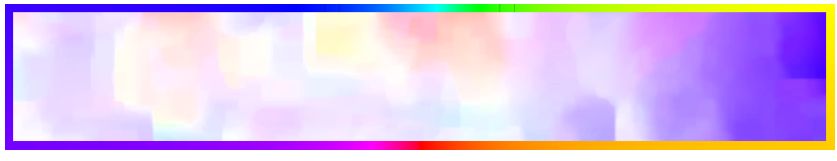
(c) Shift-map result (d) SIFT Flow result (e) Ground truth

Figure 13.5: Recovering a known image distortion. The maximum and mean error for the shift-map estimation was 7.3 and 0.7 pixels, respectively. Photo by Tristan Savatier obtained through Flickr.

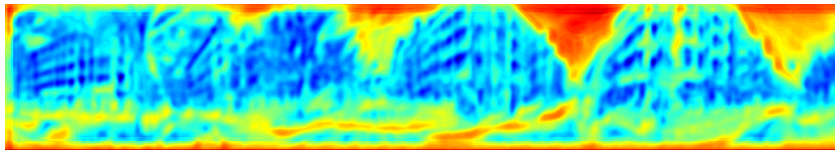
(a) Input image I

(b) Base image B

(c) Final locations of the pixels in *I*



(d) Resulting shift-map



(e) DAISY distance between the circled feature in I to all pixel locations in B

Figure 13.6: Registration of 1024×179 Hitta images. The result is a dense, highly non-linear registration. This shift-map allowed us to obtain useful point-correspondences between the images, which was not possible using SIFT alone.

Chapter 14

HEp-2 Staining Pattern Classification

The major part of my thesis is about developing new optimization methods for low-level vision. In constrast, this chapter will present something from the other end of the spectrum—a demonstration of what can be done when a good segmentation is available. The work presented here was presented at the International Conference on Pattern Recognition (ICPR, 2012) contest "HEp-2 Cells Classification." Apart from describing a classification system, this chapter will explain why all of the participants in the contest severely overestimated their classification performance.

14.1 Indirect Immunofluorescence

Indirect immunofluorescence (IIF) is a method for detecting antinuclear antibodies in patient serum. This is important when diagnosing autoimmune connective tissues disorders (Rigon et al., 2007). Using IIF images, it is possible to locate cells as well as to classify their type in the sample. The manual classification of these cells suffer from the usual problems in medical imaging, such as: (i) The result is dependent on the experience and expertise of the specialist and (ii) it requires a lot of manual work. Reliable automatic systems are in great demand.

This chapter introduces a new classification method for mitotic cells in IIF images, where each mitotic cell is classified into one out of six categories. This classification is one step in the detection of antinuclear autoantibodies (Percannella et al., 2011). The method works in an automatic setting, where it is assumed that the segmentation of the cell boundary in the image is known.

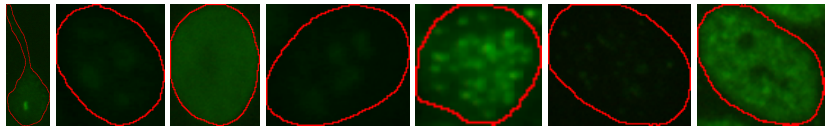


Figure 14.1: Seven random input images with given segmentation boundaries.

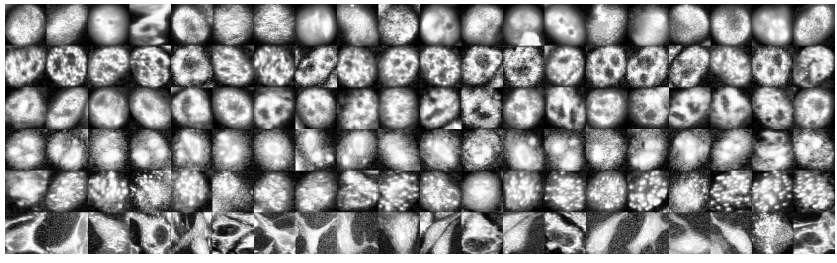


Figure 14.2: The rows show 20 samples from classes 1–6, respectively. The images have been resized to 100 × 100, converted to gray-scale and histogram equalized to enhance visibility.

Data Set. The training data set consists of 721 segmented and classified images. Figure 14.1 shows seven random images exactly as they appear in the data set. Since the test data set of the contest was not available to us, I evaluated the classification system using leave-one-out cross-validation on the training data set.

The task of the challenge was to classify a given segmented input image of a cell as one of six classes. The six classes of the data set are:

1. Homogeneous
2. Coarse-speckled
3. Fine-speckled
4. Nucleolar
5. Centromere
6. Cytoplasmatic.

Figure 14.2 shows 20 examples from each class.

Previous Work. A recent work on the same data set is the paper by Cordelli and Soda (2011). They used the full data set with 1457 cell images, evalu-

ated many different methods and achieved a best accuracy of 79.3% using AdaBoost. Accuracy is simply defined as the percentage of correctly classified images. Foggia et al. (2010) and Percannella et al. (2011) have also performed experiments on this data set, but for a different task: mitotic cell detection.

Papers using other data sets have been written by Perner et al. (2002) with 75% accuracy, Sack et al. (2003) with 83% and Soda and Iannello (2009) with 76% accuracy. These earlier works, however, used a different data set and only four classes, making direct comparison difficult. I am not aware of any other publicly available data set for this task. Prior to the ICPR contest, comparing our results to the state of the art would have been difficult.

14.2 Classification Methods

Convolutional neural nets are state of the art for this kind of "easy" image classification tasks. For example, the work by Ciresan et al. (2011) obtains a recognition rate of $99.73\% \pm 0.02\%$ on the MNIST data set of handwritten digits. On the same data set, a simple neural network trained via backpropagation (Ciresan et al., 2010) can also perform extremely well with an accuracy of 99.65%. Another recent result of neural nets is the German traffic sign recognition benchmark (Stallkamp et al., in press), where the winning entry (Ciresan et al., 2012) used a committee of convolutional neural networks and obtained an accuracy of 99.46%, significantly outperforming humans.

The method has some drawbacks, however, which makes it less useful in our context. First, neural networks have a strong tendency of overfitting. This can be alleviated by creating new training data by rotating, translating and skewing existing data. In this way, the same image is never fed to the network twice during training. In any case, the possibility of overfitting must be carefully monitored during training. Second, the training requires a vast amount of computer resources. For example, (Ciresan et al., 2011) used more than two weeks for training with a system with four graphics processing units.

The contest data set has larger images than e.g. MNIST, which makes training an ensemble of networks computationally infeasible. Instead, the classification system developed in this chapter uses a random forest (Breiman,

2001; Criminisi et al., 2011). A random forest computes averages over several hundreds of small decision trees, each of which is trained on a subset of the features and the training examples. It is not sensitive to overfitting and the total time required for training is 15 seconds.

14.3 Features Used for Classification

As a random forest classifier is relatively insensitive to irrelevant features, adding redundant and poor features is not a problem. Therefore, one should not worry about whether a specific feature is relevant when developing classification software. The relevant ones will be picked automatically. One guideline we used was that for this particular task, unlike e.g. OCR, there should be no reason to include features that are not rotationally symmetric.

The fluorescence light captured is essentially monospectral. Therefore, the first step is to project the RGB color of each pixel onto the principal component of all pixel colors in the training set. Before any further processing, the background of each image is removed using the segmentation mask provided in the data set. Some images suffer from a few bright (saturated) pixels. Taking all intensities in an image and calculating a 10 binned histogram and emptying each bin which contains less than 0.5% of the pixels reduces this problem. If two consecutive bins are empty, the image intensity is truncated at this level. The program then starts to calculate features on the newly formed image.

The first feature is the "positive" or "intermediate" flag given in the data set. The second feature is the aspect ratio of the image. The image is then thresholded at 20 intensities equally spaced from its minimum to its maximum intensity. Each threshold level gives a binary image with the following features:

- Number of objects
- Area
- Area of the convex hull
- Eccentricity
- Euler number
- Perimeter.

These features are also calculated on the segmentation mask. Each binary image is also used in the same way as the mask to create a cut-out of the

original image. On this cut-out the means of the following image properties constitute a new group of features:

- Intensity
- Standard deviation in 3×3 neighborhoods.
- Entropy (9×9 neighborhood) (Gonzalez et al., 2007)
- Range (3×3 neighborhood) (Gonzalez et al., 2007).

