Efficient Inference and Learning for Computer Vision Labelling Problems

Karteek Alahari

Thesis submitted in partial fulfillment of the requirements of the award of

Doctor of Philosophy

Oxford Brookes University

2010

Abstract

Discrete energy minimization has recently emerged as an indispensable tool for computer vision problems. It enables inference of the maximum a posteriori solutions of Markov and conditional random fields, which can be used to model labelling problems in vision. When formulating such problems in an energy minimization framework, there are three main issues that need to be addressed: (i) How to perform efficient inference to compute the optimal solution; (ii) How to incorporate prior knowledge into the model; and (iii) How to learn the parameter values. This thesis focusses on these aspects and presents novel solutions to address them.

As computer vision moves towards the era of large videos and gigapixel images, computational efficiency is becoming increasingly important. We present two novel methods to improve the efficiency of energy minimization algorithms. The first method works by "recycling" results from previous problem instances. The second simplifies the energy minimization problem by "reducing" the number of variables in the energy function. We demonstrate a substantial improvement in the running time of various labelling problems such as, interactive image and video segmentation, object recognition, stereo matching.

In the second part of the thesis we explore the use of natural image statistics for the single view reconstruction problem, where the task is to recover a theatre-stage representation (containing planar surfaces and their geometrical relationships to each other) from a single 2D image. To this end, we introduce a class of multi-label higher order functions to model these statistics based on the distribution of geometrical features of planar surfaces. We also show that this new class of functions can be solved exactly with efficient graph cut methods.

The third part of the thesis addresses the problem of learning the parameters of the energy function. Although several methods have been proposed to learn the model parameters from training data, they suffer from various drawbacks, such as limited applicability or noisy estimates due to poor approximations. We present an accurate and efficient learning method, and demonstrate that it is widely applicable.

To *Amamma*

Acknowledgements

My time in Oxford has been a thrilling and rewarding experience, thanks to many people. This is my attempt at thanking as many as I can.

Firstly, I am profoundly grateful to Phil Torr for his guidance, support and advice. This work would not have been possible without his intuition, research insight and relentless effort for perfection. Most of all, I would like to thank Phil for his boundless and infectious enthusiasm.

My thanks go to Pushmeet Kohli and Srikumar Ramalingam, who helped with many of the ideas presented in this thesis. I also thank Chris Russell, Sunando Sengupta and Paul Sturgess for proof-reading the thesis, and thus taking the blame for any undetected typos! Roberto Cipolla and Catherine Hobbs deserve a special thanks for examining my thesis at such short notice.

The stimulating atmosphere at Brookes and Oxford vision labs has had a great impact on my development as a researcher. For that I sincerely thank the Old Gang: Matthieu Bray, Patrick Buehler, Ondřej Chum, Carl Ek, Mark Everingham, Pushmeet Kohli, M. Pawan Kumar, Lubor Ladicky, Mukta Prasad, Srikumar Ramalingam, Christophe Restif, Jon Rihan, Greg Rogez, Chris Russell, Florian Schroff, Josef Sivic, Olly Woodford; and the New Crew: Matthew Blaschko, Varun Gulshan, Sam Hare, David Jarzebowski, Glenn Sheasby, Paul Sturgess, Andrea Vedaldi, Jonathan Warrell. The support staff at Brookes, in particular Stephen Allen, Helen Bainbridge, Sue Flint, Catherine Hutchinson, Doreen Jarvis, Elizabeth Maynard, Ali McNiffe, Genevieve Whitson, deserve a special mention for their help with many administrative issues over the years. I also acknowledge the generous financial support provided by EPSRC and PASCAL Network of Excellence.

I am indebted to M. Pawan Kumar (for being an inspiration that he is, and for introducing me to Stephen Fry and QI), Carl Ek (for teaching me to appreciate the finer and important things in life), Mukta Prasad (for home-cooked Indian food, and for being Mukta), Pushmeet Kohli (for helping me start off on this long journey), Paul Sturgess (for all his swimming tips), Andrew Zisserman (for his support, many words of wisdom, and advice), Alyosha Efros (for his immense berry and ice cream knowledge), P. J. Narayanan and C. V. Jawahar (for an inspiring introduction to the field), and Jayanthi Sivaswamy (for her encouragement).

I would also like to thank: Srinika Ranasinghe (for accompanying me on many food adventures in Oxford and elsewhere), Sajida Malik (for being there, and for her love for chocolate), Patrick Buehler (for organizing fun trips), The Rowlands (for "introducing" me to my home away from home – 21 Old Road), Claire Berna (for invaluable advice on Paris and for being a good friend and housemate), David Jarzebowski (for organizing road trips), Chenoa Marquis and Hajar Masri (for their ever-entertaining company), Kiran BK, ALN Kumar and Sireesh Reddy (for always reminding me that I ought to finish my thesis in reasonable time), Valerie Watmough and Rosalyn Porter (for helping me get through a difficult time), Katie Kew (for demystifying artichokes and other things), Sandeep Kakani, Abilene Pitt and Victoria Wightman (for being good friends), David Jones, Laura Myers and Fran Woodcock (for tolerating me as a housemate for two years), Sam Hare (for the conversations), Mark Rendel (for showing me around Lord's), Katzi Emms and Ben Richardson (for teaching me a tiny bit of French, which I promise to improve upon), Mrs. Hodge (for apples from her tree), Yannis Hodges-Mameletzis (for telling me a thing or two about food), Manish Jethwa and his mum (for delicious *dhoklas*), Chan Mayt (for getting me into tennis, which I'm still no good at), and many other friends I've made over the years.

Above all, I am grateful to *Amma* and *Nanna*, without whom nothing would have been possible. They have supported me in all my not-so-conventional decisions, and are responsible for everything that I am today. They will be very pleased to know that I will no longer have a student status... at last! The unconditional love and support I have always received from my little sister means a great deal to me, and for that I thank her.

Finally, thanks also to Nigel Slater, Sir David Attenborough, Michael Palin and Stephen Fry, who are no less than Gods in my own crazy world!

Soho Square Gardens, London

19th August 2010

Contents

1 Introduction 1

1.1 Computer Vision as an Optimization Problem 2

1.2 Contributions 4

1.3 Outline of the Thesis 6

1.4 Publications 7

2 Random Fields 8

2.1 Markov Random Fields 9

2.2 Conditional Random Fields 12

2.3 Maximum A Posteriori Estimation 14

2.3.1 Submodular Energy Functions 14

2.3.2 Graph Cuts 15

2.3.3 Solving Non-submodular Energy Functions 20

2.3.4 Message Passing Algorithms 22

2.4 Example Vision Problems 23

2.4.1 Image Segmentation 23

2.4.2 Stereo Matching 25

2.5 Summary 26

3 Efficient Energy Minimization 27

3.1 Introduction 28

3.1.1 Outline of the Chapter 30

3.2 Preliminaries 31

3.2.1 Approximate Energy Minimization 33

3.2.2 Computing Partially Optimal Solutions 35

3.3 Efficient Multi-label Methods 36

3.3.1 Recycling Primal and Dual Solutions 37

3.3.2 Reducing Energy Functions 41

3.4 Solving $\mathcal{P}^n$ Potts Model Efficiently 45

3.5 Experiments 49

3.5.1 Dynamic α-expansion 53

3.5.2 Using Partially Optimal Solutions 54

3.6 Summary 59

4 Exact Inference for Higher Order CRFs 61

4.1 Introduction 62

4.1.1 Outline of the Chapter 64

4.2 Notation and Preliminaries 64

4.2.1 Graph Cuts for Energy Minimization 65

4.2.2 Submodular functions 67

4.3 Problem Statement 68

4.4 Boolean encoding for multi-label variables 69

4.5 Encoding Functions 71

4.6 Application: Single View Reconstruction 78

4.7 Summary 81

5 Efficient Piecewise Parameter Learning 83

5.1 Introduction 84

5.1.1 Outline of the Chapter 87

5.2 Preliminaries 87

5.2.1 Pseudo-likelihood 88

5.2.2 Max-Margin Learning 90

5.3 The Piecewise Model 92

5.3.1 Parameter Learning 93

5.3.2 Constraint Reformulation 94

5.4 Experimental Results 96

5.4.1 Man-made Structure Database 96

5.4.2 Middlebury-2005 Dataset 98

5.4.3 Discussion 100

5.5 Summary 100

6 Discussion 102

6.1 Our Contributions 103

6.2 Future Work 104

A Datasets 106

Bibliography 109

List of Figures

1.1 Example of an image segmentation problem 3

1.2 Examples of labelling problems 5

2.1 Example of a stereo matching problem 10

2.2 Example of an object recognition problem 10

2.3 Example of a Markov random field 11

2.4 Example of an st-graph 16

2.5 Graph reparameterization 18

2.6 Constructing an st-graph corresponding to an energy function . 20

2.7 Two examples of interactive binary segmentation 24

3.1 Example images and their ground truth labellings 30

3.2 An example showing the dynamic update of edge capacities . . 39

3.3 The number of label changes 40

3.4 Graph construction for $\mathcal{P}^n$ Potts model 47

3.5 Key frames of the 'Dayton' video sequence 50

3.6 Run-times obtained by recycling primal and dual solutions . . . 52

3.7 Comparison of run-times and solution energy 53

3.8 Performance of the partially optimal solution algorithm 56

3.9 Sample results of object-based segmentation and stereo 57

3.10 Solution energies computed using TRW-S and BP algorithms . . 58

3.11 Labelling obtained using the BP algorithm 59

4.1 Converting an energy minimization problem to an st-MINCUT problem 66

4.2 Transforming a multi-label problem to an st-MINCUT problem . 70

4.3 Graph construction for characterizing a general k^{th} order multi-label function 78

4.4 Potentials for single view reconstruction 79

4.5 Single view reconstruction results 80

5.1 Toy example of a binary image segmentation problem 88

5.2 Difference between pseudo- and piecewise pseudo- likelihoods . 90

5.3 Qualitative results on the man-made structure database 98

A.1 Middlebury-2005 images 107
A.2 Man-made structure dataset 108

List of Tables

3.1 Run-times for various labelling examples using the $\mathcal{P}^n$ model . . 54

3.2 Run-times for single MRF problems 55

4.1 Coefficients in the third order binary energy function 74

4.2 Misclassification error for some of the images 81

5.1 Quantitative results on the man-made structure database . . . 99

5.2 Quantitative results on the Middlebury-2005 database 99

Chapter 1

Introduction

Many problems in computer vision, such as image segmentation, stereo matching, object recognition, single view reconstruction, have been posed as energy minimization problems [16, 18, 36, 93, 94, 95]. Such formulations involve representing the vision task in terms of an energy or cost function. An optimal solution to the problem is then obtained by finding the minima of the energy function. This approach is becoming increasingly popular due to the availability of efficient and easy to use algorithms, such as graph cuts [1, 17, 31]. In this thesis, we focus on various aspects of energy minimization approaches in the context of computer vision problems. Specifically, we are interested in image labelling problems, wherein every pixel in the image is assigned a label from a given set.

1.1 Computer Vision as an Optimization Problem

One of the main challenges in dealing with computer vision tasks is the size of the problem. Let us consider the image segmentation problem as an example, where the task is to assign every pixel in an image a label corresponding to the segment it belongs to. Figure 1.1 shows an image used in [46] and its corresponding segmentation into four regions, namely cow, grass, trees, and sky. Given a 640×480 image with each pixel taking one of four possible labels, the energy function is composed of over $300,000$ variables, and there are over $10^{180,000}$ possible labellings in the solution space. A certain cost or energy value is associated with each of these label assignments, and the lowest cost labelling corresponds to the optimal solution. Naturally, searching for the best solution (also referred to as the *Inference* problem) in such an extremely large space requires efficient optimization algorithms.

Although the problem of finding the minima of a general energy function is NP-hard [18], there exist a number of powerful algorithms which compute the exact solution for a particular family of functions in polynomial time. For instance, max-product belief propagation algorithm exactly minimizes energy functions defined over graphs with no loops [75, 115]. Similarly, certain energy functions can

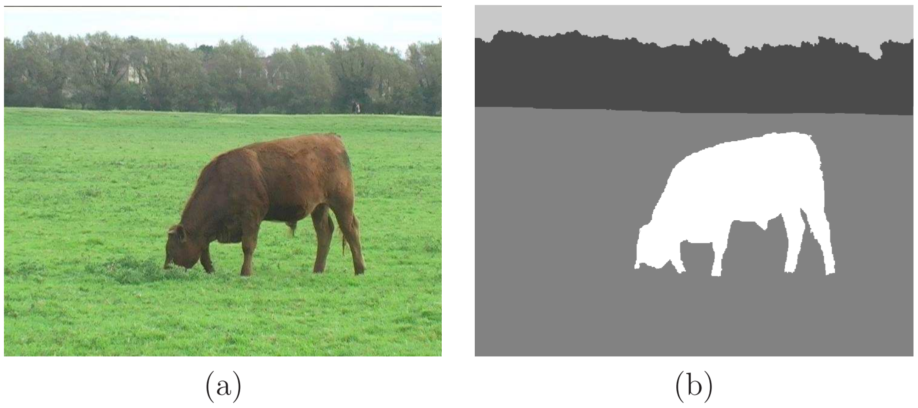


Figure 1.1: (a) A natural image used in [46]; and (b) its segmentation into regions, namely cow, grass, trees, and sky, represented by four grey scale intensity values. Each pixel in the image (a) can take any one of the four labels, which results in over $10^{180,000}$ possible labellings. An energy value is associated with each labelling, and the segmentation in (b) is obtained by finding the labelling corresponding to the lowest energy.

be minimized by solving a minimum cost st-cut (st-mincut) problem [37,50,88,89]. In the first part of this thesis, we extend the class of energy functions which can be solved efficiently. We present novel techniques that improve the computational and memory efficiency of algorithms for solving multi-label energy functions. Our methods are motivated by the observations that the performance of minimization algorithms depends on: (i) the initialization used for the variables; and (ii) the number of variables in the energy function. We reuse results from previous problem instances to initialize the variables in the new instance, and also compute partially optimal solutions to reduce the number of unlabelled variables.

There are two other issues that need to be addressed when formulating vision labelling tasks in an energy minimization framework: (i) How to model the problem; and (ii) How to set the parameter values in the energy function. The second part of the thesis explores the possibility of including natural image statistics, which have been shown to be effective for many tasks [104, 114], into the energy function. We also show how the global minima of such energy functions can be obtained.

The last few years have seen a lot of attention being devoted to the problem of learning parameters of energy functions [62,71,84,90,100,117]. These methods learn the parameters using training data images and their corresponding labels[1], rather than make the user set them manually. However, the state-of-the-art parameter learning methods suffer from various drawbacks. They can lead to poor accuracy due to noisy estimates, as noted in [84, 97], or require performing inference for every training image repeatedly, which limits their applicability. In the third part of the thesis, we present an efficient piecewise method to overcome these drawbacks. Our method decomposes the original problem into a number of smaller problems, and then performs efficient discriminative learning.

1.2 Contributions

The main contributions of this thesis are summarized below. We will discuss the relevant contributions in detail at the end of every chapter, and present a consolidated summary in section 6.1.

Efficient Inference. As shown in section 1.1, energy functions defined for computer vision problems contain an extremely large number of variables. Searching for optimal solutions in such a large space requires efficient inference algorithms. We present three efficient techniques to improve the running time of inference methods. They are readily applicable for most of the popular energy minimization algorithms in computer vision. Methods optimized using our techniques provide the same solution as the standard methods, although in a much shorter time. Furthermore, all the optimality guarantees of the original methods are retained. One of our techniques can be considered as an extension of the work in [39, 46] for the multi-label (*i.e.* more than two labels) case.

Applications. We demonstrate the benefits of our methods on various labelling problems such as, colour based segmentation, stereo matching, object class category segmentation, single view reconstruction, structure detection. Our results in all these problems are significantly better than those reported previously in the literature. Examples of the labelling problems we consider are shown in Fig-

[1]Labels are obtained from either manual or automatic annotation of images. For example, PASCAL VOC dataset [19] provides high-quality manually annotated training data.

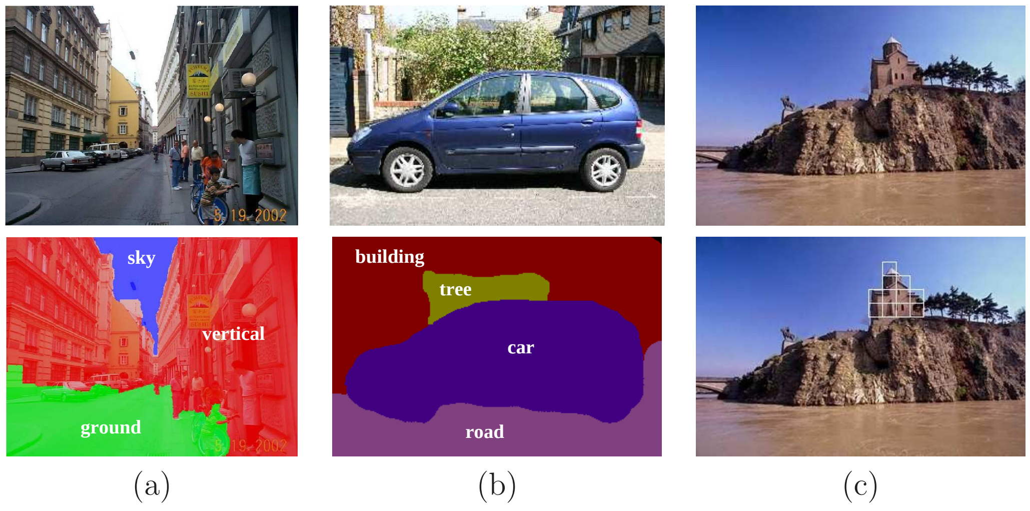


Figure 1.2: We show the benefits of our methods on various image labelling problems: (a) Single view reconstruction, (b) Object class category segmentation, (c) Structure detection. The first row shows an example image, and the second row shows the expected result, which corresponds to the minimum energy labelling. In (a), the task is to assign one of the three geometric labels, namely ground, vertical, sky, to every pixel in the image. Here we show an image from the automatic photo pop-up dataset [35]. In (b), we would like to recognize which object each pixel in the image belongs to. One of the images from the MSRC dataset [95] containing four object classes, building, car, road, tree, is shown here. In (c), the task is to find man-made structures (such as houses, cathedrals, buildings, castles) in the image. An image from the man-made structure database [62] along with the result (illustrated with white squares overlaid on the image) is shown.

ures 1.1 and 1.2. We have also made implementation of our methods publicly available.[2] In fact, most of the researchers using α-expansion, tree-reweighted message passing, and belief propagation algorithms, can replace the standard implementations with our optimized versions easily.

Using Natural Image Statistics. It is well-known that natural image statistics can be used to improve the results of many labelling problems [65, 104, 114]. We explore the use of these rich statistics for the problem of reconstructing a scene from a single 2D image.[3] We encode these learnt statistics as terms in the energy function that depend on more than two variables (referred to as higher

[2]See `http://cms.brookes.ac.uk/research/visiongroup`

[3]Note that this reconstruction problem is different from the traditional one where most pixels in the scene are assigned a 3D location. Here, the scene is approximated using three planes, which correspond to ground, vertical, and sky [36].

order terms). Unlike the work of [43], we present a method to obtain an *exact* solution for multi-label energy functions involving higher order terms.

Efficient Learning. We present a widely applicable method for learning parameters of the energy function. Unlike the previous methods, it is not limited by the efficiency of the inference step in every iteration of the learning algorithm. Our approach can also be viewed as extending max-margin based learning methods [100, 102] to a larger class of energy functions. Furthermore, our method is very easy to implement, and is suitable for multi-label energy functions.

1.3 Outline of the Thesis

In Chapter 2 we review the concepts of discrete optimization in the context of computer vision problems. We explain how vision problems can be formulated using probabilistic models such as Markov and conditional random fields. We then show that finding optimal solutions of such a model is equivalent to minimizing an energy function. We also provide details of popular (exact and approximate) energy minimization algorithms, explain under what conditions they can be applied, and discuss their limitations. Finally, we provide examples of energy functions for various image labelling problems, such as segmentation, stereo matching, single view reconstruction.

Chapter 3 introduces our methods for efficiently solving multi-label energy functions. Inspired by the dynamic computation paradigm, our first method improves the performance of the α-expansion algorithm [18]. We reuse results from previous problem instances to initialize the variables in a new (related) instance. This makes solving the new problem instance much more computationally efficient. Our second method simplifies the energy function by solving the *easy* part of the problem efficiently. Our strategy of reusing computations is then used to solve the remainder of the problem. We first present our methods for functions with energy terms containing one or two variables, and then show extensions to higher order terms. Many applications of these methods are also shown in this chapter.

In chapter 4 we address the problem of finding the exact solution of multi-label energy functions with higher order terms. We present a framework to transform a certain class of multi-label higher order functions to second order boolean functions, which can be minimized exactly using graph cuts. We show a principled way of including the rich statistics of natural images into the energy minimization framework in the form of higher order terms. In the latter part of this chapter we use these higher order terms to improve the quality of reconstruction from a single view of a scene.

Chapter 5 describes our method for learning parameters of energy functions. We begin by discussing the pros and cons of two popular paradigms, namely (approximate) maximum likelihood [62, 84] and max-margin [71, 102], for estimating the energy function parameters. We then describe our large margin piecewise learning method, which incorporates the benefits of both the paradigms. Finally, we show results on binary and multi-label energy functions to demonstrate that our model is widely applicable.

In chapter 6 we give a summary of the work presented in this thesis, and highlight our contributions. We also discuss promising avenues for future research.

Appendix A shows images from Middlebury-2005 [84] and man-made structure [62] datasets used in this thesis.

1.4 Publications

The first version of the work presented in chapter 3 for pairwise energy functions was published in CVPR 2008 [2]. An extension of this work for higher order functions, also presented in chapter 3, appeared in Transactions on PAMI [3]. The material presented in chapter 4 was published in CVPR 2008 [77]. The work presented in chapter 5 appeared in CVPR 2010 [4].

Chapter 2

Random Fields

Consider a set of random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$, and a set of labels $\mathcal{L} = \{l_1, l_2, \ldots, l_k\}$. The objective of a labelling problem defined over these random variables is to assign a label from the set $\mathcal{L}$ to each variable. Many computer vision tasks, such as image segmentation [16], stereo matching [86], object recognition [43, 94, 95], can be viewed as labelling problems. Typically, in such scenarios, the random variables correspond to pixels in an image, and the label set is defined according to the problem. For example, in the stereo matching problem, the labels represent disparity values, as shown in Figure 2.1. In the object recognition problem, each label denotes an object, as shown in Figure 2.2. In the latter part of this chapter we will discuss the formulation of these applications as labelling problems.

Random fields provide an elegant probabilistic framework to model labelling problems [29, 40, 70]. They provide a neighbourhood relationship between variables, and incorporate not only (noisy) image measurements, but also a prior model over the labelling space in a principled manner. Let $\mathcal{N}$ represent the neighbourhood of the random field, which is defined by sets $\mathcal{N}_i, \forall i \in \{1, 2, \ldots, n\}$. The set $\mathcal{N}_i$ denotes the set of all neighbours of the random variable X_i. In other words, $\mathcal{N}_i$ is the set of integers representing the indices of the neighbours of the random variable X_i. Random fields are also able to model the complex interactions between variables. Furthermore, it is possible to estimate the uncertainty in the labelling because the model is probabilistic. In this thesis, we are interested in two types of random field models, namely: (i) Markov random field; and (ii) conditional random field.

2.1 Markov Random Fields

A Markov random field (MRF) models the joint probability of the labelling $\mathbf{x}$ and the data $\mathbf{y}$, denoted by $\Pr(\mathbf{x}, \mathbf{y})$. According to the Bayes' rule, the joint probability is equal to the product of likelihood and prior probabilities as follows:

$$\Pr(\mathbf{x}, \mathbf{y}) = \Pr(\mathbf{y}|\mathbf{x}) \Pr(\mathbf{x}), \tag{2.1.1}$$

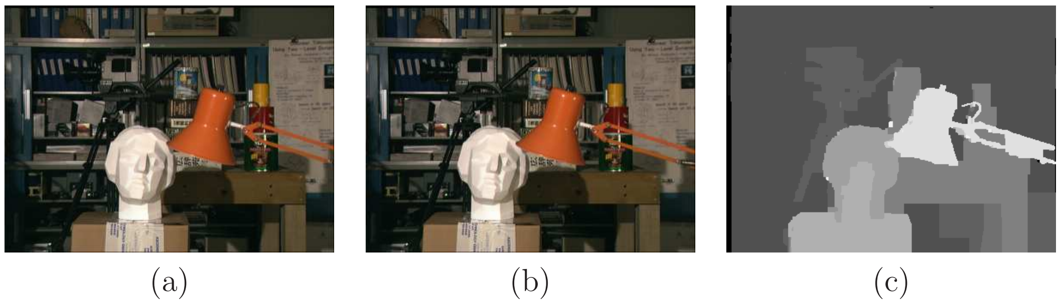


Figure 2.1: In the stereo matching problem, the task is to assign a disparity label to every pixel, given a pair of images. In the Tsukuba image pair [72] shown here, disparity gives the correspondence relationship between pixels in left and right images along every horizontal scan-line. (a) Image from left camera, (b) Image from right camera, and (c) The disparity map of the left camera image, are shown here. The lighter intensities in the disparity map (c) denote larger disparity values.

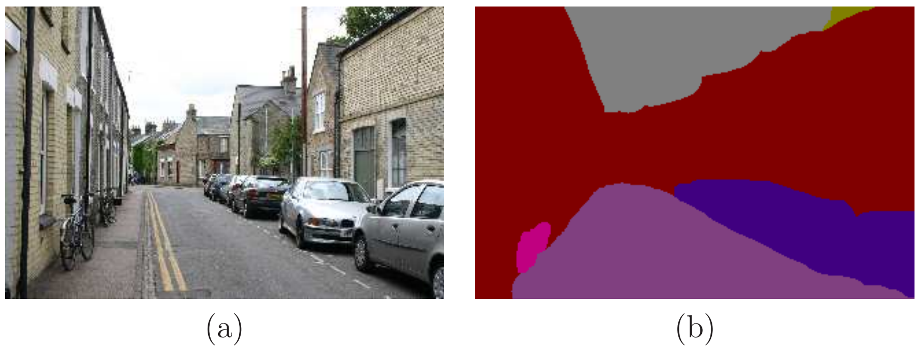


Figure 2.2: In the object recognition problem, the labels represent object classes, such as sky, road, car. (a) An image from the MSRC dataset [95], and (b) The corresponding object labelling are shown here. For instance, the region marked in red denotes 'building', and the grey region is 'sky'.

where $\Pr(\mathbf{y}|\mathbf{x})$ is the likelihood and $\Pr(\mathbf{x})$ is the prior. A random field that models the joint distribution (2.1.1) is said to be *Markovian* if satisfies the following properties [40, 70]:

$$\Pr(x_i|\{x_j : j \in \{1, 2, \dots, n\} - \{i\}\}) = \Pr(x_i|\{x_j : j \in \mathcal{N}_i\}), \forall i, \quad (2.1.2)$$

$$\Pr(\mathbf{x}) > 0, \forall \mathbf{x} \in \mathcal{L}^n. \quad (2.1.3)$$

The property (2.1.2) implies that the prior probability of the assignment $X_i = x_i$ depends only on the labelling of its neighbouring random variables given by $\mathcal{N}_i$. Figure 2.3 shows example of a Markov random field with a neighbourhood system

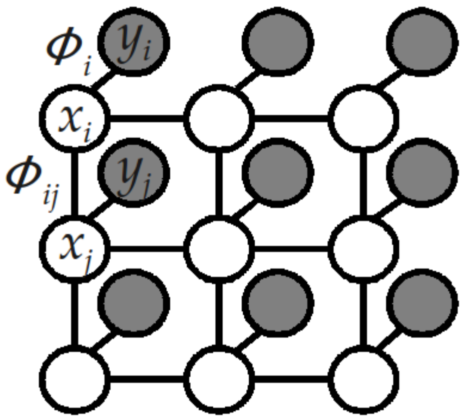


Figure 2.3: The graphical model representation of an MRF [11] consists of two kinds of nodes and undirected edges between them. The observed nodes Y_i represent the data, and are denoted by filled circles, while the hidden nodes X_i represent the random variables, and are denoted by unfilled circles. The edges between observed and hidden nodes represent the unary potentials. The edges connecting the hidden nodes represent the neighbourhood system in the random field. In this example, a hidden node is connected only to its immediate neighbour, thus representing a clique of size two. Image courtesy of M. Pawan Kumar [56].

of size two.

The joint distribution of an MRF in (2.1.1) can be written as follows:[1]

$$\Pr(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \exp(-\phi_c(\mathbf{x}_c)), \tag{2.1.4}$$

where $\mathcal{C}$ is the set of cliques formed by the neighbourhood system $\mathcal{N}$. For example, the MRF shown in Fig. 2.3 contains cliques of size two involving every pair of variables connected to each other. The term $\phi_c(\mathbf{x}_c)$ is known as the potential function of the clique c, where $\mathbf{x}_c = \{x_i, i \in c\}$. The term Z is the normalization constant[2], which ensures that the probabilities sum to one. For a pairwise MRF, such as the one shown in Fig. 2.3, the probability (2.1.4) can be re-written as:

$$\Pr(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \prod_{i \in \mathcal{V}} \exp(-\phi_i(x_i)) \prod_{(i,j) \in \mathcal{E}} \exp(-\phi_{ij}(x_i, x_j)), \tag{2.1.5}$$

where $\mathcal{V} = \{1, 2, \ldots, n\}$, and $\mathcal{E}$ is the set of edges between all pairs of neighbouring

[1]According to the Hammersley-Clifford theorem [6, 33].
[2]We will discuss the role of the partition function later in Chapter 5.

variables. The terms $\phi_i(x_i)$ and $\phi_{ij}(x_i, x_j)$ are called as the unary and pairwise potentials respectively. The Gibbs energy[3] of a labelling $\mathbf{x}$ for this MRF is given by:

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \phi_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \phi_{ij}(x_i, x_j). \tag{2.1.6}$$

The unary potential $\phi_i(x_i)$ models the likelihood of the label assignment $X_i = x_i$, while the pairwise potential $\phi_{ij}(x_i, x_j)$ models the cost of the assignment $X_i = x_i$ and $X_j = x_j$. From Fig. 2.3, note that $\phi_i(x_i)$ represents the cost of the edge connecting the observed node Y_i and the hidden node X_i, and depends on the data. On the other hand, $\phi_{ij}(x_i, x_j)$ represents the cost of the edge connecting two hidden nodes X_i and X_j, and is independent of the data. A pairwise potential commonly used in computer vision problems takes the form of Potts model, which gives a low energy value when $x_i = x_j$, and penalizes with a high energy values otherwise.

2.2 Conditional Random Fields

In many computer vision problems it may be necessary to use observed data for computing the pairwise potentials. Consider the image segmentation problem as an example (see Fig. 1.1). Constraining neighbouring pixels in the random field to take the same label results in a smoothly varying solution, but is not always ideal. If two neighbouring pixels are very different in their colour intensity values (or any other features), then they should be allowed to take different labels. One way to achieve this is by including the difference between the intensity values of the two pixels in the pairwise potential, thus making it dependent on the data. This idea of using data in the pairwise potential has been around for a few years [16,81,95]. Based on the work by Lafferty *et al.* [64], Kumar and Hebert [62] formalized the resulting probabilistic distribution as a conditional random field (CRF) model in the context of computer vision problems.[4]

A CRF can also be viewed as an MRF globally conditioned on the data. It

[3]Energy function maps any labelling $\mathbf{x} \in \mathcal{L}^n$ to a real number $E(\mathbf{x})$.

[4]Kumar and Hebert [62] refer to their model as discriminative random field. It is essentially a conditional random field model that uses a different scheme to learn the parameters of the energy function.

models the conditional probability of the labelling $\mathbf{x}$ given the data $\mathbf{y}$, assuming it satisfies the *Markovian* property, *i.e.*

$$\Pr(x_i|\{x_j : j \in \{1, 2, \dots, n\} - \{i\}\}, \mathbf{y}) = \Pr(x_i|\{x_j : j \in \mathcal{N}_i\}, \mathbf{y}), \forall i. \tag{2.2.1}$$

The conditional distribution of a pairwise random field is given by:

$$\Pr(\mathbf{x}|\mathbf{y}) = \frac{1}{Z} \prod_{i \in \mathcal{V}} \exp(-\phi_i(x_i)) \prod_{(i,j) \in \mathcal{E}} \exp(-\phi_{ij}(x_i, x_j)), \tag{2.2.2}$$

where Z is the normalization constant, and $\phi_i(x_i)$ and $\phi_{ij}(x_i, x_j)$ are the unary and pairwise potentials respectively, which both depend on data. This distribution can also be written as an energy function (similar to (2.1.6) in the MRF case).

