Image Segmentation with Joint Regularization and Histogram Separation

David Nilsson

Master's thesis
2015:E12

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Centrum Scientiarum Mathematicarum

Abstract

In this thesis optimization methods for image segmentation are studied. The common theme of all the methods is that we have a histogram model for appearance terms that we optimize jointly with smoothness. Recently it has been shown that if one assumes a histogram model for appearance, it is possible to optimize an approximation of the energy using only one graph cut, by ignoring the non-submodular volumetric penalty term. We show how to include the volumetric term using the Fast trust region framework. Fast trust region is a recently proposed method that is able to handle a large class of non-submodular energies by solving a sequence of graph cut problems. A comparison of these methods shows that Fast trust region typically obtains a lower energy value and higher segmentation quality, at the cost of requiring multiple graph cuts.

Furthermore, we extend the simple histogram term to the multi-class setting and show that it is possible to optimize it with α-expansions. This is applied to the problems of stereo depth estimation and geometric model fitting.

Acknowledgements

The author would like to thank Carl Olsson for excellent supervision and for careful reading of this thesis.

Contents

Chapter 1

Introduction

One of the main problems in computer vision and image analysis is the problem of segmenting images. This means that you want to mark interesting and coherent regions in the image, for instance a human or a car. Many problems of computer vision contains the subproblem of finding good segmentations. In this report we focus on low level segmentation, which means optimization methods to assign pixels to foreground or background. We do not pursue further possible applications such as for instance recognition or scene understanding.

All the methods presented will be graph cut based methods. The recurring theme will be that we have histogram models for the color distributions and in the optimization functions there are terms that separates the histogram bins between the segments so that preferably all the pixels in one bin are assigned to a single segment. This is then optimized jointly with the segmentation. A more common approach is to either have a given color distribution beforehand or to alternate between estimating the color distribution and solving an optimization problem.

The optimization problems are solved in different ways depending on if we only want two segments, foreground and background, or if we want more than two segments. The binary case is in its simplest form possible to solve exactly, that is, find the global minimum of the objective function, while for the multi-class case we have to settle with approximative solutions. A number of different methods for the binary case will be presented: OneCut [11], GrabCut [9] and Fast trust region [3]. For the multi-class case we will use α-expansions [2].

For the binary case we will compare the three aforementioned methods based on energy values obtained. For the multi-class problem we will test the histogram model for stereo depth estimation and geometric model fitting. The geometric model fitting problem will be to find planar regions in RGB-D images, where we beside color content also know the depth of all the pixels.

Our main contributions are the following:

- The fast trust region method is used to add a volume constraint to the OneCut method, with resulting energies being slightly lower than for GrabCut and for OneCut without the volume constraint.
- The histogram term in OneCut is extended to the multi-class setting where it is optimized using α-expansions. This is then used for stereo depth estimation and geometric model fitting.

Chapter 2

Theory

Here the theory for both binary and multi-class segmentation will be presented in detail. We will start with the binary case and present the theoretical background and optimization methods and then introduce the three methods GrabCut, OneCut and fast trust region. For the multi-class case we start with the optimization method and then describe the extension of OneCut to the multi-class setting. Finally we present two applications of multi-class segmentation: stereo depth estimation and geometric model fitting.

2.1 Binary segmentation

Here we consider the problem of binary segmentation, that is, assigning all the pixels in the image to either background or foreground. The method of doing this is by optimizing an energy function. We will derive the general form of the energy function starting from Markov random fields. Then we will talk about methods to optimize the energy function. Finally three different methods used in image segmentation will be presented: GrabCut, OneCut and Fast trust region.

We will start by giving a somewhat informal description of the energy functions we are trying to minimize to get a segmentation. The general form of the segmentation energy is

$$E(x) = \sum_{p \in \mathcal{P}} U_p(x_p) + \sum_{(p,q) \in \mathcal{E}} V_{pq}(x_p, x_q). \tag{2.1}$$

The domain we are in is $x = (x_p)_{p \in \mathcal{P}}$ where $\mathcal{P}$ denotes the set of all pixels. x_p can take the values 0 and 1, where $x_p = 0$ means that p is assigned to the background and $x_p = 1$ means that p is assigned to the foreground. The first terms, $U_p(x_p)$ gives a cost for including the pixel in the foreground or background. These values are calculated by assuming a certain model for the foreground and background. One simple example is if we want the foreground

to be centred around μ_{fg} and the background at μ_{bg}. If I_p denotes the color of p, we can set $U_p(1) = \|I_p - \mu_{fg}\|^2$ and $U_p(0) = \|I_p - \mu_{bg}\|^2$. When optimizing (2.1), this will give a bias towards assigning pixels with intensity close to μ_{fg} to the foreground. If we only relied on the $U_p(x_p)$ terms we take no account of the smoothness of the segment. This is remedied by the terms in the second sum, $V_{pq}(x_p, x_q)$.

The smoothness terms in (2.1), $V_{pq}(x_p, x_q)$, are summed over all the pixel neighbours, denoted $\mathcal{E}$. The neighbours of a pixel are the pixels directly above, below, left and right of the pixel, see Figure 2.1. $V_{pq}(x_p, x_q)$ is a potential function for the edge. If we want a segmentation that has a smooth boundary, we want to penalize neighbours where one pixel is assigned to the foreground and the other to the background. This can be done by letting $V_{pq}(1, 0) = V_{pq}(0, 1) = 1$ and $V_{pq}(1, 1) = V_{pq}(0, 0) = 0$. Then we penalize neighbouring pixels that are assigned to different labels. This also has the effect of minimizing the boundary length, since pixels on the boundary contribute to the energy via the pairwise terms.

