Minimizing Dynamic and Higher Order Energy Functions using Graph Cuts

Pushmeet Kohli

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

Doctor of Philosophy

Oxford Brookes University

November 2007

Abstract

Over the last few years energy minimization has emerged as an indispensable tool in computer vision. The primary reason for this rising popularity has been the successes of efficient graph cut based minimization algorithms in solving many low level vision problems such as image segmentation, object reconstruction, image restoration and disparity estimation. The scale and form of computer vision problems introduce many challenges in energy minimization. In this dissertation, I will focus on some aspects of these problems.

The first problem I address relates to the efficient and exact minimization of groups of similar functions which are known to be solvable in polynomial time. I will present a novel dynamic algorithm for minimizing such functions. This algorithm reuses computation from previous problem instances to solve new instances resulting in a substantial improvement in the running time. I will present the results of using this approach on the problems of interactive image segmentation, image segmentation in video, human pose estimation and segmentation, and measuring uncertainty of solutions obtained by minimizing energy functions.

The second part of my dissertation will deal with the minimization of multi-label higher order functions which are NP-hard to minimize. These functions are able to model interactions among groups of random variables and can be used to formulate many vision problems. The runtime complexity of commonly used algorithms for approximate energy minimization such as Max-product Belief Propagation or Tree-reweighted message passing grows exponentially with the clique size, which makes them inapplicable to problems with even moderate sized cliques. I will show how certain higher order energy functions can be minimized using the graph cut based expansion and swap move algorithms. This method is extremely efficient and is able to handle cliques involving thousands of variables. I will use these higher order energy functions to model the problems of object segmentation and recognition, and texture segmentation. The results of this approach will be compared with those obtained using conventional methods which model these problems using second order energy functions.

To my parents

Acknowledgements

This dissertation would not have been possible without the help and encouragement of a number of people. I start by thanking my advisor, Prof. Philip Torr who made it possible for me to come to England to pursue my graduate studies. During the course of my stay in Oxford, I have learned a lot from Phil. His passion for research is infectious, and has helped me immensely in my research. I cannot thank him enough for his time and support. I would also like to thank Prof. Ramin Zabih and Prof. David Duce for examining this dissertation.

During my PhD studies I was fortunate to work with a number of people. I have enjoyed collaborating with M. Pawan Kumar, Mathieu Bray, Yunda Sun, Jonathan Rihan, Lubor Ladicky, Vladimir Kolmogorov, Carsten Rother, Andrew Fitzgibbon, Derek Hoeim, Martin Szummer, Karteek Alahari, Sanjiv Kumar, Alexander Shekhovtsov and Nikos Komodakis. I thank them for the many enlightening discussions we have had in the last few years. I would also like to thank Alyosha Efros, Andrew Blake, Andrew Zisserman, Antonio Criminisi, Bill Triggs, Dan Huttenlocher, Hiroshi Ishikawa, John Winn, Mark Everingham, Michael Black, Olga Veksler, Ramin Zabih, Yuri Boykov and many others for conversations which have influenced my research.

My stay in Oxford and Cambridge was made pleasurable by numerous friends and colleagues who I would like to thank for their company. These include Mukta Prasad, Carl Henrik Ek, Pawan Kumar, Karteek Alahari, Christopher Russell, Jonathan Rihan, Mathieu Bray, Srikumar Ramalingam, Ankur Agarwal, Manmohan Chandrakar, Oliver Woodford, Josef Sivic, Ondra Chum, James Philbin, Florian Schroff, Patrick Buehler, Rob Fergus, Anurag Bana, Aman Iqbal, Sarah Iqbal, Christophe Restif and others. Finally, I thank my parents who have supported me in all my endeavors.

Contents

1 Energy Minimization 1

1.1 Introduction 2

1.1.1 Outline of the Dissertation 3

1.2 Energy Minimization in Computer Vision 5

1.2.1 Markov and Conditional Random Fields 5

1.2.2 Discrete Energy Minimization 7

1.2.3 Submodular Functions 8

1.2.4 Minimizing Submodular Functions 9

1.3 Graph Cuts for Energy Minimization 9

1.3.1 The st-Mincut Problem 10

1.3.2 Minimizing Submodular Functions using Graph Cuts 12

1.4 Minimizing Non-submodular Functions 14

1.4.1 LP Relaxation of the Integer Program 15

1.4.2 Partial Optimality 16

1.4.3 Summary 16

2 Minimizing Dynamic Energy Functions 17

2.1 Dynamic Graph Cuts 18

2.1.1 Dynamic Computation 18

2.1.2 Outline of the Chapter 20

2.2 Energy and Graph Reparameterization 20

2.3 Recycling Computation 21

2.3.1 Updating Residual Graphs 23

2.3.2 Computational Complexity of Update Operations 25

2.4 Improving Performance by Recycling Search Trees 26

2.4.1 Reusing Search Trees 26

2.4.2 Tree Recycling for Dynamic Graph Cuts 27

3 Applications of Dynamic Graph Cuts 29

3.1 Dynamic Image Segmentation 30

3.1.1 CRFs for Image Segmentation 30

3.1.2 Image Segmentation in Videos 33

3.1.3 Experimental Results 34

3.1.4 Reusing Flow Vs Reusing Search Trees 36

3.2 Simultaneous Segmentation and Pose Estimation of Humans 37
3.2.1 Pose Specific CRF for Image Segmentation 42
3.2.2 Formulating the Pose Inference Problem 46
3.2.3 Experiments 49
3.2.4 Shape Priors for Reconstruction 53
3.2.5 Discussion 56
3.2.6 Summary and Future Work 59
3.3 Measuring Uncertainty in Graph Cut Solutions 59
3.3.1 Introduction 60
3.3.2 Preliminaries 61
3.3.3 Uncertainty in Label Assignments 61
3.3.4 Computing Min-marginals using Graph Cuts 63
3.3.5 Computational Complexity and Experimental Evaluation 69
3.3.6 Applications of Min-marginals 69

4 Minimizing Higher Order Functions 73
4.1 Move Making Algorithms 74
4.1.1 Previous Work 75
4.1.2 Binary Moves and Move Energies 76
4.2 Characterizing Solvable $\mathcal{P}^n$ Functions 78
4.2.1 Conditions for $\alpha\beta$-swaps 78
4.2.2 Conditions for α-expansions 80
4.2.3 Graph Cuts for the $\mathcal{P}^n$ Potts Model 82
4.2.4 Planarity Preserving Clique Potentials 85
4.2.5 Proofs of Lemmas 86

5 Graph Cuts for Minimizing Higher Order Functions 88
5.1 Robust Higher Order Potentials 89
5.2 Computing Moves for Higher Order Potentials 90
5.2.1 Swap Moves 90
5.2.2 Expansion Moves 92
5.3 Proofs of Theorems 93

6 Applications of Higher Order Potentials 97
6.1 Enforcing Label Consistency in Superpixels 98
6.1.1 Object Segmentation and Recognition 99
6.1.2 Pairwise CRFs for Object Segmentation 101

6.1.3 Incorporating Higher Order Potentials 103
6.1.4 Experiments 109
6.1.5 Summary 114
6.2 Texture Based Segmentation 114
6.2.1 Pairwise CRF for Texture Segmentation 115
6.2.2 Higher Order Patch Potentials 115
6.2.3 Results 116

7 Conclusion 120
7.1 Summary 121
7.2 Our Contributions 121
7.3 Directions for Future Work 122

Bibliography 125

List of Figures

1.1 Some labelling problems in computer vision. 6
1.2 Function minimization problems. 7
1.3 Graph reparameterization. 12
1.4 Energy minimization using graph cuts. 13
2.1 Dynamic image segmentation using graph cuts. 19
2.2 Graph reparameterization. 22
2.3 Restoring consistency using graph reparameterization. 25
3.1 Interactive image segmentation. 31
3.2 Pairwise Markov Random Field model for image labelling problems. 32
3.3 Segmentation in videos using user seeds. 33
3.4 Segmentation results of the human lame walk video sequence. . 34
3.5 Running time and number of augmenting paths found by static and dynamic st-mincut algorithms. 35
3.6 Behavior of the dynamic algorithm. 37
3.7 Improving segmentation results by incorporating more information in the CRF. 40
3.8 The human upper body model. 43
3.9 Different terms of our pose specific CRF. 45
3.10 Inferring the optimal pose. 46
3.11 Optimizing the pose parameters. 48
3.12 Resolving ambiguity in pose using multiple views. 49
3.13 Results showing the effect of incorporating a shape prior on the segmentation results. 50
3.14 Segmentation results using the 2D upper body model. 51
3.15 Segmentation results obtained by Grimson-Stauffer [100] and POSECUT. 52
3.16 Segmentation (middle) and pose estimation (bottom) results from POSECUT. 53
3.17 More segmentation (middle row) and pose estimation (bottom row) results obtained using POSECUT. 54
3.18 Ambiguity in object reconstruction due to few views. 55
3.19 3D Object reconstruction using strong object-specific priors. Also shows camera positions and orientations. 56
3.20 Pose inference and 3D object reconstruction results. 57
3.21 Real time face segmentation using face detection. 58
3.22 Pruning false object detections. 58

3.23 Flow potentials of graph nodes. 64
3.24 Computing min-marginals using graph cuts. 65
3.25 Graph construction for projections of energy functions involving multiple labels. 67
3.26 Image segmentation with max-marginal probabilities. 71
4.1 Graph construction for computing the optimal moves for the $\mathcal{P}^n$ Potts model. 83
5.1 Graph construction for minimizing higher order functions. . . . 92
6.1 Using higher order potentials for object segmentation. 100
6.2 Quality sensitive region consistency prior. 103
6.3 Behaviour of the rigid P^n Potts potential and the Robust P^n model potential. 105
6.4 Object segmentation and recognition using the Robust P^n higher order potentials. 107
6.5 Generating multiple segmentations. 108
6.6 Qualitative object segmentation and recognition results. 108
6.7 Qualitative results from the higher order CRF. 110
6.8 Accurate hand labelled segmentations which were used as ground truth. 111
6.9 Qualitative results of our method. 112
6.10 The relationship between qualitative and quantitative results. . 112
6.11 Boundary accuracy evaluation using trimap segmentations. . . . 113
6.12 Pixelwise classification error in our results. 114
6.13 Segmented keyframe of the garden sequence. 116
6.14 Qualitative texture segmentation results of the garden sequence. 117
6.15 The keyframe of the ‘Dayton’ video sequence segmented into three segments. 118
6.16 Segmentation results of the ‘Dayton’ sequence. 119

List of Tables

3.1 Quantitative segmentation results of POSECUT. 50
3.2 Algorithm for computing min-marginal energies using dynamic graph cuts. 68
3.3 Time taken for min-marginal computation. 70

Chapter 1

Energy Minimization

1.1. Introduction

The last few years have seen energy minimization emerge as an indispensable tool in computer vision. This increasing popularity has primarily been the result of the successes of graph cut based minimization algorithms in solving many low level vision problems such as image segmentation, object reconstruction, image restoration and disparity estimation. These algorithms not only produce good results but are also extremely efficient.

Energy minimization generally refers to the problem of finding the values at which a function reaches its minimum value. Many important vision problems such as image segmentation can be formulated in terms of minimizing a function, usually called the *energy* or *cost* function. Although, the problem of minimizing a general function is NP-hard, there exist classes of functions which can be minimized in polynomial time. For instance, it is well known that one such class of functions which is frequently encountered in computer vision can be minimized by solving a minimum cost st-cut (st-mincut) problem.

Although graph cuts have been known in computer vision since the 1980's [32], they were not extensively used for a long time. This changed at the turn of the century when a number of papers [9,11,41] reinvigorated interest in graph cuts by showing that they can be used for energy minimization. These methods were extremely efficient and easy to implement which made them quite popular. Graph cuts have since been successfully applied to many vision problems. The scale and form of computer vision problems introduce many challenges in energy minimization. For example, computer vision problems require minimizing functions taking millions of variables as arguments. In this dissertation, we will focus on some of these problems and propose new methods for minimizing discrete functions.

The first problem dealt with in the dissertation concerns the efficient and exact minimization of groups of similar functions which can be minimized optimally by solving a st-mincut problem. I will present a novel dynamic st-mincut algorithm for minimizing such functions and demonstrate its use in solving a number of vision problems. This algorithm reuses computation from previous problem instances to solve new instances resulting in substantial improvement in the running time. I will present the results of using this approach on the problems of interactive image segmentation, image segmentation in video, human pose estimation and segmentation, and measuring uncertainty of solutions obtained by minimizing energy functions.

The second part of the dissertation will deal with the minimization of multi-label higher order functions which are NP-hard to minimize. These functions are able to model interactions among groups of random variables and can be used to formulate many vision problems. The runtime complexity of commonly used algorithms for approximate energy minimization such as Max-product Belief Propagation or Tree-reweighted message passing grows exponentially with the clique size, which makes them inapplicable to problems with even moderate sized cliques. I will show how certain higher order energy functions can be minimized using the graph cut based expansion and swap move algorithms. This method is extremely efficient and is able to handle cliques involving thousands of variables. I will use these higher order energy functions to model the problems of object segmentation and recognition, and texture segmentation. The results of this approach will be compared with those obtained using conventional methods which model these problems using second order energy functions.

1.1.1 Outline of the Dissertation

A brief outline of the dissertation follows.

Chapter 1 In chapter 1 we review the basic concepts of discrete optimization and discuss its use in solving computer vision problems. We explain the Markov and Conditional Random Field models used in computer vision and show how they can be solved by minimizing an energy function. We talk about the class of submodular functions and explain how some functions belonging to this class can be minimized by solving an st-mincut problem. The chapter concludes by briefly reviewing some of the work done in minimizing non-submodular functions.

Chapter 2 In this chapter we describe a fast new fully dynamic algorithm for the st-mincut/max-flow problem. This algorithm can be used to efficiently minimize dynamically changing submodular energy functions encountered in computer vision. Specifically, given the solution of the max-flow problem on a graph, the dynamic algorithm efficiently computes the maximum flow in a modified version of the graph. The time taken by it is roughly proportional to the total amount of change in the edge weights of the graph. Our experiments show that, when the number of changes in the graph is small, the dynamic algorithm is significantly faster than the best known static graph cut algorithm. Previous versions of this chapter appeared as [48] and [50].

Chapter 3 The dynamic graph cut algorithm described in the previous chapter can be used to *dynamically* perform MAP inference in an MRF or CRF. Such an inference procedure is extremely fast and has been used for a number of problems [12,35,49,84,102]. In this chapter we describe few applications of the dynamic graph cut algorithm. Specifically, I will present the results of using this approach on the problems of interactive image segmentation, image segmentation in video, human pose estimation and segmentation, and measuring uncertainty of solutions obtained by minimizing energy functions. Parts of this chapter previously appeared in [12,49,50,102].

Chapter 4 In chapter 4 we extend the class of energy functions for which the optimal α-expansion and $\alpha\beta$-swap moves can be computed in polynomial time. Specifically, we introduce a novel family of higher order clique potentials and show that the expansion and swap moves for any energy function composed of these potentials can be found by minimizing a submodular function. We go on to show that for a subset of these potentials (which we call the $\mathcal{P}^n$ Potts model family), the optimal move can be found by solving an st-mincut problem. The $\mathcal{P}^n$ Potts model is a higher order generalization of the well known Potts model pairwise potential. We conclude the chapter by providing an example of a useful higher order potential for which it is NP-hard to compute the optimal move. Parts of this chapter previously appeared as [46,47].

Chapter 5 In chapter 5 we introduce a novel family of higher order potentials which we call the Robust P^n model (see equation 5.1.1). We show that the optimal expansion and swap moves for functions composed of such potentials can be found by solving an st-mincut problem. Our method for computing the optimal expansion and swap moves is extremely efficient and can handle potentials defined over cliques consisting of thousands of random variables. The Robust P^n model can be used for modelling many computer vision problems. In fact the P^n Potts model introduced in the previous chapter is a member of this family of higher order potentials.

Chapter 6 In chapter 6 we show two applications of higher order potentials in computer vision. Our first example shows how higher order potentials can be used for enforcing consistency in labelling of sets of random variables. We use this method for the problem of multi-class object segmentation by augmenting the conventional CRF used for object segmentation with higher order potentials defined on image regions. These potentials encourage solutions assigning the same label to all pixels belonging to a segment. The second example deals with the problem of texture based segmentation. In both problems we compare the results of our method with those obtained using state of the art methods based on pairwise CRFs.

Chapter 7 In the last and concluding chapter of this dissertation, I give a summary of this work and list its main contributions. I end the chapter by discussing some promising directions for future research.

1.2. Energy Minimization in Computer Vision

Many problems in computer vision and scene understanding can be formulated in terms of finding the most probable values of certain hidden or unobserved variables. These variables encode a desired property of the scene and can be continuous or discrete. For the case of discrete variables, these problems are commonly referred to as *labelling problems* as they involve assigning a label to the hidden variables. Labelling problems occur in many forms, from lattice based problems of dense stereo and image segmentation [11, 104] to the use of pictorial structures for object recognition [23]. Some examples of problems which can be formulated in this manner are shown in figure 1.1.

One of the major advances in computer vision in the past few years has been the use of efficient deterministic algorithms for solving discrete labelling problems. In particular, efficient graph cut based minimization algorithms have been extremely successful in solving many low level vision problems. These methods work by inferring the maximum a posteriori (MAP) solutions of conditional and markov random fields which are generally used to model these problems.

1.2.1 Markov and Conditional Random Fields

Random fields provide an elegant probabilistic framework to formulate labelling problems. They are able to model complex interactions between hidden variables in a simple and precise manner. The power of this representation lies in the fact that the probability distribution over different labellings of the random variables factorizes, and thus allows efficient inference.

Consider a discrete random field $\mathbf{X}$ defined over a lattice $\mathcal{V} = \{1, 2, \ldots, n\}$ with a neighbourhood system $\mathcal{N}$. Each random variable $X_i \in \mathbf{X}$ is associated with a lattice point $i \in \mathcal{V}$ and takes a value from the label set $\mathcal{L} = \{l_1, l_2, \ldots, l_k\}$. The neighbourhood system $\mathcal{N}$ of the random field is defined by the sets $\mathcal{N}_i, \forall i \in \mathcal{V}$, where $\mathcal{N}_i$ denotes the set of all neighbours of the variable X_i. A clique c is a set of random variables $\mathbf{X}_c$ which are conditionally dependent on each other. Any possible assignment of labels to the random variables is called a *labelling* or *configuration*. It is denoted by the vector $\mathbf{x}$, and takes values from the set $\mathbf{L} = \mathcal{L}^n$.

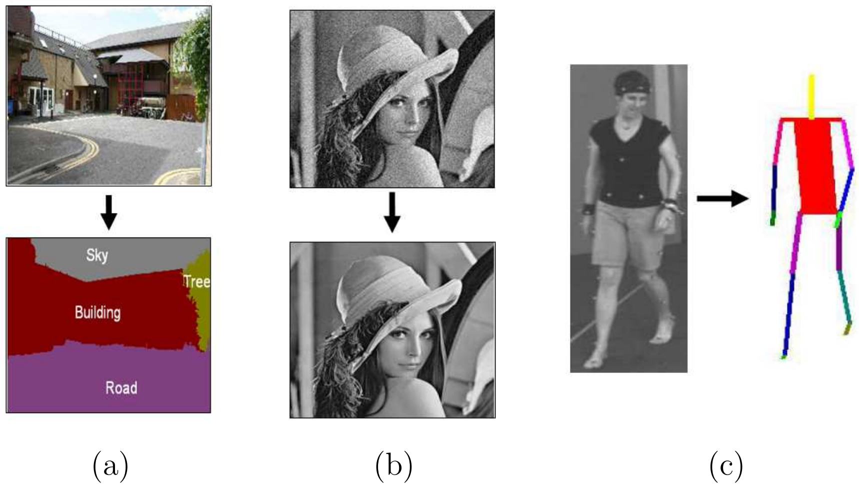


Figure 1.1: *Some labelling problems in computer vision. (a) Object segmentation and recognition: Given any image, we want to find out which object each pixel in the image belongs to. There is one discrete random variable for each pixel in the image which can take any value from a set $\mathcal{L}$ of object labels. For instance, we can use the object set {road, building, tree, sky}. (b) Image denoising: Given a noisy image of the scene, we want to infer the true colour of each pixel in the image. The problem is formulated in a manner similar to object segmentation. Again we use one discrete random variable per pixel which can take any value in* RGB *space. (c) Human pose estimation: Given an image, we want to infer the pose of the human visible in it. The problem is formulated using a vector of continuous pose variables which encode the orientation and different joint angles of the human.*

A random field is said to be a Markov random field (MRF) with respect to a neighbourhood system $\mathcal{N} = \{\mathcal{N}_v | v \in \mathcal{V}\}$ if and only if it satisfies the positivity property: $\Pr(\mathbf{x}) > 0 \;\; \forall \mathbf{x} \in \mathcal{X}^n$, and the Markovian property:

$$\Pr(x_v | \{x_u : u \in \mathcal{V} - \{v\}\}) = \Pr(x_v | \{x_u : u \in \mathcal{N}_v\}) \qquad \forall v \in \mathcal{V}. \tag{1.2.1}$$

Here we refer to $\Pr(X = \mathbf{x})$ by $\Pr(\mathbf{x})$ and $\Pr(X_i = x_i)$ by $\Pr(x_i)$. The pairwise MRF commonly used to model image labelling problems is shown in figure 3.2.

A conditional random field (CRF) may be viewed as an MRF globally conditioned on the data. The conditional distribution $\Pr(\mathbf{x}|\mathbf{D})$ over the labellings of the CRF is a *Gibbs* distribution and can be written in the form:

$$\Pr(\mathbf{x}|\mathbf{D}) = \frac{1}{Z} \exp(-\sum_{c \in \mathcal{C}} \psi_c(\mathbf{x}_c)), \tag{1.2.2}$$

where Z is a normalizing constant known as the partition function, and $\mathcal{C}$ is the set of all cliques [61]. The term $\psi_c(\mathbf{x}_c)$ is known as the potential function of the

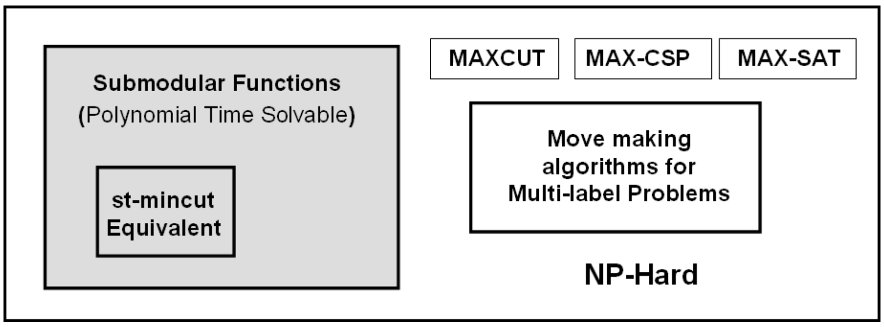

Submodular Functions
(Polynomial Time Solvable)
st-mincut Equivalent
MAXCUT
MAX-CSP
MAX-SAT
Move making algorithms for Multi-label Problems
NP-Hard

Figure 1.2: *Function minimization problems.*

clique c where $\mathbf{x}_c = \{x_i, i \in c\}$ is the vector encoding the labelling of the variables constituting the clique. The corresponding Gibbs energy is given by

$$E(\mathbf{x}) = -\log \Pr(\mathbf{x}|\mathbf{D}) - \log Z = \sum_{c \in \mathcal{C}} \psi_c(\mathbf{x}_c) \tag{1.2.3}$$

The most probable or maximum a posteriori (MAP) labelling $\mathbf{x}^*$ of the random field is defined as

$$\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathbf{L}} \Pr(\mathbf{x}|\mathbf{D}). \tag{1.2.4}$$

and can be found by minimizing the energy function E. This equivalence to MAP inference has made discrete energy minimization extremely important for problems which are solved using probabilistic methods.

1.2.2 Discrete Energy Minimization

Minimizing a discrete function is one of the core problems of optimization. Many combinatorial problems such as MAXCUT and constraint satisfaction (CSP) can be formulated in this manner.

Although minimizing a function is NP-hard in general, there exist families of energy functions for which this could be done in polynomial time. Submodular set functions constitute one such well studied family. The algorithms for minimizing general functions belonging to this class of functions have high runtime complexity. This characteristic renders them useless for most computer vision problems which involve large number of random variables. Functions belonging to certain subclasses of submodular functions can be solved relatively easily i.e. are less computationally expensive to minimize. For instance, certain families of functions can be minimized by solving a st-mincut problem for which fast and efficient algorithms are available [10, 25, 39, 88].

Many problems in computer vision result in energy functions which are not submodular, and are NP-hard to minimize. A number of approximate minimization algorithms have been proposed for solving these functions. α-expansion and $\alpha\beta$-swap [11] are two such algorithms. They are widely used to minimize energy functions involving multi-valued discrete variables. The different types of function minimization problems are shown in figure 1.2.

1.2.3 Submodular Functions

Submodular set functions are encountered in many areas of research. They are particularly useful in combinatorial optimization, probability and geometry [29,67]. Many optimization problems relating to submodular functions can be solved efficiently. In some respects they are similar to convex/concave functions encountered in continuous optimization.

Consider the set $N = \{1, 2, \ldots, n\}$. A set function $f_s : 2^N \to \mathbb{R}$ is said to be submodular if and only if for all subsets $A, B \subseteq N$ the function satisfies:

$$f_s(A) + f_s(B) \geq f_s(A \cup B) + f_s(A \cap B). \tag{1.2.5}$$

Every set function $f_s : 2^N \to \mathbb{R}$ can be written in terms of a function of binary variables $f_b : \{0, 1\}^n \to \mathbb{R}$. For each element i in set N, a corresponding binary variable $X_i \in \{0, 1\}$ is needed for this representation. Any subset G of the set N can be represented by a labelling of the binary variables. For instance, if an element i is in the subset G then the corresponding binary variables X_i will take value 1. This will be made clearer by the following example.

Example 1 *Consider a set function f_s defined over the set $N = \{1, 2\}$. As the set N contains 2 elements, the corresponding function (f_b) of binary variables takes two binary variables X_1 and X_2 as arguments. Under the above defined encoding scheme, the subset $G = \{2\}$ will be represented by the labelling $(X_1, X_2) = (0, 1)$. Similarly, the empty subset $G = \emptyset$ will result in the labelling: $(X_1, X_2) = (0, 0)$. The submodularity condition (1.2.5) for two particular subsets $A = \{1\}$ and $B = \{2\}$ of G is:*

$$f_s(\{1\}) + f_s(\{2\}) \geq f_s(\{1, 2\}) + f_s(\emptyset). \tag{1.2.6}$$

This condition for the equivalent binary function f_b becomes:

$$f_b(1, 0) + f_b(0, 1) \geq f_b(1, 1) + f_b(0, 0). \tag{1.2.7}$$

We will now extend the definition of submodularity to functions of binary variables. For this however, we will first need to define the concept of a projection of a function.

Definition 1 *A projection of a function $f : \mathcal{L}^n \to \mathbb{R}$ on s variables is a function $f^p : \mathcal{L}^s \to \mathbb{R}$ which is obtained by fixing the values of $n - s$ arguments of $f(\cdot)$. Here p refers to the set of variables whose values have been fixed.*

Example 2 *The function $f^{\{x_1=0\}}(x_2, \ldots, x_n) = f(0, x_2, \ldots, x_n)$ is a projection of the function $f(x_1, x_2, \ldots, x_n)$.*

Definition 2 *A function of one binary variable is always submodular. A function $f(x_1, x_2)$ of two binary variables $\{x_1, x_2\}$ is submodular if and only if:*

$$f(0,1) + f(1,0) \geq f(0,0) + f(1,1) \tag{1.2.8}$$

A function $f : \mathcal{L}^n \to R$ is submodular if and only if all its projections on 2 variables are submodular [8, 54].

The definition of submodularity can also be extended to functions of multi-valued variables (referred to as multi-label functions). However, this requires the existence of an ordering over the labels that each variable can take.

Definition 3 *Let $\mathcal{L}$ be a completely ordered set of labels where the label l_{i+1} is above label l_i. A second order multi-label function $f : \mathcal{L}^2 \to R$ is submodular if*

$$f(l_1, l_2) - f(l_1 + 1, l_2) - f(l_1, l_2 + 1) + f(l_1 + 1, l_2 + 1) \leq 0. \tag{1.2.9}$$

1.2.4 Minimizing Submodular Functions

Many problems in combinatorial optimization can be formulated as minimizing a submodular set function. The first strongly polynomial time algorithm for this problem was given independently by [42] and [90]. These algorithms had high runtime complexity. Although recent work has been partly successful in reducing the complexity of algorithms for general submodular function minimization, they are still quite computationally expensive and cannot be used to minimize large problems. For instance, the current best algorithm for general submodular function minimization has complexity $O(n^5 Q + n^6)$ where Q is the time taken to evaluate the function [71]. This algorithm improved upon the previous best strongly polynomial time algorithm by a factor of $n \log n$.

Certain submodular functions can be minimized by solving an st-mincut problem [8]. Specifically, all submodular functions of binary variables of order at most 3 can be minimized in this manner [54]. Researchers have shown that certain higher order functions can be transformed into submodular functions of order 2, and thus can also be minimized [27]. The same transformation technique can be used to minimize some functions of multi-valued variables [25, 39, 88].

1.3. Graph Cuts for Energy Minimization

Graph cuts have been extensively used in computer vision to compute the maximum a posteriori (MAP) solutions for various discrete pixel labelling problems such as image restoration, segmentation, voxel occupancy and stereo [9, 12, 40, 41, 53, 82, 116]. Greig *et al.* [32] were one of the first to use graph cuts in computer vision. They showed that if the pairwise potentials of a two label *pairwise* MRF were defined as an Ising model, then its exact MAP solution can be obtained in polynomial time by solving a st-mincut problem.

One of the primary reasons behind the growing popularity of graph cuts is the availability of efficient algorithms for computing the maximum flow (max-flow) in graphs of arbitrary topology [2, 10]. These algorithms have low polynomial runtime complexity, and enable fast computation of the minimum cost st-cut (st-mincut) problem. This in turn allows for the computation of globally optimal solutions for important classes of energy functions which are encountered in many vision problems [48, 54]. Even in problems where they do not guarantee globally optimal solutions, these algorithms can be used to find solutions which are strong local minima of the energy [11, 46, 55, 109]. These solutions for certain problems have been shown to be better than the ones obtained using other inference methods [104].

1.3.1 The st-Mincut Problem

In this section we provide a general overview of the st-mincut/maxflow problem and give the graph notation used in this dissertation. A directed weighted graph $G(V, E, C)$ with non-negative edge weights, is defined by a set of nodes V, a set of directed edges E, and an edge cost function $C : E \to \mathbb{R}$ which maps each edge (i, j) of the graph to a real number c_{ij}[1]. We will use n and m to denote the number of nodes $|V|$ and the number of edges $|E|$ in the graph respectively. Graphs used in the st-mincut problem have certain special nodes called the terminal nodes, namely the source s, and the sink t. The edges in the graph can be divided into two disjoint categories: t-edges which connect a node to a terminal node, and n-edges which connect nodes other than the terminal nodes with each other. We make the following assumptions in our notation: $(i, j) \in E \Rightarrow (j, i) \in E$, and $(s, i) \in E \ \wedge \ (i, t) \in E$ for all $i \in V$. These assumptions are non-restrictive as edges with zero edge weights are allowed in our formulation. Thus we can

[1]We will restrict our attention to edge cost functions of the form $C : E \to \mathbb{R}^{+} \cup \{0\}$.

conform to our notation without changing the problem.

A *cut* is a partition of the node set V into two parts S and $\overline{S} = V - S$, and is defined by the set of edges (i, j) such that $i \in S$ and $j \in \overline{S}$. The cost of the cut $(S, \overline{S})$ is given as:

$$C_{S,\overline{S}} = \sum_{i \in S, j \in \overline{S}} c_{ij}. \tag{1.3.1}$$

An st-cut is a cut satisfying the properties $s \in S$ and $t \in \overline{S}$. Given a directed weighted graph G, the st-mincut problem is that of finding a st-cut with the smallest cost. By the Ford-Fulkerson theorem [26], this is equivalent to computing the maximum flow from the source to the sink with the capacity of each edge equal to c_{ij} [2].

1.3.1.1 Formulating The Max-Flow Problem

For a network $G(V, E)$ with a non-negative capacity c_{ij} associated with each edge, the max-flow problem is to find the maximum flow f from the source node s to the sink node t subject to the edge capacity (1.3.2) and mass balance (1.3.3) constraints:

$$0 \leq f_{ij} \leq c_{ij} \quad \forall (i, j) \in E, \quad \text{and} \tag{1.3.2}$$

$$\sum_{i \in \ N(j)} (f_{ji} - f_{ij}) = 0 \quad \forall j \in V \tag{1.3.3}$$

where f_{ij} is the flow from node i to node j and $N(j)$ is the neighbourhood of node j i.e. $N(j)$ consists of all nodes connected by an edge to j [2].

Observe that we can initialize the flows in the t-edges of any node i of the graph as $f_{si} = f_{it} = \min(c_{si}, c_{it})$. This corresponds to pushing flow through these edges from the source to the sink and has no effect on the final solution of the st-mincut problem. From this it can be deduced that the solution of the st-mincut problem is invariant to the absolute value of the terminal edge capacities c_{si} and c_{it}. It only depends on the difference of these capacities $(c_{it} - c_{si})$. Adding or subtracting a constant to these capacities changes the objective function by a constant and does not affect the overall st-mincut solution as can be seen in figure 1.3. Such transformations result in a reparameterization of the graph and will be explained later in chapter 2.

1.3.1.2 Augmenting Paths, Residual Graphs

Given a flow f_{ij}, the residual capacity r_{ij} of an edge $(i, j) \in E$ is the maximum additional flow that can be sent from node i to node j using the edges (i, j) and (j, i) or formally $r_{ij} = c_{ij} - f_{ij} + f_{ji}$. A residual graph $G(f)$ of a weighted graph G consists of the node set V and the edges with positive residual capacity (with

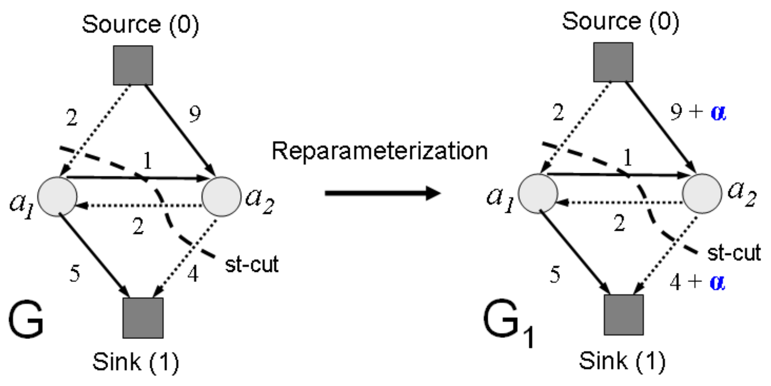


Figure 1.3: *Graph reparameterization. The figure shows a graph G, and its reparameterization G_1 obtained by adding a constant α to both the t-edges of node a_2. The edges included in the st-mincut are depicted by dotted lines. Observe that although the cost of the st-mincut in G and G_1 is different, the st-mincut includes the same edges for both graphs and thus induces the same partitioning of the graph.*

respect to the flow f). An augmenting path is a path from the source to the sink along unsaturated edges of the residual graph.

1.3.2 Minimizing Submodular Functions using Graph Cuts

The problem of finding the minimum cost st-cut (st-mincut) in any graph can be written in terms of minimizing a sum of functions defined on individual and pairs of binary variables. Conversely, any submodular function of binary or *boolean* variables which can be written as a sum of unary and pairwise terms can be minimized by finding the st-mincut in the corresponding graph. In this dissertation, we will call functions of this form 'second order functions' or 'functions of order 2'.

Definition 4 *We say that a function $f : \mathcal{L}^n \to \mathbb{R}$ is of order k if it can be written in terms of a sum of functions $f_i : \mathcal{L}^k \to \mathbb{R}$, each of which is defined on at most k variables.*

In the above definition we use the function representation which leads to the smallest order.

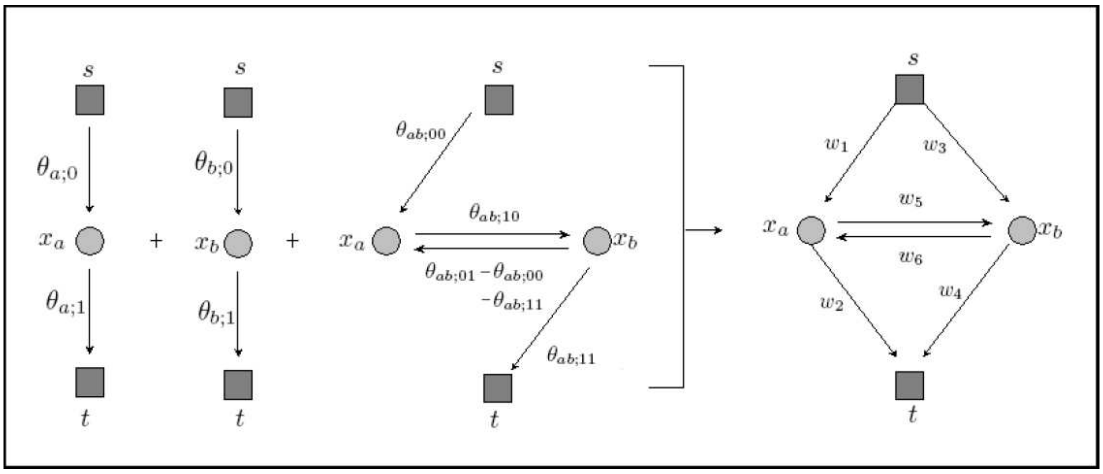


Figure 1.4: *Energy minimization using graph cuts. The figure shows how individual unary and pairwise terms of an energy function taking two binary variables are represented and combined in the graph. Multiple edges between the same nodes are merged into a single edge by adding their weights. For instance, the cost w_1 of the edge (s, x_a) in the final graph is equal to: $w_1 = \theta_{a;0} + \theta_{ab;00}$. The cost of a st-cut in the final graph is equal to the energy $E(\mathbf{x})$ of the configuration $\mathbf{x}$ the cut induces. The minimum cost st-cut induces the least energy configuration $\mathbf{x}$ for the energy function.*

Example 3 *The function $f^a(x_1, x_2, x_3) = 4x_1 + 5x_2\overline{x}_3 + 3\overline{x}_2$ is of order 2 because of the maximal order term $5x_2\overline{x}_3$. Similarly, the function*

$$f^a(x_1, x_2, x_3) = 4x_1 + 5\overline{x}_1 x_2 \overline{x}_3 \tag{1.3.4}$$

is of order 3 because of the maximal term $5\overline{x}_1 x_2 \overline{x}_3$.

Algorithms for finding the st-mincut require that all edges in the graph have non-negative weights. This condition results in a restriction on the class of energy functions that can be solved in this manner. For instance, binary second order functions can be minimized by solving a st-mincut problem only if they are submodular [54].

We will now show how second order functions of binary variables (also referred as Pseudo boolean functions) can be minimized by solving a st-mincut problem. The procedure for energy minimization using graph cuts comprises of building a graph in which each st-cut defines a configuration $\mathbf{x}$. The cost of an st-cut is equal to the energy $E(\mathbf{x}|\theta)$ of its corresponding configuration $\mathbf{x}$. Finding the minimum cost st-cut in this graph thus provides us with the configuration having the least energy. Kolmogorov and Zabih [54] described the procedure to construct graphs for minimizing pseudo-boolean functions of order at most 3. The graph constructions for functions of multi-valued variables were given later by [39] and [88].

We now explain the graph construction for minimizing energies involving binary random variables. We use the notation of [51] and write a second order function as:

$$E(\mathbf{x}|\theta) = \theta_{\text{const}} + \sum_{v \in V, i \in \mathcal{L}} \theta_{v;i} \delta_i(x_v) + \sum_{(u,v) \in E, (j,k) \in \mathcal{L}^2} \theta_{uv;jk} \delta_j(x_u) \delta_k(x_v), \quad (1.3.5)$$

where $\theta_{v;i}$ is the penalty for assigning label i to latent variable x_v, $\theta_{uv;ij}$ is the penalty for assigning labels i and j to the latent variables x_u and x_v respectively. Further, each $\delta_j(x_v)$ is an indicator function, which is defined as:

$$\delta_j(x_v) = \begin{cases} 1 & \text{if } x_v = j, \\ 0 & \text{otherwise.} \end{cases}$$

Functions of binary variables (pseudo-boolean functions) can be written as:

$$\begin{aligned} E(\mathbf{x}|\theta) &= \theta_{\text{const}} + \sum_{v \in V} (\theta_{v;1} x_v + \theta_{v;0} \overline{x}_v) \\ &\quad + \sum_{(u,v) \in E} (\theta_{st;11} x_u x_v + \theta_{st;01} \overline{x}_u x_v + \theta_{st;10} x_u \overline{x}_v + \theta_{st;00} \overline{x}_u \overline{x}_v) \quad (1.3.6) \end{aligned}$$

The individual unary and pairwise terms of the energy function are represented by weighted edges in the graph. Multiple edges between the same nodes are merged into a single edge by adding their weights. The graph construction for a two variable energy function is shown in figure 1.4. The constant term θ_{const} of the energy does not depend on $\mathbf{x}$ and thus is not considered in the energy minimization procedure. The st-mincut in this graph provides us with the minimum solution $\mathbf{x}^*$. The cost of this cut corresponds to the energy of the solution $E(\mathbf{x}^*|\theta)$. The labelling of a latent variable depends on the terminal it is disconnected from by the minimum cut. In our notation, if the node is disconnected from the source, we assign it the label zero and one otherwise.

1.4. Minimizing Non-submodular Functions

Up until now we have not addressed the problem of minimizing non-submodular functions. For certain functions belonging to this class, specifically, those defined on trees or graphs with only one cycle, it is possible to compute the exact globally optimal solution in polynomial time[1]. However, minimizing a non-submodular

[1]Graphs containing multiple loops can be transformed into trees. The tree resulting from such a conversion has sets of vertices of the original graph as its nodes. The tree width of a

function is in general a NP-hard problem. Many problems in computer vision give rise to such functions and making progress towards their solution is of paramount importance. A number of algorithms have been proposed in the literature for solving this problem. These methods can be divided into two broad categories: 1) methods which work by minimizing a related submodular function [7, 77, 85], and 2) methods which solve a relaxation of the integer programming problem arising from the function minimization problem [14, 112].