In summary, Markov and conditional random field models provide a posterior probability distribution[5] of the labelling $\mathbf{x}$, given data $\mathbf{y}$. The best labelling of a given random field is obtained by maximizing the posterior probability. This is referred to as the problem of maximum a posteriori (MAP) estimation. The maximization problem is equivalent to minimizing the corresponding Gibbs energy as follows:

$$\mathbf{x_{map}} = \arg \min_{\mathbf{x} \in \mathbf{L}} E(\mathbf{x}). \tag{2.2.3}$$

Before we discuss algorithms for finding MAP solution, we will review a couple of relevant definitions.

Energy Reparameterization. Energy functions E_1 and E_2 are called *reparameterizations* of each other if and only if $\forall \mathbf{x}, E_1(\mathbf{x}) = E_2(\mathbf{x})$ [14, 47]. Note that this simply means that all possible labellings $\mathbf{x}$ have the same energy under both functions E_1 and E_2, and does not imply that E_1 and E_2 are composed of the same potential functions.

Energy Projection. A *projection* of any function $f(\cdot)$ is a function $f^p(\cdot)$ obtained by fixing the values of some of the arguments of $f(\cdot)$. For instance, fixing the value of the first t variables of the energy function $E(x_1, x_2, \dots, x_n) : \mathcal{L}^n \to \mathbb{R}$

[5]Note that the posterior probability distribution in the case of an MRF is proportional to the joint distribution.

produces the projection $E^p(x_{t+1}, x_{t+2}, \dots, x_n) : \mathcal{L}^{n-t} \to \mathbb{R}$. In other words, we obtain a new energy function $E^p(\cdot)$ of $n-t$ variables, by fixing t of the variables in the original energy function $E(\cdot)$ of n variables.

2.3 Maximum A Posteriori Estimation

The most probable or Maximum a Posteriori (MAP) solution can be found by minimizing the corresponding Gibbs energy, as shown in (2.2.3). The problem of minimizing this energy is NP-hard in general. However, there exist a number of powerful algorithms which the compute the exact solution for a particular family of energy functions in polynomial time. Two such families of energy functions relevant to our work are: (i) *Submodular* energy functions; and (ii) Energy functions defined on tree structured MRF/CRF. Submodular energy function minimization for certain random fields has been shown to be equivalent to a graph cut (specifically st-MINCUT) problem, which has several efficient polynomial time algorithms [31, 50, 88]. Energy functions defined on tree structured random fields can be solved by a dynamic programming algorithm presented in [75]. In the remainder of this section, we will describe these algorithms and their extensions proposed in the literature.

2.3.1 Submodular Energy Functions

Submodular energy functions are an important family of functions which can be minimized in polynomial time. They are discrete analogues of convex functions, and arise in various branches of applied mathematics such as game theory, information theory, and queueing theory. Given an *ordering* over the label set $\mathcal{L}$, a function $f(\cdot)$ is submodular if all its projections on two variables satisfy the constraint:

$$f^p(a, b) + f^p(a+1, b+1) \le f^p(a, b+1) + f^p(a+1, b), \quad \forall a, b \in \mathcal{L}. \qquad (2.3.1)$$

When dealing with functions of binary random variables this constraint transforms to:

$$f^p(0,0) + f^p(1,1) \leq f^p(0,1) + f^p(1,0). \tag{2.3.2}$$

One of the first strongly polynomial time algorithms for this family was proposed independently by [38] and [91]. However, this algorithm suffers from a very high runtime complexity. Recent work by Orlin [73] has successfully reduced this complexity to $\mathrm{O}\left(n^6\right)$, where n is the number of random variables in the problem, but is still impractical for vision problems involving millions of random variables. Certain submodular functions can be efficiently minimized by solving the st-MINCUT problem [14, 32, 50, 88]. For example, submodular functions of order[6] at most three involving binary random variables can be minimized in this way [8,50]. Several methods have been proposed to extend the class of energy functions that can be posed as the st-MINCUT problem. Certain binary higher order functions[7] can be transformed into submodular functions of order two, and thus minimized efficiently [25]. Schlesinger and Flach [88] showed how to convert a multi-label submodular problem composed of unary and pairwise potentials into an st-mincut problem. Since many energy functions can be transformed to binary submodular functions of order 2, solving this class of energy functions efficiently is of great importance. We will now explain an efficient graph cut (st-MINCUT) algorithm for addressing this problem.

2.3.2 Graph Cuts

With the introduction of efficient algorithms to solve the st-MINCUT problem, graph cuts have become an indispensable tool in the computer vision community [16, 17, 99]. These algorithms have a low runtime complexity, and thus allow fast computation of the globally optimal solution of an important class of energy functions, namely submodular energy functions. As we will see in the latter sections, they can also be used to find approximate solutions of non-submodular energy functions, with strong local optimality guarantees [18, 42, 53, 57, 109].

[6]The order of an energy function is k, if it can be written as a sum of potential functions, each of which is defined on at most k random variables. For example, the order of (2.1.6) is 2.

[7]Potential functions composed of three or more variables.

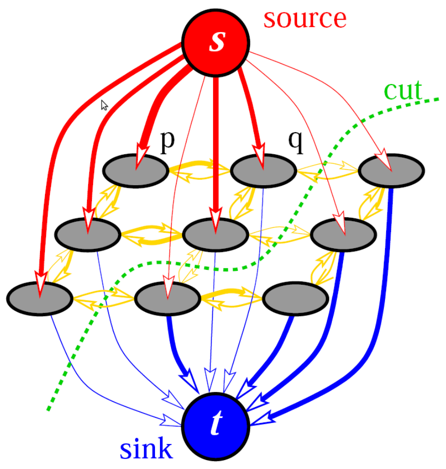


Figure 2.4: The st-MINCUT problem is defined using a directed graph with positive edge weights, such as the one shown here. It has two special nodes, source s and sink t, such that there are no edges into s and out of t. The set of nodes in $\mathcal{V}_g$ is represented as grey circles, and the n-edges (node-node) between them are shown in yellow. The t-edges (node-terminal) are shown in red or blue. The edge weights are indicated by the thickness of the edges. An st-cut (shown in green) separates the node set $\mathcal{V}_g$ into two disjoint sets – one containing the source and the other containing the sink. Image courtesy of Yuri Boykov [15].

2.3.2.1 The st-MINCUT Problem

The st-MINCUT problem is defined using a positively weighted directed graph $\mathcal{G}(\mathcal{V}_g \cup \{s,t\}, \mathcal{E}_g, C)$. Here, $\mathcal{V}_g$ denotes the set of vertices (or nodes) and $\mathcal{E}_g$ denotes the set of directed edges in the graph. The function $C : \mathcal{E}_g \to \mathbb{R}^+$ specifies the edge weights, and maps every edge $(i,j) \in \mathcal{E}_g$, to a non-negative real number c_{ij}. Graphs used in the st-MINCUT problem have two special vertices called source s and sink t, such that there are no incoming edges to the source, and no outgoing edges from the sink. These special nodes are collectively referred to as terminals. The edge set contains terminal edges (t-edges) and node edges (n-edges). The terminal edges connect the terminal nodes to every node $i \in \mathcal{V}_g$, and the node edges connect a pair of nodes $i, j \in \mathcal{V}_g$ according to some neighbourhood structure. Let us consider the binary image segmentation problem as an example. The nodes in the st-graph correspond to pixels in the image, and the terminals represent the two labels, say 0 and 1.[8] The edge weights are set according to the energy function defined for the segmentation problem, as discussed in the latter

[8]We follow the convention of s representing label 0, and t representing label 1.

part of this chapter.

Figure 2.4 shows an example of an st-graph. Given such a graph $\mathcal{G}$, an st-cut is defined as a partition of the node set $\mathcal{V}_g$ into two disjoint sets $\mathcal{V}_g^0$ and $\mathcal{V}_g^1$, such that $\mathcal{V}_g = \mathcal{V}_g^0 \cup \mathcal{V}_g^1$ (collectively exhaustive), $\mathcal{V}_g^0 \cap \mathcal{V}_g^1 = \emptyset$ (mutually exclusive), $s \in \mathcal{V}_g^0$, and $t \in \mathcal{V}_g^1$. All the nodes in the set $\mathcal{V}_g^0$ are assigned label corresponding to the source, and those in the set $\mathcal{V}_g^1$ are assigned the sink label. The cost of the st-cut $C_{\mathcal{V}_g^0,\mathcal{V}_g^1}$ is given by:

$$C_{\mathcal{V}_g^0,\mathcal{V}_g^1} = \sum_{i \in \mathcal{V}_g^0, j \in \mathcal{V}_g^1} c_{ij}. \tag{2.3.3}$$

The cost of an st-cut is equal to the cost of its associated labelling $\mathbf{x}$, *i.e.* $E(\mathbf{x})$. Now, the st-MINCUT problem is to find the st-cut with the minimum cost. The partitioning corresponding to the st-MINCUT provides the minimum cost labelling for the nodes in $\mathcal{V}_g$. According to the Ford-Fulkerson theorem [23], the st-MINCUT problem is equivalent to finding the maximum flow from the source to the sink with the weights C as edge capacities.

2.3.2.2 The Max-Flow Problem

Given a graph $\mathcal{G}(\mathcal{V}_g \cup \{s,t\}, \mathcal{E}_g, C)$, the max-flow problem is to find the maximum flow f from the source to the sink, such that the following edge capacity (2.3.4) and mass balance (2.3.5) constraints are satisfied:[9]

$$0 \leq f_{ij} \leq c_{ij}, \quad \forall (i,j) \in \mathcal{E}_g, \tag{2.3.4}$$

$$\sum_{j \in \mathcal{N}_i} f_{ij} - f_{ji} = 0, \quad \forall i \in \mathcal{V}_g, \tag{2.3.5}$$

where f_{ij} is the flow along the edge from node i to node j, and $\mathcal{N}_i$ is the set of nodes in the neighbourhood system of node i. The residual capacity r_{ij} of an edge (i,j), given a flow f_{ij}, is the maximum additional flow that can be passed from node i to node j using the edges (i,j) and (j,i), *i.e.* $r_{ij} = c_{ij} - f_{ij} + f_{ji}$. Now, a *residual graph* $\mathcal{G}(f)$, with respect to a flow f, consists of the nodes $\mathcal{V}_g$, and the edges with positive residual capacities. An *augmenting path* is defined as a path from the source to the sink along unsaturated edges, *i.e.* edges with

[9]Using the notation of [1, 41].

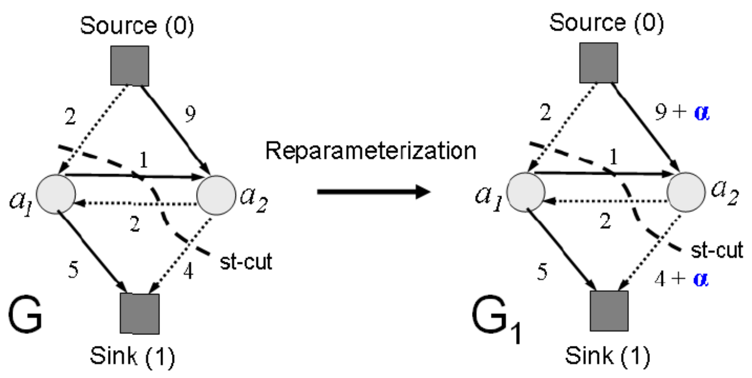


Figure 2.5: Here we show a graph G containing two nodes a_1 and a_2. The edge weights are given by the numbers beside them. The edges shown using dotted lines are part of the st-MINCUT, and the cost of this cut is $2 + 2 + 4 = 8$. We reparameterize this graph by adding a positive constant α to the t-edges of node a_2 and obtain the graph G_1. Performing st-MINCUT/max-flow on this reparameterized graph results in an identical st-MINCUT. Thus, both the graphs induce the same partitioning in the node set, although the cost of the st-MINCUT is different in the two graphs. Image courtesy of Pushmeet Kohli [46].

positive residual capacities, of the residual graph.

Max-flow algorithms typically find an augmenting path, send the maximum possible flow through it, and repeat this process until no such paths can be found [17]. The sum of all the flows obtained at each step is the maximum flow for the graph. At the end of the process, certain edges will be saturated, and the graph will be partitioned into two sets, separating the source and the sink. In other words, it produces an st-cut. It has been shown that the maximum flow value thus obtained is equal to the cost of the st-MINCUT for the graph [23]. Other max-flow algorithms, such as push-relabel algorithm [30], also provide efficient ways for achieving this, and are described in the excellent book by Ahuja *et al.* [1]. In summary, after the max-flow algorithm has terminated, the set $\mathcal{V}_g$ is partitioned into two sets: $\mathcal{V}_g^0$ (source set) and $\mathcal{V}_g^1$ (sink set), thus assigning labels to all the nodes.

Graph Reparameterization. There are certain transformations, which do not affect the labelling obtained by performing the max-flow operation. Such transformations only result in a *reparameterization* of the graph. For example, adding a constant value to the terminal edge weights c_{si} and c_{it} of any node i does not affect the labelling, as it only depends on the difference of the edge weights $(c_{si} - c_{it})$. We show this on an example taken from [46] in Figure 2.5. Another example of graph reparameterization is shown in Figure 3.2.

What can be solved? We mentioned earlier that all the edges in an st-graph must be non-negative. This naturally restricts the class of energy functions that can be represented and therefore solved using an st-graph. We now formalize this class of st-MINCUT solvable energy functions. For simplicity, let us consider the pairwise energy function in (2.1.6):

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \phi_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \phi_{ij}(x_i, x_j). \tag{2.3.6}$$

Furthermore, we assume that the random variables X_i are binary valued.[10] Following the pseudo-boolean notation in [14], we can re-write energy function (2.3.6) as:[11]

$$\begin{aligned} E(\mathbf{x}) = \quad & \sum_{i \in \mathcal{V}} \Big(\phi_i^1 x_i + \phi_i^0 \bar{x}_i \Big) \\ & + \sum_{(i,j) \in \mathcal{E}} \Big(\phi_{ij}^{00} \bar{x}_i \bar{x}_j + \phi_{ij}^{01} \bar{x}_i x_j + \phi_{ij}^{10} x_i \bar{x}_j + \phi_{ij}^{11} x_i x_j \Big), \end{aligned} \tag{2.3.7}$$

where $\bar{x}_i$ is the binary complement of x_i, *i.e.* $\bar{x}_i = 1, \text{if } x_i = 0$ and vice versa. We simplify this energy function for two binary variables x_i, x_j, and the edge (i, j) between them as follows:

$$\begin{aligned} E^p(x_i, x_j) &= \phi_i^1 x_i + \phi_i^0 \bar{x}_i + \phi_j^1 x_j + \phi_j^0 \bar{x}_j + \phi_{ij}^{00} \bar{x}_i \bar{x}_j + \phi_{ij}^{01} \bar{x}_i x_j + \phi_{ij}^{10} x_i \bar{x}_j + \phi_{ij}^{11} x_i x_j \\ &= \phi^{\text{const}} + \Big(\phi_i^1 + \phi_{ij}^{11} - \phi_{ij}^{01} \Big) x_i + \phi_i^0 \bar{x}_i + \phi_j^1 x_j + \Big(\phi_j^0 + \phi_{ij}^{00} - \phi_{ij}^{01} \Big) \bar{x}_j \\ &\quad + \Big(\phi_{ij}^{01} + \phi_{ij}^{10} - \phi_{ij}^{00} - \phi_{ij}^{11} \Big) x_i \bar{x}_j, \end{aligned} \tag{2.3.8}$$

where ϕ^{const} is a constant. Note that the coefficients of the unary terms can be varied[12] such that they are non-negative. It can be easily verified that the coefficient of the pairwise term will always be equal to $\Big(\phi_{ij}^{01} + \phi_{ij}^{10} - \phi_{ij}^{00} - \phi_{ij}^{11} \Big)$. Given

[10]Note that these assumptions are not restrictive, as many multi-label higher order functions can be transformed to binary pairwise functions (§2.3.1).

[11]We denote $\phi_i(0)$ as ϕ_i^0 and $\phi_{ij}(1,0)$ ϕ_{ij}^{10} for brevity.

[12]For example, by rewriting the equation algebraically.

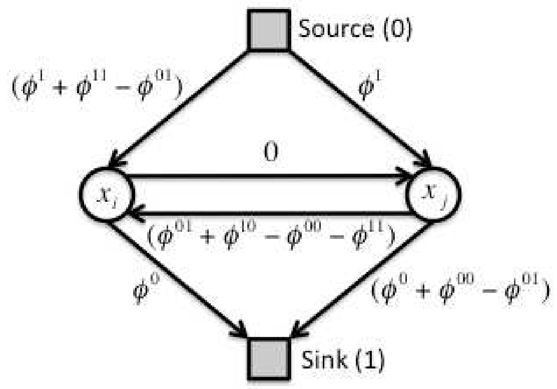

Source (0)
$(\phi^1 + \phi^{11} - \phi^{01})$
ϕ^1
0
x_i
x_j
$(\phi^{01} + \phi^{10} - \phi^{00} - \phi^{11})$
ϕ^0
$(\phi^0 + \phi^{00} - \phi^{01})$
Sink (1)

Figure 2.6: Here we construct an st-graph corresponding to the energy function (2.3.8). In our notation, we assign label 0 if a node belongs to the source set, and label 1 otherwise. Thus, the cost for node i taking label 0 (given by the coefficient of the unary term $\bar{x}_i$) is added to the t-edge (i, t). All the other t-edge costs are added in a similar fashion. The pairwise term $x_i\bar{x}_j$ represents the cost of the assignment $x_i = 1, x_j = 0$, and its coefficient is added to the n-edge (j, i). As there is no pairwise term for the assignment $x_i = 0, x_j = 1$, the n-edge (i, j) has no cost.

this form of the energy, we now construct the st-graph as shown in Figure 2.6. For this graph to be a valid st-graph, all the edge weights must be non-negative. The t-edge weights can be modified (either algebraically or by graph reparameterization), such that they are positive. For the n-edges to have non-negative weights, the condition $\left(\phi_{ij}^{01} + \phi_{ij}^{10} - \phi_{ij}^{00} - \phi_{ij}^{11}\right) \geq 0$ must be satisfied, which is the binary submodularity condition (2.3.2). This equivalence of binary pairwise submodular functions and st-MINCUT was shown by Hammer [32] and Kolmogorov and Zabih [50].

2.3.3 Solving Non-submodular Energy Functions

So far we have seen efficient algorithms for solving submodular energy functions. However, most multi-label energy functions encountered in computer vision do not satisfy the constraint (2.3.1), and thus are non-submodular. For instance, it can be clearly seen that the Potts model potential $\psi_{ij}(\cdot)$ defined as:

$$\psi_{ij}(x_i, x_j) = \begin{cases} 0 & \text{if } x_i = x_j, \\ \gamma & \text{otherwise,} \end{cases} \tag{2.3.9}$$

does not satisfy the constraint (2.3.1). Choosing $a = k$ and $b = k + 1$ in (2.3.1) we get:

$$f^p(k, k+1) + f^p(k+1, k+2) \leq f^p(k, k+2) + f^p(k+1, k+1). \qquad (2.3.10)$$

The LHS of equation (2.3.10) is equal to 2γ while the RHS is equal to γ, making the above condition false. A number of approximate or partially optimal algorithms have been proposed to solve this class of energy functions [14, 18, 47, 48, 54, 76, 83, 111]. Some of these methods provide an approximate solution either by optimizing a related submodular energy function [76, 83], or by solving a relaxation of the problem [47, 111]. The methods proposed in [14, 54] provide a globally optimal solution for only a subset of the problem. The remaining part of the problem is then solved with message passing algorithms [47, 75]. Boykov *et al.* [18] proposed efficient graph cut based α-expansion and $\alpha\beta$-swap algorithms for solving non-submodular problems. We will provide an overview of these two algorithms in the remainder of this section.

Move making algorithms. The α-expansion and $\alpha\beta$-swap algorithms are widely used for approximate energy minimization [18, 99]. They belong to the class of move making algorithms. These algorithms work by starting from an initial labelling $\mathbf{x}$ and making a series of moves (label changes) which lower the energy iteratively. Convergence is achieved when the energy cannot be decreased further. At each step, the algorithms search a move space to find the *optimal move* – one that decreases the energy of the labelling by the most amount. The move search space must be as large as possible in order to make the algorithm less likely to get stuck in local optima. Expansion and swap algorithms achieve this by using a search space that is exponentially large in the number of variables in the energy function. They perform this search efficiently for a certain class of energy functions by solving an st-MINCUT/max-flow problem.

The α-expansion algorithm is an iterative procedure, which finds an approximate MAP estimate by solving a series of st-MINCUT problems. At each step, it considers a label $\alpha \in \mathcal{L}$, and allows all the random variables to either retain their current label or change to α. This is done by solving an st-MINCUT problem, which makes the binary decision of changing or retaining the label assignment. One iteration of the algorithm involves performing expansions for all α in some order successively. The algorithm terminates when the energy cannot be reduced further for any α. Boykov *et al.* [18] showed that α-expansion is applicable if the pairwise potential functions ϕ_{ij} define a *metric*, *e.g.* Potts model (2.3.9), truncated linear model.

The $\alpha\beta$-swap algorithm also finds an approximate MAP estimate by solving a series of st-MINCUT problems. Unlike α-expansion, it considers a pair of labels $\alpha, \beta \in \mathcal{L}$ together with all the variables currently assigned α or β. It then solves an st-MINCUT problem, which can swap the label assignments of these variables. The algorithm terminates when the energy cannot be reduced further by swapping labels for any pairs of labels α, β. These moves can be computed if ϕ_{ij} defines a *semi-metric* [18], *e.g.* Potts model (2.3.9), truncated linear or truncated quadratic models. We will revisit these move making algorithms and provide more details in Chapter 3.

2.3.4 Message Passing Algorithms

Message passing algorithms are another important class of methods for addressing the MAP inference problem. These algorithms work by passing messages between nodes representing the random variables of the model. Belief Propagation (BP) is a popular and well-known message passing algorithm for MAP inference. It was originally proposed for a tree structured random field, where it is guaranteed to produce the exact MAP estimate in two iterations [75]. In the first iteration, messages are sent from the leaf nodes to the root, and in the second iteration, they are sent in the opposite direction.[13] For a general random field (with loops or cycles, *e.g.* MRF shown in Fig. 2.3), BP is not guaranteed to converge. Variants of BP have been proposed [20, 27, 112, 116] to handle such models. These algorithms have no optimality guarantees, but can provide a good estimate of the MAP solution empirically, as noted in [99]. BP messages can be computed using either a max-product [11] or a sum-product [75, 116] rule. In the former case, we take

[13]Similar to forward-backward passes in dynamic programming.

the maximum over all possible label values and obtain the MAP estimate directly. While, in the latter case, we take the sum of all possible label values and obtain a set of probability estimates, which can be used to get the MAP solution.

Wainwright *et al.* [111] proposed another belief propagation variant called tree-reweighted message passing (TRW), which was motivated by the problem of maximizing a concave lower bound on the energy. Their algorithm begins by selecting a set of trees from the random field, and computes probability distributions over each tree. These distributions are then used to reweight the messages being passed during loopy BP on each tree. The hope is that each step of loopy BP, followed by reweighting increases the lower bound on the energy. Kolmogorov [47] showed that the TRW algorithm is not guaranteed to achieve this, and proposed a sequential extension (TRW-S) to address this problem. TRW-S processed nodes in a scan-line order. Each node sent messages to its right and bottom neighbours in the forward pass, and its left and top neighbours in the backward pass. The algorithm terminates when the lower bound cannot be increased further.

To summarize, there are many algorithms to solve the MAP inference problem. Efficient graph cut based methods minimize submodular energy functions. Energy functions arising out of tree structured graphs can be solved exactly with message passing algorithms. All other classes of energy functions can be minimized approximately or partially.

2.4 Example Vision Problems

We now look at two low-level vision problems, and discuss how they can be modelled in the energy minimization framework.

2.4.1 Image Segmentation

Consider the interactive image segmentation problem shown in Fig. 2.7 [15, 16]. In this problem, the user marks red (foreground) and blue (background) strokes or regions, and the goal is to solve a binary MRF problem to estimate the foreground and background regions in the image. Note that our discussion here is

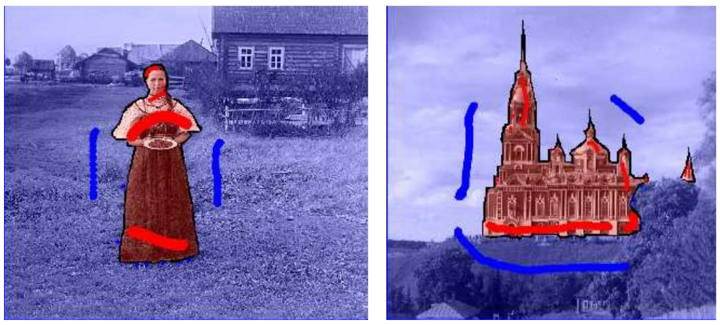


Figure 2.7: Here we show two examples of interactive binary image segmentation problem. The red and blue strokes indicate the foreground and background seed pixels respectively, which are marked by the user. These seed pixels are used to compute the RGB histogram distributions for the two regions. Note that the images are colour coded to show the expected foreground (red) and background (blue) regions. Image courtesy of Yuri Boykov [15].

focussed on the binary image segmentation problem. In Chapter 3, we will revisit this problem using multiple labels, *e.g.* as shown in Fig. 1.1. The energy corresponding to the binary segmentation problem is given by (2.1.6), where the set of vertices corresponds to pixels in the image, and the set of edges is given by the neighbourhood we choose. Here we use 4-neighbourhood as an example, *i.e.* every pixel i is connected to its 4 immediate neighbours – to the top, the right, the bottom, and the left of i. The unary potentials $\phi_i(x_i), i \in \mathcal{V}$, are defined using RGB histogram distributions $\mathcal{H}_a, a = \{0, 1\}$, of the two segment labels as follows:

$$\phi_i(x_i) = -\log p(x_i = a|\mathcal{H}_a). \tag{2.4.1}$$

The distributions $\mathcal{H}_a$ are computed using the user-specified seed pixels (available in the form of strokes or regions).

The pairwise potentials must ensure that we obtain a spatially continuous (*i.e.* smooth) segmentation, without speckles. This can be achieved using the Potts model (2.3.9), which assigns a cost γ if neighbouring pixels take different labels, and a cost 0 if they take the same label. This potential ignores image edges, and encourages pixels on either side of an edge to take the same label as well. Boykov and Jolly [16] introduced a data-dependent smoothness term to overcome this problem. Similar potentials were later used by many researchers [12, 81, 95, 101]. The edge-preserving smoothness term takes the form of a Generalized Potts model defined as:

$$\phi_{ij}(x_i, x_j) = \begin{cases} \lambda_1 + \lambda_2 \exp\left(\frac{-g^2(i,j)}{2\sigma^2}\right)\frac{1}{\text{dist}(i,j)} & \text{if } x_i \neq x_j, \\ 0 & \text{if } x_i = x_j, \end{cases} \tag{2.4.2}$$

where λ_1, λ_2 and σ are parameters of the model. The terms $g(i, j)$ and $\text{dist}(i, j)$ give the difference in RGB values and the spatial distance respectively between pixels i and j. It can be easily verified that this energy function satisfies the submodularity condition (2.3.2), and therefore be minimized using the st-MINCUT/max-flow algorithm. More sophisticated priors, such as connectivity priors [110], shape priors [44, 66, 67] can also be included in the energy function.

2.4.2 Stereo Matching

Stereo matching is the process of taking two or more images[14] and estimating a 3D model of the scene by finding matching pixels in the images and converting their 2D positions into 3D depths [98]. An example of the stereo matching problem is shown in Fig.2.1. The results of stereo matching algorithms are typically presented as a dense disparity map, where each pixel is assigned a disparity value, which indicates horizontal displacement the pixel has undergone from one image to another. It can easily seen that disparity is inversely proportional to distance from the observer, *i.e.* depth [24, 34, 98]. The stereo matching problem has been formulated as an optimization problem using an energy function similar to (2.1.6), where each pixel takes a disparity label [9, 16, 85, 96].

In the energy function we describe here, the set of vertices corresponds to pixels in the image, and the set of edges is given by 4-neighbourhood. The unary potential is a similarity measure that compares the pixel values in order to determine how likely they are to be in correspondence. This measure is computed by considering either the pixel or a region of support, *e.g.* 5×5 window, around it. A few examples of similarity measures are squared intensity difference, truncated quadratics, entropy, filter-bank responses. Interested readers are encouraged to see Chapter 11 in [98] for more details of similarity measures. The pairwise term

[14]For simplicity, we will focus on using two images in our discussion here.

is a Generalized Potts model (2.4.2), which encourages similar pixels to take the same label. This multi-label energy function can be minimized using the move making or message passing algorithms discussed in this chapter.

2.5 Summary

In this chapter we presented a review of discrete optimization concepts in the context of computer vision problems. We introduced two popular random field models, and showed that finding optimal solutions of these models is equivalent to minimizing the corresponding energy functions. We also provided details of relevant energy minimization algorithms.

Chapter 3

Efficient Energy Minimization

3.1 Introduction

Many problems in computer vision such as image segmentation, stereo matching, image restoration, and panoramic stitching involve inferring the maximum a posteriori (MAP) solution of a probability distribution defined by a discrete MRF or CRF [18, 49, 83, 99]. The MAP solution can be found by minimizing an energy or cost function. Although, minimizing a general MRF energy function is an NP-hard problem [18], there exist a number of powerful algorithms which compute the exact solution for a particular family of energy functions in polynomial time. For instance, max-product (min-sum) belief propagation exactly minimizes energy functions defined over graphs with no loops [115]. Similarly, certain submodular energy functions can be minimized by solving an st-mincut problem [17, 25, 37, 50].

Efficient approximation algorithms have also been proposed for functions which do not fall under the above classes [18, 47, 111]. Expansion and swap move making algorithms, sequential tree-reweighted message passing (TRW-S), and belief propagation (BP) are examples of popular methods for solving these functions. They have been shown to give excellent results on the discrete MRFs typically used in computer vision [18, 99]. However, these algorithms can take a considerable amount of time to solve problems which involve a large number of variables.

As computer vision moves towards the era of large videos and giga-pixel images, computational efficiency is becoming increasingly important. Indeed, the last few years have seen much attention being devoted to reducing the computational complexity of minimization algorithms [20, 39, 46, 53]. In this chapter we make two contributions to improve the efficiency of energy minimization algorithms. Our first contribution is a method which works by *recycling* results from previous problem instances, providing a simpler alternative to the recent work of [53] on dynamic energy minimization. Our second contribution is a method which simplifies the energy minimization problem by *reducing* the number of variables in the energy function, and can also be used to generate a good initialization for the dynamic α-expansion algorithm by *reusing* dual variables.

Recycling Solutions. Our first method is inspired by the dynamic computation paradigm [39, 46, 53]. It improves the performance of the α-expansion algorithm by recycling results from previous problem instances. The idea of dynamic computation has been used in the recent work of [39, 46] on minimizing submodular energy functions. In particular, [46] showed how flow can be reused in maxflow algorithms, and [39] showed how cuts (or previous labelling) can be reused. However, these methods are only applicable for the special case of dynamic MRFs[1] that are characterized by submodular energy functions. Our work extends these methods to non-submodular multi-label energy functions. It is most similar to the interesting Fast-PD algorithm proposed by Komodakis *et al.* [53], which generalizes the work of [46] and [52]. Fast-PD works by solving the energy minimization problem by a series of graph cut computations. This process is made efficient by reusing the primal and dual solutions of the linear programming (LP) relaxation of the energy minimization problem, achieving a substantial improvement in the running time. Our modified dynamic α-expansion algorithm is conceptually much simpler and easier to implement than Fast-PD whilst giving similar performance. Our method of initializing the α-expansion algorithm can make both methods orders of magnitude faster.