The means of these features are also taken on a smaller mask formed by eroding the initial mask by a 5×5 kernel of ones. Another set of features are the average gradient magnitudes of the image after it has been smoothed with a Gaussian kernel with 10 different σ equally spaced in $[0.6, 10.5]$. The final features are based on gray-level co-occurrences (Haralick et al., 1973) with offsets $\{-1, 1\}$. The co-occurrences are calculated on images which have been transformed to contain only $k \in \{2, 5, 8, 11, 14\}$ different intensities. The gray-level co-occurrences are probabilities $p(a, b)$ of intensity pairs (a, b) in the image. Four features are computed using these probabilities:

- Contrast: $\sum_{(a,b)} |a - b| p(a, b)$.
- Correlation: $\sum_{(a,b)} \frac{(a-\mu_a)(b-\mu_b)}{\sigma_a \sigma_b} p(a, b)$, where μ_a is the mean of the first component of all intensity samples (a, b) etc.
- Energy: $\sum_{(a,b)} p(a, b)^2$.
- Homogenity: $\sum_{(a,b)} \frac{p(a,b)}{1-|a-b|}$.

The described features make up a feature vector F_1 with 322 features. The same features are also calculated on the image after it has been smoothed by a gaussian kernel with $\sigma = 1$ and $\sigma = 2.5$ to reduce image noise. This forms two new feature vectors F_2 and F_3. The final feature vector is obtained by concatenation and subtraction as $(F_1, F_2 - F_1, F_3 - F_2)$. The total number of features is then 966. All the parameters (e.g. the amount of smoothing) were chosen by local optimization with cross-validation on a subset of the training data.

14.4 Results

The average time for loading a single-cell image from disk and computing all its features is 1.6 seconds on an Intel Core i5 2500K 3.3 GHz. Although this could probably be improved, I believe it is still within the requirements

	1	2	3	4	5	6
1:	142	0	6	0	1	1
2:	0	108	0	1	0	0
3:	5	0	89	0	0	0
4:	0	0	0	102	0	0
5:	1	0	2	1	204	0
6:	0	0	0	1	0	57

Table 14.1: Confusion matrix. Rows show ground truth and columns our classification.

of a system for clinical use. The time to classify the features is negligible (51µs on average). Combining this with a total training time of 15 seconds gives a quite efficient classification system. I used the default ensemble size of 500 trees, but the out-of-bag error rate (Breiman, 2001) during training suggests that using less (150) trees would have worked just as well.

The overall accuracy is 97.4% based on leave-one-out cross-validation. Figure 14.5 shows all 19 images which the system fails to correctly classify. The same information is presented in table 14.1 as a confusion matrix. The accuracy of our system may be on par with the inter-lab variability (Bizzaro et al., 1998), but additional studies are required to confirm this.

The votes from the individual trees can be used to obtain a (normalized) confidence score of each class. It is then interesting to see whether the misclassified examples have lower confidence than the correctly classified ones; see figure 14.5. Another experiment allowed the system to abstain from classifying examples for which the confidence was low, in a similar manner to (Soda and Iannello, 2009). This feature is useful in a semi-supervised setting, where easy instances are classified automatically and the harder ones are sent to a specialist for further consideration. Figure 14.3 shows that allowing 11.1% of the cells to remain unclassified yields an accuracy of 100%.

14.5 Discussion

Whether our increased accuracy with respect to Cordelli and Soda (2011) is due to better and more discriminative features or the use of random forests is a relevant question. Cordelli and Soda tried several classifiers (k-NN, SVM,

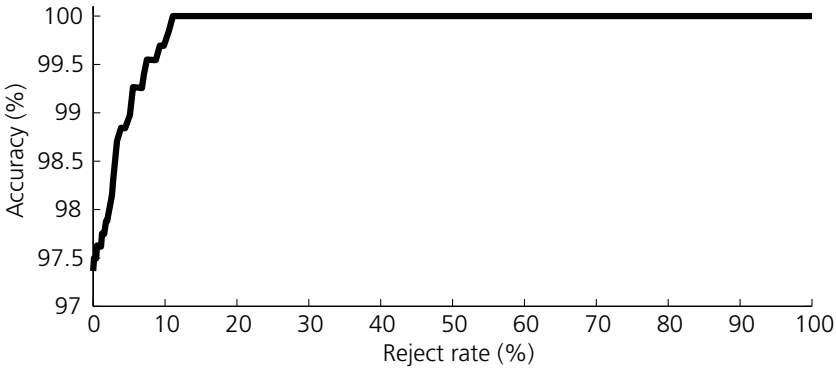


Figure 14.3: Accuracy as a function of the number of rejected examples. A reject rate of 11.1% gives a perfect accuracy.

Adaboost, etc.). Table 14.2 shows a comparison of six different classifiers on the same features:

- Random forest as described in this chapter.
- Linear SVM.
- One-versus-all AdaBoost using thresholding of single features as weak classifiers.
- k-NN, where a vector is classified as the most common class among its k neighbors using the ℓ_1 distance. Each component of the feature vector is divided by its standard deviation in the training data set.

Figure 14.4 shows how the accuracy of a random forest increases as the number of features increases.

Random forests and nearest neighbors achieve very similar performance, but the very time-efficient classification make random forests the preferable choice. This experiment strongly suggests that the main reason for the increased accuracy is the feature design. By thresholding at different intensity levels and computing descriptors for each resulting level set, more shape information of the image is incorporated in a robust manner.

Contest results. The contest results will be published by Percannella, Foggia and Soda. All 28 participants severely overestimated their classification performance. The training set, as stated above, consisted of 721 images of cells. However, the total number of microscopic images used to extract

RF	SVM	AdaBoost	1-NN	3-NN	5-NN
96.9%	88.1%	91.7%	95.0%	96.1%	95.0%

Table 14.2: Accuracy of different classifiers applied to the features in this chapter. This experiment used a training set of 321 images and a test set of 320 images.

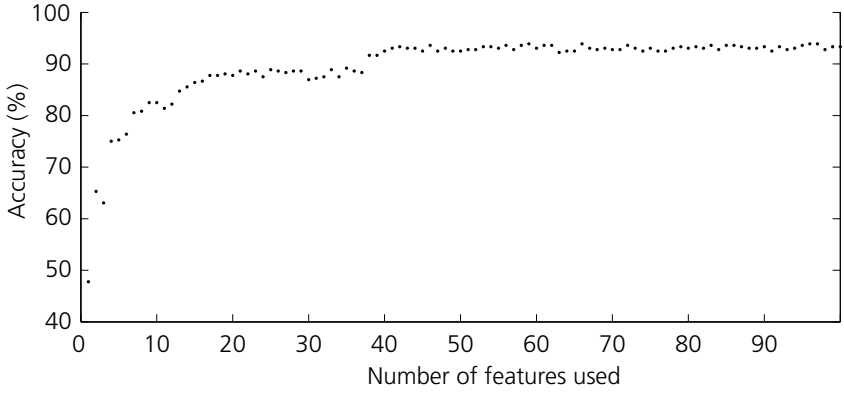


Figure 14.4: The accuracy increases as the number of features increases. This experiment used the same training and test sets as table 14.2.

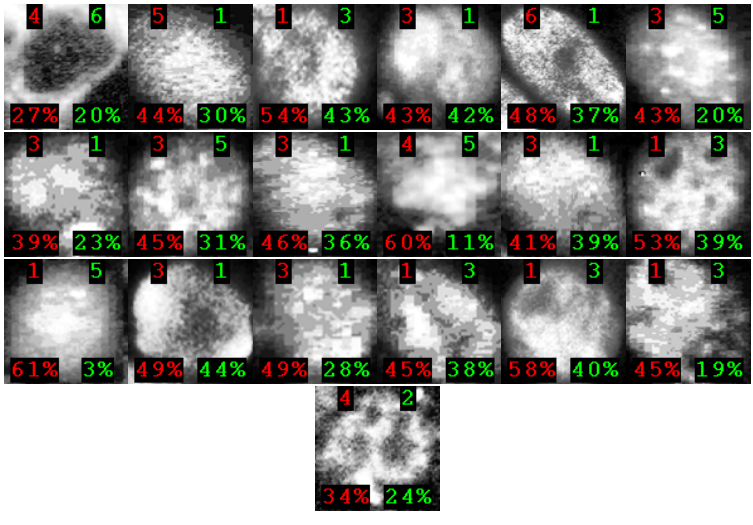


Figure 14.5: All 19 images the classification system assigns the wrong class. The red number is the incorrect classification; the green is the ground truth. Each image also shows the confidence score.

them were only 14. Cells within the same image are presumably very similar and should not be present in both the training and test set during cross-validation. A much better cross-validation study would have left all cells from the same image out and used the rest as training set. However, the cell-image correspondences were not available to the participants of the contest. Virtually all participants used some form of cross-validation to estimate their accuracy, with many of the participants ending up between 94% and 97%. The test set of the contest consisted of cells from another set of 14 different images, different from the training set, and the results on this set were almost uniformly distributed between 19% and 68%. The method in this chapter obtained about 50%[1].