1.4.1 LP Relaxation of the Integer Program

General discrete energy minimization can be seen as an integer programming problem [14]. The integer program (IP) is formulated with binary variables $\mathbf{y}$ [112]. We use the formulation introduced in [13] and later used in [55] for the metric labelling problem. The integer program for the labelling problem can be written as:

$$\min_{\mathbf{y}} \left(\theta_{\text{const}} + \sum_{v \in V, i \in \mathcal{L}} \theta_{v;i} y_{v,i} + \sum_{(u,v) \in E, (j,k) \in \mathcal{L}^2} \theta_{uv;jk} y_{uv,ij} \right), \tag{1.4.1}$$

$$\begin{aligned}
s.t. \quad & \textstyle\sum_{i \in \mathcal{L}} y_{v,i} = 1 && \forall v \in \mathcal{V} && (1.4.2)\\
& \textstyle\sum_{i \in \mathcal{L}} y_{uv,ij} = y_{v,j} && \forall j \in \mathcal{L}, \forall (u,v) \in \mathcal{E} && (1.4.3)\\
& \textstyle\sum_{j \in \mathcal{L}} y_{uv,ij} = y_{u,i} && \forall i \in \mathcal{L}, \forall (u,v) \in \mathcal{E} && (1.4.4)\\
& y_{v,i}, y_{uv,ij} \in \{0,1\} && \forall v \in \mathcal{V}, \forall (u,v) \in \mathcal{E}, i, j \in \mathcal{L}
\end{aligned}$$

The $\{0,1\}$ variable $y_{v,i}$ of the IP indicates whether variable X_v has been assigned label i or not. For instance, if X_v is assigned label k then $y_{v,k} = 1$ and $y_{v,i} = 0$ for all other values i in $\mathcal{L}$. Similarly, $y_{uv,ij} = 1$ indicates that variable X_u is assigned label i and variable X_v is assigned label j. The reader should note that the variables $y_{st,ij}$ and $y_{ts,ji}$ indicate the exact same thing. The IP formulated above is NP-hard to solve exactly. We can relax the $\{0,1\}$ constraints in the problem to $y_{s,i} > 0$, $y_{st,ij} > 0$ to get a simple linear program which can be solved using any LP solver like simplex or barrier function methods [44, 89]. The relaxation of the integrality constraints may result in a fractional solution of the optimization problem. This solution is typically converted to a valid integer solution using various rounding schemes [14, 51].

graph is the size of the biggest set in this tree minus one. This conversion procedure provides us with an algorithm to minimize non-submodular functions defined on general graphs whose complexity is exponential in the tree width of the graph.

The linear programs which arise in computer vision problems contain many variables and are thus computationally expensive to solve using the general methods mentioned above. A number of algorithms have been developed [51, 55, 112, 115] which attempt to solve the LP relaxation by exploiting the special structure of the problem. In particular, researchers have proposed a number of message passing algorithms which are able to approximately solve large LP problems extremely efficiently [51, 112].

1.4.2 Partial Optimality

Some algorithms for the minimization of non-submodular functions return a partial solution $\mathbf{x} \in \{\mathcal{L} \cup \epsilon\}^n$ of the energy [6, 8, 57, 84]. The assignment $x_i = \epsilon$ implies that no label has been assigned to random variable X_i. For instance, the QPBO algorithm [8,76,84] for minimizing energy functions of binary variables returns a partially labelled solution $\mathbf{x}$ with the property that there exists a global minimum $\mathbf{x}^*$ of the energy function such that $x_i = x_i^*$ for all labelled variables X_i. This property of the solution is called *weak persistency*. There are certain partial solutions of the energy for which a *stronger* condition called *strong persistency* holds true. This property states that if a variable X_i is labelled, then it is assigned the same label in all global minima $\mathbf{x}^*$ of the energy, i.e. $x_i = x_i^*$ for all $\mathbf{x}^* \in \{\arg\min_{\mathbf{x}} E(\mathbf{x})\}$.

It has been shown that tree-reweighted message passing gives a part of an optimal solution when applied to functions of binary variables [112]. A method for finding partially optimal solutions of multi-label energy functions was recently proposed by [57]. The key step of this algorithm is the construction of a submodular subproblem $\mathcal{P}_k$ for each label $l_k \in \mathcal{L}$. It was shown that the solution of the subproblem $\mathcal{P}_k$ can be used to isolate variables which are assigned label l_k in all the globally optimal solutions of the energy function.

1.4.3 Summary

In this chapter we have given a brief overview of the energy minimization problem. We have explained how vision problems can be formulated in terms of minimizing a function, and reviewed the commonly used methods for function minimization. The family of submodular functions was introduced in the chapter. We explained the importance of submodular functions in discrete optimization, and listed some of their properties. We also explained how second order submodular functions can be minimized by solving a st-mincut problem. In the next chapter, we explain how this procedure can be made more efficient by reusing computation from previous problem instances.

Chapter 2

Minimizing Dynamic Energy Functions

2.1. Dynamic Graph Cuts

In many real world applications, multiple similar instances of a problem need to be solved sequentially e.g. performing image segmentation on the frames of a video. The data (image) in this problem changes from one time instance to the next. Given the solution to an instance of the problem, the question arises as to whether this solution can help in solving other similar instances. In this chapter we answer this particular question positively for functions that can be minimized exactly using graph cuts. Specifically, we show how the maxflow solution corresponding to an energy minimization problem can be used for efficiently minimizing another *similar* function with slightly different energy terms.

Our algorithm records the flow obtained during the computation of the max-flow corresponding to a particular problem instance. This recorded flow is used as an initialization in the process of finding the max-flow solution corresponding to the new problem instance (as seen in figure 2.1). Our method belongs to a broad category of algorithms which are referred to as *dynamic*. These algorithms solve a problem by dynamically updating the solution of the previous problem instance. Their goal is to be more efficient than a recomputation of the solution after every change from scratch. Given a directed weighted graph, a *fully* dynamic algorithm should allow for unrestricted modification of the graph. The modification may involve addition and deletion of nodes and edges in the graph as well as changes in the cost (capacity) of any graph edge.

2.1.1 Dynamic Computation

Dynamic algorithms are not new to computer vision. They have been extensively used in computational geometry for problems such as range searching, point location, convex hull, proximity and many others. For more on dynamic algorithms used in computational geometry, the reader is referred to [15]. A number of algorithms have been proposed for the dynamic mincut problem. Thorup [105] proposed a method which had a $O(\sqrt{m})$ update time and took $O(\log n)$ time per edge to list the cut edges. Here n and m denote the number of nodes and edges in the graph respectively. However, the dynamic *st-mincut* problem has remained relatively ignored.

Gallo *et al.* [30] introduced the problem of parametric max-flow and used a partially dynamic graph cut algorithm for the problem. Their algorithm had a low polynomial time complexity but was unable to handle arbitrary changes in the

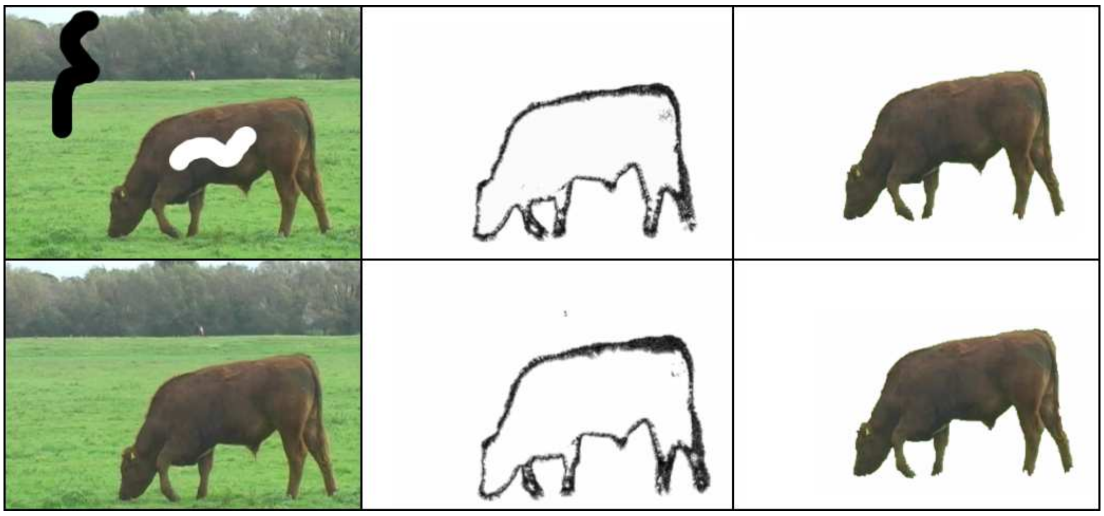


Figure 2.1: *Dynamic image segmentation using graph cuts. The images in the first column are two consecutive frames of the grazing cow video sequence. Their respective segmentations are shown in the third column. The first image in the first column also shows the user segmentation seeds (pixels marked by black (background) and white (foreground) colours). The user marked image pixels are used to learn histograms modelling foreground and background appearance (as in [9]). These histograms are used to compute a likelihood for each pixel belonging to a particular label. This likelihood is incorporated in the* CRF *used for formulating the image segmentation problem. The optimal segmentation solution (shown in column 3) is obtained by minimizing the energy function corresponding to the* CRF*. In column 2, we observe the n-edge flows obtained while minimizing the energy functions using graph cuts. It can be clearly seen that the flows corresponding to the two segmentations are similar. The flows from the first segmentation were used as an initialization for the max-flow problem corresponding to the second frame. The time taken for this procedure was much less than that taken for finding the flows from scratch.*

graph. Recently, Cohen and Tamassia [16] proposed a dynamic algorithm for the problem by showing how dynamic expression trees can be used for maintaining st-mincuts with $O(\log m)$ time for update operations. However, their algorithm could only handle series-parallel diagraphs[1].

Boykov and Jolly [9] were the first to use a *partially* dynamic st-mincut algorithm in a vision application by proposing a technique with which they could update capacities of *certain* graph edges, and recompute the st-mincut dynamically. They used this method for performing interactive image segmentation, where the user could improve segmentation results by giving additional segmentation cues (seeds) in an online fashion. Specifically, they described a method for updating the cost of t-edges in the graph. In this chapter we present a new fully dynamic algorithm for the st-mincut problem which allows for arbitrary changes in the graph[2]. Recently, Juan and Boykov [43] proposed an algorithm in which instead of *reusing flow*, they used the st-mincut solution corresponding to the previous problem for solving a new st-mincut problem.

2.1.2 Outline of the Chapter

An outline of the chapter follows. The relationship between energy and graph reparameterization is explained in section 2.2. Section 2.3 shows how exact st-mincut solutions of dynamically changing graphs can be efficiently computed by reusing flow. Specifically, it describes how the residual graph can be transformed to reflect the changes in the original graph using graph reparameterization, and discusses issues related to the computational complexity of the algorithm. In section 2.4, we describe how the process of recomputing the st-mincut/max-flow can be further optimized by using *recycled* search trees.

2.2. Energy and Graph Reparameterization

We will now explain the concept of graph reparameterization which will be used later to show how we can minimize dynamic energy functions.

Recall from equation 1.3.6 that a second order energy function can be written

[1]Series-Parallel digraphs are graphs which are planar, acyclic and connected.

[2]An earlier version of this chapter appeared as [48].

in terms of an energy parameter vector θ as:

$$\begin{aligned} E(\mathbf{x}|\theta) &= \theta_{\text{const}} + \sum_{v \in V}(\theta_{v;1}x_v + \theta_{v;1}\overline{x}_v) \\ &\quad + \sum_{(u,v)\in E}(\theta_{st;11}x_u x_v + \theta_{st;01}\overline{x}_u x_v + \theta_{st;10}x_u\overline{x}_v + \theta_{st;00}\overline{x}_u\overline{x}_v) \end{aligned} \tag{2.2.1}$$

Two energy parameter vectors θ_1 and θ_2 are called reparameterizations of each other if and only if $\forall \mathbf{x}, E(\mathbf{x}|\theta_1) = E(\mathbf{x}|\theta_2)$ [8, 51, 88, 112]. This definition simply means that all possible labellings $\mathbf{x}$ have the same energy under both parameter vectors θ_1 and θ_2, and does not imply that $\theta_1 = \theta_2$. There are a number of transformations which can be applied to an energy parameter vector θ to obtain its reparameterization $\overline{\theta}$. For instance the transformations given as:

$$\forall i \quad \overline{\theta}_{v;i} = \theta_{v;i} + \alpha, \quad \overline{\theta}_{const} = \theta_{const} - \alpha \text{ and} \tag{2.2.2}$$

$$\forall i, j \quad \overline{\theta}_{st;ij} = \theta_{st;ij} + \alpha, \qquad \overline{\theta}_{const} = \theta_{const} - \alpha \tag{2.2.3}$$

result in the reparameterization of the energy parameter vector.

As both parameters θ and $\overline{\theta}$ define the same energy function, the minimum energy labelling for both will be the same i.e.

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} E(\mathbf{x}|\theta_1) = \arg\min_{\mathbf{x}} E(\mathbf{x}|\theta_2) \tag{2.2.4}$$

This in turn implies that the graphs constructed for minimizing the energy functions $E(\mathbf{x}|\theta_1)$ and $E(\mathbf{x}|\theta_2)$ (using the procedure explained in the previous chapter) will have the same st-mincut. We call these graphs reparameterizations of each other. For any transformation of the energy function which results in such a *reparameterization* we can derive a corresponding transformation for a graph. Under these transformations the resulting graph will be a reparameterization of the original graph and thus will have the same st-mincut. The graph transformations corresponding to energy transformations given by equations (2.2.2) and (2.2.3) are shown in figure 2.2.

The transformations given above are not the only way to obtain a reparameterization. In fact pushing flow through any path in the graph can be seen as performing a valid transformation. The residual graph resulting from this flow is a reparameterization of the original graph where no flow was being passed. This can be easily observed from the fact that the residual graph has the same st-mincut as the original graph, albeit with a different cost. In the next section we show how the property of graph reparameterization can be used for updating the residual graph when the original graph has been modified and the st-mincut needs to be recomputed.

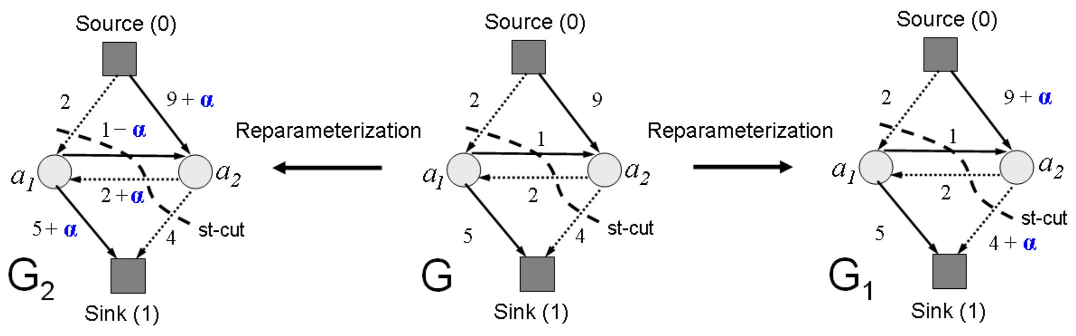


Figure 2.2: *Graph reparameterization. The figure shows a graph G, its two reparameterizations G_1 and G_2 along with their respective st-mincuts. The edges included in the st-mincut are marked by dotted lines. The reparameterized graphs G_1 and G_2 are a results of two different valid transformations of graph G. It can be clearly seen that reparameterized graphs G_1 and G_2 have the same st-mincut as graph G.*

2.3. Recycling Computation

We now show how the max-flow solution obtained while minimizing an energy function can be used to efficiently minimize other *similar* energy functions.

Consider two energy functions E_a and E_b which differ by a few terms. As we have seen in the previous chapter, this implies that the graph G_b representing energy E_b differs from that representing energy E_a (G_a) by a few edge costs. Suppose we have found the optimal solution of E_a by solving the max-flow problem on the graph G_a and now want to find the solution of E_b. Instead of following the conventional procedure of recomputing the max-flow on G_b from scratch, we perform the computation by reusing the flows obtained while minimizing E_a.

Boykov and Jolly [9], in their work on interactive image segmentation used this technique for efficiently recomputing the MAP solution when only the unary terms of the energy function change (due to addition of new hard and soft constraints by the user). However, they did not address the problem of handling changes in the pairwise terms of the energy function which result in changes in the cost of the n-edges of the graph. Our method (explained below) can handle arbitrary changes in the graph.

2.3.1 Updating Residual Graphs

The flows through a graph can be used to generate a residual graph (as explained in chapter 1). Our algorithm works by updating the residual graph obtained from the max-flow computation in graph G_a to make it represent G_b. This is done by reducing or increasing the residual capacity of an edge according to the change made to its cost going from G_a to G_b.

Recall from equation (1.3.2) that the flow in an edge of the graph has to satisfy the edge capacity constraint:

$$0 \leq f_{ij} \leq c_{ij} \quad \forall (i,j) \in E. \tag{2.3.1}$$

While modifying the residual graph, certain flows may violate the new edge capacity constraints (2.3.1). This is because flow in certain edges might be greater than the capacity of those edges under G_b. To make these flows consistent with the new edge capacities, we reparameterize the updated graph (using reparameterizations described in the previous section) to make sure that the flows satisfy the edge capacity constraints (2.3.1) of the graph. The max-flow is then computed on this reparameterized graph. This gives us the st-mincut solution of graph G_b, and hence the global minimum solution of energy E_b.

We now show how the residual graph is transformed to make sure that all edge capacity constraints are satisfied. We use the two graph transformations given in section 2.2 to increase the capacities of edges in G_b in which the flow exceeds the true capacity. These transformations lead to a reparameterization of the graph G_b. We can then find the st-mincut on this reparameterized graph to get the st-mincut solution of graph G_b.

The various changes that might occur to the graph going from G_a to G_b can be expressed in terms of changes in the capacity of t-edges and n-edges of the graph. The methods for handling these changes will be discussed now. We use c'_{si} to refer to the new edge capacity of the edge (s,i). r'_{si} and f'_{si} are used to represent the updated residual capacity and flow of the edge (s,i) respectively.

Modifying t-edge Capacities Our method for updating terminal or t-edges is similar to the one used in [9] and is described below.

The updated residual capacity of an edge (s,i) can be computed as:

$$r'_{si} = r_{si} + c'_{si} - c_{si}. \tag{2.3.2}$$

This can be simplified to: $r'_{si} = c'_{si} - f_{si}$. If the flow f_{si} is greater than the updated edge capacity c'_{si}, it violates the edge capacity constraint (2.3.1) resulting in r'_{si} becoming negative. To make the flow consistent a constant $\gamma = f_{si} - c'_{si}$ is added to the capacity of both the t-edges $\{(s,i),(i,t)\}$ connected to the node i. As has been observed in section 2.2 and in [9], this transformation is an example of graph reparameterization which does not change the minimum cut (its cost changes but not the cut itself). For an illustration see figure 1.3. The residual capacities thus become: $r^{'}_{si} = c^{'}_{si} - f_{si} + \gamma = 0$ and, $r^{'}_{it} = c_{it} - f_{it} + \gamma$, or $r^{'}_{it} = r_{it} - c^{'}_{si} + f_{si}$.

Modifying n-edge Capacities We now describe how the residual graph is updated when n-edge capacities are changed. Observe that updating edge capacities in the residual graph is simple if the new edge capacity $c^{'}_{ij}$ is greater than or equal to the old edge capacity c_{ij}. This operation involves addition of extra capacity and thus the flow cannot become inconsistent. The updated residual capacity $r^{'}_{ij}$ is obtained as:

$$r^{'}_{ij} = r_{ij} + (c^{'}_{ij} - c_{ij}). \tag{2.3.3}$$

Even if $c^{'}_{ij}$ is less than c_{ij}, the procedure still remains trivial if the flow f_{ij} is less than the new edge capacity $c^{'}_{ij}$. This is due to the fact that the reduction in the edge capacity does not affect the flow consistency of the network i.e., flow f_{ij} satisfies the edge capacity constraint (2.3.1) for the new edge capacity. The residual capacity of the edge can still be updated according to equation (2.3.3). The difference in this case is that $(c^{'}_{ij} - c_{ij})$ is negative and hence will result in the reduction of the residual capacity. In both these cases, the flow through the edge remains unchanged i.e. $f^{'}_{ij} = f_{ij}$.

The problem becomes complex when the new edge capacity $c^{'}_{ij}$ is less than the flow f_{ij}. In this case, f_{ij} violates the edge capacity constraint (2.3.1). To make f_{ij} consistent, we have to retract the excess flow (f_{ij} - $c^{'}_{ij}$) from the edge (i,j). At this point, the reader should note that a trivial solution for this operation would be to push back the flow through the augmenting path it originally came through. However such an operation would be extremely computationally expensive. We now show how we resolve this inconsistency in constant i.e. O(1) time.

The inconsistency arising from excess flow through edge (i,j) can be resolved by a single valid transformation of the residual graph. This transformation is the same as the one shown in figure 1.3 for obtaining graph G_2 from G, and does not change the st-mincut. It leads to a reparameterization of the residual graph which has non-negative residual capacity for the edge (i,j). The transformation involves adding a constant $\alpha = f_{ij} - c^{'}_{ij}$ to the capacity of edges (s,i), (i,j), and (j,t) and subtracting it from the residual capacity of edge (j,i). The residual capacity r_{ji} of edge (j,i) is greater than the flow f_{ij} passing through edge (i,j). As α is always less than f_{ij} the above transformation does not make the residual capacity of edge (j,i) negative. The procedure for restoring consistency is illustrated in figure 2.3.

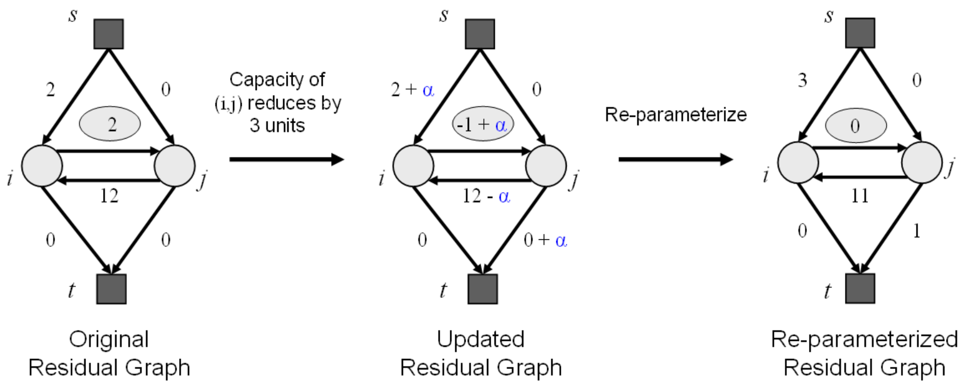


Figure 2.3: *Restoring consistency using graph reparameterization. The figure illustrates how edge capacities can be made consistent with the flow by reparameterizing the residual graph. It starts by showing a residual graph consisting of two nodes i and j obtained after a max-flow computation. For the second max-flow computation the capacity of edge (i, j) is reduced by 3 units resulting in the updated residual graph in which the residual capacity of edge (i, j) is equal to -1. To make the residual capacities positive we reparameterize the graph by adding $\alpha = 1$ to the capacity of edges (i, j), (s, i) and (j, t) and subtracting it from the capacity of edge (j, i). This gives us the reparameterized residual graph in which the edge flows are consistent with the edge capacities.*

2.3.2 Computational Complexity of Update Operations

In this section we analyze the computational complexity of various update operations that can be performed on the graph. Modifying an edge cost in the residual graph takes constant time. Arbitrary changes in the graph like addition or deletion of nodes and edges can be expressed in terms of modifying an edge cost. The time complexity of all such changes is O(1) except for deleting a node where the update time is O(k). Here k is the degree of the node to be deleted[1].

After the residual graph has been updated to reflect the changes in the energy function, the augmenting path procedure is used to find the maximum flow. This involves repeatedly finding paths with free capacity in the residual graph and saturating them. When no such paths can be found i.e., the source and sink are disconnected in the residual graph, we reach the maximum flow.

The maximum flow from the source to the sink is an upper bound on the

[1]The capacity of all edges incident on the node has to be made zero which takes O(1) time per edge.

number of augmenting paths found by the augmenting path procedure. Also, the total change in edge capacity bounds the increase in the flow ∇f defined as:

$$\nabla f \leq \sum_{i=1}^{m'} |c'_{e_i} - c_{e_i}|, \qquad \text{where} \quad e_i \in E$$

or, $\nabla f \leq m' c_{\max}$ where $c_{\max} = \max(|c'_{e_i} - c_{e_i}|)$. Thus we get a *loose* $O(m' c_{\max})$ bound on the number of augmentations, where m' is the number of edge capacity updates.

2.4. Improving Performance by Recycling Search Trees

We have seen how by dynamically updating the residual graph we can reduce the time taken to compute the st-mincut. We can further improve the running time by using a technique motivated by [10].

Typical augmenting path based methods start a new breadth-first search for (source to sink) paths as soon as all paths of a given length are exhausted. For instance, Dinic [21] proposed an augmenting path algorithm which builds search trees to find augmenting paths. This is a computationally expensive operation as it involves visiting almost all nodes of the graph and makes the algorithm slow if it has to be performed too often. To counter this, Boykov and Kolmogorov [10] proposed an algorithm in which they reused the search tree. In their experiments, this new algorithm outperformed the best-known augmenting-path and push-relabel algorithms on graphs commonly used in computer vision.

Motivated from their results we decided to reuse the search trees available from the previous max-flow computation to find the solution in the updated residual graph. This technique saved us the cost of creating a new search tree and made our algorithm substantially faster. The main differences between our algorithm and that of [10] are the presence of the tree restoration stage, and the dynamic selection of active nodes. We will next describe how the algorithm of [10] works and then explain how we modify it to recycle search trees for dynamic graph cuts.

2.4.1 Reusing Search Trees

The algorithm described in [10] maintains two non-overlapping search trees S and T with roots at the source s and the sink t respectively. In tree S all edges from each parent node to its children are non-saturated, while in tree T edges from children to their parents are non-saturated. The nodes that are not in S or T are called *free*. The nodes in the search trees S and T can be either *active* (can *grow* by acquiring new children along non-saturated edges) or *passive*. The algorithm starts by setting all nodes adjacent to the terminal nodes as *active*. The three basic stages of the algorithm are as follows:

Growth Stage The search trees S and T are grown until they touch each other (resulting in an augmenting path) or all nodes become *passive*. The active nodes explore adjacent non-saturated edges and acquire new children from the set of free nodes which now become active. As soon as all neighbours of a given active node are explored, the active node becomes passive. When an active node comes in contact with a node from the other tree, an augmenting path is found.

Augmentation Stage In this stage of the algorithm, flow is pushed through the augmenting path found in the growth stage. This results in some nodes of the trees S and T becoming *orphans* since the edges linking them to their parents become saturated. At this point, the source and sink search trees have decomposed into forests.

Adoption Stage During the adoption stage the search trees are restored by finding a new valid parent (of the same set) through a non-saturated edge for each orphan. If no qualifying parent can be found, the node is made free.

2.4.2 Tree Recycling for Dynamic Graph Cuts

We now explain our method for recycling search trees of the augmenting path algorithm. Our algorithm differs from that of [10] in the way we initialize the set of active nodes and in the presence of the Tree restoration stage.

2.4.2.1 Tree Restoration Stage

While dynamically updating the residual graph (as explained in section 2.3) certain edges of the search trees may become saturated and thus need to be deleted. This operation results in the decomposition of the trees into forests and makes certain nodes *orphans*. We keep track of all such edges and before recomputing the st-mincut on the modified residual graph restore the trees by finding a new valid parent for each of them. This process is similar to the adoption stage and is explained below.

The aim of the tree restoration stage is two fold. First to find parents for orphaned nodes, and secondly but more importantly, to make sure that the length of the path from the root node to all other nodes in the tree is as small as possible. This is necessary to reduce the time spent passing flow through an augmenting path. Note that longer augmenting paths would lead to a slower algorithm. This is because the time taken to update the residual capacities of the edges in the augmenting path during the augmentation stage is proportional to the length of the path.

The first objective of the restoration stage can be met by using the adoption stage alone. For the second objective we do the following: Suppose node i belonged to the source tree before the updates. For each graph node i which has been affected by the graph updates we check the residual capacities of its t-edges ((s,i) or (i,t)). We can encounter the following two cases:

1. $r_{si} \geq r_{it}$: The original parent of the node (in this case, the source (s)) is reassigned as the parent of the node.

2. $r_{si} < r_{it}$: The parent of the node is changed to the other terminal node 'sink' (t). This means that the node has now become a member of sink tree T. All the immediate child nodes of i are then made orphans as they had earlier belonged to the source tree.

The reassignment of parents of updated nodes according to the above mentioned rules resulted in a moderate but significant improvement in the results.

2.4.2.2 Dynamic Node Activation

The algorithm of [10] starts by marking the set of all nodes adjacent to the terminal nodes as *active*. This set is usually large and exploring all its constituent nodes is computationally expensive. However this is necessary as an augmenting path can pass through any such node.

In the case of the dynamic st-mincut problem, we can isolate a much smaller subset of nodes which need to be explored for possible augmenting paths. The key observation to be made in this regard is that all new possible augmenting paths are constrained to pass through nodes whose edges have undergone a capacity change. This results in a much smaller active set and makes the max-flow computation significantly faster. When no changes are made to the graph, all nodes in the graph remain *passive* and thus our augmenting path algorithm for computing the max-flow takes no time.

Chapter 3

Applications of Dynamic Graph Cuts

The dynamic graph cut algorithm proposed in the previous chapter can be used to *dynamically* perform MAP inference in an MRF or CRF. Such an inference procedure is extremely fast and has been used for a number of computer vision problems [12, 35, 49, 84].

In this chapter we describe some applications of dynamic graph cuts. To demonstrate the efficiency of the algorithm, we will provide quantitative results comparing its performance with the dual-search tree algorithm proposed in [10] which has been experimentally shown to be the fastest for several vision problems including image segmentation[1].

We will call the algorithm of [10] *static* since it starts afresh for each problem instance. The dynamic algorithm which reuses the search trees will be referred to as the *optimized* dynamic graph cut algorithm. It should be noted that while comparing running times the time taken to allocate memory for graph nodes was not considered. Further, to make the experimental results invariant to cache performance we kept the graphs in memory.

3.1. Dynamic Image Segmentation

Image segmentation has always remained an iconic problem in computer vision. The past few years have seen rapid progress made on it driven by the emergence of powerful optimization algorithms such as graph cuts. Early methods for performing image segmentation worked by coupling colour appearance information about the object and background with the edges present in an image to obtain good segmentations. However, this framework does not always guarantee good results. In particular, it fails in cases where the colour appearance models of the object and background are not discriminative.

A semi-automated solution to this problem was explored by Boykov and Jolly [9] in their work on interactive image segmentation. They showed how users could refine segmentation results by specifying additional constraints. This can be done by labelling particular regions of the image as 'object' or 'background' and then computing the MAP solution of the CRF again. The interactive image segmentation process is illustrated in figure 3.1.

3.1.1 CRFs for Image Segmentation

The image segmentation problem is commonly formulated using the CRF model described in chapter 1. In the context of image segmentation, the vertex set $\mathcal{V}$

[1]For the static algorithm we used the author's original implementation.

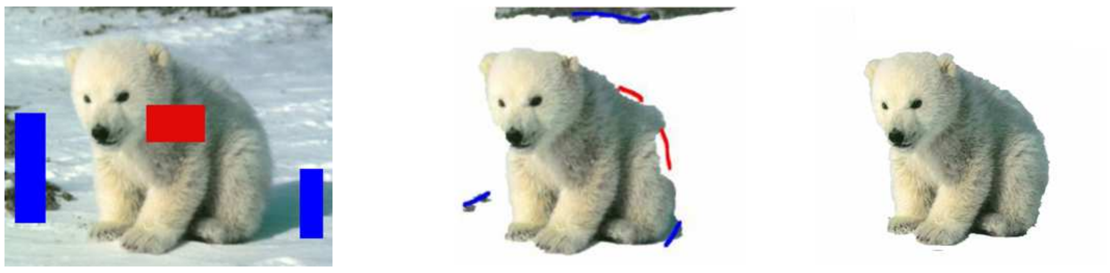

(a) (b) (c)

Figure 3.1: *Interactive image segmentation. The figure shows how good segmentation results can be obtained using a set of rough region cues supplied by the user. (a) An image with user specified segmentation cues (shown in blue and red). These cues were used to obtain the segmentation shown in image (b). This segmentation is not perfect and can be improved by specifying additional cues which are shown in (b). The final segmentation result is shown in image (c).*

corresponds to the set of all image pixels, $\mathcal{N}$ is a neighbourhood defined on this set[1], the set $\mathcal{L}$ consists of the labels representing the different image segments (which in our case are 'foreground' and 'background'), and the value x_v denotes the labelling of the pixel v of the image. Every configuration $\mathbf{x}$ of such a CRF defines a segmentation. The image segmentation problem can thus be solved by finding the least energy configuration of the CRF.

The energy function characterizing the CRFs used for image segmentation can be written as a sum of likelihood ($\phi(\mathbf{D}|x_i)$) and prior ($\psi(x_i, x_j)$) terms as:

$$\Psi_1(\mathbf{x}) = \sum_{i\in\mathcal{V}} \left(\phi(\mathbf{D}|x_i) + \sum_{j\in\mathcal{N}_i} \psi(x_i, x_j) \right) + \text{const.} \tag{3.1.1}$$

The term $\phi(\mathbf{D}|x_i)$ in the CRF energy is the data log likelihood which imposes individual penalties for assigning any label $k \in \mathcal{L}$ to pixel i. If we only take the appearance model into consideration, the likelihood is given by

$$\phi(\mathbf{D}|x_i) = -\log \Pr(i \in \mathcal{S}_k|\mathcal{H}_k) \qquad \text{if } x_i = k, \tag{3.1.2}$$

where $\mathcal{H}_k$ is the RGB (or for grey scale images, the intensity value) distribution for the segment $\mathcal{S}_k$ denoted by label $k \in \mathcal{L}$ [2]. The probability of a pixel belonging to a particular segment i.e. $\Pr(i \in \mathcal{S}_k|\mathcal{H}_k)$ is proportional to the likelihood $\Pr(I_i|\mathcal{H}_k)$, where I_i is the colour intensity of the pixel i. The likelihood $\Pr(I_i|\mathcal{H}_k)$ is generally computed from the colour histogram of the pixels belonging to the segment $\mathcal{S}_k$.

[1]In this work, we have used the standard 8-neighbourhood i.e., each pixel is connected to the 8 pixels surrounding it.

[2]In our problem, we have only 2 segments i.e., the foreground and the background.

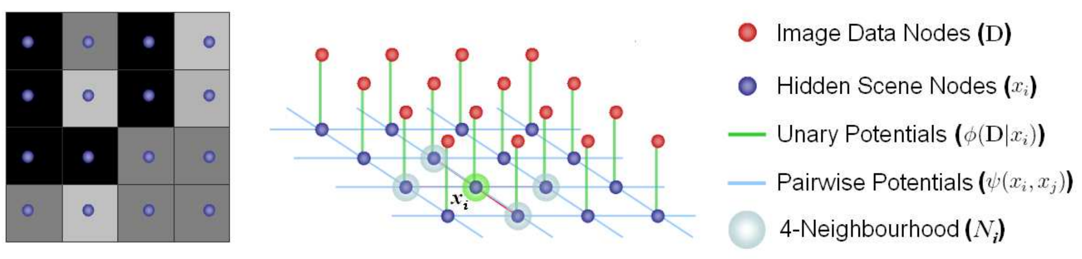

x_i
Image Data Nodes (D)
Hidden Scene Nodes (x_i)
Unary Potentials ($\phi(\mathrm{D}|x_i)$)
Pairwise Potentials ($\psi(x_i, x_j)$)
4-Neighbourhood (N_i)

Figure 3.2: *The pairwise MRF commonly used to model image labelling problems. The random field contains a hidden node corresponding to each pixel in the image. The MRF shown in the figure has a 4-neighbourhood, i.e. each node representing the random variables is connected to 4 neighbouring nodes.*

The prior $\psi(x_i, x_j)$ terms takes the form of a Generalized Potts model:

$$\psi(x_i, x_j) = \begin{cases} K_{ij} & \text{if} \quad x_i \neq x_j, \\ 0 & \text{if} \quad x_i = x_j. \end{cases} \tag{3.1.3}$$

The CRF used to model the image segmentation problem also contains a contrast term which favours pixels with similar colours having the same label [4,9]. This term is incorporated in the energy function by increasing the cost within the Potts model (for two neighbouring variables being different) in proportion to the similarity in intensities of their corresponding pixels. In our experiments, we use the function:

$$\gamma(i,j) = \lambda \exp\left(\frac{-g^2(i,j)}{2\sigma^2}\right) \frac{1}{\text{dist}(i,j)}, \tag{3.1.4}$$

where $g^2(i,j)$ measures the difference in the RGB values of pixels i and j and $\text{dist}(i,j)$ gives the spatial distance between i and j. This is a likelihood term (not prior) as it is based on the data, and hence has to be added separately from the smoothness prior. The energy function of the CRF now becomes

$$\Psi_2(\mathbf{x}) = \sum_{i \in \mathcal{V}} \left(\phi(\mathbf{D}|x_i) + \sum_{j \in \mathcal{N}_i} \left(\phi(\mathbf{D}|x_i, x_j) + \psi(x_i, x_j) \right) \right) \tag{3.1.5}$$

The contrast term of the energy function has the form

$$\phi(\mathbf{D}|x_i, x_j) = \begin{cases} \gamma(i,j) & \text{if} \quad x_i \neq x_j \\ 0 & \text{if} \quad x_i = x_j. \end{cases} \tag{3.1.6}$$

By adding this term to the energy, we have diverged from the strict definition of an MRF. The resulting energy function in fact now characterizes a Conditional Random Field [61]. The pairwise MRF commonly used to model image labelling problems is shown in figure 3.2.

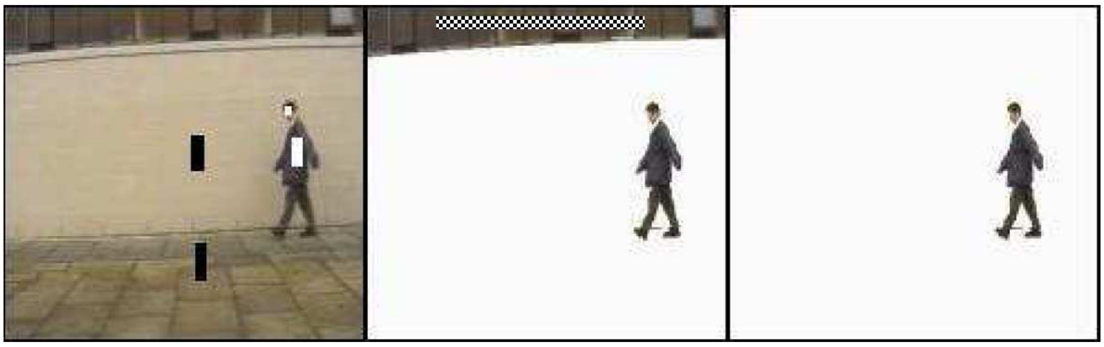


Figure 3.3: *Segmentation in videos using user seeds. The first image shows one frame of the input video with user segmentation seeds (the black and white boxes). The image pixels contained in these boxes are used to learn histograms modelling foreground and background likelihoods. The second image shows the segmentation result obtained using these likelihoods with the method of [9]. The result contains a certain portion of the background wrongly marked as the foreground due to similarity in colour. This error in the segmentation can be removed by the user by specifying a hard constraint. This involves marking a set of pixel positions in the wrongly labelled region as background (shown as the checkered region in the second image). This constraint is used for all the frames of the video sequence. The third image is the final segmentation result.*

3.1.2 Image Segmentation in Videos

The object-background segmentation problem aims to cut out user specified objects in an image [9]. We consider the case when this process has to be performed over all frames in a video sequence. The problem is formulated as follows.

The user specifies hard and soft constraints on the segmentation by providing segmentation cues or seeds on only the first frame of the video sequence. The **soft constraints** are used to build colour histograms for the *object* and *background*. These histograms are later used for calculating the likelihood term $\phi(\mathbf{D}|x_i)$ of the energy function (3.1.5) of the CRF. This is done for all the frames of the video sequence.

The **hard constraints** are used for specifying pixel positions which are constrained to take a specific label (*object* or *background*) in all the frames of the video sequence. Note that unlike soft constraints, the pixel positions specified under hard constraints do not contribute in the construction of the colour histograms for the *object* and *background*. This is different from the user-input strategy adopted in [9]. In our method the hard constraints are imposed on the segmentation by incorporating them in the likelihood term $\phi(\mathbf{D}|x_i)$. This is done by imposing

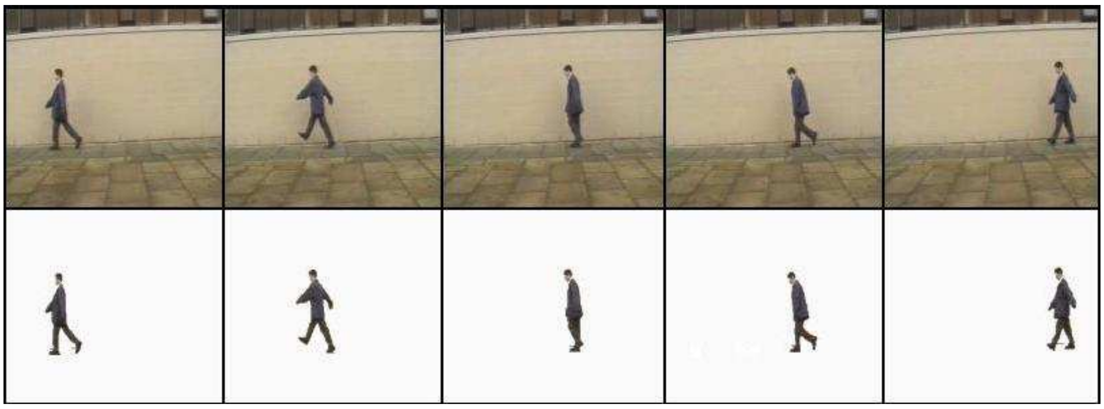


Figure 3.4: *Segmentation results of the human lame walk video sequence.*

a very high cost for a label assignment that violates the hard constraints in a manner similar to [9]. This method for specifying hard constraints has been chosen because of its simplicity. Readers should refer to [113] for a sophisticated method for specifying hard constraints for the video segmentation problem. Figure 3.3 demonstrates the use of constraints in the image segmentation process. The segmentation results are shown in figure 3.4.

