Chapter 3
Dynamic Graph Cuts and Their Applications in Computer Vision

Pushmeet Kohli and Philip H.S. Torr

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 chapter we address the problem of efficient and exact minimization of groups of similar functions which are known to be solvable in polynomial time. We 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. We 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.

3.1 Introduction

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.

Pushmeet Kohli
Microsoft Research, Cambridge, UK
e-mail: pkohli@microsoft.com

Philip H.S. Torr
Oxford Brookes University, Oxford, UK
e-mail: philiptorr@brookes.ac.uk

R. Cipolla, S. Battiato, & G.M. Farinella (Eds.): Computer Vision, SCI 285, pp. 51–108.
springerlink.com

(a) (b) (c)

Fig. 3.1 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.

Labelling problems occur in many forms, from lattice based problems of dense stereo and image segmentation [9, 66] to the use of pictorial structures for object recognition [18]. Some examples of problems which can be formulated in this manner are shown in Figure 3.1.

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.

3.1.1 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 $\mathbf{L} = \mathcal{L}^n$.

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: $p(\mathbf{x}) > 0 \;\; \forall \mathbf{x} \in \mathcal{X}^n$, and the Markovian property:

$$p(x_v | \{x_u : u \in \mathcal{V} - \{v\}\}) = p(x_v | \{x_u : u \in \mathcal{N}_v\}) \qquad \forall v \in \mathcal{V}. \tag{3.1}$$

Here we refer to $p(X = \mathbf{x})$ by $p(\mathbf{x})$ and $p(X_i = x_i)$ by $p(x_i)$. The pairwise MRF commonly used to model image labelling problems is shown in Figure 3.8.

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

$$p(\mathbf{x}|\mathbf{D}) = \frac{1}{Z} \exp(-\sum_{c \in \mathcal{C}} \psi_c(\mathbf{x}_c)), \tag{3.2}$$

where Z is a normalizing constant known as the partition function, and $\mathcal{C}$ is the set of all cliques [42]. The term $\psi_c(\mathbf{x}_c)$ is known as the potential function of the 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

$$E(\mathbf{x}) = -\log p(\mathbf{x}|\mathbf{D}) - \log Z = \sum_{c \in \mathcal{C}} \psi_c(\mathbf{x}_c) \tag{3.3}$$

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

$$\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathbf{L}} p(\mathbf{x}|\mathbf{D}). \tag{3.4}$$

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.

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 [8, 19, 27, 56].

3.2 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 [7,10,28,29,37,53,75]. Greig *et al.* [24] 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, 8]. 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 [38, 33]. 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 [9, 39, 32, 69]. These solutions for certain problems have been shown to be better than the ones obtained using other inference methods [66].

3.2.1 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 Chapter. 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 $C : E \rightarrow \mathbb{R}$ 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 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:

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

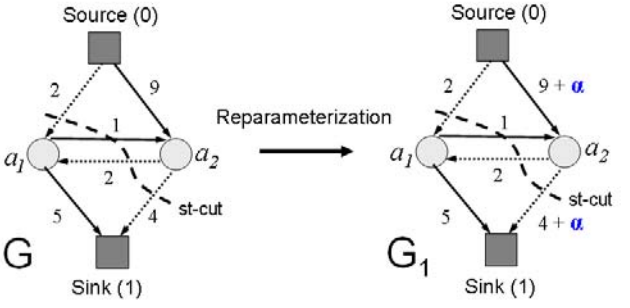


Fig. 3.2 Graph reparameterization. The figure shows a graph G, and its reparameterization G_1 obtained by adding a constant α to both the t-edges of node a_2. The edges included in the st-mincut are depicted by dotted lines. Observe that although the cost of the st-mincut in G and G_1 is different, the st-mincut includes the same edges for both graphs and thus induces the same partitioning of the graph.

$$C_{S,\overline{S}} = \sum_{i \in S, j \in \overline{S}} c_{ij}. \tag{3.5}$$

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 [20], 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].

3.2.2 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 (3.6) and mass balance (3.7) constraints:

$$0 \leq f_{ij} \leq c_{ij} \quad \forall (i,j) \in E, \text{ and} \tag{3.6}$$

$$\sum_{i \in N(j)} (f_{ji} - f_{ij}) = 0 \quad \forall j \in V \tag{3.7}$$

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 3.2. Such transformations result in a reparameterization of the graph and will be explained later in Section 3.3.

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

3.2.4 Minimizing 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 Chapter, we will call functions of this form 'second order functions' or 'functions of order 2'.

Definition 3.1. We say that a function $f : \mathcal{L}^n \rightarrow \mathbb{R}$ is of order k if it can be written in terms of a sum of functions $f_i : \mathcal{L}^k \rightarrow \mathbb{R}$, 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.

Example 3.1. 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

$$f^a(x_1,x_2,x_3) = 4x_1 + 5\overline{x}_1x_2\overline{x}_3 \tag{3.8}$$

is of order 3 because of the maximal term $5\overline{x}_1x_2\overline{x}_3$.

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 [38].

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 [38] described the procedure to construct graphs for

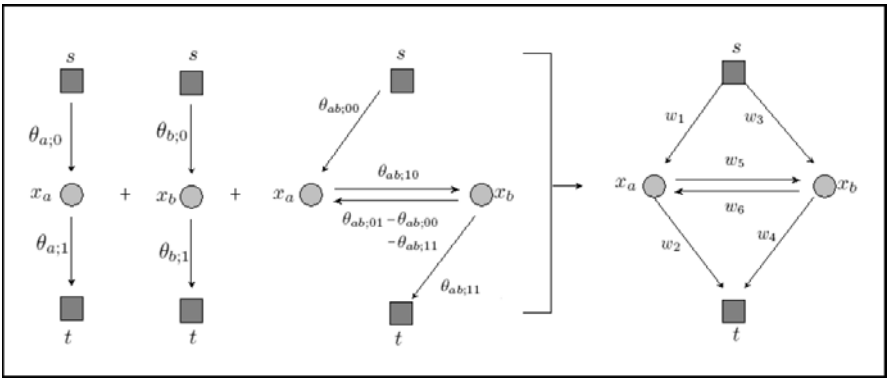


Fig. 3.3 Energy minimization using graph cuts. The Figure shows how individual unary and pairwise terms of an energy function taking two binary variables are represented and combined in the graph. Multiple edges between the same nodes are merged into a single edge by adding their weights. For instance, 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.

minimizing pseudo-boolean functions of order at most 3. The graph constructions for functions of multi-valued variables were given later by [27] and [56].

We now explain the graph construction for minimizing energies involving binary random variables. We use the notation of [35] and write a second order function as:

$$E(\mathbf{x}|\theta) = \theta_{\text{const}} + \sum_{v \in V, i \in \mathcal{L}} \theta_{v;i}\delta_i(x_v) + \sum_{(u,v) \in E, (j,k) \in \mathcal{L}^2} \theta_{uv;jk}\delta_j(x_u)\delta_k(x_v), \quad (3.9)$$

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:

$$\begin{aligned} E(\mathbf{x}|\theta) = \theta_{\text{const}} &+ \sum_{v \in V} (\theta_{v;1}x_v + \theta_{v;0}\overline{x}_v) \\ &+ \sum_{(u,v) \in E} (\theta_{st;11}x_u x_v + \theta_{st;01}\overline{x}_u x_v + \theta_{st;10}x_u \overline{x}_v + \theta_{st;00}\overline{x}_u \overline{x}_v). \end{aligned} \quad (3.10)$$

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 3.3. 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.

3.3 Minimizing Dynamic Energy Functions Using 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. Similarly, successive sub-problems solved in move making algorithms algorithms such as swap, expansion [9], or fusion move [44, 74] are also similar.

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 Section 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 3.4). 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.

3.3.1 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 [11]. A number of algorithms have been proposed for the dynamic mincut problem. Thorup [67] 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.

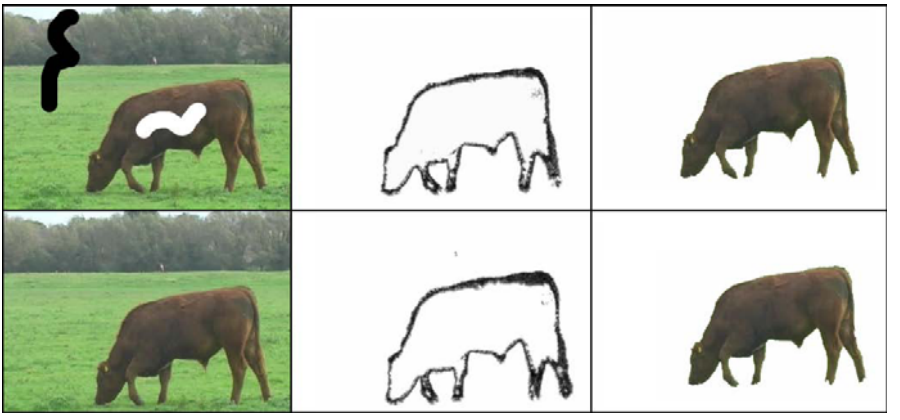


Fig. 3.4 Dynamic image segmentation using graph cuts. The images in the first column are two consecutive frames of the grazing cow video sequence. Their respective segmentations are shown in the third column. The first image in the first column also shows the user segmentation seeds (pixels marked by black (background) and white (foreground) colours). The user marked image pixels are used to learn histograms modelling foreground and background appearance (as in [7]). 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.

Gallo *et al.* [22] 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 graph. Recently, Cohen and Tamassia [12] 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[2].

Boykov and Jolly [7] 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 Section we present a new fully dynamic algorithm

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

for the st-mincut problem which allows for arbitrary changes in the graph[3]. Juan and Boykov [30] 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. Our work has been extended to the minimization of multi-label non-submodular function in [3].

An outline of the Section follows. The relationship between energy and graph reparameterization is explained in Section 3.3.2. Section 3.4 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 3.5, we describe how the process of recomputing the st-mincut/max-flow can be further optimized by using *recycled* search trees.