Foggia and Soda reported that an independent immunologist obtained an accuracy of about 70%. However, the immunologist did not even have access to histogram equalization and, as figure 14.1 shows, some images in the data set are very dark. It is therefore possible that a human expert would perform better than 70%.

If nothing else, this shows the importance of correct cross-validation studies for estimating the accuracy of unseen data. The training set of the contests had undocumented structure, which led all participants to overestimate their accuracy.

[1] I do not know the exact value at the time of writing.

Part V

Conclusion

Chapter 15

Recent Work

The work in this thesis has been published in conferences and journals over the course of several years. In some cases other researchers have continued the work. This chapter is a brief account of some of these works. It is not exhaustive and should be seen only as a help to the interested reader rather than a complete account.

15.1 Generalized Roof Duality

Beyond boolean sets. Our first paper on generalized roof duality (Kahl and Strandmark, 2011) was published toward the end of the year. We considered pseudo-boolean functions exclusively, but many theorems have analogues when the domain $B = \{0, 1\}$ is replaced by $\mathbf{L} = \{0, \dots, L-1\}$. Windheuser et al. (2012) have developed this theory. For ordered sets (where min and max are available), the same definition of submodularity can be used; a function $f : \mathbf{L}^n \to \mathbf{R}$ is submodular if

$$f(\boldsymbol{x} \vee \boldsymbol{y}) + f(\boldsymbol{x} \wedge \boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y}), \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbf{L}^n. \tag{15.1}$$

This is the exact same definition as (3.3) found on page 21. Submodular functions can be minimized in polynomial time even when B is replaced by $\mathbf{L}$.

A submodular relaxation can be defined in the same way as for the boolean case (page 39):

$$g(\boldsymbol{x}, \bar{\boldsymbol{x}}) = f(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \mathbf{L}^n, \tag{A}$$

$$g \text{ submodular}, \tag{B}$$

$$g(\boldsymbol{x}, \boldsymbol{y}) = g(\bar{\boldsymbol{y}}, \bar{\boldsymbol{x}}), \quad \forall (\boldsymbol{x}, \boldsymbol{y}) \in \mathbf{L}^{2n} \text{ (symmetry)}. \tag{C}$$

The only question is how the negation operation $\bar{\cdot}$ should be defined. It turns out that a good choice is

$$\bar{x} = L - (x + 1). \tag{15.2}$$

Just as in the quadratic case for pseudo-boolean functions, the optimal relaxation can be constructed explicitly when f is a sum of functions of arity two (with two arguments). Windheuser et al. do this and show that the resulting relaxation is the same as a construction proposed earlier (Kohli et al., 2008) and thus prove its optimality. Higher arities are not considered by Windheuser et al.

This generalization also has persistency. If $(\boldsymbol{x}^*, \boldsymbol{y}^*) \in \operatorname{argmin} g(\boldsymbol{x}, \boldsymbol{y})$ and $\boldsymbol{x} \in \operatorname{argmin}(f)$, then

$$x_i^* \leq x_i \leq \bar{y}_i^* \text{ for all } i. \tag{15.3}$$

Just as in the boolean case, $(\boldsymbol{x}^*, \boldsymbol{y}^*) = (\mathbf{0}, \mathbf{0})$ gives no additional information.

The usefulness of this generalization of generalized roof duality depends on whether $\mathbf{L}$ has a natural ordering and this is true for some applications, such as stereo and image denoising. Somehow choosing the best ordering for the non-natural cases is perhaps possible and would be interesting to investigate. A more general notion of submodularity when $\mathbf{L}$ is only partially ordered also exists (Kolmogorov, 2011).

Symmetric extensions. Chapter 5 outlined a few other methods that can perform better than generalized roof duality. Fredriksson et al. (2012)[1] proposed another based on symmetrization (Boros et al., 2008). The symmetric extension of f is the function

$$\phi(\boldsymbol{x}, x_0) = x_0 f(\boldsymbol{x}) + \bar{x}_0 f(\bar{\boldsymbol{x}}). \tag{15.4}$$

The function ϕ is symmetric in the sense that $\phi(\boldsymbol{x}) = \phi(\bar{\boldsymbol{x}})$. We obtain the original function f if we set $x_0 = 1$. The key insight is that we can fix other variables of ϕ and still obtain persistency for f when using roof duality on ϕ:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \min_{\boldsymbol{x}, x_0} \phi(\boldsymbol{x}, x_0) = \min_{\substack{\boldsymbol{x}, x_0 \\ x_k = 1}} \phi(\boldsymbol{x}, x_0). \tag{15.5}$$

This procedure sometimes gives additional assigned variables for f.

[1] I was a co-author, but did not discuss this work in previous chapters.

15.2 Parallel Minimum Cut

Message passing. Others have continued the work in chapter 7 on parallel minimum cuts after we published those results in 2010. Schwing et al. (2011) used splitting and duality for distributed message passing, which works for general discrete problems. They report good results for medium-scale problems split into 3×3 subproblems.

Push-relabel. Shekhovtsov and Hlavac (2011*a*,*b*) have proposed a combination of the work in this thesis and the work by Delong and Boykov (2008). Just as in chapter 7, their method splits the graph into several pieces which are solved independently with augmenting paths. However, the communication between subproblems consists of pushes and relabel operations instead of duality. In this way, they are able to obtain an upper bound on the number of iterations until convergence, which is very nice. The downside is that a few heuristic techniques are needed to make the algorithm fast in practice—this is often the case for methods based on push-relabel. I think this approach shows great promise.

Bundle methods. This thesis has focused on supergradients to maximize the dual function d (see page 13). Kappes et al. (2012) use a different approach based on bundle methods. They obtained similar results for minimum cuts (as chapter 7 showed, they converge quickly), but for other problems bundle methods could significantly outperform methods based on supergradients.

Cache-efficiency. As we have seen throughout the thesis, graphs in computer vision are often very regular—all node neighborhoods typically look the same. This fact can be exploited to significantly reduce the memory cost of storing the graph. Jamriska et al. (2012) did this and also demonstrated a significant speed-up due to taking better advantage of the microprocessor cache.

Graph reductions. Scheuermann and Rosenhahn (2011) have proposed yet another way of reducing the memory consumption of minimum cuts. With a pre-processing step, they can find nodes which provably cannot be in different parts of any minimum cut. These two nodes are then joined

into one. The result is an increased processing speed and a reduced memory consumption by a factor of two.

15.3 Parametric Models

Convex relaxations. We first presented the work in chapter 9 in 2009 and, as Brown et al. (2011) notes, "there is surprisingly little work done in the direction of global methods in which the values are unknown." The values they refer to are the mean values of the background and foreground. Brown et al. proceed to introduce another method for the two-phase problem studied in chapter 9. They discretize the unit interval of possible values for μ_1 and μ_0 into N values. Their convex relaxation is often able to solve the problem optimally without having to resort to branch and bound. However, their experiments only use N equal to 4 and 5, with running times until convergence ranging up to almost an hour. Trying all $\binom{N}{2}$ combinations of the parameter values seems more efficient. The method still holds potential, as it is more general (the authors perform three-phase experiments) and future work might be able to increase its efficiency.

Bae et al. (2011) introduced a method which seems to allow much finer discretizations. They also stated that "Little work has been devoted to global optimization over the regions and parameters simultaneously in the image segmentation models."

Branch and bound. Lempitsky et al. have a more recent work (2012), which discusses segmentation with branch and bound extensively and includes the two-phase Mumford-Shah functional as a special case. For many problems, only about 1/100th of the tree needs to be traversed.

15.4 Curvature

Shekhovtsov et al. (2012) have recently taken a different approach to curvature regularization based on patches. Each $m \times m$ patch in the resulting segmentation has a cost according to

$$E(\boldsymbol{x}_P) = \min_{y \in \{1,\dots,k\}} \Big(\langle \boldsymbol{w}_y, \boldsymbol{x}_P \rangle + c_y \Big), \tag{15.6}$$

where $\boldsymbol{x}_P$ is a vector with the variables associated with an $m \times m$ patch. Each patch is penalized by the affine function giving the smallest value out of k possibilities. These linear functions can be chosen such that they together approximate the curvature of the segmentation boundary of the patch. Shekhovtsov et al. use TRW-S to minimize the resulting objective function. This approach can also be used for other regularization priors. However, it is unclear how accurate the approximation becomes and the optimization problem seems even harder than our original formulation.

Bibliography

Adams, W. P., J. Bowers Lassiter and H. D. Sherali. 1998. "Persistency in 0-1 Polynomial Programming." *Mathematics of Operations Research* 23(2):359–389. Cited on page 69.