When combining the two terms in (2.1) the first term makes sure that we are assigning the right pixels to the foreground, measured by how well they fit the assumed model and the second term makes the solution smooth along the boundaries of the segmentation. Combining the two terms will ideally result in a smooth segmentation of the right region.

2.1.1 Markov random field model

In this section the energy function that is used in the rest of the text is derived in the context of Markov random fields largely following the derivation found in [8].

A Markov random field can be defined on a graph $G = (V, E)$, with nodes V and edges E, and for every node a corresponding random variable, x_p. We consider both binary variables $x_p \in \{0, 1\}$ and the multiclass case $x_p \in \{1, \ldots, N\}$, but the derivation is the same for both cases and no specification is made in this section. The Markov property can be expressed as

$$\Pr(x_p \mid x_{V \setminus p}) = \Pr(x_p \mid x_{\mathcal{N}(p)}), \tag{2.2}$$

that is, the random variable x_p is conditionally independent of all variables given its neighbours $\mathcal{N}(p)$. It can be showed, through Hammersley-Cliffords theorem, that the distribution of x factorizes over maximal cliques. A clique is defined as a sub graph where all nodes are directly connected. A maximal clique is a clique that is not a subset of another clique. If $\mathcal{C}$ denote the set of maximal cliques, we get

$$\Pr(x) = \prod_{c \in \mathcal{C}} \phi_c(x_c), \tag{2.3}$$

where $\phi_c(x_c)$ is called a potential function. To use this result for inference we

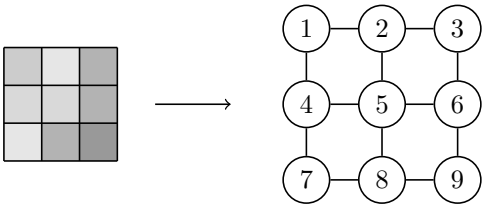

Figure 2.1: A grid graph used for image segmentation. Every pixel corresponds to a node and there are edges for neighbouring pixels. The cliques are exactly the set of neighbours.

condition on the observed values I_p and use Bayes formula.

$$\begin{aligned} \Pr(x \mid I) &\propto \Pr(I \mid x) \Pr(x) \\ &= \prod_p \Pr(I_p \mid x_p) \prod_{c \in \mathcal{C}} \phi_c(x_c). \end{aligned} \tag{2.4}$$

Instead of maximizing the likelihood we minimize the negative log-likelihood.

$$\begin{aligned} &- \ln \left(\prod_p \Pr(I_p \mid x_p) \prod_{c \in \mathcal{C}} \phi_c(x_c) \right) \\ &= \sum_p - \ln(\Pr(I_p \mid x_p)) + \sum_{c \in \mathcal{C}} V_c(x_c), \end{aligned} \tag{2.5}$$

where $V_c(x_c) = -\ln(\phi_c(x_c))$. In the special case of a grid graph, see Figure 2.1, the cliques are the neighbours, so the expression reduces to

$$\sum_p - \ln(\Pr(I_p \mid x_p)) + \sum_{(p,q) \in \mathcal{E}} V_{pq}(x_p, x_q), \tag{2.6}$$

where $\mathcal{E}$ is the set of edges. Maximizing the posterior likelihood $\Pr(x \mid I)$ is the same as minimizing the above expression. So what we are doing is maximizing the posterior likelihood given the model V_{pq} and $\Pr(I_p \mid x_p)$. That is,

$$\begin{aligned} \hat{x} &= \operatorname*{argmax}_x \Pr(x \mid I) \\ &= \operatorname*{argmin}_x \left(\sum_p - \ln(\Pr(I_p \mid x_p)) + \sum_{(p,q) \in \mathcal{E}} V_{pq}(x_p, x_q) \right). \end{aligned} \tag{2.7}$$

2.1.2 Optimizing MRF-MAP

We start by defining a graph cut and then showing how this is used to solve the minimization problem encountered in (2.7). Here we only solve it for binary $x_p \in \{0, 1\}$. In later sections we consider the multiclass problem, which is solved by a reduction to a sequence of binary optimization problems.

To define a graph cut, we have a graph $G = (V, E)$ with two special nodes s and t called the source and the sink respectively. Also, for every edge $e \in E$ there is

a weight $w_e \geq 0$. We partition the nodes into sets S and $T = V \setminus S$ such that $s \in S$ and $t \in T$ and S is connected. The value of the cut of this partition is the sum of the edge weights for edges that have one end in S and the other in T. Formally,

$$\text{cut}(S,T) = \sum_{\substack{e=(p,q)\in E \\ p\in S, q\in T}} w_e. \tag{2.8}$$

The min cut problem is the problem of finding the S that minimizes $\text{cut}(S, V\setminus S)$. It turns out that there is a polynomial time algorithm that solves this problem. By the max-flow min-cut theorem, the problem of minimizing the cut has the same optimal value as maximizing a flow from s to t in the graph with edge weights being the capacities. From a max-flow solution it is possible to obtain S and T. In Figure 2.2 is an illustration of max-flow and min-cut on a very simple graph. There are implementations of min-cut solvers available especially optimized for grid graphs typically encountered in computer vision. Throughout this report the implementation by Kolmogorov is used [1]. For a more thorough description of network flows, see any textbook on algorithms.