3.1.3 Experimental Results

In this section we demonstrate the performance of our dynamic graph cut algorithm on the image segmentation problem. We compare the time taken by our algorithm with that needed by the algorithm proposed in [10].

In the interactive image segmentation experiments, we observed that dynamic graph cuts resulted in a massive improvement in the running time. For the image shown in figure 3.1, the time taken by the static st-mincut algorithm to compute the refined solution (from scratch) was 120 milliseconds. The dynamic algorithm computed the same solution in 45 milliseconds, while the dynamic (optimized) algorithm only required 25 milliseconds.

We now discuss the results of image segmentation in videos. The video sequences used in our tests had between one hundred to a thousand image frames. For all the video sequences dynamically updating the residual graph produced a decrease in the number of augmenting paths. Further, the dynamic algorithms (normal and optimized) were substantially faster than the *static* algorithm. The average running times per image frame for the static, dynamic and optimized-dynamic algorithms for the human lame walk sequence[3] of size 368x256 were 91.4,

[3]Courtesy Derek Magee, University of Leeds.

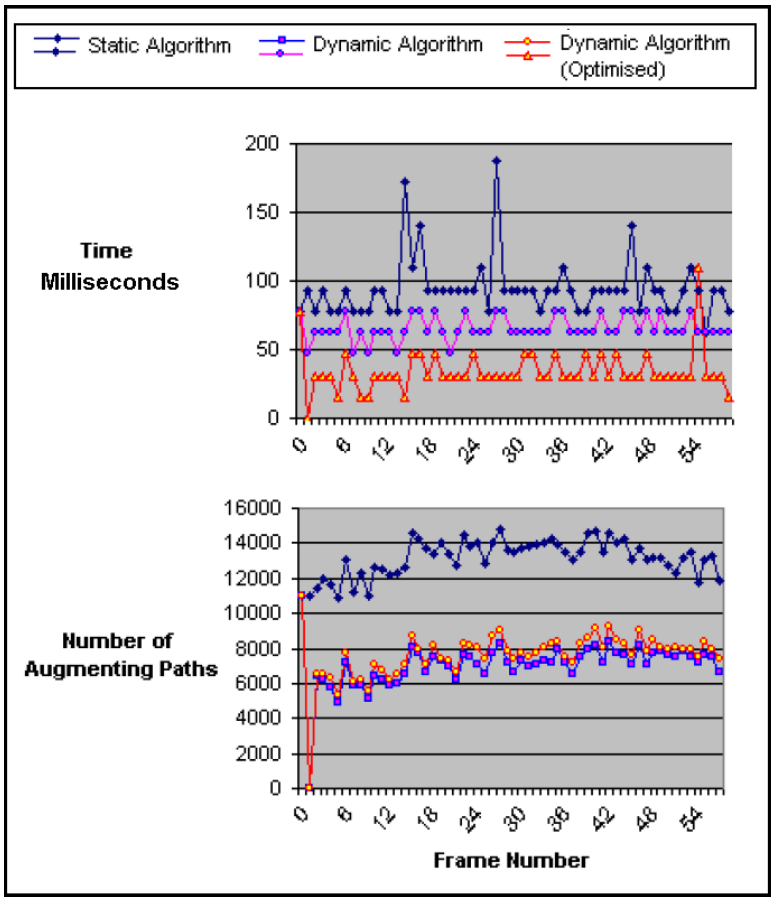


Figure 3.5: *Running time and number of augmenting paths found by static and dynamic st-mincut algorithms. Observe that as the first and second frames of the video sequence are the same, the residual graph does not need to be updated, which results in no augmenting paths found by the dynamic algorithms when segmenting frame 2. Further, the optimized dynamic algorithm takes no time for computing the segmentation for the second image frame as the* CRF*s corresponding to the first and second image frames are the same and thus no modifications were needed in the residual graph and search trees. However, the normal dynamic algorithm takes a small amount of time since it recreates the search trees for every problem instance from scratch.*

66.0, and 33.6 milliseconds and for the grazing cow sequence of size 720x578 were 188.8, 151.3, and 78.0 milliseconds respectively. The time taken by the dynamic algorithm includes the time taken to recycle the search trees. The experiments were performed on a Pentium 4 2.8 GHz machine.

The graphs in figure 3.5 show the performance of the algorithms on the first sixty frames of the human lame walk sequence. Observe that the number of augmenting paths found is lowest for the dynamic algorithm, followed by the dynamic (optimized) and then the static algorithm. The use of more augmenting paths by the dynamic (optimized) algorithm is due to the utilization of recycled search trees which produce long augmenting paths.

3.1.4 Reusing Flow Vs Reusing Search Trees

In this section, the relative contributions of reusing flow and search trees in improving the running time of the dynamic algorithm are discussed.

The procedure for constructing a search tree has linear time complexity and thus should be quite fast. Further as seen in figure 3.5 using a fresh search tree after every graph update results in fewer augmenting paths. From these results it might appear that recycling search trees would not yield a significant improvement in running time. However this is not the case in practice as seen in figure 3.6. This is because although the complexity of search tree construction is linear in the number of edges in the graph, the time taken for tree construction is still substantial. This is primarily due to the nature of graphs used in computer vision problems. The number of nodes/edges in these graphs may be of the order of millions. For instance, when segmenting an image of size 640×480, max-flow on a graph consisting of roughly 3×10^5 nodes and more than 2 million edges needs to be computed. The total time taken for this operation is 90 milliseconds (msec) out of which almost 15 msec is spent on constructing the search tree.

The time taken by the dynamic algorithm to compute the st-mincut decreases with the decrease in the number of changes made to the graph. However, as the time taken to construct the search tree is independent of the number of changes, it remains constant. This results in a situation where if only a few changes to the graph are made (as in the case of min-marginal computation [49]), the dominant part of computation time is spent on constructing the search tree itself. By reusing search trees we can get rid of this constant cost of creating a search tree and replace it with a change dependent tree restoration cost.

The exact amount of speed-up contributed by reusing flow and search trees techniques varies with the problem instance. For a typical interactive image segmentation example, the first st-mincut computation takes 120 msec out of which 30 msec is spent on constructing the search tree. We need to recompute the st-mincut after further user interaction (which results in changes in the graphs). For the later st-mincut computation, if we construct a new search tree then the time taken by the algorithm is 45 msec (a speed up of roughly 3 times) out of which 30 msec is used for tree creation and 15 msec is used for flow computation. However, if we use reuse the search trees, then the algorithm takes only 25 msec (a speed up of 5 times) out of which 7 msec is used for recycling the tree and 18

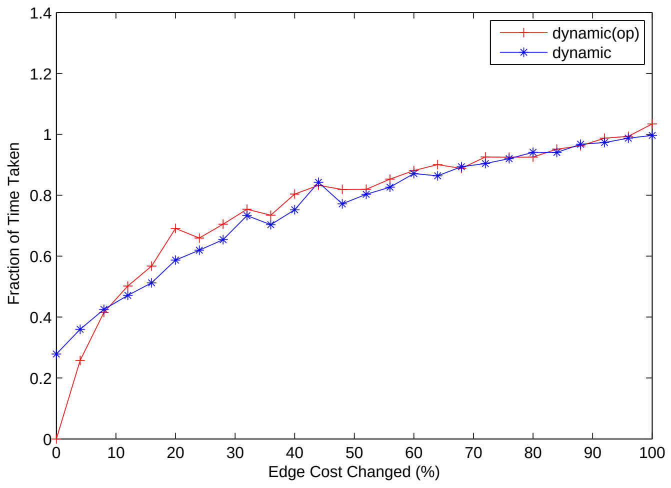


Figure 3.6: *Behavior of the dynamic algorithm. The figure illustrates how the time taken by the dynamic algorithm (with/without tree recycling) changes with the number of modifications made to the graph. The graph shows the fraction of time taken to compute the st-mincut in the updated residual graph (with/without tree recycling) compared to that taken for computing the st-mincut in the original graph using the algorithm of [10]. For this experiment, we used a graph consisting of* 1×10^5 *nodes which were connected in a 8-neighbourhood. The dynamic algorithm with tree recycling is referred as dynamic(op).*

msec is used for flow computation.

Our results indicate that when a small number of changes are made to the graph the recycled search tree works quite well in obtaining short augmenting paths. The time taken for recycling search trees is also small compared to the time taken to create a new search tree in a large graph. With increased change in the graph the advantage in using the recycled search tree fades due to the additional number of flow augmentations needed as a result of longer augmentation paths obtained from the search tree.

3.2. Simultaneous Segmentation and Pose Estimation of Humans

In this section of the dissertation I will present a novel algorithm for performing integrated segmentation and 3D pose estimation of a human body from multiple views. Unlike other state of the art methods which focus on either segmentation or pose estimation individually, our approach tackles these two tasks together. Our method works by optimizing a cost function based on a Conditional Random Field (CRF). This has the advantage that all information in the image (edges, background and foreground appearances), as well as the prior information on the shape and pose of the subject can be combined and used in a Bayesian framework. Optimizing such a cost function would have been computationally infeasible earlier. However, our recent research in dynamic graph cuts allows this to be done much more efficiently than before. We demonstrate the efficacy of our approach on challenging motion sequences. Although we target the human pose inference problem in this work, our method is completely generic and can be used to segment and infer the pose of any rigid, deformable or articulated object.

Human pose inference is an important problem in computer vision. It stands at the crossroads of various important applications ranging from Human Computer Interaction (HCI) to surveillance. The importance and complexity of this problem can be gauged by observing the number of papers which have tried to deal with it [1, 20, 23, 31, 45, 62, 68, 79, 92, 96, 98, 107]. Most algorithms which perform pose estimation require the segmentation of humans as an essential introductory step [1, 45, 92]. This precondition limits the use of these techniques to scenarios where good segmentations are made available by enforcing strict studio conditions like blue-screening. Otherwise a preprocessing step must be performed in an attempt to segment the human, such as [100]. These approaches however cannot obtain good segmentations in challenging scenarios which have: complex foreground and background, multiple objects in the scene, and moving camera/background. Some pose inference methods exist which do not need segmentations. These rely on features such as chamfer distance [31], appearance [96], or edge and intensity [98]. However, none of these methods is able to efficiently utilize all the information present in an image, and fail if the feature detector they are using fails. This is partly because the feature detector is not coupled to the knowledge of the pose and nature of the object to be segmented.

The question is then, how to simultaneously obtain the segmentation and human pose using all available information contained in the images?

Some elements of the answer to this question have been described by Kumar *et al.* [59]. Addressing the object segmentation problem, they report that ***"samples from the Gibbs distribution defined by a Markov Random Field very rarely give rise to realistic shapes"***. As an illustration of this statement, figure 3.7(*b*) shows the segmentation result corresponding to the maximum a posteriori (MAP) solution of the Conditional Random Field (CRF) incorporating information about the image edges and appearances of the object and background. It can be clearly seen that this result is nowhere close to the ground truth.

Shape priors and segmentation In recent years, a number of papers have tried to couple MRFs or CRFs used for modelling the image segmentation problem, with information about the nature and shape of the object to be segmented [28, 38, 59, 119]. One of the earliest methods for combining MRFs with a shape prior was proposed by Huang *et al.* [38]. They incrementally found the MAP solution of an extended MRF[1] integrated with a probabilistic deformable model. They were able to obtain a refined estimate of the object contour by using belief propagation in the area surrounding the contour of this deformable model. This process was iterated till convergence.

The problem however was still far from being completely solved since objects in the real world change their shapes constantly and hence it is difficult to ascertain what would be a good choice for a prior on the shape. This complex and important problem was addressed by the work of Kumar *et al.* [59]. They modelled the segmentation problem by combining CRFs with layered pictorial structures (LPS) which provided them with a realistic shape prior described by a set of latent shape parameters. Their cost function was a weighted sum of the energy terms for different shape parameters (samples). The weights of this energy function were obtained by using the Expectation-Maximization (EM) algorithm. During this optimization procedure, a graph cut had to be computed in order to obtain the segmentation score each time any parameter of the CRF was changed. This made their algorithm extremely computationally expensive.

Although their approach produced good results, it had some shortcomings. It was focused on obtaining good segmentations and did not provide the pose of the object explicitly. Moreover, a lot of effort had to be spent to learn the exemplars for different parts of the LPS model. Recently, Zhao and Davis [119] exploited the idea of object-specific segmentation to improve object recognition and detection. Their method worked by coupling the twin problems of object detection and segmentation in a single framework. They matched exemplars to objects in the image using chamfer matching and thus like [59] also suffered from the problem of maintaining a huge exemplar set for complex objects. We will describe how we overcome the problem of maintaining a huge exemplar set by using a simple articulated stickman model, which is not only efficiently renderable, but also provides a robust human-like segmentation and accurate pose estimate. To make our algorithm computationally efficient we use the dynamic graph cut algorithm.

[1]It is named an *extended* MRF due to the presence of an extra layer in the MRF to cope with the shape prior.

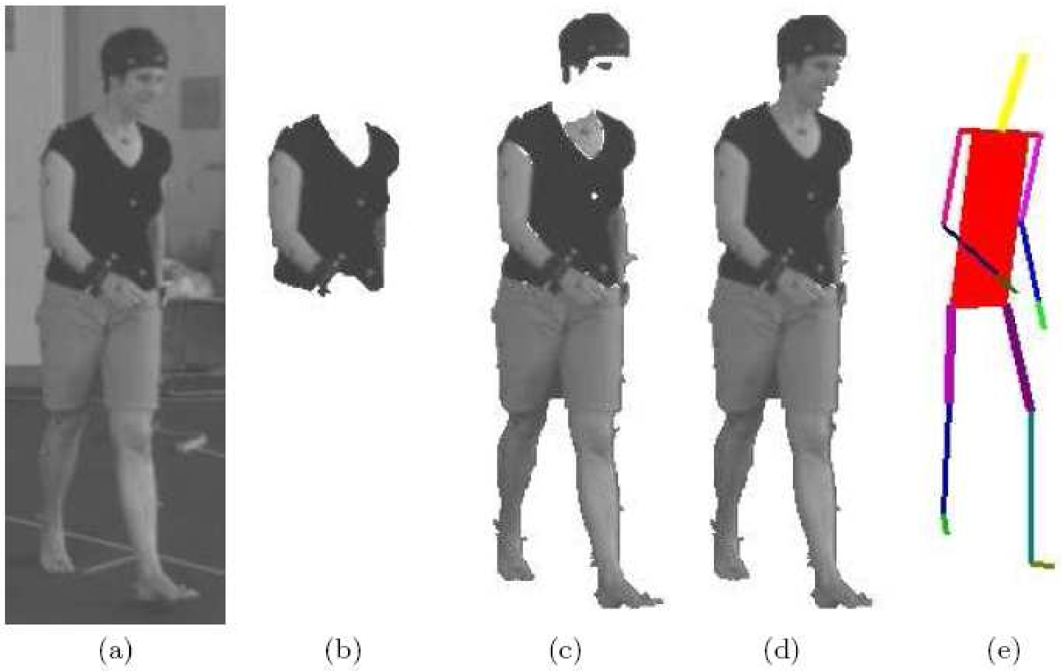


Figure 3.7: *Improving segmentation results by incorporating more information in the CRF. (a) Original image. (b) The segmentation obtained corresponding to the* MAP *solution of a* CRF *consisting of colour likelihood and contrast terms as described in [9]. We give the exact formulation of this CRF in section 3.1.1. (c) The result obtained when the likelihood term of the CRF also takes into account the Gaussian Mixture Models (GMM) of individual pixel intensities as described in section 3.2.1.1. (d) Segmentation obtained after incorporating a 'pose-specific' shape prior in the CRF as explained in Section 3.2.1.2. The prior is represented as the distance transform of a stickman which guarantees a human-like segmentation. (e) The stickman model after optimization of its 3D pose (see Section 3.2.2). Observe how incorporating the individual pixel colour models in the CRF (c) gives a considerably better result than the one obtained using the standard appearance and contrast based representation (b). However the segmentation still misses the face of the subject. The incorporation of a stickman shape prior ensures a human-like segmentation (d) and provides simultaneously (after optimization) the 3D pose of the subject (e).*

Shape Priors in Level Sets Prior knowledge about the shape to be segmented has also been used in level set methods for obtaining an object segmentation. Like [59] these methods learn the prior using a number of training shapes. Leventon *et al.* [65] performed principal component analysis on these shapes to get an embedding function which was integrated in the evolution equation of the level set. More recently, Cremers *et al.* [18] have used kernel density estimation and intrinsic alignment to embed more complex shape distributions. Compared to [59] and [119] these methods have a more compact representation of the shape prior. However, they suffer from the disadvantage that equations for level set evolution may lead to a local minima.

Human Pose Estimation In the last few years, several techniques have been proposed for tackling the pose inference problem. In particular, the works of Agarwal and Triggs [1] using relevance vector machines and that of Shakhnarovich *et al.* [92] based on parameter sensitive hashing induced a lot of interest and have been shown to give good results. Some methods for human pose estimation in monocular images use a tree-structured model to capture the kinematic relations between parts such as the torso and limbs [23, 68, 79]. They then use efficient inference algorithms to perform exact inference in such models. In their recent work, Lan and Huttenlocher [62] show how the tree-structured restriction can be overcome while not greatly increasing the computational cost of estimation.

Overview of the Method Our method does not require a feature extraction step but uses all the data in the image. We formulate the problem in a Bayesian framework building on the object-specific CRF [59] and provide an efficient method for its solution called PoseCut. We include a human *pose-specific* shape prior in the CRF used for image segmentation, to obtain high quality segmentation results. We refer to this integrated model as a *pose-specific* CRF. Unlike Kumar *et al.* [59], our approach does not require the laborious process of learning exemplars. Instead we use a simple articulated stickman model, which together with an CRF is used as our shape prior. The experimental results show that this model suffices to ensure human-like segmentations.

Given an image, the solution of the pose-specific CRF is used to measure the quality of a 3D body pose. This cost function is then optimized over all pose parameters using dynamic graph cuts to provide both an object-like segmentation and the pose. The astute reader will notice that although we focus on the human pose inference problem, our method is in-fact general and can be used to segment and/or infer the pose of any object. We believe that our methodology is completely novel and we are not aware of any published methods which perform simultaneous segmentation and pose estimation. To summarize, the novelties of our approach include:

- An efficient method for combined object segmentation and pose estimation (PoseCut).
- Integration of a simple 'stickman prior' based on the skeleton of the object in a CRF to obtain a *pose-specific* CRF which helps us in obtaining high quality object pose estimate and segmentation results.

3.2.1 Pose Specific CRF for Image Segmentation

The CRF framework for image segmentation described in section 3.1.1 uses likelihood terms which are only based on the pixel colour. This term is quite weak and thus does not always guarantee good results. In particular, it fails in cases where the colour appearance models of the object and background are not discriminative as seen in figure 3.7(b). The problem becomes even more pronounced in the case of humans where we have to deal with the various idiosyncracies of human clothing.

From the work of Boykov and Jolly [9] on interactive image segmentation we made the following interesting observations:

- ***Simple user supplied shape cues used as rough priors for the object segmentation problem produced excellent results.***
- ***The exact shape of the object can be induced from the edge information embedded in the image.***

Taking these into consideration, we hypothesized that the accurate exemplars used in [59] to generate shape priors were in-fact an overkill and could be replaced by much simpler models. Motivated by these observations we decided against using a sophisticated shape prior. We have used two simple models in our work which are described below.

Stickman model We used a simple articulated stickman model for the full body human pose estimation problem. The model is shown in figure 3.7(e). It is used to generate a rough pose-specific shape prior on the segmentation. As can been seen from the segmentation results in figure 3.7(d), the stickman model helped us to obtain excellent segmentation results. The model has 26 degrees of freedom consisting of parameters defining absolute position and orientation of the

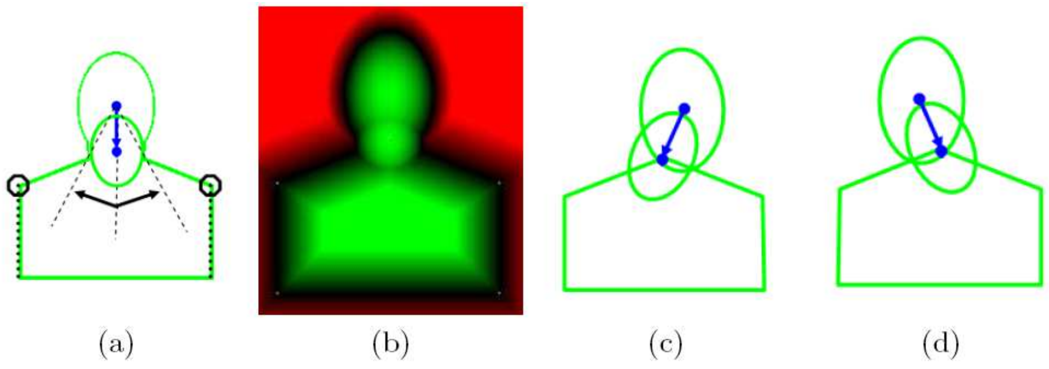


Figure 3.8: *The human upper body model. (a) The human upper body model parameterized by 6 parameters encoding the x and y location of the two shoulders, the length of the neck, and the angle of the neck with respect to the vertical. (b) The shape prior generated using the model. Pixels more likely to belong to the foreground/background are green/red. (c) and (d) The model rendered in two poses.*

torso, and the various joint angle values. There were no constraints or joint-limits incorporated in our model.

The Upper body Model The second model was primarily designed for the problem of segmenting the human speaker in video conference scenarios. The model can be seen in figure 3.8. It is parameterized by 6 parameters which encode the x and y location of the two shoulders and the length and angle of the neck.

We now describe how the image segmentation problem can be modeled using a *pose-specific* CRF. Our pose specific CRF is obtained by augmenting the conventionally used CRF model for image segmentation (see section 3.1.1) with potentials based on the shape of the object to be segmented, and appearances of individual pixels.

3.2.1.1 Modeling Pixel Intensities by Gaussian Mixture Models

The CRF defined in section 3.1.1 performs poorly when segmenting images in which the appearance models of the foreground and background are not highly discriminative. When working on video sequences, we can use a background model developed using the Grimson-Stauffer [100] algorithm to improve our results. This algorithm works by representing the colour distribution of each pixel position in the video as a Gaussian Mixture Model (GMM). The likelihoods of a pixel for being background or foreground obtained by this technique are integrated in our CRF. Figure 3.7(c) shows the segmentation result obtained after incorporating this information in our CRF formulation.

3.2.1.2 Incorporating the Pose-specific Shape Prior

Though the results obtained from the above formulation look decent, they are not perfect. Note that there is no prior on the segmentation to look human like. Intuitively, incorporating such a constraint in the CRF would improve the segmentation. In our case, this prior should be *pose-specific* as it depends on what pose the object (the human) is in. Kumar *et. al.* [59], in their work on interleaved object recognition and segmentation, used the result of the recognition to develop a shape prior over the segmentation. This prior was defined by a set of latent variables which favoured segmentations of a specific pose of the object. They called this model the Object Category Specific CRF, which had the following energy function:

$$\Psi_3(\mathbf{x}, \mathbf{\Theta}) = \sum_i \Big(\phi(\mathbf{D}|x_i) + \phi(x_i|\mathbf{\Theta}) + \sum_j \big(\phi(\mathbf{D}|x_i, x_j) + \psi(x_i, x_j)\big)\Big) \tag{3.2.1}$$

with posterior $p(\mathbf{x}, \mathbf{\Theta}|\mathbf{D}) = \frac{1}{Z_3}\exp(-\Psi_3(\mathbf{x}, \mathbf{\Theta}))$. Here $\mathbf{\Theta} \in \mathbb{R}_p$ is used to denote the vector of the object pose parameters. The shape-prior term of the energy function for a particular pose of the human is shown in figure 3.9(e). This is a distance transform generated from the stick-man model silhouette using the fast implementation of Felzenszwalb and Huttenlocher [22].

The function $\phi(x_i|\mathbf{\Theta})$ was chosen such that given an estimate of the location and shape of the object, pixels falling near to that shape were more likely to be labelled as 'foreground' and vice versa. It has the form: $\phi(x_i|\mathbf{\Theta}) = -\log p(x_i|\mathbf{\Theta})$. We follow the formulation of [59] and define $p(x_i|\mathbf{\Theta})$ as

$$p(x_i = \text{figure}|\mathbf{\Theta}) = 1 - p(x_i = \text{ground}|\mathbf{\Theta}) = \frac{1}{1 + \exp(\mu * (d(i, \mathbf{\Theta}) - d_r))}, \tag{3.2.2}$$

where $d(i, \mathbf{\Theta})$ is the distance of a pixel i from the shape defined by $\mathbf{\Theta}$ (being negative if inside the shape). The parameter d_r decides how 'fat' the shape should be, while parameter μ determines the ratio of the magnitude of the penalty that points outside the shape have to face, compared to the points inside the shape.

3.2.1.3 Inference in the CRF using graph cuts

Recall from chapter 1 that energy functions like the one defined in (3.2.1) can be solved using graph cuts if they are *sub-modular* [54]. A function $f : \{0, 1\}^n \to \mathbb{R}$

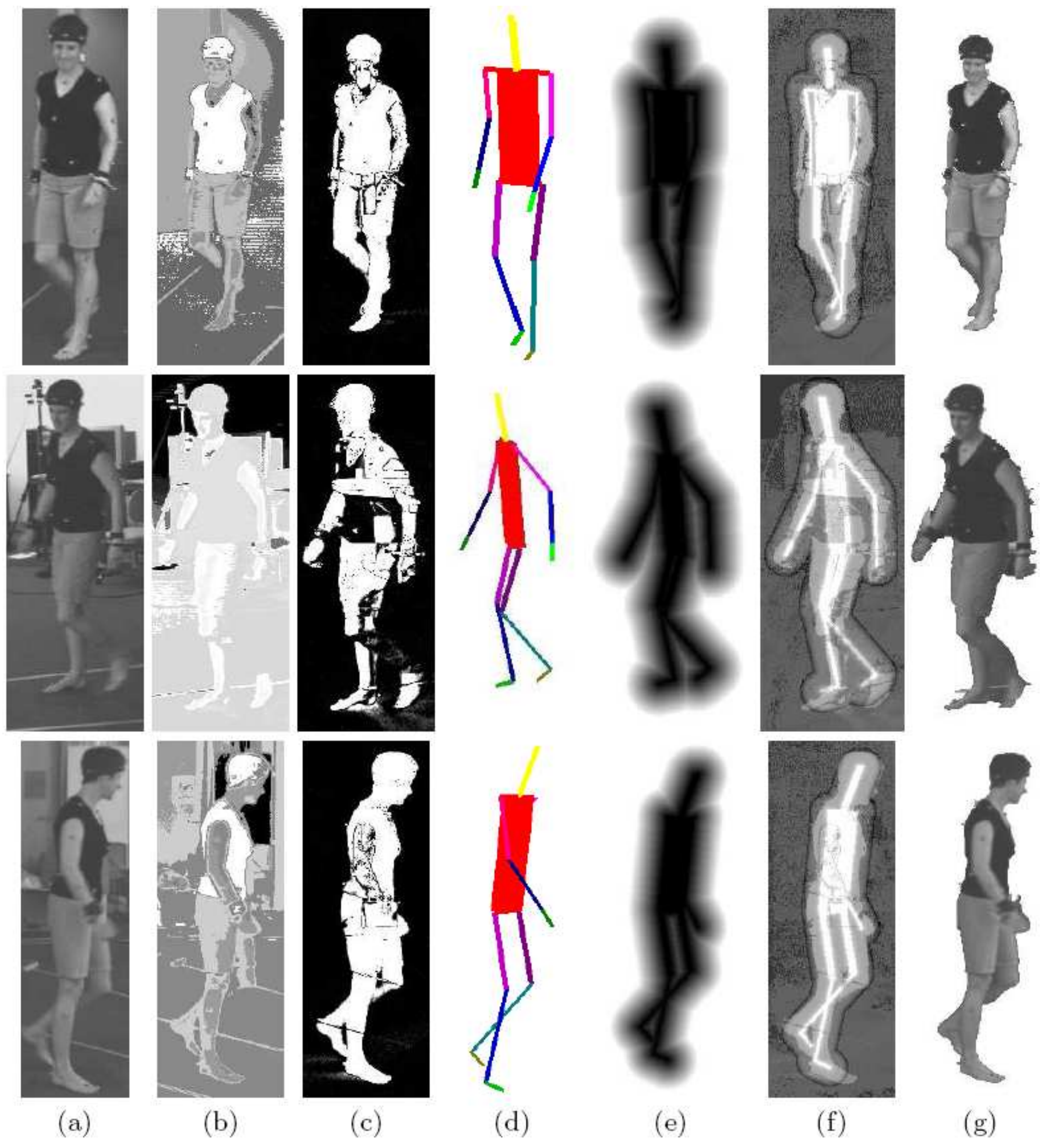


Figure 3.9: *Different terms of our pose specific CRF. (a) Original image. (b) The ratios of the likelihoods of pixels being labelled foreground/background ($\phi(\mathbf{D}|\mathbf{x}_i =$ 'fg'$) - \phi(\mathbf{D}|\mathbf{x}_i =$ 'bg'$)$). These values are derived from the colour intensity histograms. (c) The segmentation results obtained by using the GMM models of pixel intensities. (d) The stickman in the optimal pose (see Sections 3.2.1.2 and 3.2.2). (e) The shape prior (distance transform) corresponding to the optimal pose of the stickman. (f) The ratio of the likelihoods of being labelled foreground/background using all the energy terms (colour histograms defining appearance models, GMMs for individual pixel intensities, and the pose-specific shape prior (see sections 3.1.1,3.2.1.1 and 3.2.1.2)) $\Psi_3(x_i =$ 'fg'$, \mathbf{\Theta}) - \Psi_3(x_i =$ 'bg'$, \mathbf{\Theta})$. (g) The segmentation result obtained from our algorithm which is the MAP solution of the energy Ψ_3 of the pose-specific CRF.*

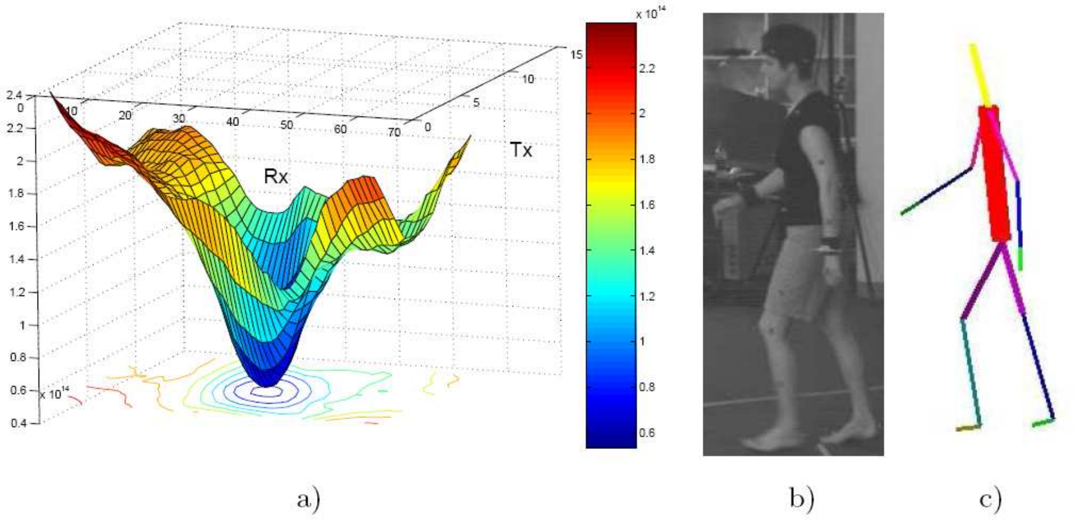


Figure 3.10: *Inferring the optimal pose. a) The values of* $\min_{\mathbf{x}} \Psi_3(\mathbf{x}, \mathbf{\Theta})$ *obtained by varying the global translation and rotation of the shape prior in the x-axis. b) Original image. c) The pose obtained corresponding to the global minimum of the energy.*

is submodular if and only if all its projections on 2 variables ($f^p : \{0,1\}^2 \rightarrow \mathbb{R}$) satisfy:

$$f^p(0,0) + f^p(1,1) \leq f^p(0,1) + f^p(1,0). \qquad (3.2.3)$$

For the pairwise potentials, this condition can be seen as implying that the energy for two labels taking similar values should be less than the energy for them taking different values. In our case, this is indeed the case and thus we can find the optimal configuration $\mathbf{x}^* = \min_{\mathbf{x}} \Psi_3(\mathbf{x}, \mathbf{\Theta})$ using a single graph cut. The labels of the latent variable in this configuration give the segmentation solution.

3.2.2 Formulating the Pose Inference Problem

Since the segmentation of an object depends on its estimated pose, we would like to make sure that our shape prior reflects the actual pose of the object. This takes us to our original problem of finding the pose of the human in an image. In order to solve this, we start with an initial guess of the object pose and optimize it to find the correct pose. When dealing with videos, a good starting point for this process would be the pose of the object in the previous frame. However, more sophisticated methods could be used based on object detection [101] at the expense of increasing the computation time.

One of the key contributions of this work is to show how given an image of the object, the pose inference problem can be formulated in terms of an optimization problem over the CRF energy given in (3.2.1). Specifically, we solve the problem:

$$\mathbf{\Theta_{opt}} = \arg\min_{\mathbf{\Theta},\mathbf{x}} \Psi_3(\mathbf{x}, \mathbf{\Theta})). \tag{3.2.4}$$

The minimization problem defined above contains both discrete ($\mathbf{x} \in \{0,1\}^n$) and continuous ($\mathbf{\Theta} \in \mathbb{R}_P$) valued variables and thus is a mixed integer programming problem. The large number of variables involved in the energy function $\Psi_3(\mathbf{x}, \mathbf{\Theta}))$ make it especially challenging to minimize. To solve the minimization problem (3.2.4), we decompose it as: $\mathbf{\Theta_{opt}} = \arg\min_{\mathbf{\Theta}} \mathcal{F}(\mathbf{\Theta})$, where

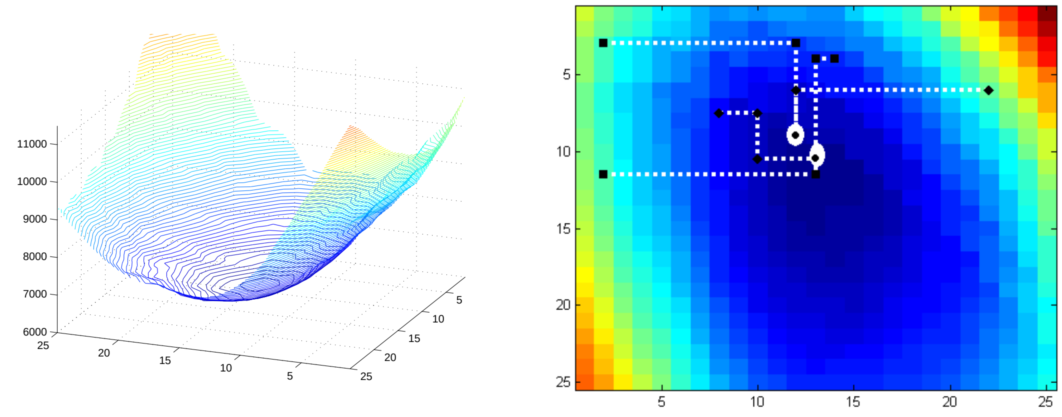
$$\mathcal{F}(\mathbf{\Theta}) = \min_{\mathbf{x}} \Psi_3(\mathbf{x}, \mathbf{\Theta})). \tag{3.2.5}$$

For any value of $\mathbf{\Theta}$, the function $\Psi_3(\mathbf{x}, \mathbf{\Theta}))$ is submodular in $\mathbf{x}$ and thus can be minimized in polynomial time by solving a single st-mincut problem to give the value of $\mathcal{F}(\mathbf{\Theta})$.

We will now explain how we minimize $\mathcal{F}(\mathbf{\Theta})$ to get the optimal value of the pose parameters. Figure 3.10 shows how the function $\mathcal{F}(\mathbf{\Theta})$ depends on parameters encoding the rotation and translation of our stickman model in the x-axes. It can be seen that the function surface is unimodal in a large neighbourhood of the optimal solution. Hence, given a good initialization of the pose Θ, it can be reliably optimized using any standard optimization algorithm like gradient descent. In our experiments, we used the Powell minimization [74] algorithm for optimization.

Figure 3.11(a) shows how the function $\mathcal{F}(\mathbf{\Theta})$ changes with changes to the neck angle and length parameters of the upper body model shown in figure 3.8. Like in the case of the 3D stickman model, the energy surface is well behaved near the optimal pose parameters. Our experiments showed that the Powell minimization algorithm is able to converge to almost the same point for different initializations (see figure 3.11(b)).

Failure Modes It can be seen that the function $\mathcal{F}(\mathbf{\Theta})$ is not unimodal over the whole domain and contains local minima. This multi-modality of $\mathcal{F}(\mathbf{\Theta})$ can cause a gradient descent algorithm to get trapped and converge to a local minimum. In our experiments we observed that these spurious minima lie quite far from the globally optimal solution. We also observed that the pose of the human subject generally does not change substantially from one frame to the next. This lead us to use the pose estimate from the previous frame as an initialization for the current frame. This good initialization for the pose estimate made sure that spurious minima do not effect our method.



(a) (b)

Figure 3.11: *Optimizing the pose parameters. (a) The values of* $\min_{\mathbf{x}} \Psi_3(\mathbf{x}, \mathbf{\Theta})$ *obtained by varying the rotation and length parameters of the neck. (b) The image shows five runs of the Powell minimization algorithm [74] which are started from different initial solutions.*

The failure rate of our method can be further improved by using object detection systems which provide a better initialization of the pose of the object. Scenarios where the method still converges to a local minima can be detected and dealt with using the strategy discussed in section 3.2.5 which was used in our recent work on object detection and segmentation [81].

Resolving Ambiguity using multiple views The human pose inference problem in the context of monocular images suffers from ambiguity. This is because of the one-many nature of the mapping that relates a human shape as seen in an image and the corresponding human pose. In other words, many possible poses can explain the same human shape. This ambiguity can be resolved by using multiple views of the object ('human'). Our framework has the advantage that information from multiple views can be integrated into a single optimization framework. Specifically, when dealing with multiple views we solve the problem:

$$\mathbf{\Theta_{opt}} = \arg\min_{\mathbf{\Theta}} \Big(\sum_{\text{Views}} \min_{\mathbf{x}} (\Psi_3(\mathbf{x}, \mathbf{\Theta})) \Big). \tag{3.2.6}$$

The framework is illustrated in figure 3.12.

Dynamic energy minimization using graph cuts As explained earlier global minima of energies like the one defined in (3.2.1) can be found by graph cuts [54]. The time taken for computing a graph cut for a reasonably sized CRF is of the order of seconds. This would make our optimization algorithm extremely

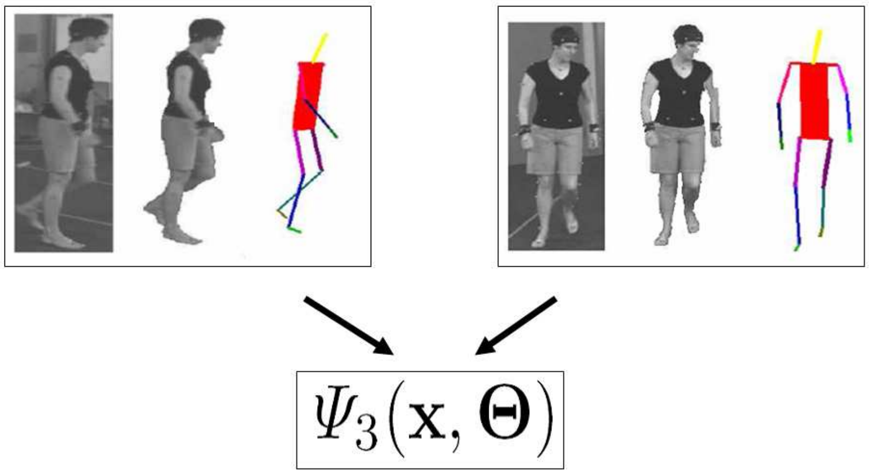


Figure 3.12: *Resolving ambiguity in pose using multiple views. The figure shows how information from different views of the human can be integrated in a single energy function, which can be used to find the true pose of the human subject.*

slow since we need to compute the global optimum of $\Psi_3(\mathbf{x}, \boldsymbol{\Theta})$ with respect to $\mathbf{x}$ multiple number times for different values of $\boldsymbol{\Theta}$. The graph cut computation can be made significantly faster by using the dynamic graph cut algorithm proposed in chapter 2. This algorithm works by using the solution of the previous graph cut computation for solving the new instance of the problem. We obtained a speed-up in the range of 15-20 times by using the dynamic graph cut algorithm.

3.2.3 Experiments

We now discuss the results obtained by our method. We provide the segmentation and pose estimation results individually.

3.2.3.1 Segmentation Results

As expected, the experimental results show that the segmentation results improve considerably as we increase the amount of information in our CRF framework. Figure 3.13 shows how integrating more information in the CRF improves the segmentation results. Quantitative results for the segmentation problem are shown in table 3.1.