3.3.2 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 3.10 that a second order energy function can be written in terms of an energy parameter vector θ as:

$$E(\mathbf{x}|\theta) = \theta_{\text{const}} + \sum_{v \in V} (\theta_{v;1} x_v + \theta_{v;1} \overline{x}_v) + \sum_{(u,v) \in E} (\theta_{st;11} x_u x_v + \theta_{st;01} \overline{x}_u x_v + \theta_{st;10} x_u \overline{x}_v + \theta_{st;00} \overline{x}_u \overline{x}_v). \quad (3.11)$$

Two energy parameter vectors θ_1 and θ_2 are called reparameterizations of each other if and only if $\forall \mathbf{x}, E(\mathbf{x}|\theta_1) = E(\mathbf{x}|\theta_2)$ [6,35,56,72]. 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:

$$\forall i \quad \overline{\theta}_{v;i} = \theta_{v;i} + \alpha, \; \overline{\theta}_{const} = \theta_{const} - \alpha \text{ and} \quad (3.12)$$

$$\forall i, j \quad \overline{\theta}_{st;ij} = \theta_{st;ij} + \alpha, \quad \overline{\theta}_{const} = \theta_{const} - \alpha \quad (3.13)$$

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.

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} E(\mathbf{x}|\theta_1) = \arg\min_{\mathbf{x}} E(\mathbf{x}|\theta_2) \quad (3.14)$$

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 Section)

[3] An earlier version of this Section appeared as [33].

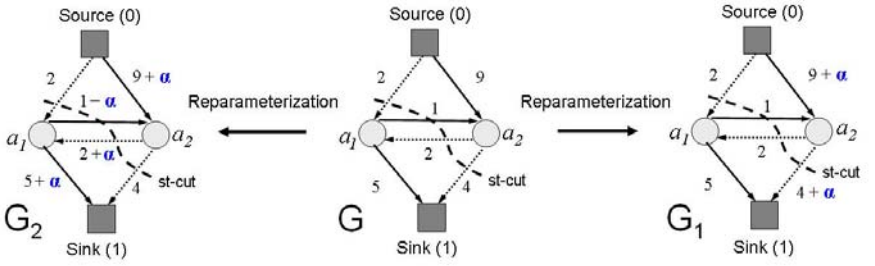


Fig. 3.5 Graph reparameterization. The Figure shows a graph G, its two reparameterizations G_1 and G_2 along with their respective st-mincuts. The edges included in the st-mincut are marked by dotted lines. The reparameterized graphs G_1 and G_2 are a results of two different valid transformations of graph G. It can be clearly seen that reparameterized graphs G_1 and G_2 have the same st-mincut as graph G.

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 (3.12) and (3.13) are shown in Figure 3.5.

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.

3.4 Recycling Computation

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 Section, 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 [7], 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.

3.4.1 Updating Residual Graphs

The flows through a graph can be used to generate a residual graph (as explained in Section 3.1). 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 (3.6) that the flow in an edge of the graph has to satisfy the edge capacity constraint:

$$0 \leq f_{ij} \leq c_{ij} \quad \forall (i,j) \in E. \tag{3.15}$$

While modifying the residual graph, certain flows may violate the new edge capacity constraints (3.15). 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 (3.15) 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 3.3.2 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 [7] and is described below. The updated residual capacity of an edge (s,i) can be computed as:

$$r'_{si} = r_{si} + c'_{si} - c_{si}. \tag{3.16}$$

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 (3.15) 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 3.3.2 and in [7], 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 3.2. 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:

$$r'_{ij} = r_{ij} + (c'_{ij} - c_{ij}). \tag{3.17}$$

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 (3.15) for the new edge capacity. The residual capacity of the edge can still be updated according to equation (3.17). 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 (3.15). 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 3.2 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

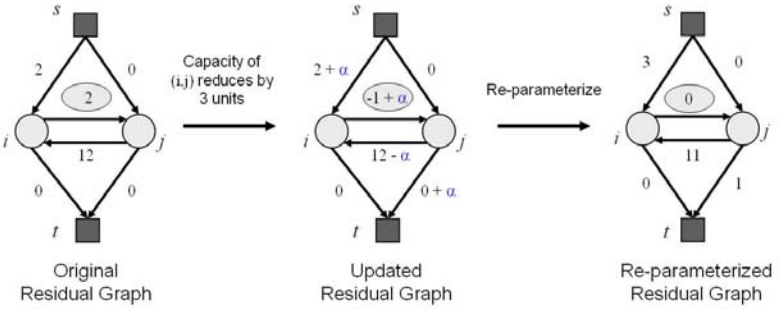


Fig. 3.6 Restoring consistency using graph reparameterization. The Figure illustrates how edge capacities can be made consistent with the flow by reparameterizing the residual graph. It starts by showing a residual graph consisting of two nodes i and j obtained after a max-flow computation. For the second 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.

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 3.6.

3.4.2 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[4].

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 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:

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

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

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.

3.5 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 [8].

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 [16] 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 [8] 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 [8] are the presence of the tree restoration stage, and the dynamic selection of active nodes. We will next describe how the algorithm of [8] works and then explain how we modify it to recycle search trees for dynamic graph cuts.

3.5.1 Reusing Search Trees

The algorithm described in [8] 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.

3.5.2 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 [8] in the way we initialize the set of active nodes and in the presence of the tree restoration stage.

Tree Restoration Stage

While dynamically updating the residual graph (as explained in Section 3.4) 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.

Dynamic Node Activation

The algorithm of [8] 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.

3.6 Dynamic Image Segmentation

The dynamic graph cut algorithm proposed in the previous Section 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 [34, 10, 25, 54].

We now 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 [8] which has been experimentally shown to be the fastest for several vision problems including image segmentation[5].

We will call the algorithm of [8] *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.

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

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

(a) (b) (c)

Fig. 3.7 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).

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 [7] 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 3.7.

3.6.1 CRFs for Image Segmentation

The image segmentation problem is commonly formulated using the CRF model described in Section 3.1. In the context of image segmentation, the vertex set $\mathcal{V}$ corresponds to the set of all image pixels, $\mathcal{N}$ is a neighbourhood defined on this set[6], 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:

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

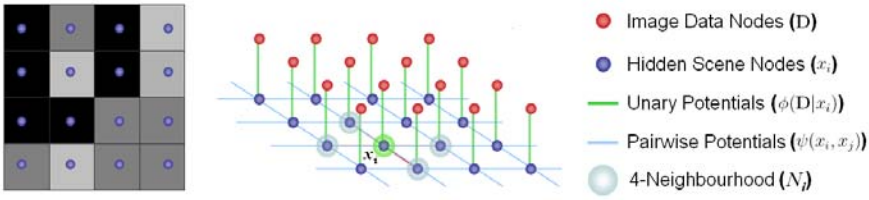


Fig. 3.8 The pairwise MRF commonly used to model image labelling problems. The random field contains a hidden node corresponding to each pixel in the image. The MRF shown in the figure has a 4-neighbourhood, i.e., each node representing the random variables is connected to 4 neighbouring nodes.

$$\Psi_1(\mathbf{x}) = \sum_{i \in \mathcal{V}} \left(\phi(\mathbf{D}|x_i) + \sum_{j \in \mathcal{N}_i} \psi(x_i, x_j) \right) + \text{const.} \tag{3.18}$$

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

$$\phi(\mathbf{D}|x_i) = -\log p(i \in \mathcal{S}_k|\mathcal{H}_k) \qquad \text{if } x_i = k, \tag{3.19}$$

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 $k \in \mathcal{L}$ [7]. The probability of a pixel belonging to a particular segment i.e. $p(i \in \mathcal{S}_k|\mathcal{H}_k)$ is proportional to the likelihood $p(I_i|\mathcal{H}_k)$, where I_i is the colour intensity of the pixel i. The likelihood $p(I_i|\mathcal{H}_k)$ is generally computed from the colour histogram of the pixels belonging to the segment $\mathcal{S}_k$.

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

$$\psi(x_i, x_j) = \begin{cases} K_{ij} & \text{if } x_i \neq x_j, \\ 0 & \text{if } x_i = x_j. \end{cases} \tag{3.20}$$

The CRF used to model the image segmentation problem also contains a contrast term which favours pixels with similar colours having the same label [5, 7]. 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:

$$\gamma(i, j) = \lambda \exp\left(\frac{-g^2(i, j)}{2\sigma^2} \right) \frac{1}{\text{dist}(i, j)}, \tag{3.21}$$

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

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

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

$$\Psi_2(\mathbf{x}) = \sum_{i \in \mathcal{V}} \left(\phi(\mathbf{D}|x_i) + \sum_{j \in \mathcal{N}_i} (\phi(\mathbf{D}|x_i, x_j) + \psi(x_i, x_j)) \right) \tag{3.22}$$

The contrast term of the energy function has the form

$$\phi(\mathbf{D}|x_i, x_j) = \begin{cases} \gamma(i,j) & \text{if } x_i \neq x_j \\ 0 & \text{if } x_i = x_j. \end{cases} \tag{3.23}$$

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 [42]. The pairwise MRF commonly used to model image labelling problems is shown in Figure 3.8.

3.6.2 Image Segmentation in Videos

The object-background segmentation problem aims to cut out user specified objects in an image [7]. 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 (3.22) 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 [7]. 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 a very high cost for a label assignment that violates the hard constraints in a manner similar to [7]. This method for specifying hard constraints has been chosen because of its simplicity. Readers should refer to [73] for a sophisticated method for specifying hard constraints for the video segmentation problem. Figure 3.9 demonstrates the use of constraints in the image segmentation process. The segmentation results are shown in Figure 3.10.

3.6.3 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 [8].

Fig. 3.9 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 [7]. 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.

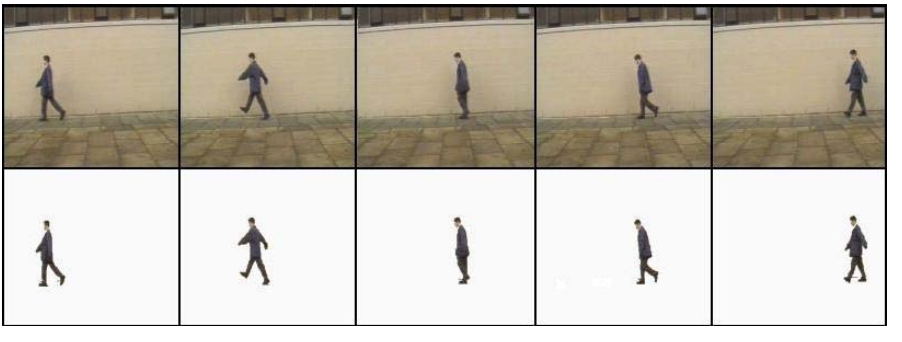


Fig. 3.10 Segmentation results of the human lame walk video sequence.