Adams, W. P. and P. M. Dearing. 1994. "On the equivalence between roof duality and Lagrangian duality for unconstrained 0-1 quadratic programming problems." *Discrete Applied Mathematics* 48(1):1–20. Cited on pages 25, 68 and 69.

Agarwal, S. and K. Mierle. 2012. *Ceres Solver: Tutorial & Reference*. Google Inc. http://code.google.com/p/ceres-solver. Cited on pages 78 and 80.

Agarwal, S., N. Snavely, I. Simon, S. M. Seitz and R. Szeliski. 2009. Building Rome in a Day. In *Int. Conf. Computer Vision*. Cited on page 2.

Alahari, K., P. Kohli and P. H. S. Torr. 2010. "Dynamic Hybrid Algorithms for MAP Inference in Discrete MRFs." *IEEE Trans. Pattern Analysis and Machine Intelligence* 32:1846–1857. Cited on page 106.

Armato III, S. G. and W. F. Sensakovic. 2004. "Automated lung segmentation for thoracic CT: Impact on computer-aided diagnosis." *Academic Radiology* 11(9):1011–1021. Cited on page 194.

Bae, E., J. Yuan and X. C. Tai. 2011. Simultaneos convex optimization of regions and region parameters in image segmentation models. In *Dagstuhl Seminar Proceedings*. Springer. Cited on pages 133 and 218.

Bertsekas, D. P. 1999. *Nonlinear programming*. Athena Scientific. Cited on pages 11, 13, 14, 15 and 89.

Bhusnurmath, A. and C. J. Taylor. 2008. "Graph Cuts via ℓ_1 Norm Minimization." *IEEE Trans. Pattern Analysis and Machine Intelligence* 30(10):1866–1871. Cited on pages 85 and 86.

Billionet, A. and M. Minoux. 1985. "Maximizing a supermodular pseudo-boolean function: a polynomial algorithm for cubic functions." *Discrete Appl. Math.* 12:1–11. Cited on pages 38 and 50.

Billionnet, A. and A. Sutter. 1992. "Persistency in Quadratic 0-1 Optimization." *Math. Programming* 54(1-3):115–119. Cited on page 20.

Billionnet, A. and M. Minoux. 1985. "Maximizing a supermodular pseudoboolean function: A polynomial algorithm for supermodular cubic functions." *Discrete Applied Mathematics* 12(1):1 – 11. Cited on page 24.

Bizzaro, N., R. Tozzoli, E. Tonutti, A. Piazza, F. Manoni, A. Ghirardello, D. Bassetti, D. Villalta, M. Pradella and P. Rizzotti. 1998. "Variability between methods to determine ANA, anti-dsDNA and anti-ENA autoantibodies: a collaborative study with the biomedical industry." *Journal of Immunological Methods* 219(1–2):99 – 107. Cited on page 208.

Blake, A., P. Kohli and C. Rother. 2011. *Markov Random Fields for Vision and Image Processing*. MIT Press. Cited on page 26.

Boros, E. and P. L. Hammer. 2002. "Pseudo-boolean optimization." *Discrete Applied Mathematics* 123:155–225. Cited on pages 19, 20, 26, 27 and 44.

Boros, E., P. L. Hammer and G. Tavares. 2006. Preprocessing of unconstrained quadratic binary optimization. Technical report RUTCOR RRR 10-2006. Cited on page 26.

Boros, E., P. L. Hammer, R. Sun and G. Tavares. 2008. "A max-flow approach to improved lower bounds for quadratic unconstrained binary optimization (QUBO)." *Discrete Optimization* 5(2):501–529. Cited on pages 20, 37 and 216.

Boros, E., P. L. Hammer and X. Sun. 1991. Network flows and minimization of quadratic pseudo-boolean functions. Technical report RUTCOR RRR 17-1991. Cited on page 26.

Boyd, S. and L. Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press. Cited on pages 11, 13 and 131.

Boykov, Y. and M.-P. Jolly. 2000. Interactive organ segmentation using graph cuts. In *Conf. Medical Image Computing and Computer-Assisted Intervention*. Cited on page 3.

Boykov, Y., O. Veksler and R. Zabih. 1998. Markov Random Fields with Efficient Approximations. In *Conf. Computer Vision and Pattern Recognition*. Cited on page 83.

Boykov, Y., O. Veksler and R. Zabih. 2001. "Fast approximate energy minimization via graph cuts." *IEEE Trans. Pattern Analysis and Machine Intelligence* 23(11):1222–1239. Cited on pages 29, 30, 31 and 95.

Boykov, Y. and V. Kolmogorov. 2003. Computing Geodesics and Minimal Surfaces via Graph Cuts. In *Int. Conf. Computer Vision*. Cited on pages 93 and 179.

Boykov, Y. and V. Kolmogorov. 2004. "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision." *IEEE Trans. Pattern Analysis and Machine Intelligence* 26(9):1124–1137. Cited on pages 23, 83, 84, 85, 98, 99, 107 and 193.

Breiman, L. 2001. "Random forests." *Machine learning* 45(1):5–32. Cited on pages 205 and 208.

Bresson, X., S. Esedoḡlu, P. Vandergheynst, J. Thiran and S. Osher. 2007. "Fast Global Minimization of the Active Contour/Snake Model." *Journal of Mathematical Imaging and Vision* 28(2):151–167. Cited on page 117.

Brown, E. S., T. F. Chan and X. Bresson. 2011. "Completely Convex Formulation of the Chan-Vese Image Segmentation Model." *Int. Journal Computer Vision* 98(1):103–121. Cited on page 218.

Bruckstein, A. M., A. N. Netravali and T. J. Richardson. 2001. "Epi-convergence of discrete elastica." *Applicable Analysis, Bob Caroll Special Issue* 79:137–171. Cited on pages 138 and 151.

Burkardt, J. 2010. "Voronoi Diagram of Points on the Unit Sphere." http:// people.sc.fsu.edu / ~jburkardt / m_src / sphere_voronoi / sphere_voronoi.html. Cited on page 179.

Carr, P. and R. Hartley. 2009. Minimizing energy functions on 4-connected lattices using elimination. In *Int. Conf. Computer Vision*. Cited on pages 73, 83 and 106.

Casta, C., P. Clarysse, J. Schaerer and J. Pousin. 2009. "Evaluation of the Dynamic Deformable Elastic Template model for the segmentation of the heart in MRI sequences.". Cited on page 186.

Chambolle, A. 2004. "An Algorithm for Total Variation Minimization and Applications." *Journal of Mathematical Imaging and Vision* 20:89–97. Cited on page 122.

Chambolle, A. 2005. Total Variation Minimization and a Class of Binary MRF Models. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*. Vol. 3757 of *Lecture Notes in Computer Science* Springer pp. 136–152. Cited on pages 115, 119, 121 and 122.

Chan, T. and L. Vese. 2001. Active contours without edges. In *IEEE Transactions on Image Processing*. Vol. 10 pp. 266–277. Cited on pages 115 and 117.

Chan, T., S. Esedoglu and M. Nikolova. 2006. "Algorithms for Finding Global Minimizers of Image Segmentation and Denoising Models." *SIAM Journal on Applied Mathematics* 66(5):1632–1648. Cited on pages 115 and 117.

Chen, Y., T. A. Davis, W. W. Hager and S. Rajamanickam. 2008. "Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate." *ACM Transactions on Mathematical Software (TOMS)* 35(3):22. Cited on page 80.

Ciresan, D. C., U. Meier, J. Masci and J. Schmidhuber. 2012. "Multi Column Deep Neural Network for Traffic Sign Classification." *Neural Networks (to appear)* . Cited on page 205.

Ciresan, D. C., U. Meier, L. M. Gambardella and J. Schmidhuber. 2010. "Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition." Neural Computation, Volume 22, Number 12, December 2010. Cited on page 205.

Ciresan, D. C., U. Meier, L. M. Gambardella and J. Schmidhuber. 2011. Convolutional Neural Network Committees For Handwritten Character Classification. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*. IEEE pp. 1135–1139. Cited on page 205.

Constantinides, C., Y. Chenoune, N. Kachenoura, E. Roullot, E. Mousseaux, A. Herment and F. Frouin. 2009. "Semi-automated cardiac segmentation on cine magnetic resonance images using GVF-Snake deformable models.". Cited on page 186.

Cordelli, E. and P. Soda. 2011. Color to grayscale staining pattern representation in IIF. In *CBMS*. IEEE pp. 1–6. Cited on pages 204 and 208.

Coyne, J. A. 2009. *Why evolution is true*. Oxford University Press. Cited on page 1.

Crama, Y. 1989. "Recognition problems for special classes of polynomials in 0-1 variables." *Math. Programming* 44(1–3):139–155. Cited on page 22.

Cremers, D. and L. Grady. 2006. Statistical Priors for Efficient Combinatorial Optimization Via Graph Cuts. In *European Conf. Computer Vision*. Graz, Austria: . Cited on page 20.