So far we have only talked about the min cut problem from a graph theoretical perspective. Now will will turn to an algebraic version of the min cut problem where it instead is formulated as a minimization problem of binary variables $x_p \in \{0,1\}$ denoting if $p \in \mathcal{P}$ is in S or T. If we consider the energy $E(x_1, x_2, x_3)$, with $x_k \in \{0,1\}$ of the cut associated with $S = \{k : x_k = 0\}$ and $T = \{k : x_k = 1\}$, we get for the graph in Figure 2.2 that

$$\begin{aligned}\text{cut}(S,T) = E(x_1,x_2,x_3) = 4x_1 + \bar{x}_1 + 3x_2 + 2\bar{x}_2 + x_3 + 5\bar{x}_3 \\ + x_1\bar{x}_2 + x_2\bar{x}_1 + x_2\bar{x}_3 + x_3\bar{x}_2,\end{aligned} \tag{2.9}$$

where $\bar{x}$ denotes negation, defined by $\bar{x} = 1 - x$. We can verify that for the min cut, see Figure 2.2, with $S = \{1,2\}$ and $T = \{3\}$, we get $E(0,0,1) = 5$ as expected and comparing to Figure 2.2 we see that the non-zero terms of $E(0,0,1)$ corresponds exactly to the edges in the cut.

In general considering algebraic expressions instead of a graph is convenient. For a function $E(x_1, \ldots, x_N)$ we define

$$E(x_1,\ldots,x_N) = \sum_{i=1}^{N} U_i(x_i) + \sum_{i,j=1}^{N} V_{ij}(x_i,x_j), \tag{2.10}$$

where U are called unary terms and V pairwise terms. Note the similarity with (2.7). In general the second sum is not over all i and j, rather only a small subset. As showed in [7], for the function E to be efficiently optimized by graph cut methods there are no restriction on the unary terms, but all pairwise terms must satisfy

$$V_{ij}(0,0) + V_{ij}(1,1) \leq V_{ij}(1,0) + V_{ij}(0,1). \tag{2.11}$$

This property is called submodularity and if it holds E can be efficiently optimized. Note that negative values are allowed and that $V_{ij}(1,1)$ and $V_{ij}(0,0)$ can be non-zero. This requires modification of the graph[1], since the weights

[1]For instance, if we added x_1x_2 to $E(x_1,x_2,x_3)$, how would we change the graph in Figure 2.2? It is not trivial at all.

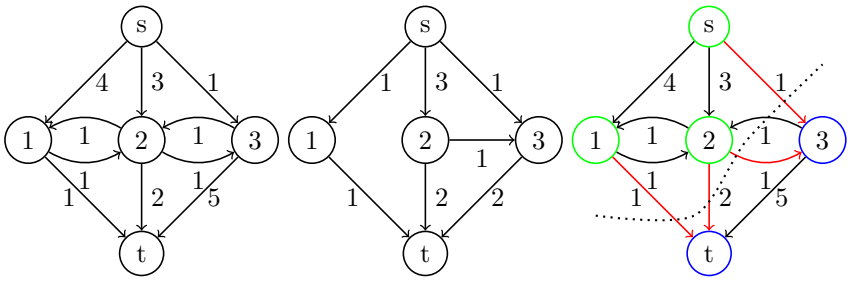

Figure 2.2: From left to right: A flow network, a maximum flow and a minimum cut. The final partition is that S is the green nodes and T is the blue nodes and the red edges are included in the cut, with the sum of the weights being 5, which can be verified is also the maximum flow.

must be non-negative, but it can be accounted for when submodularity holds. All the details are given in [7].

2.1.3 GrabCut

The idea behind GrabCut [9] is to start with an initial segmentation and iteratively improve it. From the previous segmentation color distributions of the foreground and background are estimated. This is then used to again solve the optimization problem, but with different unary terms. Algorithm 1 describes the method.

To estimate the color distribution, we first specify the model. We use a histogram model for the distributions. Define a probability distribution by counting frequencies for all bins θ_b. Each pixel is assigned to one bin, and we have $\cup_b \theta_b = \Omega$ and $\theta_i \cap \theta_j = \emptyset$ if $i \neq j$. Furthermore, we partition each bin into pixels in the foreground θ_b^S and pixels in the background $\theta_b^{\bar{S}}$. Assume that pixel p belongs to bin b, then $\Pr(I_p \mid S_p = 1) = |\theta_b^S|/|S|$, by counting elements in bins. This is then used for the unary terms, see (2.7), as

$$U_p(1) = -\ln(|\theta_b^S|/|S|), \quad U_p(0) = -\ln(|\theta_b^{\bar{S}}|/|\bar{S}|), \tag{2.12}$$

where $U_p(1)$ is the cost of assigning p to the foreground, $U_p(0)$ is the cost of assigning p to the background and p is in the bin with index b. We note that if a bin is empty for the foreground in one iteration, then pixels in the bin can never switch to foreground, regardless of the smoothness structure of the pixels. This is a very bad property and a simple fix is to set a lower bound on the probabilities. Note that basically any model can be used for GrabCut, but we limit ourselves to bin counting since this is what we will use later in the OneCut method.