Simplifying energy functions. Most energy minimization problems encountered while solving computer vision problems are composed of "*easy*" and "*difficult*" components [48, 54]. For instance, the variables labelled by the QPBO algorithm [14, 48] constitute the easy component, while the rest constitute the difficult component. The globally optimal labels for variables constituting the easy component of the MRF energy function can be found in a few iterations of the minimization algorithm, while those of the difficult part typically cannot be found in polynomial time (in the number of variables). Energy minimization algorithms generally do not take advantage of this decomposition, and process all the random variables at every iteration.

We propose a novel strategy which solves a given discrete MRF in two phases.

[1]MRFs that vary over time [39, 46].

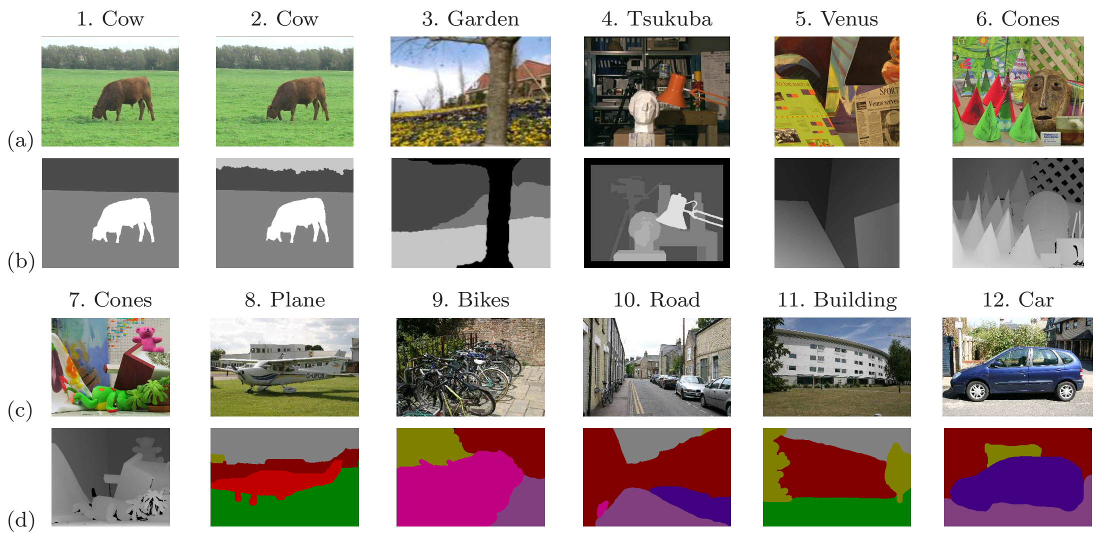


Figure 3.1: *Some of the images (a,c) and their ground truth labellings (b,d) used in our experiments. 1-3 Colour-based segmentation problems with 3, 4, 4 labels respectively. 4-7 Stereo matching problems with 16, 20, 60, 60 labels respectively. 8-12 Object-based segmentation problems with 4, 5, 5, 7, 8 labels respectively. (This figure is best viewed in colour.)*

In the first phase a partially optimal solution of the energy function is computed [14, 48, 54]. In such solutions, not all variables are assigned a label. However, the set of variables which are assigned a label, are guaranteed to take the same labelling in at least one of the optimal solutions of the energy function. This is referred to as the property of *partial optimality*. Using the partial solutions to fix values of these variables results in a *projection* (cf. section 2.2) of the original energy function [50]. In the second phase we minimize this simplified energy which depends on fewer variables, and is consequentially easier and faster to minimize compared to the original energy function. This approach is applicable to many popular energy minimization approaches such as α-expansion, BP, Fast-PD and TRW-S. We also show how to achieve a substantial speed-up in the minimization of the simplified energy by reusing results from computations performed to find the partially optimal solution.

3.1.1 Outline of the Chapter

In section 3.2, we briefly review the notation and the algorithms for minimizing multi-label energy functions [14, 18, 47, 54]. Section 3.3 presents our two methods to improve the running time of such algorithms. Specifically, it describes methods to: (a) recycle the primal and dual solutions for obtaining a good initialization for the new problem instance; and (b) reduce energy functions and reuse the resulting residual graphs. Our methods are also applicable for certain higher order energy functions, such as those containing the $\mathcal{P}^n$ model potentials proposed by Kohli *et al.* [43]. We discuss this extension in section 3.4 using the problem of interactive texture based image and video segmentation as an example. We also prove that partially optimal solutions can be computed for this model. In section 3.5, we evaluate the performance of our methods on the problems of colour and object based segmentation [16, 94, 95], and stereo matching [99]. A few examples of these problems are shown in Fig. 3.1. Summary and discussion are provided in section 3.6.

3.2 Preliminaries

We denote each pixel i in the image with a random variable X_i, which takes a value from the label set $\mathcal{L} = \{l_1, l_2, \ldots, l_k\}$. A labelling $\mathbf{x}$ refers to any possible assignment of labels to the random variables and takes values from the set $\mathcal{L}^n$, where n is the number of pixels. For example, the label set corresponds to disparities in the case of stereo matching problem, and image segments in the case of colour-based segmentation problem. Fig. 3.1 shows a few of the segmentation and stereo matching problems we consider in this work.

Given a neighbourhood system $\mathcal{N}$, a clique c is specified by a set of random variables $\mathbf{X}_c$ such that $\forall i, j \in c, i \in \mathcal{N}_j$ and $j \in \mathcal{N}_i$, where $\mathcal{N}_i$ and $\mathcal{N}_j$ are the sets of all neighbours of variable X_i and X_j respectively. An energy function $E : \mathcal{L}^n \to \mathbb{R}$, which maps any labelling to a real number $E(\mathbf{x})$, can be written as:

$$E(\mathbf{x}) = \sum_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c), \tag{3.2.1}$$

where $\mathcal{C}$ is the set of all cliques. The term $\phi_c(\mathbf{x}_c)$ is known as the potential function of the clique c, where $\mathbf{x}_c = \{x_i, i \in c\}$. Note that this is a generalization of the unary and pairwise potential functions typically used in computer vision.

The unary potential $\phi_i(x_i)$ represents the cost of the assignment: $X_i = x_i$, and is defined by considering cliques of size 1 (*i.e.* treating each pixel as a clique). The pairwise potential $\phi_{ij}(x_i, x_j)$ represents the cost of the assignment: $X_i = x_i$ and $X_j = x_j$, and is obtained by considering cliques of size 2. We will initially explain our methods using pairwise energy functions of the form:

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \phi_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \phi_{ij}(x_i, x_j), \tag{3.2.2}$$

where $\mathcal{V}$ is the set of all random variables and $\mathcal{E}$ is the set of all pairs of interacting variables. In section 3.4 we will provide details of the proposed methods for higher order functions.

The unary potential ϕ_i can be obtained in many ways. For example, in a colour-based image segmentation problem it is common to use the RGB distribution for computing the potential. In a stereo matching problem the unary potentials are typically obtained using a window-based correlation measure. Object-based segmentation problems can learn the potential using a boosting procedure [103]. The exact form of all these potentials will be explained in section 3.5. The pairwise potential ϕ_{ij} commonly takes the form of the Potts model (or its contrast-sensitive variant [16]), and is given by:

$$\phi_{ij}(x_i, x_j) = \begin{cases} 0 & \text{if } x_i = x_j, \\ \gamma & \text{otherwise.} \end{cases} \tag{3.2.3}$$

The contrast-sensitive variant modulates the cost γ of two neighbouring nodes taking different labels with the difference in feature values and spatial distance between the nodes. This is also referred to as an edge-preserving pairwise potential, as two nodes lying on either side of an edge are likely to have different feature values and thus can take different labels.

This fairly simple but effective energy function in equation (3.2.2) cannot be solved exactly. Recall (§2.3.1) that multi-label energy functions can be solved exactly iff they satisfy the submodularity condition given by:

$$E^p(a, b) + E^p(a+1, b+1) \leq E^p(a, b+1) + E^p(a+1, b), \tag{3.2.4}$$

for all $a, b \in \mathcal{L}$ and for all its projections on two variables. Here $E^p(\cdot)$ is a projection of the original energy function $E(\cdot)$. Choosing $a = k$ and $b = k + 1$ in (3.2.4) we get:

$$E^p(k, k+1) + E^p(k+1, k+2) \leq E^p(k, k+2) + E^p(k+1, k+1). \tag{3.2.5}$$

In the case of the Potts model (3.2.3), the LHS of equation (3.2.5) is equal to 2γ while the RHS is equal to γ making the above condition false. Thus, Potts model is not submodular for multi-label energy functions, and hence cannot be solved exactly [18, 50]. Many algorithms have been proposed to find approximate or partially optimal solutions of these energy functions [14, 18, 48, 54, 111]. We provide a brief summary of some of these algorithms, which are relevant to our work, in the next section.

3.2.1 Approximate Energy Minimization

Approximate algorithms for solving multi-label energy functions can be broadly classified into move-making and message passing algorithms.

Move making algorithms. The α-expansion and $\alpha\beta$-swap algorithms are widely used for approximate energy minimization [18, 99]. These algorithms work by starting from an initial labelling $\mathbf{x}$ and making a series of label changes (moves), which lower the energy at each step. An optimal move, which is the move decreasing the energy of the labelling by the most amount, is found efficiently at every step from the large[2] move space. Convergence is achieved when the energy cannot be decreased further.

The α-expansion move allows any random variable to either retain its current label or take a label α. One iteration of the algorithm involves performing expansion moves for all $\alpha \in \mathcal{L}$ in some order successively. The iterations are repeated until the energy cannot be decreased any further. Boykov *et al.* [18] showed that the optimal expansion moves for certain energy functions of the form (3.2.2) can be computed in polynomial time by solving an st-mincut problem. They showed

[2]Exponential in the number of variables in the energy function.

that if the pairwise potentials ϕ_{ij} define a *metric*, then the energy function (3.2.2) can be minimized using α-expansion. In other words, ϕ_{ij} should satisfy the following conditions:

$$\begin{aligned} \phi_{ij}(a,b) &= 0 \iff a = b, && (3.2.6)\\ \phi_{ij}(a,b) &= \phi_{ij}(b,a) \geq 0, && (3.2.7)\\ \phi_{ij}(a,c) &\leq \phi_{ij}(a,b) + \phi_{ij}(b,c), && (3.2.8)\end{aligned}$$

for all $a, b, c \in \mathcal{L}$.

The $\alpha\beta$-swap move allows any random variable whose current label is α or β to either take a label α or β. One iteration of the algorithm involves performing swap moves for all pairs of labels $\alpha, \beta \in \mathcal{L}$ in some order successively. These iterations are repeated until convergence. Optimal swap moves for energy functions of the form (3.2.2) can be computed in polynomial time if ϕ_{ij} defines a *semi-metric*, *i.e.* satisfies conditions (3.2.6) and (3.2.7) [18].

Message passing algorithms. The other class of algorithms for approximate energy minimization work by passing messages between nodes representing the different random variables of the model. Max-product belief propagation (BP) is one such method for MAP inference proposed by Pearl [75]. A message from node X_i to X_j indicates how likely it is for X_j to take a certain label from X_i's perspective. The BP algorithm was originally designed for tree structured graphs where it is guaranteed to provide the exact MAP solution within two iterations [75]. In the first iteration the messages are sent from the leaf nodes of the tree towards the root. Messages are then sent from the root towards the leaf nodes in the second iteration. After these iterations, the belief of taking a label $l_p, \forall p \in 1, \ldots, k$, is computed for every node X_i using the unary potential $\phi_i(x_i = l_p)$ and the messages from all its neighbours corresponding to the label l_p. The node is then assigned a label according to its maximum belief. This method is not guaranteed to converge for the grid (loopy) graphs we use in computer vision. However, it has been applied to loopy graphs with some success [20, 26, 27, 99]. In this case, the iterations are repeated until the rate of change of messages from one iteration to the next falls below a certain threshold, thus resulting in an approximate solution.

Wainwright *et al.* [111] proposed the tree-reweighted message passing (TRW) algorithm, which decomposes the graph into a set of trees and performs BP on them. The messages being passed are reweighted with sets of probability distributions over each tree. The TRW algorithm also computes the lower bound on the energy, and aims to increase this bound in successive iterations. Kolmogorov [47] developed an improved sequential version of TRW, referred to as TRW-S, by processing the nodes in a scan-line order. TRW-S has two useful properties: (a) The lower bound estimate is guaranteed not to decrease in every iteration; and (b) The lower bound estimate is guaranteed to converge, unlike the original TRW algorithm. Other variants of message passing algorithms have also been proposed [51, 89, 113].

3.2.2 Computing Partially Optimal Solutions

Certain algorithms for minimizing non-submodular functions (such as (3.2.3)) return a partial solution $\mathbf{x} \in (\mathcal{L} \cup \{\epsilon\})^n$ of the energy [14, 45, 48, 54, 82]. Here, the assignment $x_i = \epsilon$ implies that no label has been given to random variable X_i. In other words, these algorithms assign labels to a subset of the random variables. Consider the QPBO algorithm [14, 48] as an example. It minimizes energy functions composed of binary random variables, and returns a partially labelled solution $\mathbf{x}$ with the following property: there exists a global minimum $\mathbf{x}^*$ of the energy function such that $x_p = x_p^*$ for all variables X_p that are labelled, *i.e.* $x_p \neq \epsilon$. This property of a partial solution is called *weak persistency.* There are certain partial solutions of the energy for which a *stronger* condition called *strong persistency* holds true. The strong persistency property states that if a variable X_p is labelled, then it is assigned the same label in all global minima $\mathbf{x}^*$ of the energy, *i.e.* $x_p = x_p^*$ for all $\mathbf{x}^* \in \{\arg\min_{\mathbf{x}} E(\mathbf{x})\}$.

Recently, there has been some interest in developing methods for computing partially optimal solutions of multi-label energy functions [45, 54]. The work of [45] addresses this problem by transforming the multi-label energy function to a function involving binary variables [37, 88]. The resulting binary energy function is then minimized by applying the QPBO algorithm. This approach produced interesting results, but is computationally expensive. The method proposed by Kovtun [54] to find partially optimal solutions constructs a submodular subproblem $\mathcal{P}_k$ for each label $l_k \in \mathcal{L}$. The random variables which are assigned label l_k after solving the subproblem $\mathcal{P}_k$ have an optimality certificate associated to them. An additional advantage of this method is the submodularity property satisfied by the subproblems, thus making them efficiently solvable (cf. §2.3.1). Partially optimal solutions obtained by the methods described here help us isolate the variables which have been assigned a label, and reduce the original energy minimization problem.

3.3 Efficient Multi-label Methods

We now present methods to improve the performance of algorithms for minimizing multi-label energy functions arising from discrete MRFs or CRFs. For ease of understanding, we explain the working of these techniques in the context of the α-expansion algorithm. However, our methods are general and are applicable to all popular algorithms such as $\alpha\beta$-swap, BP, Fast-PD and TRW-S (sequential TRW). Experimental results using all these algorithms are presented in the latter sections. We also limit our discussion to energy functions with unary and pairwise terms, *e.g.* (3.2.2), in this section. Methods for higher order terms are presented in section 3.4.

The techniques proposed in this chapter are inspired from the observations that the computation time of energy minimization algorithms primarily depends on: (a) The initialization used; and (b) The number of variables involved in the energy function. Thus, our primary goals are:

1. To generate a good initialization for the current problem instance, which results in a reduction in the amount of computation required for solving the problem.

2. To reduce the number of variables involved in the energy function in an efficient manner.

3.3.1 Recycling Primal and Dual Solutions

We achieve our first goal of obtaining a good initialization by recycling results from previous (related) problem instances. We call this method for α-expansion, the dynamic α-expansion algorithm. As discussed earlier (cf. §3.2.1), the α-expansion algorithm works by making a series of label changes, called moves, which lower the energy at each step. This iterative algorithm starts with an initial labelling. Each step considers a label $\alpha \in \mathcal{L}$, and solves the binary problem of assigning variables this label or retaining their current label. One iteration of the method involves performing expansion moves for all the labels in some order successively. The iterations are repeated until the energy cannot be reduced further for any label α. We denote the binary energy function corresponding to a particular 'α' move by $E^\alpha(\mathbf{x}^\alpha)$, and is defined as:

$$E^\alpha(\mathbf{x}^\alpha) = \sum_{i \in \mathcal{V}} \phi_i^\alpha(x_i^\alpha) + \sum_{(i,j) \in \mathcal{E}} \phi_{ij}^\alpha(x_i^\alpha, x_j^\alpha), \tag{3.3.1}$$

where $x_i^\alpha, x_j^\alpha \in \{0,1\}$, and correspond to x_i and x_j in the multi-label energy function respectively. The assignment $x_i^\alpha = 0$ implies that $x_i = \alpha$ in the multi-label energy function, while the assignment $x_i^\alpha = 1$ implies x_i retains its current label. The unary potential $\phi_i^\alpha(x_i^\alpha)$ is given by:

$$\phi_i^\alpha(x_i^\alpha) = \begin{cases} \phi_i(x_i = \alpha) & \text{if } x_i^\alpha = 0, \\ \phi_i(x_i = x_i^{\text{cur}}) & \text{if } x_i^\alpha = 1, \end{cases} \tag{3.3.2}$$

where x_i^{cur} is the current label assignment for X_i. The pairwise potentials, for the Potts model in (3.2.3), are defined as:

$$\phi_{ij}^\alpha(x_i^\alpha, x_j^\alpha) = \begin{cases} 0 & \text{if } x_i^\alpha = 0, x_j^\alpha = 0, \\ \gamma(1 - \delta(x_i^{\text{cur}} - x_j^{\text{cur}})) & \text{if } x_i^\alpha = 1, x_j^\alpha = 1, \\ \gamma & \text{otherwise}, \end{cases} \tag{3.3.3}$$

where $\delta(x_i^{\text{cur}} - x_j^{\text{cur}}) = 1$, if $x_i^{\text{cur}} = x_j^{\text{cur}}$, and 0 otherwise.

The above binary function is pairwise and submodular, if the pairwise potentials of the original multi-label energy function satisfy the metric conditions:

(3.2.6), (3.2.7), and (3.2.8). Thus, the binary energy function (3.3.1) can be minimized exactly by solving the equivalent st-mincut problem (cf. §2.3.2.1). The st-mincut problem is called the primal problem, and its solution, *i.e.* the labels assigned to all variables $x_i^\alpha, \forall i \in \mathcal{V}$ correspond to the primal solution. The st-mincut is found by solving the dual problem of maxflow on the same graph. The dual solution corresponds to the feasible flow solution of the maxflow problem. A new st-graph is built for solving each α-expansion move.

Recycling Flow across Iterations. When solving an expansion move in a particular iteration, we propose to recycle the flow from the corresponding move in the previous iteration to make the new computation faster. In the first iteration of the algorithm, we build one graph $G_i^1, i = 1, \ldots, k$, for each label expansion. The optimal expansion move for a given label l_i is computed by solving the st-mincut/maxflow problem on the graph G_i^1. Maxflow problems corresponding to all the labels are solved just as in standard α-expansion. In iterations $u > 1$ of the algorithm, instead of creating a new graph G_i^u for a label expansion, we recycle the corresponding graph G_i^{u-1} from the previous iteration exploiting the fact that the two graphs are similar. We use dynamic graph cuts technique proposed by Kohli and Torr [46] to achieve this. Given the solution of the maxflow problem on a graph, their method efficiently computes the maxflow in a modified version of the graph. Inspired by this idea, we update the maxflow solution of the graph G_i^{u-1} to obtain a good initialization for the graph G_i^u.

The dynamic update step involves changing the flows and the residual edge capacities, such that all edges satisfy the capacity constraints. In other words, we require that the flow in an edge is not more than its capacity. We illustrate the dynamic update step with an example in Fig. 3.2. It shows the case where the edge capacity between two nodes changes from one iteration to another. This change violates the capacity constraints of the edge, and is handled by reparameterizing the graph such that the final solution is not affected. The time complexity of all such updates is $\mathrm{O}(1)$, except for deleting an n-degree node where it is $\mathrm{O}(n)$.[3]

After the update operations, the maxflow algorithm is performed on the new

[3]A node is deleted by making the capacity of all the edges incident on it zero, which takes $\mathrm{O}(1)$ time per edge.

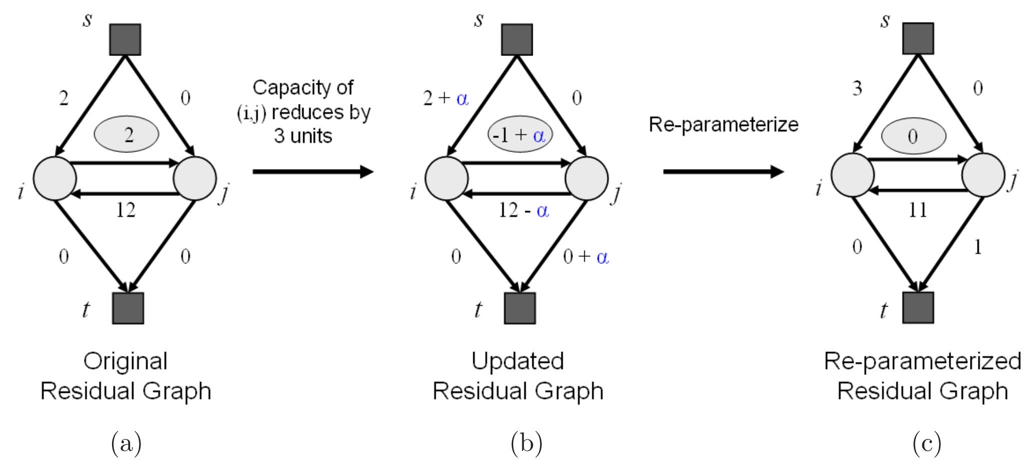


Figure 3.2: *We illustrate the dynamic update step using a graph containing two nodes i and j. Consider the expansion move in iteration u for label l_m. Performing maxflow computation on the graph corresponding to this move results in a residual graph shown in (a). In this example we assume that graphs G_m^{u+1} and G_m^u differ in the capacity of the edge (i,j) by 3 units. Incorporating this difference in the residual graph (a) violates the capacity constraint, i.e. residual edge capacity of the edge is negative. The edge capacities are made non-negative by reparameterizing the graph (cf. §2.3.2.2), without affecting the final solution. The graph is reparameterized by adding a constant $\alpha = 1$ to the capacity of the edges (i,j), (s,i) and (j,t), and subtracting it from the capacity of the edge (j,i), as shown in (b). The new residual graph, which corresponds to expansion move in iteration $u+1$ for label l_m is shown in (c). Maxflow computation on this graph is efficient [46]. Image courtesy of Pushmeet Kohli [46].*

residual graph. The efficiency of this computation depends on the number of update operations performed (see Fig. 9 in [46]). In the worst case, when all the edges are updated, this approach provides no speed-up and is as fast as the standard algorithm. However, our method is guaranteed to give some speed-up, because the number of changes in the graphs decrease in the latter iterations [18]. An example of this is shown in Fig. 3.3, a plot of the number of label changes, which corresponds to the changes in the graphs, against the iterations of the α-expansion algorithm. This leads to a decrease in the number of update and maxflow computations over time. Hence, the optimal moves in these iterations are computed efficiently.

For large problems, *i.e.* when the number of labels, k, or the number of pixels, n, is very large, maintaining multiple dual solutions may not be viable due to memory requirements. This issue can be overcome by working with a projected

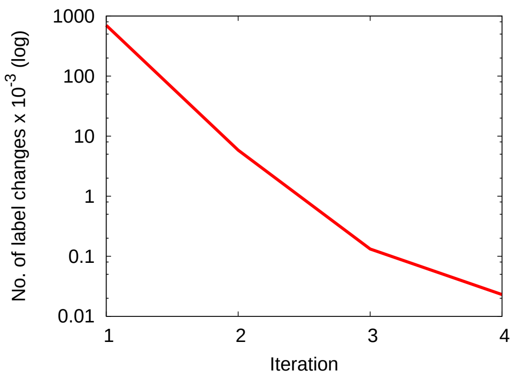


Figure 3.3: *The number of label changes in each iteration of the α-expansion algorithm. We use the stereo matching problem (Tsukuba image [86]) as an example here, and show that the number of label changes decreases in the latter iterations. Note that the number of label changes corresponds to the changes in the expansion move graphs from one iteration to another, i.e. G_α^{u-1} to G_α^u, $\forall\alpha$. Our strategy of recycling graphs in these iterations leads to a significant speed-up.*

energy function obtained from a partially optimal solution (cf. section 3.3.2). Thus our method is not only time-efficient but also memory-efficient if the projected energy function involves a small subset of random variables. The recycle scheme for single MRFs is summarized as follows:

1. Construct graphs $G_i^1, i = 1, \dots, k = |\mathcal{L}|$, in the first iteration.

2. Compute the maxflow solutions to get the optimal moves.

3. For iterations $u > 1$,

    - Update graphs from iteration $u - 1$.

    - Compute the new maxflow solutions for the residual graphs.

Efficiently Solving Dynamic MRFs For dynamic MRFs [46, 53], the task is to solve a problem where the data changes from one problem instance to the next. For instance, this occurs when solving a labelling problem on the image frames of a video sequence. The conventional method to solve such a problem is to use the standard α-expansion algorithm on each problem instance (*e.g.* each time instance) independently. This method is inefficient, given that the image frames are highly correlated, and would require a lot of computation time. We address this issue by recycling both the primal and dual solutions. The primal solution is generated by recycling the labelling of the previous problem instance, while the dual solution is computed by recycling the residual graphs (as in the single MRF case). Intuitively, if the data changes minimally from one problem instance to the next, the solution of a particular problem instance provides a good initialization for the subsequent one.

Consider a labelling problem defined on a video sequence. The first frame in the video sequence is labelled using the single MRF method described above. The primal and dual solutions thus obtained are used to initialize the maxflow/st-mincut problems for the next frame. The labelling (primal solution) of a frame t is initialized with the solution obtained for frame $t-1$. The graphs $G_i^1(t), i = 1, \ldots, k$, (dual solution) corresponding to the first iteration for frame t are obtained by dynamically updating the graphs from the last iteration for frame $t-1$. With these initializations the maxflow problem for each label is solved as in the single MRF case. In summary,

1. Solve frame 1 as a 'single MRF'.
2. For all frames $t > 1$,
    - Initialize the labelling (primal) using the solution of frame $t - 1$.
    - Initialize the graph flow (dual) from the corresponding solutions for frame $t - 1$.
    - Solve as a 'single MRF'.

These techniques for α-expansion provide similar speed-ups as the Fast-PD algorithm [53] as shown in section 3.5.1.

3.3.2 Reducing Energy Functions

We now propose a method to simplify (or reduce the number of unknown variables in) the MRF by solving the *easy* part. Our reduce strategy is applicable to many popular energy minimization approaches such as α-expansion, BP, TRW-S and Fast-PD, as illustrated in section 3.5. We also show how computations performed

Algorithm 1: *Pseudo-code for computing the partially optimal solution of an energy function. An auxiliary problem $\mathcal{P}_j$ for each label l_j is formulated as an st-mincut problem. The solution computed is used to project the energy function E by fixing the values of the labelled variables. After the iteration terminates we obtain a new energy function, E^p, comprising of all the unlabelled variables.*

```
input  : $\mathbf{X}$, $\mathcal{L} = \{l_1, \ldots, l_k\}$, $E$
output: Partially optimal solution
$s_j$: Set of variables taking label $l_j$ in the partially optimal solution;
$E^p \leftarrow E$;
for $j \leftarrow 1$ to $k$ do
    $\mathcal{P}_j \leftarrow$ Auxiliary problem for label $l_j$;
    $s_j \leftarrow$ Solve($E^p$, $\mathcal{P}_j$) (cf. §3.3.2);
    $E^p \leftarrow$ Project($E^p$, $s_j$);
end
```

during this procedure can be used to efficiently initialize the dynamic α-expansion algorithm described in the previous section.

As discussed earlier (§3.2.2), there are two main algorithms for obtaining partially optimal solutions of non-submodular multi-label energy functions. It would be interesting to compare these partially optimal solution algorithms for the segmentation and stereo problems, but is beyond the scope of our work. We chose to use the algorithm proposed by Kovtun [54] because it is an order of magnitude faster than the QPBO-based method. The key step of the Kovtun method is the construction of k auxiliary problems $\mathcal{P}_m$, one for each label $l_m \in \mathcal{L}$. Kovtun showed that the solution of problem $\mathcal{P}_m$ could be used to find variables that have the persistency property (described in §3.2.2). Thus, by solving all subproblems $\mathcal{P}_m$, $\forall l_m \in \mathcal{L}$, a partial solution which satisfies strong persistency can be obtained.

Specifically, problem $\mathcal{P}_m$ is the minimization of the following binary energy function

$$E^m(\mathbf{x}^m) = \sum_{i \in \mathcal{V}} \phi_i^m(x_i^m) + \sum_{(i,j) \in \mathcal{E}} \phi_{ij}^m(x_i^m, x_j^m), \tag{3.3.4}$$

where $x_i^m, x_j^m \in \{0, 1\}$, and correspond to x_i and x_j in the multi-label energy function respectively. The assignment $x_i^m = 0$ implies that $x_i = l_m$ in the multi-label energy function, while the assignment $x_i^m = 1$ implies the optimal label for

X_i has not been assigned yet. The unary potential $\phi_i^m(x_i^m)$ is given by:

$$\phi_i^m(x_i^m) = \begin{cases} \phi_i(x_i = l_m) & \text{if } x_i^m = 0, \\ \phi_i(x_i = l_i^{\min}) & \text{if } x_i^m = 1, \end{cases} \tag{3.3.5}$$

where $l_i^{\min} = \arg\min_{l \in \mathcal{L} - \{l_m\}} \phi_i(x_i = l)$. For the case of Potts model, the pairwise potentials are defined as:[4]

$$\phi_{ij}^m(x_i^m, x_j^m) = \begin{cases} 0 & \text{if } x_i^m = 0, x_j^m = 0, \\ 0 & \text{if } x_i^m = 1, x_j^m = 1, \\ \gamma & \text{otherwise.} \end{cases} \tag{3.3.6}$$

$E^m(\mathbf{x}^m)$ defines a submodular energy function and can be minimized by solving an st-mincut problem. Let $\mathbf{x}^{m*}$ denote the optimal solution of the subproblem $\mathcal{P}_m$. We extract a partially optimal solution $\mathbf{x} \in (\mathcal{L} \cup \{\epsilon\})^n$ of the multi-label function $E(\mathbf{x})$ as:

$$x_i = \begin{cases} l_m & \text{if } x_i^m = 0, \\ \epsilon & \text{otherwise.} \end{cases} \tag{3.3.7}$$

We repeat this process for all the labels $l_m \in \mathcal{L}$, and merge the solutions to obtain the final partially optimal solution of the original energy function $E(\mathbf{x})$.

To make this procedure computationally efficient, we project the energy function after every subproblem computation. This involves fixing values of all variables whose optimal labels have already been extracted from the solution of previous subproblem $\mathcal{P}_m$. This reduces the number of unknown variables in the multi-label energy function and makes the computation of subsequent auxiliary problems faster. We summarize this approach in Fig. 1. Our hope is that after solving all auxiliary problems, we would be left with a projection of the original energy function which involves far fewer variables compared to the original function $E(\mathbf{x})$. The experiments described in the next section on MRFs commonly encountered in computer vision confirm this behaviour.

The energy function projection obtained from the procedure described above corresponds to the *difficult* component of the energy function. It depends on

[4]Although the algorithm proposed in [54] only handles Potts model energy functions, it can be easily extended to general energy functions [55].

the variables whose optimal labels were not found. Thus, the original problem is now reduced to finding the labels of these variables. This can be done using any algorithm for approximate energy minimization. Results of this method are shown in Table 3.2. In the rest of this section, we show how this process can be made more efficient by reusing the solutions of the auxiliary problems solved during the partial optimality algorithm. Again, we will describe our approach using the α-expansion algorithm for ease of understanding.

Reusing solutions from the partial optimality algorithm. The remainder of the original problem, which corresponds to the difficult part of the energy function, can also be solved efficiently. From (3.3.1) and (3.3.4), it can be seen that the energy functions corresponding to the subproblems of the partial optimality and α-expansion algorithms have the same form. Thus, we can potentially reuse the solutions of the partial optimality subproblems to make the computation of the α-expansion moves faster. Specifically, we use the dual (flow) solutions of the partial optimality problems to generate an initialization for the expansion moves of the first iteration of the α-expansion algorithm (in a manner similar to that described in §3.3.1).