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 3.7, 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

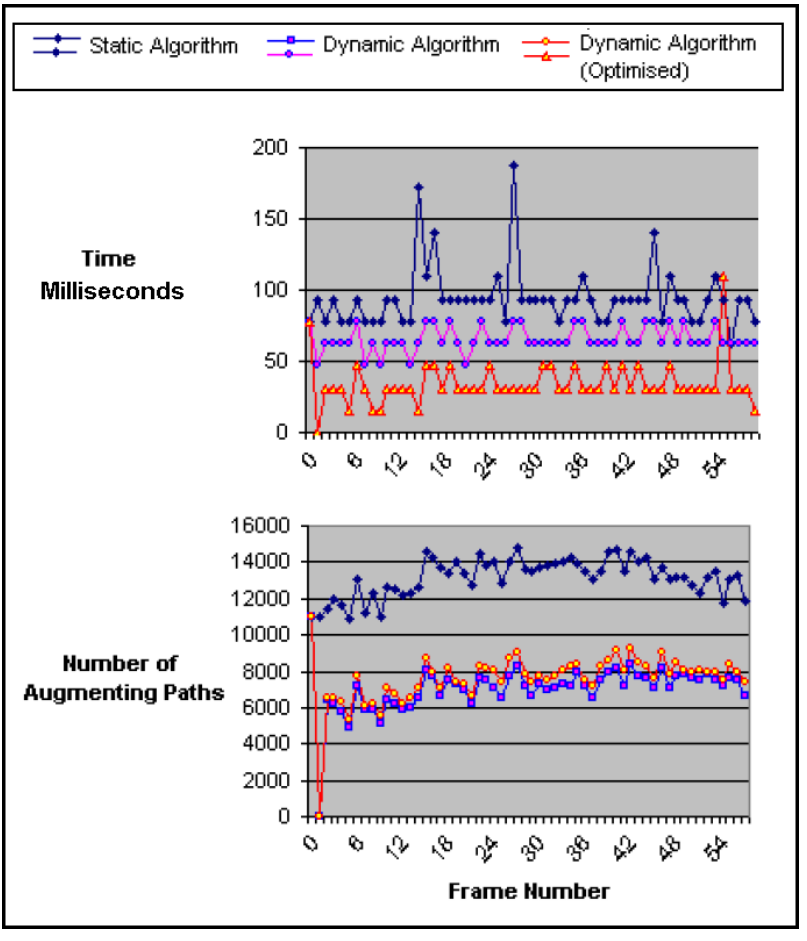


Fig. 3.11 Running time and number of augmenting paths found by static and dynamic st-mincut algorithms. Observe that as the first and second frames of the video sequence are the same, the residual graph does not need to be updated, which results in no augmenting paths found by the dynamic algorithms when segmenting frame 2. Further, the optimized dynamic algorithm takes no time for computing the segmentation for the second image frame as the CRFs 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.

running times per image frame for the static, dynamic and optimized-dynamic algorithms for the human lame walk sequence[8] of size 368x256 were 91.4, 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

[8] Courtesy Derek Magee, University of Leeds.

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 3.11 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.

3.6.4 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 3.11 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 3.12. 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 [34]), 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

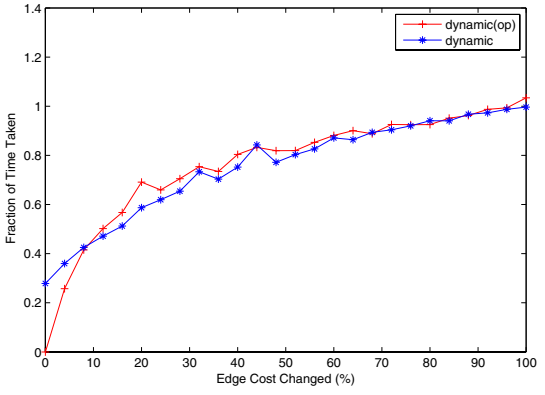


Fig. 3.12 Behavior of the dynamic algorithm. The Figure illustrates how the time taken by the dynamic algorithm (with/without tree recycling) changes with the number of modifications made to the graph. The graph shows the fraction of time taken to compute the st-mincut in the updated residual graph (with/without tree recycling) compared to that taken for computing the st-mincut in the original graph using the algorithm of [8]. 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).

times) out of which 7 msec is used for recycling the tree and 18 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.

3.7 Simultaneous Segmentation and Pose Estimation of Humans

In this Section we 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, 18, 31, 57, 23, 58, 60, 68, 43, 15, 46, 51]. Most algorithms which perform pose estimation require the segmentation of humans as an essential introductory step [1, 31, 57]. 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 [62]. 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 [23], appearance [58], or edge and intensity [60]. 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.* [40]. 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 3.13(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 [40, 26, 21, 78]. One of the earliest methods for combining MRFs with a shape prior was proposed by Huang *et al.* [26]. They incrementally found the MAP solution of an extended MRF[9] integrated with a probabilistic deformable model. They were able to obtain a refined estimate of the object

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

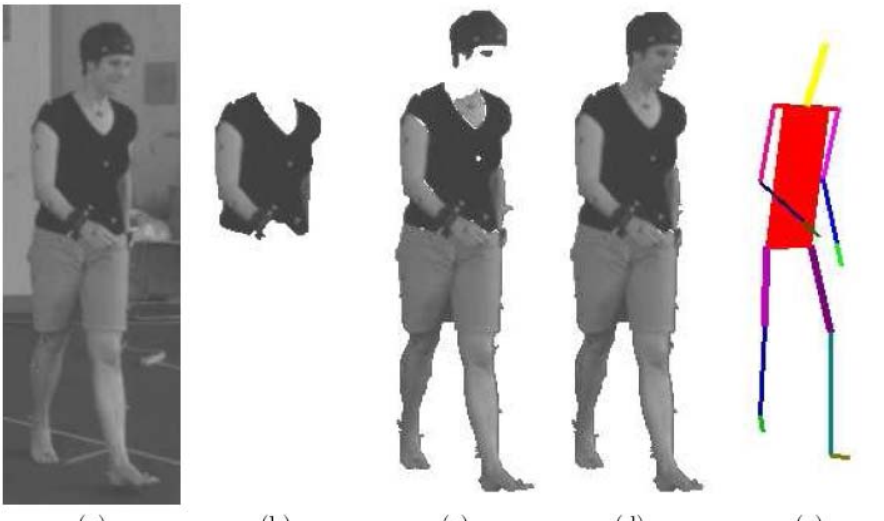


Fig. 3.13 Improving segmentation results by incorporating more information in the CRF. (a) Original image. (b) The segmentation obtained corresponding to the MAP solution of a CRF consisting of colour likelihood and contrast terms as described in [7]. We give the exact formulation of this CRF in Section 3.6.1. (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 3.7.1. (d) Segmentation obtained after incorporating a 'pose-specific' shape prior in the CRF as explained in Section 3.7.1. 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 3.7.2). 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).

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.* [40]. 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 [78] 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 [40] 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.

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 [40] these methods learn the prior using a number of training shapes. Leventon *et al.* [45] 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.* [13] have used kernel density estimation and intrinsic alignment to embed more complex shape distributions. Compared to [40] and [78] 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.* [57] 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 [46, 18, 51]. They then use efficient inference algorithms to perform exact inference in such models. In their recent work, Lan and Huttenlocher [43] 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 [40] 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.* [40], 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.

3.7.1 Pose Specific CRF for Image Segmentation

The CRF framework for image segmentation described in Section 3.6.1 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 3.13(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 [7] 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 [40] 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 3.13(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 3.13(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 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 3.14. 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 3.6.1) with potentials based on the shape of the object to be segmented, and appearances of individual pixels.

Modeling Pixel Intensities by Gaussian Mixture Models

The CRF defined in Section 3.6.1 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 [62] 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 3.13(c) shows the segmentation result obtained after incorporating this information in our CRF formulation.

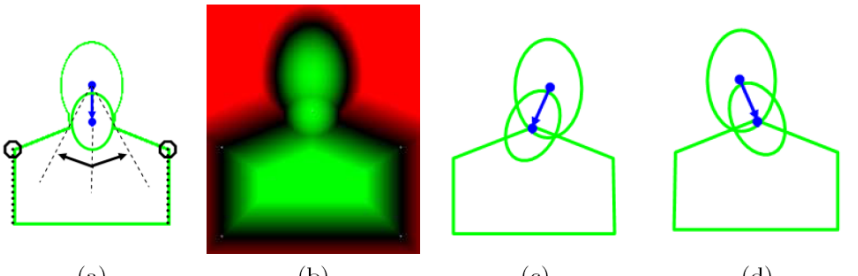


Fig. 3.14 The human upper body model. (a) The human upper body model parameterized by 6 parameters encoding the x and y location of the two shoulders, the length of the neck, and the angle of the neck with respect to the vertical. (b) The shape prior generated using the model. Pixels more likely to belong to the foreground/background are green/red. (c) and (d) The model rendered in two poses.

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.* [40], 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:

$$\Psi_3(\mathbf{x},\omega) = \sum_i (\phi(\mathbf{D}|x_i) + \phi(x_i|\omega) + \sum_j (\phi(\mathbf{D}|x_i,x_j) + \psi(x_i,x_j))) \tag{3.24}$$

with posterior $p(\mathbf{x},\omega|\mathbf{D}) = \frac{1}{Z_3}\exp(-\Psi_3(\mathbf{x},\omega))$. Here $\omega \in \mathbb{R}_p$ 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 3.15(e). This is a distance transform generated from the stick-man model silhouette using the fast implementation of Felzenszwalb and Huttenlocher [17].

The function $\phi(x_i|\omega)$ 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: $\phi(x_i|\omega) = -\log p(x_i|\omega)$. We follow the formulation of [40] and define $p(x_i|\omega)$ as

$$p(x_i = \text{figure}|\omega) = 1 - p(x_i = \text{ground}|\omega) = \frac{1}{1+\exp(\mu * (d(i,\omega) - d_r))}, \tag{3.25}$$

where $d(i,\omega)$ is the distance of a pixel i from the shape defined by ω (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.

Inference in the CRF Using Graph Cuts

Recall from Section 3.1 that energy functions like the one defined in (3.24) can be solved using graph cuts if they are *sub-modular* [38]. A function $f : \{0,1\}^n \to \mathbb{R}$ is submodular if and only if all its projections on 2 variables ($f^p : \{0,1\}^2 \to \mathbb{R}$) satisfy:

$$f^p(0,0) + f^p(1,1) \leq f^p(0,1) + f^p(1,0). \tag{3.26}$$

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 $\mathbf{x}^* = \min_{\mathbf{x}} \Psi_3(\mathbf{x},\omega)$ using a single graph cut. The labels of the latent variable in this configuration give the segmentation solution.