Criminisi, A., J. Shotton and E. Konukoglu. 2011. Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning. Technical Report MSR-TR-2011-114 Microsoft Research. Cited on page 206.

Crow, F. 1984. "Summed-area tables for texture mapping." 18(3):207–212. Cited on page 140.

Darbon, J. 2007. A Note on the Discrete Binary Mumford-Shah Model. In *MIRAGE*. Vol. 4418 of *Lecture Notes in Computer Science* Springer pp. 283–294. Cited on page 118.

Delong, A., A. Osokin, H. Isack and Y. Boykov. 2010. Fast approximate energy minimization with label costs. In *Conf. Computer Vision and Pattern Recognition*. Cited on page 20.

Delong, A. and Y. Boykov. 2008. A Scalable graph-cut algorithm for N-D grids. In *Conf. Computer Vision and Pattern Recognition*. Cited on pages 85 and 217.

Delong, A. and Y. Boykov. 2009. Globally Optimal Segmentation of Multi-Region Objects. In *Int. Conf. Computer Vision*. Cited on pages 31, 175 and 180.

Dixit, N., R. Keriven and N. Paragios. 2005. GPU-Cuts: Combinatorial Optimisation, Graphic Processing Units and Adaptive Object Extraction. Technical Report 05-07 CERTIS. Cited on page 85.

El-Zehiry, N. and L. Grady. 2010. Fast Global Optimization of Curvature. In *Conf. Computer Vision and Pattern Recognition*. Cited on pages 144, 146 and 147.

Everett, H. 1963. "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources." *Operations Research* 11:399–417. Cited on page 13.

Felzenszwalb, P. F. and D. McAllester. 2007. "The generalized A* architecture." *Journal of Artificial Intelligence Research* 29(1):153–190. Cited on page 159.

Fix, A., A. Grubner, E. Boros and R. Zabih. 2011. A Graph Cut Algorithm for Higher-order Markov Random Fields. In *Int. Conf. Computer Vision*. Barcelona, Spain: . Cited on pages 28, 38, 59, 61, 65, 77, 81 and 82.

Foggia, P., G. Percannella, P. Soda and M. Vento. 2010. Early experiences in mitotic cells recognition on HEp-2 slides. In *CBMS*, ed. T. S. Dillon, D. L. Rubin, W. Gallagher, A. S. Sidhu and A. Tsymbal. IEEE pp. 38–43. Cited on page 205.

Fredriksson, J., C. Olsson, P. Strandmark and F. Kahl. 2012. Tighter Relaxations for Higher-Order Models based on Generalized Roof Duality. In *HiPot: ECCV 2012 Workshop on Higher-Order Models and Global Constraints in Computer Vision*. Cited on page 216.

Freedman, D. and P. Drineas. 2005. Energy Minimization via Graph Cuts: Settling What is Possible. In *Conf. Computer Vision and Pattern Recognition*. San Diego, USA: . Cited on page 38.

Fujishige, S. 2005. *Submodular functions and optimization*. Vol. 58 Elsevier Science. Cited on page 21.

Fujishige, S. and S. Iwata. 2006. "Bisubmodular Function Minimization." *SIAM J. Discrete Math.* 19(4):1065–1073. Cited on page 42.

Gallagher, A. C., D. Batra and D. Parikh. 2011. Inference for Order Reduction in Markov Random Fields. In *Conf. Computer Vision and Pattern Recognition*. Colorado Springs, USA: . Cited on page 38.

Glover, F., B. Alidaee, C. Rego and G. Kochenberger. 2002. "One-pass heuristics for large-scale unconstrained binary quadratic problems." *European Journal of Operational Research* 137(2):272–287. Cited on page 19.

Goldberg, A. V. and R. E. Tarjan. 1986. A new approach to the maximum flow problem. In *ACM symposium on Theory of computing*. pp. 136–146. Cited on page 85.

Goldberg, A. V., S. Hed, H. Kaplan, R. E. Tarjan and R. F. Werneck. 2011. Maximum Flows By Incremental Breadth-First Search. In *European Symposium on Algorithms, ALGO ESA*. Cited on pages 73 and 193.

Goldschlager, L. M., R. A. Shaw and J. Staples. 1982. "The Maximum Flow Problem is Log Space Complete for P." *Theoretical Computer Science* 21(1):105–111. Cited on page 84.

Gonzalez, R. C., R. E. Woods and S. L. Eddins. 2007. *Digital Image Processing Using MATLAB*. Prentice Hall Press. Cited on page 207.

Goodman, Jacob E. and Joseph O'Rourke, eds. 1997. *Handbook of discrete and computational geometry*. CRC Press, Inc. Cited on page 179.

Grady, L. 2010. "Minimal Surfaces Extend Shortest Path Segmentation Methods to 3D." *IEEE Trans. on Pattern Analysis and Machine Intelligence* 32(2):321–334. Cited on pages 139 and 168.

Grady, L. and M. P. Jolly. 2008. Weights and topology: A study of the effects of graph construction on 3d image segmentation. In *Conf. Medical Image Computing and Computer-Assisted Intervention*. Cited on page 179.

Greig, D. M., B. T. Porteous and A. H. Seheult. 1989. "Exact Maximum A Posteriori Estimation for Binary Images." *Journal of the Royal Statistical Society* . Cited on page 83.

Hales, T. C. 2001. "The Honeycomb Conjecture." *Discrete & Computational Geometry* 25(1):1–22. Cited on page 151.

Hammer, P. L., P. Hansen and B. Simeone. 1984. "Roof duality, Complementation and Persistency in Quadratic 0-1 Optimization." *Math. Programming* 28(2):121–155. Cited on pages 20, 25, 26, 46, 68 and 71.

Haralick, R. M., K. Shanmugam and I. H. Dinstein. 1973. "Textural features for image classification." *Systems, Man and Cybernetics, IEEE Transactions on* 3(6):610–621. Cited on page 207.

Hoos, H. H. and T. Stützle. 2000. SATLIB: An Online Resource for Research on SAT. In *SAT 2000*. IOS Press pp. 283–292. Cited on page 66.

Hsu, Lucas, Rob Kusner and John Sullivan. 1992. "Minimizing the squared mean curvature integral for surfaces in space forms." *Experimental Mathematics* 1:191–207. Cited on page 167.

Hu, S., E. A. Hoffman and J. M. Reinhardt. 2001. "Automatic lung segmentation for accurate quantitation of volumetric X-ray CT images." *Medical Imaging, IEEE Transactions on* 20(6):490 –498. Cited on page 194.

Huang, S., J. Liu, L. Lee, S. Venkatesh, L. Teo, C. Au and W. Nowinski. 2009. "Segmentation of the Left Ventricle from Cine MR Images Using a Comprehensive Approach.". Cited on page 186.

Hussein, M., A. Varshney and L. Davis. 2007. On Implementing Graph Cuts on CUDA. In *Workshop on General Purpose Processing on Graphics Processing Units*. Cited on page 85.

Ishikawa, H. 2009*a*. Higher-Order Clique Reduction in Binary Graph Cut. In *Conference on Computer Vision and Pattern Recognition*. Cited on pages 77 and 79.

Ishikawa, H. 2009*b*. Higher-order gradient descent by fusion-move graph cut. In *International Conference on Computer Vision*. Cited on pages 77, 79, 80, 81 and 82.

Ishikawa, H. 2011. "Transformation of General Binary MRF Minimization to the First Order Case." *IEEE Trans. Pattern Analysis and Machine*

Intelligence 33(6):1234–1249. Cited on pages 27, 28, 38, 54, 59, 60, 77, 79, 80 and 81.

Ivănescu, P. L. (Hammer). 1965. "Some Network Flow Problems Solved with Pseudo-Boolean Programming." *Operations Research* 13(3):388–399. Cited on page 24.

Iwata, S. 2001. "A Faster Scaling Algorithm for Minimizing Submodular Functions." *SIAM Journal on Computing* 32:833–840. Cited on page 21.

Iwata, S. 2002. "A Fully Combinatorial Algorithm for Submodular Function Minimization." *Journal of Combinatorial Theory, Series B* 84:203–212. Cited on page 21.

Jamriska, O., D. Sykora and A. Hornung. 2012. Cache-efficient graph cuts on structured grids. In *Conf. Computer Vision and Pattern Recognition*. Cited on page 217.

Jancsary, J., S. Nowozin and C. Rother. 2012. Loss-Specific Training of Non-Parametric Image Restoration Models: A New State of the Art. In *European Conference on Computer Vision*. Cited on page 77.

Jolly, M. 2009. "Fully Automatic Left Ventricle Segmentation in Cardiac Cine MR Images Using Registration and Minimum Surfaces.". Cited on page 186.