Algorithm 1 GrabCut

1: $S =$ Initial segmentation
2: **while** S has not converged **do**
3: $\quad \theta =$ estimate distributions using S
4: $\quad S = \operatorname{argmin}_S E(S;\theta)$
5: **end while**

2.1.4 OneCut

Previously we arrived at the problem of minimizing

$$E(S) = \sum_{p\in\mathcal{P}} U_p(S_p) + \sum_{(p,q)\in\mathcal{E}} V_{pq}(S_p, S_q), \tag{2.13}$$

where $U_p(S_p) = -\ln \Pr(I_p \mid S_p)$. If we have an explicit model and given parameters for the background and foreground, we can calculate all these probabilities and solve the optimization problem. In the GrabCut method we iterate between estimating the distributions and solving an optimization problem. For OneCut [11] we consider a simple color distribution model and simultaneously optimize with respect to color distributions and smoothness using only one graph cut. Consider again the unary term in (2.13). We use histogram bin counting as described in the GrabCut section. Rewriting the unary terms in (2.13) using the bin notation and switching from summation over the pixels to summation over the bins, we obtain

$$\begin{aligned}
&\sum_{p\in\mathcal{P}} U_p(S_p) \\
=&\sum_{p\in\mathcal{P}} -\ln \Pr(I_p|S_p) \\
=&\sum_b \left[-|\theta_b^S| \ln\left(\frac{|\theta_b^S|}{|S|}\right) - |\theta_b^{\bar{S}}| \ln\left(\frac{|\theta_b^{\bar{S}}|}{|\bar{S}|}\right)\right] \\
&= \sum_b \left[|\theta_b^S| \ln|S| + |\theta_b^{\bar{S}}| \ln|\bar{S}| - |\theta_b^S| \ln|\theta_b^S| - |\theta_b^{\bar{S}}| \ln|\theta_b^{\bar{S}}| \right] \\
&= |S| \ln|S| + |\bar{S}| \ln|\bar{S}| - \sum_b \left(|\theta_b^S| \ln|\theta_b^S| + |\theta_b^{\bar{S}}| \ln|\theta_b^{\bar{S}}| \right) \\
&= h_\Omega(S) - \sum_b h_{\Omega_b}(S_b),
\end{aligned} \tag{2.14}$$

where $h_\Omega(S) = |S| \ln|S| + |\Omega \setminus S| \ln|\Omega \setminus S|$. To interpret the last expression in (2.14), consider the function $h_\Omega(S)$. In Figure 2.3 is a plot of how these terms depends on $|S|$. The first term is a global volume balancing term and the terms in the sum over bins separate the bins between foreground and background.

Now we can introduce the OneCut method. Instead of the expression in (2.14) we consider a simpler version by discarding $h_\Omega(S)$ and replacing the summation of $h_{\Omega_b}(S_b)$ with the simpler terms $\min\{|\theta_b^S|, |\theta_b^{\bar{S}}|\}$. The following function is

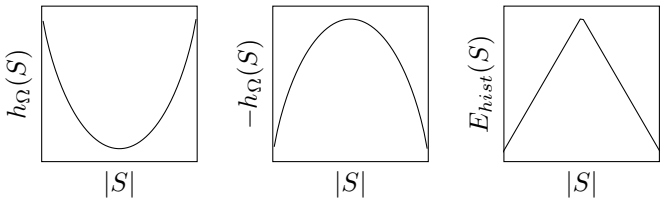

Figure 2.3: The volume balancing term, the histogram separation term and the $E_{hist}(S)$ term.

introduced

$$E_{hist}(S) = \sum_b \min\{|\theta_b^S|, |\theta_b^{\bar{S}}|\}. \tag{2.15}$$

The OneCut energy to minimize is

$$E(S) = E_{hist}(S) + E_{smoothness}(S). \tag{2.16}$$

The optimization of $E_{hist}(S)$ is most conveniently expressed using binary variables. Let x_p be binary variables being 1 if $p \in S$ and 0 otherwise. Introduce extra variables x_b for every bin. Then the term $E_{hist}(S)$ can be expressed as[2]

$$\begin{aligned} E_{hist}(S) &= \sum_b \min\{|\theta_b^S|, |\theta_b^{\bar{S}}|\} \\ &= \sum_b \min\{\sum_{p\in\theta_b} x_p, \sum_{p\in\theta_b} \bar{x}_p\} \\ &= \sum_b \left(\bar{x}_b \sum_{p\in\theta_b} x_p + x_b \sum_{p\in\theta_b} \bar{x}_p \right), \end{aligned} \tag{2.17}$$

where we note that the pairwise terms are submodular. Note that all x_b select the minimum of the two choices corresponding to $x_b = 1$ and $x_b = 0$ when optimized.

2.1.5 Fast trust region

We have so far introduced two different methods to solve (2.7) for a certain distribution model. GrabCut approximates a solution by iterating between estimating foreground and background distribution and solving (2.7) with specified distributions. For OneCut we used a simple distribution model and could solve the optimization problem using one graph cut. In this section the fast trust region method will be introduced. The main thing about fast trust region is that we are able to handle more difficult energies by considering Taylor expansions. Like GrabCut, it iteratively finds better solutions by refining the previous solution.

[2]Note that the third equality strictly speaking is wrong and we should write $\min_{x_b}$ before the parenthesis. We will not do this since the notation would become quite horrible later on. It is always assumed that we are minimizing the energy functions. This abuse of notation will be present at a number of occasions later on and is only pointed out here.

We consider an energy function of the form

$$E(S) = R(S) + Q(S), \tag{2.18}$$

where $R(S)$ is a regional constraint and $Q(S)$ is the normal smoothness terms, or more generally any term that can be optimized with graph cuts. Later we will use $R(S) = h_\Omega(S)$, as defined in (2.14). Since $R(S)$ in general is non-linear it can not be solved by the standard min-cut method. Instead of $R(S)$ we consider a Taylor expansion of $R(S)$ at the previous segmentation S_0. That is, instead of optimizing $E(S)$ we iteratively optimize an approximative energy

$$\tilde{E}(S) = U_0(S) + Q(S), \tag{2.19}$$

where $U_0(S)$ is a Taylor expansion of $R(S)$ at S_0. The details about how to compute the Taylor expansion can be found in [4] and will not be given here.