As discussed before, the potential improvement in computation time depends on the similarity of the two subproblems. Therefore, by making the subproblems of the partial optimality and the α-expansion algorithms similar, we can improve the running time. We note that for unassigned labels we have some choice as to their initialization, and a natural question arises as to whether any particular initialization is better. Consider the expansion and partial optimality subproblems with respect to a label $\alpha \in \mathcal{L}$, *i.e.* $l_m = \alpha$ in (3.3.5). From (3.3.2) and (3.3.5) it can be seen that the unary potentials of the partial optimality and α-expansion subproblems are identical if the current label assignment for X_i, $x_i^{\text{cur}} = l_i^{\min}$. This can be done by initializing the labelling for the α-expansion algorithm as: $x_i = l_i^{\min}$, where $l_i^{\min} = \arg\min_{l \in \mathcal{L}} \phi(x_i = l)$. The pairwise potentials may differ at most by the constant γ for the case $x_i^\alpha = 1, x_j^\alpha = 1$ (cf. (3.3.3) and (3.3.6)). This change makes the two problems similar, and potentially provides an improvement in computation time using our reuse strategy. Experimental results shown in Fig. 3.8 confirm this expected behaviour. Our proposed methods—*reduce*, *reuse* and *recycle*—can be used jointly as follows:

1. Compute the partially optimal solution and project the energy function. (*Reduce*)
2. To label the remaining nodes using α-expansion,
    - Initialize the labelling of each node i to $l_i^{\min} = \arg\min_{l \in \mathcal{L}} \phi_i(x_i = l)$.
    - Update the residual graphs from the k auxiliary problems to construct graphs for the first α-expansion iteration. (*Reuse*)
    - Restart the maxflow algorithms to compute optimal moves, using flow recycling between expansion moves. (*Recycle*)

So far, we have seen efficient methods to minimize multi-label energy functions composed of unary and pairwise potentials. Such energy functions are, however, unable to capture the rich statistics of natural scenes, making them severely restrictive [65]. Higher order clique potentials, which are defined on sets of interacting random variables, have been shown to overcome this limitation [42,43,65,74,79], but with a large computational cost typically. The following section aims to address the computational issues of higher order energy functions.

3.4 Solving $\mathcal{P}^n$ Potts Model Efficiently

Consider the problem of minimizing energy functions which contain higher order clique potentials. Specifically, we are interested in clique potentials which take the form of a $\mathcal{P}^n$ Potts model introduced in [42]. The $\mathcal{P}^n$ Potts model potential for cliques of size n is defined as:

$$\phi_c(\mathbf{x}_c) = \begin{cases} \gamma_k & \text{if } x_i = l_k, \forall i \in c, \\ \gamma_{\max} & \text{otherwise,} \end{cases} \tag{3.4.1}$$

where $\gamma_{\max} > \gamma_k, \forall l_k \in \mathcal{L}$. It can be easily verified that the standard Potts model in (3.2.3) is a special case of this model with $n = 2$ and $\gamma_k = 0, \forall k$. Energy functions containing $\mathcal{P}^n$ Potts model potentials can be solved using the α-expansion and $\alpha\beta$-swap move making algorithms. The optimal expansion/swap move is computed by minimizing a binary energy function using the st-mincut algorithm as shown in [42]. Although our methods are applicable to both these move making algorithms, we describe them in the context of α-expansion for ease of understanding. The higher order binary energy function corresponding to a particular 'α' move will be denoted by $E_h^\alpha(\mathbf{x}^\alpha)$. It is defined as:

$$E_h^\alpha(\mathbf{x}^\alpha) = \sum_{i\in\mathcal{V}} \phi_i^\alpha(x_i^\alpha) + \sum_{(i,j)\in\mathcal{E}} \phi_{ij}^\alpha(x_i^\alpha, x_j^\alpha) + \sum_{\substack{c\in\mathcal{C}\\|c|>2}} \phi_c^\alpha(\mathbf{x}_c^\alpha), \tag{3.4.2}$$

where $x_i^\alpha, x_j^\alpha \in \{0,1\}$, $\mathbf{x}_c^\alpha = \{x_i^\alpha, \forall i \in c\}$. The unary potential $\phi_i^\alpha(x_i^\alpha)$ and the pairwise potential $\phi_{ij}^\alpha(x_i^\alpha, x_j^\alpha)$ are given in equations (3.3.2) and (3.3.3) respectively. The clique potential $\phi_c^\alpha(\mathbf{x}_c^\alpha)$ forms a $\mathcal{P}^n$ Potts model, and is given by:

$$\phi_c^\alpha(\mathbf{x}_c^\alpha) = \begin{cases} \gamma_\alpha & \text{if } x_i^\alpha = 0, \forall i \in c, \\ \gamma & \text{if } x_i^\alpha = 1, \forall i \in c, \\ \gamma_{\max} & \text{otherwise,} \end{cases} \tag{3.4.3}$$

where $\gamma = \gamma_\beta$ if $x_i^{\text{cur}} = \beta \in \mathcal{L}$, for all $i \in c$, and $\gamma = \gamma_{\max}$ otherwise. This move energy function is submodular and can be solved using the st-mincut algorithm on the graph shown in Fig. 3.4. The reader is referred to [42] for more details of the graph construction.

Recycling Solutions. Once the st-mincut graph corresponding to the higher order move energy is built, our methods for recycling primal and dual solutions (cf. §3.3.1) are directly applicable. When solving an expansion move in a particular iteration, we recycle the flow from the corresponding move in the previous iteration to make the new computation faster.

Computing Partially Optimal Solutions. We now propose a method to efficiently compute partially optimal solutions of energy functions containing $\mathcal{P}^n$ Potts potentials. As in §3.3.2, our method is based on the algorithm proposed by Kovtun [54]. An auxiliary problem P_m, for label $l_m \in \mathcal{L}$, is the minimization

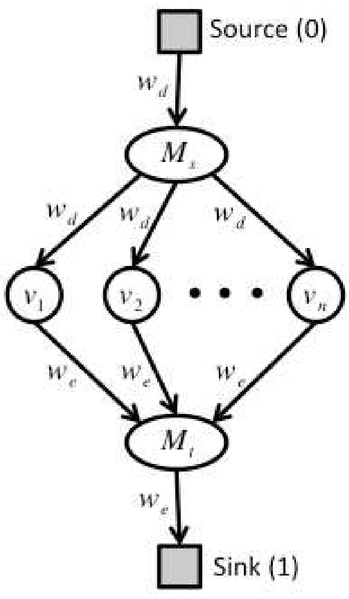


Figure 3.4: *Graph construction for computing the optimal α-expansion move for the $\mathcal{P}^n$ Potts model is shown here. The nodes $v_1, v_2, \cdots, v_n$ represent the pixels in the clique. There are also two auxiliary nodes M_s and M_t. After the computation of st-mincut, if v_i is connected to the source then $x_i^\alpha = 0$, and if v_i is connected to the sink then $x_i^\alpha = 1$. The weights of the graph are given by $w_d = \gamma_{\max} - \gamma_\alpha$ and $w_e = \gamma_{\max} - \gamma$.*

of the following higher order binary energy function:

$$E_h^m(\mathbf{x}^m) = \sum_{i \in \mathcal{V}} \phi_i^m(x_i^m) + \sum_{(i,j) \in \mathcal{E}} \phi_{ij}^m(x_i^m, x_j^m) + \sum_{\substack{c \in \mathcal{C} \\ |c|>2}} \phi_c^m(\mathbf{x}_c^m), \tag{3.4.4}$$

where $x_i^m, x_j^m \in \{0,1\}$, $\mathbf{x}_c^m = \{x_i^m, \forall i \in c\}$. Note that x_i^m, x_j^m correspond to x_i and x_j respectively in the multi-label energy function. The unary potential $\phi_i^m(x_i^m)$ and the pairwise potential $\phi_{ij}^m(x_i^m, x_j^m)$ are given by equations (3.3.5) and (3.3.6) respectively. The $\mathcal{P}^n$ Potts clique potential $\phi_c^m(\mathbf{x}_c^m)$ is defined as:

$$\phi_c^m(\mathbf{x}_c^m) = \begin{cases} \gamma_m & \text{if } x_i^m = 0, \forall i \in c, \\ \min\limits_{k \in \mathcal{L}, k \neq m} \gamma_k & \text{if } x_i^m = 1, \forall i \in c, \\ \gamma_{\max} & \text{otherwise.} \end{cases} \tag{3.4.5}$$

It can be easily verified that $E_h^m(\mathbf{x}^m)$ is a submodular energy function [88]. We now provide the relevant notation to prove Theorem 1 in [54], which leads to the persistency property, for the case of $\mathcal{P}^n$ Potts model.

For every auxiliary problem P_m we consider any ordering of the label set $\mathcal{L} = \{l_1, l_2, \ldots, l_k\}$, such that label l_m is the *highest* label. This allows us to define a partial ordering on the set of label pairs $(a, a') \in \mathcal{L} \times \mathcal{L}$. The maximum and minimum for any two label pairs (a, a') and (b, b') are defined as $(a, a') \vee (b, b') = (a \vee b, a' \vee b')$ and $(a, a') \wedge (b, b') = (a \wedge a', b \wedge b')$ respectively. Similarly, the maximum and minimum of any pair of labellings $\mathbf{x}_c$ and $\mathbf{x}'_c$ is denoted by $\mathbf{x}_c \bigvee \mathbf{x}'_c$ and $\mathbf{x}_c \bigwedge \mathbf{x}'_c$ respectively. We also define the *lowest* optimal labelling $\widehat{\mathbf{x}_c^m}$ as follows:

$$\widehat{\mathbf{x}_c} = \bigwedge_{\mathbf{x}_c^* = \arg\min_{\mathbf{x}_c} E(\mathbf{x}_c)} \mathbf{x}_c^*. \tag{3.4.6}$$

Using this notation, the submodularity condition in equation (3.2.4) can be written as:

$$f(\mathbf{x}_c) + f(\mathbf{x}'_c) \geq f(\mathbf{x}_c \bigvee \mathbf{x}'_c) + f(\mathbf{x}_c \bigwedge \mathbf{x}'_c). \tag{3.4.7}$$

Let $\mathbf{y}^m \in \mathcal{L}^n$ denote the partially optimal solution after solving the auxiliary problem corresponding to label l_m (*i.e.* $E_h^m(\mathbf{x})$). In other words, the labelling $x_i^m = 0$ is equivalent to $y_i^m = l_m$, and $x_i^m = 1$ to the random variable X_i retaining the initial label.

Theorem 3.4.1 *An arbitrary solution of the initial problem $\mathbf{x}^* = \arg\min_{\mathbf{x}} E_h(\mathbf{x})$ satisfies the following condition: $\mathbf{x}^* \bigwedge \widehat{\mathbf{y}^m} = \widehat{\mathbf{y}^m}$, where $\widehat{\mathbf{y}^m}$ denotes the lowest optimal labelling for the auxiliary problem P_m.*

This theorem states that the lowest optimal labelling for a pixel in the original problem is not lower than the label given to the corresponding pixel in the auxiliary problem solution. This allows us to assign optimal labels to all pixels which take the label l_m in the solution for the auxiliary problem P_m, thus showing that the persistency property holds for our higher order energy function. We use the following Lemma to prove the theorem.

Lemma 3.4.2 *Let $\widehat{\mathbf{x}}$ be the lowest optimal labelling for a submodular problem, and $\mathbf{x}^*$ be any arbitrary labelling satisfying the condition: $\mathbf{x}^* \bigwedge \widehat{\mathbf{x}} \neq \widehat{\mathbf{x}}$, then $E_h(\mathbf{x}^*) > E_h(\mathbf{x}^* \bigvee \widehat{\mathbf{x}})$.*[5]

[5]The lemma can be proved easily using the submodularity condition in equation (3.4.7) and the definition of lowest optimal labelling, *i.e.* $E_h(\mathbf{x}^* \bigwedge \widehat{\mathbf{x}}) > E_h(\widehat{\mathbf{x}})$. See [54] for more details.

Proof of Th. 3.4.1: Our proof is similar to that given in [54]. Let us assume for any labelling $\mathbf{x}$, $\mathbf{x} \bigwedge \widehat{\mathbf{y}^m} \neq \widehat{\mathbf{y}^m}$. From Lemma 3.4.2 it follows that:

$$E_h^m(\mathbf{x} \bigvee \widehat{\mathbf{y}^m}) < E_h^m(\mathbf{x}). \tag{3.4.8}$$

The following inequality is obtained from equations (3.4.1) and (3.4.5):

$$\phi_c^m(\mathbf{x}_c \vee \widehat{\mathbf{y}_c^m}) - \phi_c^m(\mathbf{x}_c) \geq \phi_c(\mathbf{x}_c \vee \widehat{\mathbf{y}_c^m}) - \phi_c(\mathbf{x}_c). \tag{3.4.9}$$

Also, from [54],

$$\phi_{ij}^m(x_i \vee \widehat{y_i^m}, x_j \vee \widehat{y_j^m}) - \phi_{ij}^m(x_i, x_j) \geq \phi_{ij}(x_i \vee \widehat{y_i^m}, x_j \vee \widehat{y_j^m}) - \phi_{ij}(x_i, x_j). \tag{3.4.10}$$

Using inequalities (3.4.8), (3.4.9) and (3.4.10) it can be easily shown that,

$$E_h(\mathbf{x} \bigvee \widehat{\mathbf{y}^m}) < E_h(\mathbf{x}), \tag{3.4.11}$$

which proves that any labelling $\mathbf{x}$ that does not satisfy the condition $\mathbf{x} \bigwedge \widehat{\mathbf{y}^m} = \widehat{\mathbf{y}^m}$ has a higher energy compared to $\mathbf{x} \bigvee \widehat{\mathbf{y}^m}$, which is a solution containing the auxiliary problem solution. ∎

Thus, the persistency property holds for our higher order energy function. We extract a partially optimal solution of the multi-label function $E_h(\mathbf{x})$ using equation (3.3.7). The final partially optimal solution is obtained by repeating this process for all the labels, and merging the solutions.

3.5 Experiments

We evaluated our methods on a variety of multi-label MRF problems such as stereo matching [18], colour-based [16], object-based [94, 95], and texture-based [42] segmentation. The details of the unary and pairwise potentials of the energy functions used for formulating these problems are given below.

Colour-based Segmentation. For the colour-based segmentation problem, we used the energy function defined in [16]. The unary potentials $\phi_i(x_i), i \in \mathcal{V}$,

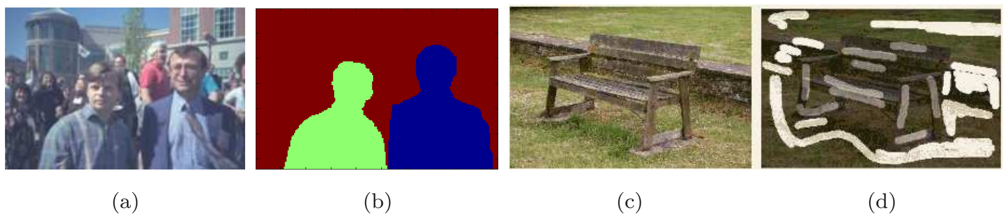


Figure 3.5: *(a) The key frame of the 'Dayton' video sequence, and (b) its segmentation. (c) An image from the* MSRC-21 *database, and (d) the brush strokes marked by the user indicating the segment labels. The key frame segments and brush strokes are used to learn the colour histogram models and the patch dictionaries.*

are defined using the RGB distributions $\mathcal{H}_a, a = l_1, \ldots, l_k$, of the k segment labels as follows:

$$\phi_i(x_i) = -\log p(x_i = a|\mathcal{H}_a). \tag{3.5.1}$$

The distributions $\mathcal{H}_a$ are obtained using user-specified constraints. These constraints can be segmentation seeds marked by the user to indicate segment labels (see Fig. 3.5(d)). The pairwise potentials encourage contiguous segments while preserving the image edges [16], and take the form of a Generalized Potts model defined as:

$$\phi_{ij}(x_i, x_j) = \begin{cases} \lambda_1 + \lambda_2 \exp\left(\frac{-g^2(i,j)}{2\sigma^2}\right)\frac{1}{\text{dist}(i,j)} & \text{if } x_i \neq x_j, \\ 0 & \text{if } x_i = x_j, \end{cases} \tag{3.5.2}$$

where λ_1, λ_2 and σ are parameters of the model. The terms $g(i,j)$ and $\text{dist}(i,j)$ give the difference in RGB values and the spatial distance respectively between pixels i and j. We used the following parameter values for all our experiments with this energy function: $\lambda_1 = 5, \lambda_2 = 100$ and $\sigma = 5$. Segmentation results are shown on the well-known garden image and a cow image used in [39, 46].

Stereo Matching. We used the pairwise energy function in [54] for the stereo matching problem. The unary potentials of the energy are computed with a fixed size window-based method. Windows of size 15×15 centred over every pixel i in the left image and its corresponding pixel in the right image (for a given disparity) are used. The cost of labelling pixel i with this disparity is given by the normalized sum of squared colour intensity differences between the left and right image window pixels. The pairwise potentials take the form of a Potts model (3.2.3). Stereo matching results are shown on “Tsukuba”, “Venus”, “Cones”, “Teddy” images from the Middlebury stereo data set [86]. The Potts model smoothness cost γ was set to 20 for all our experiments on this energy function.

Object-based Segmentation. For this problem we used the energy function defined in [95]. The unary potentials of this energy are based on shape-texture, colour, and location features. It is given by:

$$\phi_i(x_i) = \theta_T \phi_T(x_i) + \theta_{col}\phi_{col}(x_i) + \theta_l \phi_l(x_i), \tag{3.5.3}$$

where θ_T, θ_{col}, θ_l are model parameters. The component $\phi_T(x_i)$ is learnt using a boosted classifier [103]. The classifier combines discriminative texture and shape filter response features and models the texture, layout, and textural context of object classes. The colour component potential $\phi_{col}(x_i)$ is computed using Gaussian Mixture Models (GMMs) in the CIELab colour space. The location potential $\phi_l(x_i)$ captures the relation between absolute location of the pixel and the object class label. The reader is referred to [95] for more details on computing these potentials. The pairwise potentials take the form of a contrast sensitive Potts model (3.5.2). We evaluated our algorithms on this energy function using images from the MSRC-21 database.

Texture-based Segmentation. In this problem, the task is to segment an image, given a set of distinct textures, such as texton histograms [92] or a dictionary of RGB patches, together with their object class labels. The unary potential is specified by RGB distributions, while the pairwise potential is a contrast sensitive Potts model (3.5.2), similar to the colour-based segmentation example. The rich statistics of natural images provide by texture information [68, 107] are encoded in the form of $\mathcal{P}^n$ Potts higher order potential. Following the work of [42], we represent the texture of each object class $s \in \{1, 2, \cdots, n\}$, using a dictionary $\mathbf{P}_s$ of $n_p \times n_p$ RGB patches. The higher order potential $\phi_c(\mathbf{x}_c)$ of a clique patch c is

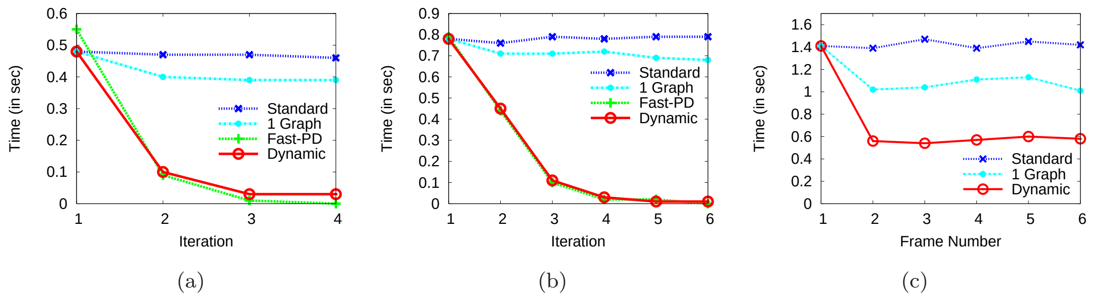


Figure 3.6: *Recycling primal and dual solutions for (a), (b) single and (c) dynamic* MRF *problems: Comparison of run-times of standard and dynamic versions of α-expansion, and Fast-PD are shown for (a) object-based segmentation problem: 'Building' image from the* MSRC-21 *data set [95], (b) stereo matching problem: Tsukuba (Left image), and (c) colour-based segmentation problem: cow video sequence [39,46]. In (a), (b) reusing the dual solution provides a speed-up of at least 4-10 times in subsequent iterations. In some cases, the first iteration of Fast-PD was slightly slower compared to both versions of α-expansion algorithm, but the overall computation time was better than 'standard' and comparable to 'dynamic'. For example, times for the 'Building' image are: Fast-PD: 0.65s, dynamic: 0.64s, standard: 1.88s. Note that the run-times of Fast-PD and our dynamic version are very similar in (a) and (b). In (c) the dynamic version reuses primal and dual solutions from the previous frames in the video sequence and results in 3-4 times speed-up. We also show that the strategy of maintaining only one graph while recycling solutions (denoted by '1 Graph') provides insignificant speed-up (see text).*

given by:

$$\phi_c(\mathbf{x}_c) = \begin{cases} \lambda_3 G(c,s) & \text{if } x_i = s, \forall i \in c, \\ \lambda_4 & \text{otherwise,} \end{cases} \tag{3.5.4}$$

where λ_3 and λ_4 are model parameters. The function $G(c,s)$ is the minimum difference between the RGB values of clique patch c and all patches in the dictionary $\mathbf{P}_s$. The patch dictionaries are learnt from a manually segmented key frame in the case of video segmentation, *e.g.* Dayton sequence (Fig. 3.5(b)), or user-marked brushed strokes in the case of an image segmentation, *e.g.* Bench image (Fig. 3.5(d)). We used patches of size 4×4, with the following parameters: $\lambda_1 = 0.6, \lambda_2 = 6, \lambda_3 = 0.6, \lambda_4 = 6.5$ and $\sigma = 5$. More details of the higher order potential can be found in [42].

The following sections describe the results of primal and dual, and partially optimal solution initializations. Standard, publicly available implementations are

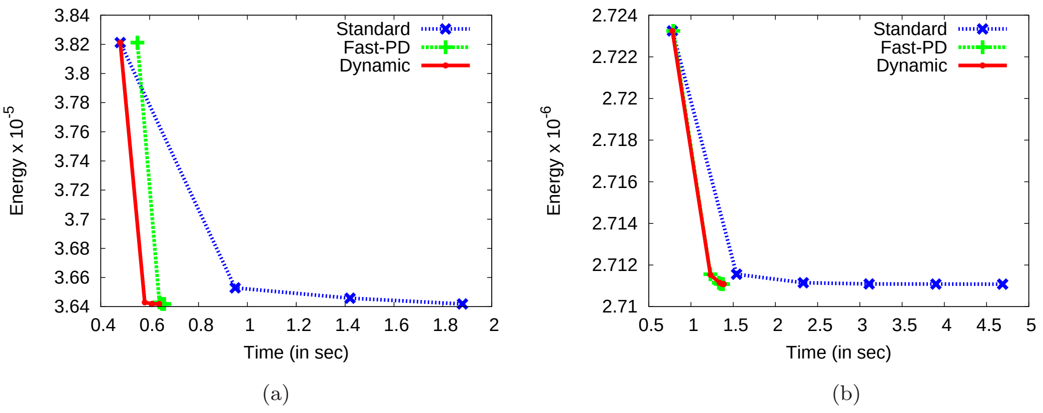


Figure 3.7: *Comparison of run-times and solution energy of standard and dynamic versions of α-expansion and Fast-PD are shown for (a) 'Building' image, (b) Tsukuba (Left image). Although there is a small change in energy after iteration 1, Standard α-expansion spends much more time compared to our Dynamic version to obtain a new lower energy solution. The time vs energy plot for Fast-PD is very similar to dynamic α-expansion, except for iteration 1 in (a), where Fast-PD takes 0.07 seconds more than our dynamic algorithm.*

used for comparison.[6] All experiments were performed on a Intel Core 2 Duo, 2.4 GHz, 3GB RAM machine. Source code for the proposed methods is available at `http://cms.brookes.ac.uk/research/visiongroup`.

3.5.1 Dynamic α-expansion

We now discuss the effect of various primal and dual solution initializations on the α-expansion algorithm. We tested a simple of way of using the flow/cut from the solution of the previous expansion move (*i.e.* with a different label) as an initialization for the current move. From (3.3.1) it can be observed that the energy functions corresponding to two consecutive moves are substantially different. Hence, this scheme provides no significant speed-up. Fig. 3.6 confirms this expected behaviour.

In Figures 3.6(a) and 3.6(b) we show the results of the proposed 'recycle' strategy for two single MRF examples. The primal and dual solutions are recycled across iterations (cf. §3.3.1). The standard and dynamic versions take the same time in the first iteration, as no flow is recycled. In the subsequent iterations, the

[6]We thank V. Kolmogorov, N. Komodakis and M. Pawan Kumar for providing the original implementation of their methods for comparison.

	Time (in seconds)		
	α-exp	dyn α-exp	opt α-exp
Dayton (3)	1.31	0.49	0.21
Garden (4)	1.20	0.44	0.19
Bench (3)	1.76	0.59	0.38
Beach (4)	1.59	0.51	0.25

Table 3.1: *Running times (in seconds) for various examples (from [43]) using the $\mathcal{P}^n$ Potts model. Results are shown for the α-expansion algorithm. 'α-exp' refers to the times obtained using the standard alpha expansion algorithm and 'dyn α-exp' refers to the dynamic version (which recycles primal and dual solutions). 'opt α-exp' refers to the optimized version which computes the partially optimal solution followed by α-expansion on the energy projection. It is observed that both 'dyn' and 'opt' methods provide a speed-up of at least 3-6 times compared to the standard method. The numbers in () denote the number of labels in the problem.*

dynamic version provides a speed-up of 4-10 times. Similar results were observed for other problems as well. The approach of initializing both primal and dual solutions in a dynamic MRF was tested on the cow video sequence [39,46]. These run-times for a sequence of 6 images are shown in Fig. 3.6(c). Our initialization method provides a speed-up of 3-4 times in this case. The graphs also compare the dynamic methods with Fast-PD [53]. Note that our methods resulted in very similar run-times compared to Fast-PD. Fig. 3.7 shows a comparison of run-time and solution energy for standard and dynamic versions of α-expansion. From Fig. 3.6 and Fig. 3.7 we see that the speed-up achieved by our dynamic version is due the fact that small changes in energy can be computed very efficiently. Table 3.1 shows the speed-up obtained for the $\mathcal{P}^n$ Potts model. Our approach provides a speed-up of at least 3-5 times compared to the standard α-expansion algorithm.

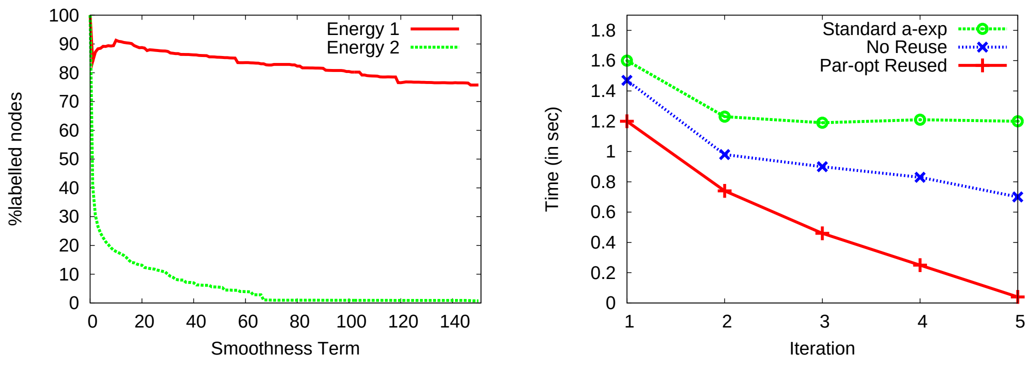
3.5.2 Using Partially Optimal Solutions

We now show the results of our partially optimal solution based method (cf. §3.3.2) on a variety of energy minimization algorithms for the problems defined above. Specifically, α-expansion, BP and TRW-S algorithms are used in the experiments. Optimized versions of BP and TRW-S refer to the computation of

	Time (in seconds)						
	α-exp	Fast-PD	opt α-exp	BP	opt BP	TRW-S	opt TRW-S
Colour-based Segmentation:							
Cow (3)	2.53	1.31	**0.21**	95.93	**0.32**	98.36	**0.33**
Cow (4)	3.75	1.72	**0.38**	108.32	**0.42**	111.69	**0.43**
Garden (4)	0.28	0.14	**0.04**	5.59	**0.17**	5.89	**0.21**
Stereo:							
Tsukuba (16)	5.74	1.47	**0.84**	38.19	**4.47**	41.74	**4.67**
Venus (20)	11.87	3.07	**3.03**	67.04	**14.97**	71.46	**16.02**
Cones (60)	42.23	9.48	**4.36**	173.35	**29.41**	182.66	**30.70**
Teddy (60)	44.25	9.56	**8.27**	172.30	**60.35**	182.50	**63.77**
Object-based Segmentation:							
Plane (4)	0.39	0.35	**0.15**	9.41	**0.29**	9.89	**0.30**
Bikes (5)	0.82	0.54	**0.22**	10.69	**0.64**	11.19	**0.70**
Road (5)	0.91	0.51	**0.18**	10.67	**0.60**	11.26	**0.62**
Building (7)	1.32	0.89	**0.38**	12.70	**2.57**	13.52	**2.66**
Car (8)	0.99	0.53	**0.11**	13.68	**0.23**	14.42	**0.24**

Table 3.2: *Running times for various single* MRF *problems: Comparison of the run-times (in seconds) of the standard and optimized (opt) versions of α-expansion (α-exp),* BP, TRW-S *is shown. The optimized version refers to computing the partial solution followed by solving the energy projection with the corresponding algorithm. The optimized versions are significantly faster in all the examples. The speed-up obtained depends on the nature and difficulty of the problem. The run-times shown for both* BP *and* TRW-S *versions correspond to the first 70 iterations. The number of iterations was chosen such that acceptable qualitative results (segmentation or stereo map) were obtained for all the problems. Some of the smaller problems produce results after 30-40 iterations, while others take 70-80 iterations. A better comparison of time vs energy is shown in Fig. 3.7 and Fig. 3.10. The numbers in () denote the number of labels in each problem.*

partially optimal solution followed by running the corresponding algorithm on the projected energy function. A comparison of the run-times for all these algorithms is shown in Table 3.2. It is observed that our method achieves a speed-up is 10-15 times for most of the examples. In some cases (*e.g.* Cow image with 3 labels), the speed-up is more than 100 times for optimized versions of TRW-S and BP algorithms. The amount of speed-up depends on the strength of the pairwise terms and the number of labels in the problem. The speed-up increases with a decrease in both the number of labels and the strength of the pairwise terms. This is because the pairwise potential of the partial optimality auxiliary problem (3.3.6) is closely related to that in the original problem (3.2.3). Images



(a) (b)

Figure 3.8: *(a) The percentage of nodes labelled by the partially optimal solution algorithm by varying the smoothness cost for two energy functions. The Tsukuba stereo matching problem with energy functions given in [54] (Energy 1) and [99] (Energy 2) is used as the example here. For the smoothness cost* $\gamma = 20$*, only 13% of the nodes are labelled in the case of 'Energy 2'. (b) The energy function in [99] (Energy 2) with smoothness cost* $\gamma = 20$ *is used for this experiment on the Tsukuba sequence. The speed-up obtained by reusing the flows from the partially optimal solution auxiliary problems (Par-opt) for this smoothness cost is shown. Reusing the flows provides a run-time improvement of at least 5 times in the last two iterations, and more than 2 times overall improvement. Note that even when the partially optimal solution algorithm fails, we obtain a significant speed-up.*

with highly textured regions also show orders of magnitude speed-up for segmentation and stereo problems. Table 3.1 shows the speed-up obtained for the $\mathcal{P}^n$ Potts model for various examples (from [42]). Using partially optimal solutions provides a speed-up of at least 4-6 times compared to standard α-expansion.