Kahl, F. and P. Strandmark. 2011. Generalized Roof Duality for Pseudo-Boolean Optimization. In *Int. Conf. Computer Vision*. Barcelona, Spain: . Cited on pages 9 and 215.

Kahl, F. and P. Strandmark. 2012. "Generalized roof duality." *Discrete Applied Mathematics* 160(16–17):2419–2434. Cited on page 9.

Kappes, J. H., B. Savchynskyy and C. Schnorr. 2012. A bundle approach to efficient MAP-inference by Lagrangian relaxation. In *Conf. Computer Vision and Pattern Recognition*. Cited on page 217.

Kirsanov, D. and S. J. Gortler. 2004. A Discrete Global Minimization Algorithm for Continuous Variational Problems. Technical Report TR-14-04 Harvard. Cited on page 152.

Klodt, M., T. Schoenemann, K. Kolev, M. Schikora and D. Cremers. 2008. An Experimental Comparison of Discrete and Continuous Shape Optimization Methods. In *European Conf. Computer Vision*. Cited on page 86.

Kohli, P., A. Shekhovtsov, C. Rother, V. Kolmogorov and P. H. S. Torr. 2008. On partial optimality in multi-label mrfs. In *Int. Conf. Machine learning*. ACM. Cited on page 216.

Kohli, P., M. P. Kumar and P. H. S. Torr. 2009. "P^3 & Beyond: Move Making Algorithms for Solving Higher Order Functions." *IEEE Trans. Pattern Analysis and Machine Intelligence* 31(9):1645–1656. Cited on page 20.

Kohli, P. and P. H. S. Torr. 2005. Efficiently Solving Dynamic Markov Random Fields Using Graph Cuts. In *Int. Conf. Computer Vision*. Cited on page 84.

Kohli, P. and P. H. S. Torr. 2007. "Dynamic graph cuts for efficient inference in markov random fields." *IEEE Trans. Pattern Analysis and Machine Intelligence* . Cited on pages 84 and 89.

Kolda, T. G., R. M. Lewis and V. Torczon. 2003. "Optimization by direct search: New perspectives on some classical and modern methods." *SIAM review* pp. 385–482. Cited on page 30.

Kolev, K. and D. Cremers. 2009. Continuous Ratio Optimization via Convex Relaxation with Applications to Multiview 3D Reconstruction. In *Conf. Computer Vision and Pattern Recognition*. Cited on page 129.

Kolmogorov, V. 2010. Generalized roof duality and bisubmodular functions. In *Neural Information Processing Systems*. Cited on pages 38, 39, 42 and 69.

Kolmogorov, V. 2011. "Submodularity on a Tree: Unifying $L^\natural$-Convex and Bisubmodular Functions." *Mathematical Foundations of Computer Science 2011* pp. 400–411. Cited on page 216.

Kolmogorov, V. 2012. "Generalized roof duality and bisubmodular functions." *Discrete Applied Mathematics* 160(4–5):416–426. Cited on page 44.

Kolmogorov, V. and C. Rother. 2007. "Minimizing nonsubmodular functions with graph cuts - A review." *IEEE Trans. Pattern Analysis and Machine Intelligence* 29(7):1274–1279. Cited on pages 19, 20 and 37.

Kolmogorov, V. and R. Zabih. 2004. "What energy functions can be minimized via graph cuts?" *IEEE Trans. Pattern Analysis and Machine Intelligence* 26(2):147–159. Cited on page 31.

Kolmogorov, V. and R. Zabih. 2006. Graph Cut Algorithms for Binocular Stereo with Occlusions. In *Handbook of Mathematical Models in Computer Vision*. Springer. Cited on page 197.

Kolmogorov, V. and Y. Boykov. 2005. What Metrics Can Be Approximated by Geo-Cuts, Or Global Optimization of Length/Area and Flux. In *Int. Conf. Computer Vision*. Cited on page 93.

Kolmogorov, V., Y. Boykov and C. Rother. 2007. Applications of parametric maxflow in computer vision. In *Int. Conf. Computer Vision*. Cited on pages 129 and 130.

Komodakis, N. and G. Tziritas. 2007. "Approximate Labeling via Graph Cuts Based on Linear Programming." *IEEE Trans. Pattern Analysis and Machine Intelligence* 29(8):1436–1453. Cited on pages 83 and 106.

Komodakis, N. and N. Paragios. 2009. Beyond pairwise energies: Efficient optimization for higher-order MRFs. In *Conf. Computer Vision and Pattern Recognition*. Miami, USA: . Cited on page 20.

Komodakis, N., N. Paragios and G. Tziritas. 2007. MRF Optimization via Dual Decomposition: Message-Passing Revisited. In *Int. Conf. Computer Vision*. Cited on pages 6, 13, 83, 90, 101 and 105.

Komodakis, N., N. Paragios and G. Tziritas. 2011. "MRF Energy Minimization and Beyond via Dual Decomposition." *IEEE Trans. Pattern Analysis and Machine Intelligence* 33(3):531–552. Cited on page 106.

Kushal, A. and J. Ponce. 2006. Modeling 3D objects from stereo views and recognizing them in photographs. In *European Conf. Computer Vision*. Cited on page 199.

Ladicky, L., C. Russell, P. Kohli and P. H. S. Torr. 2010. Graph Cut Based Inference with Co-occurrence Statistics. In *European Conf. Computer Vision*. Crete, Greece: . Cited on page 20.

Lan, X., S. Roth, D. Huttenlocher and M. Black. 2006*a*. "Efficient belief propagation with learned higher-order markov random fields." *European Conference on Computer Vision* . Cited on pages 77 and 82.

Lan, X., S. Roth, D. P. Huttenlocher and M.J. Black. 2006*b*. Efficient Belief Propagation with Learned Higher-Order Markov Random Fields. In *European Conf. Computer Vision*. Graz, Austria: . Cited on page 20.

Lempitsky, V., A. Blake and C. Rother. 2008. Image Segmentation by Branch-and-Mincut. In *European Conf. Computer Vision*. Marseille, France: pp. 15–29. Cited on page 118.

Lempitsky, V., A. Blake and C. Rother. 2012. "Branch-and-Mincut: Global Optimization for Image Segmentation with High-Level Priors." *Journal of Mathematical Imaging and Vision* . in press. Cited on page 218.

Lempitsky, V., C. Rother, S. Roth and A. Blake. 2010. "Fusion Moves for Markov Random Field Optimization." *IEEE Trans. Pattern Analysis and Machine Intelligence* 32(8):1392–1405. Cited on pages 20, 29 and 86.

Lempitsky, V. and Y. Boykov. 2007. Global Optimization for Shape Fitting. In *Conf. Computer Vision and Pattern Recognition*. Minneapolis, USA: . Cited on pages 83, 95 and 98.

Lin, X., B. Cowan and A. Young. 2006. Model-based graph cut method for segmentation of the left ventricle. In *IEEE-EMBS*. Cited on page 3.

Liu, C., J. Yuen, A. Torralba, J. Sivic and W. T. Freeman. 2008. SIFT Flow: Dense Correspondence across Different Scenes. In *European Conf. Computer Vision*. Cited on pages 196 and 200.

Liu, J. and J. Sun. 2010. Parallel Graph-cuts by Adaptive Bottom-up Merging. In *Conf. Computer Vision and Pattern Recognition*. Cited on pages 85, 96 and 99.

Liu, J., J. Sun and H.-Y. Shum. 2009. "Paint selection." *ACM Transactions on Graphics* 28(3):1–7. Cited on pages 85 and 99.

Lorenzo-Valdés, M. et al. 2004. "Segmentation of 4D cardiac MR images using a probabilistic atlas and the EM algorithm." *Medical Image Analysis* 8(3):255–265. Cited on page 3.

Lovász, L. 1983. "Submodular functions and convexity." *Mathematical Programming: The State of the Art* pp. 235–257. Cited on pages 21 and 37.

Lowe, D. 2004. "Distinctive image features from scale-invariant keypoints." *Int. Journal Computer Vision* 20:91–110. Cited on page 197.

Lu, S. H. and A. C. Williams. 1987. "Roof duality for polynomial 0-1 optimization." *Math. Programming* 37(3):357–360. Cited on page 38.

Lu, Y., P. Radau, K. Connelly, A. Dick and G. Wright. 2009. "Automatic Image-Driven Segmentation of Left Ventricle in Cardiac Cine MRI.". Cited on page 186.

Madsen, K., H. Bruun and O. Tingleff. 1999. Methods for non-linear least squares problems. Technical report Informatics and Mathematical Modelling, Technical University of Denmark. Cited on page 18.

Marak, L., J. Cousty, L. Najman and H. Talbot. 2009. "4D Morphological segmentation and the MICCAI LV-segmentation grand challenge.". Cited on page 186.