Since a Taylor expansion is a local approximation, we want solutions not to be too far away from the starting solution S_0, i.e. it should minimize $\tilde{E}(S)$ within a trust region $\|S - S_0\| \leq \delta$ where the approximation is sufficiently accurate. The solution proposed in [3] solves this problem by considering the Lagrangian

$$L_\lambda(S) = \tilde{E}(S) + \lambda \sum_p \phi_0(S_p; S_0), \tag{2.20}$$

where $\phi_0(S_p; S_0)$ is the signed distance function of S_0, equal to the distance from p to the boundary of S_0 if $S_p = 1$ and 0 if $S_p = 0$. In Figure 2.4 is an example of this function. It measures the distance to the boundary. If a point is inside S_0 the distance is negative and if a point is outside S_0 it is positive, forcing for large λ the minimizer of $L_\lambda(S)$ to be not too far away from the last segmentation S_0. As showed in [3], and intuitively by looking at Figure 2.4, it holds approximately that $\lambda \sim \frac{1}{d}$, so by adjusting λ we can adjust the size of the trust region.

Algorithm 2 describes the method. It iteratively updates the segmentation. It starts by computing the Taylor expansion around the current segmentation S. Then it solves the optimization problem within the trust region using $L_\lambda(S)$. The energy reductions are then computed, both the predicted ΔP and the actual ΔA. If the actual energy has decreased then the segmentation is updated. If the approximation is good, checked with $\Delta A / \Delta P \geq \tau$ then the trust region is expanded by decreasing λ. Otherwise the trust region is decreased.

The fast trust region framework will be used for the OneCut energy in (2.14). Here the regional term is $h_\Omega(S)$, which is not submodular. To use the fast trust region method we need the Taylor expansion of $h_\Omega(S)$ at S_0. This can be computed following the methodology in [4] with the result being, up to a constant

$$\ln\left(\frac{|S_0|}{|\Omega| - |S_0|}\right)|S|. \tag{2.21}$$

$\sqrt{5}$	$\sqrt{2}$	1	1
$\sqrt{2}$	1	0	0
1	0	-1	-1
1	0	-1	-2

Figure 2.4: Example of the signed distance function $\phi_0(S)$. Gray pixels are in the segmentation S_0 and white pixels are not in the segmentation. Including pixels far away from S_0 will give a high cost and not including pixels in the interior of S_0 will also give a high cost, giving a bias towards only changing close to the boundary.

Algorithm 2 Fast trust region

$S =$ Initial segmentation
while S has not converged **do**
    $U_0 =$ Taylor expansion of $R(S)$ at S
    $S^* = \operatorname{argmin}_S L_\lambda(S)$
    $\Delta P = \tilde{E}(S) - \tilde{E}(S^*)$
    $\Delta A = E(S) - E(S^*)$
    if $\Delta A \geq 0$ **then**
        $S = S^*$
    end if
    if $\Delta A / \Delta P \geq \tau$ **then**
        $\lambda = \lambda / \alpha$
    else
        $\lambda = \lambda \cdot \alpha$
    end if
end while

2.2 Multi-class segmentation

Previously we have written about segmenting images into two classes: foreground and background. In this part the problem of segmenting the image into N regions is studied. First the general framework with the energy function will be presented. This is solved by α-expansions which will be described in the next section. The trick in OneCut to rewrite (2.14) will be imitated in a multi-class setting and the optimization procedure will be described in detail. Finally the problems of stereo depth estimation and geometric model fitting will be presented.

To solve a multi-class segmentation problem, one approach is to minimize a function of the form

$$E(S) = \sum_{p \in \mathcal{P}} D_p(S_p) + \sum_{(p,q) \in \mathcal{E}} V_{pq}(S_p, S_q), \tag{2.22}$$

where $S_p \in \{1, 2, \ldots, N\}$, unlike as for binary segmentation $S_p \in \{0, 1\}$. It turns out that to exactly solve this problem is NP-hard even for simple choices of D and V, so an exact solution is not achievable [2]. However, there are two efficient methods to approximate a solution: α-expansions and $\alpha\beta$-swaps. We will only use α-expansions. If we partition the different pixels as P_i for different labels $i \in \{1, 2, \ldots, N\}$, let P_i denote the current candidate solution and P_i' a new solution, then P_i' is an α-expansion of P_i if $P_\alpha \subset P_\alpha'$ and $P_i' \subset P_i$ for $i \neq \alpha$. That is, the label α is allowed to expand and all other nodes either change to α or keep the old label.

In [2], optimization methods that finds the local minimum of $E(S)$ within an α-expansion of the previous solution are presented. The problems are reduced to binary optimization problems that can be solved by min-cuts. There is an approximation guarantee for α-expansions on the form

$$E(S) \leq 2 \max_{(p,q) \in \mathcal{E}} \left(\frac{\max_{i \neq j} V_{pq}(i, j)}{\min_{i \neq j} V_{pq}(i, j)} \right) E(S^*), \tag{2.23}$$

where S is the local minimum using α-expansions and S^* is the global minimum.

2.2.1 α-expansions

In this section the α-expansion method will be described in greater detail than the overview in the previous section. Algorithm 3 describes the general procedure of the α-expansion algorithm. The hard part of the algorithm is the Expand procedure and it is this part that will be described in detail. In [2] the graph constructions used for α-expansions are described. Here an algebraic version that essentially is the same will be presented.