In order to demonstrate the performance of our method, we compare our segmentation results with those obtained using the method proposed in [100]. It can be seen from the results in figure 3.15 that the segmentations obtained using the method of [100] are not accurate: They contain "speckles" and often

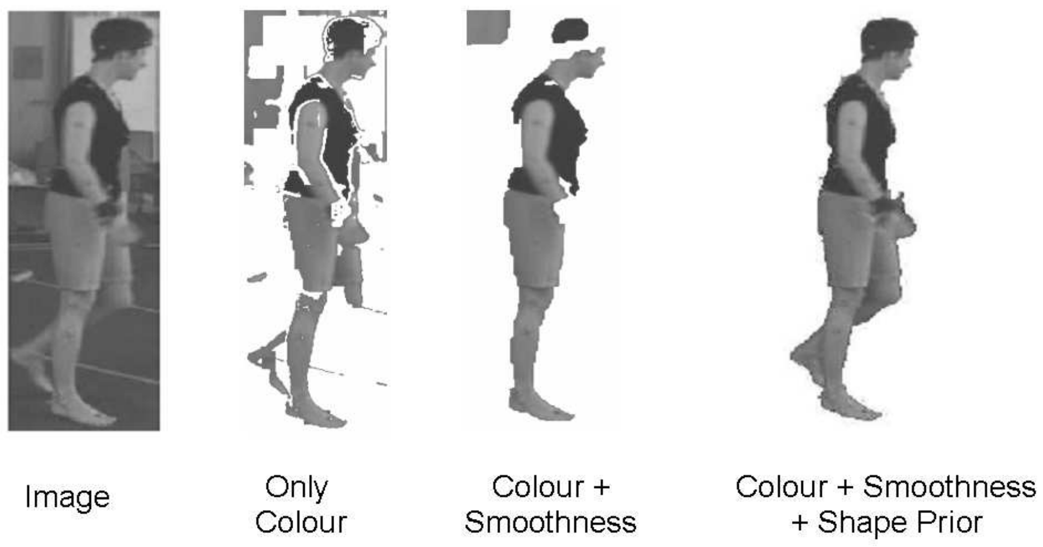


Figure 3.13: *Results showing the effect of incorporating a shape prior on the segmentation results. The first image is the original image to be segmented. The second, third and fourth images show the segmentation results obtained using colour, colour + smoothness prior and colour + smoothness + shape prior respectively.*

Information Used	**Correct object pixels**	**All correct pixels**
Colour	45.73%	95.2%
Colour + GMM	82.48%	96.9%
Colour + GMM +Shape	97.43%	99.4%

Table 3.1: *Quantitative segmentation results. The table shows the effect of adding more information in the Bayesian framework on the quantitative segmentation accuracy. The accuracy was computed over all the pixels in the image. The ground truth for the data used in this experiment was generated by hand labelling the foreground and background regions in the images.*

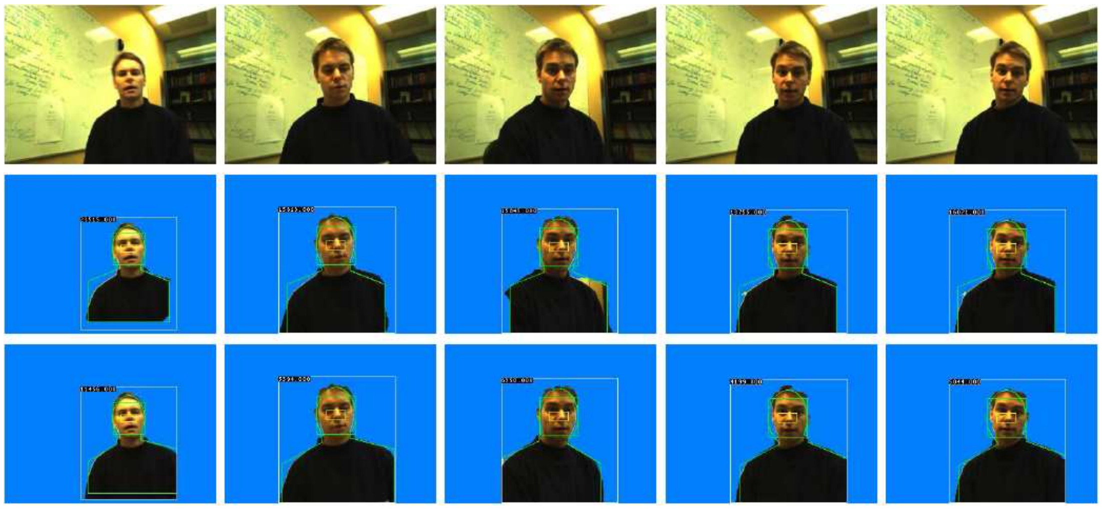


Figure 3.14: *Segmentation results using the 2D upper body model. The first row shows some frames from the video sequence. The second row shows the initial values of the pose parameters of the model and the resulting segmentations. The last row shows the final pose estimate and segmentation obtained using our method.*

segment the shadows of the feet as foreground. This is expected as they use only a pixelwise term to differentiate the background from the foreground and do not incorporate any spatial term which could offer a better "smoothing". In contrast, PoseCut which uses a pairwise potential term (as any standard graph cut approach) and a shape prior (which guarantees a human-like segmentation), is able to provide accurate results.

Our experiments on segmenting humans using the 2D upper body model (figure 3.8) also produced good results. For these experiments, video sequences from the Microsoft Research bilayer video segmentation dataset [52] were used. The results of our method are shown in figure 3.14.

3.2.3.2 Segmentation and pose estimation

Figures 3.16 and 3.17 present the segmentations and the pose estimates obtained using PoseCut. The first data set comprises of three views of human walking circularly. The time needed for computation of the 3D pose estimate, on an Intel Pentium 2GHz machine, when dealing with 644×484 images, is about 50 seconds per view[2]. As shown in these figures, the pose estimates match the original images accurately. In Figures 3.16 and 3.17, it should be noted that the appearance models of the foreground and background are quite similar: for

[2]However, this could be speeded up by computing the parameters of the CRF in an FPGA (Field programmable gate array).

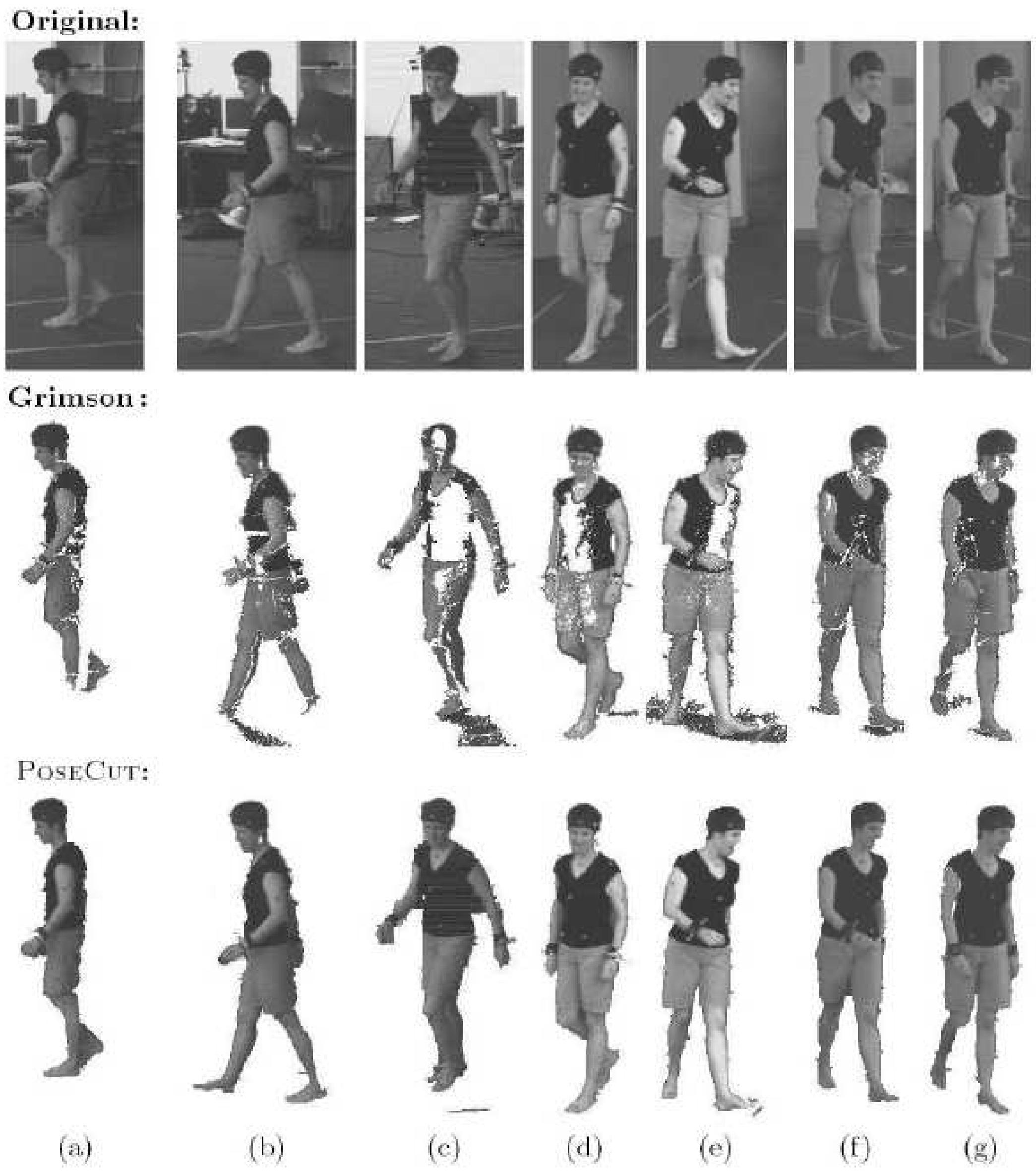


Figure 3.15: *Segmentation results obtained by Grimson-Stauffer [100] and* POSECUT.

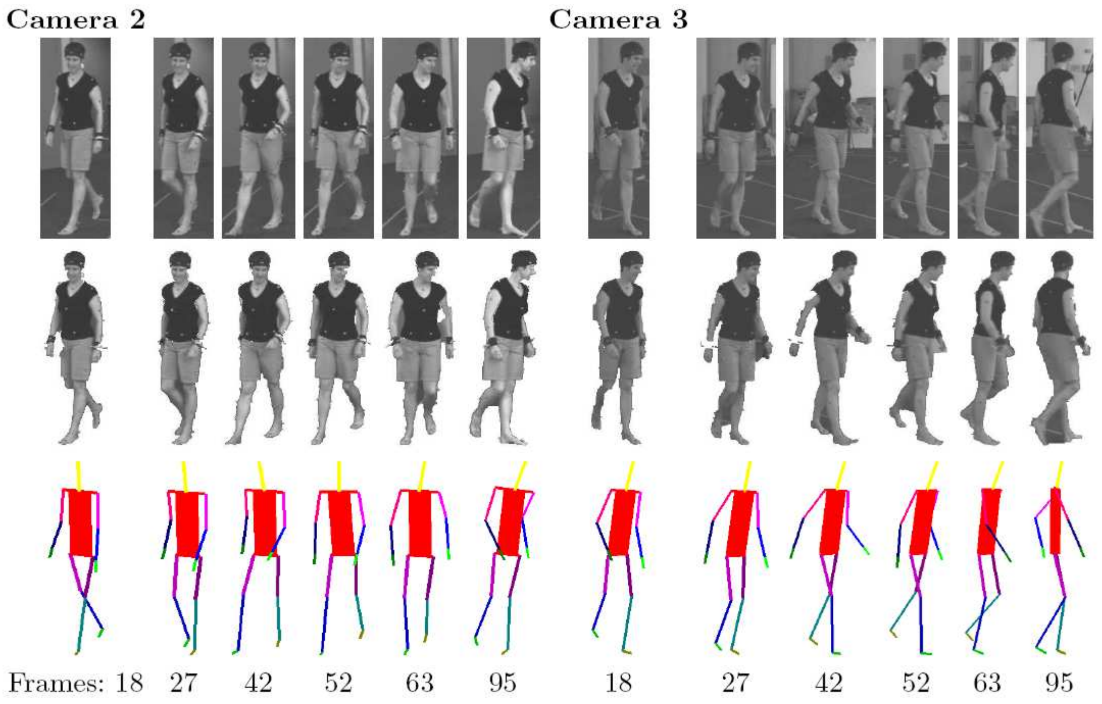


Figure 3.16: *Segmentation (middle) and pose estimation (bottom) results from* PoseCut.

instance, in Figure 3.17, the clothes of the subject are black in colour and the floor in the background is rather dark. The accuracy of the segmentation obtained in such challenging conditions demonstrates the robustness of PoseCut. An interesting fact to observe in Figure 3.16 about frame 95 is that the torso rotation of the stickman does not exactly conform with the original pose of the object. However, the segmentation of these frames is still accurate.

3.2.4 Shape Priors for Reconstruction

Obtaining a 3D reconstruction of an object from multiple images is a fundamental problem in computer vision. Reflecting the importance of the problem a number of methods have been proposed for its solution. These range from methods such as shape from silhouettes [103] and space carving [60] to image based methods [87]. However, the problem of obtaining accurate reconstructions from sparse multiple views still remains far from being solved. The primary problem afflicting reconstruction methods is the inherent ambiguity in the problem (as shown in figure 3.18) which arises from the many-one nature of the mapping that relates 3D objects and their images.

Intuitively the ambiguity in the object reconstruction can be overcome by us-

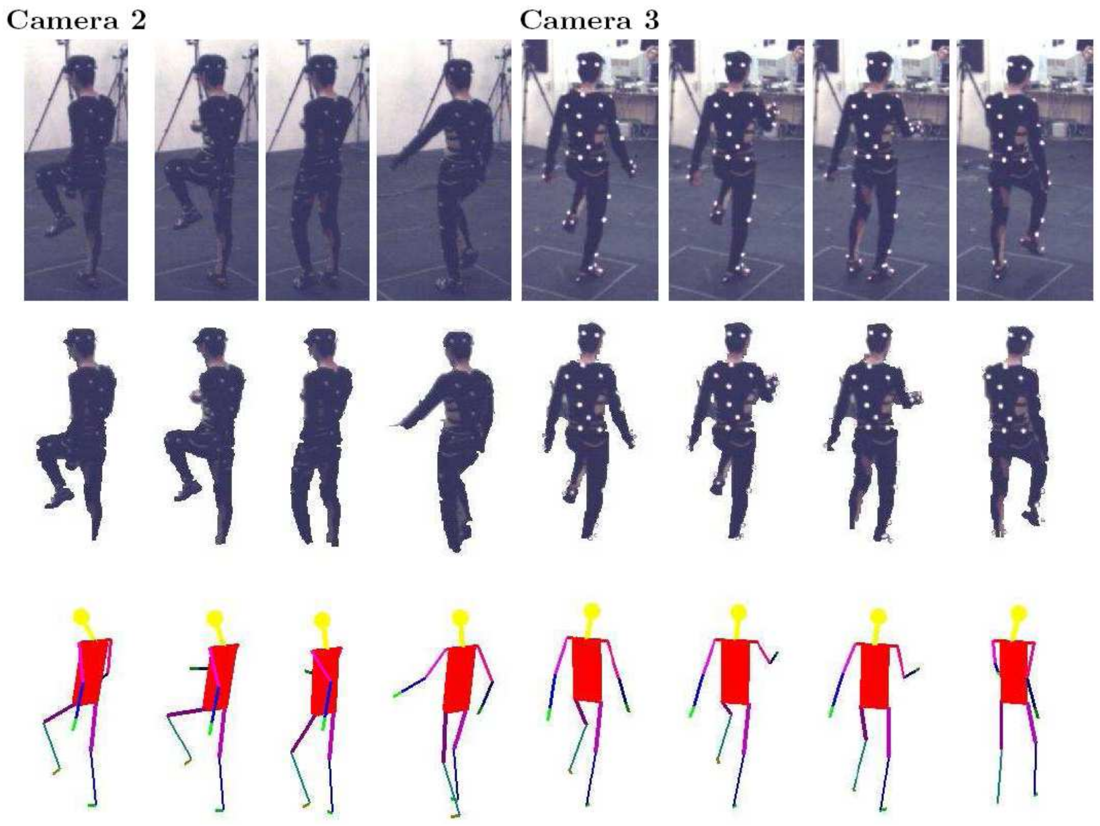


Figure 3.17: *Segmentation (middle row) and pose estimation (bottom row) results obtained using* PoseCut. *Observe that although the foreground and background appearances are similar, our algorithm is able to obtain good segmentations.*

ing prior knowledge. Researchers have long understood this fact and weak priors such as surface smoothness have been used in a number of methods [53, 99, 111]. Such priors help in recovering from the errors caused by noisy data. Although they improve results, they are weak and do not carry enough information to guarantee a unique solution. Taking inspiration from the success of using strong prior knowledge for image segmentation, we use 3D shape priors to overcome the ambiguity inherent in the problem of 3D reconstruction from multiple views.

Our framework uses a volumetric scene representation and integrates conventional reconstruction measures such as photoconsistency, surface smoothness and visual hull membership with a strong object specific prior. Simple parametric models of objects are used as strong priors in our framework. Our method not only gives an accurate object reconstruction, but also provides us the pose or state of the object being reconstructed. This work previously appeared in [102].

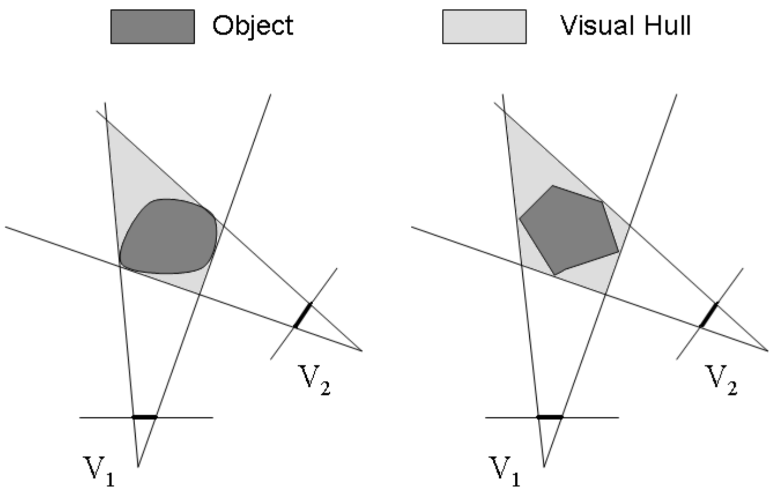
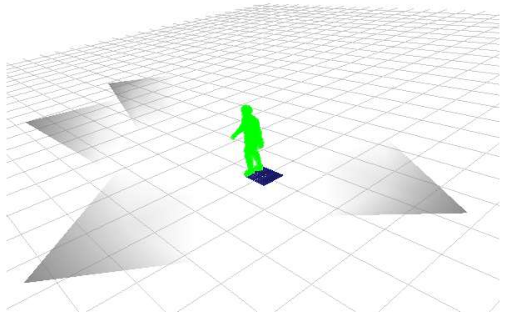


Figure 3.18: *Ambiguity in object reconstruction due to few views. The figure shows how two completely different objects can have the same visual hull. Further, if both objects have the same colour, the photo hull and their projections on multiple viewpoints would also be the same.*

Experimental Results The data set for our experiments consists of video sequences of four views of a human subject walking in a circle. This data set was earlier used in [3]. It comes with silhouettes of the human subject obtained using pixel wise background intensity modelling. The positions and orientations of the 4 cameras with respect to the object are shown in figure 3.19(i).

The first step in our method is the computation of the visual hull. The procedure starts with the quantization of the volume of interest as a grid of cubical voxels of equal size. Once this is done, each voxel center is projected into the input images. If any of the projections falls outside the silhouette, then the voxel is discarded. All remaining voxels constitute the visual hull. Some visual hull results are shown in figure 3.19(ii). It can be observed that because of the skewed distribution of the cameras, the visual hull is quite different from the true object reconstruction. Further, as object segmentations are not accurate, it has large errors. The prominent defects in the visual hull results include: (*i*) the presence of holes because of segmentation errors in the object silhouettes (bottom row (b)), (*ii*) the presence of auxiliary parts caused by shadows, (*iii*) the *third-arm effect* resulting from self-occlusion and ambiguity in the reconstruction due to the small number of views (bottom row (a)). It can be seen that our reconstruction results do not suffer from these errors (bottom row (c) and (d)). The final results of our method for a few frames of the human walking sequence



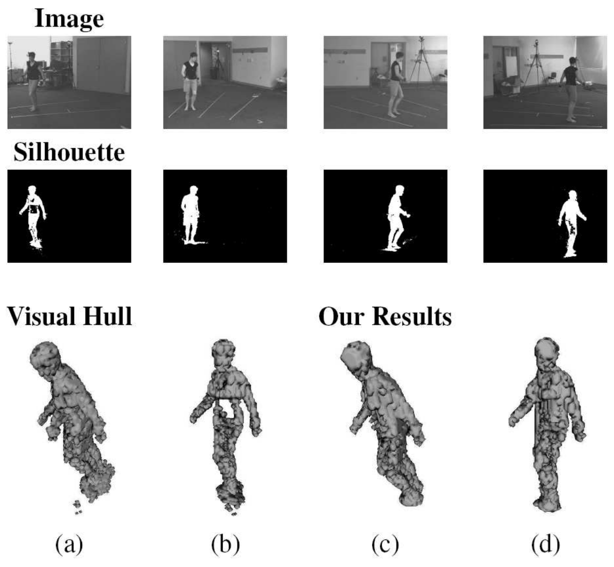


Figure 3.19: *i) Camera Positions and Reconstruction. The figure shows the position and orientations of the four cameras which were used to obtain the images which constituted the data-set for our first experiment. We also see the reconstruction result generated by our method. ii) 3D Object Reconstruction using Strong Object-Specific priors. The first and second rows show the images and silhouettes used as the data. Two views of the visual hull generated using the data are shown in the first two columns of the bottom row ((a) and (b)). The visual hull is noisy and contains artifacts like the spurious third arm caused by the ambiguity in the problem. We are able to overcome such problems by using strong prior knowledge. The reconstructions obtained by our method are shown in column 3 and 4 ((c) and (d)).*

are shown in figure 3.20.

3.2.5 Discussion

Localizing the object in the image and inferring its pose is a computationally expensive task. Once a rough estimate of the object pose is obtained, the segmentation can be computed extremely efficiently using graph cuts [12]. In our work on real time face detection and segmentation [81], we showed how an off the shelf face-detector such as the one described in [110] can be coupled with a CRF to get accurate segmentation and improved face detection results in real time.

The object (face) localization estimate (obtained from any generic face detector) was incorporated in a discriminative CRF framework to obtain robust and

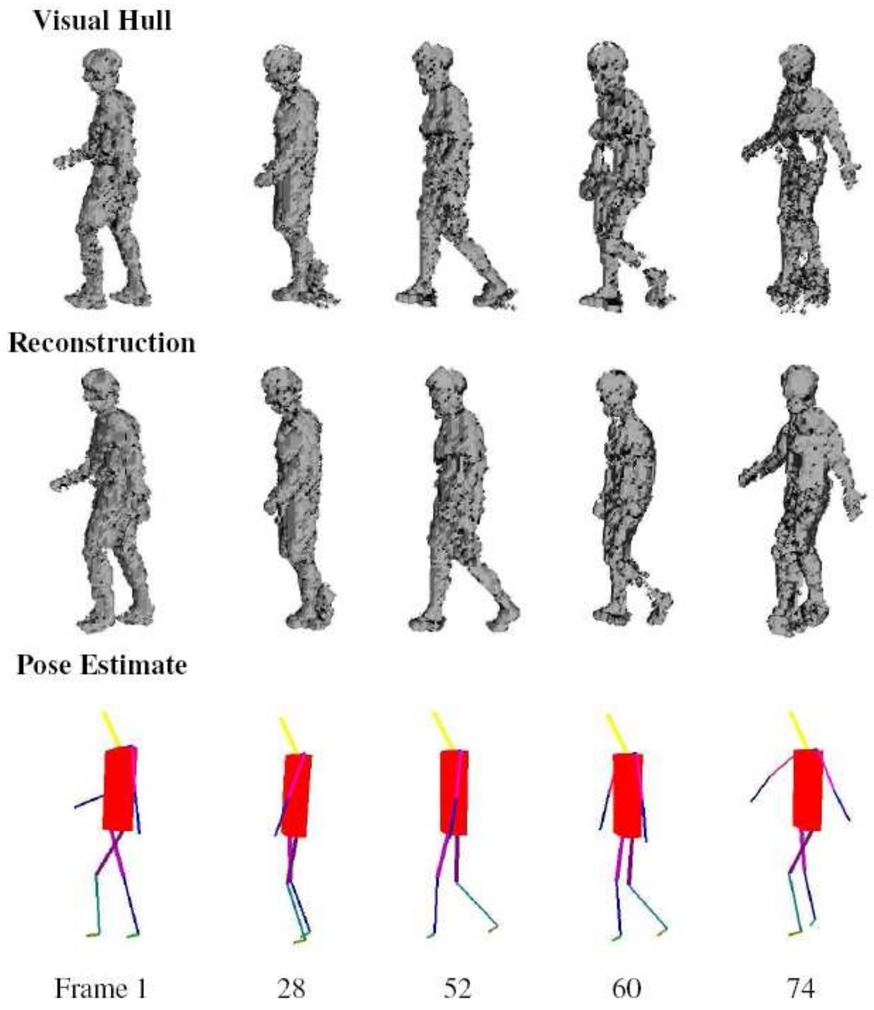


Figure 3.20: *Pose inference and 3D object reconstruction results. The data used in this experiment is taken from [3]. It consists of 4 views of a human subject walking in a circular path. Middle row: Reconstruction result. Bottom row: Pose estimate. Observe that we are able to get excellent reconstruction and pose estimation results even when the visual hull contains large errors (as seen in frame 60 and 74).*

accurate face segmentation results as shown in figure 3.21. The energy $E(\mathbf{x}^*)$ of any segmentation solution $\mathbf{x}^*$ is the negative log of the probability, and can be viewed as a measure of how uncertain that solution is. The higher the energy of a segmentation, the lower the probability that it is a good segmentation. Intuitively, if the face detection is correct, the resulting segmentation obtained from our method should have high probability and hence have low energy compared to that of false detections. This characteristic of the energy of the segmentation solution can be used to prune out false face detections thus improving the face

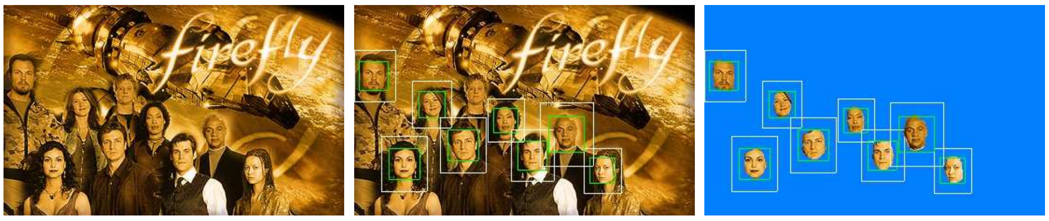


Figure 3.21: *Real time face segmentation using face detection. The first image in the first row shows the original image. The second image shows the face detection results. The third image shows the segmentation obtained using shape priors generated from the detection and localization results.*

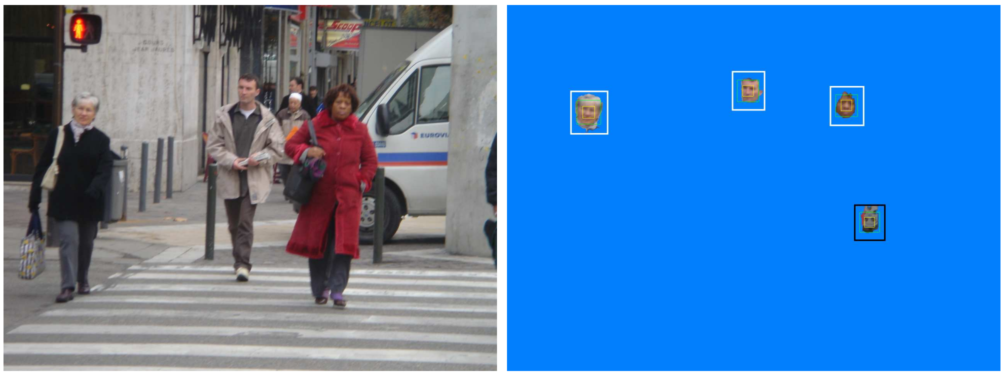


Figure 3.22: *Pruning false object detections. The figure shows an image from the INRIA pedestrian data set. After running our algorithm, we obtain four face segmentations, one of which (the one bounded by a black square) is a false detection. The energy-per-pixel values obtained for the true detections were 74, 82 and 83 while that for the false detection was 87. As you can see the energy of false detection is higher than that of the true detections, and can be used to detect and remove it.*

detection accuracy. The procedure is illustrated in figure 3.22. A similar strategy was recently used in [78].

3.2.6 Summary and Future Work

This work sets out a novel method for performing simultaneous segmentation and 3D pose estimation (PoseCut). The problem is formulated in a Bayesian framework which has the ability to utilize all information available (prior as well as observed data) to obtain good results. We showed how a rough pose-specific shape prior could be used to improve segmentation results significantly. We also gave a new formulation of the pose inference problem as an energy minimization problem and showed how it could be efficiently solved using dynamic graph cuts. The experiments demonstrate that our method is able to obtain excellent segmentation and pose estimation results. This method was recently also used for the problem of reconstructing objects from multiple views [102].

Searching over Pose Manifolds It is well known that the set of all human poses constitutes a low-dimensional manifold in the complete pose space [96,107]. Most work in exploiting this fact for human pose inference has been limited to finding linear manifolds in pose spaces. The last few years have seen the emergence of non-linear dimensionality reduction techniques for solving the pose inference problem [97]. Recently, Urtasun *et al.* [107] showed how Scaled Gaussian Process Latent Variable Models (SGPLVM) can be used to learn prior models of human pose for 3D people tracking. They showed impressive pose inference results using monocular data. Optimizing over a parametrization of this low dimensional space instead of the 26D pose vector would intuitively improve both the accuracy and computation efficiency of our algorithm. Thus the use of dimensionality reduction algorithms is an important area to be investigated. The directions for future work also include using an appearance model per limb, which being more discriminative could help provide more accurate segmentations and pose estimates.

3.3. Measuring Uncertainty in Graph Cut Solutions

Over the years researchers have asked the question whether it might be possible to compute a measure of uncertainty associated with the graph cut solutions. In this section we answer this particular question positively by showing how the min-marginals associated with the label assignments of a random field can be efficiently computed using a new algorithm based on dynamic graph cuts. The min-marginal energies obtained by our proposed algorithm are exact, as opposed to the ones obtained from other inference algorithms like loopy belief propagation and generalized belief propagation. We also show how these min-marginals can be used to compute a confidence measure for label assignments in the image segmentation problem.

3.3.1 Introduction

Graph cuts based minimization algorithms do not provide an uncertainty measure associated with the solution they produce. This is a serious drawback since researchers using these algorithms do not obtain any information regarding the probability of a particular latent variable assignment in a graph cut solution. Inference algorithms like LBP [73], GBP [118], and TRW [51,112] provide the user with marginal or min-marginal energies associated with each latent variable. However, these algorithms are not guaranteed to find the optimal solution for graphs of arbitrary topology. Note that for tree-structured graphs, the simple max-product belief propagation algorithm gives the exact max-marginal probabilities/min-marginal energies[1] for different label assignments in $O(nl^2)$ time where n is the number of latent variables, and l is the number of labels a latent variable can take.

We address the problem of efficiently computing the min-marginals associated with the label assignments of any latent variable in a MRF[2]. Our method works on all MRFs that can be solved exactly using graph cuts. First, we give the definition of *flow potentials* (defined in section 3.3.3.1) of a graph node. We show that the min-marginals associated with the labellings of a binary random variable are related to the flow-potentials of the node representing that variable in the graph constructed in the energy minimization procedure. In fact the exact min-marginal energies can be found by computing these *flow-potentials*. We then show how flow potential computation is equivalent to minimizing *projections* of the original energy function[3].

[1]We will explain the relation between max-marginal probabilities and min-marginal energies later in section 3.3.2. To make our notation consistent with recent work in graph cuts, we formulate the problem in terms of min-marginal energies (subsequently referred to as simply min-marginals).

[2]This first version of this section appeared as [49].

[3]A projection of the function $f(x_1, x_2, ..., x_n)$ can be obtained by fixing the values of some of the variables in the function $f(.)$. For instance $f_1(x_2, ..., x_n) = f(0, x_2, ..., x_n)$ is a projection of the function $f(.)$.

Minimizing a *projection* of an energy function is a computationally expensive operation and requires a graph cut to be computed. In order to obtain the min-marginals corresponding to all label assignments of all random variables, we need to compute a graph cut $O(nl)$ number of times. In this work, we present an algorithm based on dynamic graph cuts [48] which solves these $O(nl)$ graph cuts extremely quickly. Our experiments show that the running time of this algorithm i.e., the time taken for it to compute the min-marginals corresponding to all latent variable label assignments is of the same order of magnitude as the time taken to solve a single st-mincut problem.

3.3.2 Preliminaries

As explained in chapter 1, a *pairwise* MRF can be solved by minimizing a second order energy function. The energy of the MAP configuration of the MRF can be computed by solving the problem:

$$\psi(\theta) = \min_{\mathbf{x} \in \mathbf{L}} E(\mathbf{x}|\theta). \tag{3.3.1}$$

The MAP solution of the MRF will be referred to as the *optimal solution*.

3.3.2.1 Min-marginal energies

A min-marginal is a function that provides information about the minimum values of the energy E under different constraints. Following the notation of [51], we define the min-marginal energies $\psi_{v;j}, \psi_{uv;ij}$ as:

$$\begin{aligned} \psi_{v;j}(\theta) &= \min_{\mathbf{x} \in \mathbf{L}, x_v = j} E(\mathbf{x}|\theta), \quad \text{and} \\ \psi_{uv;ij}(\theta) &= \min_{\mathbf{x} \in \mathbf{L}, x_u = i, x_v = j} E(\mathbf{x}|\theta). \end{aligned} \tag{3.3.2}$$

In words, given an energy function E whose value depends on the variables $(x_1, \ldots, x_n)$, $\psi_{v;j}(\theta)$ represents the minimum energy value obtained if we fix the value of variable x_v to j and minimize over all remaining variables. Similarly, $\psi_{uv;ij}(\theta)$ represents the value of the minimum energy in the case when the values of variables x_u and x_v are fixed to i and j respectively.

3.3.3 Uncertainty in Label Assignments

Now we show how min-marginals can be used to compute a confidence measure for a particular latent variable label assignment. Given the function $\Pr(\mathbf{x}|\mathbf{D})$, which specifies the probability of a configuration of the MRF, the max-marginal

$\mu_{v;j}$ gives us the value of the maximum probability over all possible configurations of the MRF in which $x_v = j$. Formally, it is defined as:

$$\mu_{v;j} = \max_{\mathbf{x} \in \mathbf{L}; x_v = j} \Pr(\mathbf{x}|\mathbf{D}) \tag{3.3.3}$$

Inference algorithms like max-product belief propagation produce the max-marginals along with the MAP solution. These max-marginals can be used to obtain a confidence measure σ for any latent variable labelling as:

$$\sigma_{v;j} = \frac{\max_{\mathbf{x} \in \mathbf{L}, x_v = j} \Pr(\mathbf{x}|\mathbf{D})}{\sum_{k \in \mathcal{L}} \max_{\mathbf{x} \in \mathbf{L}, x_v = k} \Pr(\mathbf{x}|\mathbf{D})} = \frac{\mu_{v;j}}{\sum_{k \in \mathcal{L}} \mu_{v;k}} \tag{3.3.4}$$

where $\sigma_{v;j}$ is the confidence for the latent variable x_v taking label j. This is the ratio of the max-marginal corresponding to the label assignment $x_v = j$ to the sum of the max-marginals for all possible label assignments.

We now proceed to show how these max-marginals can be obtained from the min-marginal energies computed by our algorithm. Recall from equation (1.2.3) that the energy and probability of a labelling are related as:

$$E(\mathbf{x}) = -\log \Pr(\mathbf{x}|\mathbf{D}) - \text{const} \tag{3.3.5}$$

Substituting the value of $\Pr(\mathbf{x}|\mathbf{D})$ from equation (3.3.5) in equation (3.3.3), we get

$$\begin{aligned} \mu_{v;j} &= \max_{\mathbf{x} \in \mathbf{L}; x_v = j} (\exp{(-E(\mathbf{x}|\theta) - \text{const})}) && (3.3.6) \\ &= \frac{1}{Z} \exp{(-\min_{\mathbf{x} \in \mathcal{X}; x_v = j} E(\mathbf{x}|\theta))}, && (3.3.7) \end{aligned}$$

where Z is the partition function. Combining this with equation (3.3.2a), we get

$$\mu_{v;j} = \frac{1}{Z} \exp{(-\psi_{v;j}(\theta))}. \tag{3.3.8}$$

As an example consider a binary label object-background image segmentation problem, where there are two possible labels i.e., object (‘*ob*’) and background (‘*bg*’). The confidence measure $\sigma_{v;ob}$ associated with the pixel v being labelled as object can be computed as:

$$\sigma_{v;ob} = \frac{\mu_{v;ob}}{\mu_{v;ob} + \mu_{v;bg}} = \frac{\frac{1}{Z} \exp{(-\psi_{v;ob}(\theta))}}{\frac{1}{Z} \exp{(-\psi_{v;ob}(\theta))} + \frac{1}{Z} \exp{(-\psi_{v;bg}(\theta))}}, \tag{3.3.9}$$

$$\text{or} \quad \sigma_{v;ob} = \frac{\exp{(-\psi_{v;ob}(\theta))}}{\exp{(-\psi_{v;ob}(\theta))} + \exp{(-\psi_{v;bg}(\theta))}} \tag{3.3.10}$$

Note that the Z’s cancel and thus we can compute the confidence measure from the min-marginal energies alone without knowledge of the partition function.

3.3.3.1 Flow Potentials in Graphs

Given a directed weighted graph $G(V, E, C)$ with non-negative edge weights and flows f flowing through the edges E, we define the *source/sink flow potential* of a graph node $v \in V$ as the maximum amount of net flow that can be pumped into/from it without invalidating any edge capacity (1.3.2) or mass balance constraint (1.3.3) with the exception of the mass balance constraint of the node v itself. Formally, we can define the source flow potential of node v as:

$$f_v^s = \max_{\mathbf{f}} \sum_{i \in N(v)} f_{iv} - f_{vi}$$

Subject to:

$$0 \leq f_{ij} \leq c_{ij} \quad \forall (i,j) \in E, \quad \text{and} \tag{3.3.11}$$

$$\sum_{i \in \ N(j) \backslash \{s,t\}} (f_{ji} - f_{ij}) = f_{sj} - f_{jt} \quad \forall j \in V \backslash \{s, t, v\} \tag{3.3.12}$$

where $\max_{\mathbf{f}}$ represents the maximization over the set of all edge flows

$$\mathbf{f} = \{f_{ij}, \ \forall (i,j) \in E\}. \tag{3.3.13}$$

Similarly, the *sink flow potential* f_v^t of a graph node v is defined as:

$$f_v^t = \max_{\mathbf{f}} \sum_{i \in N(v)} f_{vi} - f_{iv} \tag{3.3.14}$$

subject to constraints (3.3.11) and (3.3.12).

The computation of a flow potential of a node is not a trivial process and in essence requires a graph cut to be computed as explained in figure 3.24. The flow potentials of a particular graph node are shown in figure 3.23. Note that in a residual graph $G(f_{\max})$ where $f_{\max}$ is the maximum flow, all nodes on the sink side of the st-mincut are disconnected from the source and thus have the source flow potential equal to zero. Similarly, all nodes belonging to the source have the sink flow potential equal to zero. We will later show that the flow-potentials we have just defined are intimately linked to the min-marginal energies of latent variable label assignments.

3.3.4 Computing Min-marginals using Graph Cuts

We now explain the procedure for the computation of min-marginal energies using graph cuts. The total flow f_{total} flowing from the source s to the sink t in a graph can be found by computing the difference between the total amount of flow coming in to a terminal node and that going out as:

$$f_{\text{total}} = \sum_{i \in N(s)} (f_{si} - f_{is}) = \sum_{i \in N(t)} (f_{it} - f_{ti}). \tag{3.3.15}$$

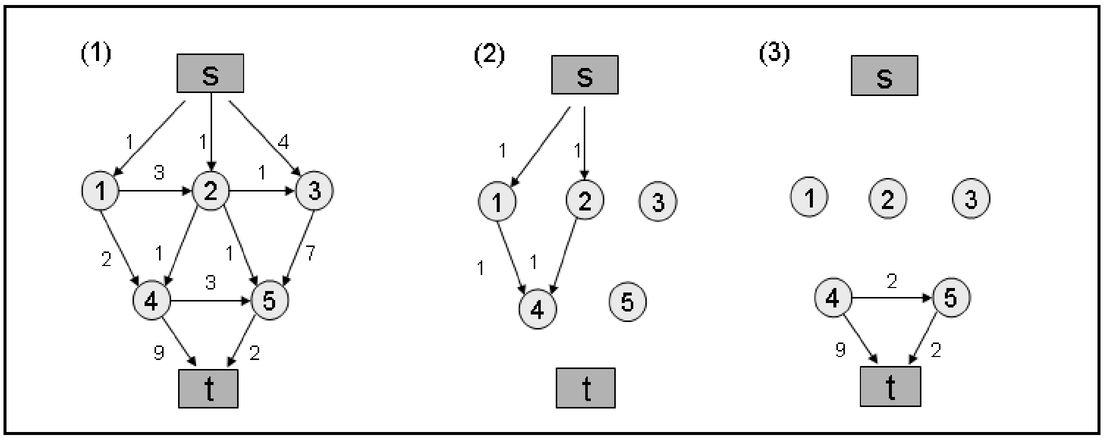


Figure 3.23: *Flow potentials of graph nodes. The figure shows a directed graph having seven nodes, two of which are the terminal nodes, the source s and the sink t. The number associated with each directed edge in this graph is a capacity which tells us the maximum amount of flow that can be passed through it in the direction of the arrow. The flow potentials for node 4 in this graph when no flow is passing through any of the edges are* $f_4^s = 2$ *and* $f_4^t = 11$.

The cost of the st-mincut in an energy representing graph is equal to the energy of the optimal configuration. From the Ford-Fulkerson theorem, this is also equal to the maximum amount of flow $f_{\max}$ that can be transferred from the source to the sink. Hence, from the minimum energy (3.3.1) and total flow equation (3.3.15) for a graph in which maxflow has been achieved i.e. $f_{\text{total}} = f_{\max}$, we obtain:

$$\psi(\theta) = \min_{\mathbf{x}\in\mathbf{L}} E(\mathbf{x}|\theta) = f_{\max} = \sum_{i\in N(s)} (f_{si} - f_{is}). \tag{3.3.16}$$

Note that flow cannot be pushed into the source i.e. $f_{is} = 0, \forall i \in V$, thus $\psi(\theta) = \sum_{i\in N(s)} f_{si}$. The MAP configuration $\mathbf{x}^*$ of a MRF is the one having the least energy and is defined as $\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbf{L}} E(\mathbf{x}|\theta)$.