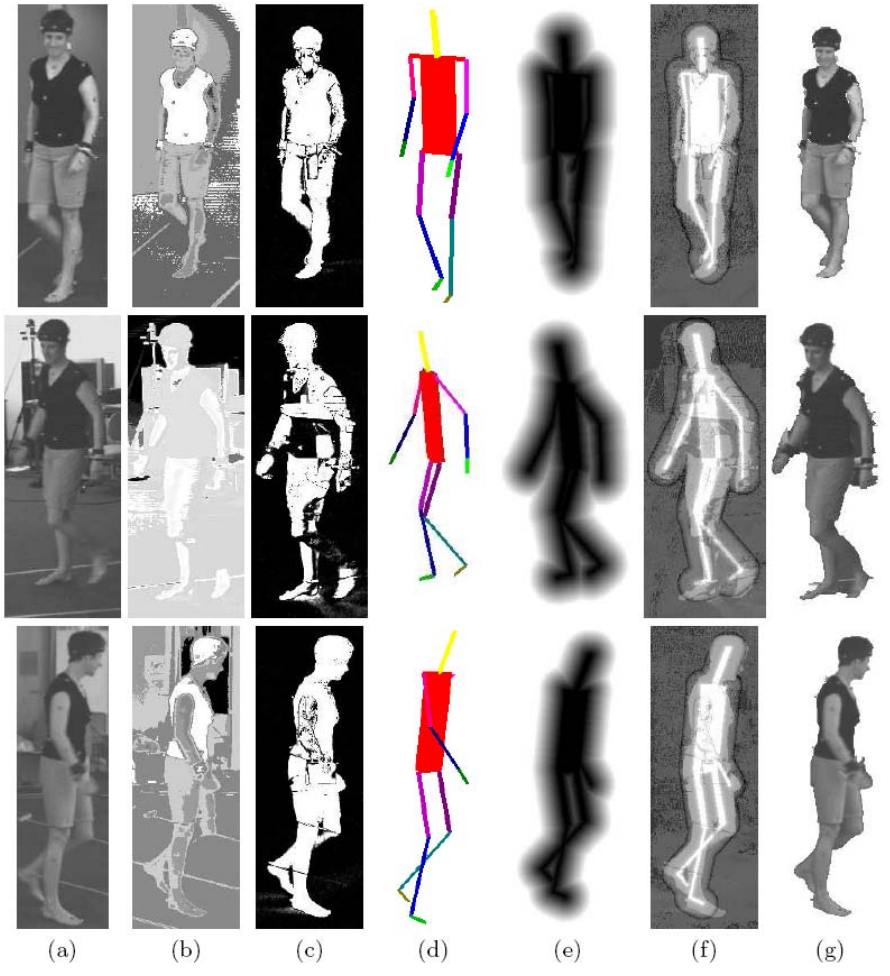


Fig. 3.15 Different terms of our pose specific CRF. (a) Original image. (b) The ratios of the likelihoods of pixels being labelled foreground/background ($\phi(\mathbf{D}|\mathbf{x}_i = \text{'fg'}) - \phi(\mathbf{D}|\mathbf{x}_i = \text{'bg'})$). These values are derived from the colour intensity histograms. (c) The segmentation results obtained by using the GMM models of pixel intensities. (d) The stickman in the optimal pose (see Sections 3.7.1 and 3.7.2). (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 3.6.1,3.7.1 and 3.7.1)) $\Psi_3(x_i = \text{'fg'}, \omega) - \Psi_3(x_i = \text{'bg'}, \omega)$. (g) The segmentation result obtained from our algorithm which is the MAP solution of the energy Ψ_3 of the pose-specific CRF.

3.7.2 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 [63] 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 (3.24). Specifically, we solve the problem:

$$\omega_{\mathbf{opt}} = \arg\min_{\omega,\mathbf{x}} \Psi_3(\mathbf{x},\omega)). \tag{3.27}$$

The minimization problem defined above contains both discrete ($\mathbf{x} \in \{0,1\}^n$) and continuous ($\omega \in \mathbb{R}_P$) valued variables and thus is a mixed integer programming problem. The large number of variables involved in the energy function $\Psi_3(\mathbf{x},\omega))$ make it especially challenging to minimize. To solve the minimization problem (3.27), we decompose it as: $\omega_{\mathbf{opt}} = \arg\min_{\omega} \mathcal{F}(\omega)$, where

$$\mathcal{F}(\omega) = \min_{\mathbf{x}} \Psi_3(\mathbf{x},\omega)). \tag{3.28}$$

For any value of ω, the function $\Psi_3(\mathbf{x},\omega))$ 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 $\mathcal{F}(\omega)$.

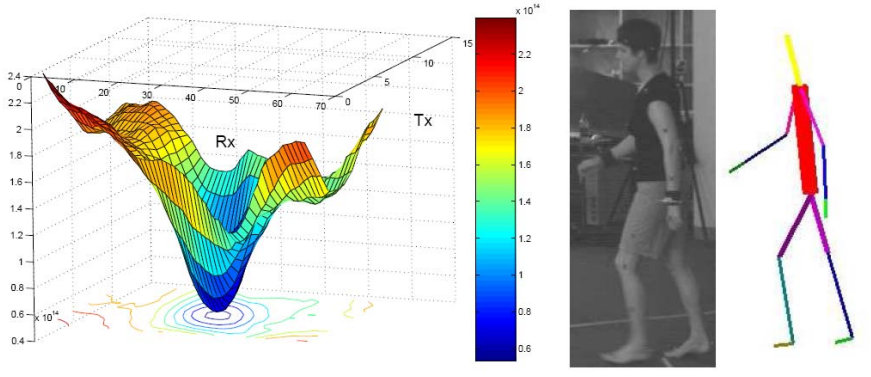
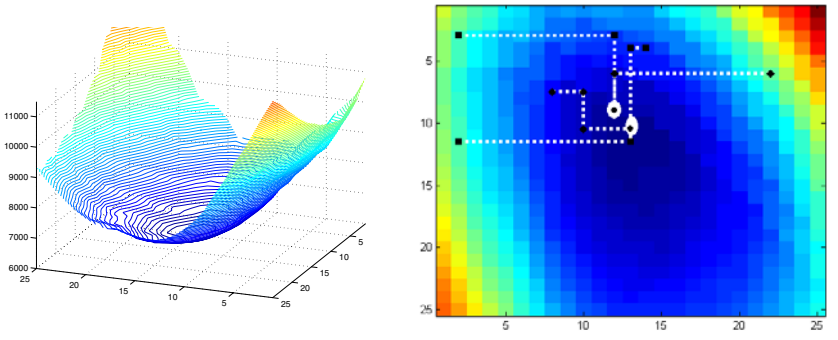


Fig. 3.16 Inferring the optimal pose. a) The values of $\min_{\mathbf{x}} \Psi_3(\mathbf{x},\omega)$ obtained by varying the global translation and rotation of the shape prior in the x-axis. b) Original image. c) The pose obtained corresponding to the global minimum of the energy.



(a) (b)

Fig. 3.17 Optimizing the pose parameters. (a) The values of $\min_{\mathbf{x}} \Psi_3(\mathbf{x}, \omega)$ obtained by varying the rotation and length parameters of the neck. (b) The image shows five runs of the Powell minimization algorithm [49] which are started from different initial solutions.

We will now explain how we minimize $\mathcal{F}(\omega)$ to get the optimal value of the pose parameters. Figure 3.16 shows how the function $\mathcal{F}(\omega)$ 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 ω, it can be reliably optimized using any standard optimization algorithm like gradient descent. In our experiments, we used the Powell minimization [49] algorithm for optimization.

Figure 3.17(a) shows how the function $\mathcal{F}(\omega)$ changes with changes to the neck angle and length parameters of the upper body model shown in Figure 3.14. 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 3.17(b)).

Failure Modes

It can be seen that the function $\mathcal{F}(\omega)$ is not unimodal over the whole domain and contains local minima. This multi-modality of $\mathcal{F}(\omega)$ 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.

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 3.7.5 which was used in our recent work on object detection and segmentation [52].

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:

$$\omega_{\mathbf{opt}} = \arg\min_{\omega}\Big(\sum_{\text{Views}} \min_{\mathbf{x}}(\Psi_3(\mathbf{x},\omega))\Big). \tag{3.29}$$

The framework is illustrated in Figure 3.18.

Dynamic Energy Minimization Using Graph Cuts

As explained earlier global minima of energies like the one defined in (3.24) can be found by graph cuts [38]. 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 slow since we need to compute the global optimum of $\Psi_3(\mathbf{x},\omega)$ with respect to $\mathbf{x}$ multiple number times for different values of ω. The graph cut computation can be made significantly faster by using the dynamic graph cut algorithm proposed in Section 3.3. This algorithm works by using the solution of the previous

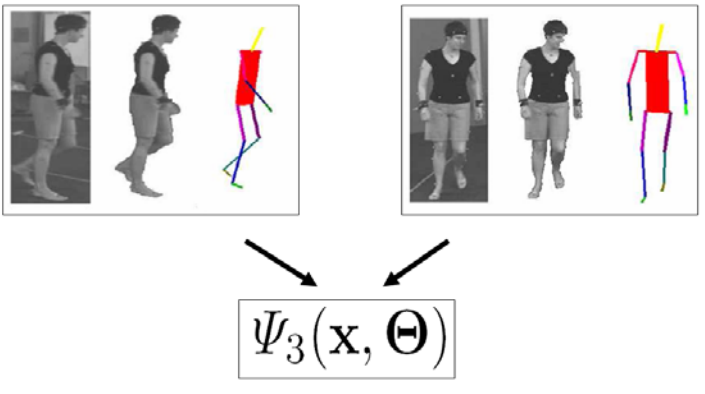


Fig. 3.18 Resolving ambiguity in pose using multiple views. The Figure shows how information from different views of the human can be integrated in a single energy function, which can be used to find the true pose of the human subject.

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.

3.7.3 Experiments

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

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 3.19 shows how integrating more information in the CRF improves the segmentation results. Quantitative results for the segmentation problem are shown in Table 3.1.