Assume that we have a current segmentation S and we wish to expand label α. Define x_p for all pixels and let $x_p = 1$ if pixel p changes to label α and let $x_p = 0$ if pixel p keeps the old label S_p. This is a binary problem solvable by

graph cuts. If $S_p = \alpha$ we force $x_p = 1$. Now introduce the energy function

$$E(x) = \sum_p D'_p(x_p) + \sum_{(p,q)\in\mathcal{E}} V'_{pq}(x_p, x_q), \tag{2.24}$$

with

$$D'_p(0) = \begin{cases} \infty & S_p = \alpha \\ D_p(S_p) & S_p \neq \alpha \end{cases}, \quad D'_p(1) = \begin{cases} 0 & S_p = \alpha \\ D_p(\alpha) & S_p \neq \alpha \end{cases}. \tag{2.25}$$

Note that we force $x_p = 1$ for $S_p = \alpha$. For $S_p = S_q$ we have

$$\begin{aligned} V'_{pq}(1,0) = V_{pq}(\alpha, S_q), \quad V'_{pq}(0,1) = V_{pq}(S_p, \alpha), \\ V'_{pq}(0,0) = 0, \quad V'_{pq}(1,1) = 0, \end{aligned} \tag{2.26}$$

and for $S_p \neq S_q$

$$\begin{aligned} V'_{pq}(1,0) = V_{pq}(\alpha, S_q), \quad V'_{pq}(0,1) = V_{pq}(S_p, \alpha), \\ V'_{pq}(0,0) = V_{pq}(S_p, S_q), \quad V'_{pq}(1,1) = 0, \end{aligned} \tag{2.27}$$

where we note that submodularity requires V_{pq} to satisfy the triangle inequality $V_{pq}(S_p, S_q) \leq V_{pq}(S_p, \alpha) + V_{pq}(\alpha, S_q)$. It is assumed that $V_{pq}(\alpha, \beta) = 0$ if $\alpha = \beta$. The Expand method in Algorithm 3 has now been described, and it finds the minimum of $E(x)$, which can be done efficiently. In [2] is a proof that optimization of this construction indeed is the local minimum of the global energy function restricted to a single expansion move.

Algorithm 3 α-expansion

```
S = Initial segmentation
while S has not converged do
    for α ∈ {1, 2, ..., N} do
        S = Expand(S, α)
    end for
end while
```

2.2.2 Multi-class segmentation with OneCut

In the GrabCut section we introduced a bin model for the color distributions. We have bins θ_b. For every bin, the pixels have assigned labels and we partition θ_b into $\theta_b = \cup_l \theta_b^l$ where l denotes the label. We will do the same trick as for OneCut in (2.14) and discarding global volume balancing. The probabilities are computed in the same way as for binary segmentation, but are computed for each label.

$$\begin{aligned} \sum_{p\in\mathcal{P}} D_p(S_p) &= \sum_{p\in\mathcal{P}} -\ln \Pr(I_p|S_p) \\ &= \sum_l \sum_b -|\theta_b^l| \ln\left(\frac{|\theta_b^l|}{|S_l|}\right) \\ &= \sum_l \sum_b -|\theta_b^l| \left(\ln|\theta_b^l| - \ln|S_l|\right) \\ &= \sum_l |S_l| \ln|S_l| - \sum_l \sum_b |\theta_b^l| \ln|\theta_b^l| \end{aligned}. \tag{2.28}$$

Now we proceed as for OneCut and discard the first part of the last expression and define

$$E_{hist}(S) = -\sum_l \sum_b |\theta_b^l| \ln |\theta_b^l|. \tag{2.29}$$

The rest of this section will contain details about how to optimize the above expression and can be skipped all together if one is not interested in the implementation details. The term $E_{hist}(S)$ contains the expression $-x\ln(x)$ which we will approximate using linear functions. Since $-x\ln(x)$ is a concave function we can approximate it as a minimum of linear functions. We consider an approximation of $f(x) = -x\ln(x) \approx \sum_i f_i(x)$ where each f_i is a minimum of two linear functions, $f_i(x) = \min\{a_i^L x + b_i^L, a_i^U x + b_i^U\}$ where we assume $a_i^L \geq a_i^U$. All the f_i are supposed to have different breakpoints, meaning the number x such that $a_i^L x + b_i^L = a_i^U x + b_i^U$. The breakpoints are set at fixed values, for instance logarithmic in scale. Finding a and b is done by solving a linear system of equations obtained by setting the values at the breakpoints to fixed values and also making sure that the breakpoints of all f_i are at the right values. We will implement the minimum in a way similar to that in (2.17) by introducing additional variables, denoted y.

Remember that we are performing α-expansions and that $x_p = 1$ if p is assigned to α and $x_p = 0$ if it keeps the old label S_p. If $S_p = \alpha$ we force $x_p = 1$. We will need to express both $|\theta_b^l|, l \neq \alpha$ and $|\theta_b^\alpha|$. This can be done as follows

$$|\theta_b^\alpha| = \sum_{q\in\theta_b} x_q, \qquad |\theta_b^l| = \sum_{q\in P_l\cap\theta_b} \bar{x}_q, l \neq \alpha. \tag{2.30}$$

Now we are ready to rewrite $E_{hist}(S)$. We will split the labels between α and not α.