Let a be the label for random variable x_v under the MAP solution and b be any label other than a. Then in the case of $x_v = a$, the min-marginal energy $\psi_{v;x_v^*}(\theta)$ is equal to the minimum energy i.e.

$$E(\mathbf{x}|\theta) = \psi(\theta) \tag{3.3.17}$$

Thus it can be seen that the maximum flow equals the min-marginals for the case when the latent variables take their respective MAP labels.

The min-marginal energy $\psi_{v;b}(\theta)$ corresponding to the non-optimal label b can be computed by finding the minimum value of the energy function projection E'

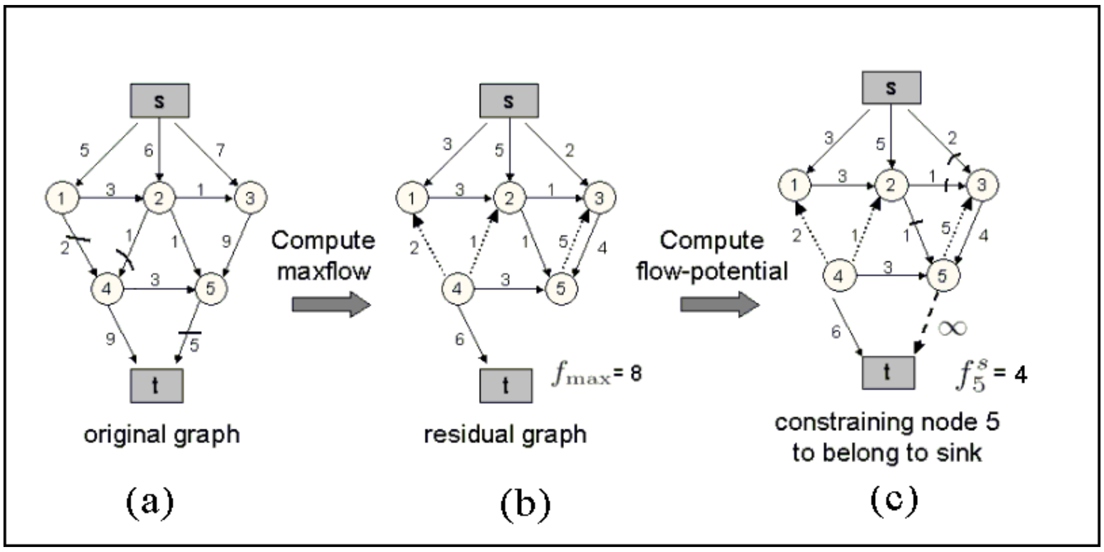


Figure 3.24: *Computing min-marginals using graph cuts. In (a) we see the graph representing the original energy function. This is used to compute the minimum value of the energy $\psi(\theta)$ which is equal to the max-flow $f_{\max} = 8$. The residual graph obtained after the computation of max-flow is shown in (b). In (c) we show how the flow-potential f_5^s can be computed in the residual graph by adding an infinite capacity edge between it and the sink and computing the max-flow again. The addition of this new edge constrains node 5 to belong to sink side of the st-cut. A max-flow computation in the graph (c) yields $f_5^s = 4$. This from theorem 1, we obtain the min-marginal $\psi_{5;c} = 8 + 4 = 12$, where T(c) = source(s). The dotted arrows in (b) and (c) correspond to edges in the residual graph whose residual capacities are due to flow passing through the edges in their opposite direction.*

obtained by constraining the value of x_v to b as:

$$\psi_{v;b}(\theta) = \min_{\mathbf{x}\in\mathcal{L}^n, x_v=b} E(\mathbf{x}|\theta) \tag{3.3.18}$$

$$\text{or, } \psi_{v;b}(\theta) = \min_{(\mathbf{x}-x_v)\in\mathcal{L}^{n-1}} E(x_1, .., x_{v-1}, b, x_{v+1}..x_n|\theta). \tag{3.3.19}$$

In the next sub-section, we will show that this constraint can be enforced in the original graph construction used for minimizing $E(x|\theta)$ by modifying certain edge weights ensuring that the latent variable x_v takes the label b. The exact modifications needed in the graph for the binary label case are given first, while those required in the graph for the multi-label case are discussed later.

3.3.4.1 Min-marginals and Flow potentials

We now show how in the case of binary variables, flow-potentials in the residual graph $G(f_{\max})$ are related to the min-marginal energy values. Again, a and b are used to represent the MAP and non-MAP label respectively.

Theorem 1 *The min-marginal energies of a binary latent variable x_v are equal to the sum of the max-flow and the source/sink flow potentials of the node representing it in the residual graph $G(f_{\max})$ corresponding to the max-flow solution i.e.*

$$\psi_{v;j}(\theta) = \min_{x \in \mathbf{L}, x_v = j} E(\mathbf{x}|\theta) = \psi(\theta) + f_v^{T(j)} = f_{\max} + f_v^{T(j)} \tag{3.3.20}$$

where $T(j)$ is the terminal corresponding to the label j, and $f_{\max}$ is the value of the maximum flow in the graph G representing the energy function $E(\mathbf{x}|\theta)$.

Proof The proof is trivial for the case where the latent variable takes the optimal label. We already know that the value of the min-marginal $\psi_{v;a}(\theta)$ is equal to the lowest energy $\psi(\theta)$. Further, the flow potential of the node for the terminal corresponding to the label assignment is zero since the node is disconnected from the terminal $T(a)$ by the minimum cut[4].

We already know from (3.3.19) that the min-marginal $\psi_{v;b}(\theta)$ corresponding to the non-optimal label b can be computed by finding the minimum value of the function E under the constraint $x_v = b$. This constraint can be enforced in our original graph (used for minimizing $E(x|\theta)$) by adding an edge with infinite weight between the graph node and the terminal corresponding to the label a, and then computing the st-mincut on this updated graph[5]. In section 3.3.4.3 we shall explain how to solve the new st-mincut problem efficiently using the dynamic graph cut algorithm proposed in the previous chapter.

It can be easily seen that the additional amount of flow that would now flow from the source to the sink is equal to the flow potential $f_v^{T(b)}$ of the node. Thus the value of the max-flow now becomes equal to $\psi(\theta) + f_v^{T(b)}$ where $T(b)$ is the terminal corresponding to the label b. The whole process is shown graphically in figure 3.24. ∎

We have shown how minimizing an energy function with constraints on the value of a latent variable, is equivalent to computing the flow potentials of a node

[4]The amount of flow that can be transferred from the node to the terminal $T(a)$ in the residual graph is zero since otherwise it would contradict our assumption that the max-flow solution has been achieved.

[5]Adding an infinite weight edge between the node and the terminal $T(a)$ is equivalent to putting a hard constraint on the variable x_v to have the label b. Observe that the addition of an infinite weight edge can be realized by using an edge whose weight is more than the sum of all other edges incident on the node. This condition would make sure that the edge is not saturated during the max-flow computation.

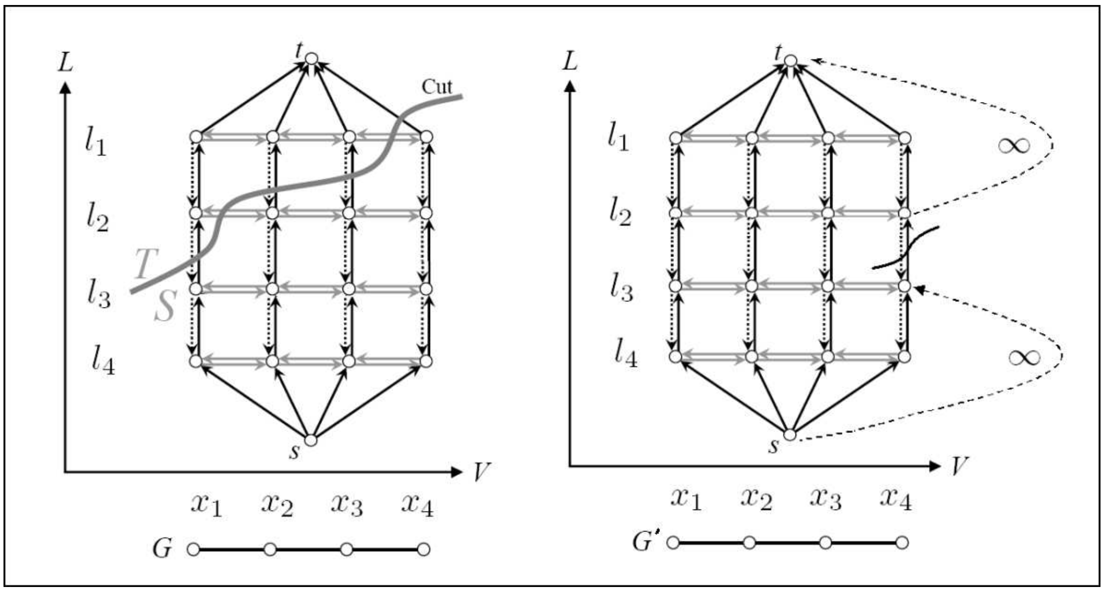


Figure 3.25: *Graph construction for projections of energy functions involving multiple labels. The first graph G shows the graph construction proposed by Ishikawa [39] for minimizing energy functions representing* MRF*s involving latent variables which can take more than 2 labels. All the label sets $\mathcal{L}$, $v \in V$ consist of 4 labels namely l_1, l_2, l_3 and l_4. The* MAP *configuration of the* MRF *induced by the st-mincut is found by observing which data edges are cut (data edges are depicted as black arrows). Four of them are in the cut here (as seen in graph G), representing the assignments $x_1 = l_2$, $x_2 = l_3$, $x_3 = l_3$, and $x_4 = l_4$. The graph G' representing the projection $E' = E(x_1, x_2, x_3, l_2)$ can be obtained by inserting infinite capacity edges from the source and the sink to the tail and head node respectively of the edge representing the label l_2 for latent variable x_4.*

in the residual graph $G(f_{\max})$. Note that a similar procedure can be used to compute the min-marginal $\psi_{uv;ij}(\theta)$ by taking the projection and enforcing hard constraints on pairs of latent variables.

3.3.4.2 Extension to Multiple labels

Graph cuts can also be used to exactly optimize convex energy functions which involve variables taking multiple labels [39, 88]. Graphs representing the projections of such energy functions can be obtained by incorporating hard constraints in a fashion analogous to the one used for binary variables. In the graph construction for multiple labels proposed by Ishikawa [39], the label of a discrete latent variable is found by observing which data edge is cut. The value of a variable can be constrained or 'fixed' in this graph construction by making sure that the data

1. Construct graph G for minimizing the MRF energy E.
2. Compute the maximum s-t flow in the graph. This induces the residual graph G_r consisting of unsaturated edges.
3. For computing each min-marginal, perform the following operations:
   - 3.1 Obtain the energy projection E' corresponding to the latent variable assignment.
   - 3.2 Construct the graph G' to minimize E'.
   - 3.3 Use dynamic graph cut updates as given in [48] to make G_r consistent with G', thus obtaining the new graph $G^{'}_r$.
   - 3.4 Compute the min-marginal by minimizing E' using the dynamic (optimized) st-mincut algorithm on $G^{'}_r$.

Table 3.2: *Algorithm for computing min-marginal energies using dynamic graph cuts.*

edge corresponding to the particular label is cut. This can be realized by adding edges of infinite capacity from the source and the sink to the tail and head node of the edge respectively as shown in figure 3.25. The cost of the st-mincut in this modified graph will give the exact value of min-marginal energy associated with that particular labelling. It should be noted here that the method of Ishikawa [39] applies to a restricted class of energy functions. These do not include energies with non-convex priors (such at the Potts model) which are used in many computer vision applications. Measuring uncertainty in solutions of such energies is thus still an open problem.

3.3.4.3 Minimizing Energy Function Projections using Dynamic Graph Cuts

Having shown how min-marginals can be computed using graph cuts, we now explain how this can be done efficiently. As explained in the proof of Theorem 1, we can compute min-marginals by minimizing projections of the energy function. However, it might be thought that such a process is extremely computationally expensive as a graph cut has to be computed for every min-marginal computation. However, when modifying the graph in order to minimize the projection $E^{'}$ of the energy function, only a few edge weights have to be changed[6] as seen in figure 3.24, where only one infinite capacity edge had to inserted in the graph. We have shown earlier how the st-mincut can be recomputed rapidly for such minimal changes in the problem by using dynamic graph cuts. Our proposed algorithm for min-marginal computation is given in Table 3.2.

3.3.5 Computational Complexity and Experimental Evaluation

We now discuss the computational complexity of our algorithm, and report the time taken by it to compute min-marginals in MRFs of different sizes. In step (3.4) of the algorithm given in Table 3.2, the amount of flow computed is equal to the difference in the min-marginal $\psi_{v;j}(\theta)$ of the particular label assignment and the minimum energy $\psi(\theta)$. Let $\mathcal{Q}$ be the set of all label assignments whose corresponding min-marginals have to be computed. Then the number of augmenting paths to be found during the whole algorithm is bounded from above by:

$$U = \psi(\theta) + \sum_{q \in \mathcal{Q}} (\psi_q(\theta) - \psi(\theta)). \tag{3.3.21}$$

For the case of binary random variables, assuming that we want to compute all latent variable min-marginals i.e.

$$\mathcal{Q} = \{(u; i) : u \in V, i \in \mathcal{L}\}, \text{and} \tag{3.3.22}$$

$$q_{max} = \max_{q \in \mathcal{Q}} (\psi_q(\theta) - \psi(\theta)), \tag{3.3.23}$$

the complexity of the above algorithm becomes $\mathrm{O}((\psi(\theta) + nq_{max})T(n, m))$, where $T(n, m)$ is the complexity of finding an augmenting path in the graph with n nodes and m edges and pushing flow through it. Although the worst case complexity $T(n, m)$ of the augmentation operation is $\mathrm{O}(m)$, we observe experimentally that using the dual search tree algorithm of [10], we can get a much better amortized time performance. The average time taken by our algorithm for computing the min-marginals in random MRFs of different sizes is given in table 3.3.

3.3.6 Applications of Min-marginals

Min-marginal energies have been used for a number of different purposes. However, prior to our work, the use of min-marginals in computer vision was severely

[6]The exact number of edge weights that have to be changed is of the order of the number of variables whose value is being fixed for obtaining the projection.

MRF size	10^5	2×10^5	4×10^5	8×10^5
4-neighbourhood	0.18, 0.70	0.46, 1.34	0.92, 3.156	2.17, 8.21
8-neighbourhood	0.40, 1.53	1.39, 3.59	2.42, 8.50	5.12, 15.61

Table 3.3: *Time taken for min-marginal computation. For a sequence of randomly generated* MRFs *of a particular size and neighbourhood system, a pair of times (in seconds) is given in each cell of the table. On the left is the average time taken to compute the* MAP *solution using a single graph cut while on the right is the average time taken to compute the min-marginals corresponding to all latent variable label assignments. The dynamic algorithm with tree-recycling was used for this experiment. All experiments were performed on an Intel Pentium 2GHz machine.*

restricted. This was primarily due to the fact that they were computationally expensive to compute for MRFs having a large number of latent variables. Our new algorithm is able to handle large MRFs which opens up possibilities for many new applications. For instance, in the experiments shown in figure 3.26, the time taken for computing all min-marginals for a MRF consisting of 2×10^5 binary latent variables was 1.2 seconds. This is roughly four times the time taken for computing the MAP solution of the same MRF by solving a single st-mincut problem.

Min-marginals as a confidence measure We had shown in section 3.3.2 how min-marginals can be used to compute a confidence measure for any latent variable assignment in a MRF. Figure 3.26 shows the confidence values obtained for the MRF used for modeling the two label (foreground and background) image-segmentation problem as defined in [9]. Ideally we would like the confidence map to be black and white showing extremely 'low' or 'high' confidence for a particular label assignment. However, as can be seen from the result, the confidence map contains regions of different shades of grey. Such confidence maps can be used for many vision applications. For instance, they could be used in interactive image segmentation to direct user interaction at regions which have high uncertainty. They can also be applied in coarse-to-fine techniques for efficient computation of low level vision problems. Here confidence maps could be used to isolate variables which have low confidence in the optimal label assignment. These variables can be solved at higher resolution to get a better solution.

Computing the M most probable configurations One of the most important uses of min-marginals has been to find the M most probable configurations

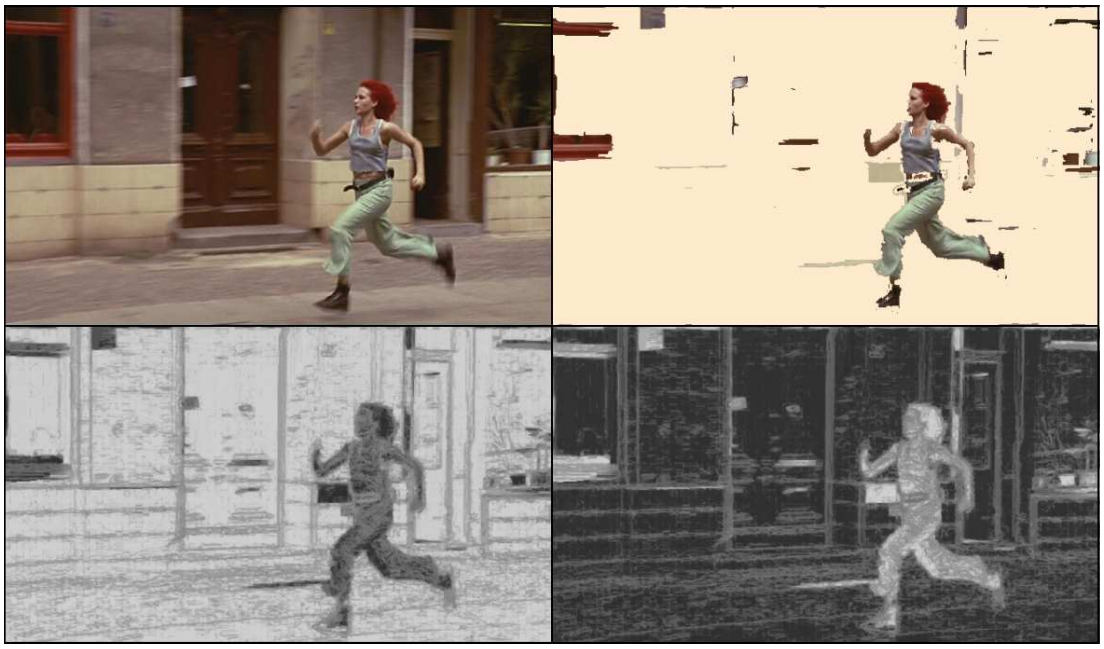


Figure 3.26: *Image segmentation with max-marginal probabilities. The first image is a frame of the movie Run Lola Run. The second shows the binary foreground-background segmentation where the aim was to segment out the human. The third and fourth images shows the confidence values obtained by our algorithm for assigning pixels to be foreground and background respectively. In the image, the max-marginal probability is represented in terms of decreasing intensity of the pixel. Our algorithm took 1.2 seconds for computing the max-marginal probabilities for each latent variable label assignment. The time taken to compute the* MAP *solution was 0.3 seconds.*

(or labellings) for latent variables in a Bayesian network [117]. Dawid [19] showed how min-marginals on junction trees can be computed. This method was later used by [70] to find the M most probable configurations of a probabilistic graphical network. The method of [19] is guaranteed to run in polynomial time for tree-structured networks. However, for arbitrary graphs, its worst case complexity is exponential in the number of the nodes in the graphical model. By using our method, the M most probable solutions of some graphical models with loops can be computed in reasonable time.

We end this section by giving a brief summary of this work. We have addressed the long-standing problem of computing the exact min-marginals for graphs with arbitrary topology in polynomial time. We propose a novel algorithm based on dynamic graph cuts [48] that computes the min-marginals extremely efficiently. Our algorithm makes it feasible to compute exact min-marginals for MRFs with large number of latent variables. This opens up many new applications for min-marginals which were not feasible earlier.

Chapter 4

Minimizing Higher Order Functions

The energy functions typically used for modelling computer vision problems are written as a sum of unary and pairwise clique potentials. This representation severely restricts the expressive power of these models making them unable to capture the rich statistics of natural scenes [63]. Higher order clique potentials are able to model complex interactions of random variables, and thus could overcome this problem. Researchers have long recognized this fact and have used higher order functions to improve the expressive power of the MRF and CRF frameworks [63, 72, 82]. The initial work in this regard has been quite promising and higher order clique potentials have been shown to improve results. However, their use has still been very limited. This is primarily due to the lack of efficient algorithms for minimizing energy functions composed of such potentials.

Traditional inference algorithms such as BP are quite computationally expensive for potentials defined on higher order cliques. Lan *et al.* [63] recently made some progress towards solving this problem. They proposed approximation methods for BP to make efficient inference possible in higher order MRFs. However their results indicate that BP gave results comparable to standard gradient descent. In contrast, we use powerful move making algorithms such as α-expansion and $\alpha\beta$-swap to minimize such functions. In this chapter, we provide a characterization of energy functions defined on cliques of size 3 or more which can be solved using these algorithms. More specifically, we prove that the optimal α-expansion and $\alpha\beta$-swap moves for this class of functions can be computed in polynomial time by minimizing a submodular function. It should be noted that our results are a generalization of the class of energy functions specified by [11]. We also give examples of higher order potential functions for which it is NP-hard to compute the optimal move.

4.1. Move Making Algorithms

Many energy functions encountered in computer vision are NP-hard to minimize. They are instead solved using algorithms for approximate energy minimization. These algorithms can be divided into two broad categories: message passing algorithms such as belief propagation and its variants [54, 112, 118], and move making algorithms such as Iterated Conditional Modes (ICM), α-expansion, and $\alpha\beta$-swap.

Move making algorithms start from an initial solution and proceed by making a series of changes which lead to solutions having lower energy. At each step, the algorithms search a move space and choose the move which leads to the solution having the lowest energy. This move is referred to as the *optimal* move. The algorithm is said to converge when no lower energy solution can be found.

The size of the move space is the defining characteristic of any move making algorithm. A large move space means that bigger changes to the current solution can be made. This makes the algorithm less prone to getting stuck in local minima and also results in a faster rate of convergence. Our work deals with two particular *large* move making algorithms, namely *α-expansion* and *$\alpha\beta$-swap* [11], whose move space size increases exponentially with the number of variables involved in the energy function.

Both α-expansion and $\alpha\beta$-swap are extremely efficient and have been shown to produce good results for a number of computer vision problems [104]. The moves of these algorithms can be encoded by a vector of binary variables $\mathbf{t} = \{t_i, \forall i \in \mathcal{V}\}$. At each step of these algorithms, the *optimal* move, i.e. the move decreasing the energy of the labelling by the most amount is computed in polynomial time. However, this can only be done for a certain class of energy functions.

Boykov *et al.* [11] provided a characterization of potential functions for which the optimal moves can be computed by solving an st-mincut problem. However, their results were limited to potential functions defined over cliques of size at most two. We call this class of energy functions $\mathcal{P}^2$. In this chapter we extend the results of [11] by providing a characterization of higher order energy functions (which are defined over cliques of sizes 3 and beyond) for which the optimal moves can be computed in polynomial time. We refer to the class of functions defined on cliques of size n as $\mathcal{P}^n$. It should be noted that this class is different from the class $\mathcal{F}^n$ of energy functions which involve only binary random variables [27, 54].

4.1.1 Previous Work

We will now describe the move making algorithms of [11] for approximate energy minimization, and explain the conditions under which they can be applied. Boykov *et al.* [11] addressed the problem of minimizing energy functions consisting of unary and pairwise potentials. These functions can be written as

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{i \in \mathcal{V}, j \in \mathcal{N}_j} \psi_{ij}(x_i, x_j). \tag{4.1.1}$$

They proposed two move making algorithms called α-expansion and $\alpha\beta$-swap for this problem. These algorithms work by starting from an initial labelling $\mathbf{x}$ and making a series of changes (moves) which lower the energy iteratively. Convergence is achieved when the energy cannot be decreased further. At each step, the move decreasing the energy of the labelling by the most amount is made. We will refer to such a move as *optimal.*

Boykov *et al.* [11] showed that the optimal moves for certain energy functions of the form (4.1.1) can be computed by solving an st-mincut problem. Specifically, they showed that if the pairwise potential functions ψ_{ij} define a metric then the energy function in equation (4.1.1) can be approximately minimized using α-expansion. Similarly, if ψ_{ij} is a semi-metric, it can be minimized using $\alpha\beta$-swap.

4.1.1.1 Metric and Semi-metric Potential functions

We now provide the constraints under which pairwise potentials are said to define a metric or a semi-metric.

Semimetric: A potential function $\psi_{ij}(a,b)$ defined on a clique of two random variables $\{x_i, x_j\}$ is said to be a *semi-metric* if it satisfies the conditions

$$\psi_{ij}(a,b) = 0 \iff a = b, \tag{4.1.2}$$

$$\psi_{ij}(a,b) = \psi_{ij}(b,a) \geq 0 \tag{4.1.3}$$

for all labels a and b in $\mathcal{L}$.

Metric: A potential function is *metric* if in addition to the above mentioned constraints it also satisfies the condition

$$\psi_{ij}(a,d) \leq \psi_{ij}(a,b) + \psi_{ij}(b,d) \tag{4.1.4}$$

for all a, b and d in $\mathcal{L}$. For example, the function $\psi_{ij}(a,b) = |a-b|^2$ is a semi-metric but not a metric as it does not always satisfy condition (4.1.4).

4.1.2 Binary Moves and Move Energies

The moves of both α-expansion and $\alpha\beta$-swap algorithms can be encoded by a vector of binary variables $\mathbf{t} = \{t_i, \forall i \in \mathcal{V}\}$. A *transformation* function $T(\mathbf{x}, \mathbf{t})$ takes the current labelling $\mathbf{x}$ and a move $\mathbf{t}$ and returns the new labelling $\hat{\mathbf{x}}$ which has been induced by the move. The energy of a move $\mathbf{t}$ (denoted by $E_m(\mathbf{t})$) is defined as the energy of the labelling $\hat{\mathbf{x}}$ it induces, i.e. $E_m(\mathbf{t}) = E(T(\mathbf{x}, \mathbf{t}))$. The optimal move is defined as $\mathbf{t}^* = \arg\min_{\mathbf{t}} E(T(\mathbf{x}, \mathbf{t}))$.

The optimal move $\mathbf{t}^*$ can be computed in polynomial time if the function $E_m(\mathbf{t})$ is submodular. Recall that a function $f : \mathcal{L}^n \to R$ is submodular if and only if all its projections on 2 variables are submodular [8, 54]. This definition implies that for the optimal move to be computable in polynomial time all projections of $E_m(\mathbf{t})$ on two variables should be submodular, i.e.

$$E_m^p(0,0) + E_m^p(1,1) \leq E_m^p(0,1) + E_m^p(1,0), \quad \forall p \in \mathcal{V} \times \mathcal{V}. \tag{4.1.5}$$

4.1.2.1 The α-expansion algorithm

An α-expansion move allows any random variable to either retain its current label or take label 'α'. One iteration of the algorithm involves making moves for all α in $\mathcal{L}$ in some order successively. Recently an alternative interpretation of α-expansions was given in [55]. They showed that α-expansion can be seen as a special case of a primal-dual schema based method for solving the energy minimization problem.

The transformation function $T_\alpha()$ for an α-expansion move transforms the label of a random variable X_i as

$$T_\alpha(x_i, t_i) = \begin{cases} x_i & \text{if} \quad t_i = 0 \\ \alpha & \text{if} \quad t_i = 1. \end{cases} \tag{4.1.6}$$

The optimal α-expansion move can be computed in polynomial time if the energy function $E_\alpha(\mathbf{t}) = E(T_\alpha(\mathbf{x}, \mathbf{t}))$ satisfies constraint (4.1.5). Substituting the value of E_α in (4.1.5) we get the constraint

$$E^p(x_i, x_j) + E^p(\alpha, \alpha) \leq E^p(x_i, \alpha) + E^p(\alpha, x_j), \quad \forall p \in \mathcal{V} \times \mathcal{V}. \tag{4.1.7}$$

The condition given in [11], i.e. if the pairwise potential functions ψ_{ij} define a metric then the energy function can be approximately minimized using the α-expansion algorithm, easily follows by observing that all metric potentials satisfy equation (4.1.7).

4.1.2.2 The $\alpha\beta$-swap algorithm

An $\alpha\beta$-swap move allows a random variable whose current label is α or β to either take label α or β. One iteration of the algorithm involves performing swap moves for all α and β in $\mathcal{L}$ in some order successively. The transformation function $T_{\alpha\beta}()$ for an $\alpha\beta$-swap transforms the label of a random variable x_i as

$$T_{\alpha\beta}(x_i, t_i) = \begin{cases} \alpha & \text{if} \quad x_i = \alpha \text{ or } \beta \text{ and } t_i = 0, \\ \beta & \text{if} \quad x_i = \alpha \text{ or } \beta \text{ and } t_i = 1. \end{cases} \tag{4.1.8}$$

The optimal $\alpha\beta$-swap move can be computed in polynomial time if the energy function

$$E_{\alpha\beta}(\mathbf{t}) = E(T_{\alpha\beta}(\mathbf{x}, \mathbf{t})) \tag{4.1.9}$$

satisfies (4.1.5). As before, substituting the value of $E_{\alpha\beta}$ in (4.1.5) we get the constraint

$$E^p(\alpha, \alpha) + E^p(\beta, \beta) \leq E^p(\alpha, \beta) + E^p(\beta, \alpha), \quad \forall p \in \mathcal{V} \times \mathcal{V}. \tag{4.1.10}$$

The condition given in [11], i.e. if the pairwise potential functions ψ_{ij} define a semimetric then the energy function can be approximately minimized using the $\alpha\beta$-swap algorithm, can be easily verified by observing that all semi-metric potentials satisfy equation (4.1.10).

4.2. Characterizing Solvable $\mathcal{P}^n$ Functions

In this section we show how the above mentioned move algorithms can be used to minimize higher order energy functions. Specifically, we characterize a class of higher order clique potentials for which the expansion and swap moves can be computed in polynomial time. Recall that $\mathcal{P}^n$ functions are defined on cliques of size at most n. From the additivity theorem [54] it follows that the optimal moves for all energy functions composed of these clique potentials can be computed in polynomial time. We constrain the clique potentials to take the form:

$$\psi_c(\mathbf{x}_c) = f_c(\mathcal{Q}_c(\oplus, \mathbf{x}_c)). \tag{4.2.1}$$

where $\mathcal{Q}_c(\oplus, \mathbf{x}_c)$ is a functional defined as:

$$\mathcal{Q}_c(\oplus, \mathbf{x}_c) = \oplus_{i,j\in c}\phi_c(x_i, x_j). \tag{4.2.2}$$

Here f_c is an arbitrary function of $\mathcal{Q}_c$, ϕ_c is a pairwise function defined on all pairs of random variables in the clique c, and $\oplus$ is an operator applied on these functions $\phi_c(x_i, x_j)$.

4.2.1 Conditions for $\alpha\beta$-swaps

We will now specify the constraints under which all $\alpha\beta$-swap moves for higher order clique potentials can be computed in polynomial time. For the moment we consider the case $\oplus = \sum$, i.e.

$$\mathcal{Q}_c(\mathbf{x}_c) = \sum_{i,j\in c} \phi_c(x_i, x_j). \tag{4.2.3}$$

Theorem 2 *The optimal $\alpha\beta$-swap move for any α and $\beta \in \mathcal{L}$ can be computed in polynomial time for a potential function $\psi_c(\mathbf{x}_c)$ of the form (4.2.1) if $f_c(\cdot)$ is a concave*[1] *non-decreasing function, $\oplus = \sum$ and $\phi_c(\cdot,\cdot)$ satisfies the constraints*

$$\phi_c(a, b) = \phi_c(b, a) \quad \forall a, b \in \mathcal{L} \tag{4.2.4}$$

$$\phi_c(a, b) \geq \phi_c(d, d) \quad \forall a, b, d \in \mathcal{L} \tag{4.2.5}$$

[1] A function $f(x)$ is concave if for any two points (a, b) and λ where $0 \leq \lambda \leq 1$: $\lambda f(a) + (1 - \lambda)f(b) \leq f(\lambda a + (1 - \lambda)b)$.

Proof

To prove that the optimal swap move can be computed in polynomial time we need to show that all projections on two variables of any $\alpha\beta$-swap move energy are submodular. From equation (4.1.10) this implies that $\forall i, j \in c$ the condition:

$$\begin{aligned}\psi_c(\{\alpha,\alpha\}\cup \mathbf{x}_{c\setminus\{i,j\}}) &+ \psi_c(\{\beta,\beta\}\cup \mathbf{x}_{c\setminus\{i,j\}}) \leq \\ \psi_c(\{\alpha,\beta\}\cup \mathbf{x}_{c\setminus\{i,j\}}) &+ \psi_c(\{\beta,\alpha\}\cup \mathbf{x}_{c\setminus\{i,j\}}) \qquad (4.2.6)\end{aligned}$$

should be satisfied. Here $\mathbf{x}_{c\setminus\{i,j\}}$ denotes the labelling of all variables $X_u, u \in c$ except i and j. We use $(\{l_a, l_b\}\cup\mathbf{x}_{c\setminus\{i,j\}})$ to denote a labelling of variables in clique c in which variable x_i and x_j have been assigned labels l_a and l_b respectively. The cost of any configuration $(\{x_i, x_j\} \cup \mathbf{x}_{c\setminus\{i,j\}})$ of the clique can be written as

$$\begin{aligned}\psi_c(\{x_i,x_j\}\cup \mathbf{x}_{c\setminus\{i,j\}}) &= f_c(\mathcal{Q}_c(\{x_i,x_j\}\cup \mathbf{x}_{c\setminus\{i,j\}})) \\ &= f_c(\phi_c(x_i,x_j) + \mathcal{Q}_{c\setminus\{i,j\}}(\mathbf{x}_{c\setminus\{i,j\}}) + \\ &\sum_{u\in c\setminus\{i,j\}} \phi_c(x_i,\mathbf{x}_u) + \sum_{u\in c\setminus\{i,j\}} \phi_c(x_j,\mathbf{x}_u)) \qquad (4.2.7)\end{aligned}$$

Let D represent $\mathcal{Q}_{c\setminus\{i,j\}}(\mathbf{x}_{c\setminus\{i,j\}})$, D_α represent $\sum_{u\in c\setminus\{i,j\}} \phi_c(\alpha,\mathbf{x}_u)$, and D_β represent $\sum_{u\in c\setminus\{i,j\}} \phi_c(\beta,\mathbf{x}_u)$. Using equation (4.2.7), the condition (4.2.6) becomes

$$\begin{aligned}& f_c(\phi_c(\alpha,\beta) + D_\alpha + D_\beta + D) + f_c(\phi_c(\beta,\alpha) + D_\beta + D_\alpha + D) \quad (4.2.8)\\ \geq\ & f_c(\phi_c(\alpha,\alpha) + 2D_\alpha + D) + f_c(\phi_c(\beta,\beta) + 2D_\beta + D).\end{aligned}$$

As $\phi_c(\beta,\alpha) = \phi_c(\alpha,\beta)$ from constraint (4.2.4) this condition transforms to:

$$\begin{aligned}2f_c(\phi_c(\alpha,\beta) + D_\alpha + D_\beta + D) \geq \qquad (4.2.9)\\ f_c(\phi_c(\alpha,\alpha) + 2D_\alpha + D) + f_c(\phi_c(\beta,\beta) + 2D_\beta + D).\end{aligned}$$

To prove (4.2.9) we need lemma 1.

Lemma 1 *For a non decreasing concave function $f(x)$:*

$$2c \geq a + b \implies 2f(c) \geq f(a) + f(b). \qquad (4.2.10)$$

Proof in section 4.2.5.

Using the above lemma together with the fact that

$$2\phi_c(\alpha,\beta) \geq \phi_c(\alpha,\alpha) + \phi_c(\beta,\beta) \quad \forall \alpha,\beta \in \mathcal{L} \qquad (4.2.11)$$

(see constraint (4.2.5)), we see that the theorem holds true. ∎

The class of clique potentials described by Theorem 2 is a strict generalization of the class specified by the constraints of [11]. This can be verified by considering only pairwise cliques, choosing $f_c()$ as a linear increasing function, and constraining $\phi_c(a,a) = 0, \forall a \in \mathcal{L}$. Recall that all linear functions are concave.

4.2.2 Conditions for α-expansions

In this section we characterize the higher order clique potentials for which the optimal α-expansion move can be computed in polynomial time $\forall \alpha \in \mathcal{L}, \mathbf{x} \in \mathbf{L}$.

Theorem 3 *The optimal α-expansion move for any $\alpha \in \mathcal{L}$ can be computed in polynomial time for a potential function $\psi_c(\mathbf{x}_c)$ of the form (4.2.1) if $f_c(\cdot)$ is an increasing linear function, $\oplus = \sum$ and $\phi_c(\cdot,\cdot)$ is a metric.*

Proof
To prove that the optimal expansion move can be computed in polynomial time we need to show that all projections of any α-expansion move energy on two variables of the clique are submodular. From equation (4.1.7) this implies that $\forall i,j \in c$ the condition

$$\begin{aligned}\psi_c(\{\alpha,\alpha\} \cup \mathbf{x}_{c\setminus\{i,j\}}) &+ \psi_c(\{a,b\} \cup \mathbf{x}_{c\setminus\{i,j\}}) \leq \\ \psi_c(\{a,\alpha\} \cup \mathbf{x}_{c\setminus\{i,j\}}) &+ \psi_c(\{\alpha,b\} \cup \mathbf{x}_{c\setminus\{i,j\}}) \end{aligned} \tag{4.2.12}$$

is satisfied. Here a and b are the current labels of the variables X_i and X_j respectively.

Let D represent $\mathcal{Q}_{c\setminus\{i,j\}}(\mathbf{x}_{c\setminus\{i,j\}})$, and D_l represent $\sum_{u \in c\setminus\{i,j\}} \phi_c(l, \mathbf{x}_u)$ for any label l. Then, using equation (4.2.7) the constraint (4.2.12) becomes

$$\begin{aligned} & f_c(\phi_c(\alpha,b) + D_\alpha + D_b + D) \qquad (4.2.13)\\ + \;& f_c(\phi_c(a,\alpha) + D_a + D_\alpha + D) \\ \geq \;& f_c(\phi_c(\alpha,\alpha) + 2D_\alpha + D) \\ + \;& f_c(\phi_c(a,b) + D_a + D_b + D). \end{aligned}$$

Let $R_1 = \phi_c(\alpha,b) + D_\alpha + D_b + D$, $R_2 = \phi_c(a,\alpha) + D_a + D_\alpha + D$, $R_3 = \phi_c(\alpha,\alpha) + 2D_\alpha + D$, and $R_4 = f_c(\phi_c(a,b) + D_a + D_b + D)$. Since $\phi_c(\cdot,\cdot)$ is a metric, we observe that

$$\phi_c(\alpha,b) + \phi_c(a,\alpha) \geq \phi_c(\alpha,\alpha) + \phi_c(a,b) \tag{4.2.14}$$

$$\Rightarrow R_1 + R_2 \geq R_3 + R_4. \tag{4.2.15}$$

Thus, we require a function f such that

$$R_1 + R_2 \geq R_3 + R_4 \implies f(R_1) + f(R_2) \geq f(R_3) + f(R_4). \tag{4.2.16}$$

The following lemma provides us the form of this function.

Lemma 2 *For a function $f : \mathbb{R} \to \mathbb{R}$,*

$$y_1 + y_2 \geq y_3 + y_4 \implies f(y_1) + f(y_2) \geq f(y_3) + f(y_4) \tag{4.2.17}$$

for all $y_1, y_2, y_3, y_4 \in \mathbb{R}$ if and only if f is linear. Proof in section 4.2.5.

Since $f(\cdot)$ is linear, this proves the theorem. ■

It should be noted that the class of higher order potentials specified by the above theorem is a small subset of the family of functions which can be used under $\alpha\beta$-swap (characterized in Theorem 2). In fact it is the same class of energy functions which was specified in [11], i.e. $\mathcal{P}^2$. This can be verified by observing that the potentials specified by Theorem 3 can be represented as a sum of metric pairwise functions. This raises the question whether we can define a class of higher order clique potentials which cannot be decomposed into a set of $\mathcal{P}^2$ potentials and can still be solved using α-expansion. To answer this we need to define the $\mathcal{P}^n$ Potts model.

4.2.2.1 $\mathcal{P}^n$ Potts Model

We now introduce the $\mathcal{P}^n$ Potts model family of higher order clique potentials. This family is a strict generalization of the Generalized Potts model [11] and can be used for modelling many problems in Computer Vision.

We define the $\mathcal{P}^n$ Potts model potential for cliques of size n as

$$\psi_c(\mathbf{x}_c) = \begin{cases} \gamma_k & \text{if} \quad x_i = l_k, \forall i \in c, \\ \gamma_{\max} & \text{otherwise.} \end{cases} \tag{4.2.18}$$

where $\gamma_{\max} > \gamma_k$, $\forall l_k \in \mathcal{L}$. For a two variable clique, this reduces to the $\mathcal{P}^2$ Potts model potential defined as:

$$\psi_{ij}(a,b) = \begin{cases} \gamma_k & \text{if} \quad a = b = l_k, \\ \gamma_{\max} & \text{otherwise.} \end{cases} \tag{4.2.19}$$

Further, if we use $\gamma_k = 0$, for all l_k, this function becomes an example of a *metric*.