An analysis of the partially optimal solution algorithm shows that in some cases very few nodes may be labelled. One such case is when the smoothness cost γ is very high, as shown in Fig. 3.8(a). For illustration purposes we chose the Tsukuba stereo problem, which showed the most significant change in the number of labelled nodes. We used two energy functions [54, 99] on the stereo problem to demonstrate the effect of varying the smoothness term. The unary potential in [54] is computed using a normalized cross correlation approach on pixel windows of size 15×15, while [99] uses the sub-pixel window approach proposed by [10]. The pairwise potential in both cases is the Potts model given by (3.2.3). As the smoothness cost is increased, the percentage of labelled nodes decreases, and the projected component of the energy function remains large.

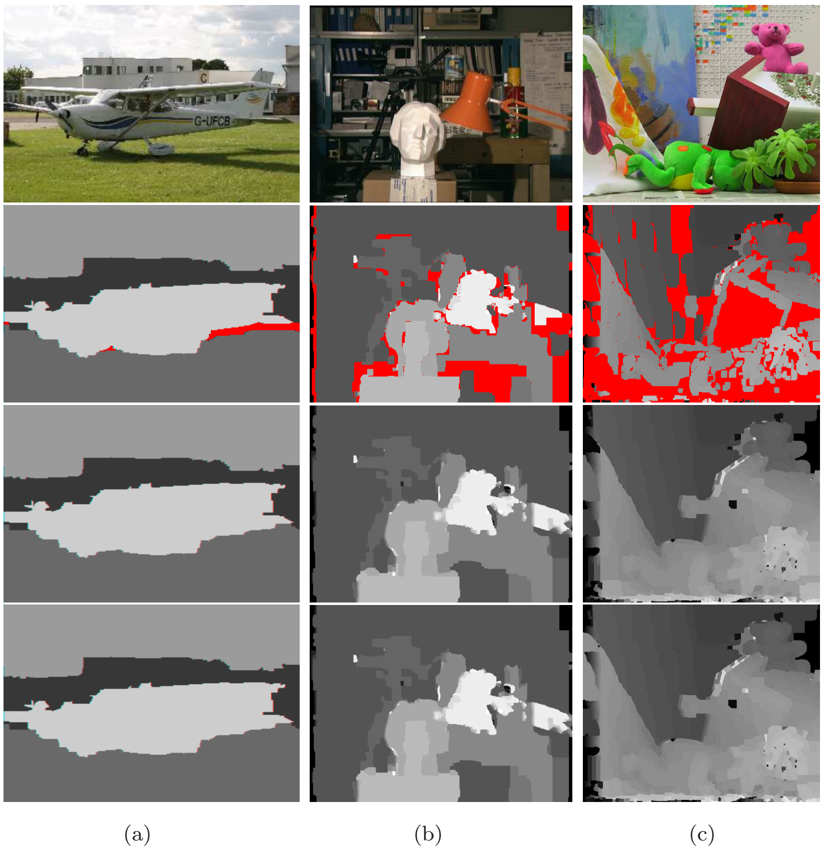
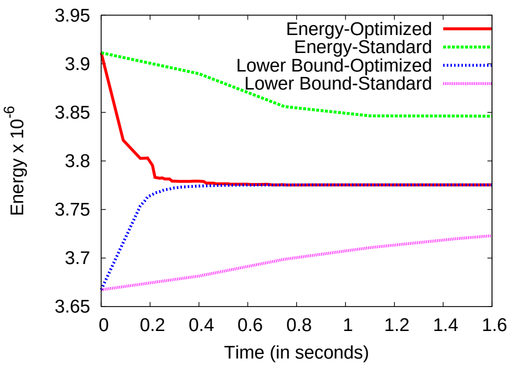


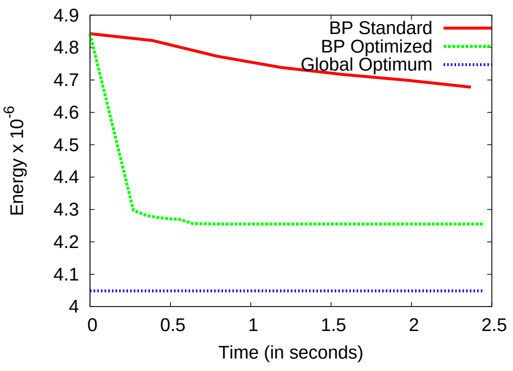
Figure 3.9: *A sample result of object-based segmentation is shown in (a) Plane. Some of the stereo matching results are shown in (b) Tsukuba-Left and (c) Teddy-Left. The first row shows the original images. The second row shows the partially optimal solution. The regions marked in red denote the unlabelled pixels, which have low texture detail. Our method provides more than* 6× *speed-up even when majority of the nodes are unlabelled in the Teddy example. (This figure is best viewed in colour.)*

The decrease is more dramatic using the energy function in [99]. This effect is perhaps because the partially optimal solution algorithm relies on strong unary potentials. In the case of [99], a large smoothness term dominates the unary potentials, and leads to many unlabelled nodes. Thus, only a small improvement in run-time performance is achieved. However, our strategy of reusing the flow from the partially optimal solution auxiliary problems always provides improved performance in these cases (see Fig. 3.8(b)).

Segmentation and stereo matching results of some of the images used in our experiments are shown in Fig. 3.9. Note that even when majority of the nodes are unlabelled in the partially optimal solution, *e.g.* Teddy sequence in Fig. 3.9(c), our method provides more than 6 times speed-up. The proposed method is not



(a)



(b)

Figure 3.10: *(a) Energy of the solution and lower bound obtained by running* TRW-S *algorithm on the Road image example [95]. Note that optimized* TRW-S *algorithm finds better energies (lower solution energy and higher lower bound) at any given point in time. It also finds an optima in only 0.64 seconds. Standard* TRW-S *converged to this energy after 37.24 seconds. Thus, the optimized version is more than 50 times faster. (b) Solution energies obtained by running standard and optimized* BP *algorithm on the Building image example [95]. Optimized* BP *refers to the computation of partially optimal solution followed by running the* BP *algorithm on the projected energy function. It finds an energy closer to the global optimum, while standard* BP *does not reach this energy even after 30 seconds.*

only computationally efficient, but also provides a lower energy solution empirically in the case of TRW-S and BP. Furthermore, the optimality of the solutions is not compromised. Fig. 3.10(a) compares the energies of the solutions and lower bounds obtained using standard and optimized versions of TRW-S. The optimized version using the energy function projection converges to the global optima of the energy in only 0.64 seconds. Fig. 3.10(b) compares the energies of the solution obtained using the standard and optimized BP algorithms. Optimized BP converges to a low energy, although not the global optima, in 0.85 seconds, while standard BP converges to a much higher energy in 11.12 seconds. Standard BP solves the original (large) problem and converges to a local optima. On the other hand, optimized BP solves the projected energy function defined on a subset of nodes and converges to a better local optima. Empirically, we observe that BP is more likely to provide a better local optima on the smaller problem (defined by the projected energy function), which is easier to solve compared to the original large problem. The solutions corresponding to these energies are shown in Fig. 3.11. Note that the optimized BP solution is closer to the global optima in

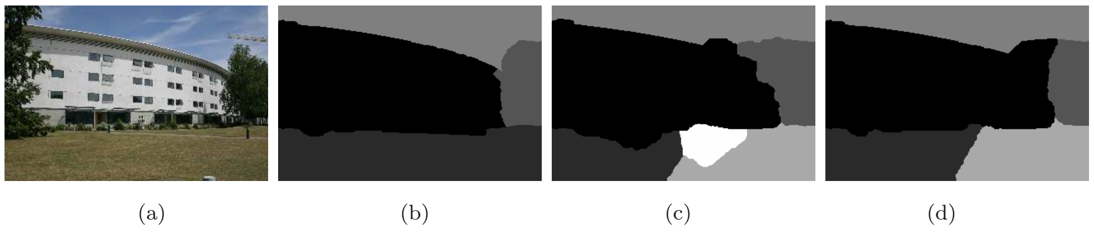


Figure 3.11: *(a) Building image [95], and (b) the global optimum solution computed by the* TRW-S *algorithm. Solutions obtained using (c) standard* BP, *and (d) optimized* BP *with an 8-neighbourhood. Neither the optimized nor the standard versions converge to the optimal solution. However, optimized* BP *is closer to the optima.*

this case.

3.6 Summary

This chapter proposes techniques for improving the performance of algorithms for solving multi-label MRFs. As there are no disadvantages in using them and many advantages we would expect them to become standard. Our methods work by recycling solutions from previous problem instances, and reducing energy functions utilizing algorithms for generating partially optimal solutions. Our work on recycling the dual (flow) solution for computing optimal label moves across successive iterations of the α-expansion algorithm results in a dynamic algorithm. It can be seen as an extension of the work of [39, 46] for minimizing multi-label non-submodular energy functions. Experimental results show that our methods provide a substantial improvement in the performance of α-expansion, TRW-S, and BP algorithms. Our method also provides similar or better performance compared to Fast-PD. We expect that our techniques for simplifying energy functions, and the subsequent recycling of computations performed during this procedure can also be used to make Fast-PD faster. The main contributions of this chapter are:

1. Proposing novel efficient methods for solving multi-label energy functions.

2. Extending the work on dynamic graph cuts to a certain class of non-submodular energy functions.

3. Proving that partially optimal solutions can be computed for $\mathcal{P}^n$ Potts model.

4. Demonstrating that our efficient methods are widely applicable.

Chapter 4

Exact Inference for Higher Order CRFs

In the previous chapter we have addressed the problem of efficient approximate inference for energy functions involving certain higher order potentials. This chapter addresses the problem of exactly inferring the MAP solution of such multi-label higher order energy functions. We present a framework to transform special classes of multi-label higher order functions to submodular second order boolean functions (referred to as $\mathcal{F}_s^2$), which can be minimized exactly using graph cuts, and we also characterize those classes. The basic idea is to use two or more boolean variables to encode the states of a single multi-label variable. There are many ways in which this can be done and much interesting research lies in finding ways which are optimal or minimal in some sense. We study the space of possible encodings and find the ones that can transform the most general class of functions to $\mathcal{F}_s^2$.

4.1 Introduction

Recall (§2.3.1) that a special class of functions called *submodular* functions can be minimized globally in polynomial time. These functions are discrete analogues of continuous convex functions. The current best algorithm for general submodular function minimization has complexity $\mathrm{O}\left(n^5 Q + n^6\right)$, where n is the number of random variables and Q is the time taken to evaluate the function [73]. This makes their use infeasible for problems in computer vision which, in general, involve a large number of variables. However, certain subclasses of submodular functions can be minimized much more efficiently. For example, boolean submodular functions of order[1] at most three can be minimized by solving an st-MINCUT problem, for which efficient algorithms are known [8, 32, 50]. Freedman and Drineas [25] extended this work and proved that a subclass of submodular boolean functions of order four or more can be minimized. It was also shown that multi-label CRFs with convex energy functions of order two can be minimized in polynomial time [37, 88]. However, it has not been known what the analogue of this is for higher order cliques. We aim to study this in the chapter.

Most labelling problems in computer vision involve multi-label MRFs or CRFs [18,

[1]Clique size in a CRF corresponds to order of the energy function.

86]. Furthermore, the use of higher order clique structures has proved beneficial [43,65,79] for solving certain computer vision problems. However, efficient st-mincut based algorithms used for minimizing submodular second order boolean functions are not directly applicable to these functions. Our work overcomes this restriction by showing how we can transform some submodular multi-label higher order functions to submodular boolean functions, thus enabling their exact minimization. Before proceeding further, we briefly introduce our notation for denoting different classes of energy functions. Let $\mathcal{F}_s^k$ and $\bar{\mathcal{F}}_s^k$ denote the class of submodular and non-submodular boolean energy functions of order k respectively. Similarly, let $\mathcal{M}_s^k$ and $\bar{\mathcal{M}}_s^k$ denote the class of submodular and non-submodular multi-label energy functions of order k respectively.

A generic transformation framework. The basic idea of our framework is to use two or more boolean variables to encode the states of a single multi-label variable. While doing this, we have to ensure that the minimum cost labelling of the boolean problem also encodes the minimum cost labelling of the multi-label energy function. There are several possible ways to encode a multi-label variable using boolean variables. In the rest of the chapter, we use the term *encoding* to refer to the mapping between the labellings of a multi-label variable and its corresponding binary variables. The term *transformation* refers to the conversion of multi-label energy functions to functions of binary variables.

It is important to study different transformations because the choice of transformation dictates the size of the resulting boolean function, and the class of multi-label functions that can be transformed to $\mathcal{F}_s^2$. For example, Ishikawa [37] described a transformation that used l boolean variables to encode a single l-label variable. Using this transformation pairwise convex functions of the difference of labels, which is a subclass of $\mathcal{M}_s^2$, can be transformed to $\mathcal{F}_s^2$. Later, Schlesinger and Flach [88] gave a concise definition of submodularity for (ordered) multi-label functions, and used $l-1$ boolean variables to transform any function in $\mathcal{M}_s^2$ to $\mathcal{F}_s^2$. In this chapter, we study the space of all possible transformations and find the subclasses of multi-label functions that they can transform to $\mathcal{F}_s^2$. In other words, the transformations we develop will lead to submodular boolean functions under some constraints. These constraints will serve to characterize the class of

$\mathcal{M}_k$ that can be minimized exactly.

The main novelties of our work are as follows:

- A principled framework for transforming $\mathcal{M}_s^k$ functions to $\mathcal{F}_s^2$.
- The identification of constraints that enable the transformation of $\mathcal{M}_s^k$ of any order into $\mathcal{F}_s^2$ in polynomial time.
- The result that there exists no polynomial transformation from submodular multi-label functions of order four or more ($\mathcal{M}_s^{k\geq 4}$) to submodular boolean second order functions ($\mathcal{F}_s^2$).
- The use of higher order functions to improve the performance of single view 3D reconstruction algorithms [36].

4.1.1 Outline of the Chapter

In section 4.2 we describe the basic theory of pseudo-boolean optimization and its relation to minimizing multi-label higher order functions with st-MINCUT algorithms. The problem statement is formalized in section 4.3. Section 4.4 shows how to encode multi-label variables using boolean (or binary) ones. A characterization of multi-label higher order functions that can be transformed to $\mathcal{F}_s^2$ in polynomial time is given in section 4.5. We describe the single view 3D reconstruction problem, and provide details of our solution in section 4.6. In this section we also present a comparison with the work of [36]. Other potential applications of our work and directions for future research are discussed in section 4.7.

4.2 Notation and Preliminaries

Let $\mathbb{B}$ denote the boolean set $\{0, 1\}$, and $\mathbb{R}$ the set of reals. Let the vector $\mathbf{x} = (x_1, ..., x_n) \in \mathbb{B}^n$, and $\mathcal{V} = \{1, 2, ..., n\}$, be the set of all boolean variables and their indices respectively. A pseudo-boolean function $f : \mathbb{B}^n \to \mathbb{R}$, is a function which takes a boolean vector as an argument and returns a real number. These functions can be uniquely represented using a multi-linear polynomial form [14].

The following is an example of a pseudo-boolean function:

$$f(x_1, x_2, x_3, x_4) = 2 - 4x_2x_4 + 7x_1x_2x_3, \tag{4.2.1}$$

containing four boolean variables. Another useful representation known as *posiform* involves the *complements* $(\bar{x}_1, ..., \bar{x}_n)$ of variables. Such a representation for the above example is:

$$\phi(x_1, x_2, x_3, x_4) = -2 + 4\bar{x}_4 + 4\bar{x}_2x_4 + 7x_1x_2x_3. \tag{4.2.2}$$

An important property of the posiform representation is that all the coefficients, except the constant, are non-negative [14].

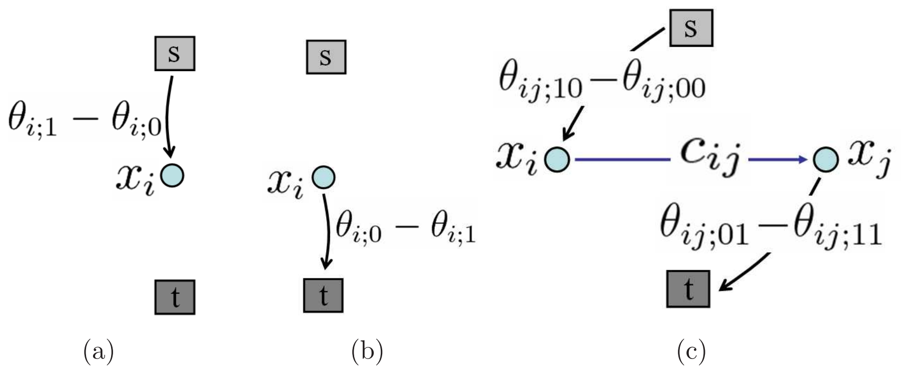
4.2.1 Graph Cuts for Energy Minimization

Here we recap some of the graph cuts and st-MINCUT concepts introduced in section 2.3.2. We denote the st-MINCUT graph with $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which has directed, non-negative edge weights and two special nodes, namely, the source s and the sink t representing labels 0 and 1 respectively. The st-MINCUT problem involves finding the st-cut with the minimum cost. Any $\mathcal{F}_s^2$ function can be minimized exactly by computing the st-MINCUT in an equivalent graph [50]. The key idea is to design a graph such that cuts in the graph correspond to labellings of the binary variables, with the cost of the cut equal to the cost of the labelling (plus a constant). We call this an *equivalent graph.*

Consider a second order boolean energy:

$$E_b(\mathbf{x}) = \sum_{i \in \mathcal{V}} E_b(x_i) + \sum_{(i,j) \in \mathcal{E}} E_b(x_i, x_j), \tag{4.2.3}$$

where $E_b(x_i)$ and $E_b(x_i, x_j)$ represent the first and second order terms of the binary energy function respectively. Let $\theta_{i;a}$ be the cost of assignment $x_i = a$, and $\theta_{ij;ab}$ be the cost of the assignment $x_i = a, x_j = b$ $(a, b \in \mathbb{B})$. The graph constructed for minimizing a $\mathcal{F}_s^2$ function has a vertex i for each boolean random variable $x_i \in \mathbb{B}$. There is a mapping between st-cuts in the graph and label



s
$\theta_{i;1} - \theta_{i;0}$
x_i
t
(a)
s
x_i
$\theta_{i;0} - \theta_{i;1}$
t
(b)
s
$\theta_{ij;10} - \theta_{ij;00}$
x_i
c_{ij}
x_j
$\theta_{ij;01} - \theta_{ij;11}$
t
(c)

Figure 4.1: *Converting an energy minimization problem to an st-*MINCUT *problem [50]. (a) and (b) show how unary potentials are represented using edges in the graph, while (c) shows the same for submodular pairwise potentials.*

assignments. A node i in the source set implies $x_i = 0$, while i in the sink set implies $x_i = 1$. We now show how to create the equivalent graphs for functions belonging to the classes $\mathcal{F}_s^1$ and $\mathcal{F}_s^2$.

The class $\mathcal{F}_s^1$. The unary term $E_b(x_i)$ of the energy can be written as: $E_b(x_i) = \theta_{i;0}\bar{x}_i + \theta_{i;1}x_i$. If $\theta_{i;1} - \theta_{i;0} \geq 0$, we write the energy as: $E_b(x_i) = (\theta_{i;1} - \theta_{i;0})x_i + \theta_{i;0}$. The minimization of this energy is equivalent to finding the st-MINCUT in the graph shown in Fig. 4.1(a). Cutting the edge (s, i) is equivalent to the assignment $x_i = 1$. Similarly, if $\theta_{i;1} - \theta_{i;0} < 0$, we write the energy as $E_b(x_i) = (\theta_{i;0} - \theta_{i;1})\bar{x}_i + \theta_{i;1}$, and the corresponding graph is given in Fig. 4.1(b).

The class $\mathcal{F}_s^2$. The pairwise energy $E_b(x_i, x_j) = \theta_{ij;00}\bar{x}_i\bar{x}_j + \theta_{ij;01}\bar{x}_i x_j + \theta_{ij;10}x_i\bar{x}_j + \theta_{ij;11}x_i x_j$ can be written as: $E_b(x_i, x_j) = c_{ij}\bar{x}_i x_j + (\theta_{ij;10} - \theta_{ij;00})x_i + (\theta_{ij;10} - \theta_{ij;11})\bar{x}_j + \theta_{ij;00} + \theta_{ij;11} - \theta_{ij;10}$, where $c_{ij} = (\theta_{ij;01} + \theta_{ij;10} - \theta_{ij;00} - \theta_{ij;11})$. The equivalent graph construction is given in Fig. 4.1(c). Since our overall goal is to transform multi-label functions to $\mathcal{F}_s^2$, we do not focus on $\mathcal{F}_s^3$ and higher order functions [25, 50].

Multi-label functions. Let $\mathcal{G}_m = (\mathcal{V}_m, \mathcal{E}_m)$ be a directed graph with a set of vertices $\mathcal{V}_m = \{1, 2, ..., m\}$, and edges $\mathcal{E}_m$. Let y_i be a variable taking values in some discrete space $\mathcal{L} = \{1, 2, ..., l\}$, and let $\mathbf{y} = \{y_1, ..., y_m\}$. We use $\boldsymbol{\Theta}$ to denote the set of higher order potentials whose sum defines the energy function. The unary potential is denoted by $\Theta_{i;a}$, pairwise by $\Theta_{ij;ab}$, where $i, j \in \mathcal{V}_m$, and

$a, b \in \mathcal{L}$. Let $\mathbf{i} = i_1 i_2 ... i_k \in \mathcal{V}_m^k = \mathcal{V}_m \times \mathcal{V}_m ... \mathcal{V}_m$ (k times) and $\mathbf{a} = a_1 a_2 ... a_k \in \mathcal{L}^k = \mathcal{L} \times \mathcal{L} ... \mathcal{L}$ (k times). Under this notation, a k^{th} order energy function is written as:

$$E_m(\mathbf{y}) = \sum_{\mathbf{i} \in \mathcal{V}_m^k, \mathbf{a} \in \mathcal{L}^k} \Theta_{\mathbf{i};\mathbf{a}} \prod_{i \in \mathbf{i}, a \in \mathbf{a}} \delta(y_i, a), \tag{4.2.4}$$

where

$$\delta(y_i, a) = \begin{cases} 1 & \text{if } y_i = a, \\ 0 & \text{otherwise.} \end{cases} \tag{4.2.5}$$

4.2.2 Submodular functions

Submodular functions are set functions $f : 2^n \rightarrow \mathbb{R}$, satisfying the following condition:

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y), \tag{4.2.6}$$

where X and Y are subsets of the set $\mathcal{V}$, and $\cup$ and $\cap$ denote union and intersection of sets respectively. We briefly describe how the above definition of submodularity maps to functions of boolean variables [50]. A function of one boolean variable is always submodular. A function $\theta : \mathbb{B}^2 \rightarrow \mathbb{R}$ of two boolean variables $\{x_i, x_j\}$ is submodular if and only if:

$$\theta_{ij:00} + \theta_{ij:11} \leq \theta_{ij:01} + \theta_{ij:10}. \tag{4.2.7}$$

A function $\theta : \mathbb{B}^n \rightarrow R$, is submodular if and only if all its projections on 2 variables are submodular [14,50]. The submodularity conditions can be extended to multi-label variables. Let $\mathcal{L}$ be a completely ordered set, where between every pair of states l_1 and l_2, an ordering (above/below) is present. A function $\Theta : \mathcal{L}^2 \rightarrow R$, is submodular if:

$$\Theta_{ij:l_1 l_2} + \Theta_{ij:(l_1+1)(l_2+1)} \leq \Theta_{ij:(l_1+1)l_2} + \Theta_{ij:l_1(l_2+1)}, \tag{4.2.8}$$

for all l_1, l_2 [88]. Using the work of Schlesinger [87] on permuted submodular functions we can find an ordering (if it exists) for which the functions become submodular. Thus, we can work with a notion of submodularity of multi-label functions which is independent of the ordering of the labels. A function $\Theta : \mathcal{L}^m \rightarrow R$ is submodular if and only if all its projections on 2 variables are submodular [22].

4.3 Problem Statement

The main goal of this chapter is to obtain a boolean second order function $E_b(\mathbf{x})$, equivalent to a given multi-label higher order function $E_m(\mathbf{y})$, in polynomial time. The boolean function also needs to satisfy the following conditions:

- There is an encoding $\mathcal{T} : \mathcal{L}^{|\mathcal{V}_m|} \to \mathbb{B}^{|\mathcal{V}|}$ which is 1-1 between the feasible labellings of $\mathbf{x}$ and $\mathbf{y}$, and bijective between the set of optimal labellings of the boolean and multi-label variables.

- The minimum value of $E_m(\mathbf{y})$ over $\mathbf{y}$ is equal to the minimum value of $E_b(\mathbf{x})$ over $\mathbf{x}$:
$$\min_{\mathbf{x}} E_b(\mathbf{x}) = \min_{\mathbf{y}} E_m(\mathbf{y}). \tag{4.3.1}$$
The energy functions need not be equal at labellings that are not their respective minima.

We also want to answer the following questions:

1. What is the class of multi-label higher order functions for which we will always be able to find an equivalent $\mathcal{F}_s^2$ function? We characterize the class by finding the constraints on the potentials $\mathbf{\Theta}$ of the function.

2. How can the boolean function with the smallest number of variables be obtained?

We now summarize the three important steps in our algorithm, before proceeding to explain them in detail.

1. A second order pseudo-boolean function is constructed which enforces 1-1 mapping between the feasible labellings of $\mathbf{y}$ and $\mathbf{x}$ (See §4.4).

2. Encoding functions that can replace all occurrences of $\mathbf{y}$ in $E_m(\mathbf{y})$ using $\mathbf{x}$ are computed (See §4.5).

3. We transform the problem of minimizing the multi-label energy function into that of minimizing a $\mathcal{F}_s^2$ function (See §4.5).

For simplicity, we demonstrate our method on a specific 4-label energy function. The algorithm is presented as an interplay between graph constructions and transformation of energy functions. As studied in §4.2.1, both operations are closely related.

4.4 Boolean encoding for multi-label variables

In this section we propose a method to construct a second order pseudo-boolean function such that the labellings of the boolean variables have a 1-1 mapping with the labellings of the original multi-label variables. For example, in Fig. 4.2(a) we show a graph construction[2] to encode a 4-label variable y_1 using three boolean variables $\{x_1, x_2, x_3\}$. The encoding representing the change of variables is:

$$\{y_1 = 1\} \leftrightarrow \{x_1 = 1, x_2 = 1, x_3 = 1\}, \tag{4.4.1}$$

$$\{y_1 = 2\} \leftrightarrow \{x_1 = 0, x_2 = 1, x_3 = 1\}, \tag{4.4.2}$$

$$\{y_1 = 3\} \leftrightarrow \{x_1 = 0, x_2 = 0, x_3 = 1\}, \tag{4.4.3}$$

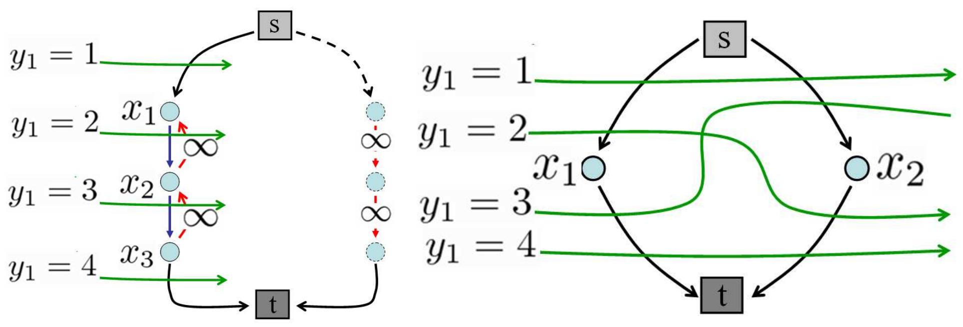
$$\{y_1 = 4\} \leftrightarrow \{x_1 = 0, x_2 = 0, x_3 = 0\}. \tag{4.4.4}$$

Since three binary variables can take eight (2^3) different labellings, the remaining four labellings ($2^3 - 4$) are not mapped to any labellings of y_1. In order to ensure a bijective encoding between the binary variables and the multi-label variable, these labellings need to be made infeasible. This can be achieved by assigning a very high cost to the unused labellings. In the above encoding the unused labellings are given by $x_1x_2x_3 = \{010, 101, 100, 110\}$. Thus, we have the following penalty term:

$$P(\mathbf{x}) = \lambda(\bar{x}_1 x_2 \bar{x}_3 + x_1 \bar{x}_2 x_3 + x_1 \bar{x}_2 \bar{x}_3 + x_1 x_2 \bar{x}_3), \tag{4.4.5}$$

where $\lambda \to \infty$. This can also be seen as using the following third order penalty

[2]This is sometimes referred to as the *battleship* construction.



(a) (b)

Figure 4.2: *(a) The battleship transformation [37, 88, 108]: The cuts in the graph are annotated by green arrows. Four possible cuts are shown and each cut corresponds to the assignment of one of the four labels to y_1. For example, if the edge (x_1, x_2) is cut then the labelling for $x_1x_2x_3$ is 011 and the corresponding labelling for y_1 is 2. Overall, the four labels of a multi-label variable $y_1 = \{1, 2, 3, 4\}$ are mapped to the labellings of three binary variables $x_1x_2x_3 = \{111, 011, 001, 000\}$. (b) The log transformation: The four labels of $y_1 = \{1, 2, 3, 4\}$ are mapped to the labellings (cuts) of two binary variables $x_1x_2 = \{11, 10, 01, 00\}$.*

function:

$$\begin{pmatrix} \phi_{123;000} & \phi_{123;001} \\ \phi_{123;010} & \phi_{123;011} \\ \phi_{123;100} & \phi_{123;101} \\ \phi_{123;110} & \phi_{123;111} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ \infty & 0 \\ \infty & \infty \\ \infty & 0 \end{pmatrix}. \tag{4.4.6}$$

It can be easily verified that the above function is submodular. It has four $(2^3 - 4)$ penalty terms (corresponding to ∞ values in (4.4.6)) to restrict the infeasible labellings. The penalty function in (4.4.5) can be simplified to:

$$P(\mathbf{x}) = \lambda x_1 \bar{x}_2 + \lambda x_2 \bar{x}_3, \tag{4.4.7}$$

using simple boolean algebra. The two pairwise terms in $P(\mathbf{x})$ correspond to the edges (x_2, x_1) and (x_3, x_2) with ∞ costs in Fig. 4.2(a).[3]

A natural question to ask would be whether a different encoding is possible for a 4-label problem. To address this question we consider Fig. 4.2(b), where two boolean variables are used to encode a 4-label problem. We refer to this graph construction as the *log* transformation, since it uses $\log(l)$ boolean nodes

[3]In practice, we do not need an edge with infinite cost, but some edge having a cost greater than the sum of all edge costs.

to encode an l-label variable. In this work, we chose a specific transformation to describe our algorithm. So from this point onwards, we will propose algorithms specific to the battleship transformation shown in Fig. 4.2(a). It can be shown that this transformation handles the most general class of energy functions [77].

4.5 Encoding Functions

Our overall goal is to transform a given multi-label higher order energy function into a boolean one. To do this, we need to define a boolean function which maps the labels of the multi-label variable to that of the encoding boolean variables. We refer to these functions as *encoding functions*. They enable us to replace multi-label variables in the energy function by boolean ones. More precisely, an *encoding* function is defined as $f_{y_1;a}(x_1, x_2, x_3) : \mathbb{B}^3 \to \mathbb{B}$, such that $f_{y_1;a}(x_1, x_2, x_3) = 1$, when $y_1 = a$, and 0 otherwise. The following example is shown to illustrate the key ideas.

Let us assume that the function $f_{y_1;a}(x_1, x_2, x_3)$ is linear.[4] We assume the following representation for the linear function using four unknown parameters c_0, c_1, c_2, and c_3:

$$f_{y_1;a} = c_0 + c_1x_1 + c_2x_2 + c_3x_3. \tag{4.5.1}$$

Returning to our example (4.4.1—4), the possible solutions for the triplet $x_1x_2x_3$ are $(111, 011, 001, 000)$. When $y_1 = 1$, $x_1x_2x_3 = 111$. This can be written as $f_{y_1;1}(x_1 = 1, x_2 = 1, x_3 = 1) = 1$ and $f_{y_1;1}(x_1, x_2, x_3) = 0$ for other values of x_1, x_2 and x_3. Since there are only four possible solutions for the boolean variables $x_1x_2x_3$, we obtain the following conditions:

$$\begin{aligned}
f_{y_1;1}(x_1 = 1, x_2 = 1, x_3 = 1) &= c_0 + c_1 + c_2 + c_3 = 1, \\
f_{y_1;1}(x_1 = 0, x_2 = 1, x_3 = 1) &= c_0 + c_2 + c_3 = 0, \\
f_{y_1;1}(x_1 = 0, x_2 = 0, x_3 = 1) &= c_0 + c_3 = 0, \\
f_{y_1;1}(x_1 = 0, x_2 = 0, x_3 = 0) &= c_0 = 0.
\end{aligned}$$

[4]The function $f_{y_1;a}$ need not always be linear, *e.g.* the log construction has a bilinear encoding function.