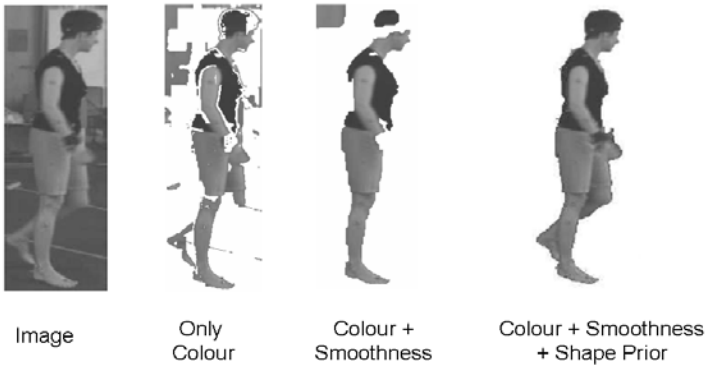


Fig. 3.19 Results showing the effect of incorporating a shape prior on the segmentation results. The first image is the original image to be segmented. The second, third and fourth images show the segmentation results obtained using colour, colour + smoothness prior and colour + smoothness + shape prior respectively.

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.

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%

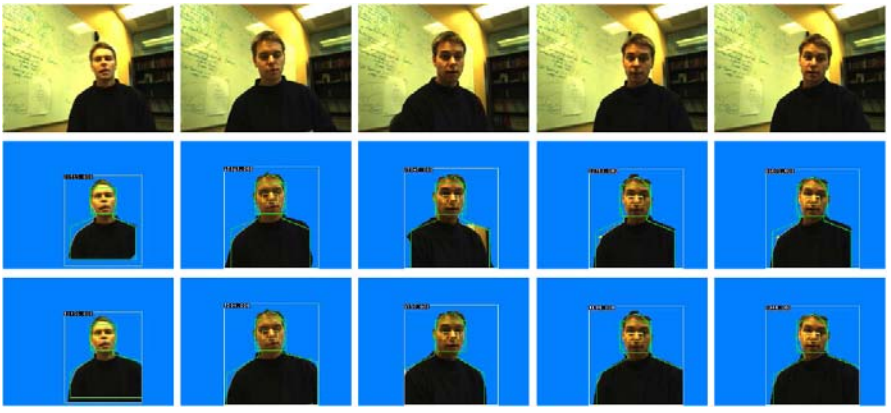


Fig. 3.20 Segmentation results using the 2D upper body model. The first row shows some frames from the video sequence. The second row shows the initial values of the pose parameters of the model and the resulting segmentations. The last row shows the final pose estimate and segmentation obtained using our method.

In order to demonstrate the performance of our method, we compare our segmentation results with those obtained using the method proposed in [62]. It can be seen from the results in Figure 3.21 that the segmentations obtained using the method of [62] are not accurate: They contain "speckles" and often 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 3.14) also produced good results. For these experiments, video sequences from the Microsoft Research bilayer video segmentation dataset [36] were used. The results of our method are shown in Figure 3.20.

Segmentation and Pose Estimation

Figures 3.22 and 3.23 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[10]. As shown in these Figures, the pose estimates match the original images accurately. In Figures 3.22 and 3.23, it should be noted that the appearance models of the foreground and background are quite similar: for instance, in Figure 3.23, the clothes of the subject are black in colour and the floor in the background is

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

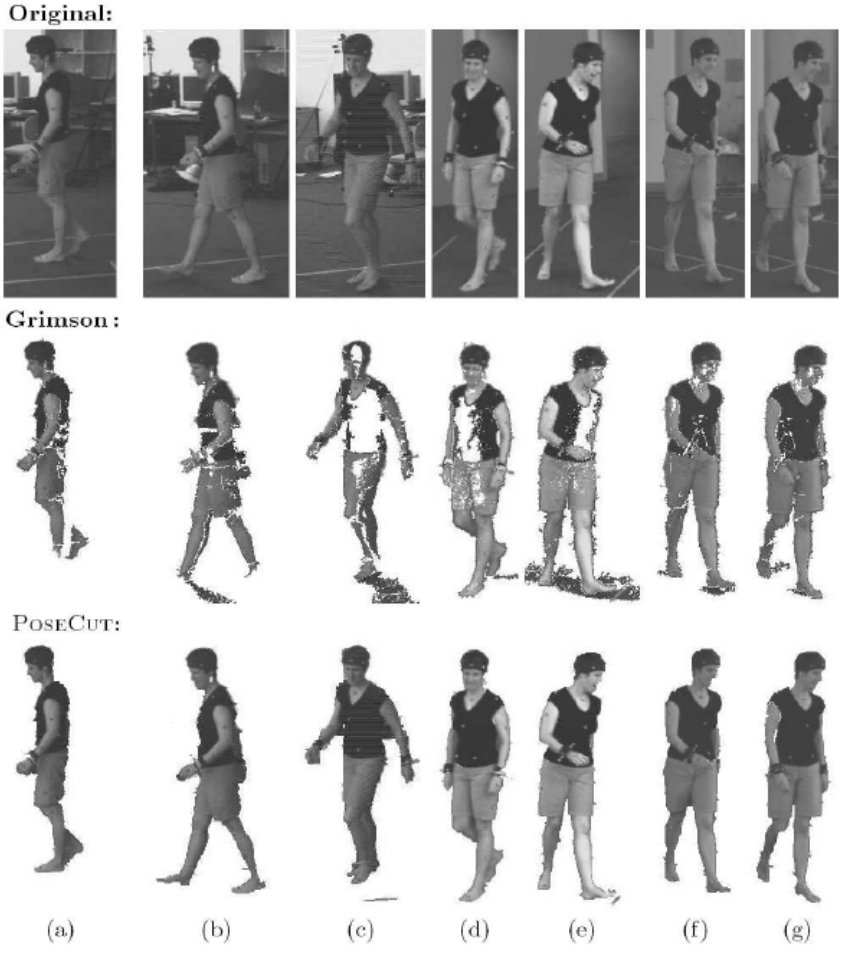


Fig. 3.21 Segmentation results obtained by Grimson-Stauffer [62] and POSECUT.

rather dark. The accuracy of the segmentation obtained in such challenging conditions demonstrates the robustness of POSECUT. An interesting fact to observe in Figure 3.22 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.

3.7.4 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

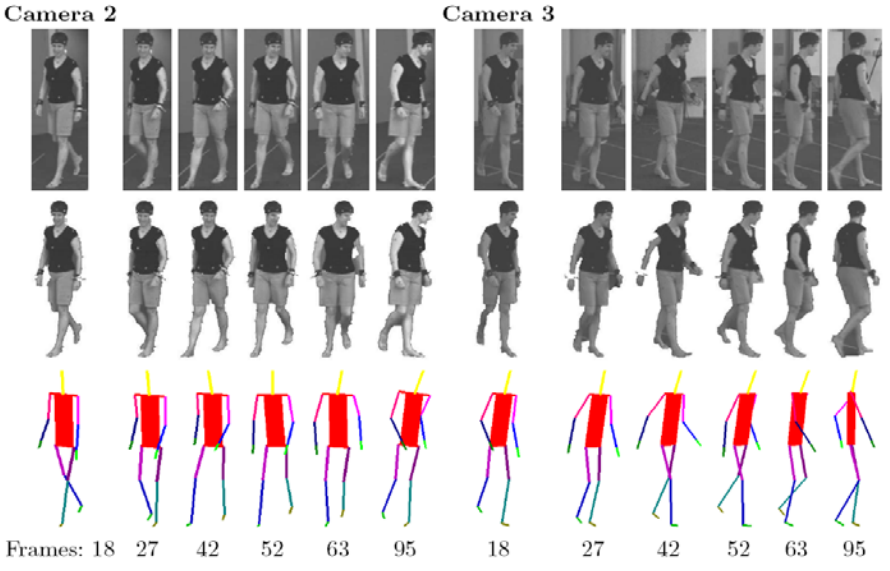


Fig. 3.22 Segmentation (middle) and pose estimation (bottom) results from POSECUT.

methods have been proposed for its solution. These range from methods such as shape from silhouettes [65] and space carving [41] to image based methods [55]. 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 3.24) 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 using prior knowledge. Researchers have long understood this fact and weak priors such as surface smoothness have been used in a number of methods [37,61,71]. 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 [64].

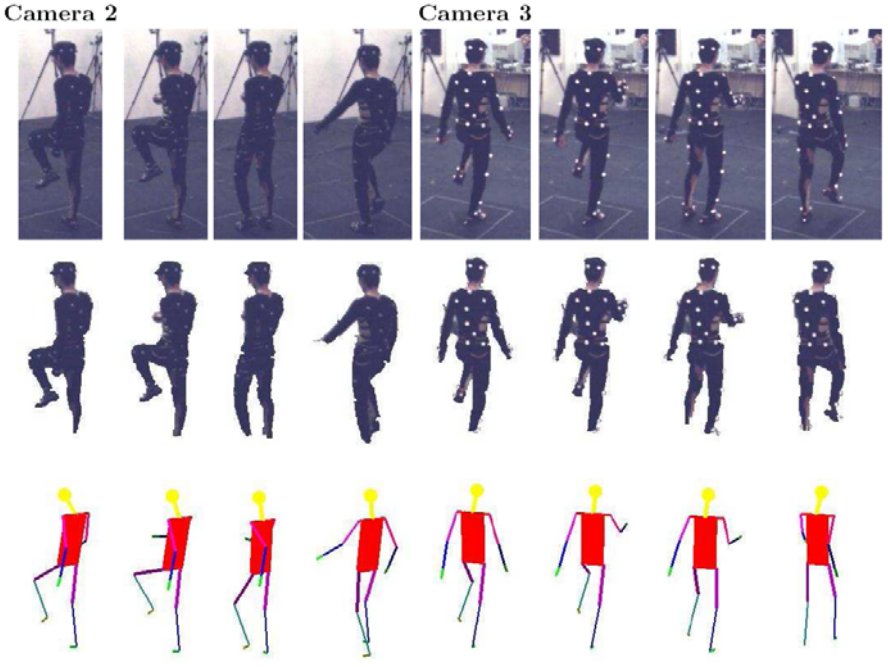


Fig. 3.23 Segmentation (middle row) and pose estimation (bottom row) results obtained using POSECUT. Observe that although the foreground and background appearances are similar, our algorithm is able to obtain good segmentations.

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 [4]. 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 3.25(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 3.25(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)).