4.2.2.2 Going Beyond $\mathcal{P}^2$

We now show how the class of potential functions specified in Theorem 3 can be extended by using: $\oplus =$'max' instead of $\oplus = \sum$ which has been used up till now. To this end we define $\mathcal{Q}_c(\mathbf{x}_c)$ as

$$\mathcal{Q}_c(\mathbf{x}_c) = \max_{i,j \in c} \phi_c(x_i, x_j). \tag{4.2.20}$$

Theorem 4 *The optimal α-expansion move for any $\alpha \in \mathcal{L}$ can be computed in polynomial time for a potential function $\psi_c(\mathbf{x}_c)$ of the form (4.2.1) if $f_c(\cdot)$ is a increasing linear function, $\oplus =$ 'max' and $\phi_c(\cdot,\cdot)$ defines a $\mathcal{P}^2$ Potts Model (4.2.19).*

Proof
The cost of any configuration $\{x_i, x_j\} \cup \mathbf{x}_{c\setminus\{i,j\}}$ of the clique under $\oplus =$'max' can be written as

$$\begin{aligned}
\psi_c(\{x_i, x_j\} \cup \mathbf{x}_{c\setminus\{i,j\}}) &= f_c(\mathcal{Q}_c(\{x_i, x_j\} \cup \mathbf{x}_{c\setminus\{i,j\}})) \qquad (4.2.21)\\
&= f_c(\max(\phi_c(x_i, x_j), \mathcal{Q}_{c\setminus\{i,j\}}(\mathbf{x}_{c\setminus\{i,j\}}),\\
&\quad \max_{u\in c\setminus\{i,j\}} \phi_c(x_i, \mathbf{x}_u), \max_{u\in c\setminus\{i,j\}} \phi_c(x_j, \mathbf{x}_u))) \qquad (4.2.22)
\end{aligned}$$

Substituting this value of ψ_c in constraint (4.2.12) and again using D to represent $\mathcal{Q}_{c\setminus\{i,j\}}(\mathbf{x}_{c\setminus\{i,j\}})$ and D_l represent $\sum_{u\in c\setminus\{i,j\}} \phi_c(l, \mathbf{x}_u)$ for any label l, we get:

$$\begin{aligned}
& f_c(\max(\phi_c(\alpha, b), D_\alpha, D_b, D))\\
+ \;& f_c(\max(\phi_c(a, \alpha), D_a, D_\alpha, D))\\
\geq \;& f_c(\max(\phi_c(\alpha, \alpha), D_\alpha, D_\alpha, D))\\
+ \;& f_c(\max(\phi_c(a, b), D_a, D_b, D)). \qquad (4.2.23)
\end{aligned}$$

As f_c is a linear function, from lemma 2 we see that the above condition is true if:

$$\begin{aligned}
&\max(\phi_c(\alpha, b), D_\alpha, D_b, D) + \max(\phi_c(a, \alpha), D_a, D_\alpha, D) \geq\\
&\max(\phi_c(\alpha, \alpha), D_\alpha, D_\alpha, D) + \max(\phi_c(a, b), D_a, D_b, D).
\end{aligned}$$

We only consider the case $a \neq \alpha$ and $b \neq \alpha$. For all other cases it can be easily seen that the above inequality is satisfied by a equality. As ϕ_c is a $\mathcal{P}^2$ Potts model potential, the LHS of the above inequality is always equal to $2\gamma_{\max}$. As the maximum value of the RHS is $2\gamma_{\max}$ the above inequality is always true. This proves the Theorem. ∎

The class of higher order potentials specified by the above Theorem is the same as the family of clique potentials defined by the $\mathcal{P}^n$ Potts model (4.2.18) for a clique c of size n. This proves that the optimal α-expansion move for the $\mathcal{P}^n$ Potts model can be computed in polynomial time. In the next section we will show how the optimal α-expansion and $\alpha\beta$-swap moves for this subset of potential functions can be found by solving an st-mincut problem.

4.2.3 Graph Cuts for the $\mathcal{P}^n$ Potts Model

We now consider the minimization of energy functions whose clique potentials take the form of a $\mathcal{P}^n$ Potts model (see equation (4.2.18)). Specifically, we show that the optimal $\alpha\beta$-swap and α-expansion moves for such potential functions can be computed by solving an st-mincut problem. The graph in which the st-mincut

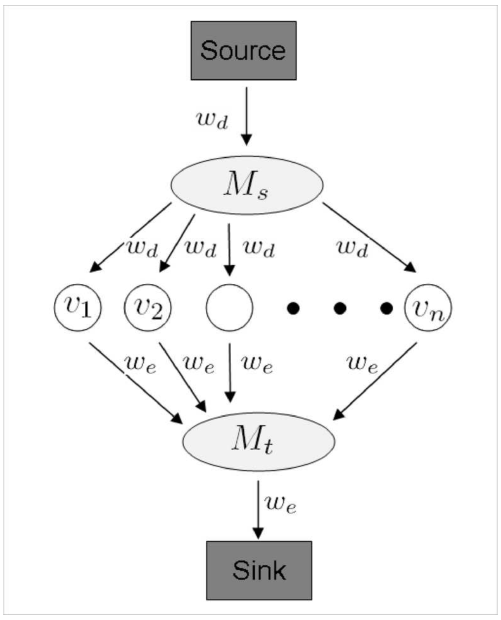


Figure 4.1: *Graph construction for computing the optimal moves for the $\mathcal{P}^n$ Potts model.*

needs to be computed is shown for only a single clique potential. However, the additivity theorem [54] allows us to construct the graph for an arbitrary number of potentials by simply merging those corresponding to individual cliques.

4.2.3.1 $\alpha\beta$-swap

Given a clique c, our aim is to find the optimal $\alpha\beta$-swap move (denoted by $\mathbf{t}_c^*$). Since the clique potential $\psi_c(\mathbf{x}_c)$ forms a $\mathcal{P}^n$ Potts model, we do not need to consider the move from a configuration in which any variable in the clique is assigned a label other than α or β. In this scenario the clique potential only adds a constant to the $\alpha\beta$-swap move energy and thus can be ignored without changing the optimal move. For all other configurations, the potential function after an $\alpha\beta$-swap move $\mathbf{t}_c = \{t_i, i \in c\}$ (where $t_i \in \{0, 1\}$) is given by

$$\psi_c(T_{\alpha\beta}(\mathbf{x}_c, \mathbf{t}_c)) = \begin{cases} \gamma_\alpha & \text{if} \quad t_i = 0, \forall i \in c, \\ \gamma_\beta & \text{if} \quad t_i = 1, \forall i \in c, \\ \gamma_{\max} & \text{otherwise.} \end{cases} \tag{4.2.24}$$

Further, we can add a constant κ to all possible values of the clique potential without changing the optimal move $\mathbf{t}_c^*$. We choose $\kappa = \gamma_{\max} - \gamma_\alpha - \gamma_\beta$. Note that since $\gamma_{\max} \geq \gamma_\alpha$ and $\gamma_{\max} \geq \gamma_\beta$, the following hold true:

$$\gamma_\alpha + \kappa \geq 0, \quad \gamma_\beta + \kappa \geq 0, \tag{4.2.25}$$

$$\gamma_\alpha + \kappa + \gamma_\beta + \kappa = \gamma_{\max} + \kappa. \tag{4.2.26}$$

Without loss of generality, we assume $\mathbf{t}_c = \{t_1, t_2, \ldots, t_n\}$. Fig. 4.1 shows the graph construction corresponding to the above values of the clique potential. Here, the node v_i corresponds to move variable t_i. In other words, after the computation of the st-mincut if v_i is connected to the source (i.e. it belongs to the source set) then $t_i = 0$ and if v_i is connected to the sink (i.e. it belongs to the sink set) then $t_i = 1$. In addition, there are two extra nodes denoted by M_s and M_t respectively. The weights of the graph are given by $w_d = \gamma_\beta + \kappa$ and $w_e = \gamma_\alpha + \kappa$. Note that all the weights are positive (see equations (4.2.25)). In order to show that this graph corresponds to the clique potential in equation (4.2.24) (plus the constant κ) we consider three cases:

- $t_i = 0, \forall i \in c$: In this case, the st-mincut corresponds to the edge connecting M_t with the sink which has a cost $w_e = \gamma_\alpha + \kappa$. Recall that the cost of an st-mincut is the sum of weights of the edges included in the st-mincut which go from the source set to the sink set.
- $t_i = 1, \forall i \in c$: In this case, the st-mincut corresponds to the edge connecting the source with M_s which has a cost $w_d = \gamma_\beta + \kappa$.
- All other cases: The st-mincut is given by the edges connecting M_t with the sink and the source with M_s. The cost of the cut is $w_d + w_e = \gamma_\alpha + \kappa + \gamma_\beta + \kappa = \gamma_{\max} + \kappa$ (from equation (4.2.26)).

Thus, we can find the optimal $\alpha\beta$-swap move for minimizing energy functions whose clique potentials form an $\mathcal{P}^n$ Potts model using an st-mincut operation.

4.2.3.2 α-expansion

Given a clique $\mathbf{x}_c$, our aim is to find the optimal α-expansion move $\mathbf{t}_c^*$. Again, since the clique potential $\psi_c(\mathbf{x}_c)$ forms a $\mathcal{P}^n$ Potts model, its value after an α-expansion move $\mathbf{t}_c$ is given by

$$\psi_c(T_\alpha(\mathbf{x}_c, \mathbf{t}_c)) = \begin{cases} \gamma & \text{if} \quad t_i = 0, \forall i \in c, \\ \gamma_\alpha & \text{if} \quad t_i = 1, \forall i \in c, \\ \gamma_{\max} & \text{otherwise}, \end{cases} \tag{4.2.27}$$

where $\gamma = \gamma_\beta$ if $x_i = \beta$ for all $i \in c$ and $\gamma = \gamma_{\max}$ otherwise. The above clique potential is similar to the one defined for the $\alpha\beta$-swap move in equation (4.2.24). Therefore, it can be represented using a graph by adding a constant $\kappa = \gamma_{\max} - \gamma_\alpha - \gamma$. This proves that the optimal α-expansion move can be computed by solving an st-mincut operation.

4.2.4 Planarity Preserving Clique Potentials

In this chapter we characterized a class of higher order potentials for which the optimal expansion and swap moves can be computed in polynomial time. However, there exist a number of useful higher order potential functions for which it is NP-hard to compute the optimal expansion and swap moves. In this section, we discuss one such potential which encourages planarity in disparity estimation.

Most approaches for disparity estimation assume a Potts model prior on the disparities of neighbouring pixels. This prior favours regions of constant disparity and penalizes discontinuities in the disparity map. This has the severe side-effect of assigning a high cost to planar regions in the scene which are not orthogonal to the camera-axis.

The above problem can be rectified by using a higher order clique potential which is *planarity preserving.* We define this potential as follows. Consider a higher order clique consisting of three variables X_1,X_2, and X_3 which can take a disparity label from the set $\mathcal{L}$ =$\{1, 2, \ldots, k\}$. As before, we will use x_i to denote a labelling of variable X_i. We say that the clique potential function $\psi_c()$ is planarity preserving if it is defined as:

$$\psi_c(x_1, x_2, x_3) = \begin{cases} \gamma_1 & \text{if } x_2 - x_1 = x_3 - x_2 = \delta, \forall\delta \\ \gamma_2 & \text{otherwise.} \end{cases} \tag{4.2.28}$$

where $\gamma_1 < \gamma_2$. This potential function favours labellings of random variables which are planar. For example, for the 3 variables (X_1, X_2, X_3) the cost of the planar configurations $(1, 2, 3)$ or $(1, 1, 1)$ is γ_1 which is less than the cost of the non-planar configuration $(1, 2, 1)$ which is γ_2.

We will now show the optimal α-expansion moves for this family of clique potentials cannot be always computed in polynomial time.

Theorem 5 *The optimal expansion move for the clique potential ψ_c of the form (4.2.28) cannot be always computed in polynomial time for all α and configurations* $\mathbf{x}$.

Proof
We prove the theorem by contradiction. We need to show that the move energy of a particular α-expansion move is non-submodular.

Consider the configuration $\{x_1, x_2, x_3\} = \{1, 2, 1\}$ on which we want to perform a α-expansion move with $\alpha = 3$. The move energy E_m is a function of the three move variables t_1,t_2, and t_3 and is defined as:

$$E_m(t_1,t_2,t_3) = E_m^{123} = \begin{array}{|c|c|}\hline E_m(0,0,0) & E_m(0,0,1) \\ \hline E_m(0,1,0) & E_m(0,1,1) \\ \hline E_m(1,0,0) & E_m(1,0,1) \\ \hline E_m(1,1,1) & E_m(1,1,1) \\ \hline \end{array}$$

which can be written in terms of ψ_c as:

$$E_m^{123} = \begin{array}{|c|c|}\hline \psi_c(T_\alpha(\{0,0,0\},\mathbf{x}) & \psi_c(T_\alpha(\{0,0,1\},\mathbf{x}) \\ \hline \psi_c(T_\alpha(\{0,1,0\},\mathbf{x}) & \psi_c(T_\alpha(\{0,1,1\},\mathbf{x}) \\ \hline \psi_c(T_\alpha(\{1,0,0\},\mathbf{x}) & \psi_c(T_\alpha(\{1,0,1\},\mathbf{x}) \\ \hline \psi_c(T_\alpha(\{1,1,0\},\mathbf{x}) & \psi_c(T_\alpha(\{1,1,1\},\mathbf{x}) \\ \hline \end{array}$$

For $\alpha = 3$ and $\mathbf{x} = \{1,2,1\}$ this becomes:

$$\begin{array}{|c|c|}\hline \psi_c(1,2,1) & \psi_c(1,2,3) \\ \hline \psi_c(1,3,1) & \psi_c(1,3,3) \\ \hline \psi_c(3,2,1) & \psi_c(3,2,3) \\ \hline \psi_c(3,3,1) & \psi_c(3,3,3) \\ \hline \end{array} = \begin{array}{|c|c|}\hline \gamma_2 & \gamma_1 \\ \hline \gamma_2 & \gamma_2 \\ \hline \gamma_1 & \gamma_2 \\ \hline \gamma_2 & \gamma_1 \\ \hline \end{array}$$

From the definition of submodularity all projections of the move energy E_m^{123} need to be submodular. Consider the submodularity constraints of the projection $E_m(0,t_2,t_3)$:

$$E_m(0,0,0) + E_m(0,1,1) \leq E_m(0,0,1) + E_m(0,1,0) \tag{4.2.29}$$

or $\gamma_2 + \gamma_2 \leq \gamma_1 + \gamma_2$. This constraint is not satisfied as $\gamma_1 < \gamma_2$ in the definition of the planarity preserving potential. ∎

A similar result can also be proved for $\alpha\beta$-swap moves considering the configuration $\mathbf{x} = \{1,2,1\}$ and the $1,3$-swap move.

4.2.5 Proofs of Lemmas

Lemma 1 *For a non decreasing concave function $f(\cdot)$*

$$2c \geq a + b \implies 2f(c) \geq f(a) + f(b). \tag{4.2.30}$$

Proof
Given: $c \geq \frac{a+b}{2}$.

$$\implies f(c) \geq f(\frac{a+b}{2}) \quad \text{(f is non decreasing)} \tag{4.2.31}$$

$$\implies f(c) \geq \frac{f(a)+f(b)}{2} \quad \text{(f is concave)} \tag{4.2.32}$$

$$\implies 2f(c) \geq f(a) + f(b) \tag{4.2.33}$$

∎

Lemma 2 *For a function $f : \mathbb{R} \to \mathbb{R}$,*

$$y_1 + y_2 \geq y_3 + y_4 \implies f(y_1) + f(y_2) \geq f(y_3) + f(y_4) \tag{4.2.34}$$

for all $y_1, y_2, y_3, y_4 \in \mathbb{R}$ if and only if f is linear.

Proof
We only prove the 'only if' part. The proof for the forward implication ('if') is trivial.

$$x + \epsilon \geq (x - \epsilon) + 2\epsilon \tag{4.2.35}$$

$$f(x) + f(\epsilon) \geq f(x - \epsilon) + f(2\epsilon) \tag{4.2.36}$$

$$f(x) - f(x - \epsilon) \geq f(2\epsilon) - f(\epsilon) \tag{4.2.37}$$

Similarly, starting from $x + \epsilon \leq (x - \epsilon) + 2\epsilon$, we get

$$f(x) - f(x - \epsilon) \leq f(2\epsilon) - f(\epsilon). \tag{4.2.38}$$

From equations (4.2.37) and (4.2.38) we get:

$$f(x) - f(x - \epsilon) = f(2\epsilon) - f(\epsilon). \tag{4.2.39}$$

Taking limits with $\epsilon \to 0$ we get the condition that the derivative (slope) of the function is constant. Hence, $f(\cdot)$ is linear. ▮

Chapter 5

Graph Cuts for Minimizing Higher Order Functions

In the previous chapter we reviewed the expansion and swap move making algorithms for approximate energy minimization. We then characterized a class of higher order clique potentials for which the optimal expansion and swap moves can be computed by minimizing a submodular function. However, minimizing a general submodular function is quite computationally expensive and thus it is infeasible to apply this procedure for minimizing large functions encountered in computer vision.

In this chapter we introduce a novel family of higher order potentials which we call the Robust P^n model. This family contains the P^n Potts model (which was introduced in the previous chapter) as well as its *robust* variants, and can be used for modelling many computer vision problems. We will show that the optimal *swap* and *expansion* moves for energy functions composed of such potentials can be computed using graph cuts. The complexity of our algorithm for computing the optimal move increases linearly with the size of the clique. This enables us to handle potential functions defined over very large cliques.

5.1. Robust Higher Order Potentials

The Robust P^n model potentials take the form:

$$\psi_c(\mathbf{x}_c) = \min\{\min_{k \in \mathcal{L}}((|c| - n_k(\mathbf{x}_c))\theta_k + \gamma_k), \gamma_{\max}\} \tag{5.1.1}$$

where $|c|$ is the number of variables in clique c, $n_k(\mathbf{x}_c)$ denotes the number of variables in clique c which take the label k in labelling $\mathbf{x}_c$, and $\gamma_k, \theta_k, \gamma_{\max}$ are potential function parameters which satisfy the constraints:

$$\theta_k = \frac{\gamma_{\max} - \gamma_k}{Q} \text{ and } \gamma_k \leq \gamma_{\max}, \forall k \in \mathcal{L}. \tag{5.1.2}$$

Q is called the truncation parameter of the potential and satisfies the constraint $2Q < |c|$. Recall that the P^n Potts model is defined as:

$$\psi_c(\mathbf{x}_c) = \begin{cases} \gamma_k & \text{if} \quad x_i = l_k, \forall i \in c, \\ \gamma_{\max} & \text{otherwise.} \end{cases} \tag{5.1.3}$$

where $\gamma_{\max} \geq \gamma_k$, $\forall l_k \in \mathcal{L}$. It can be seen that the Robust P^n model (5.1.1) becomes a P^n Potts model (5.1.3) when the truncation parameter is set to 1 i.e. $Q = 1$.

Example 4 *Consider the set of variables* $\mathbf{X} = \{X_1, X_2, \ldots, X_5\}$ *where each* $X_i, i \in \{1, 2, \ldots, 5\}$ *takes a value for the label set* $\mathcal{L} = \{a, b, c\}$. *The* P^n *Potts model assigns the cost* $\gamma_{\max}$ *to all labellings of the random variables except those* *where all variables X_i take the same label. Thus, the configuration $\mathbf{x} = \{a, a, b, a, c\}$ will be assigned cost $\gamma_{\max}$ even though there are only 2 variables (specifically, X_3 and X_5) which are assigned labels (b and c) different from the dominant label a. In contrast, the Robust P^n model with truncation 3, i.e. $Q = 3$, assigns the cost: $\gamma_a + \frac{(\gamma_{\max}-\gamma_a)}{3} \times 2$ to the same configuration.*

5.2. Computing Moves for Higher Order Potentials

We will now explain how the optimal swap and expansion moves for energy functions containing potential functions belonging to the Robust P^n model family (5.1.1) can be computed using graph cuts. In what follows we show that any swap or expansion move energy for higher order potentials of the form (5.1.1) can be converted to a sub-modular pairwise function. These pairwise functions corresponding to all potentials constituting the energy can be combined together to get a composite move energy. This move energy function is a second order sub-modular function and can be minimized exactly by solving an st-mincut problem (as shown in chapter 1) to get the optimal swap and expansion move.

5.2.1 Swap Moves

Recall from equation (4.1.8) that the transformation function for a swap move is defined as

$$T_{\alpha\beta}(x_i, t_i) = \begin{cases} \alpha & \text{if} \quad x_i = \alpha \text{ or } \beta \text{ and } t_i = 0, \\ \beta & \text{if} \quad x_i = \alpha \text{ or } \beta \text{ and } t_i = 1. \end{cases} \tag{5.2.1}$$

As expressed in the transformation function, only variables which are currently assigned labels α or β can take part in a $\alpha\beta$-swap move. We call these variables *active* and denote the vector of their indices by c_a. $\mathbf{t}_{c_a}$ will be used to denote the corresponding vector of move variables. Similarly, variables in the clique which do not take part in the swap move are called *passive*, and the set of their indices is denoted by c_p. The functions $n_k^m(\mathbf{t}_{c_a}), k \in \{0, 1\}$ count the number of move variables assigned the label k in the move $\mathbf{t}_{c_{\alpha\beta}}$, or formally:

$$n_1^m(\mathbf{t}_{c_a}) = |c_a| - n_0^m(\mathbf{t}_{c_a}) = \sum_{i \in c_a} t_i. \tag{5.2.2}$$

The move energy of a $\alpha\beta$-swap move from the current labelling $\mathbf{x}_c^p$ is equal to the energy of the new labelling $\mathbf{x}_c^n$ induced by the move and is given as

$$\psi_c^m(\mathbf{t}_{c_a}) = \psi_c(\mathbf{x}_c^n). \tag{5.2.3}$$

The new labelling $\mathbf{x}_c^n$ is obtained by combining the old labelling of the passive variable $\mathbf{X}_{c_p}$ with the new labelling of the active variables $\mathbf{X}_{c_a}$ as:

$$\mathbf{x}_c^n = \mathbf{x}_{c_p}^p \cup T_{\alpha\beta}(\mathbf{x}_{c_a}^p, \mathbf{t}_{c_a}). \tag{5.2.4}$$

On substituting the value of $\mathbf{x}_c^n$ from (5.2.4) in (5.2.3), and using the definition of the Robust P^n model (5.1.1) we get:

$$\begin{aligned}\psi_c^m(\mathbf{t}_{c_a}) &= \psi_c(\mathbf{x}_{c_p}^p \cup T_{\alpha\beta}(\mathbf{x}_{c_a}^p, \mathbf{t}_{c_a})). & (5.2.5)\\ &= \min\{\min_{k\in\mathcal{L}}((|c| - n_k(\mathbf{x}_{c_p}^p \cup T_{\alpha\beta}(\mathbf{x}_{c_a}^p, \mathbf{t}_{c_a})))\theta_k + \gamma_k), \gamma_{\max}\} & (5.2.6)\end{aligned}$$

It can be easily observed that if the number of active variables (those assigned label α or β) in the clique are less than $|c| - Q$, i.e. $|c_a| \leq |c| - Q$ the expression

$$(|c| - n_k(\mathbf{x}_{c_p}^p \cup T_{\alpha\beta}(\mathbf{x}_{c_a}^p, \mathbf{t}_{c_a})))\theta_k + \gamma_k \tag{5.2.7}$$

is greater than $\gamma_{\max}$ for both $k = \alpha$ and $k = \beta$. Thus, in this case the move energy $\psi_c^m(\mathbf{t}_{c_a})$ is independent of $\mathbf{t}_{c_a}$ and is equal to the constant:

$$\eta = \min\{\min_{k\in\mathcal{L}\setminus\{\alpha,\beta\}}(((|c| - n_k(\mathbf{x}_c^p))\theta_k + \gamma_k), \gamma_{\max}\} \tag{5.2.8}$$

which can be ignored while computing the swap moves. However, this is not true when $|c_a| > |c| - Q$. In this case the move energy can be written as:

$$\psi_c^m(\mathbf{x}_{c_a}^p, \mathbf{t}_{c_a}) = \min\{(|c_a| - n_0^m(\mathbf{t}_{c_a}))\theta_\alpha + \lambda_\alpha, (|c_a| - n_1^m(\mathbf{t}_{c_a}))\theta_\beta + \lambda_\beta, \lambda_{\max}\} \tag{5.2.9}$$

where $\lambda_\alpha = \gamma_\alpha + Q_{\alpha\beta}\theta_\alpha$, $\lambda_\beta = \gamma_\beta + Q_{\alpha\beta}\theta_\beta$, $\lambda_{\max} = \gamma_{\max}$ and $Q_{\alpha\beta} = |c| - |c_a| = |c_p|$. The minimization in (5.2.9) can be removed by defining the move energy as:

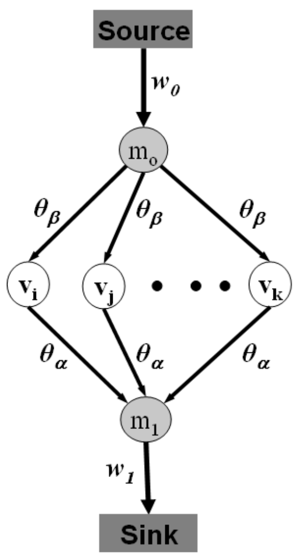
$$\psi_c^m(\mathbf{t}_{c_a}) = \begin{cases} \lambda_\alpha + (|c_a| - n_0^m(\mathbf{t}_{c_a}))\theta_\alpha & \text{if} \quad n_0^m(\mathbf{t}_{c_a}) > |c_a| - \hat{Q} \\ \lambda_\beta + n_0^m(\mathbf{t}_{c_a})\theta_\beta & \text{if} \quad n_0^m(\mathbf{t}_{c_a}) \leq \hat{Q} \\ \lambda_{\max} & \text{otherwise} \end{cases} \tag{5.2.10}$$

where $\hat{Q} = Q - Q_{\alpha\beta}$. Next we will show that this higher order move energy can be written as a second order submodular function with the addition of the auxiliary binary variables m_0 and m_1.

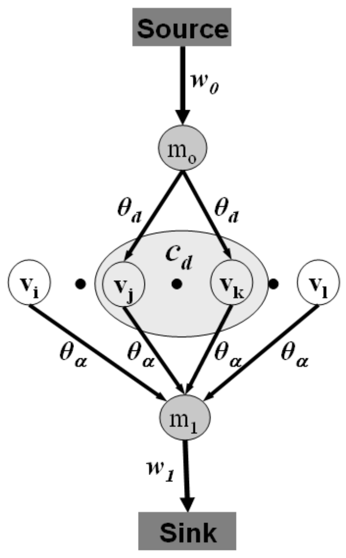
Theorem 6 *The higher order move energy (5.2.9) can be transformed into a pairwise energy function by introducing binary meta-variables m_0 and m_1 as:*

$$\psi_c^m(\mathbf{t}_c) = \min_{m_0,m_1} w_0(1-m_0) + \theta_\beta m_0 \sum_{i\in c_a}(1-t_i) + w_1 m_1 + \theta_\alpha(1-m_1)\sum_{i\in c_a} t_i - \delta. \tag{5.2.11}$$

where $w_0 = \lambda_\alpha + \delta$, $w_1 = \lambda_\beta + \delta$, and $\delta = \lambda_{\max} - \lambda_\alpha - \lambda_\beta$.



(a) Swap Move



(b) Expansion Move

Figure 5.1: *(a) Graph construction for minimizing the swap energy function (5.2.11). (b) Graph construction for minimizing the expansion move energy function (5.2.15). For every binary move variable t_i in the energy function there is a corresponding graph node v_i. The minimum cost source sink cut (st-mincut) divides the graph into two sets: the source set (S) and the sink set (T). $v_i \in (\mathcal{S})$ implies $t_i = 1$ while $v_i \in (\mathcal{T})$ implies $t_i = 0$.*

Proof
The proof for the theorem can be found in section 5.3.

The property $\gamma_{\max} \geq \gamma_k, \forall k \in \mathcal{L}$ of the clique potential (5.1.1) implies that all coefficients of the energy function (5.2.11) are non-negative. The function is thus *submodular* and can be minimized by solving a st-mincut problem [54]. The graph construction corresponding to the energy (5.2.11) is shown in figure (5.1a). The constant δ in (5.2.11) does not affect the minimization problem i.e. it does not change the move having the least energy and thus is ignored.

5.2.2 Expansion Moves

We now describe how optimal expansion moves can be computed for the higher order potentials (5.1.1).

Let c_k denote the set of variables in clique c that have been assigned label k in the current solution $\mathbf{x}_c^p$. We find the *dominant* label $d \in \mathcal{L}$ such that $|c_d| > |c| - Q$ where $d \neq \alpha$. The constraint $2Q < |c|$ of the Robust P^n model makes sure that there is at most one such label. If we find such a label in the current labelling, then the expansion move energy can be written as:

$$\begin{aligned}\psi_c^m(\mathbf{t}_c) &= \psi_c(T_\alpha(\mathbf{x}_c^p, \mathbf{t}_c)) & (5.2.12)\\ &= \min\{\lambda_\alpha + \theta_\alpha \sum_{i \in c} t_i, \lambda_d + \theta_d \sum_{i \in c_d}(1 - t_i), \lambda_{\max}\}. & (5.2.13)\end{aligned}$$

where $\lambda_\alpha = \gamma_\alpha$, $\lambda_d = \gamma_d + Q_d\theta_d$, $\lambda_{\max} = \gamma_{\max}$ and $Q_d = |c| - |c_d|$. Without the minimization operator the function definition (5.2.13) becomes:

$$\psi_c^m(\mathbf{t}_c, \mathbf{t}_{c_d}) = \begin{cases} \lambda_\alpha + (|c| - n_0^m(\mathbf{t}_c))\theta_\alpha & \text{if} \quad n_0^m(\mathbf{t}_c) > |c| - Q \\ \lambda_d + n_0^m(\mathbf{t}_{c_d})\theta_d & \text{if} \quad n_0^m(\mathbf{t}_{c_d}) \leq Q - Q_d \\ \lambda_{\max} & \text{otherwise} \end{cases} \quad (5.2.14)$$

Next we will show that this higher order move energy can be written as a second order submodular function with the addition of the auxiliary binary variables m_0 and m_1.

Theorem 7 *The expansion move energy (5.2.13) can be transformed into the pairwise function:*

$$\psi_c^m(\mathbf{t}_c) = \min_{m_0, m_1} w_0(1-m_0)+\theta_d m_0 \sum_{i \in c_d}(1-t_i)+w_1 m_1+\theta_\alpha(1-m_1)\sum_{i \in c} t_i - \delta. \quad (5.2.15)$$

where $w_0 = \lambda_\alpha + \delta$, $w_1 = \lambda_d + \delta$, and $\delta = \lambda_{\max} - \lambda_\alpha - \lambda_d$.

Proof
The proof for the theorem can be found in section 5.3.

The energy function (5.2.15) is submodular and can be minimized by finding the st-mincut in the graph shown in figure (5.1b). If a dominant label cannot be found then the move energy can be written as:

$$\psi_c^m(\mathbf{t}_c) = \min\{\lambda_\alpha + \theta_\alpha \sum_{i \in c} t_i, \lambda_{\max}\} \quad (5.2.16)$$

where $\lambda_\alpha = \gamma_\alpha$, and $\lambda_{\max} = \gamma_{\max}$. This can be transformed to the binary pairwise energy: $w_1 m_1 + \theta_\alpha(1 - m_1)\sum_{i \in c} t_i + \lambda_\alpha$, where $w_1 = \lambda_{\max} - \lambda_\alpha$. The proof for this transformation is similar to the one shown for Theorem 7.

5.3. Proofs of Theorems

Theorem 6 The higher order move energy (5.2.9) can be transformed into a pairwise energy function by introducing binary meta-variables m_0 and m_1 as:

$$\psi_c^m(\mathbf{t}_c) = \min_{m_0, m_1} w_0(1-m_0)+\theta_\beta m_0 \sum_{i \in c_a}(1-t_i)+w_1 m_1+\theta_\alpha(1-m_1)\sum_{i \in c_a} t_i - \delta. \quad (5.3.1)$$

where $w_0 = \lambda_\alpha + \delta$, $w_1 = \lambda_\beta + \delta$, and $\delta = \lambda_{\max} - \lambda_\alpha - \lambda_\beta$.
Proof
We decompose the function (5.3.1) as:

$$\psi_c^m(\mathbf{t}_c) = f^0(\mathbf{t}_{c_a}) + f^1(\mathbf{t}_{c_a}) - \delta \text{ where} \tag{5.3.2}$$

$$\begin{aligned}
f^0(\mathbf{t}_{c_a}) &= \min_{m_0} w_0(1-m_0) + n_0^m(\mathbf{t}_{c_a})\theta_\beta m_0 && (5.3.3)\\
&= \min_{m_0} (\lambda_\alpha + \delta)(1-m_0) + \theta_\beta m_0 n_0^m(\mathbf{t}_{c_a}) && (5.3.4)\\
&= \min_{m_0} (\lambda_{\max} - \lambda_\beta)(1-m_0) + \frac{\gamma_{\max} - \gamma_\beta}{Q} m_0 n_0^m(\mathbf{t}_{c_a}) && (5.3.5)
\end{aligned}$$

On substituting the value of $\lambda_{\max}$ and λ_β we get

$$\begin{aligned}
f^0(\mathbf{t}_{c_a}) &= \min_{m_0} (\gamma_{\max} - \gamma_\beta - Q_{\alpha\beta}\theta_\beta)(1-m_0) + \frac{\gamma_{\max} - \gamma_\beta}{Q} m_0 n_0^m(\mathbf{t}_{c_a}) && (5.3.6)\\
&= \min_{m_0} (\gamma_{\max} - \gamma_\beta)(1-m_0) + \frac{\gamma_{\max} - \gamma_\beta}{Q} m_0 (n_0^m(\mathbf{t}_{c_a}) + Q_{\alpha\beta}) - Q_{\alpha\beta}\theta_\beta\\
&= \begin{cases} \lambda_{\max} - \lambda_\beta & \text{if } \; n_0^m(\mathbf{t}_{c_a}) > Q - Q_{\alpha\beta} \\ n_0^m(\mathbf{t}_{c_a})\theta_\beta & \text{if } \; n_0^m(\mathbf{t}_{c_a}) \le Q - Q_{\alpha\beta} \end{cases} && (5.3.7)
\end{aligned}$$

Similarly,

$$\begin{aligned}
f^1(\mathbf{t}_{c_a}) &= \begin{cases} \lambda_{\max} - \lambda_\alpha & \text{if } n_1^m(\mathbf{t}_{c_a}) > Q - Q_{\alpha\beta} \\ (|c_a| - n_0^m(\mathbf{t}_{c_a}))\theta_\alpha & \text{if } n_1^m(\mathbf{t}_{c_a}) \le Q - Q_{\alpha\beta} \end{cases} && (5.3.8)\\
&= \begin{cases} \lambda_{\max} - \lambda_\alpha & \text{if } n_0^m(\mathbf{t}_{c_a}) \le |c_a| - (Q - Q_{\alpha\beta}) \\ (|c_a| - n_0^m(\mathbf{t}_{c_a}))\theta_\alpha & \text{if } n_0^m(\mathbf{t}_{c_a}) > |c_a| - (Q - Q_{\alpha\beta}). \end{cases} && (5.3.9)
\end{aligned}$$

Adding equations (5.3.7) and (5.3.9) and from the constraint $2Q < |c|$ we get $f^0(\mathbf{t}_{c_a}) + f^1(\mathbf{t}_{c_a}) =$

$$\begin{cases} \lambda_{\max} - \lambda_\beta + (|c_a| - n_0^m(\mathbf{t}_{c_a}))\theta_\alpha & \text{if} \quad n_0^m(\mathbf{t}_{c_a}) > |c_a| - \hat{Q} \\ n_0^m(\mathbf{t}_{c_a})\theta_\beta + \lambda_{\max} - \lambda_\alpha & \text{if} \quad n_0^m(\mathbf{t}_{c_a}) \le \hat{Q} \\ \lambda_{\max} - \lambda_\alpha + \lambda_{\max} - \lambda_\beta & \text{otherwise,} \end{cases} \tag{5.3.10}$$

where $\hat{Q} = Q - Q_{\alpha\beta}$. Substituting this in (5.3.2) and simplifying we get:

$$\psi_c^m(\mathbf{t}_{c_a}) = \begin{cases} \lambda_\alpha + (|c_a| - n_0^m(\mathbf{t}_{c_a}))\theta_\alpha & \text{if } n_0^m(\mathbf{t}_{c_a}) > |c_a| - \hat{Q} \\ \lambda_\beta + n_0^m(\mathbf{t}_{c_a})\theta_\beta & \text{if } n_0^m(\mathbf{t}_{c_a}) \le \hat{Q} \\ \lambda_{\max} & \text{otherwise.} \end{cases} \tag{5.3.11}$$

which is same as (5.2.10). ∎

Theorem 7 The expansion move energy (5.2.13) can be transformed into the pairwise function:

$$\psi_c^m(\mathbf{t}_c) = \min_{m_0,m_1} w_0(1-m_0)+\theta_d m_0 \sum_{i\in c_d}(1-t_i)+w_1 m_1+\theta_\alpha(1-m_1)\sum_{i\in c} t_i-\delta. \quad (5.3.12)$$

where $w_0 = \lambda_\alpha + \delta$, $w_1 = \lambda_d + \delta$, and $\delta = \lambda_{\max} - \lambda_\alpha - \lambda_d$.

Proof

We decompose the move energy (5.3.12) as:

$$\psi_c^m(\mathbf{t}_c) = f^0(\mathbf{t}_{c_d}) + f^1(\mathbf{t}_c) - \delta \text{ where} \quad (5.3.13)$$

$$\begin{aligned}
f^0(\mathbf{t}_{c_d}) &= \min_{m_0} w_0(1-m_0) + n_0^m(\mathbf{t}_{c_d})\theta_d m_0 && (5.3.14)\\
&= \min_{m_0}(\lambda_\alpha+\delta)(1-m_0) + \theta_d m_0 n_0^m(\mathbf{t}_{c_d}) && (5.3.15)\\
&= \min_{m_0}(\gamma_{\max}-\gamma_d-Q_d\theta_d)(1-m_0) + \frac{\gamma_{\max}-\gamma_d}{Q} m_0 n_0^m(\mathbf{t}_{c_d}) && (5.3.16)\\
&= \min_{m_0}(\gamma_{\max}-\gamma_d)(1-m_0) + \frac{\gamma_{\max}-\gamma_d}{Q} m_0(n_0^m(\mathbf{t}_{c_d})+Q_d) - Q_d\theta_d \\
&= \begin{cases} \lambda_{\max}-\lambda_d & \text{if } \; n_0^m(\mathbf{t}_{c_d}) > Q-Q_d \\ n_0^m(\mathbf{t}_{c_d})\theta_d & \text{if } \; n_0^m(\mathbf{t}_{c_d}) \le Q-Q_d, \end{cases} && (5.3.17)
\end{aligned}$$

$$\begin{aligned}
\text{and } f^1(\mathbf{t}_c) &= \min_{m_1} w_1 m_1 + n_1^m(\mathbf{t}_c)\theta_\alpha(1-m_1) && (5.3.18)\\
&= \min_{m_1}(\lambda_d+\delta)m_1 + \theta_\alpha(1-m_1)n_1^m(\mathbf{t}_c) && (5.3.19)\\
&= \min_{m_1}(\gamma_{\max}-\gamma_\alpha)m_1 + \frac{\gamma_{\max}-\gamma_\alpha}{Q}(1-m_1)n_1^m(\mathbf{t}_c) && (5.3.20)\\
&= \begin{cases} \lambda_{\max}-\lambda_\alpha & \text{if } \; n_1^m(\mathbf{t}_c) \ge Q \\ n_1^m(\mathbf{t}_c)\theta_\alpha & \text{if } \; n_1^m(\mathbf{t}_c) < Q. \end{cases} && (5.3.21)\\
&= \begin{cases} \lambda_{\max}-\lambda_\alpha & \text{if } \; n_0^m(\mathbf{t}_c) \le |c|-Q \\ n_1^m(\mathbf{t}_c)\theta_\alpha & \text{if } \; n_0^m(\mathbf{t}_c) > |c|-Q \end{cases} && (5.3.22)
\end{aligned}$$

Adding (5.3.17) and (5.3.22) and using the relations

$$\begin{aligned}
n_0^m(t_{c_d}) \le Q-Q_d &\implies n_0^m(\mathbf{t}_c) \le |c|-Q && (5.3.23)\\
n_0^m(\mathbf{t}_c) > |c|-Q &\implies n_0^m(t_{c_d}) > Q-Q_d && (5.3.24)
\end{aligned}$$

which are derived from the constraints $2Q < |c|$ and $|c_d| > |c|-Q$, we get:

$$f^0(\mathbf{t}_{c_a}) + f^1(\mathbf{t}_{c_a}) = \begin{cases} \lambda_{\max}-\lambda_d+(|c|-n_0^m(\mathbf{t}_c))\theta_\alpha & \text{if } n_0^m(\mathbf{t}_c) > |c|-Q \\ n_0^m(\mathbf{t}_{c_d})\theta_d+\lambda_{\max}-\lambda_\alpha & \text{if } n_0^m(\mathbf{t}_{c_d}) \le Q-Q_d \\ \lambda_{\max}-\lambda_\alpha+\lambda_{\max}-\lambda_d & \text{otherwise.} \end{cases} \quad (5.3.25)$$

Substituting in (5.3.13) and simplifying we get:

$$\psi_c^m(\mathbf{t}_c, \mathbf{t}_{c_d}) = \begin{cases} \lambda_\alpha + (|c| - n_0^m(\mathbf{t}_c))\theta_\alpha & \text{if } n_0^m(\mathbf{t}_c) > |c| - Q \\ \lambda_d + n_0^m(\mathbf{t}_{c_d})\theta_d & \text{if } n_0^m(\mathbf{t}_{c_d}) \leq Q - Q_d \\ \lambda_{\max} & \text{otherwise} \end{cases} \tag{5.3.26}$$

which is the same as (5.2.14). ∎

Chapter 6

Applications of Higher Order Potentials

In this chapter we show how higher order potential functions can be used for modelling computer vision problems. We use potentials which take the form of the Robust P^n model (5.1.1). This enables us to use the st-mincut based method explained in chapter 5 for minimizing the resulting energy functions.

In the first part of the chapter we propose a novel framework for labelling problems which is capable of merging regions from multiple image segmentations in a principled manner. Many recently proposed methods for image labelling problems such as object segmentation [34] and single view reconstruction [36] work by merging image segments (so called superpixels) generated using unsupervised image segmentation algorithms. They are inspired from the observation that pixels constituting these superpixels often have the same label; for instance, they may belong to the same object or may have the same surface orientation. The methods used for integrating or merging segments are heuristics which lack any optimality guarantees.