On solving the above linear system, we get $f_{y_1;1} = x_1$. Using the same approach we solve for $f_{y_1;2}$, $f_{y_1;3}$ and $f_{y_1;4}$.

$$\begin{pmatrix} \delta(y_1,1) \\ \delta(y_1,2) \\ \delta(y_1,3) \\ \delta(y_1,4) \end{pmatrix} = \begin{pmatrix} f_{y_1;1} \\ f_{y_1;2} \\ f_{y_1;3} \\ f_{y_1;4} \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 - x_1 \\ x_3 - x_2 \\ 1 - x_3 \end{pmatrix}. \tag{4.5.2}$$

With the encoding functions in place, we can finally address the energy transformation problem. The main idea is straightforward; the encoding functions are used to replace all occurrences of the multi-label variable in the energy function by boolean variables. This substitution produces a pseudo-boolean higher order function. We study this reduction and give a characterization of the class of multi-label higher order energy functions that can be transformed to $\mathcal{F}_s^2$, and thus be minimized exactly using graph cuts.

We first show that it is possible to transform all functions in class $\mathcal{M}_s^k$ to functions in $\mathcal{F}_s^2$, if $k \leq 2$. This is not a new result and follows from [87, 88]. We then go on to show that it is not possible to transform all functions in $\mathcal{M}_s^k$ to $\mathcal{F}_s^2$ in polynomial time when $k \geq 4$.

The class $\mathcal{M}_s^1$. We now show how to transform a first order energy function $E_m(\mathbf{y})$, involving a single 4-label variable y_1, to a first order boolean energy function $E_b(\mathbf{x})$, composed of three boolean variables $\mathbf{x} = \{x_1, x_2, x_3\}$. Let $\mathcal{L} = \{1, 2, 3, 4\}$. The energy $E_m(\mathbf{y})$ can be written as:

$$E_m(y) = \sum_{a \in \mathcal{L}} \Theta_{1;a} \delta(y_1, a). \tag{4.5.3}$$

We replace all occurrences of $\delta(y_1, a)$ using the corresponding boolean functions $f_{y_1;a}(x_1, x_2, x_3)$ given in (4.5.2). This results in an energy function that depends only on $\mathbf{x}$ as shown below:

$$E_b(\mathbf{x}) = \Theta_{1;1}x_1 + \Theta_{1;2}(x_2 - x_1) + \Theta_{1;3}(x_3 - x_2) + \Theta_{1;4}(1 - x_3). \tag{4.5.4}$$

Since the above energy function belongs to $\mathcal{F}_s^1$, all multi-label first order functions can be minimized exactly.

The class $\mathcal{M}_s^2$. Let y_1 and y_2 be two 4-label random variables in the following second order energy function:

$$E_m(\mathbf{y}) = \sum_{a,b\in\mathcal{L}} \Theta_{12;ab}\delta(y_1,a)\delta(y_2,b). \tag{4.5.5}$$

We transform this energy into a boolean energy function $E_b(\mathbf{x})$, involving triplets (x_1,x_2,x_3) and (x'_1,x'_2,x'_3) replacing y_1 and y_2 respectively. The encoding function $f_{i;a}$, given by (4.5.2), is used to replace $\delta(y_i,a)$, resulting in the following boolean energy function:

$$E_b(\mathbf{x}) = \sum_{i,j\in\{1,2,3\}} \alpha_{ij}x_ix'_j + L_1, \tag{4.5.6}$$

where $\alpha_{ij} = (\Theta_{12;ij} - \Theta_{12;(i+1)j} - \Theta_{12;i(j+1)} + \Theta_{12;(i+1)(j+1)})$, and L_1 stands for some first order terms. According to [25, 32], if the coefficients of all quadratic terms in a boolean second order energy function are non-positive, then the energy function is submodular. Thus, for the above energy function in equation (4.5.6) to be submodular, we need to ensure that $\alpha_{ij} \leq 0$, *i.e.*

$$\Theta_{12;ij} - \Theta_{12;(i+1)j} - \Theta_{12;i(j+1)} + \Theta_{12;(i+1)(j+1)} \leq 0. \tag{4.5.7}$$

Note that the above condition is nothing but the submodularity condition for second order multi-label functions (See (4.2.8)). Thus we prove that all submodular four-label second order functions $\mathcal{M}_s^2$ can be transformed to $\mathcal{F}_s^2$. Similarly, we can show that this approach generalizes to functions with more than four labels.

The class $\mathcal{M}_s^3$. Here we focus on transforming energy functions involving cliques of size three. Let y_1, y_2 and y_3 be three multi-label variables in a third order energy function $E_m(\mathbf{y})$, given by:

$$E_m(\mathbf{y}) = \sum_{a,b,c} \Theta_{123;abc}\delta(y_1,a)\delta(y_2,b)\delta(y_3,c). \tag{4.5.8}$$

α_{ijk}	$\Theta_{123;ijk} + \Theta_{123;(i+1)(j+1)k} - \Theta_{123;(i+1)jk} - \Theta_{123;i(j+1)k} - \Theta_{123;ij(k+1)} - \Theta_{123;(i+1)(j+1)(k+1)} + \Theta_{123;(i+1)j(k+1)} + \Theta_{123;i(j+1)(k+1)}$
α_{ij}	$\Theta_{123;ij4} + \Theta_{123;(i+1)(j+1)4} - \Theta_{123;(i+1)j4} - \Theta_{123;i(j+1)4}$
β_{ij}	$\Theta_{123;4ij} + \Theta_{123;4(i+1)(j+1)} - \Theta_{123;4(i+1)j} - \Theta_{123;4i(j+1)}$
γ_{ij}	$\Theta_{123;i4j} + \Theta_{123;(i+1)4(j+1)} - \Theta_{123;(i+1)4j} - \Theta_{123;i4(j+1)}$
α_i	$\Theta_{123;i44} - \Theta_{123;(i+1)44}$
β_i	$\Theta_{123;4i4} - \Theta_{123;4(i+1)4}$
γ_i	$\Theta_{123;44i} - \Theta_{123;44(i+1)}$
L_0	$\theta_{123;4,4,4}$

Table 4.1: *Coefficients in the third order binary energy function (equation (4.5.9)).*

We use three boolean triplets (x_1, x_2, x_3), (x'_1, x'_2, x'_3) and (x''_1, x''_2, x''_3) to encode y_1, y_2 and y_3 respectively. After replacing $\delta(y_i, a)$ with $f_{i;a}$ and applying algebraic transformations we can rewrite the energy function using boolean variables as:

$$\begin{aligned} E(\mathbf{x}) \quad = \quad & \sum_{i,j,k\in\{1,2,3\}} \alpha_{ijk} x_i x'_j x''_k + \sum_{i,j\in\{1,2,3\}} \alpha_{ij} x_i x'_j + \sum_{i,j\in\{1,2,3\}} \beta_{ij} x'_i x''_j + \\ & \sum_{i,j\in\{1,2,3\}} \gamma_{ij} x_i x''_j + \sum_{i\in\{1,2,3\}} \alpha_i x_i + \sum_{i\in\{1,2,3\}} \beta_i x'_i + \\ & \sum_{i\in\{1,2,3\}} \gamma_i x''_i + L_0, \end{aligned} \qquad (4.5.9)$$

where the coefficients of the trilinear, bilinear and unary terms are functions of Θ, and are given in Table 4.1.

We observe that the transformed energy function $E(\mathbf{x})$, is of order three. We are interested in reducing this energy function to a second order one in order to minimize it using any st-mincut algorithm. To do this we will first transform the above function involving the sum of first order, second order and third order terms to a function involving only third order terms. We can always rewrite a first order term, such as x_1, as $x_1x_2 + x_1\bar{x}_2$ using simple boolean operations, where x_2 can be any variable other than x_1. Similarly, it is also possible to rewrite a second order term, such as x_1x_2, as $x_1x_2x_3 + x_1x_2\bar{x}_3$. Thus, the above function shown in (4.5.9) can also be written as a sum of third order binary functions:

$$E_b(\mathbf{x}) = \sum_{i,j,k\in\{1,2,3\},a,b,c\in\mathbb{B}} \theta_{ijk;abc}\delta(x_i, a)\delta(x'_j, b)\delta(x''_k, c), \qquad (4.5.10)$$

where $\delta(x_i, a) = x_i$, if $a = 1$, and $\delta(x_i, a) = \bar{x}_i$, if $a = 0$. Now, the function

(4.5.10) is a sum of third order terms of the form:

$$E_b(x_i, x'_j, x''_k) = \theta_{ijk;abc}\delta(x_i, a)\delta(x'_j, b)\delta(x''_k, c). \tag{4.5.11}$$

If each third order function belongs to the class $\mathcal{F}^3_s$, then we can obtain a graph construction for the whole function using the techniques described in [25, 50]. However, the individual third order functions in (4.5.10) need not be submodular. We use the following result from [22] to express $E_b(\mathbf{x})$ as a sum of submodular third order functions along with a constant.

Lemma 4.5.1 *Let G be a function in $\mathcal{M}^3_s$ defined as a sum of third order functions, as given below:*

$$G = \sum_{i,j,k \in \mathcal{V}_m, a,b,c \in \mathcal{L}} \Theta_{ijk;abc}\delta(y_i, a)\delta(y_j, b)\delta(y_k, c), \tag{4.5.12}$$

where each $\Theta_{ijk;abc}$ need not be in $\mathcal{M}^3_s$. Then, there exists an equivalent transformation satisfying the following condition:

$$\begin{aligned}\tilde{\Theta}_{ijk;abc}\delta(y_i, a)\delta(y_j, b)\delta(y_k, c) &= \Theta_{ijk;abc}\delta(y_i, a)\delta(y_j, b)\delta(y_k, c) + \\ &\quad \sum_{l,m \in \{i,j,k\}} \Psi_{lm;ab}\delta(y_l, a)\delta(y_m, b),\end{aligned} \tag{4.5.13}$$

where $\tilde{\Theta}_{ijk;abc}$ is a function in $\mathcal{M}^3_s$, and the sum of pairwise energy terms introduced during the transformation is a constant.

Using this lemma, we transform our energy function in (4.5.10) as shown below:

$$E(\mathbf{x}) = \sum_{i,j,k \in \{1,2,3\}; a,b,c \in \mathbb{B}} \tilde{\theta}_{ijk;abc}\delta(x_i, a)\delta(x_j, b)\delta(x_k, c) - \mathbf{\Psi}, \tag{4.5.14}$$

where $\mathbf{\Psi}$, a constant, refers to the spawned pairwise potentials during the transformation, and all $\tilde{\theta}$ belong to $\mathcal{F}^3_s$. Each individual $\tilde{\theta}_{ijk;abc}$ in $\mathcal{F}^3_s$ is now transformed to $\mathcal{F}^2_s$ using the method given in [25, 50]. Note that the above transformation is only possible when the original multi-label energy function, $E_b(\mathbf{x})$ in (4.5.10), belongs to $\mathcal{M}^3_s$. We now provide the conditions which will ensure this. In order to do this, we first explain the notion of submodularity using the concept of derivatives [22]. The derivative of a k^{th} order function $\Theta_{\mathbf{i};\mathbf{a}}$ with respect to a variable y_j is given by:[5]

$$\Delta_j \Theta_{\mathbf{i};\mathbf{a}} = \begin{cases} \Theta_{\mathbf{i};\mathbf{a}} - \Theta_{\mathbf{i};a_1 \ldots a_{j-1}(a_j-1)a_{j+1}\ldots a_k} & a_j > 1, \\ 0 & a_j = 1. \end{cases}$$

Derivatives can also be obtained with respect to several variables as shown below:

$$\Delta_{\mathbf{j}} \Theta_{\mathbf{i};\mathbf{a}} = \Delta_{j_1} \ldots \Delta_{j_k} \Theta_{\mathbf{i};\mathbf{a}}, \quad j_1, \ldots, j_k \in \mathbf{j}. \tag{4.5.15}$$

The submodularity condition is equivalent to saying that the second derivative of $E(\mathbf{x})$ with respect to any two variables x_i and x_j is less than or equal to zero, for all values of the remaining variables [22]. Using this definition, the energy function in (4.5.10) is submodular, if the following condition is satisfied:

$$\alpha_{ij} + \sum_{k \in \{1,2,3\}, c_k \in \mathbb{B}} c_k \alpha_{ijk} \leq 0. \tag{4.5.16}$$

In summary, submodular multi-label third order functions $\mathcal{M}_s^3$ can be transformed to $\mathcal{F}_s^2$, if they satisfy the additional constraint (4.5.16).

The class $\mathcal{M}_s^k$. We now consider the problem of transforming fourth or higher order functions. We will show that not all functions in $\mathcal{M}_s^k$, $k \geq 4$, can be transformed to the class $\mathcal{F}_s^2$ in polynomial time. To prove this we need the following lemma.

Lemma 4.5.2 *The recognition of submodularity in quartic (degree 4) posiforms is co-NP-complete*[6] *[28].*

In other words, this lemma shows that it is a hard problem to say whether a general posiform, involving quartic or higher order terms, defines a submodular function or not.

[5]Recall that a_j specifies the label taken by variable y_j.
[6]A problem $\mathcal{X}$ is co-NP if and only if its complement $\bar{\mathcal{X}}$ is in NP.

Theorem 4.5.3 *There is no preserving transformation with respect to $\mathcal{F}^k$ ($\bar{\mathcal{F}}_s^k \cup \mathcal{F}_s^k$) for $k \geq 4$, which works in polynomial time*[7] *unless $P = NP$.*

We say that a transformation $\mathcal{T}_p : \mathcal{F}^k \longrightarrow \mathcal{F}^2$ is a *preserving* transformation, if it satisfies the following conditions:

- If $f \in \mathcal{F}_s^k$, then $\mathcal{T}(f) \in \mathcal{F}_s^2$.
- If $f \notin \mathcal{F}_s^k$, then $\mathcal{T}(f) \notin \mathcal{F}_s^2$.

Proof of Th. 4.5.3: If such a transformation exists, we can transform any function in $\mathcal{F}^k$ to $\mathcal{F}^2$. Since submodularity can be checked in $\mathcal{F}^2$ in polynomial time[8], this gives a way to check whether any function in $\mathcal{F}^k$ is submodular or not in polynomial time, which is in contradiction with the Lemma 4.5.2. ∎

The above theorem states that it is not possible to transform all functions in $\mathcal{M}_s^k$ to $\mathcal{F}_s^2$ in polynomial time. However, we show that a subclass of $\mathcal{M}_s^k$ can still be transformed to $\mathcal{F}_s^2$.

Characterizing $\mathcal{F}_s^2$-transformable $\mathcal{M}^k$ functions: We will now characterize some $\mathcal{M}^k$ functions that can be transformed to $\mathcal{F}_s^2$ function in polynomial time. The characterization will be specified by a set of constraints on the potentials of the multi-label higher order functions. We will refer to these constraints as $\xi \leq 0$. Using the derivative definition of submodularity [22], the constraints $\xi \leq 0$ that will enable us to transform $\mathcal{M}_s^k$ functions to $\mathcal{F}_s^2$ functions are:

$$\Delta_{\mathbf{j}}\Theta_{\mathbf{i};\mathbf{a}} \leq 0, \quad \mathbf{j} \subseteq \mathbf{i}, \quad |\mathbf{j}| \geq 2. \tag{4.5.17}$$

For illustrative purposes we now present the graph construction for functions belonging to a subclass of the $\mathcal{M}_s^k$ family. The functions belonging to this subclass have the form:

$$\Theta_{\mathbf{k};\mathbf{i}} = \begin{cases} \alpha & \exists i \in \mathbf{i} : y_i < l_i, \\ 0 & \text{otherwise.} \end{cases} \tag{4.5.18}$$

[7]We say that a transformation works in polynomial time when we can compute a second order multi-linear polynomial expression for $\mathcal{T}_s(f)$ in $O(n^k)$ time, where n is the number of variables, and k is the order of the boolean function.

[8]The recognition of submodularity in $\mathcal{F}^2$ can be done in polynomial time by checking the coefficients of the quadratic terms [32].

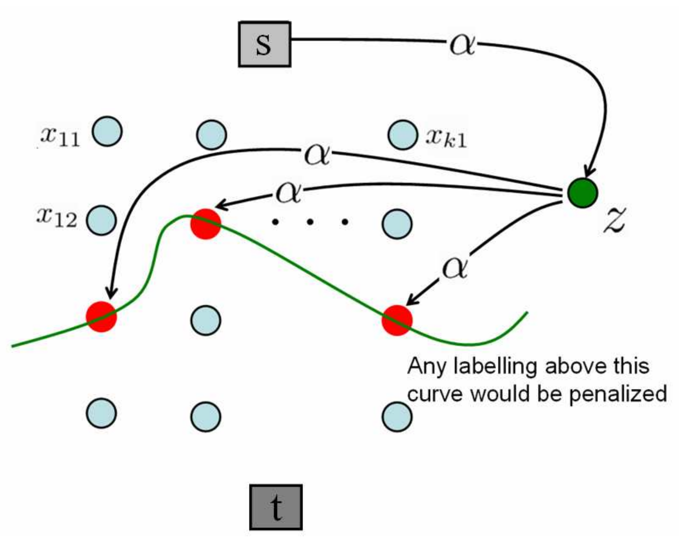


Figure 4.3: *The graph construction for characterizing a general k^{th} order multi-label energy function. The variable z is an auxiliary node that is connected to k boolean nodes and the source with the same edge cost α. As a result if all the k boolean nodes take the label* 0 *then the cost of the cut is 0. In all other cases there is a uniform cost of α. This can be seen as a generalization of the graph construction given in [50].*

The corresponding graph construction is shown in Fig. 4.3. We connect a set of encoding variables to an auxiliary node z, and connect z to the source node s with edges having the same cost α. It is important to observe the functionality of z: for a group of variables $y_j, j \in \mathbf{i}$, if any variable y_i takes a label less than a specified label l_i there is a penalty of α. Our method can automatically find the required auxiliary nodes and various edge costs for the graph needed to minimize any $\mathcal{M}_s^k$-function that satisfy constraints (4.5.17).

4.6 Application: Single View Reconstruction

We now show how the higher order functions characterized in the previous section can be used to improve single view reconstruction results. Given a 2D image of a scene, the goal is to recover a theatre stage representation containing major surfaces and their geometrical relationships to each other. Hoiem *et al.* [36] formulated this as a classification problem, where every pixel in the image is assigned one of the three labels, namely, support (surfaces that lie parallel to the

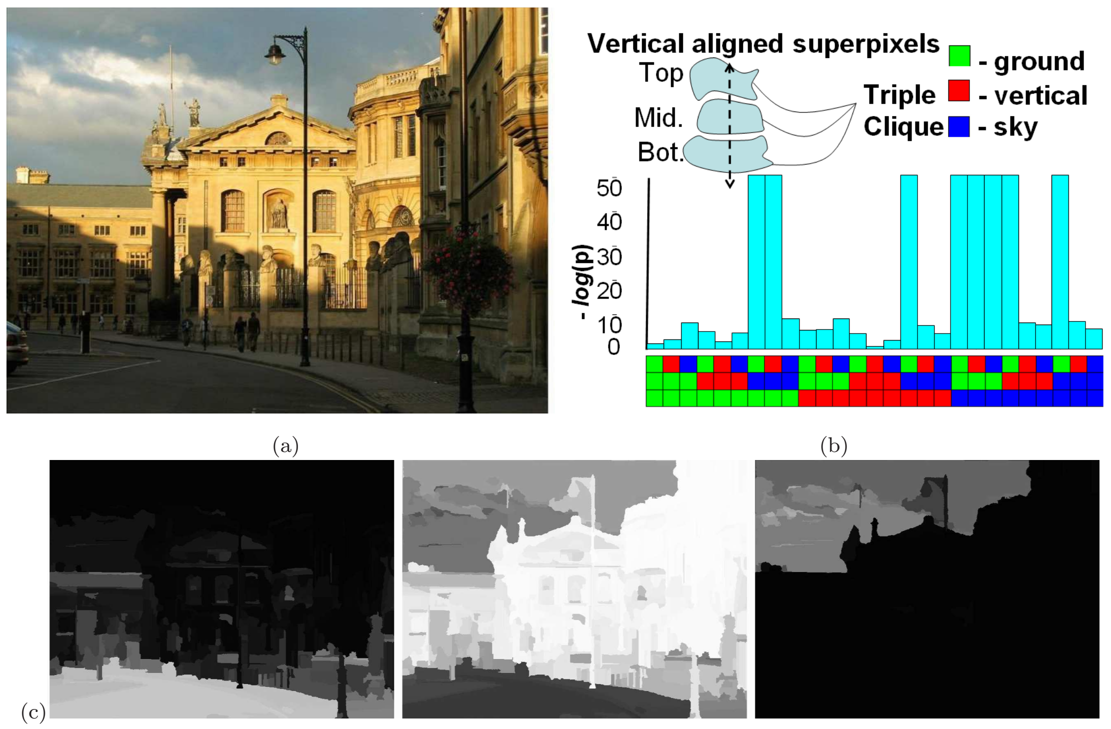


Figure 4.4: *(a) Original image. (b) Triplets of vertically aligned superpixels are chosen from the superpixel segmented images. The labellings for individual triplet combinations are studied from several ground truth images. Negative log-likelihoods are computed for each of these triplets and used as third order priors in the labelling problem, formulated as an energy minimization task. (c) The three columns, from left to right, show the unary likelihood images of ground, vertical and sky respectively.*

ground plane), vertical (surfaces that *rise* from the ground plane), and sky. They obtained impressive results by learning appearance based models of the three classes. Their method works as follows. The given image is first segmented into superpixels [20] (see second column of Fig. 4.5), which provide spatial support for computing features like texture filter responses and vanishing points. Using boosted decision tree classifiers, geometrical likelihoods are computed for individual superpixels (cf. Fig. 4.4). The final geometrical labelling is achieved using these likelihoods along with pairwise smoothness priors in an energy minimization framework.

In this work, we focus on improving the results in [36] using priors obtained from natural statistics. Such priors can only be imposed through CRFs with higher order cliques [65, 79]. The superpixels extracted from the image act as

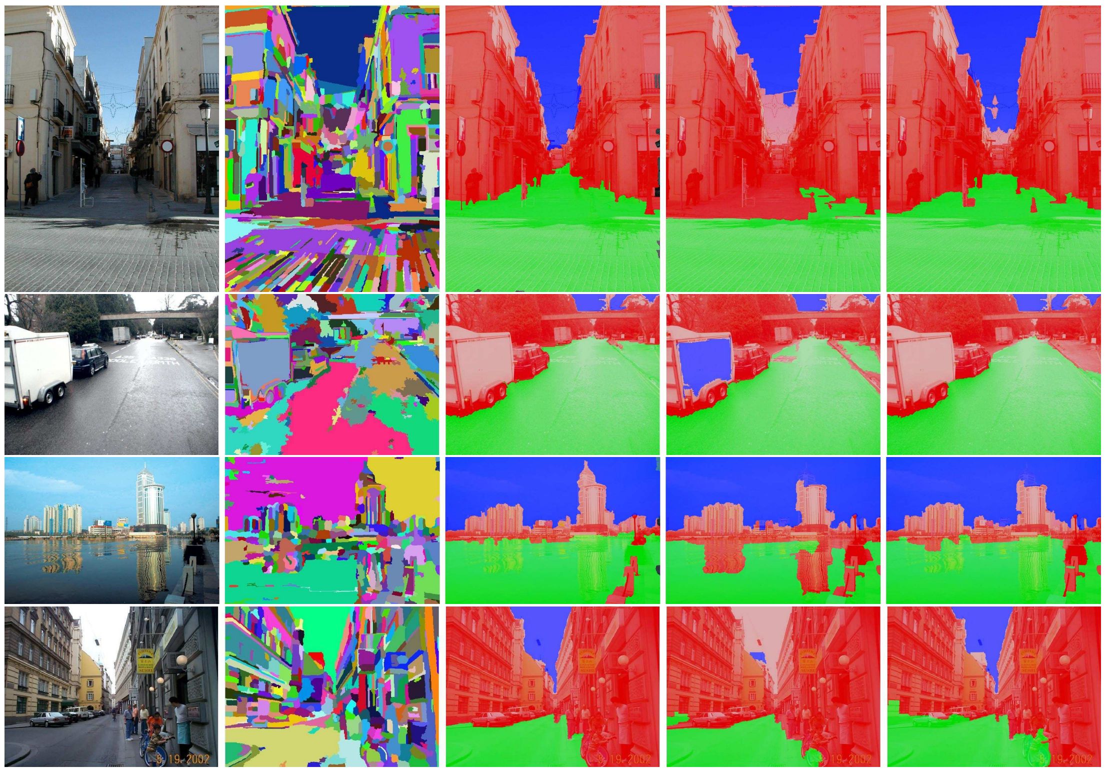


Figure 4.5: *Original image, superpixel segmentation, ground truth labelling, results from [36] and our results are shown (left to right). Street, highway, buildings and road are the images in rows 1 to 4 respectively. (Best viewed in colour)*

nodes (variables) in a higher order CRF. The most probable labelling of the superpixels is found by minimizing an equivalent energy function. We minimize a third order three-label energy function, where the three labels for each superpixel correspond to ground, vertical and sky.

The unary likelihoods $\theta_{i;a}$ of the energy function are computed using boosted decision tree classifiers.[9] Motivated by the work of [114], we compute the second and third order energy terms using natural statistics. Yang and Purves [114] study the distribution of geometrical features like size, shape and depth of planar surfaces, from a large training database. Using a similar approach, the second order terms are computed by learning the statistics of all neighbouring superpixel pairs in the training dataset.

As the images are generally taken by people standing on the ground, with the optical axis approximately parallel to the ground, there is a natural ordering of the superpixels labels in the vertical direction. To capture this ordering, we study the

[9] `http://www.cs.cmu.edu/~dhoiem/projects/software.html`.

Image	Results of [36]	Our method
street	20.78	5.82
highway	19.47	7.32
buildings	31.94	13.36
road	18.52	10.82
college	29.47	13.26

Table 4.2: *Here we show the error percentages obtained by our method in comparison to [36]. It is computed using the ground truth provided in the dataset. Note that our method significantly improves the accuracy of single view reconstruction.*

distribution of the labelling of vertically-aligned superpixel triplets from several ground truth labelled images. These statistics, in the form of negative log likelihoods, are shown in Fig. 4.4(b). The likelihoods are directly used as the higher order potential $\theta_{ijk;abc}$ in the energy function. As an example, to see the effectiveness of natural statistics, consider the cost of the triplet labelling [Top:Ground, Middle:Vertical, Bottom:Sky] from the figure. Given the label ordering, this configuration is unlikely to occur naturally, and thus has a high cost. We use our algorithm explained in §4.5 to construct the equivalent boolean graph. A simple truncation method is used to remove the negative edges in the graph [83].

We observed significant improvement over the results of [36], as shown in Fig. 4.5. The labelling accuracy is summarized in Table 4.2. The accuracy is reported in terms of the misclassification of individual pixels in the image. In Fig. 4.5 we show the original image, superpixel segmentation, results using only pairwise clique potentials, and our results using higher order clique potentials. In the street image shown in the first row of Fig. 4.5, the ground between the two buildings is incorrectly labelled as vertical, when only pairwise smoothness prior is used. On the other hand, the usage of higher order priors results in the correct labelling. The major advantage comes from the ability to impose priors based on natural statistics. For example, in the second row of Fig. 4.5, unary potentials favour the labelling sky for the van due to its high similarity to the 'sky' region. However, our method using priors learned from natural statistics obtains the correct labelling.

4.7 Summary

We presented a principled framework to transform a certain class of multi-label higher order functions to submodular second order boolean functions, which can be minimized exactly using graph cuts. Our key idea is to use two or more boolean variables to encode the states of a single multi-label variable. Our transformations can be used for other vision problems, such as stereo [37], panoramic stitching [69], image restoration. Recently, the transformation proposed by [37] was used to develop a new move algorithm [108]. Similar techniques can be proposed for the transformations proposed in our work. Our framework can also transform any higher order multi-label function, for instance, potentials learnt using the fields of experts model [79], to a boolean second order function. If the resulting second order boolean energy function is non-submodular, then we can use QPBO techniques [14]. The main contributions of this chapter are:

1. A principled way to incorporate natural image statistics into the single view reconstruction problem, and to show that they can be solved efficiently.

2. Demonstrating that our novel priors provide a significant improvement.

3. Presenting the constraints that enable the transformation of $\mathcal{M}_s^k$ into $\mathcal{F}_s^2$ in polynomial time, and thus extending the subclass of st-MINCUT-solvable submodular energy functions.

Chapter 5

Efficient Piecewise Parameter Learning

In the previous chapters we have addressed the problem of inference in Conditional Random Field models, assuming that the model parameters are given or set empirically. This chapter deals with the problem of learning the model parameters efficiently from training data. Although several methods have been proposed to deal with this problem, they suffer from various drawbacks. Learning the parameters involves computing the partition function, which is intractable for several low-level vision problems. To overcome this, state-of-the-art structured learning methods frame the problem as one of large margin estimation. Iterative solutions have been proposed to solve the resulting convex optimization problem. Each iteration involves solving an inference problem over all the labels, which limits the efficiency of these structured methods. In this chapter we present an efficient large margin piecewise learning method which is widely applicable. We show how the resulting optimization problem can be reduced to an equivalent convex problem with a small number of constraints, and solve it using an efficient scheme.

5.1 Introduction

A Conditional Random Field (CRF) is defined over a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes a set of vertices and $\mathcal{E}$ is the set of edges, which specifies a pairwise relationship between the vertices.[1] The vertices represent discrete random variables $\mathbf{X} = \{X_1, \cdots, X_N\}$. A labelling of a CRF corresponds to a classification of the vertices by assigning a label to each vertex (variable) from a set of labels $\mathcal{L} = \{1, \cdots, K\}$. In other words, a labelling is specified by a binary vector $\mathbf{x} = \{x_{1:1}, \cdots, x_{1:K}, x_{2:1} \cdots, x_{N:K}\}$, where N is the number of vertices, *i.e.* $|\mathcal{V}| = N$. Each binary indicator variable $x_{i:k} = 1$, if the corresponding random variable x_i takes the label $k \in \mathcal{L}$, and $x_{i:k} = 0$ otherwise. Also, $\sum_k x_{i:k} = 1, \forall i$. In the context of the vision problems we have seen so far, the vertices correspond to image pixels, and the labels can be image segments, disparity, object categories, etc. Given some observed data (denoted by $\mathbf{D}$), a CRF models the conditional

[1]Note that we have assumed a pairwise CRF. However, this assumption is not restrictive since any CRF can be converted to an equivalent pairwise CRF, *e.g.* using a method similar to the one described in [116], and efficient inference algorithms are available for many such CRFs [63].

probability of a labelling $\mathbf{x}$ as follows[2]:

$$\Pr(\mathbf{x}|\mathbf{D},\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{\substack{i\in\mathcal{V}\\ k\in\mathcal{L}}} \exp(x_{i:k}\boldsymbol{\theta}_k^\top h_i(\mathbf{D})) \prod_{\substack{(i,j)\in\mathcal{E}\\ k,l\in\mathcal{L}}} \exp(x_{i:k}x_{j:l}\boldsymbol{\theta}_{kl}^\top \nu_{ij}(\mathbf{D})), \quad (5.1.1)$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_k, \boldsymbol{\theta}_{kl}) \in \mathcal{R}^{d\times 1}$ are the parameters of the CRF. The vectors $h_i(\mathbf{D})$ and $\nu_{ij}(\mathbf{D})$ represent features for the vertex $i \in \mathcal{V}$ and the edge $(i,j) \in \mathcal{E}$ respectively. The unary potential $\exp(x_{i:k}\boldsymbol{\theta}_k^\top h_i(\mathbf{D}))$ denotes the cost of the assignment $X_i = k$, while the pairwise potential $\exp(x_{i:k}x_{j:l}\boldsymbol{\theta}_{kl}^\top \nu_{ij}(\mathbf{D}))$ denotes the cost of the assignment: $X_i = k$ and $X_j = l$. The normalizing factor $Z(\boldsymbol{\theta})$ given by:

$$Z(\boldsymbol{\theta}) = \sum_{\mathbf{y}'\in\mathcal{L}^N} \prod_{\substack{i\in\mathcal{V}\\ k\in\mathcal{L}}} \exp(x'_{i:k}\boldsymbol{\theta}_k^\top h_i(\mathbf{D})) \prod_{\substack{(i,j)\in\mathcal{E}\\ k,l\in\mathcal{L}}} \exp(x'_{i:k}x'_{j:l}\boldsymbol{\theta}_{kl}^\top \nu_{ij}(\mathbf{D})), \quad (5.1.2)$$

is the partition function. When using a CRF model (with known priors), there are two main issues that need to be addressed: (i) How to set the value of the parameters $\boldsymbol{\theta}$; and (ii) How to perform inference in order to obtain the optimal labelling, *i.e.* the labelling with the maximum conditional probability $\Pr(\mathbf{x}|\mathbf{D},\boldsymbol{\theta})$. The latter issue has received great attention and several inference algorithms have been proposed in the literature (for an overview, see [99]). Our work described in the previous chapters also addresses the inference problem. However, parameter estimation in a CRF model still remains a challenging problem, with considerable progress being made in the past few years.