$$\begin{aligned}
E_{hist}(S) &= \sum_l \sum_b -|\theta_b^l| \ln |\theta_b^l| \\
&= \sum_b \sum_i f_i\left(|\theta_b^\alpha|\right) + \sum_{l\neq\alpha} \sum_b \sum_i f_i\left(|\theta_b^l|\right) \\
&= \sum_b \sum_i \left[\bar{y}_{\alpha,b,i}\left(a_i^L|\theta_b^\alpha| + b_i^L\right) + y_{\alpha,b,i}\left(a_i^U|\theta_b^\alpha| + b_i^U\right)\right] \\
&+ \sum_{l\neq\alpha} \sum_b \sum_i \left[y_{l,b,i}\left(a_i^L|\theta_b^l| + b_i^L\right) + \bar{y}_{l,b,i}\left(a_i^U|\theta_b^l| + b_i^U\right)\right] \\
&= \sum_b \sum_i \left[\bar{y}_{\alpha,b,i}\left(a_i^L\left(\sum_{q\in\theta_b} x_q\right) + b_i^L\right) + y_{\alpha,b,i}\left(a_i^U\left(\sum_{q\in\theta_b} x_q\right) + b_i^U\right)\right] \\
&+ \sum_{l\neq\alpha} \sum_b \sum_i \left[y_{l,b,i}\left(a_i^L\left(\sum_{q\in P_l\cap\theta_b} \bar{x}_q\right) + b_i^L\right) + \bar{y}_{l,b,i}\left(a_i^U\left(\sum_{q\in P_l\cap\theta_b} \bar{x}_q\right) + b_i^U\right)\right].
\end{aligned} \tag{2.31}$$

Consider the first line in the last equality. The interaction terms are $\bar{y}_{\alpha,b,i}a_i^L x_q + y_{\alpha,b,i}a_i^U x_q$ which are submodular if and only if $a_i^U \leq a_i^L$, which holds by construction of the linear approximations. The terms with $l \neq \alpha$ are submodular by the same reasoning. The interaction terms are $y_{l,b,i}a_i^L \bar{x}_q + \bar{y}_{l,b,i}a_i^U \bar{x}_q$ which are submodular if and only if $a_i^U \leq a_i^L$.

To actually implement it, rewrite the first sum as

$$\begin{aligned}
&\sum_b \sum_i \left[\bar{y}_{\alpha,b,i} \left(a_i^L \left(\sum_{q \in \theta_b} x_q \right) + b_i^L \right) + y_{\alpha,b,i} \left(a_i^U \left(\sum_{q \in \theta_b} x_q \right) + b_i^U \right) \right] \\
&= \sum_b \sum_i \sum_{q \in \theta_b} \left[a_i^L x_q \bar{y}_{\alpha,b,i} + a_i^U x_q y_{\alpha,b,i} \right] + \sum_b \sum_i \left[b_i^L \bar{y}_{\alpha,b,i} + b_i^U y_{\alpha,b,i} \right] \\
&= \sum_i \sum_{q \in P} \left[a_i^L x_q \bar{y}_{\alpha,b(q),i} + a_i^U x_q y_{\alpha,b(q),i} \right] + \sum_b \sum_i \left[b_i^L \bar{y}_{\alpha,b,i} + b_i^U y_{\alpha,b,i} \right],
\end{aligned} \tag{2.32}$$

where we note that $\sum_b \sum_{q \in \theta_b}$ is the sum over all the pixels, and write $b(q)$ to get the bin of a pixel. Similarly, rewrite the second sum as

$$\begin{aligned}
&\sum_{l \neq \alpha} \sum_b \sum_i \left[y_{l,b,i} \left(a_i^L \left(\sum_{q \in P_l \cap \theta_b} \bar{x}_q \right) + b_i^L \right) + \bar{y}_{l,b,i} \left(a_i^U \left(\sum_{q \in P_l \cap \theta_b} \bar{x}_q \right) + b_i^U \right) \right] \\
&= \sum_{l \neq \alpha} \sum_b \sum_i \sum_{q \in P_l \cap \theta_b} \left[a_i^L y_{l,b,i} \bar{x}_p + a_i^U \bar{y}_{l,b,i} \bar{x}_p \right] + \sum_{l \neq \alpha} \sum_b \sum_i \left[b_i^L y_{l,b,i} + b_i^U \bar{y}_{l,b,i} \right] \\
&= \sum_{l \neq \alpha} \sum_i \sum_{q \in P_l} \left[a_i^L y_{l,b(q),i} \bar{x}_p + a_i^U \bar{y}_{l,b(q),i} \bar{x}_p \right] + \sum_{l \neq \alpha} \sum_b \sum_i \left[b_i^L y_{l,b,i} + b_i^U \bar{y}_{l,b,i} \right].
\end{aligned} \tag{2.33}$$

These sums are straight forward to implement.

2.2.3 Stereo depth estimation

For the stereo setup, two cameras are placed next to each other with parallel principal axis and the same orientation, see Figure 2.5. The 3D-point projects to x_l in the left camera and to x_r in the right camera. Since the projection of the second camera center onto the first image is the direction of the x-axis in the first image, we get that the epipolar line through x_l is the line parallel with the x-direction, as showed in the image. The difference in x-coordinates of the two projections is called the disparity and is defined $d = x_l - x_r$. It can be showed that the disparity is inversely proportional to the depth of the 3D-point.