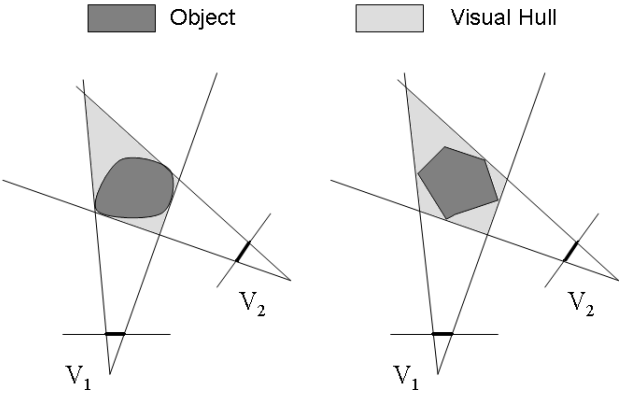


Fig. 3.24 Ambiguity in object reconstruction due to few views. The Figure shows how two completely different objects can have the same visual hull. Further, if both objects have the same colour, the photo hull and their projections on multiple viewpoints would also be the same.

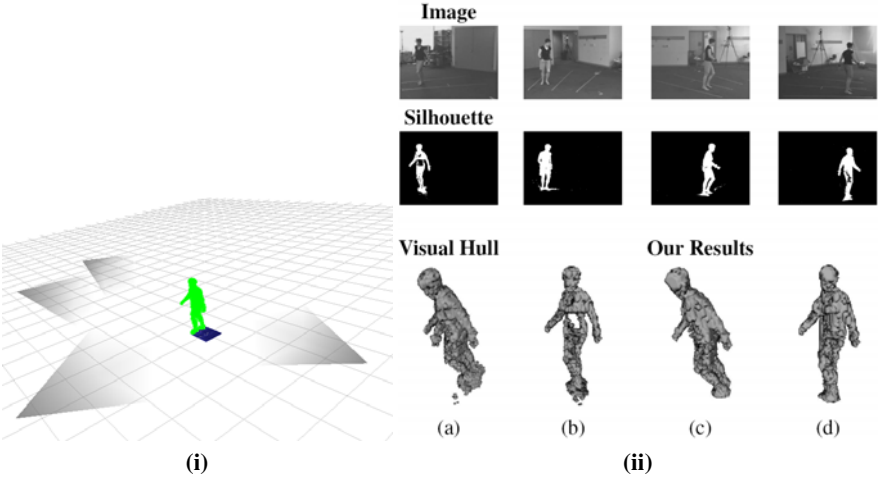


Fig. 3.25 i) Camera Positions and Reconstruction. The Figure shows the position and orientations of the four cameras which were used to obtain the images which constituted the dataset for our first experiment. We also see the reconstruction result generated by our method. ii) 3D Object Reconstruction 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)).

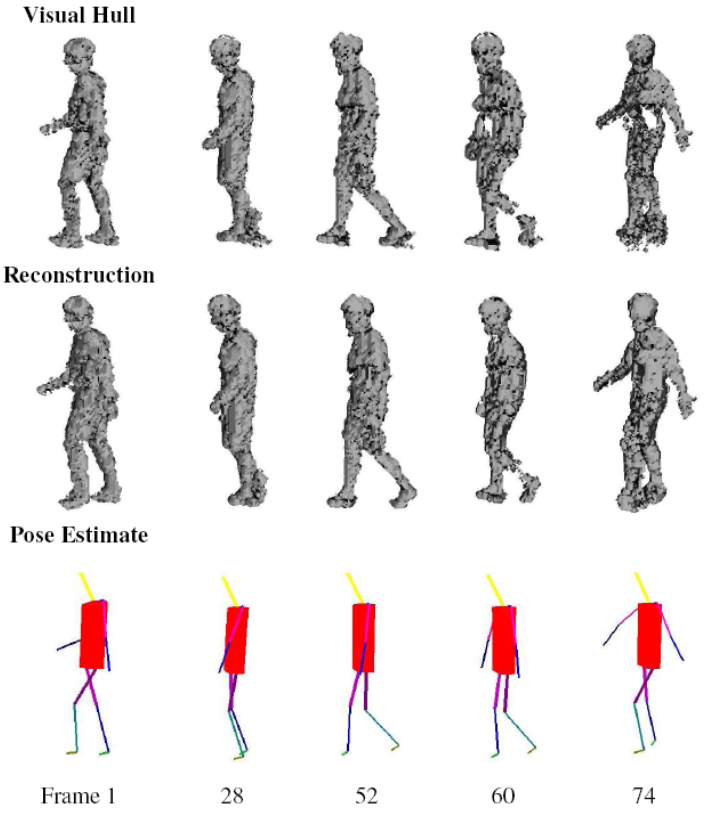


Fig. 3.26 Pose inference and 3D object reconstruction results. The data used in this experiment is taken from [4]. It consists of 4 views of a human subject walking in a circular path. Middle row: Reconstruction result. Bottom row: Pose estimate. Observe that we are able to get excellent reconstruction and pose estimation results even when the visual hull contains large errors (as seen in frame 60 and 74).

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 are shown in Figure 3.26.

3.7.5 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 [10]. In our work on real time face detection and segmentation [52], we showed how an off the shelf face-detector such as the one described in [70] can be coupled with a CRF to get accurate segmentation and improved face detection results in real time.

Fig. 3.27 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.

The object (face) localization estimate (obtained from any generic face detector) was incorporated in a discriminative CRF framework to obtain robust and accurate face segmentation results as shown in Figure 3.27. 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 detection accuracy. The procedure is illustrated in Figure 3.28. A similar strategy was recently used in [50].

3.7.6 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

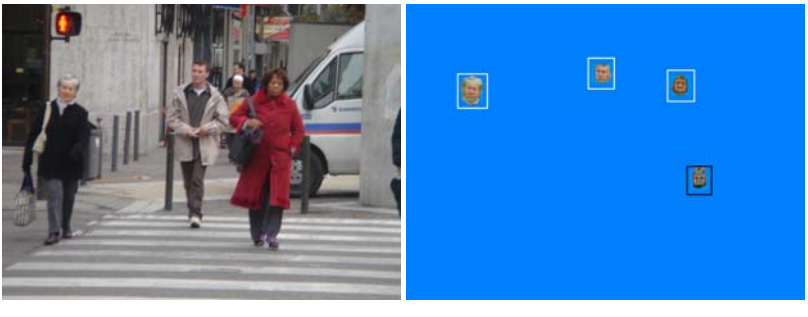


Fig. 3.28 Pruning false object detections. The Figure shows an image from the INRIA pedestrian data set. After running our algorithm, we obtain four face segmentations, one of which (the one bounded by a black square) is a false detection. The energy-per-pixel values obtained for the true detections 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.

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 [64].

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 [68, 58]. 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 [59]. Recently, Urtasun *et al.* [68] 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.

3.8 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.

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 Loopy Belief Propagation (LBP) [48], Generalized Belief Propagation (GBP) [77], and Tree-reweighted message passing (TRW) [35, 72] 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[11] 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. 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 3.8.1) 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[12].

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. We present an algorithm based on dynamic graph cuts [33] 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.

3.8.1 Preliminaries

As explained in Section 3.1, 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:

$$\psi(\theta) = \min_{\mathbf{x} \in \mathbf{L}} E(\mathbf{x}|\theta). \tag{3.30}$$

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

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 [35], we define the min-marginal energies $\psi_{v;j}, \psi_{uv;ij}$ as:

[11] We will explain the relation between max-marginal probabilities and min-marginal energies later in Section 3.8.1. 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).

[12] 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 $f(.)$.

$$\psi_{v;j}(\theta) = \min_{\mathbf{x}\in\mathbf{L},x_v=j} E(\mathbf{x}|\theta), \quad \text{and} \tag{3.31}$$
$$\psi_{uv;ij}(\theta) = \min_{\mathbf{x}\in\mathbf{L},x_u=i,x_v=j} E(\mathbf{x}|\theta).$$

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.

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 $p(\mathbf{x}|\mathbf{D})$, 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:

$$\mu_{v;j} = \max_{\mathbf{x}\in\mathbf{L};x_v=j} p(\mathbf{x}|\mathbf{D}) \tag{3.32}$$

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:

$$\sigma_{v;j} = \frac{\max_{\mathbf{x}\in\mathbf{L},x_v=j} p(\mathbf{x}|\mathbf{D})}{\sum_{k\in\mathcal{L}} \max_{\mathbf{x}\in\mathbf{L},x_v=k} p(\mathbf{x}|\mathbf{D})} = \frac{\mu_{v;j}}{\sum_{k\in\mathcal{L}} \mu_{v;k}} \tag{3.33}$$

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 (3.3) that the energy and probability of a labelling are related as:

$$E(\mathbf{x}) = -\log p(\mathbf{x}|\mathbf{D}) - \text{const} \tag{3.34}$$

Substituting the value of $p(\mathbf{x}|\mathbf{D})$ from equation (3.34) in equation (3.32), we get

$$\mu_{v;j} = \max_{\mathbf{x}\in\mathbf{L};x_v=j} \left(\exp\left(-E(\mathbf{x}|\theta) - \text{const}\right)\right) \tag{3.35}$$
$$= \frac{1}{Z}\exp\left(-\min_{\mathbf{x}\in\mathcal{X};x_v=j} E(\mathbf{x}|\theta)\right), \tag{3.36}$$

where Z is the partition function. Combining this with equation (3.31a), we get

$$\mu_{v;j} = \frac{1}{Z}\exp\left(-\psi_{v;j}(\theta)\right). \tag{3.37}$$

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:

$$\sigma_{v;ob} = \frac{\mu_{v;ob}}{\mu_{v;ob} + \mu_{v;bg}} = \frac{\frac{1}{Z}\exp\left(-\psi_{v;ob}(\theta)\right)}{\frac{1}{Z}\exp\left(-\psi_{v;ob}(\theta)\right) + \frac{1}{Z}\exp\left(-\psi_{v;bg}(\theta)\right)}, \tag{3.38}$$

$$\text{or} \quad \sigma_{v;ob} = \frac{\exp\left(-\psi_{v;ob}(\theta)\right)}{\exp\left(-\psi_{v;ob}(\theta)\right) + \exp\left(-\psi_{v;bg}(\theta)\right)} \tag{3.39}$$

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.

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 (3.6) or mass balance constraint (3.7) 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:

$$0 \leq f_{ij} \leq c_{ij} \quad \forall (i,j) \in E, \text{ and} \tag{3.40}$$

$$\sum_{i \in N(j) \setminus \{s,t\}} (f_{ji} - f_{ij}) = f_{sj} - f_{jt} \quad \forall j \in V \setminus \{s,t,v\} \tag{3.41}$$

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

$$\mathbf{f} = \{f_{ij},\ \forall (i,j) \in E\}. \tag{3.42}$$

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

$$f_v^t = \max_{\mathbf{f}} \sum_{i \in N(v)} f_{vi} - f_{iv} \tag{3.43}$$

subject to constraints (3.40) and (3.41).