Consider the partition function in (5.1.2), which contains a sum over the entire label space $\mathcal{L}^N$. To estimate the cost of computing the partition function, let us assume a CRF defined over a 300×200 image with $|\mathcal{L}| = 10$. In this case, the partition function is a sum of 10^{60000} terms, and its computation is clearly intractable. Hence, methods for estimating parameters of a CRF model must be designed to overcome this issue. Based on the way in which the partition function is handled, recent parameter learning methods can be broadly classified into three categories – maximum likelihood based methods [62, 84, 97], large margin based approaches [71, 100, 102], and other iterative methods [90, 117].

[2]Using the notation of [5].

Owing to the intractability of computing the partition function $Z(\boldsymbol{\theta})$ for computer vision applications, maximum likelihood based methods resort to using approximations, such as pseudo-likelihood [62], some form of local training [97], mode of the model distribution [84], or sampling [80]. While these likelihood approximation methods have shown encouraging results, they can lead to poor accuracy due to noisy estimates, as noted in [84,97]. Methods using a large margin approach pose parameter estimation as a convex optimization problem.[3] The convex problem is solved iteratively, and each iteration involves performing inference for every training image, which can be computationally expensive. By restricting themselves to a subset of random field models, some methods [100, 102] provide efficient solutions. The method proposed by Taskar *et al.* [102] uses approximate inference for multi-label problems, and exact inference for binary labelling problems in each iteration. Szummer *et al.* [100] use dynamic graph cuts [46] to perform inference in successive iterations efficiently.[4]

Another large margin approach [71] uses the structured output regression formulation proposed by Tsochantaridis *et al.* [105]. The algorithm employs a cutting-plane method to solve the quadratic optimization algorithm. The model parameters are updated using the most violated constraint (in this case, the labelling with the smallest cost value) in every iteration. Finding the exact most violated constraint is not tractable for random fields commonly encountered in computer vision, thus approximation algorithms are used. Other iterative based methods are either limited to CRFs with a few hundred nodes, thus impractical for the labelling problems we consider [90], or require an initial model with pre-set parameters [117].

In summary, previous methods can lead to poor accuracy due to approximations, or are restricted to a subset of random field models. We aim to address these issues in this chapter. To obtain an efficient and accurate learning scheme, we decompose the random field into tree-structured graphs, where each graph comprises of variable X_i and its corresponding Markov blanket, which is the set

[3]Large margin based parameter learning approaches eliminate the partition function by considering the gain of the true labels over any other labelling. We will discuss this in more detail in §5.2.2.

[4]Note that this work can extended to a larger class of energy functions using our efficient dynamic α-expansion (§3.3.1).

of its neighbours. This decomposition results in an optimization problem with a large number of constraints. We reduce this problem to an equivalent convex problem with a small number of constraints, similar to the approach of [60]. An efficient method to solve it using stochastic gradient descent is then proposed. One of the main advantages of our method is the ease of training, as demonstrated in the sections to follow.

5.1.1 Outline of the Chapter

In section 5.2 we formulate the parameter learning problem. We also describe two methods – pseudo-likelihood and max-margin learning – related to our work. Section 5.3 explains our piecewise large margin approach for parameter learning. Details of the optimization problem and the gradient descent approach are also given here. Implementation details and experimental results on the man-made structure [62] and Middlebury-2005 [84] datasets are shown in section 5.4. In this section we also present a comparison with other parameter learning methods. Section 5.4.3 presents a few generalizations of our model. Concluding remarks are provided in section 5.5.

5.2 Preliminaries

We begin by formulating the CRF parameter estimation problem. The unary and pairwise potentials are given by $\exp(x_{i:k}\boldsymbol{\theta}_k^\top h_i(\mathbf{D}))$ and $\exp(x_{i:k}x_{j:l}\boldsymbol{\theta}_{kl}^\top \nu_{ij}(\mathbf{D}))$ respectively (from (5.1.1)). The unary and pairwise feature vectors ($h_i(\mathbf{D})$ and $\nu_{ij}(\mathbf{D})$) can be defined in many ways. For example, in case of the image segmentation problem, the unary feature vector of a vertex can be composed of functions of the intensity, colour and texture, while the pairwise feature of an edge can be a difference of the feature vectors of the two vertices the edge connects.

Given a set of training data $\mathbf{D} = \{\mathbf{D}^m, m = 1, \cdots, M\}$, along with their ground truth labels $\mathbf{X} = \{\mathbf{x}^m, m = 1, \cdots, M\}$, the problem of parameter estimation is to obtain a value for the parameter $\boldsymbol{\theta}$, such that the model assigns a high probability to the correct labelling and a low one to all possible incorrect labellings. In

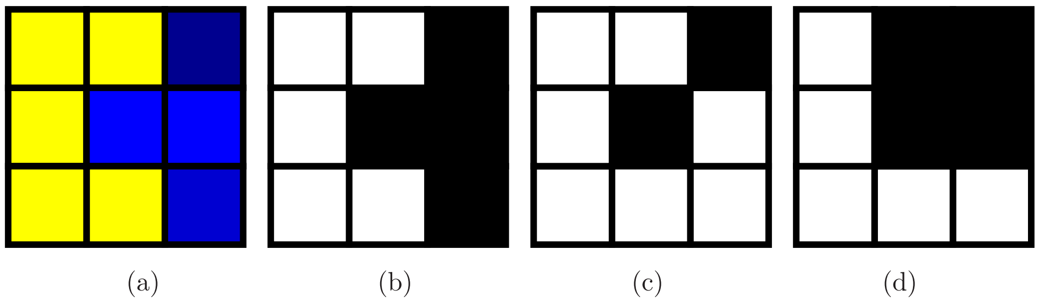


Figure 5.1: *We consider a binary image segmentation problem in this example. (a) Toy example of a 3×3 image, (b) Ground truth labelling showing the two segments – foreground (white) and background (black), and (c,d) Two incorrect segmentations from the set of possible $2^9 - 1$ segmentations. The training data consists of images, their ground truth labels, and all the possible incorrect labellings.*

the context of foreground-background segmentation problem, an element of the training set, $\mathbf{D}^m$, corresponds to an image, and the ground truth labels contain binary values representing foreground or background at each pixel. The model is learnt such that we obtain a high probability to the correct segmentation and a low one to all other possible segmentations. Fig. 5.1 shows a toy example of a 3×3 image grid. Here, we consider a binary image segmentation problem and illustrate a training image, its ground truth segmentation, and a few possible incorrect segmentations. The image and its ground truth segmentation is referred to as the positive training example, while the image and an incorrect segmentation is the negative training example. Note that the number of negative examples is exponentially large. Pseudo-likelihood and Max-margin learning are two popular methods to learn the parameters in this setting.

5.2.1 Pseudo-likelihood

The maximum likelihood estimate of the parameters $\hat{\boldsymbol{\theta}}$ (using equation (5.1.1)) is given by:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \sum_{m=1}^{M} \sum_{\substack{i \in \mathcal{V} \\ k \in \mathcal{L}}} x_{i:k}^m \boldsymbol{\theta}_k^\top h_i(\mathbf{D}^m) + \sum_{\substack{(i,j) \in \mathcal{E} \\ k,l \in \mathcal{L}}} x_{i:k}^m x_{j:l}^m \boldsymbol{\theta}_{kl}^\top \nu_{ij}(\mathbf{D}^m) - \log Z^m(\boldsymbol{\theta}), \quad (5.2.1)$$

where m indexes over the training images and M denotes the number of training images. Solving this estimation problem for loopy random fields commonly encountered in computer vision is intractable, as discussed in §5.1.

A common approach to overcome this issue is the use of pseudo-likelihood [7] to approximate the likelihood [12, 62]. The estimation problem now becomes:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \sum_{m=1}^{M} \sum_{i \in \mathcal{V}} PL(i, \mathbf{D}^m). \tag{5.2.2}$$

Here $PL(\cdot)$ is the pseudo-likelihood, and is given by:

$$PL(i, \mathbf{D}^m) = \sum_{k \in \mathcal{L}} x_{i:k}^m \boldsymbol{\theta}_k^\top h_i(\mathbf{D}^m) + \sum_{\substack{j \in \mathcal{N}_i \\ k,l \in \mathcal{L}}} x_{i:k}^m x_{j:l}^m \boldsymbol{\theta}_{kl}^\top \nu_{ij}(\mathbf{D}^m) - z_i^m + b, \tag{5.2.3}$$

where

$$z_i^m = \log \sum_{x_i^m \in \mathcal{L}} \prod_{k \in \mathcal{L}} \exp(x_{i:k}^m \boldsymbol{\theta}_k^\top h_i(\mathbf{D}^m)) \prod_{\substack{j \in \mathcal{N}_i \\ k,l \in \mathcal{L}}} x_{i:k}^m x_{j:l}^m \boldsymbol{\theta}_{kl}^\top \nu_{ij}(\mathbf{D}^m), \tag{5.2.4}$$

is the local partition function, b is a constant, and $\mathcal{N}_i$ is the Markov blanket at vertex i, *i.e.* the set of its neighbours in the random field model. For example, in the 4-neighbourhood case used for CRF based image segmentation, the Markov blanket of a pixel i is the set of 4 pixels—above, below, left of, and right of the pixel. This problem can be solved by gradient-descent like approaches [62] or auto-regression [12]. One of the main advantages of using pseudo-likelihood is the asymptotic guarantee (*i.e.* as the size of the data tends to infinity) that its maximum matches that of the original likelihood. However, parameter learning methods using pseudo-likelihood can lead to poor accuracy due to noisy estimates, as noted in [84, 97].

Another approach to approximate the likelihood estimation in (5.2.1) is to use the piecewise pseudo-likelihood (PWPL) model proposed by Sutton and McCallum [97]. Here, the likelihood is conditioned on all the variables in the factor graph associated with the variable. Figure 5.2 illustrates the difference between PWPL and pseudo-likelihood models. They show interesting results on linear-chain CRFs. However, it is not clear if this method generalizes to large random

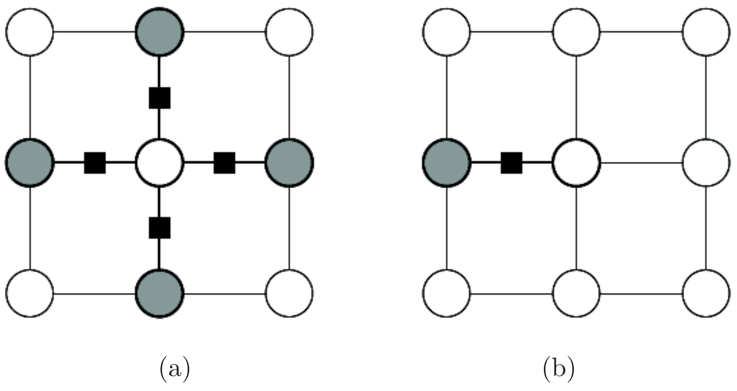


Figure 5.2: *Difference between pseudo-likelihood (*PL*), in (a), and piecewise pseudo-likelihood [97] (*PWPL*), in (b) is shown here. In* PL*, a variable is conditioned on all its neighbours in the Markov blanket, while in* PWPL *it is conditioned only on the neighbours within a single factor. Figure taken from [97].*

field problems, involving millions of variables, commonly occurring in computer vision.

5.2.2 Max-Margin Learning

Taskar *et al.* [102] proposed an alternative approach to learn the parameters of a random field discriminatively. Consider the logarithm of the probability in equation (5.1.1). It can be re-written, according to the notation in [5], as:

$$\log \Pr(\mathbf{x}|\mathbf{D}, \boldsymbol{\theta}) = \boldsymbol{\theta}\mathbf{F}\mathbf{x} - \log Z(\boldsymbol{\theta}), \tag{5.2.5}$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_k; \boldsymbol{\theta}_{kl})$ with the operator (;) denoting vector concatenation. The vector $\mathbf{x}$ contains the labels of all the variables in the random field, and the matrix $\mathbf{F}$ is composed of unary and pairwise features, *i.e.* $h_i(\mathbf{D})$ and $\nu_{ij}(\mathbf{D})$. Given a training image[5] $(\mathbf{D}, \hat{\mathbf{x}})$, the goal is to maximize the confidence in the true label assignment $\hat{\mathbf{x}}$ with respect to all other possible assignments $\mathbf{x} \neq \hat{\mathbf{x}}$.[6] This gain of the true label assignment $\hat{\mathbf{x}}$ over a possible assignment $\mathbf{x}$ is defined by:

$$\log \Pr(\hat{\mathbf{x}}|\mathbf{D}, \boldsymbol{\theta}) - \log \Pr(\mathbf{x}|\mathbf{D}, \boldsymbol{\theta}) = \boldsymbol{\theta}\mathbf{F}(\hat{\mathbf{x}} - \mathbf{x}). \tag{5.2.6}$$

[5]For ease of understanding we describe this approach using one training image. It can be easily extended to multiple images easily, *e.g.* by concatenation.

[6]This objective is similar to that in support vector machines [106].

The problem of maximizing the gain of the true label assignment can be formulated as the following quadratic program (QP):

$$\max \quad \gamma \qquad \text{s.t.} \qquad \boldsymbol{\theta}\mathbf{F}(\hat{\mathbf{x}} - \mathbf{x}) \geq \gamma\ell(\hat{\mathbf{x}}, \mathbf{x}); \quad \|\boldsymbol{\theta}\|^2 \leq 1, \tag{5.2.7}$$

where the gain depends on the number of misclassified labels in $\mathbf{x}$, and is denoted by $\ell(\hat{\mathbf{x}}, \mathbf{x})$. This optimization problem can be re-formulated as the following QP by dividing through by γ and adding a slack variable ξ for non-separable data:

$$\min \quad \frac{1}{2}\|\boldsymbol{\theta}\|^2 + C\xi, \tag{5.2.8}$$

$$\text{subject to} \quad \boldsymbol{\theta}\mathbf{F}(\hat{\mathbf{x}} - \mathbf{x}) \geq \ell(\hat{\mathbf{x}}, \mathbf{x}), \forall \mathbf{x} \in \mathcal{L}^N. \tag{5.2.9}$$

This quadratic program has a constraint for every possible label assignment $\mathbf{x}$, resulting in an exponentially large optimization problem. Taskar *et al.* [102] replaced the exponential set of linear constraints with a single equivalent non-linear constraint using $\max_{\mathbf{x} \in \mathcal{L}^N} \mathbf{x}$. Finding this single constraint[7] involves performing inference at every step of the algorithm. The advantage of max-margin framework is that it eliminates the partition function by using the gain (5.2.6). However, it can be computationally expensive if the inference step to find the constraint cannot be performed efficiently.

In summary, pseudo-likelihood learning approximates the partition function and is easy to compute. However, it can lead to poor accuracy. On the other hand, max-margin learning eliminates the partition function, but suffers from computational issues for certain random fields. Inspired by the successes of pseudo-likelihood and max-margin learning, we present a new method, which has the benefits of the two approaches. We first decompose the random field into distinct *pieces* (according to pseudo-likelihood structure), and treat each *piece* as an individual training exemplar. We then perform efficient discriminative learning (similar to the max-margin approach) with these exemplars. In other words, our proposed approach is a max-margin piecewise learning method, which exploits the pseudo-likelihood graph structure. Our discriminative approach is not only efficient, but also applicable to any random field model. We describe the details

[7]Also referred to as the most violated constraint.

of our method in the following section.

5.3 The Piecewise Model

Consider the pseudo-likelihood method to approximate the joint likelihood of the labelling discussed in §5.2.1. The joint likelihood of the image labelling is approximated as product of pseudo-likelihood terms defined over each pixel. The pseudo-likelihood term for a pixel i, denoted by $PL(i, \mathbf{D}^m)$, is given by equation (5.2.3). This term depends only on the labels taken by the pixel i and its immediate neighbours, *i.e.* its Markov blanket. We can interpret pseudo-likelihood as the (exact) likelihood of the vertex i in a new tree-structured graph consisting of the pixel i and its Markov blanket. In our piecewise framework we consider each of these new graphs[8] as an individual training exemplar to learn the parameters. The energy function defined on the tree-structured graph for a vertex i, in vector form, is given by[9]:

$$E^{\mathbf{i}}(\mathbf{x}) = \boldsymbol{\theta}^{\top}\mathbf{f}(\mathbf{i}, \mathbf{j}, \mathbf{D}, \mathbf{x}) + b, \tag{5.3.1}$$

where $\mathbf{i}$ is the set of all nodes in the tree-structured graph, $\boldsymbol{\theta} = (\boldsymbol{\theta}_k; \boldsymbol{\theta}_{kl}), \forall k, l \in \mathcal{L}$,[10] is the parameter vector, which is to be learnt, and $\mathbf{j} = \{j | j \in \mathcal{N}_i\}$, is the set of neighbours of the vertex i. The feature vector $\mathbf{f}(\mathbf{i}, \mathbf{j}, \mathbf{D}, \mathbf{x})$ is formed by concatenating the unary and the pairwise features of all the nodes in the tree-structured graph. The number of possible labellings for each pseudo-likelihood tree structure is given by $|\mathcal{L}|^{N_p}$, where N_p is the number of vertices in the tree. For example, the number of vertices in the tree is 5 when using a 4-neighbourhood CRF. Among the set of possible labellings, one of them is the ground truth labelling, which is referred to as the positive training example. All the other labellings form the negative example set, which is exponentially large. Let M_+ and M_- denote the number of positive and negative training examples in the entire training dataset respectively. Furthermore, the feature vectors corresponding to the m^{th} positive and the n^{th} negative training example are denoted by $\mathbf{f}_+^m(\cdot)$ and $\mathbf{f}_-^n(\cdot)$ respectively.

[8] One for each pixel in the image.

[9] For brevity we have dropped the index m over training images.

[10] The operator (;) denotes vector concatenation.

5.3.1 Parameter Learning

The parameter vector $\boldsymbol{\theta}$ and the bias b should ideally satisfy the following margin constraints:

$$\begin{aligned} \boldsymbol{\theta}^\top \mathbf{f}_+^m(i, \mathbf{j}, \mathbf{D}, \mathbf{x}) + b &\geq 1, \quad \forall m \in \{1, \cdots, M_+\}, \\ \boldsymbol{\theta}^\top \mathbf{f}_-^n(i, \mathbf{j}, \mathbf{D}, \mathbf{x}) + b &\leq -1, \quad \forall n \in \{1, \cdots, M_-\}. \end{aligned} \tag{5.3.2}$$

These constraints ensure that the parameters discriminate between the positive and negative examples with respect to the quality function $E^{\mathbf{i}}(\cdot)$ in equation (5.3.1). The most discriminative parameter vector is obtained by maximizing the margin. This is equivalent to minimizing $\|\boldsymbol{\theta}\|^2$, the L^2 norm of the parameter vector. However, it is not always possible to separate the data by solving this hard-margin optimization problem. It is common to introduce slack variables in such cases [106]. The optimal parameter vector is then learnt by solving the following soft-margin optimization problem:

$$(\boldsymbol{\theta}^*, b^*) = \arg\min_{\boldsymbol{\theta}, b} \frac{1}{2}\|\boldsymbol{\theta}\|^2 + C\Big(\sum_m \xi_+^m + \sum_n \xi_-^n\Big), \tag{5.3.3}$$

$$\text{subject to} \quad \boldsymbol{\theta}^\top \mathbf{f}_+^m(i, \mathbf{j}, \mathbf{D}, \mathbf{x}) + b \geq 1 - \xi_+^m, \quad \forall m, \tag{5.3.4}$$

$$\boldsymbol{\theta}^\top \mathbf{f}_-^n(i, \mathbf{j}, \mathbf{D}, \mathbf{x}) + b \leq -1 + \xi_-^n, \quad \forall n, \tag{5.3.5}$$

$$\xi_+^m \geq 0, \quad \forall m \in \{1, \cdots, M_+\}, \tag{5.3.6}$$

$$\xi_-^n \geq 0, \quad \forall n \in \{1, \cdots, M_-\}. \tag{5.3.7}$$

The tradeoff between the accuracy and regularization of the parameter vector is controlled by the user-defined constant $C \geq 0$. The slack variables ξ_+^m and ξ_-^n denote the hinge loss for positive and negative examples respectively.

The above convex problem is seemingly easy to solve. However, it cannot be solved efficiently because the inequality (5.3.5) specifies $|\mathcal{L}|^{N_p} - 1$ constraints for each tree-structured training example. Felzenszwalb *et al.* [21] and Kumar *et al.* [60] proposed methods to address similar issues in other learning problems. An iterative method proposed for the supervised case in [21] approximates the large optimization problem using a small subset of constraints. The algorithm alternates between two steps: (i) Given a current estimate of the parameters, a subset of labellings that maximize $\boldsymbol{\theta}^\top \mathbf{f}_-^n$ for each negative example is found using max-sum belief propagation (BP) [75]; and (ii) Using the subset of labellings obtained in step (i), a new parameter vector and bias are computed. As noted in [60], this method is susceptible to local minima and is heavily dependent on obtaining a good initial estimate of the parameters. Recently, Kumar *et al.* [60] proposed an efficient algorithm to obtain the globally optimal solution to this problem. The key step in their approach is reducing the original large problem to an equivalent one with a polynomial number of constraints. We explain this reduction step in the context of our learning problem in the next section.

5.3.2 Constraint Reformulation

The main bottleneck in solving problem (5.3.3) is the inequality (5.3.5), which specifies an exponentially large number of constraints. For example, consider a stereo matching problem where every pixel in the image can be assigned any one of 25 disparity labels. Assuming that the CRF is defined using the 4-neighbourhood structure, each tree-structured negative exemplar results in nearly 10 million constraints. The inequality (5.3.5) can be reduced to an equivalent set of $\mathrm{O}\left(N_p|\mathcal{L}|^2\right)$ constraints, where N_p is the number of nodes in the pseudo-likelihood graph, and $|\mathcal{L}|$ is the number of labels [60]. We begin by reformulating inequality (5.3.5) as follows:

$$t^n + b \leq -1 + \xi_-^n, \tag{5.3.8}$$

$$t^n \geq \boldsymbol{\theta}^\top \mathbf{f}_-^n(i, \mathbf{j}, \mathbf{D}, \mathbf{x}), \quad \forall n. \tag{5.3.9}$$

In other words, we introduce t^n, which is an upper bound on the set of values $\boldsymbol{\theta}^\top \mathbf{f}_-^n(i, \mathbf{j}, \mathbf{D}, \mathbf{x}), \forall n$. We now show that this upper bound can be specified by a polynomial number, specifically $\mathrm{O}\left(N_p|\mathcal{L}|^2\right)$, of constraints. We define variables $S^n_{ij;y_{i:k}}$ using $|\mathcal{L}|$ constraints such that,

$$S^n_{ji;x_{i:k}} \geq \boldsymbol{\theta}_l^\top x_{j:l} h_j(\mathbf{D}) + \boldsymbol{\theta}_{kl}^\top \, x_{i:k} x_{j:l} \nu_{ij}(\mathbf{D}), \quad \forall x_{j:l}, l \in \mathcal{L}, \tag{5.3.10}$$

where $j \in \mathbf{i} - \{i\}$ and $k \in \mathcal{L}$. Note that $\mathbf{i}$ is the set of vertices in the pseudo-likelihood graph. Since one $S^n_{ij;y_{i:k}}$ is defined for each $j \in \mathbf{i} - \{i\}$ and $l \in \mathcal{L}$, the number of constraints is $\mathrm{O}\left(N_p|\mathcal{L}|^2\right)$. The smallest value of $S^n_{ji;x_{i:k}}$ which satisfies the above inequality is the message that j passes to i when performing max-sum BP on the pseudo-likelihood graph with potentials given in (5.1.1). Thus, the upper bound is specified by:

$$t^n \geq \boldsymbol{\theta}^\top x_{i:k} h_i(\mathbf{D}) + \sum_{j \in \mathcal{V}_p - \{i\}} S^n_{ji;x_{i:k}}, \quad \forall x_{i:k}, k \in \mathcal{L}, \tag{5.3.11}$$

The inequality (5.3.5) can now be replaced by inequalities (5.3.9), (5.3.10), and (5.3.11) in the soft-margin optimization problem (5.3.3). The original optimization problem is now reformulated as:

$$(\boldsymbol{\theta}^*, b^*) = \arg\min_{\boldsymbol{\theta}, b} \ \frac{1}{2}\|\boldsymbol{\theta}\|^2 + C(\sum_m \xi^m_+ + \sum_m \xi^n_-), \tag{5.3.12}$$

$$\begin{aligned}
\text{s.t.} \quad & \boldsymbol{\theta}^\top \mathbf{f}^m_+(i, \mathbf{j}, \mathbf{D}, \mathbf{x}) + b \geq \quad 1 - \xi^m_+, \quad \xi^m_+ \geq 0, \forall m, \\
& t^n + b \leq -1 + \xi^n_-, \quad \xi^n_- \geq 0, \forall n, \\
& t^n \geq \boldsymbol{\theta}^\top x_{i:k} h_i(\mathbf{D}) + \sum_{j \in \mathcal{V}_p - \{i\}} S^n_{ji;x_{i:k}}, \forall x_{i:k}, n, \\
& S^n_{ji;x_{i:k}} \geq \boldsymbol{\theta}_l^\top x_{j:l} h_j(\mathbf{D}) + \boldsymbol{\theta}_{kl}^\top \ x_{i:k} x_{j:l} \nu_{ij}(\mathbf{D}), \forall x_{j:l}, n.
\end{aligned}$$

The number of constraints can be further reduced if the pairwise features $\nu_{ij}(\mathbf{x})$ are restricted to form a Potts model, as shown in [60]. In fact, this is applicable to other commonly used pairwise features such as truncated linear, and truncated quadratic models using the distance transform technique of [20]. The optimization problem (5.3.12) can be solved using the dual decomposition method as shown in [60]. We follow an alternative method and solve the problem in the primal itself.

Stochastic Gradient Descent. The form of the problem (5.3.12) is very similar to the Support Vector Machine (SVM) learning problem. Many methods exist in literature to solve the SVM learning problem. We use a Stochastic Gradient Descent algorithm because of its efficiency [13]. It is an iterative algorithm to solve linear SVMs, where every iteration consists of choosing a random training sample, and updating the weight vector. The iterative updates are chosen according to the quasi-Newton method described in [13]. Empirically, we found that using a vanilla stochastic gradient descent provided a very similar result, but required a larger number of iterations. The gradient at every step is computed by performing max-sum BP on the chosen training sample. We repeated the update step over the entire training set a few times until convergence. Note that any other efficient online SVM solver can be used instead of this gradient descent method. We chose to use this algorithm owing to its theoretical and empirical advantages when solving max-margin problems similar to (5.3.12). The reader is referred to [78] for a discussion on these advantages.

5.4 Experimental Results

We evaluated the proposed learning framework on two publicly available datasets, namely man-made structure database [62] and the Middlebury-2005 stereo vision data in [84].[11] Images from these datasets are shown in Appendix A. We compare our results with those reported in [62, 71].

5.4.1 Man-made Structure Database

This dataset contains images of man-made structures, such as houses, cathedrals, buildings. The task is to detect these structures in natural scenes, and assign *structured* or *non-structured* labels to the pixels in the image. The training and the test set contain 108 and 129 images respectively. The images are selected from the Corel image database, and are of size 256×384 pixels. Each image is divided into non-overlapping 16×16 pixel-blocks, and each such block is assigned a ground truth annotation (*structured* or *non-structured*) manually. The block-level quantization introduces noise in the labels of the blocks lying on object boundaries, which leads to errors in quantitative evaluation. Kumar and Hebert [62] circumvent this problem by not counting a misclassification that is adjacent to

[11] We thank S. Kumar and Y. Li for help with datasets used in this work.

a block with ground truth label *structured* as false positive. We follow the same procedure in order to perform a fair comparison with their work. In all, the training set contains $3,004$ structured and $36,269$ non-structured blocks. Each pixel-block is represented as a node (random variable) in the CRF framework, thus resulting in a 16×24 grid structure.

Feature computation. We used the feature set described in [62]. The unary feature $h_i(\mathbf{D})$ of a node (pixel-block) i was computed using histograms of intensity gradients at various scales. Each histogram count was weighted by the gradient magnitude at that pixel. The histograms were also smoothed to alleviate the problem of hard binning. The average *spikeness* (computed using central-shift moments) of the smoothed histogram was used an indicator of the structuredness of the pixel-block. An orientation based feature obtained by passing the absolute difference between the locations of the two highest peaks of the histogram through sinusoidal non-linearity was also used. Three scales (16×16, 32×32, and 64×64 pixel windows) were considered to compute the features, and in each scale, three moment and two orientation based features were computed. Two features were additionally chosen from these multiscale features using highest peaks from the histograms. A 14-dimensional vector is composed by taking the first two moments and orientation based features at each scale, and the two additional 'peak' features. The unary feature vector contains the 14 moment and orientation features, their squares and all their pairwise products. Thus, $h_i(\mathbf{D})$ is a 119-dimensional unary feature vector. The pairwise feature vector $\nu_{ij}(\mathbf{D})$ is a difference of unary feature vectors $h_i(\mathbf{D})$ and $h_j(\mathbf{D})$.