Martin, D., C. Fowlkes, D. Tal and J. Malik. 2001. A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *Int. Conf. Computer Vision*. Cited on pages 81, 92, 94 and 126.

McIntosh, C. and G. Hamarneh. 2011. "Medial-based Deformable Models in Non-convex Shape-spaces for Medical Image Segmentation using Genetic Algorithms." *IEEE transactions on medical imaging* . Cited on page 2.

Middleton, L. and J. Sivaswamy. 2005. *Hexagonal Image Processing: A Practical Approach*. Springer-Verlag New York, Inc. Cited on page 151.

Mitchell, S. C., B. P. F. Lelieveldt, R. J. van der Geest, H. G. Bosch, J. H. C. Reiber and M. Sonka. 2002. "Multistage hybrid active appearance model matching: segmentation of left and right ventricles in cardiac MR images." *IEEE Trans. Medical Imaging* 20(5):415–423. Cited on page 3.

Mumford, D. and T. Shah. 1989. Optimal Approximation by Piecewise Smooth Functions and Associated Variational Problems. In *Comm. on Pure and Applied Mathematics*. Cited on pages 4, 115 and 117.

Nemhauser, G. L., L. A. Wolsey and M. L. Fisher. 1978. "An analysis of approximations for maximizing submodular set functions – I." *Matematical Programming* 14(1):265–294. Cited on page 21.

Nemhauser, G. L. and L. E. Trotter. 1975. "Vertex packings: Structural properties and algorithms." *Mathematical Programming* 8:232–248. Cited on page 71.

Nesterov, Y. 2004. *Introductory lectures on convex optimization: A basic course*. Kluwer Academic Publishers. Cited on pages 11, 12, 15 and 16.

Nielsen, H. B. 1999. Damping parameter in Marquardt's method. Technical Report IMM-REP-1999-05 Department of Mathematical Modelling, Technical University of Denmark. Cited on page 18.

Nilsson, D.-E. 1999. "Vision Optics and Evolution." *BioScience* 39(5):298–307. Cited on page 1.

Nocedal, J. and S. J. Wright. 2006. *Numerical Optimization*. Springer. Cited on pages 11, 18, 80 and 162.

Nowozin, S. and C. H. Lampert. 2010. "Global Interactions in Random Field Models: A Potential Function Ensuring Connectedness." *SIAM J. Imaging Sciences* 3(4):1048–1074. Cited on page 20.

O'Brien, S., O. Ghita and P. Whelan. 2009. "Segmenting the Left Ventricle in 3D Using a Coupled ASM and a Learned Non-Rigid Spatial Model.". Cited on page 186.

Olsson, C., M. Byröd, N. C. Overgaard and F. Kahl. 2009. Extending Continuous Cuts: Anisotropic Metrics and Expansion Moves. In *International Conference on Computer Vision*. Cited on page 6.

Papadimitriou, C. H. and K. Steiglitz. 1998. *Combinatorial Optimization; Algorithms and Complexity*. Dover Publications. Cited on page 92.

Paragios, N. 2002. "A Variational Approach for the Segmentation of the Left Ventricle in Cardiac Image Analysis." *Int. Journal Computer Vision* 50(3):345–362. Cited on page 3.

Percannella, G., P. Soda and M. Vento. 2011. Mitotic HEp-2 cells recognition under class skew. In *Int. Conf. Image Analysis and Processing – Volume Part II*. Springer-Verlag pp. 353–362. Cited on pages 203 and 205.

Perner, P., H. Perner and B. Müller. 2002. "Mining knowledge for HEp-2 cell image classification." *Artificial Intelligence in Medicine* 26:161–173. Cited on page 205.

Pock, T. 2008. Fast Total Variation for Computer Vision PhD thesis Graz University of Technology. Cited on page 124.

Pock, T., M. Unger, D. Cremers and H. Bischof. 2008. Fast and Exact Solution of Total Variation Models on the GPU. In *CVPR Workshop on Visual Computer Vision on GPUs*. Cited on pages 101 and 124.

Potetz, B. 2007. Efficient belief propagation for vision using linear constraint nodes. In *Conference on Computer Vision and Pattern Recognition*. Cited on pages 77 and 82.

Pressley, A. 2010. *Elementary differential geometry*. Springer. Cited on page 170.

Pritch, Y., E. Kav-Venaki and S. Peleg. 2009. Shift-Map Image Editing. In *Int. Conf. Computer Vision*. Cited on pages 104, 195, 197, 198, 199, 200 and 201.

Promislow, S. D. and V. R. Young. 2005. "Supermodular Functions on Finite Lattices." *Order* 22(4):389–413. Cited on pages 38 and 55.

Rantzer, A. 2009. Dynamic Dual Decomposition for Distributed Control. In *American Control Conference*. Cited on page 13.

Raphael, C. 2001. "Coarse-to-fine dynamic programming." *IEEE Trans. Pattern Analysis and Machine Intelligence* 23(12):1379–1390. Cited on pages 157 and 159.

Raphael, C. S. and S. Geman. 1997. Grammatical approach to mine detection. In *Proceedings of SPIE*. Vol. 3079 p. 316. Cited on page 157.

Renka, Robert J. 1997. "Algorithm 772: STRIPACK: Delaunay triangulation and Voronoi diagram on the surface of a sphere." *ACM Trans. Math. Softw.* 23:416–434. Cited on page 179.

Rhys, J. 1970. "A selection problem of shared fixed costs and networks." *Management Science* 17:200–207. Cited on page 25.

Rigon, A., P. Soda, D. Zennaro, G. Iannello and A. Afeltra. 2007. "Indirect immunofluorescence in autoimmune diseases: assessment of digital images for diagnostic purpose." *Cytometry Part B: Clinical Cytometry* 72(6):472–477. Cited on page 203.

Roth, S. and M. J. Black. 2009. "Fields of experts." *International Journal of Computer Vision* 82(2):205–229. Cited on pages 77 and 78.

Rother, C., P. Kohli, W. Feng and J. Jia. 2009. Minimizing sparse higher order energy functions of discrete variables. In *Conf. Computer Vision and Pattern Recognition*. Cited on page 38.

Rother, C., V. Kolmogorov and A. Blake. 2004. "GrabCut": interactive foreground extraction using iterated graph cuts. In *ACM Transactions on Graphics*. pp. 309–314. Cited on page 83.

Rother, C., V. Kolmogorov, V. Lempitsky and M. Szummer. 2007*a*. Optimizing Binary MRFs via Extended Roof Duality. In *Conf. Computer Vision and Pattern Recognition*. Cited on pages 26 and 188.

Rother, C., V. Kolmogorov, V. Lempitsky and M. Szummer. 2007*b*. Optimizing Binary MRFs via Extended Roof Duality. Technical Report MSR-TR-2007-46 Microsoft. Cited on page 26.

Rousson, M. and R. Deriche. 2002. A variational framework for active and adaptative segmentation of vector valued images. In *In Proc. IEEE Workshop on Motion and Video Computing*. pp. 56–62. Cited on page 132.

Rudin, L. I., S. Osher and E. Fatemi. 1992. "Nonlinear Total Variation Based Noise Removal Algorithms." *Physica D* 60:259–268. Cited on page 121.

Sack, U., S. Knoechner, H. Warschkau, U. Pigla, F. Emmrich and M. Kamprad. 2003. "Computer-assisted classification of HEp-2 immunofluorescence patterns in autoimmune diagnostics." *Autoimmunity Reviews* 2(5):298–304. Cited on page 205.

Sarti, A., C. Corsi, E. Mazzini and C. Lamberti. 2004. "Maximum likelihood segmentation with Rayleigh distribution of ultrasound images." *Computers in Cardiology* pp. 329–332. Cited on page 132.

Scheuermann, B. and B. Rosenhahn. 2011. Slimcuts: Graphcuts for high resolution images using graph reduction. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer pp. 219–232. Cited on page 217.

Schoenemann, T., F. Kahl and D. Cremers. 2009. Curvature Regularity for Region-based Image Segmentation and Inpainting: A Linear Programming Relaxation. In *Int. Conf. Computer Vision*. Cited on pages 137, 138, 139, 140, 141, 142 and 152.

Schoenemann, T., F. Kahl, S. Masnou and D. Cremers. 2012. "A Linear Framework for Region-Based Image Segmentation and Inpainting Involving Curvature Penalization." *Int. Journal Computer Vision* . Cited on pages 6 and 137.

Schwing, A., T. Hazan, M. Pollefeys and R. Urtasun. 2011. Distributed Message Passing for Large Scale Graphical Models. In *Conf. Computer Vision and Pattern Recognition*. Cited on page 217.