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 3.30. The flow potentials of a particular graph node are shown in figure 3.29. 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

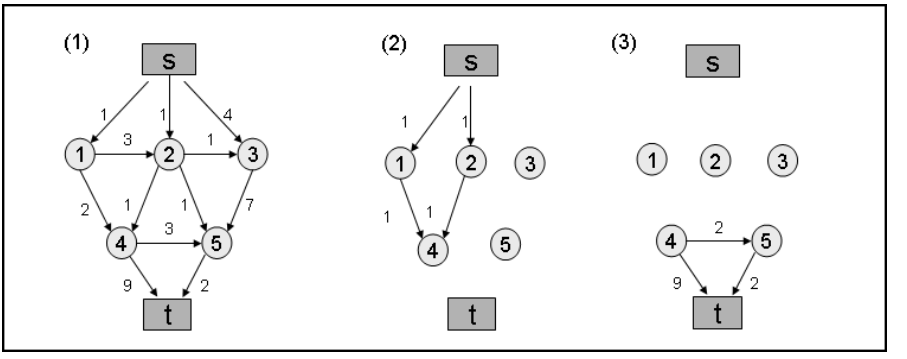


Fig. 3.29 Flow potentials of graph nodes. The figure shows a directed graph having seven nodes, two of which are the terminal nodes, the source s and the sink t. The number associated with each directed edge in this graph is a capacity which tells us the maximum amount of flow that can be passed through 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 $f_4^t = 11$.

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.

3.8.2 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:

$$f_{\text{total}} = \sum_{i \in N(s)} (f_{si} - f_{is}) = \sum_{i \in N(t)} (f_{it} - f_{ti}). \tag{3.44}$$

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 (3.30) and total flow equation (3.44) for a graph in which maxflow has been achieved i.e. $f_{\text{total}} = f_{\max}$, we obtain:

$$\psi(\theta) = \min_{\mathbf{x} \in \mathbf{L}} E(\mathbf{x}|\theta) = f_{\max} = \sum_{i \in N(s)} (f_{si} - f_{is}). \tag{3.45}$$

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 $\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathbf{L}} E(\mathbf{x}|\theta)$.

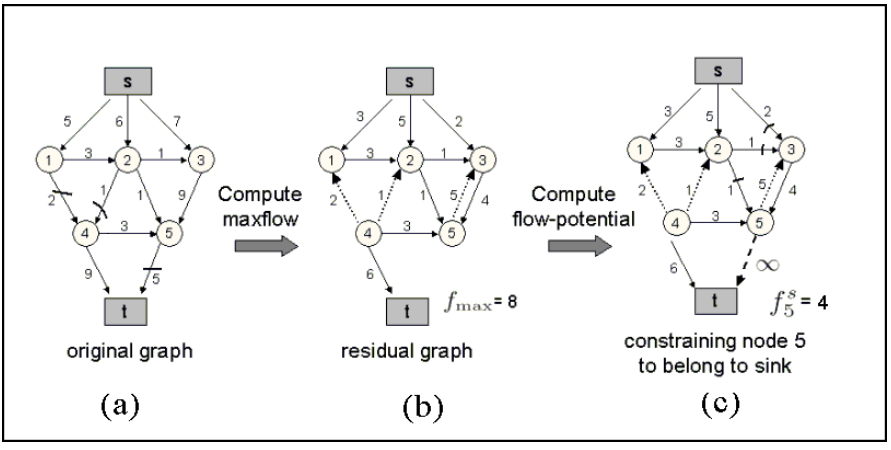


Fig. 3.30 Computing min-marginals using graph cuts. In (a) we see the graph representing the original energy function. This is used to compute the minimum value of the energy $\psi(\theta)$ which is equal to the max-flow $f_{\max} = 8$. The residual graph obtained after the computation of max-flow is 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.

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.

$$E(\mathbf{x}|\theta) = \psi(\theta) \tag{3.46}$$

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' obtained by constraining the value of x_v to b as:

$$\psi_{v;b}(\theta) = \min_{\mathbf{x} \in \mathcal{L}^n, x_v = b} E(\mathbf{x}|\theta) \tag{3.47}$$

$$\text{or, } \psi_{v;b}(\theta) = \min_{(\mathbf{x}-x_v) \in \mathcal{L}^{n-1}} E(x_1, .., x_{v-1}, b, x_{v+1}..x_n|\theta). \tag{3.48}$$

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.

3.8.3 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 3.1. *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.*

$$\psi_{v;j}(\theta) = \min_{x \in \mathbf{L}, x_v = j} E(\mathbf{x}|\theta) = \psi(\theta) + f_v^{T(j)} = f_{\max} + f_v^{T(j)} \tag{3.49}$$

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[13].

We already know from (3.48) 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[14]. In Section 3.8.5 we shall explain how to solve the new st-mincut problem efficiently using the dynamic graph cut algorithm proposed in the previous Section.

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 3.30.

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 in the

[13] 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.

[14] 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.

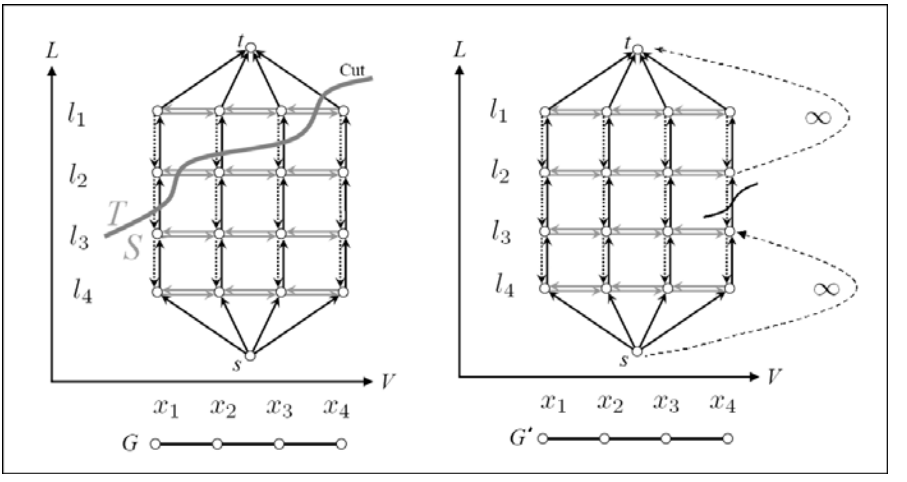


Fig. 3.31 Graph construction for projections of energy functions involving multiple labels. The first graph G shows the graph construction proposed by Ishikawa [27] for minimizing energy functions representing MRFs involving latent variables which can take more than 2 labels. All the label sets $\mathcal{L}$, $v \in V$ 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.

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.

3.8.4 Extension to Multiple Labels

Graph cuts can also be used to exactly optimize convex energy functions which involve variables taking multiple labels [27, 56]. 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 [27], 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 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 3.31. 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 [27] ap-

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

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:
   a. Obtain the energy projection E' corresponding to the latent variable assignment.
   b. Construct the graph G' to minimize E'.
   c. Use dynamic graph cut updates as given in [33] to make G_r consistent with G', thus obtaining the new graph $G_r^{'}$.
   d. Compute the min-marginal by minimizing E' using the dynamic (optimized) st-mincut algorithm on $G_r^{'}$.

plies 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.

3.8.5 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[15] as seen in figure 3.30, 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.

3.8.6 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

[15] 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.

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:

$$U = \psi(\theta) + \sum_{q \in \mathcal{Q}} (\psi_q(\theta) - \psi(\theta)). \tag{3.50}$$

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

$$\mathcal{Q} = \{(u;i) : u \in V, i \in \mathcal{L}\}, \text{and} \tag{3.51}$$

$$q_{max} = \max_{q \in \mathcal{Q}} (\psi_q(\theta) - \psi(\theta)), \tag{3.52}$$

the complexity of the above algorithm becomes $O((\psi(\theta) + nq_{max})T(n,m))$, 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 $O(m)$, we observe experimentally that using the dual search tree algorithm of [8], 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.

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.

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

3.8.7 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 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 3.32, 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.

Fig. 3.32 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.

Min-Marginals as a Confidence Measure

We had shown in Section 3.8.1 how min-marginals can be used to compute a confidence measure for any latent variable assignment in a MRF. Figure 3.32 shows the confidence values obtained for the MRF used for modeling the two label (foreground and background) image-segmentation problem as defined in [7]. 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 (or labellings) for latent variables in a Bayesian network [76].

Dawid [14] showed how min-marginals on junction trees can be computed. This method was later used by [47] to find the M most probable configurations of a probabilistic graphical network. The method of [14] 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. 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 [33] 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.

References

1. Agarwal, A., Triggs, B.: 3d human pose from silhouettes by relevance vector regression. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 882–888 (2004)
2. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows. Prentice Hall, Eaglewood Cliffs (1993)
3. Alahari, K., Kohli, P., Torr, P.H.S.: Reduce, reuse & recycle: Efficiently solving multi-label mrfs. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2008)
4. Bhatia, S., Sigal, L., Isard, M., Black, M.J.: 3d human limb detection using space carving and multi-view eigen models. In: Proceedings of the ANM Workshop (2004)
5. Blake, A., Rother, C., Brown, M., Perez, P., Torr, P.H.S.: Interactive image segmentation using an adaptive GMMRF model. In: Pajdla, T., Matas, J(G.) (eds.) ECCV 2004. LNCS, vol. 3021, pp. 428–441. Springer, Heidelberg (2004)
6. Boros, E., Hammer, P.L.: Pseudo-boolean optimization. Discrete Applied Mathematics 123(1-3), 155–225 (2002)
7. Boykov, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In: Proceedings of the International Conference on Computer Vision, pp. 105–112 (2001)
8. Boykov, Y., Kolmogorov, V.: An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. IEEE Transaction on Pattern Analysis and Machine Intelligence 26(9), 1124–1137 (2004)
9. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Transaction on Pattern Analysis and Machine Intelligence 23(11), 1222–1239 (2001)
10. Bray, M., Kohli, P., Torr, P.H.S.: Posecut: Simultaneous segmentation and 3d pose estimation of humans using dynamic graph-cuts. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 642–655. Springer, Heidelberg (2006)
11. Chiang, Y., Tamassia, R.: Dynamic algorithms in computational geometry. IEEE Special Issue on Computational Geometry 80, 362–381 (1992)
12. Cohen, R.F., Tamassia, R.: Dynamic expression trees and their applications. In: Proceedings of the Symposium on Discrete Algorithms, pp. 52–61 (1991)