Our approach uses higher order conditional random fields (CRFs) to define potentials on sets of pixels. These novel higher order potentials enforce label consistency in image regions and can be seen as a strict generalization of the commonly used pairwise contrast sensitive smoothness potential. We test our method on the problem of multi-class object segmentation by augmenting the conventional CRF used for object segmentation with higher order potentials defined on image regions. Experiments on challenging data sets show that integration of higher order potentials quantitatively and qualitatively improves results leading to much better definition of object boundaries. We believe our method can be used to yield similar improvements for many other labelling problems.

Higher order potentials provide a probabilistic formulation for a wide variety of exemplar based applications in computer vision, such as 3D reconstruction [69] and object recognition [58]. In the second half of this chapter, we show how higher order potentials defined over texture patch exemplars can be used for the problem of texture segmentation. Our experiments show that the use of such potentials leads to significant improvements in the segmentation results.

6.1. Enforcing Label Consistency in Superpixels

In recent years an increasingly popular way to solve labelling problems like object segmentation, stereo and single view reconstruction is to formulate them using image segments (or superpixels) generated using unsupervised segmentation algorithms such as mean shift or normalized cut [34, 36, 75]. These methods are inspired from the observation that pixels constituting some of these segments often have the same label; for instance, they may belong to the same object or may have the same surface orientation. This approach has the benefit that higher order features based on all the pixels constituting the segment can be computed and used for classification. Further, as inference now only needs to be performed over small number of superpixels rather than all the pixels in the image, this approach substantially decreases the running time of the algorithm. The final advantage of this method is that the problem of scene understanding is decoupled from the image resolution given by the hardware; it is conducted using more natural primitives that are independent of resolution.

Methods based on grouping segments make the assumption that segments are consistent with object boundaries in the image [34], i.e. segments do not contain multiple objects. As observed by [37] and [86] this is not always the case and segments obtained using unsupervised segmentation methods are often wrong. To overcome these problems [37] and [86] use multiple segmentations of the image (instead of only one) in the hope that although most segmentations are bad, some are correct and thus would prove useful for their task. They merge the multiple superpixels using heuristic algorithms which lack any optimality guarantees and thus may produce bad results. *What is needed then are algorithms that can compute the solution of the labelling problem (using features based on superpixels) in a principled manner.* In this chapter we address this problem and show that it can be solved using potential functions defined on sets of pixels. Such potentials can be coupled with conventional unary and pairwise cues using higher order CRFs. We test the performance of this method on the problem of object segmentation and recognition. Our experiments show that the results of our approach are significantly better than the ones obtained using pairwise CRF models (see figure 6.1).

6.1.1 Object Segmentation and Recognition

Combined object segmentation and recognition is one of the most challenging and fundamental problems in computer vision. The last few years have seen the emergence of object segmentation algorithms which integrate *object specific* top-down information with *image based* low-level features [5,33,38,59,66]. These methods have produced excellent results on challenging data sets. However, they typically only deal with one object at a time in the image independently and do not provide a framework for understanding the whole image. Further, their models become prohibitively large as the number of classes increases. This prevents their application to scenarios where segmentation and recognition of many object classes is desired.

Shotton *et al.* [95] recently proposed a method (*Textonboost*) to overcome this

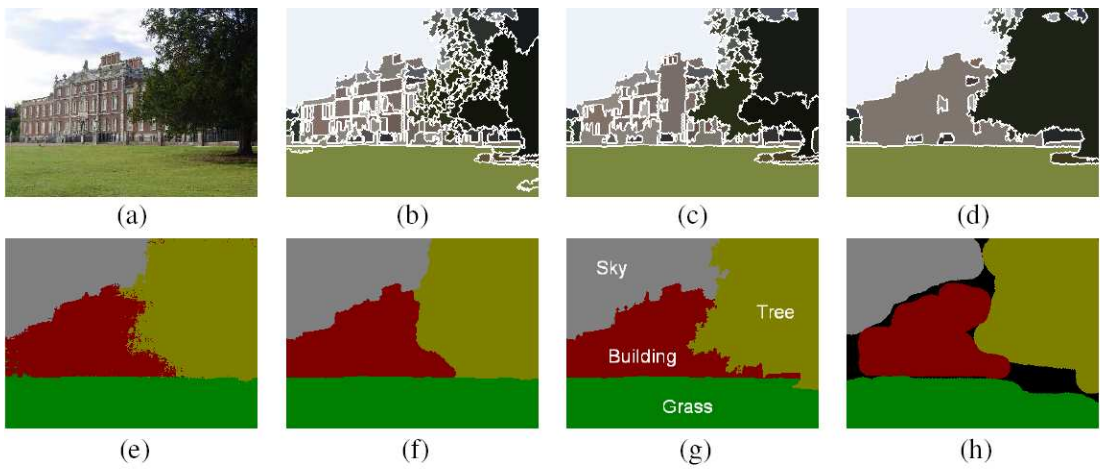


Figure 6.1: *Using higher order potentials for object segmentation. (a) An image from the MSRC-23 dataset. (b),(c) and (d) Unsupervised image segmentation results generated by using different parameters values in the mean-shift segmentation algorithm [17]. (e) The object segmentation obtained using the unary likelihood potentials from Textonboost [95]. (f) The result of performing inference in the pairwise CRF defined in section 6.1.2. (g) Our segmentation result obtained by augmenting the pairwise CRF with higher order potentials defined on the segments shown in (b),(c) and (d). (h) The rough hand labelled segmentations provided in the MSRC data set. It can be clearly seen that the use of higher order potentials results in a significant improvement in the segmentation result. For instance, the branches of the tree are much better segmented.*

problem. In contrast to using explicit models to encode object shape they used a boosted combination of *texton* features which jointly modeled shape and texture. They combine the result of textons with colour and location based likelihood terms in a condition random field (CRF). Although their method produced good segmentation and recognition results, the rough shape and texture model caused it to fail at object boundaries. The problem of extracting accurate boundaries of objects is considerably more challenging. In what follows we show that incorporation of higher order potentials defined on superpixels dramatically improves the object segmentation result. In particular, it leads to results with much better definition of object boundaries as shown in figure 6.1.

The higher order energy functions characterizing the higher order CRFs arising from our work take the form of a Robust P^n model (5.1.1) and thus can be minimized efficiently using the algorithm given in the previous chapter.

This section proposes a general framework for solving labelling problems which has the ability of utilizing higher order potentials defined on segments. We test this framework on the problem of object segmentation and recognition by integrating label consistency and shape based terms defined on segments with conventional unary and pairwise potentials. We show how inference in this framework can be efficiently performed by extending our recent work on minimizing energy function with higher order cliques [46]. To summarize, the novelties of our approach include:

1. A novel higher order region consistency potential which is a strict generalization of the commonly used pairwise contrast sensitive smoothness potential.

2. The application of higher order CRFs for object segmentation and recognition which integrate the above mentioned higher order potentials with conventional unary and pairwise potentials based on colour, location, texture, and smoothness.

We now give a brief outline of the sections to follow. In section 6.1.2 we show how pairwise CRFs can be used to model labelling problems like object segmentation. In section 6.1.3 we augment the pairwise CRF model by incorporating novel higher order potentials based on superpixel segmentations. The experimental results of our method are given in section 6.1.4. These include qualitative and quantitative results on well known and challenging data sets for object segmentation and recognition. The conclusions and directions for future work are listed in section 6.1.5.

6.1.2 Pairwise CRFs for Object Segmentation

The CRF models commonly used for object segmentation are characterized by energy functions defined on unary and pairwise cliques as:

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{i \in \mathcal{V}, j \in \mathcal{N}_i} \psi_{ij}(x_i, x_j). \tag{6.1.1}$$

Here $\mathcal{V}$ corresponds to the set of all image pixels, $\mathcal{N}$ is a neighbourhood defined on this set which is commonly chosen to be either a 4 or 8 neighbourhood. The labels constituting the label set $\mathcal{L}$ represent the different objects. The random variable x_i denotes the labelling of pixel i of the image. Every possible assignment of the random variables $\mathbf{x}$ (or configuration of the CRF) defines a segmentation.

The unary potential ψ_i of the CRF is defined as the negative log of the likelihood of an object label assigned to pixel i. It can be computed from the colour of the pixel and the appearance model for each object. However, colour alone is not a very distinguishing feature and fails to produce accurate segmentations.

This problem can be overcome by using sophisticated potential functions based on colour, texture, location, and shape priors as shown by [4, 12, 59, 83, 95]. The unary potential used by us can be written as:

$$\psi_i(x_i) = \theta_T \psi_T(x_i) + \theta_{col} \psi_{col}(x_i) + \theta_l \psi_l(x_i) \tag{6.1.2}$$

where θ_T, θ_{col}, and θ_l are parameters weighting the potentials obtained from TextonBoost(ψ_T) [95], colour(ψ_{col}) and location(ψ_l) respectively. The pairwise terms ψ_{ij} of the CRF take the form of a contrast sensitive Potts model:

$$\psi(x_i, x_j) = \begin{cases} 0 & \text{if } x_i = x_j, \\ g(i,j) & \text{otherwise,} \end{cases} \tag{6.1.3}$$

where the function $g(i,j)$ is an edge feature based on the difference in colors of neighbouring pixels [9]. It is typically defined as:

$$g(i,j) = \theta_p + \theta_v \exp(-\theta_\beta ||I_i - I_j||^2), \tag{6.1.4}$$

where I_i and I_j are the colour vectors of pixel i and j respectively. θ_p, θ_v, and θ_β are model parameters whose values are learned using training data. We refer the reader to [9, 83, 95] for more details.

6.1.2.1 Inferring the Most Probable Segmentation

The object segmentation problem can be solved by finding the least energy configuration of the CRF defined above. As the pairwise potentials of the energy function (6.1.1) are of the form of a Potts model it can be minimized approximately using the well known α-expansion algorithm [11]. The resulting segmentation can be seen in figure 6.1. We also tried other energy minimization algorithms such as sequential tree-reweighted message passing (TRW-S) [51, 112]. The α-expansion algorithm was preferred because it was faster and gave a solution with lower energy compared to TRW-S.

6.1.2.2 The Need for Higher Order CRFs

The use of smoothness potentials in the CRF model makes it favour smooth object boundaries. Although this improves results in most cases it also introduces an undesirable side effect. Smoothness potentials make the model incapable of extracting the fine contours of certain object classes such as trees and bushes. As seen in the results, segmentations obtained using pairwise CRFs tend to be oversmooth and quite often do not match the actual object contour. In the next section we show how these results can be significantly improved by using higher order terms derived from multiple segmentations obtained from an unsupervised image segmentation method.

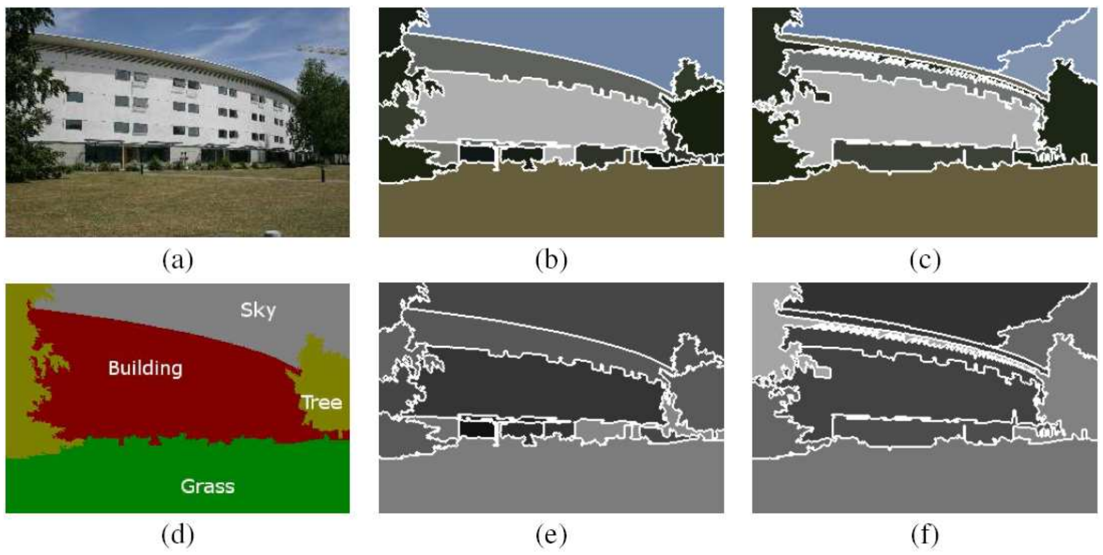


Figure 6.2: *Quality sensitive region consistency prior. (a) An image from the MSRC data set. (b) and (c) Two different segmentations of the image obtained using different parameter values for the mean-shift algorithm. (d) A hand labelled object segmentation of the image. (e) and (f) The value of the variance based quality function $G(c)$ (see equation 6.1.8) computed over the segments of the two segmentations. Segments with high quality values are darker. It can be clearly seen that segments which contain multiple object classes have been assigned low quality. For instance, the top segment of the left tree in segmentation (c) includes a part of the building and thus is brighter in the image (f) indicating low quality. Potentials defined on such segments will have a lower labelling inconsistency cost and will have less influence in the CRF.*

6.1.3 Incorporating Higher Order Potentials

We augment the pairwise CRF model explained above by incorporating higher order potentials defined on sets or regions of pixels. The Gibbs energy of this higher order CRF can now be written as:

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{i \in \mathcal{V}, j \in \mathcal{N}_i} \psi_{ij}(x_i, x_j) + \sum_{c \in \mathcal{S}} \psi_c(\mathbf{x}_c), \tag{6.1.5}$$

where $\mathcal{S}$ refers to the set of all regions or segments, and ψ_c are higher order potentials defined on them. We will now describe how these potentials are defined.

6.1.3.1 Region based consistency potential

Methods based on grouping regions for segmentation make the assumption that all pixels constituting a particular segment (or region) belong to the same object [34].

Instead of using this assumption as a hard constraint, we wish to embed it as a *soft constraint* in the CRF model. This potential which we refer to as the *region consistency potential* is similar to the smoothness prior present in pairwise CRFs [9]. It favours all pixels belonging to a segment taking the same label, and as will be shown later is particularly useful in obtaining object segmentations with fine boundaries. It takes the form of a $\mathcal{P}^n$ Potts model [46]:

$$\psi_c^p(\mathbf{x}_c) = \begin{cases} 0 & \text{if} \quad x_i = l_k, \forall i \in c, \\ \theta_p^h |c|^{\theta_\alpha} & \text{otherwise.} \end{cases} \tag{6.1.6}$$

where $|c|$ is the cardinality of the pixel set c which in our case is the number of pixels constituting superpixel c. The expression $\theta_p^h |c|^{\theta_\alpha}$ gives the label inconsistency cost, i.e. the cost added to the energy of a labelling in which different labels have been assigned to the pixels constituting the segment. The parameters θ_p^h and θ_α are learned from the training data by cross validation as described in section 6.1.4. The reader should note that this potential cannot be expressed in a pairwise CRF model.

6.1.3.2 Quality sensitive consistency potential

Not all segments obtained using unsupervised segmentation are equally good, for instance, some segments may contain multiple object classes. A region consistency potential defined over such a segment will encourage an incorrect labelling of the image. This is because the potential (6.1.6) does not take the quality or *goodness* of the segment. It assigns the same penalty for breaking 'good' segment as it assigns to 'bad' ones. This problem of the consistency potential can be overcome by defining a quality sensitive higher order potential (see figure 6.2). This new potential works by modulating the label inconsistency cost with a function of the quality of the segment (which is denoted by $G(c)$). Any method for estimating the segment quality can be used in our framework. A good example would be the method of [80] which uses inter and intra region similarity to measure the quality or goodness of a segment. Formally, the potential function is written as:

$$\psi_c^v(\mathbf{x}_c) = \begin{cases} 0 & \text{if } x_i = l_k, \forall i \in c, \\ |c|^{\theta_\alpha}(\theta_p^h + \theta_v^h G(c)) & \text{otherwise.} \end{cases} \tag{6.1.7}$$

For our experiments, we use the variance of the response of a unary feature evaluated on all constituent pixels of a segment to measure the quality of a segment, i.e.

$$G(c) = \exp\left(-\theta_\beta^h \frac{\| \sum_{i \in c}(f(i) - \mu)^2 \|}{|c|}\right), \tag{6.1.8}$$

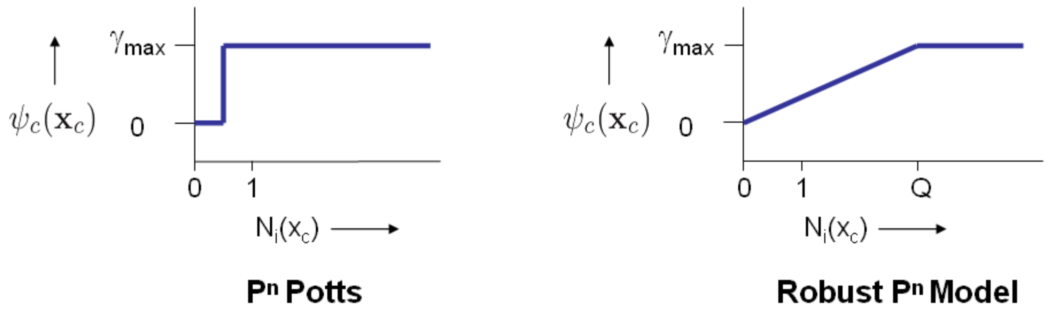


Figure 6.3: *Behaviour of the rigid P^n Potts potential and the Robust P^n model potential. The figure shows how the cost enforced by the two higher order potentials changes with the number of variables in the clique not taking the dominant label i.e. $N_i(\mathbf{x}_c) = min_k(|c| - n_k(\mathbf{x}_c))$.*

where $\mu = \frac{\sum_{i \in c} f(i)}{|c|}$ and $f()$ is a function evaluated on all constituent pixels of the superpixel c. If we restrict our attention to only pairwise cliques i.e. $|c| = 2$, the variance sensitive potential becomes

$$\psi_c^v(x_i, x_j) = \begin{cases} 0 & \text{if } x_i = x_j, \\ |c|^{\theta_\alpha}(\theta_p^h + \theta_v^h \exp(-\theta_\beta^h \|f(i) - f(j)\|^2)) & \text{otherwise.} \end{cases} \quad (6.1.9)$$

This is the same as the pairwise potential (6.1.3) commonly used in pairwise CRFs for different image labelling problems [9, 83]. Thus, the variance sensitive potential can be seen as a higher order generalization of the contrast preserving potential. The variance function response over two segmentations of an image is shown in figure 6.2.

6.1.3.3 Making the potentials robust

The P^n Potts model enforces label consistency very rigidly and thus might not be able to deal with inaccurate superpixels or resolve conflicts between overlapping regions of pixels. This phenomenon is illustrated in figure 6.4 wherein a part of the bird is merged with the 'sky' superpixel and results in an inaccurate segmentation. Intuitively, this problem can be resolved using the *Robust* higher order potentials defined as:

$$\psi_c^v(\mathbf{x}_c) == \begin{cases} N_i(\mathbf{x}_c)\frac{1}{Q}\gamma_{\max} & \text{if } N_i(\mathbf{x}_c) \leq Q, \\ \gamma_{\max} & \text{otherwise,} \end{cases} \quad (6.1.10)$$

where $N_i(\mathbf{x}_c)$ denotes the number of variables in the clique c not taking the dominant label, i.e. $N_i(\mathbf{x}_c) = \min_k(|c| - n_k(\mathbf{x}_c))$, $\gamma_{\max} = |c|^{\theta_\alpha}(\theta_p^h + \theta_v^h G(c))$, and Q is the truncation parameter which controls the rigidity of the higher order clique potential. This potential is of the form of the Robust P^n model (5.1.1) introduced in the previous chapter, where we showed how energy functions composed of such potentials can be minimized using move making algorithms such as α-expansion and $\alpha\beta$-swap.

Unlike the rigid P^n Potts model, this potential function gives rise to a cost that is a linear truncated function of the number of inconsistent variables (see figure 6.3). This enables the robust potential to allow some variables in the clique to take different labels. In the image shown in figure 6.4, the Robust P^n model potential allows some pixels of the 'sky' segment to take the label 'bird' thus producing a much better segmentation. Experimental results are shown for multiple values of the truncation parameter Q. More qualitative results can be seen in figure 6.7.

6.1.3.4 Generating multiple segmentations

We now explain how the set $\mathcal{S}$ of segments used for defining the higher order energy function (6.1.5) was generated. Our framework is quite flexible and can handle multiple overlapping or non-overlapping segments. The computer vision literature contains algorithms for sampling the likely segmentations of an image [106] or for generating multi-scale segmentations [93]. However, following in the footsteps of [86] we choose to generate multiple segmentations by varying the parameters of the mean shift segmentation algorithm [17]. This method belongs to the class of unsupervised segmentation algorithms which work by clustering pixels on the basis of low level image features [17, 24, 94]. They have been shown to give decent results which have proved to be useful for many applications [36, 37, 113].

The kernel used in the mean shift algorithm is defined as the product of spatial and range kernels. The spatial domain contains the (x, y) coordinates, while the range domain contains pixel colour information in LUV space. An assumption of Euclidian metric in both of them allows the use of a single bandwidth parameter for each domain, h_s for spatial and h_r for range.

Shown in figure 6.5 are segmentation results obtained using 2 different spatial $\{7, 18\}$ and 3 different range parameter values $\{6.5, 9.5, 15\}$. It can be seen that the results do not change dramatically on small images by modifying h_s. The only difference occurs on very noisy parts of the image like trees and bushes. By increasing the range parameter h_r we can get a range of segmentations which vary from over-segmented to under-segmented. We decided to use three segmentations with parameters $(h_s, h_r) = \{(7, 6.5), (7, 9.5), (7, 15)\}$.

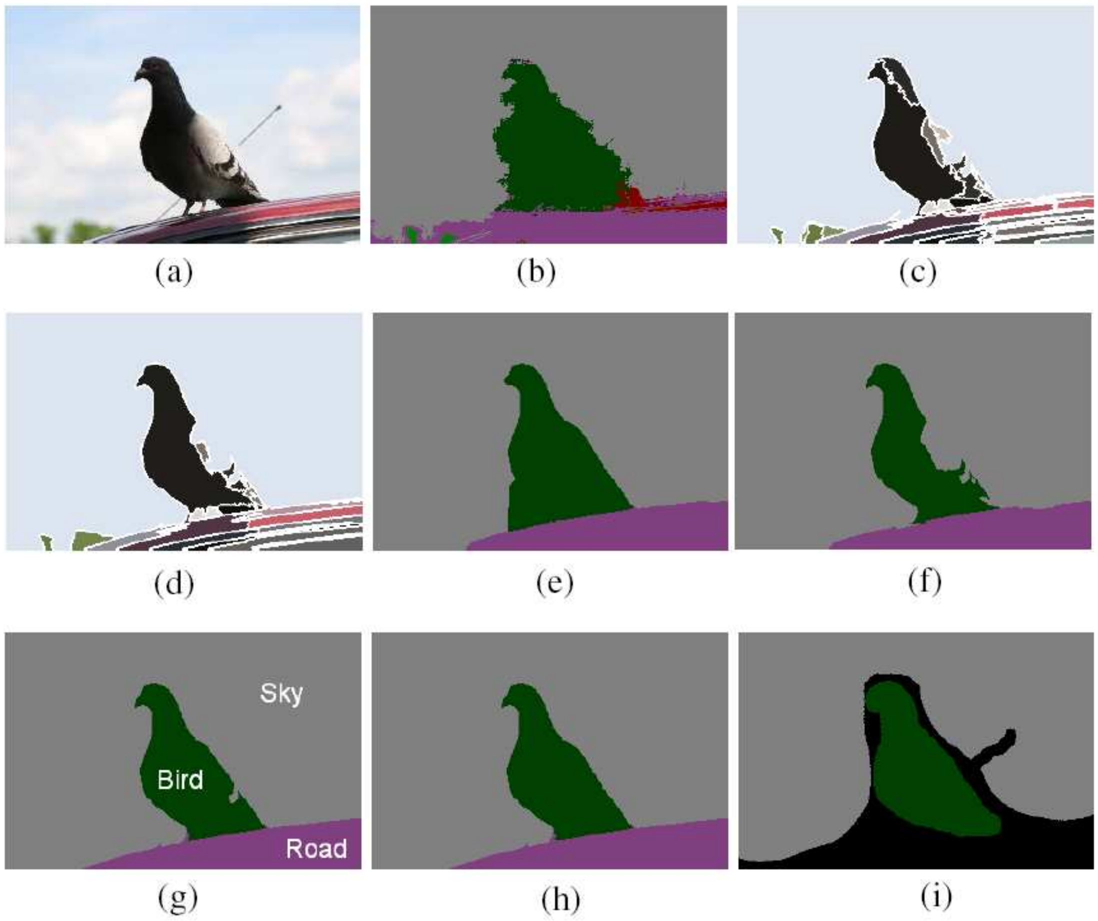


Figure 6.4: *Object segmentation and recognition using the Robust P^n higher order potentials (5.1.1). (a) Original Image. (b) Labelling from unary likelihood potentials from Textonboost [95]. (c) and (d) Segmentations obtained by varying the parameters of the Mean shift algorithm for unsupervised image segmentation [17]. (e) Result obtained using pairwise potential functions as described in [95]. (f) Result obtained using P^n Potts model potentials defined on the segments (or superpixels) shown in (c) and (d). These higher order potentials encourage all pixels in a superpixel to take the same label. The P^n Potts model enforces label consistency in regions very rigidly thus causing certain pixels belonging to the 'bird' to erroneously take the label 'sky' as they were included in the 'sky' superpixel. This problem can be overcome by using the Robust P^n model potentials defined in (5.1.1) which are robust and allow some variables in the clique to take different labels. (g) and (h) show results obtained by using the robust potentials with truncation parameter Q equal to $0.1|c|$ and $0.2|c|$ respectively. Here $|c|$ is equal to the size of the superpixel over which the Robust P^n model potential is defined. (i) Hand labelled segmentation from the* MSRC *dataset.*

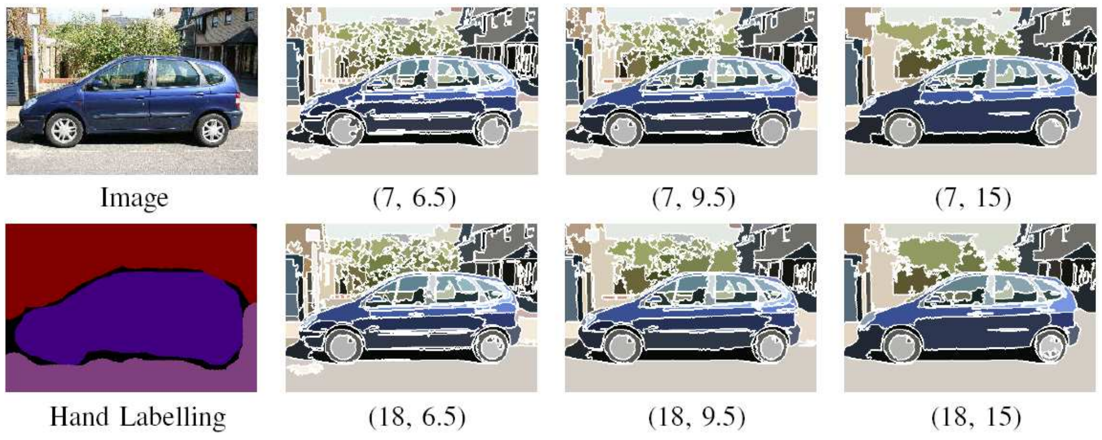


Figure 6.5: *Generating multiple segmentations. The figure shows the segmentations obtained by using different parameters in the mean-shift algorithm. The parameters used for generating the segmentation are written below it in the format* (h_s, h_r)*, where* h_s *and* h_r *are the bandwidth parameters for the spatial and range (colour) domains.*

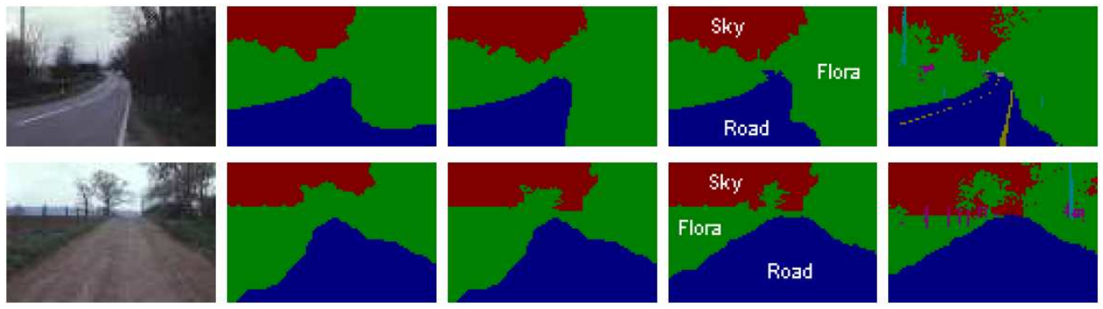


Figure 6.6: *Qualitative object segmentation and recognition results. The first column shows the original image from the Sowerby-7 dataset. Column 2 shows the result of performing inference in the pairwise* CRF *model described in section 6.1.2. The result obtained using the* P^n *Potts potential (6.1.7) is shown in column 3. The results of using the Robust* P^n *potential (6.1.10) is shown in column 4. The hand labelled segmentation used as ground truth is shown in column 5.*

Inference in Higher Order CRFs The MAP solution of the higher order CRF is inferred by minimizing the energy function composed of the potentials described above. This can be done using the algorithm proposed in the previous chapter.

6.1.4 Experiments

In this section we provide the details of our experiments. For comparative evaluation of our method we implemented the state of the art TextonBoost [95] algorithm which uses a pairwise CRF. We then augmented the CRF model by adding higher order potentials defined on segments obtained from mean-shift [17].

6.1.4.1 Datasets

We tested both the pairwise CRF and higher order CRF models on the MSRC-23 [95] and Sowerby-7 [34] datasets. The MSRC dataset contains 23 object classes and comprises of 591 colour images of 320×213 resolution. The Sowerby dataset contains 7 object classes and comprises of 104 colour images of 96×64 resolution. In our experiments, 50% of the images in the dataset were used for training and the remaining were used for testing.

6.1.4.2 Setting CRF parameters

The optimal values for different parameters of the higher order CRF were found in a manner similar to the one used for the pairwise CRF in [95]. The model parameters were learned by minimizing the overall pixelwise classification error rate on a set of validation images - a subset of training images which were not used for training unary potentials.

A simple method for selecting parameter values is to perform cross-validation for every combination of unary, pairwise and higher order parameters within a certain discretized range. Unfortunately, the space of possible parameter values is high dimensional and doing an exhaustive search is infeasible even with very few discretization levels for each parameter. We used a heuristic to overcome this problem. First we learned the weighting between unary potentials from colour, location and Textonboost. Then we kept these weights constant and learned the optimal parameters for pairwise potentials. Pairwise and higher order potentials have similar functionality in the framework, thus learning of higher order parameters from the model with optimal unary and pairwise parameters would lead to very low weights of higher order potentials. Instead we learned optimal higher order parameters in CRF with only unary and higher order potentials and in the last step the ratio between pairwise and higher order potentials.

The final trained coefficients for the MSRC dataset were $\theta_T = 0.52$, $\theta_{col} = 0.21$, $\theta_l = 0.27$, $\theta_p = 1.0$, $\theta_v = 4.5$, $\theta_\beta = 16.0$, $\theta_\alpha = 0.8$, $\theta_p^h = 0.2$, $\theta_v^h = 0.5$, $\theta_\beta^h = 12.0$.

The results of our experiments show that integration of higher order P^n Potts model potentials quantitatively and qualitatively improves segmentation results.

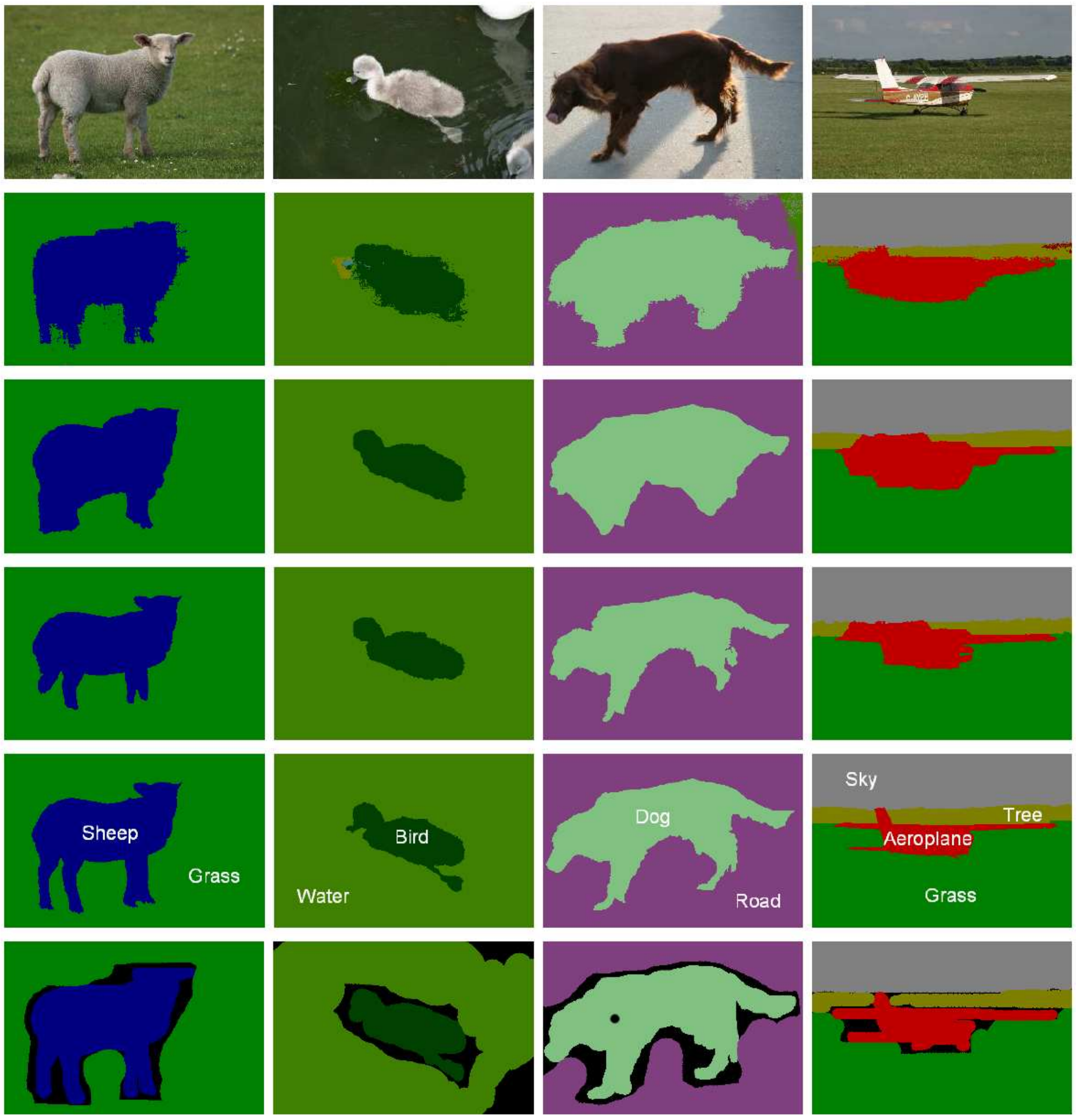


Figure 6.7: *Some qualitative results. Please see in colour. First Row: Original Image. Second Row: Unary likelihood labelling from Textonboost [95]. Third Row: Result obtained using a pairwise contrast preserving smoothness potential as described in [95]. Fourth Row: Result obtained using the P^n Potts model potential [46]. Fifth Row: Results using the Robust P^n model potential (5.1.1) with truncation parameter $Q = 0.1|c|$, where $|c|$ is equal to the size of the superpixel over which the Robust P^n higher order potential is defined. Sixth Row: Hand labelled segmentations. Observe that the results obtained using the Robust P^n model are significantly better than those obtained using other methods. For instance, the leg of the sheep and bird have been accurately labelled which was missing in other results. Same can be said about the tail and leg of the dog, and the wings of the aeroplane.*

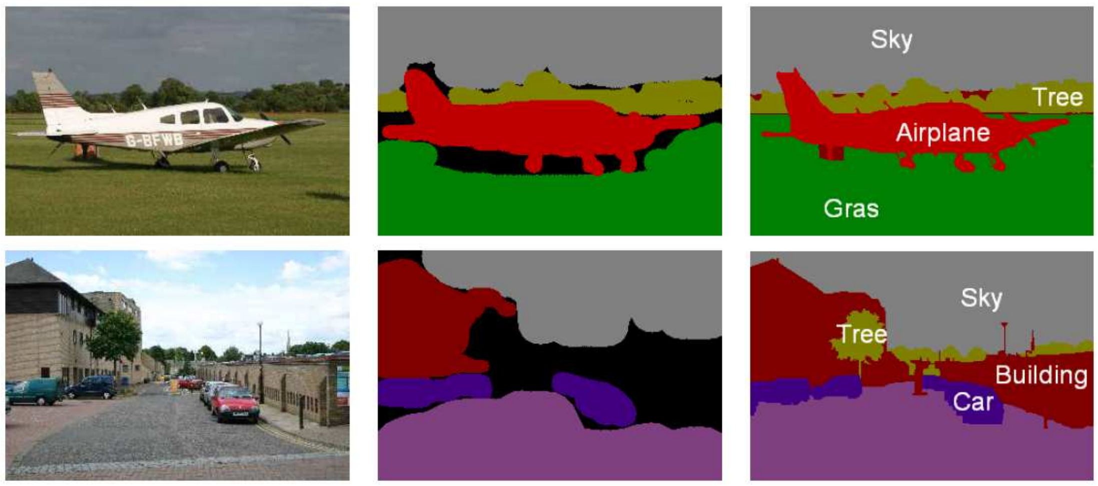


Figure 6.8: *Accurate hand labelled segmentations which were used as ground truth. The figure shows some images from the* MSRC *data set (column 1), the hand labelled segmentations that came with the data set (column 2), and the new segmentations hand labelled by us which were used as ground truth (column 3).*

The use of the robust potentials lead to further improvements (see figure 6.4,6.6, 6.7 and 6.9). Inference on both the pairwise and higher order CRF model was performed using the graph cut based expansion move algorithm [11,46]. The optimal expansion moves for the energy functions containing the Robust P^n potential (6.1.10) were computed using the method given in section 5.2.

6.1.4.3 Quantitative Segmentation Results

The hand labelled 'ground truth' images that come with the MSRC-23 data set are quite rough. In fact qualitatively they always looked worse than the results obtained from our method. The hand labelled images suffer from another drawback. A significant numbers of pixels in these images have not been assigned any label. These unlabelled pixels generally occur at object boundaries and are critical in evaluating the accuracy of a segmentation algorithm. It should be noted that obtaining an accurate and fine segmentation of the object is important for many tasks in computer vision.

Ground Truth In order to get a good estimate of our algorithm's accuracy, we generated accurate segmentations which preserved the fine object boundaries present in the image. Generating these segmentations is quite time consuming. It takes between 15-60 minutes to hand label one image. We hand labelled 27 images from the MSRC data set. Figure 6.8 shows the original hand labelled

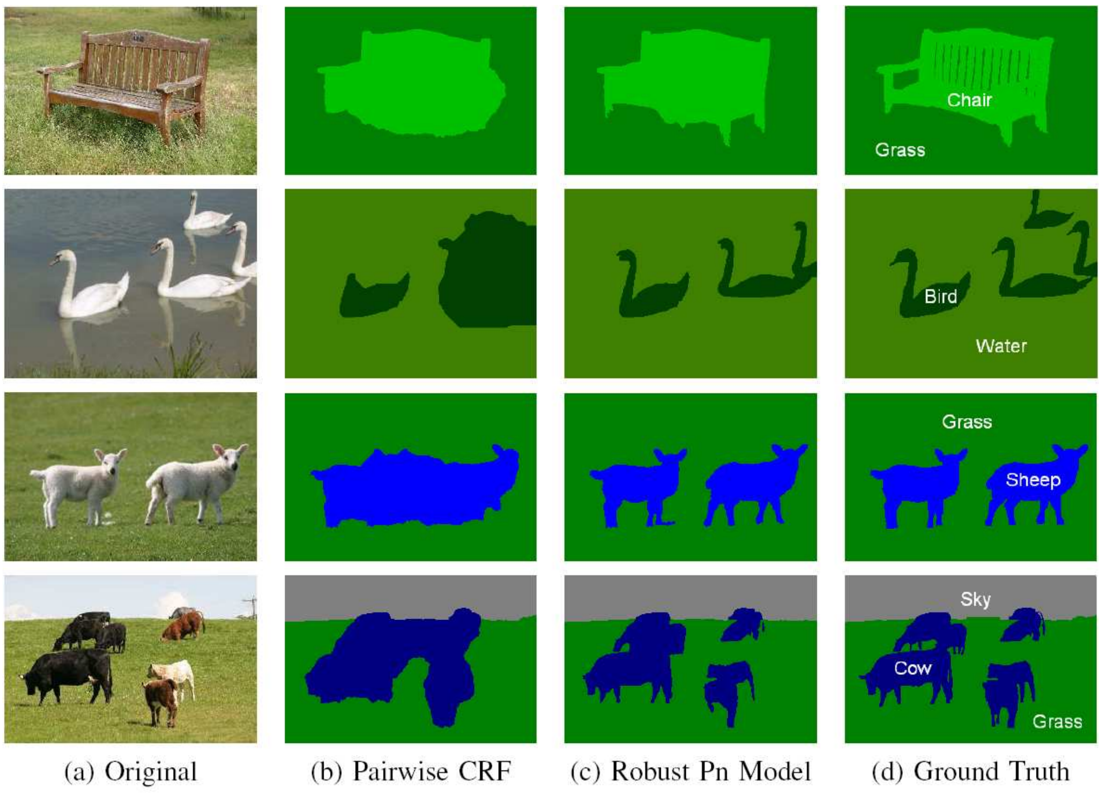


Figure 6.9: *Qualitative results of our method. (a) Original Images. (b) Segmentation result obtained using the pairwise CRF (explained in section 6.1.2). (c) Results obtained by incorporating the Robust P^n higher order potential (6.1.10) defined on segments. (d) Hand labelled result used as ground truth.*

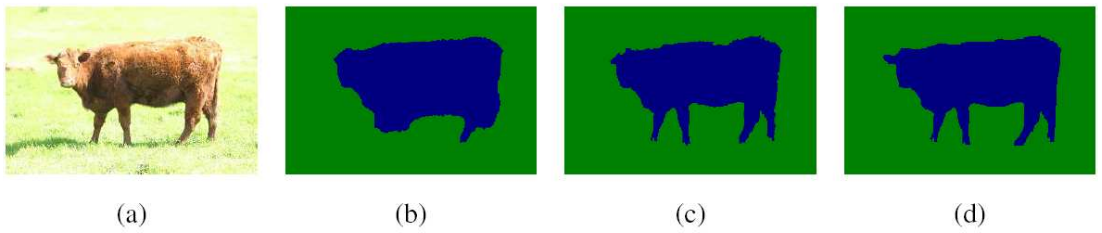


Figure 6.10: *The relationship between qualitative and quantitative results. (a) Original Image. (b) Segmentation result obtained using the pairwise CRF (explained in section 6.1.2). Overall pixelwise accuracy for the result is 95.8%. (c) Results obtained by incorporating the Robust P^n higher order potential (6.1.10) defined on segments. Overall pixelwise accuracy for this result is 98.7%. (d) Hand labelled result used as ground truth. It can be seen that even a small difference in the pixelwise accuracy can produce a massive difference in the quality of the segmentation.*

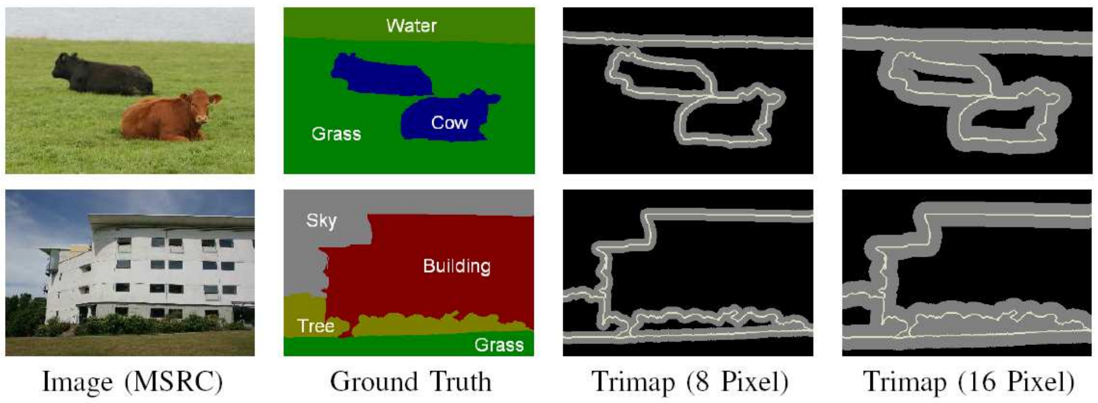


Figure 6.11: *Boundary accuracy evaluation using trimap segmentations. The first column shows some images from the MSRC dataset [95]. The ground truth segmentation of these images are shown in the column 2. Column 3 shows the trimap used for measuring the pixel labelling accuracy. The evaluation region is coloured gray and was generated by taking an 8 pixel band surrounding the boundaries of the objects. The corresponding trimaps for an evaluation band width of 16 pixels are shown in column 4.*

images of the MSRC data set and the new segmentations manually labelled by us which were used as ground truth.