Shekhovtsov, A., P. Kohli and C. Rother. 2012. "Curvature prior for MRF-based segmentation and shape inpainting." *Pattern Recognition* pp. 41–51. Cited on pages 218 and 219.

Shekhovtsov, A. and V. Hlavac. 2011*a*. A Distributed Mincut/Maxflow Algorithm Combining Path Augmentation and Push-Relabel. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*. Vol. 6819 of *Lecture Notes in Computer Science* Springer. Cited on page 217.

Shekhovtsov, A. and V. Hlavac. 2011*b*. A Distributed Mincut/Maxflow Algorithm Combining Path Augmentation and Push-Relabel. Research report K333–43/11, CTU–CMP–2011–03 Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University. Cited on page 217.

Snir, M. and S. Otto. 1998. *MPI—The Complete Reference: The MPI Core*. Cambridge, MA, USA: MIT Press. Cited on page 97.

Soda, P. and G. Iannello. 2009. "Aggregation of classifiers for staining pattern recognition in antinuclear autoantibodies analysis." *Trans. Info. Tech. Biomed.* 13:322–329. Cited on pages 205 and 208.

Stallkamp, J., M. Schlipsing, J. Salmen and C. Igel. in press. "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition." *Neural Networks* . Cited on page 205.

Strandmark, P. 2010. Early Vision Optimization: Parametric Models, Parallelization and Curvature. Licentiate thesis, Lund University. Cited on pages 4 and 10.

Strandmark, P. 2012. "Persistency for Higher-Order Pseudo-Boolean Optimization." *Tiny Transactions on Computer Science* 1. (Not cited.)

Strandmark, P. and F. Kahl. 2010*a*. Parallel and Distributed Graph Cuts. In *Swedish Symposium on Image Analysis*. Cited on page 10.

Strandmark, P. and F. Kahl. 2010*b*. Parallel and Distributed Graph Cuts by Dual Decomposition. In *Conf. Computer Vision and Pattern Recognition*. San Francisco, USA: . Cited on pages 8 and 99.

Strandmark, P. and F. Kahl. 2011*a*. Curvature Regularization for Curves and Surfaces in a Global Optimization Framework. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*. Vol. 6819 of *Lecture Notes in Computer Science* Springer. Cited on page 9.

Strandmark, P. and F. Kahl. 2011*b*. Mesh Types for Curvature Regularization. In *Swedish Symposium on Image Analysis*. Cited on page 10.

Strandmark, P. and F. Kahl. 2012. Pseudo-Boolean Optimization: Theory and Applications in Vision. In *Swedish Symposium on Image Analysis*. Cited on page 10.

Strandmark, P., F. Kahl and N. C. Overgaard. 2009*a*. Optimal Levels for the Two-phase, Piecewise Constant Mumford-Shah Functional. In *Swedish Symposium on Image Analysis*. Cited on page 10.

Strandmark, P., F. Kahl and N. C. Overgaard. 2009*b*. Optimizing Parametric Total Variation Models. In *Int. Conf. Computer Vision*. Cited on pages 8 and 133.

Strandmark, P., F. Kahl and T. Schoenemann. 2011. "Parallel and distributed vision algorithms using dual decomposition." *Computer Vision and Image Understanding* 115(12):1721–1732. Cited on page 8.

Strandmark, P., J. Ulén and F. Kahl. 2012. HEp-2 Staining Pattern Classification. In *Int. Conf. Pattern Recognition*. Cited on page 10.

Strekalovskiy, E. and D. Cremers. 2011. Generalized Ordering Constraints for Multilabel Optimization. In *Int. Conf. Computer Vision*. Cited on page 159.

Sullivan, J. M. 1990. Crystalline Approximation Theorem for Hypersurfaces. Phd thesis, Princeton Univ. Cited on page 139.

Svärm, L. and P. Strandmark. 2010*a*. Shift-map Image Registration. In *Int. Conf. Pattern Recognition*. Cited on page 8.

Svärm, L. and P. Strandmark. 2010*b*. Shift-map Image Registration. In *Swedish Symposium on Image Analysis*. Cited on page 10.

Tola, E., V. Lepetit and P. Fua. 2009. "DAISY: An Efficient Dense Descriptor Applied to Wide Baseline Stereo." *IEEE Trans. Pattern Analysis and Machine Intelligence* . Cited on page 197.

Triggs, B., P. McLauchlan, R. Hartley and A. Fitzgibbon. 2000. "Bundle adjustment—a modern synthesis." *Vision algorithms: theory and practice* pp. 153–177. Cited on page 78.

Turing, A. 1950. "Computing Machinery and Intelligence." *Mind* LIX(236):433–460. Cited on page 1.

Ukil, S. and J. M. Reinhardt. 2009. "Anatomy-guided lung lobe segmentation in X-ray CT images." *IEEE Trans. Medical Imaging* 28(2):202–214. Cited on page 194.

Ulén, J., P. Strandmark and F. Kahl. 2011. Optimization for Multi-Region Segmentation of Cardiac MRI. In *MICCAI Workshop on Statistical Atlases and Computational Models of the Heart: Imaging and Modelling Challenges*. Cited on page 9.

Ulén, J., P. Strandmark and F. Kahl. 2013. "An Efficient Optimization Framework for Multi-Region Segmentation based on Lagrangian Duality." *IEEE Trans. Medical Imaging* . To appear. Cited on page 9.

Unger, M., T. Pock, D. Cremers and H. Bischof. 2008. TVSeg - Interactive Total Variation Based Image Segmentation. In *British Machine Vision Conf.* Cited on page 101.

Vineet, V. and P. J. Narayanan. 2008. CUDA cuts: Fast graph cuts on the GPU. In *Computer Vision and Pattern Recognition Workshops, CVPRW*. Cited on pages 85, 106 and 108.

Vineet, V. and P. J. Narayanan. 2009. Solving Multi-label MRFs using Incremental alpha-expansion move on the GPUs. In *Asian Conference on Computer Vision*. Cited on page 85.

Viola, P. and M. Jones. 2004. "Robust Real-Time Face Detection." *Int. Journal Computer Vision* 57(2):137–154. Cited on page 140.

Wardetzky, M., M. Bergou, D. Harmon, D. Zorin and E. Grinspun. 2007. "Discrete quadratic curvature energies." *Comput. Aided Geom. Des.* 24(8-9):499–518. Cited on pages 168 and 169.

Werlberger, M., W. Trobin, T. Pock, A. Wedel, D. Cremers and H. Bischof. 2009. Anisotropic Huber-L1 Optical Flow. In *British Machine Vision Conf.* Cited on page 101.

Werner, T. 2008. High-arity interactions, polyhedral relaxations, and cutting plane algorithm for MAP-MRF. In *Conf. Computer Vision and Pattern Recognition*. Anchorage, USA: . Cited on page 20.

Wijnhout, J., D. Hendriksen, H. van Assen and R. van der Geest. 2009. "LV Challenge LKEB Contribution: Fully Automated Myocardial Contour Detection.". Cited on page 186.

Willmore, T. J. 1965. "Note on Embedded Surfaces." *An. Sti. Univ. "Al. I. Cuza" Iasi Sect. I a Mat. (N.S.)* pp. 493–496. Cited on page 167.

Winder, S., G. Hua and M. Brown. 2009. Picking the best DAISY. In *Conf. Computer Vision and Pattern Recognition*. pp. 178–185. Cited on page 197.

Windheuser, T., H. Ishikawa and D. Cremers. 2012. Generalized Roof Duality for Multi-Label Optimization: Optimal Lower Bounds and Persistency. In *European Conference on Computer Vision*. Cited on pages 69, 215 and 216.

Woodford, O. J., P. H. S. Torr, I. D. Reid and A. W. Fitzgibbon. 2009. "Global Stereo Reconstruction under Second-Order Smoothness Priors." *IEEE Trans. Pattern Analysis and Machine Intelligence* 31(12):2115–2128. Cited on pages 20 and 62.

Xu, M., P. M. Thompson and A. W. Toga. 2004. "An Adaptive Level Set Segmentation on a Triangulated Mesh." *IEEE Trans. on Medical Imaging* 23(2):191–201. Cited on page 152.

Živný, S., D. A. Cohen and P. G. Jeavons. 2009. "The Expressive Power of Binary Submodular Functions." *Discrete Appl. Math.* 157(15):3347–3358. Cited on pages 22, 38, 46, 47, 53 and 69.

Živný, S. and P. G. Jeavons. 2008. Which submodular functions are expressible using binary submodular functions? Technical Report CS-RR-08-08 Oxford University Computing Laboratory. Cited on page 55.

Živný, S. and P. G. Jeavons. 2010. "Classes of Submodular Constraints Expressible by Graph Cuts." *Constraints* 15(3):430–452. Cited on page 53.

Index