13. Cremers, D., Osher, S., Soatto, S.: Kernel density estimation and intrinsic alignment for shape priors in level set segmentation. International Journal of Computer Vision 69(3), 335–351 (2006)
14. Dawid, P.: Applications of a general propagation algorithm for probabilistic expert systems. Statistics and Computing 2, 25–36 (1992)
15. Deutscher, J., Davison, A., Reid, I.: Automatic partitioning of high dimensional search spaces associated with articulated body motion capture. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 669–676 (2001)
16. Dinic, E.A.: Algorithm for solution of a problem of maximum flow in networks with power estimation. Soviet Math. Dokl. 11, 1277–1280 (1970)
17. Felzenszwalb, P.F., Huttenlocher, D.: Distance transforms of sampled functions. Technical Report TR2004-1963, Cornell University (2004)
18. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient matching of pictorial structures. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 2066–2073 (2000)
19. Flach, B.: Strukturelle bilderkennung. Technical report, Universit at Dresden (2002)
20. Ford, L.R., Fulkerson, D.R.: Flows in Networks. Princeton University Press, Princeton (1962)
21. Freedman, D., Zhang, T.: Interactive graph cut based segmentation with shape priors. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 755–762 (2005)
22. Gallo, G., Grigoriadis, M.D., Tarjan, R.E.: A fast parametric maximum flow algorithm and applications. SIAM Journal on Computing 18:18, 30–55 (1989)
23. Gavrila, D.M., Davis, L.S.: 3D model-based tracking of humans in action: a multi-view approach. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 73–80 (1996)
24. Greig, D., Porteous, B., Seheult, A.: Exact maximum a posteriori estimation for binary images. RoyalStat 51(2), 271–279 (1989)
25. Hengel, A., Dick, A., Thormhlen, T., Ward, B., Torr, P.H.S.: Rapid interactive modelling from video with graph cuts. In: Proceedings of Eurographics (2006)
26. Huang, R., Pavlovic, V., Metaxas, D.N.: A graphical model framework for coupling MRFs and deformable models. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 739–746 (2004)
27. Ishikawa, H.: Exact optimization for markov random fields with convex priors. IEEE Transaction on Pattern Analysis and Machine Intelligence 25, 1333–1336 (2003)
28. Ishikawa, H., Geiger, D.: Occlusions, discontinuities, and epipolar lines in stereo. In: Burkhardt, H.-J., Neumann, B. (eds.) ECCV 1998. LNCS, vol. 1406, pp. 232–248. Springer, Heidelberg (1998)
29. Ishikawa, H., Geiger, D.: Segmentation by grouping junctions. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 125–131 (1998)
30. Juan, O., Boykov, Y.: Active graph cuts. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 1023–1029 (2006)
31. Kehl, R., Bray, M., Van Gool, L.: Full body tracking from multiple views using stochastic sampling. In: Proceedings of the International onference on Computer Vision and Pattern Recognition, pp. 129–136 (2005)
32. Kohli, P., Kumar, M.P., Torr, P.H.S.: P^3 and beyond: Solving energies with higher order cliques. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2007)

33. Kohli, P., Torr, P.H.S.: Efficiently solving dynamic markov random fields using graph cuts. In: Proceedings of the International Conference on Computer Vision, pp. 922–929 (2005)
34. Kohli, P., Torr, P.H.S.: Measuring uncertainty in graph cut solutions: Efficiently computing min-marginal energies using dynamic graph cuts. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 30–43. Springer, Heidelberg (2006)
35. Kolmogorov, V.: Convergent tree-reweighted message passing for energy minimization. IEEE Transaction on Pattern Analysis and Machine Intelligence 28(10), 1568–1583 (2006)
36. Kolmogorov, V., Criminisi, A., Blake, A., Cross, G., Rother, C.: Bi-layer segmentation of binocular stereo video. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 407–414 (2005)
37. Kolmogorov, V., Zabih, R.: Multi-camera scene reconstruction via graph cuts. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2352, pp. 82–96. Springer, Heidelberg (2002)
38. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? IEEE Transaction on Pattern Analysis and Machine Intelligence 26(2), 147–159 (2004)
39. Komodakis, N.: A new framework for approximate labeling via graph cuts. In: Proceedings of the International Conference on Computer Vision, pp. 1018–1025 (2005)
40. Kumar, M.P., Torr, P.H.S., Zisserman, A.: Obj cut. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 18–25 (2005)
41. Kutulakos, K.N., Seitz, M.: A theory of shape by space carving. International Journal of Computer Vision 38(3) (2000)
42. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labelling sequence data. In: Proceedings of the International Conference on Machine Learning, pp. 282–289 (2001)
43. Lan, X., Huttenlocher, D.P.: Beyond trees: Common-factor models for 2d human pose recovery. In: Proceedings of the International Conference on Computer Vision, pp. 470–477 (2005)
44. Lempitsky, V.S., Roth, S., Rother, C.: Fusionflow: Discrete-continuous optimization for optical flow estimation. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2008)
45. Leventon, M.E., Grimson, W.E.L., Faugeras, O.D.: Statistical shape influence in geodesic active contours. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 1316–1323 (2000)
46. Mori, G., Ren, X., Efros, A.A., Malik, J.: Recovering human body configurations: Combining segmentation and recognition. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 326–333 (2004)
47. Nilsson, D.: An efficient algorithm for finding the m most probable configurations in bayesian networks. Statistics and Computing 8(2), 159–173 (1998)
48. Pearl, J.: Fusion, propagation, and structuring in belief networks. Artificial Intelligence 29(3), 241–288 (1986)
49. Press, W., Flannery, B., Teukolsky, S., Vetterling, W.: Numerical recipes in C. Cambridge Uni. Press, Cambridge (1988)
50. Ramanan, D.: Using segmentation to verify object hypotheses. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2007)
51. Ramanan, D., Forsyth, D.A.: Finding and tracking people from the bottom up. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 467–474 (2003)

52. Rihan, J., Kohli, P., Torr, P.H.S.: Objcut for face detection. In: Kalra, P.K., Peleg, S. (eds.) ICVGIP 2006. LNCS, vol. 4338, pp. 576–584. Springer, Heidelberg (2006)
53. Roth, S., Black, M.J.: Fields of experts: A framework for learning image priors. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 860–867 (2005)
54. Rother, C., Kolmogorov, V., Lempitsky, V., Szummer, M.: Optimizing binary MRFs via extended roof duality. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2007)
55. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International Journal of Computer Vision 47(1-3), 7–42 (2002)
56. Schlesinger, D., Flach, B.: Transforming an arbitrary minsum problem into a binary one. Technical Report TUD-FI06-01, Dresden University of Technology (2006)
57. Shakhnarovich, G., Viola, P., Darrell, T.J.: Fast pose estimation with parameter-sensitive hashing. In: Proceedings of the International Conference on Computer Vision, pp. 750–757 (2003)
58. Sidenbladh, H., Black, M.J., Fleet, D.J.: Stochastic tracking of 3D human figures using 2D image motion. In: Vernon, D. (ed.) ECCV 2000. LNCS, vol. 1843, pp. 702–718. Springer, Heidelberg (2000)
59. Sminchisescu, C., Jepson, A.D.: Generative modeling for continuous non-linearly embedded visual inference. In: Proceedings of the International Conference on Machine Learning (2004)
60. Sminchisescu, C., Triggs, B.: Covariance scaled sampling for monocular 3D body tracking. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 447–454 (2001)
61. Snow, D., Viola, P., Zabih, R.: Exact voxel occupancy with graph cuts. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 345–352 (2000)
62. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 246–252 (1999)
63. Stenger, B., Thayananthan, A., Torr, P.H.S., Cipolla, R.: Filtering using a tree-based estimator. In: Proceedings of the International Conference on Computer Vision, pp. 1063–1070 (2003)
64. Sun, Y., Kohli, P., Bray, M., Torr, P.H.S.: Using strong shape priors for stereo. In: Kalra, P.K., Peleg, S. (eds.) ICVGIP 2006. LNCS, vol. 4338, pp. 882–893. Springer, Heidelberg (2006)
65. Szeliski, R.: Rapid octree construction from image sequences. Computer Vision Graphics and Image Processing 58, 23–32 (1993)
66. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M.F., Rother, C.: A comparative study of energy minimization methods for markov random fields. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 16–29. Springer, Heidelberg (2006)
67. Thorup, M.: Fully-dynamic min-cut. In: Proceedings of the ACM Symposium on Theory of Computing, pp. 224–230 (2001)
68. Urtasun, R., Fleet, D.J., Hertzmann, A., Fua, P.: Priors for people tracking from small training sets. In: Proceedings of the International Conference on Computer Vision, pp. 403–410 (2005)
69. Veksler, O.: Graph cut based optimization for MRFs with truncated convex priors. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2007)

70. Viola, P.A., Jones, M.J.: Robust real-time face detection. International Journal of Computer Vision 57(2), 137–154 (2004)
71. Vogiatzis, G., Torr, P.H.S., Cipolla, R.: Multi-view stereo via volumetric graph-cuts. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 391–398 (2005)
72. Wainwright, M.J., Jaakkola, T., Willsky, A.S.: Map estimation via agreement on trees: message-passing and linear programming. IEEE Transactions on Information Theory 51(11), 3697–3717 (2005)
73. Wang, J., Bhat, P., Colburn, A., Agrawala, M., Cohen, M.F.: Interactive video cutout. ACM Transaction on Graphics 24(3), 585–594 (2005)
74. Woodford, O.J., Torr, P.H.S., Reid, I.D., Fitzgibbon, A.W.: Global stereo reconstruction under second order smoothness priors. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (2008)
75. Xiao, J., Shah, M.: Motion layer extraction in the presence of occlusion using graph cut. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 972–979 (2004)
76. Yanover, C., Weiss, Y.: Finding the m most probable configurations in arbitrary graphical models. In: Proceedings of the International Conference on Neural Information Processing Systems (2004)
77. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Generalized belief propagation. In: Proceedings of the International Conference on Neural Information Processing Systems, pp. 689–695 (2000)
78. Zhao L., Davis, L. S.: Closely coupled object detection and segmentation. In: Proceedings of the International Conference on Computer Vision, pp. 454–461 (2005)