Evaluating Accuracy Typically the performance of a segmentation algorithm is measured by counting the total number of mislabelled pixels in the image. We believe this measure is not appropriate for measuring the segmentation accuracy if the user is interested in obtaining segmentations with fine object boundaries. As only a small fraction of image pixels lie on the boundary of an object, a large qualitative improvement in the quality of the segmentation will result in only a small increase in the percentage pixel-wise accuracy. This phenomenon is illustrated in figure 6.10.

With this fact in mind, we evaluate the quality of a segmentation by counting the number of pixels misclassified in the region surrounding the actual object boundary and not over the entire image. The error was computed for different widths of the evaluation region. The evaluation regions for some images from the MSRC dataset are shown in figure 6.11. The accuracy of different segmentation methods is plotted in the graph shown in figure 6.12.

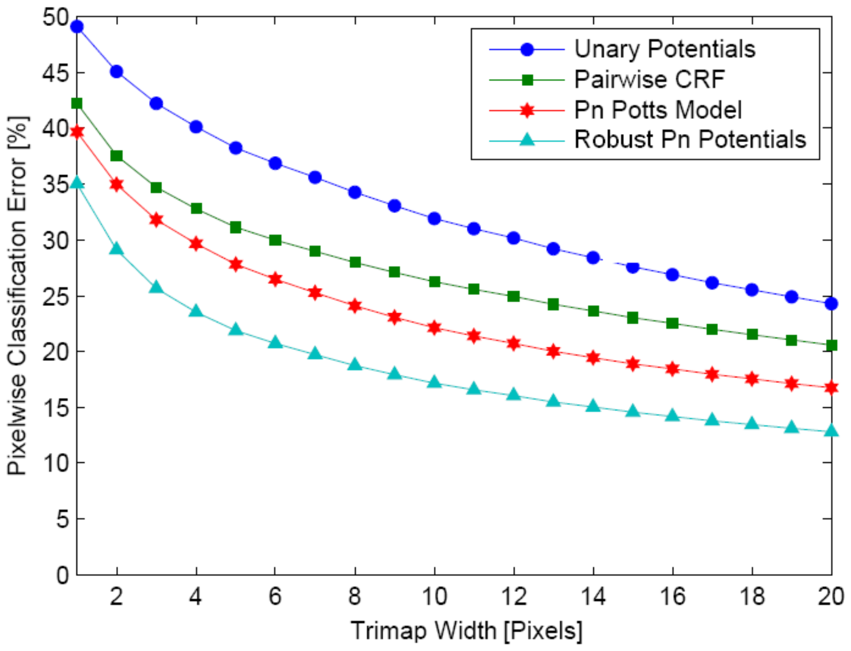


Figure 6.12: *Pixelwise classification error in our results. The graph shows how the overall pixelwise classification error varies as we increase the width of the evaluation region.*

6.1.5 Summary

In this part of the chapter we proposed a novel framework for labelling problems which is capable of utilizing features based on sets of pixels in a principled manner. We tested this approach on the problem of multi-class object segmentation and recognition. Our experiments showed that incorporation of P^n Potts and Robust P^n model type potential functions (defined on segments) in the conditional random field model for object segmentation improved results. We believe this method is generic and can be used to solve many other labelling problems. In the future we would like to investigate the use of more sophisticated higher order potentials based on the shape and appearance of image segments. We believe that such potentials would be more discriminative and will result in even better performance.

6.2. Texture Based Segmentation

We now consider the problem of texture based segmentation. This problem can be stated as follows. Given a set of distinct textures (e.g. a dictionary of RGB patches or histograms of textons [91]) together with their object class labels, the task is to segment an image. In other words, the pixels of the image should be labelled as belonging to one of the object classes (e.g. see Fig. 6.14).

The above problem can be formulated within a probabilistic framework using a CRF [61]. A CRF represents the conditional distribution of a set of random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ given the data $\mathbf{D}$. Each of the variables can take one label $x_i \in \mathcal{L} = \{1, 2, \ldots, n_s\}$. In our case, n_s is the number of distinct object classes, a variable X_i represents a pixel $\mathbf{D}_i$ and $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ describes a segmentation. The most (or a highly) probable (i.e. maximum a posteriori) segmentation can be obtained by (approximately) minimizing the corresponding Gibbs energy.

6.2.1 Pairwise CRF for Texture Segmentation

For the problem of segmentation, it is common practice to assume a pairwise CRF where the cliques are of size at most two [4, 9, 83]. In this case, the Gibbs energy of the CRF is of the form:

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{i \in \mathcal{V}, j \in \mathcal{N}_i} \psi_{ij}(x_i, x_j), \tag{6.2.1}$$

where $\mathcal{N}_i$ is the neighbourhood of pixel $\mathbf{D}_i$ (defined in this work as the 8-neighbourhood). The unary potential $\psi_i(x_i)$ is specified by the RGB distributions $\mathcal{H}_a, a = 1, \ldots, n_s$ of the segments as

$$\psi_i(x_i) = -\log p(\mathbf{D}_i | \mathcal{H}_a), \text{ when } x_i = a. \tag{6.2.2}$$

The pairwise potentials $\psi_{ij}(x_i, x_j)$ are defined such that they encourage contiguous segments whose boundaries lie on image edges, i.e.

$$\psi_{ij}(x_i, x_j) = \begin{cases} \lambda_1 + \lambda_2 \exp\left(\frac{-g^2(i,j)}{2\sigma^2}\right) & \text{if} \quad x_i \neq x_j, \\ 0 & \text{if} \quad x_i = x_j, \end{cases} \tag{6.2.3}$$

where λ_1, λ_2 and σ are some parameters. The term $g(i, j)$ represents the difference between the RGB values of pixels $\mathbf{D}_i$ and $\mathbf{D}_j$. We refer the reader to [9] for details. Note that the pairwise potentials $\psi_{ij}(x_i, x_j)$ form a metric. Hence, the energy function in equation (6.2.1) can be minimized using both $\alpha\beta$-swap and α-expansion algorithms.

6.2.2 Higher Order Patch Potentials

The $\mathcal{P}^n$ functions presented in chapter 4 allow us to go beyond the pairwise CRF framework by incorporating texture information as higher order cliques. Unlike

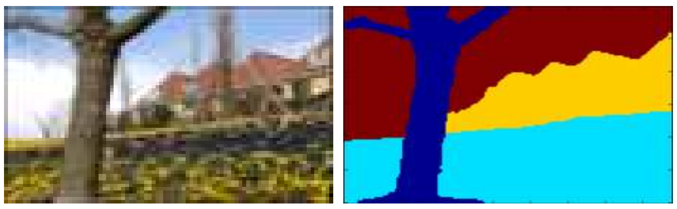


Figure 6.13: *Segmented keyframe of the garden sequence. The left image shows the keyframe while the right image shows the corresponding segmentation provided by the user. The four different colours indicate pixels belonging to the four segments namely sky, house, garden and tree.*

the distributions $\mathcal{H}_a$ which describe the potential for one variable X_i, texture captures rich statistics of natural images [64,108]. In this work, we represent the texture of each object class $s \in \{1, 2, \cdots, n_s\}$ using a dictionary $\mathbf{P}_s$ of $n_p \times n_p$ RGB patches. Note, however, that our framework is independent of the representation of texture. As we will describe later, the likelihood of a patch of the image $\mathbf{D}$ belonging to the segment s can be computed using the dictionary $\mathbf{P}_s$.

The resulting texture based segmentation problem can be formulated using a CRF composed of higher order cliques. We define the Gibbs energy of this CRF as

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{i \in \mathcal{V}, j \in \mathcal{N}_i} \psi_{ij}(x_i, x_j) + \sum_{c \in \mathcal{C}} \psi_c(\mathbf{x}_c), \tag{6.2.4}$$

where c is a clique which represents the patch $\mathbf{D}_c = \{\mathbf{D}_i, i \in c\}$ of the image $\mathbf{D}$ and $\mathcal{C}$ is the set of all cliques. Note that we use overlapping patches $\mathbf{D}_c$ such that $|\mathcal{C}| = n$. The unary potentials $\psi_i(x_i)$ and the pairwise potentials $\psi_{ij}(x_i, x_j)$ are given by equations (6.2.2) and (6.2.3) respectively. The clique potentials $\psi_c(\mathbf{x}_c)$ are defined such that they form a $\mathcal{P}^n$ Potts model ($n = n_p^2$), i.e.

$$\psi_c(\mathbf{x}_c) = \begin{cases} \lambda_3 G(c, s) & \text{if} \quad x_i = s, \forall i \in c, \\ \lambda_4 & \text{otherwise.} \end{cases} \tag{6.2.5}$$

Here $G(c, s)$ is the minimum difference between the RGB values of patch $\mathbf{D}_c$ and all patches belonging to the dictionary $\mathbf{P}_s$. Note that the above energy function encourages a patch $\mathbf{D}_c$ which is similar to a patch in $\mathbf{P}_s$ to take the label s. Since the clique potentials form a $\mathcal{P}^n$ Potts model, they can be minimized using the $\alpha\beta$-swap and α-expansion algorithms as described in section 5.2.

6.2.3 Results

We tested our approach for segmenting frames of a video sequence. A *keyframe* of the video was manually segmented and used to learn the distributions $\mathcal{H}_a$ and the

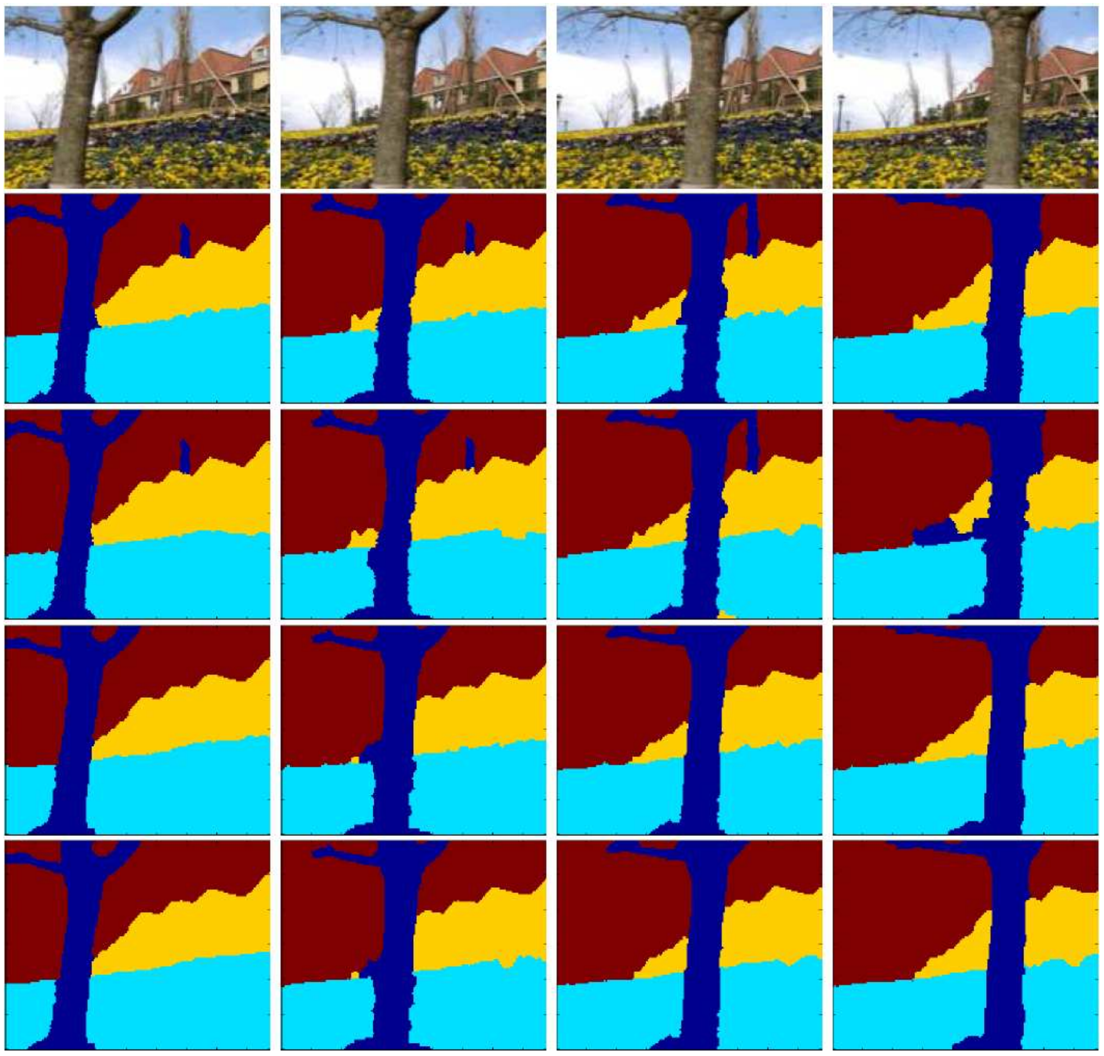


Figure 6.14: *Qualitative texture segmentation results of the garden sequence. The first row shows four frames of the garden sequence. The second row shows the segmentation obtained by minimizing the energy of the pairwise* CRF *(in equation (6.2.1)) using the $\alpha\beta$-swap algorithm. The four different colours indicate the four segments. The segmentations obtained using α-expansion to minimize the same energy are shown in the third row. The fourth row shows the results obtained by minimizing the energy containing higher order clique terms which form a $\mathcal{P}^n$ Potts model (given in equation (6.2.4)) using the $\alpha\beta$-swap algorithm. The fifth row shows the results obtained using the α-expansion algorithm to minimize the energy in equation (6.2.4). The use of higher order cliques results in more accurate segmentation.*

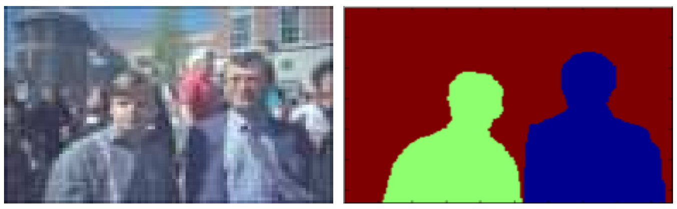


Figure 6.15: *The keyframe of the 'Dayton' video sequence segmented into three segments.*

dictionary of patches $\mathbf{P}_s$. The $\alpha\beta$-swap and α-expansion algorithms were used to perform segmentation on the other frames. In all our experiments, we used patches of size 4×4, together with the following parameter setting: $\lambda_1 = 0.6$, $\lambda_2 = 6$, $\lambda_3 = 0.6$, $\lambda_4 = 6.5$ and $\sigma = 5$. All experiments were performed on an Intel Pentium 2GHz machine.

Fig. 6.13 shows the segmented keyframe of the well-known garden sequence. Fig. 6.14 (row 2) shows the segmentation obtained for four frames by minimizing the energy function of the pairwise CRF (defined in equation (6.2.1)) using the $\alpha\beta$-swap algorithm. Note that these frames are different from the keyframe (see Fig. 6.14 (row 1)). The results obtained by the α-expansion algorithm are shown in Fig. 6.14 (row 3). The α-expansion algorithm takes an average of 3.7 seconds per frame compared to the 4.7 seconds required by the $\alpha\beta$-swap algorithm. Note that the segmentations obtained by both the algorithms are inaccurate due to small clique sizes.

Fig. 6.14 (row 4) shows the segmentations obtained when the energy function of the higher order CRF (defined in equation (6.2.4)) is minimized using $\alpha\beta$-swap. Fig. 6.14 (row 5) shows the results obtained using the α-expansion algorithm. On average, α-expansion takes 4.42 seconds while $\alpha\beta$-swap takes 5 seconds which is comparable to the case when the pairwise CRF is used. For both $\alpha\beta$-swap and α-expansion, the use of higher order cliques provides more accurate segmentation than the pairwise CRF formulation.

Fig. 6.15 shows another example of a segmented keyframe from a video sequence. The segmentations obtained for four frames of this video are shown in Fig. 6.16. Note that even though we do not use motion information, the segmentations provided by higher order cliques are comparable to the methods based on layered motion segmentation.

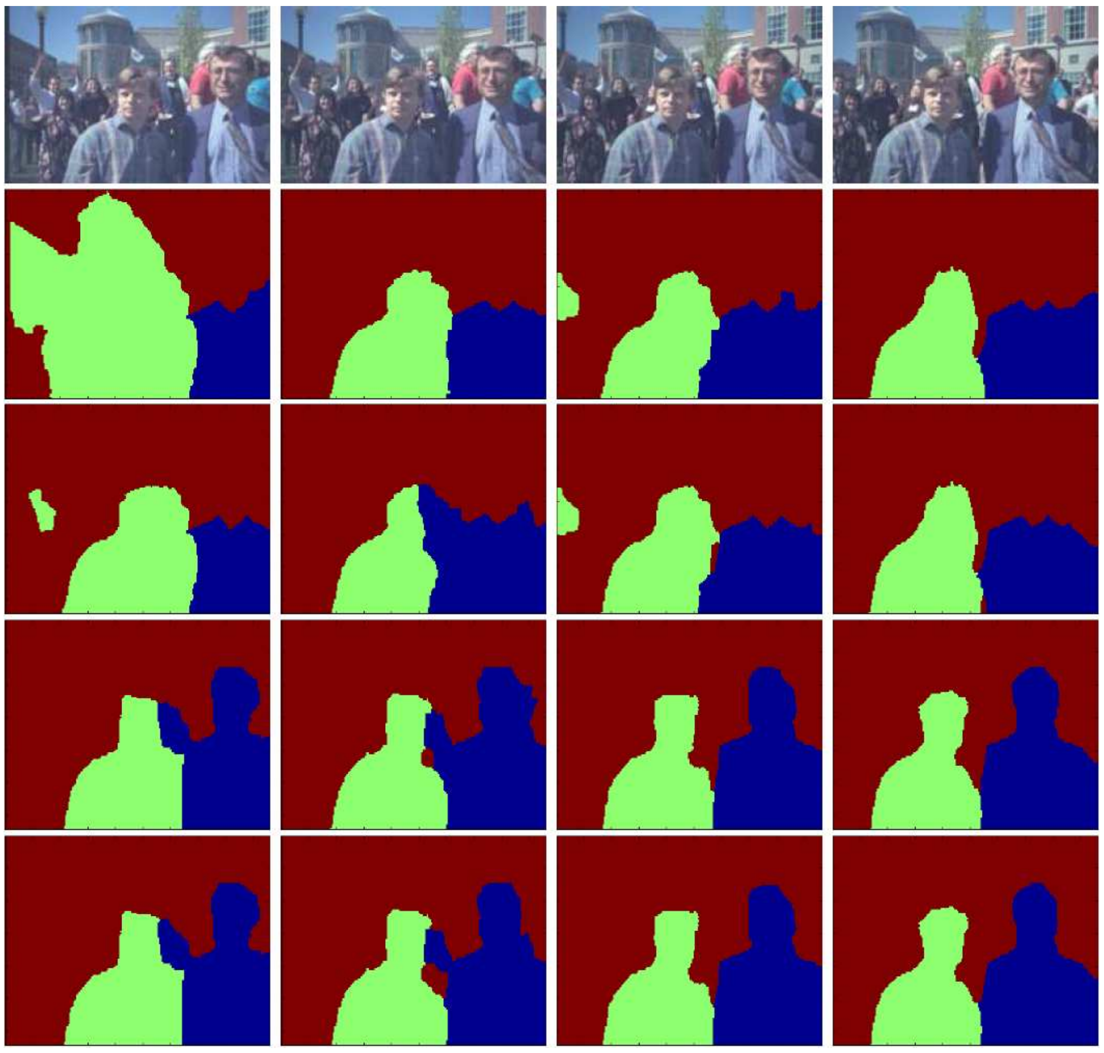


Figure 6.16: *Segmentation results of the 'Dayton' sequence. Rows 2 and 3 show the results obtained for the frames shown in row 1 by minimizing the energy function in equation (6.2.1) using $\alpha\beta$-swap and α-expansion respectively. Row 4 and 5 show the segmentations obtained by minimizing the energy in equation (6.2.4) using $\alpha\beta$-swap and α-expansion respectively. The use of higher order cliques results in more accurate segmentation.*

Chapter 7

Conclusion

7.1. Summary

This dissertation addresses the problem of minimizing functions of discrete variables using graph cuts. The last few decades have seen function minimization being used extensively in computer vision where it is called discrete energy minimization. This rise in popularity has primarily been on the back of its very successful application in solving low level vision problems such as image segmentation, object reconstruction and disparity estimation. The scale and form of computer vision problems introduce many challenges in solving this problem, some of which have been dealt in this thesis.

The first part of this dissertation targets the problem of minimizing multiple similar functions. We showed how algorithms for solving the minimization problem can be made efficient by reusing (or 'recycling') computation performed for solving previous problem instances. This strategy results in dramatic improvements in the running time, which makes us able to solve problems[1] which were considered intractable due to computational cost earlier.

Higher order potential functions can be used to model the rich statistics of natural scenes which are considered extremely important for solving vision problems [82,114]. These functions however cannot be minimized efficiently using the graph cut based move making algorithms generally used in computer vision for energy minimization. We have studied this problem in detail in this dissertation and provided a characterization of higher order functions which can be minimized using move-making algorithms. We proposed new methods for efficiently solving such functions and showed the results of these methods on the problems of object segmentation and recognition.

7.2. Our Contributions

The major contributions discussed in this dissertation are listed below.

Dynamic Graph Cuts We proposed a fast new fully dynamic algorithm for the st-mincut/max-flow problem which can be used to efficiently minimize dynamically changing submodular energy functions encountered in computer vision. This algorithm enables the efficient minimization of several similar energy functions.

[1]such as uncertainty estimation, refer to chapter 3.

Simultaneous Segmentation/Reconstruction and Pose Estimation We proposed a novel algorithm for performing integrated segmentation/reconstruction and pose estimation of an object. Unlike other state of the art methods which focus on either segmentation or pose estimation individually, our approach tackles these two tasks together. Our method works by efficiently optimizing a cost function based on a Conditional Random Field (CRF) using the dynamic graph cut algorithm. This framework has the advantage that all information in the image (edges, background and foreground appearances), as well as the prior information on the shape and pose of the subject can be combined and used in a Bayesian framework.

Measuring Uncertainty in Graph Cut Solutions We proposed an efficient algorithm for computing min-marginals in graphical models with loops. This algorithm is based on the dynamic graph cut algorithm and runs in polynomial time. The min-marginal energies obtained by our proposed algorithm are exact, as opposed to the ones obtained from other inference algorithms like loopy belief propagation and generalized belief propagation. This algorithm can be used on all functions which can be minimized exactly using the st-mincut algorithm.

Characterization of Solvable Higher Order Potentials We provided a characterization of energy functions defined on cliques of size 3 or more which can be solved using move making algorithms. We proved that the optimal α-expansion and $\alpha\beta$-swap moves for this class of functions can be computed in polynomial time by minimizing a submodular function. The class of functions characterized by us is a generalization of the class of energy functions specified by [11].

Solving Robust P^n Model Potentials using Graph Cuts We introduced a novel family of higher order potentials which we call the Robust P^n model. We showed that the optimal expansion and swap moves for energy functions composed of such potentials can be found by solving a st-mincut problem. Our method for computing the optimal expansion and swap moves is extremely efficient and can handle potentials defined over cliques consisting of thousands of random variables.

Higher Order Potentials for Merging Superpixels We proposed a general framework for solving labelling problems which has the ability of incorporating information derived from sets of pixels (superpixels or segments). We showed how this framework can be used to merge regions from multiple image segmentations in a principled manner.

7.3. Directions for Future Work

I conclude this dissertation by providing some directions for future work. It was shown in the thesis how dynamic computation can be used to make minimization of submodular functions faster. There is a clear need for extending these results for minimizing non-submodular functions which are quite common in vision. Some progress on this front has already been made by the work of [56] who showed how solutions of previous problems can be reused while minimizing certain non-submodular functions of multi-valued variables. However, the problem of extending these results for the general class of non-submodular still remains an open problem. Another interesting and promising direction of research is the development of efficient dynamic algorithms for functions which change in a predefined manner. For instance, algorithms for parametric maxflow can efficiently minimize dynamic energy functions if they change in a particular way [30]. Development and application of such algorithms to computer vision problems should intuitively improve performance.

Researchers have discovered that many vision problems require the minimization of energy functions which are non-submodular. As we have discussed earlier in chapter 1, no algorithms with polynomial run time complexity are known for minimizing such functions. The algorithms commonly used for solving this problem are only able to return a local minima or a partially optimal solution. Further, these algorithms provide no approximation guarantees while solving general energy functions. Even for functions for which such guarantees exist, they are too weak to be useful in practice. For instance, the alpha-expansion algorithm provides a very loose bound of 2 for Potts model potentials. The development of efficient approximate algorithms with tighter bounds on the solutions is an important direction for future work.

The use of higher order relations in modeling vision problems has become increasingly popular in the last couple of years. However, such functions generally cannot be handled using graph cut based methods for energy minimization. Recent work on graph cut based methods has contributed in trying to overcome this limitation [27, 46]. However, these methods have restricted their attention to particular subclasses of submodular higher order potentials. Developing efficient algorithms for exact and approximate minimization of general higher order functions is still an interesting and challenging problem to solve.

I conclude this dissertation by the following observations: Graph cut based minimization algorithms have made a deep impact on the field of computer vision. Not only have they provided extremely efficient and robust solutions for some of the most challenging vision problems, but have also contributed in encouraging researchers to adopt the optimization paradigm to formulate their problems. The recent interest in the development of new graph cut based algorithms shows that these methods will remain popular in computer vision for a long time.

Bibliography

[1] A. Agarwal and B. Triggs. 3d human pose from silhouettes by relevance vector regression. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume II, pages 882–888, 2004.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows.* Prentice Hall, Eaglewood Cliffs, NJ, 1993.

[3] S. Bhatia, L. Sigal, M. Isard, and M. J. Black. 3d human limb detection using space carving and multi-view eigen models. In *ANM Workshop*, volume I, page 17, 2004.

[4] A. Blake, C. Rother, M. Brown, P. Perez, and P. H. S. Torr. Interactive image segmentation using an adaptive GMMRF model. In *European Conference on Computer Vision*, pages I: 428–441, 2004.

[5] E. Borenstein and J. Malik. Shape guided object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 969–976, 2006.

[6] E. Boros, P. L. Hammer, and G. Tavares. Preprocessing of unconstrained quadratic binary optimization. Technical Report RRR 10-2006, RUTCOR, Apr 2006.

[7] E. Boros, P. L. Hammer, and G. Tavares. Local search heuristics for quadratic unconstrained binary optimization (QUBO). *J. Heuristics*, 13(2):99–132, 2007.

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

[9] Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *International Conference on Computer Vision*, pages I: 105–112, 2001.

[10] 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.

[11] 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.

[12] M. Bray, P. Kohli, and P. H. S. Torr. Posecut: Simultaneous segmentation and 3d pose estimation of humans using dynamic graph-cuts. In *European Conference on Computer Vision*, pages 642–655, 2006.

[13] C. Chekuri, S. Khanna, J. Naor, and L. Zosin. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Symposium on Discrete Algorithms*, pages 109–118, 2001.

[14] C. Chekuri, S. Khanna, J. Naor, and L. Zosin. A linear programming formulation and approximation algorithms for the metric labeling problem. *SIAM Journal of Discrete Mathematics*, 18(3):608–625, 2005.

[15] Y. Chiang and R. Tamassia. Dynamic algorithms in computational geometry. In *IEEE Special Issue on Computational Geometry*, volume 80, pages 362–381, 1992.

[16] R. F. Cohen and R. Tamassia. Dynamic expression trees and their applications. In *Symposium on Discrete Algorithms*, pages 52–61, 1991.

[17] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619, 2002.

[18] D. Cremers, S. Osher, and S. Soatto. Kernel density estimation and intrinsic alignment for shape priors in level set segmentation. *International Journal of Computer Vision*, 69(3):335–351, 2006.

[19] P. Dawid. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36, 1992.

[20] J. Deutscher, A. Davison, and I. Reid. Automatic partitioning of high dimensional search spaces associated with articulated body motion capture. In *IEEE Conference on Computer Vision and Pattern Recognition (2)*, pages 669–676, 2001.

[21] E. A. Dinic. Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Math. Dokl.*, 11:1277–1280, 1970.

[22] P. F. Felzenszwalb and D. Huttenlocher. Distance transforms of sampled functions. Technical Report TR2004-1963, Cornell University, 2004.

[23] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient matching of pictorial structures. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073, 2000.

[24] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[25] B. Flach. Strukturelle bilderkennung. Technical report, Universit at Dresden, 2002.

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

[27] D. Freedman and P. Drineas. Energy minimization via graph cuts: Settling what is possible. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 939–946, 2005.

[28] D Freedman and T. Zhang. Interactive graph cut based segmentation with shape priors. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume I, pages 755–762, 2005.

[29] S. Fujishige. *Submodular functions and optimization.* North-Holland, Amsterdam, 1991.

[30] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM J. on Comput.*, 18:18:30–55, 1989.

[31] D. M. Gavrila and L. S. Davis. 3D model-based tracking of humans in action: a multi-view approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 73–80, 1996.

[32] D. Greig, B. Porteous, and A. Seheult. Exact maximum a posteriori estimation for binary images. *RoyalStat*, B: 51(2):271–279, 1989.

[33] X. He, R. S. Zemel, and M. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *IEEE Conference on Computer Vision and Pattern Recognition (2)*, pages 695–702, 2004.

[34] X. He, R. S. Zemel, and D. Ray. Learning and incorporating top-down cues in image segmentation. In *European Conference on Computer Vision*, pages 338–351, 2006.

[35] A. Hengel, A. Dick, T. Thormhlen, B. Ward, and P. H. S. Torr. Rapid interactive modelling from video with graph cuts. In *Eurographics*, 2006.

[36] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. *ACM Trans. Graph.*, 24(3):577–584, 2005.

[37] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *International Conference on Computer Vision*, pages 654–661, 2005.

[38] R. Huang, V. Pavlovic, and D. N. Metaxas. A graphical model framework for coupling MRFs and deformable models. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume II, pages 739–746, 2004.

[39] H. Ishikawa. Exact optimization for markov random fields with convex priors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25:1333–1336, October 2003.

[40] H. Ishikawa and D. Geiger. Occlusions, discontinuities, and epipolar lines in stereo. In *European Conference on Computer Vision*, pages 232–248, 1998.

[41] H. Ishikawa and D. Geiger. Segmentation by grouping junctions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 125–131, 1998.

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

[43] O. Juan and Y. Boykov. Active graph cuts. In *IEEE Conference on Computer Vision and Pattern Recognition (1)*, pages 1023–1029, 2006.

[44] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *ACM Symposium on Theory of Computing*, pages 302–311, 1984.

[45] R. Kehl, M. Bray, and L. Van Gool. Full body tracking from multiple views using stochastic sampling. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume II, pages 129 – 136, 2005.

[46] P. Kohli, M. P. Kumar, and P. H. S. Torr. P^3 and beyond: Solving energies with higher order cliques. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[47] P. Kohli, M. P. Kumar, and P. H. S. Torr. Solving energies with higher order cliques. Technical report, Oxford Brookes University, UK, 2007.

[48] P. Kohli and P. H. S. Torr. Efficiently solving dynamic markov random fields using graph cuts. In *International Conference on Computer Vision*, volume II, pages 922–929, 2005.

[49] P. Kohli and P. H. S. Torr. Measuring uncertainty in graph cut solutions: Efficiently computing min-marginal energies using dynamic graph cuts. In *European Conference on Computer Vision*, pages 30–43, 2006.

[50] 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.

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

[52] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother. Bi-layer segmentation of binocular stereo video. In *IEEE Conference on Computer Vision and Pattern Recognition (2)*, pages 407–414, 2005.

[53] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *European Conference on Computer Vision*, pages 82–96, 2002.

[54] 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.

[55] N. Komodakis and G. Tziritas. A new framework for approximate labeling via graph cuts. In *International Conference on Computer Vision*, pages 1018–1025, 2005.

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

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

[58] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Extending pictorial structures for object recognition. In *British Machine Vision Conference*, pages II: 789–798, 2004.

[59] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Obj cut. In *IEEE Conference on Computer Vision and Pattern Recognition (1)*, pages 18–25, 2005.

[60] K. N. Kutulakos and M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3), 2000.

[61] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labelling sequence data. In *International Conference on Machine Learning*, pages 282–289, 2001.

[62] X. Lan and D. P. Huttenlocher. Beyond trees: Common-factor models for 2d human pose recovery. In *International Conference on Computer Vision*, pages 470–477, 2005.

[63] X. Lan, S. Roth, D. P. Huttenlocher, and M. J. Black. Efficient belief propagation with learned higher-order markov random fields. In *European Conference on Computer Vision*, pages 269–282, 2006.

[64] T. Leung and J. Malik. Recognizing surfaces using three-dimensional textons. In *International Conference on Computer Vision*, pages 1010–1017, 1999.

[65] M. E. Leventon, W. E. L. Grimson, and O. D. Faugeras. Statistical shape influence in geodesic active contours. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1316–1323, 2000.

[66] A. Levin and Y. Weiss. Learning to combine bottom-up and top-down segmentation. In *European Conference on Computer Vision*, pages 581–594, 2006.

[67] L. Lovasz. Submodular functions and convexity. In *Mathematical Programming: The State of the Art*, pages 235–257, 1983.

[68] G. Mori, X. Ren, A. A. Efros, and J. Malik. Recovering human body configurations: Combining segmentation and recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (2)*, pages 326–333, 2004.

[69] A. Neubeck, A. Zalesny, and L. van Gool. 3d texture reconstruction from extensive BTF data. In *Texture*, pages 13–18, 2005.

[70] D. Nilsson. An efficient algorithm for finding the m most probable configurations in bayesian networks. *Statistics and Computing*, 8(2):159–173, 1998.

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

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

[73] J. Pearl. Fusion, propagation, and structuring in belief networks. *Artif. Intell.*, 29(3):241–288, 1986.

[74] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical recipes in C*. Cambridge Uni. Press, 1988.

[75] A. Rabinovich, S. Belongie, T. Lange, and J. M. Buhmann. Model order selection and cue combination for image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (1)*, pages 1130–1137, 2006.

[76] A. Raj, G. Singh, and R. Zabih. MRF's for MRI's: Bayesian reconstruction of mr images via graph cuts. In *IEEE Conference on Computer Vision and Pattern Recognition (1)*, pages 1061–1068, 2006.

[77] A. Raj and R. Zabih. A graph cut algorithm for generalized image deconvolution. In *International Conference on Computer Vision*, pages 1048–1054, 2005.

[78] D. Ramanan. Using segmentation to verify object hypotheses. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[79] D. Ramanan and D. A. Forsyth. Finding and tracking people from the bottom up. In *IEEE Conference on Computer Vision and Pattern Recognition (2)*, pages 467–474, 2003.

[80] X. Ren and J. Malik. Learning a classification model for segmentation. In *International Conference on Computer Vision*, pages 10–17, 2003.

[81] J. Rihan, P. Kohli, and P. H. S. Torr. Objcut for face detection. In *Indian Conference on Computer Vision, Graphics and Image Processing*, pages 576–584, 2006.

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

[83] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: interactive foreground extraction using iterated graph cuts. In *ACM Trans. Graph.*, pages 309–314, 2004.

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

[85] C. Rother, S. Kumar, V. Kolmogorov, and A. Blake. Digital tapestry. In *IEEE Conference on Computer Vision and Pattern Recognition (1)*, pages 589–596, 2005.

[86] B. C. Russell, W. T. Freeman, A. A. Efros, J. Sivic, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *IEEE Conference on Computer Vision and Pattern Recognition (2)*, pages 1605–1614, 2006.

[87] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002.

[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] A. Schrijver. *A Theory of Linear and Integer Programming.* John Wiley and Sons, 1998.

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

[91] F. Schroff, A. Criminisi, and A. Zisserman. Single-histogram class models for image segmentation. In *Indian Conference on Computer Vision, Graphics and Image Processing*, 2006.

[92] G. Shakhnarovich, P. Viola, and T.J. Darrell. Fast pose estimation with parameter-sensitive hashing. In *International Conference on Computer Vision*, pages 750–757, 2003.

[93] E. Sharon, A. Brandt, and R. Basri. Segmentation and boundary detection using multiscale intensity measurements. In *IEEE Conference on Computer Vision and Pattern Recognition (1)*, pages 469–476, 2001.

[94] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[95] J. Shotton, J. M. Winn, C. Rother, and A. Criminisi. *TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision*, pages 1–15, 2006.

[96] H. Sidenbladh, M. J. Black, and D. J. Fleet. Stochastic tracking of 3D human figures using 2D image motion. In *European Conference on Computer Vision*, pages 702–718, 2000.

[97] C. Sminchisescu and A. D. Jepson. Generative modeling for continuous non-linearly embedded visual inference. In *International Conference on Machine Learning*, 2004.

[98] C. Sminchisescu and B. Triggs. Covariance scaled sampling for monocular 3D body tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 447–454, 2001.

[99] D. Snow, P. Viola, and R. Zabih. Exact voxel occupancy with graph cuts. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 345–352, 2000.

[100] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 246–252, 1999.

[101] B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla. Filtering using a tree-based estimator. In *International Conference on Computer Vision*, pages 1063–1070, 2003.

[102] Y. Sun, P. Kohli, M. Bray, and P. H. S. Torr. Using strong shape priors for stereo. In *Indian Conference on Computer Vision, Graphics and Image Processing*, pages 882–893, 2006.

[103] R. Szeliski. Rapid octree construction from image sequences. *Computer Vision Graphics and Image Processing*, 58:23–32, 1993.

[104] 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. In *European Conference on Computer Vision*, pages 16–29, 2006.

[105] M. Thorup. Fully-dynamic min-cut. In *ACM Symposium on Theory of Computing*, pages 224–230, 2001.

[106] Z. Tu and S. C. Zhu. Image segmentation by data-driven markov chain monte carlo. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):657–673, 2002.

[107] R. Urtasun, D, J. Fleet, A. Hertzmann, and P. Fua. Priors for people tracking from small training sets. In *International Conference on Computer Vision*, pages 403–410, 2005.

[108] M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 691–698, 2003.

[109] O. Veksler. Graph cut based optimization for MRFs with truncated convex priors. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[110] P. A. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[111] G. Vogiatzis, P.H.S. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume II, pages 391–398, 2005.

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

[113] J. Wang, P. Bhat, A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. *ACM Trans. Graph.*, 24(3):585–594, 2005.

[114] Y. Weiss and W. T. Freeman. What makes a good model of natural images? In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[115] T. Werner. A linear programming approach to max-sum problem: A review. Research Report CTU–CMP–2005–25, Center for Machine Perception, Czech Technical University, December 2005.

[116] J. Xiao and M. Shah. Motion layer extraction in the presence of occlusion using graph cut. In *IEEE Conference on Computer Vision and Pattern Recognition (2)*, pages 972–979, 2004.

[117] C. Yanover and Y. Weiss. Finding the m most probable configurations in arbitrary graphical models. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

[118] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *NIPS*, pages 689–695, 2000.

[119] L. Zhao and L. S. Davis. Closely coupled object detection and segmentation. In *International Conference on Computer Vision*, pages 454–461, 2005.