Results. The weight vectors corresponding to the unary and pairwise features have 119-dimensions each. These were learnt using our piecewise model (§5.3). The algorithm was run until convergence (on average 120 iterations, depending on the initialization). The learnt unary parameters are used as is, but the pairwise terms are truncated using a common approximation [83] such that graph cut inference is possible [17, 50]. The qualitative results are shown in Figure 5.3. It can be observed that our performance is comparable to the state-of-the-art results [62] on this dataset. Table 5.1 shows a quantitative evaluation of our

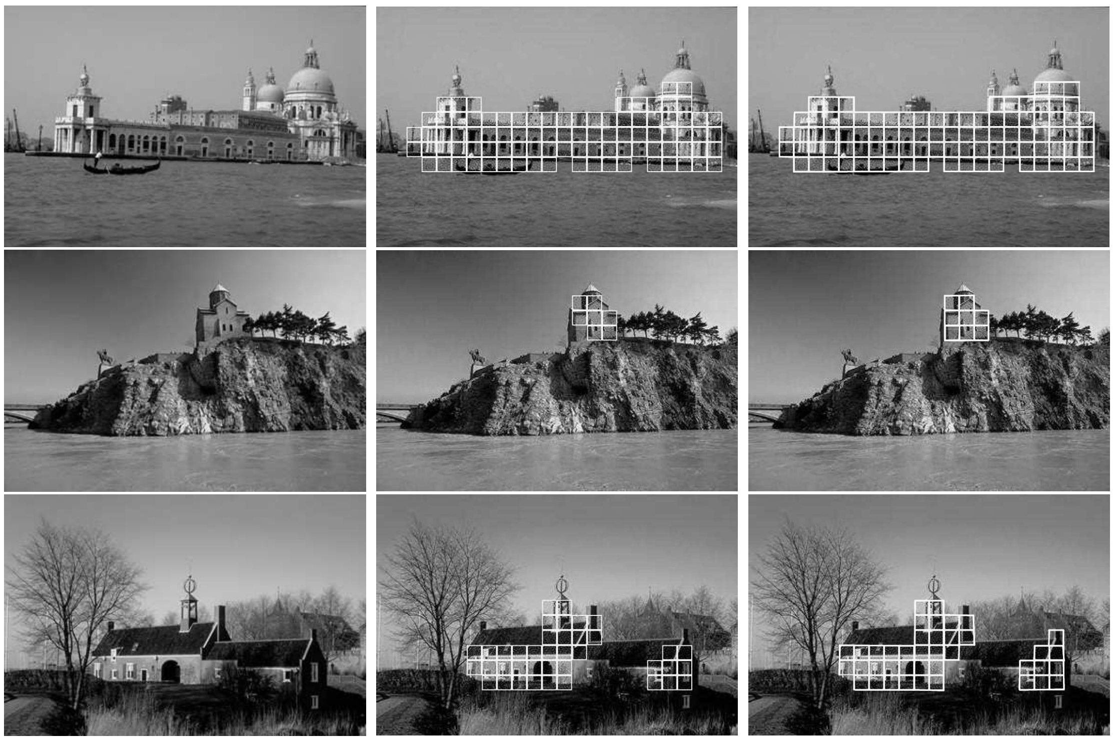


Figure 5.3: *Qualitative results on the man-made structure database. We show (a) the original image; (b) result of [62]; and (c) result of our method on three sample images from this database. The white squares overlaid on the image denote the presence of a structure. Our results correspond to an average false positive per image of 1.40. It can be observed that our performance is comparable to, if not better than, [62].*

method in terms of false positive and detection rates. We obtain a similar false positive rate and better detection percentage compared to [62]. However, our approach is computationally efficient. Our training procedure takes 409 seconds to converge compared to 627 seconds of their method on a 1.5 GHz Pentium machine. Furthermore, our approach can be easily generalized to multi-class problems, as demonstrated in the following section.

5.4.2 Middlebury-2005 Dataset

This dataset contains 9 stereo pairs (left and right images constitute a pair) in all. The problem is to compute the disparity between the left and the right image, *i.e.* for every pixel in the left image find a corresponding pixel in the right image. Since ground truth disparities are not available for three of the pairs (Computer,

Method	FP per image	DR %
MRF shown in [62]	2.36	57.20
DRF [62]	**1.37**	70.50
DRF [61]	1.76	72.54
Our method	1.40	**72.60**

Table 5.1: *Quantitative results on the man-made structure database. We show the average False Positive (*FP*) and Detection Rates (*DR*) on the test set containing 129 images. A comparison with both the Discriminative Random Field (*DRF*) methods proposed in [62] and [61] is also shown. Bold fonts indicate the lowest false positive error rate or the highest detection rate. Note that our performance is comparable to these methods. In fact, we provide a better false positive (per image) measure and similar detection rate accuracy. However, our method is computationally efficient and scales well to multi-class problems.*

Method	Art	Books	Dolls	Laundry	Moebius	Reindeer	*Average*
Grid structure in [71]	14.66	19.12	12.70	19.16	10.88	11.72	14.71
Long-range in [71]	**12.11**	**15.68**	**12.14**	**15.82**	**10.80**	15.26	13.64
Our method (*without long-range edges*)	12.94	16.24	12.21	16.72	10.82	**11.10**	**13.34**

Table 5.2: *Quantitative results showing the error rates measured as the percentage of bad pixels in the non-occluded regions on the Middlebury-2005 database. We compare our results with the models using the standard loss function (*i.e. *ignore the pixels in the occluded region when comparing with ground truth result) in [71]. 'Grid structure' refers to the model without long-range edges, and 'Long-range' is the one with these edges. Average denotes the average error rate over all the images. Bold fonts indicate the best performance (or lowest error rate). Note that our method shows better results than 'Grid structure' on all the images, and shows comparable performance to 'Long-range' on most of the images.*

Drumsticks, Dwarves), they were discarded for this performance evaluation. We used the other images, namely, Art, Books, Dolls, Laundry, Moebius, Reindeer, in a leave-one-out training framework (*i.e.* for each stereo pair problem, we train the model on all the other pairs). As noted in [71], these scenes are more challenging than the previous ones on the Middlebury Stereo Evaluation page [86]. This experimental setup is identical to that in [71]. The unary features are composed of Birchfield-Tomasi matching costs for each disparity label, and the pairwise term is a difference of disparity labels. The number of disparity levels for each image pair is identical to that used in [71]. Inference is performed on the learnt energy function using the α-expansion move making algorithm [18].

Results. A quantitative evaluation of our method is shown in Table 5.2. Error rate is measured as the percentage of incorrectly labelled pixels in the non-occluded regions. Our piecewise method performs better than the 'Grid structure' model proposed in [71]. We achieve an average error rate of 13.34 compared to 14.71 of 'Grid structure' model. As we do not use long-range edges in our approach, we perform slightly worse than the 'Long-range' model. We believe including these edges in our energy function will significantly improve the results.

5.4.3 Discussion

In the formulation discussed so far, we restricted ourselves to decomposing a CRF into sub-graphs corresponding to the Markov blanket of a single pixel. This is not an inherent limitation of the framework. Any other tree structured sub-graph, including scan-lines, can be solved in the same way. In these cases, our approach will efficiently find the most violating constraint using the trick proposed by [60].

Under our formulation, each Markov blanket (or sub-graph) is an individual training exemplar, and a unique slack variable corresponds to each sub-graph, while existing max-margin approaches treat the entire image as a single exemplar [100]. Of the two approaches, ours should be more robust to errors in data annotation — for example consider the problem of learning models for image segmentation. In these problems [62, 95], annotation of the training and test set must be done by hand, and it is common to find inaccurate ground truth labelling in large regions of the image, particularly near object boundaries. Such data is often inseparable in these regions, and global approaches such as [100] can only learn a limited amount from these images. By way of contrast, our decomposition of the image into sub-graphs allows us to disregard some of these mislabelled exemplars while learning from the remainder of the image.

5.5 Summary

This chapter presents a novel method to compute the parameters of a Conditional Random Field model. Inspired by the advantages of pseudo-likelihood and max-margin learning methods, we propose a piecewise discriminative learning framework. Our method first decompose the random field into sub-graphs, and treats each sub-graph as an individual training exemplar. We then perform efficient discriminative learning with these exemplars. We show the effectiveness of our approach on two publicly available datasets. The main contributions of the chapter are:

1. Proposing a parameter learning method applicable for large random fields commonly used in computer vision.
2. Demonstrating the efficiency of the method in terms of memory and computation.
3. Presenting an efficient max-margin based method for a larger class of random field models.
4. Showing that our method is easily applicable for multi-label problems.

Chapter 6

Discussion

In this thesis, we addressed three main issues that arise when formulating labelling problems in an energy minimization framework, *viz.* (i) How to perform efficient inference to compute the optimal solution; (ii) How to incorporate prior knowledge into the model; and (iii) How to learn the parameters of an energy function. Specifically, our work is focussed on modelling computer vision labelling problems, such as image segmentation, stereo matching, single view reconstruction, object recognition.

6.1 Our Contributions

In Chapter 3, we presented methods to improve the efficiency of energy minimization algorithms. Our first method works by recycling results from previous problem instances. The second method simplifies the miniminization problem by reducing the number of variables in the energy functions. We also showed how the reduction step can be used to generate effective problem initializations. We demonstrated that our methods for improving computational efficiency can be used for a wide range of miniminization algorithms, such as α-expansion, $\alpha\beta$-swap, BP, and TRW-S. Our method for recycling solutions extended the work on dynamic graph cuts[1] to certain non-submodular energy functions. We also proved that our method for reducing the number of variables is applicable for an important class of higher order energy functions. A substantial improvement in the running time of many large labelling problems was demonstrated.

In Chapter 4, we demonstrated how natural image priors can be used to improve single view 3D reconstruction results. We introduced a new class of multi-label higher order functions to model these priors, and showed that the resulting energy function can be solved exactly. There are three main contributions of this work. Firstly, we presented a framework to transform certain multi-label higher order functions to boolean submodular second order functions, which can be minimized exactly using graph cuts. Secondly, we extended the sub-class of submodular energy functions that can be formulated as st-MINCUT problems.

[1]Recall that the original dynamic graph cuts was only limited to binary submodular energy functions.

Thirdly, we introduced higher order potential for single view reconstruction, based on the distribution of geometrical features of planar surfaces.

In Chapter 5, we addressed the important problem of learning the parameters of energy functions. Previous attempts to solve this problem suffer from various drawbacks, such as limited applicability or noisy estimates due to poor approximations. Our proposed method is applicable for any pairwise Markov/conditional random field model, and shows impressive results on challenging publicly available datasets. We can also interpret our approach as extending the class of energy functions where efficient max-margin learning methods are viable. We demonstrated that our learning method can be used with equal ease for binary and multi-label energy functions. Lastly, we showed that our method is efficient in terms of memory and computational complexity.

6.2 Future Work

Recently, many new energy minimization algorithms have been proposed in the literature. New move making algorithms [57, 59, 108] have extended the class of energy functions efficiently solved by α-expansion and $\alpha\beta$-swap. There has also been a renewed interest in proposing integer programming relaxation methods for discrete energy minimization [58]. All these methods provide a very promising direction for solving a large class of energy functions with approximation guarantees. However, solving them for large computer vision problems can be computationally expensive. It would be interesting to explore our proposed ideas for making algorithms efficient in light of these recent advancements.

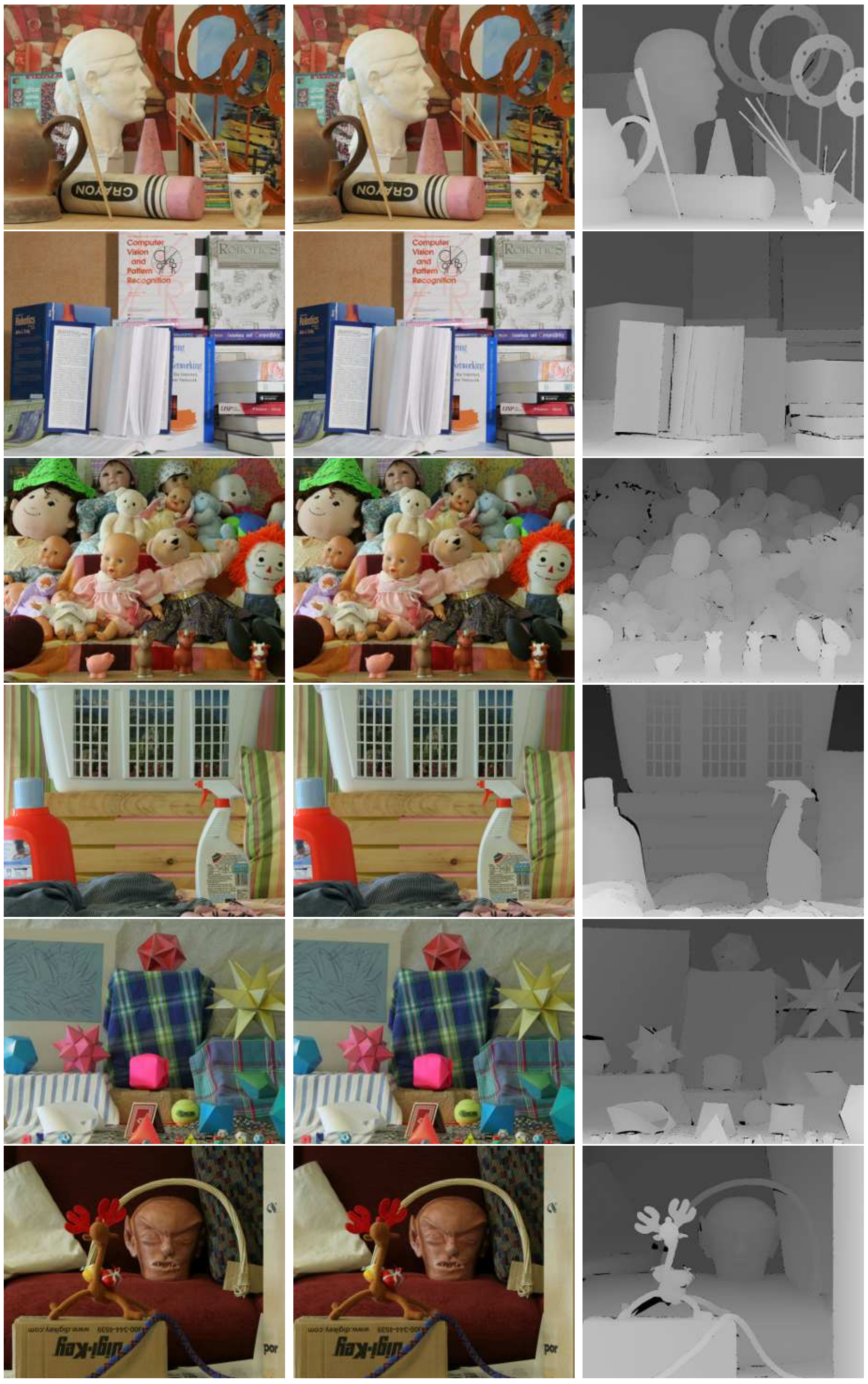
We believe that our work on extending the sub-class of submodular higher order functions that can be solved exactly is of great interest to the community. However, there is still a large set of functions for which no algorithms with polynomial run-time exist. Existing algorithms can only provide a locally or partially optimal solution. In fact, most of these algorithms provide no or very loose approximation guarantees. The development of exact or approximation algorithms with tighter bounds on the solutions for these problems remains a challenging problem.

Our work in chapter 5 is focussed on learning parameters of a pairwise random field. In the past few years there has been much interest in using higher order random fields, where the potentials can be functions of hundreds of random variables. These include potentials defined on groups of pixels (superpixels) [43, 63]. It is possible to learn parameters in this context by converting these random fields to equivalent pairwise models. However, such an approach is limited to a small class of energy functions due to the lack of widely applicable and efficient inference algorithms. Therefore, learning parameters in higher order energy functions is still an interesting and challenging problem to be explored.

Another potential direction for future research is to learn the structure of the random field. At the moment, we are imposing a structure on the labelling problem in terms of unary, pairwise, and higher order potentials. It would be more appropriate to learn the order and structure of the random field from a set of training data. There has been some work [90] in this area, but is limited to very small random fields with a few hundred variables. It is not clear if their approach is scalable to the large models in computer vision. In the future, the hope is to be able to give all our supervised training data to a black box, which would come up with the best random field structure for the task, and also provide solutions for unseen (test) data.

Appendix A

Datasets



(a) (b) (c)

Figure A.1: *Images from Middlebury-2005 dataset used in our experiments in Chapter 5. (a) The left image, (b) The right image, and (c) Ground truth disparity map for 'Arts', 'Books', 'Dolls', 'Laundry', 'Moebius', and 'Reindeer' are shown (top to bottom).*

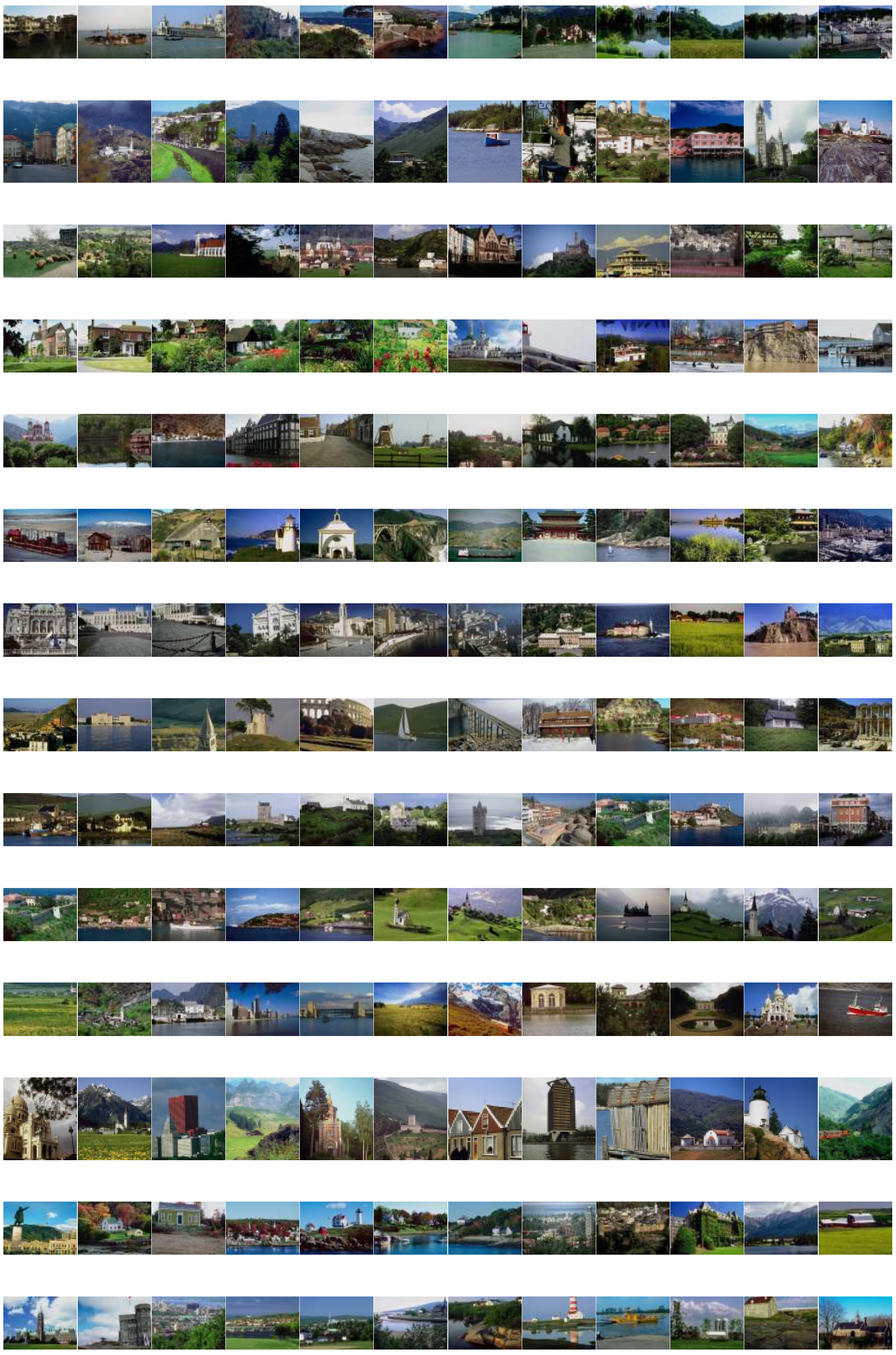


Figure A.2: *Some of the images from man-made structure dataset used in our experiments in Chapter 5. This dataset is available for download at:* `http://www.cs.cmu.edu/~skumar`.

Bibliography

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] K. Alahari, P. Kohli, and P. H. S. Torr. Reduce, reuse & recycle: Efficiently solving multi-label MRFs. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.

[3] K. Alahari, P. Kohli, and P. H. S. Torr. Dynamic hybrid algorithms for map inference in discrete MRFs. *IEEE Trans. Pattern Anal. Mach. Intell.*, In press, 2010.

[4] K. Alahari, C. Russell, and P. H. S. Torr. Efficient piecewise learning for conditional random fields. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010.

[5] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of Markov random fields for segmentation of 3D scan data. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 169–176, 2005.

[6] J. Besag. Spatial interactions and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B*, 36(2):192–236, 1974.

[7] J. Besag. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society. Series D*, 24(3):179–195, 1975.

[8] A. Billionnet and M. Minoux. Maximizing a supermodular pseudo-boolean function: A polynomial algorithm for supermodular cubic functions. *Discrete Applied Mathematics*, 12(1):1–11, 1985.

[9] S. T. Birchfield, B. Natarajan, and C. Tomasi. Correspondence as energy-based segmentation. *Image and Vision Computing*, 25(8):1329–1340, 2007.

[10] S. T. Birchfield and C. Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(4):401–406, 1998.

[11] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[12] A. Blake, C. Rother, M. Brown, P. Perez, and P. H. S. Torr. Interactive image segmentation using an adaptive GMMRF model. In *Proc. European Conf. Computer Vision*, volume 1, pages 428–441, 2004.

[13] A. Bordes, L. Bottou, and P. Gallinari. SGD-QN: Careful quasi-newton stochastic gradient descent. *J. Machine Learning Research*, 10:1737–1754, 2009.

[14] E. Boros and P. L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1-3):155–225, 2002.

[15] Y. Boykov and G. Funka-Lea. Graph cuts and efficient N-D image segmentation. *Int'l J. Computer Vision*, 70(2):109–131, 2006.

[16] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *Proc. IEEE Int'l Conf. Computer Vision*, volume 1, pages 105–112, 2001.

[17] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, 2004.

[18] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001.

[19] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) challenge. *Int'l J. Computer Vision*, 88(2):303–338, June 2010.

[20] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *Int'l J. Computer Vision*, 70(1):41–54, 2006.

[21] P. F. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.

[22] B. Flach and D. Schlesinger. Best labelling search for a class of higher order gibbs models. *Pattern Recognition and Image Anal.*, 14(2):249–254, 2004.

[23] L. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, Princeton, NJ, 1962.

[24] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach.* Prentice Hall, New Jersey, 2003.

[25] D. Freedman and P. Drineas. Energy minimization via graph cuts: Settling what is possible. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 939–946, 2005.

[26] W. Freeman, E. Pasztor, and O. Carmichael. Learning low-level vision. *Int'l J. Computer Vision*, 40(1):25–47, 2000.

[27] B. Frey and D. MacKay. A revolution: Belief propagation in graphs with cycles. In *Advances in Neural Information Processing Systems*, 1997.

[28] G. Gallo and B. Simeone. On the supermodular knapsack problem. *Mathematical Programming: Series A and B*, 45(2):295–309, 1989.

[29] S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(6):721–741, 1984.

[30] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *J. ACM*, 35(4):921–940, 1988.

[31] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *J. Royal Statist. Soc. B*, 51(2):271–279, 1989.

[32] P. L. Hammer. Some network flow problems solved with pseudo-boolean programming. *Operations Research*, 13:388–399, 1965.

[33] J. M. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. Unpublished, 1971.

[34] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[35] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. In *ACM SIGGRAPH*, pages 577–584, 2005.

[36] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *Int'l J. Computer Vision*, 75(1):151–172, 2007.

[37] H. Ishikawa. Exact optimization for Markov random fields with convex priors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(10):1333–1336, 2003.

[38] S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *J. ACM*, 48(4):761–777, 2001.

[39] O. Juan and Y. Boykov. Active graph cuts. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 1023–1029, 2006.

[40] R. Kindermann and J. L. Snell. *Markov Random Fields and Their Applications*. American Mathematical Society, 1980.

[41] P. Kohli. *Minimizing Dynamic and Higher Order Energy Functions using Graph Cuts*. PhD thesis, Oxford Brookes University, November 2007.

[42] P. Kohli, M. P. Kumar, and P. H. S. Torr. P^3 and beyond: Move making algorithms for solving higher order functions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(9):1645–1656, 2009.

[43] P. Kohli, L. Ladicky, and P. H. S. Torr. Robust higher order potentials for enforcing label consistency. *Int'l J. Computer Vision*, 82:302–324, 2009.

[44] P. Kohli, J. Rihan, M. Bray, and P. H. S. Torr. Simultaneous segmentation and pose estimation of humans using dynamic graph cuts. *Int'l J. Computer Vision*, 79(3):285–298, 2008.

[45] P. Kohli, A. Shekhovtsov, C. Rother, V. Kolmogorov, and P. H. S. Torr. On partial optimality in multi-label MRFs. In *Proc. Int'l Conf. Machine Learning*, pages 480–487, 2008.

[46] P. Kohli and P. H. S. Torr. Dynamic graph cuts for efficient inference in Markov random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(12):2079–2088, 2007.

[47] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1568–1583, 2006.

[48] V. Kolmogorov and C. Rother. Minimizing non-submodular functions with graph cuts: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(2), 2007.

[49] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *Proc. European Conf. Computer Vision*, volume 3, pages 82–96, 2002.

[50] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):147–159, 2004.

[51] N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *Proc. IEEE Int'l Conf. Computer Vision*, 2007.

[52] N. Komodakis and G. Tziritas. A new framework for approximate labeling via graph cuts. In *Proc. IEEE Int'l Conf. Computer Vision*, pages 1018–1025, 2005.

[53] N. Komodakis, G. Tziritas, and N. Paragios. Fast, approximately optimal solutions for single and dynamic MRFs. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007.

[54] I. Kovtun. Partial optimal labeling search for a NP-Hard subclass of (max,+) problems. In *DAGM Symposium*, pages 402–409, 2003.

[55] I. Kovtun. *Image Segmentation based on Sufficient Conditions for Optimality in NP-complete Classes of Structural Labeling Problems.* PhD thesis, IRTC ITS Nat. Academy of Science Ukraine, Kiev, 2004. In Ukranian.

[56] M. P. Kumar. *Combinatorial and Convex Optimization for Probabilistic Models in Computer Vision*. PhD thesis, Oxford Brookes University, 2008.

[57] M. P. Kumar and D. Koller. MAP estimation of semi-metric MRFs via hierarchical graph cuts. In *Proc. Conf. Uncertainity in Artificial Intelligence*, 2009.

[58] M. P. Kumar, V. Kolmogorov, and P. H. S. Torr. An analysis of convex relaxations for MAP estimation of discrete MRFs. *J. Machine Learning Research*, 10:71–106, 2009.

[59] M. P. Kumar and P. H. S. Torr. Improved moves for truncated convex models. In *Advances in Neural Information Processing Systems*, 2008.

[60] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Efficient discriminative learning of parts-based models. In *Proc. IEEE Int'l Conf. Computer Vision*, 2009.

[61] S. Kumar and M. Hebert. Discrimative fields for modelling spatial dependencies in natural images. In *Advances in Neural Information Processing Systems*, 2003.

[62] S. Kumar and M. Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *Proc. IEEE Int'l Conf. Computer Vision*, 2003.

[63] L. Ladicky, C. Russell, P. Kohli, and P. H. S. Torr. Associative hierarchical CRFs for object class image segmentation. In *Proc. IEEE Int'l Conf. Computer Vision*, 2009.

[64] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labelling sequence data. In *Proc. Int'l Conf. Machine Learning*, pages 282–289, 2001.

[65] X. Lan, S. Roth, D. P. Huttenlocher, and M. J. Black. Efficient belief propagation with learned higher-order Markov random fields. In *Proc. European Conf. Computer Vision*, volume 2, pages 269–282, 2006.

[66] V. Lempitsky, A. Blake, and C. Rother. Image segmentation by branch-and-mincut. In *Proc. European Conf. Computer Vision*, pages 15–29, 2008.

[67] V. Lempitsky and Y. Boykov. Global optimization for shape fitting. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007.

[68] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *Int'l J. Computer Vision*, 43(1):29–44, 2001.

[69] A. Levin, A. Zomet, S. Peleg, and Y. Weiss. Seamless image stitching in the gradient domain. In *ECCV*, pages 377–389, 2004.

[70] S. Z. Li. *Markov Random Field Modeling in Computer Vision.* Springer-Verlag, 2000.

[71] Y. Li and D. P. Huttenlocher. Learning for stereo vision using the structured support vector machine. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.

[72] Y. Nakamura, T. Matsuura, K. Satoh, and Y. Ohta. Occlusion detectable stereo – occlusion patterns in camera matrix. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 371–378, 1996.

[73] J. B. Orlin. A faster stongly polynomial time algorithm for submodular function minimization. In *Integer Programming and Combinatorial Optimization*, pages 240–251, 2007.

[74] R. Paget and I. D. Longstaff. Texture synthesis via a noncausal nonparametric multiscale Markov random field. *IEEE Trans. Image Processing*, 7(6):925–931, 1998.

[75] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[76] A. Raj and R. Zabih. A graph cut algorithm for generalized image deconvolution. In *Proc. IEEE Int'l Conf. Computer Vision*, volume 2, pages 1048–1054, 2005.

[77] S. Ramalingam, P. Kohli, K. Alahari, and P. H. S. Torr. Exact inference in multi-label CRFs with higher order cliques. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.

[78] N. Ratliff, J. A. Bagnell, and M. Zinkevich. (Online) subgradient methods for structured prediction. In *AIStats*, 2007.

[79] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 860–867, 2005.

[80] S. Roth and M. J. Black. On the spatial statistics of optical flow. *Int'l J. Computer Vision*, 74(1):33–50, 2007.

[81] C. Rother, V. Kolmogorov, and A. Blake. GrabCut: Interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH*, pages 309–314, 2004.

[82] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary MRFs via extended roof duality. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007.

[83] C. Rother, S. Kumar, V. Kolmogorov, and A. Blake. Digital tapestry. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 589–596, 2005.

[84] D. Scharstein and C. Pal. Learning conditional random fields for stereo. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007.

[85] D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. *Int'l J. Computer Vision*, 28(2):155–174, 1998.

[86] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithm. *Int'l J. Computer Vision*, 47:7–42, 2002.

[87] D. Schlesinger. Exact solution of permuted submodular minsum problems. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 28–38, 2007.

[88] D. Schlesinger and B. Flach. Transforming an arbitrary minsum problem into a binary one. Technical Report TUD-FI06-01, Dresden University of Technology, April 2006.

[89] M. I. Schlesinger and V. Hlavac. *Ten Lectures on Statistical and Structural Pattern Recognition.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[90] M. Schmidt, K. Murphy, G. Fung, and R. Rosales. Structure learning in random fields for heart motion abnormality detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2008.

[91] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Combinatorial Theory, Series B*, 80(2):346–355, 2000.

[92] F. Schroff, A. Criminisi, and A. Zisserman. Single-histogram class models for image segmentation. In *Proc. Indian Conf. Vision Graphics and Image Processing*, 2006.

[93] J. Shotton, A. Blake, and R. Cipolla. Multi-scale categorical object recognition using contour fragments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(7):1270–1281, 2008.

[94] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.

[95] J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *Int'l J. Computer Vision*, 81(1):2–23, 2009.

[96] J. Sun, N. Zheng, and H.-Y. Shum. Stereo matching using belief propagation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(7):787–800, 2003.

[97] C. Sutton and A. McCallum. Piecewise pseudolikelihood for efficient training of conditional random fields. In *Proc. Int'l Conf. Machine Learning*, pages 863–870, 2007.

[98] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer, In press, 2010.

[99] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. F. Tappen, and C. Rother. A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(6):1068–1080, 2008.

[100] M. Szummer, P. Kohli, and D. Hoiem. Learning CRFs using graph cuts. In *Proc. European Conf. Computer Vision*, pages 582–595, 2008.

[101] M. Tappen. Utilizing variational optimization to learn Markov random fields. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007.

[102] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems*, 2003.

[103] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(5):854–869, 2007.

[104] A. Torralba and A. Oliva. Statistics of natural image categories. *Network: Comp. Neural Syst.*, 14(3):391–412, 2003.

[105] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proc. Int'l Conf. Machine Learning*, pages 823–830, 2004.

[106] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[107] M. Varma and A. Zisserman. A statistical approach to material classification using image patch exemplars. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(11):2032–2047, 2009.

[108] O. Veksler. Graph cuts based optimization for MRFs with truncated convex priors. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007.

[109] O. Veksler. Multi-label moves for MRFs with truncated convex priors. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, 2009.

[110] V. Vicente, V. Kolmogorov, and C. Rother. Graph cut based image segmentation with connectivity priors. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.

[111] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. MAP estimation via agreement on trees: Message-passing and linear programming. *IEEE Trans. on Information Theory*, 51(11):3697–3717, 2005.

[112] Y. Weiss and W. T. Freeman. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Trans. Inf. Theory*, 47(2):736–744, 2001.

[113] T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(7):1165–1179, 2007.

[114] Z. Yang and D. Purves. Image/source statistics of surfaces in natural scenes. *Network: Comp. Neural Syst.*, 14(3):371–390, 2003.

[115] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems*, pages 689–695, 2000.

[116] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Bethe free energy, kikuchi approximations, and belief propagation algorithms. Technical Report TR2001-16, MERL, 2001.

[117] L. Zhang and S. Seitz. Estimating optimal parameters for MRF stereo from a single image pair. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(2):331–342, 2007.