To use graph cuts for stereo depth estimation, we seek an energy to minimize, namely

$$\begin{aligned}
E(S) &= E_{data}(S) + E_{smoothness}(S) + E_{hist}(S) \\
&= \sum_{p \in \mathcal{P}} D_p(S_p) + \sum_{(p,q) \in \mathcal{E}} V_{pq}(S_p, S_q) - \sum_l \sum_b |\theta_b^l| \ln |\theta_b^l|,
\end{aligned} \tag{2.34}$$

where S_p denotes the disparity of the pixel. This formulation is the energy function defined in previous sections and it is solved using α-expansions. Note that the disparities are only allowed to be values in a predefined set of disparities,

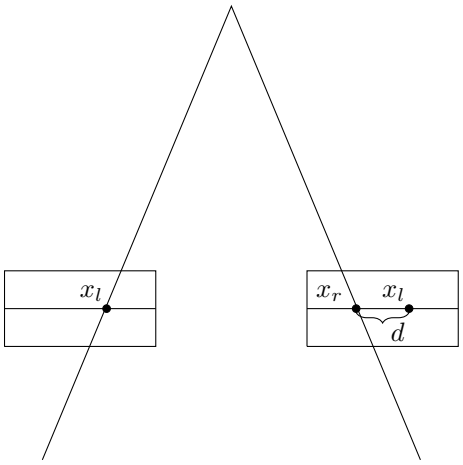

Figure 2.5: Camera setup for stereo.

typically only integers, which gives a depth map with discrete depths and not continuous.

The interaction terms $V_{pq}(S_p, S_q)$ in the multi-class setting, does in general only depend on whether $S_p = S_q$ or $S_p \neq S_q$, since the class numbers only are used for indices. In stereo, the interaction term may be defined as

$$V_{pq}(S_p, S_q) = \min(|S_p - S_q|, t), \tag{2.35}$$

where t is some threshold. The idea is that large disparity changes are penalized, but a large discontinuity may be present, as it typically is for real scenes.

The data term $D_p(S_p)$ is defined for $p = (x, y)$ by comparing intensities in a patch around (x, y) in the first image, denoted by I_1 and a patch around $(x + f, y)$ in the second image, denoted I_2, where f is the disparity associated with the label S_p. Then we use the cross-correlation between these patches

$$\mathrm{NCC}(I_1, I_2) = \frac{1}{n-1} \sum_{i=1}^{n} \frac{(I_1(i) - \bar{I}_1)(I_2(i) - \bar{I}_2)}{\sigma(I_1)\sigma(I_2)}, \tag{2.36}$$

where $\bar{I}_k$ is the mean and $\sigma(I_k)$ is the standard deviation. $D_p(S_p)$ is set to minus this cross correlation. A high correlation thus gives a low cost. The patch is typically the 3×3 neighbourhood around the pixel.

2.2.4 Geometric model fitting

In this section the geometric model fitting problem will be presented. The problem is to assign every pixel one of many possible hypotheses while keeping track of global regularization. The method presented in this section is described in greater details in [6]. The stereo depth estimation problem is a special case of this problem where the hypotheses are the different disparities and we wish

to assign every pixel a disparity. The reader should have this case in mind when reading the rest of the section, which might otherwise be a bit abstract. Another possible application described in [6] is find homographies between image pairs. It is well know that if 3D-points on a plane are projected onto two cameras, then there is a homography between the projected points [5]. The goal is then to find the best homography for each pixel and also taking account of smoothness of the overall solution. This will segment the image into planar regions.

A problem we will consider later is to find planar regions in an RGB-D image. In an RGB-D image we have both colors and depth of all the pixels, see Figure 2.6. A typical approach to this and also the homography problem is to use RANSAC and greedily select the sample that yields the most inliers [5]. The inlier pixels are then considered solved and a new RANSAC iteration is performed with the pixels that were not inliers in the first. While this approach in all the steps finds the best hypothesis it can fail to give a globally coherent solution. The proposed method in [6] is to minimize an energy function with labels being a set of hypotheses and the data terms measure how well the pixel fits the hypotheses.

Suppose we have different hypotheses corresponding to all the labels and data terms that assign a cost between a pixel and a hypothesis. We minimize the following function

$$\begin{aligned} E(S) &= E_{data}(S) + E_{smoothness}(S) + E_{hist}(S) + |\{S\}| \\ &= \sum_{p\in\mathcal{P}} D_p(S_p) + \sum_{(p,q)\in\mathcal{E}} V_{pq}(S_p, S_q) - \sum_l \sum_b |\theta_b^l| \ln |\theta_b^l| + |\{S\}|. \end{aligned} \tag{2.37}$$

Note the similarity with (2.34). The last term denotes the number of hypotheses that are used. In general there might be many hypotheses and we only want the final solution to use a few of those.

This is solved using α-expansions as described in the previous sections. The only new thing is the term $|\{S\}|$ which count the number of hypotheses that are assigned to at least one pixel. To implement this, introduce additional variables z_l for every label (hypothesis). Then the minimum is implemented in the standard way, with $\mathcal{P}_l$ denoting the pixels assigned label l,

$$\begin{aligned} |\{S\}| &= \sum_l \min\{1, |\mathcal{P}_l|\} \\ &= z_\alpha + \bar{z}_\alpha \sum_{p\in\mathcal{P}} x_p + \sum_{l\neq\alpha} \left(\bar{z}_l + z_l \sum_{p\in\mathcal{P}_l} \bar{x}_p \right), \end{aligned} \tag{2.38}$$

where all pairwise terms are submodular. Remember that $x_p = 1$ means the pixel is changed to α and $x_p = 0$ means it keeps the old label S_p.

The new thing with geometric model fitting is the problem of generating hypotheses, and once we have those, setting the values for the data terms $D_p(S_p)$. In stereo depth estimation we have different disparities as hypotheses and the cross correlation measure described in the previous section is used to calculate the values of the data terms. The problem of finding planar regions in RGB-D images will be described in greater detail now. Once we have the data terms we

Figure 2.6: An RGB-D image with the RGB content to the left, D content in the middle and to the right the data term corresponding to the RANSAC proposal with the most inliers.

can solve the problem in exactly the same way as for stereo depth estimation. Algorithm 4 outlines the procedure to find hypotheses and assign values to the data terms. Essentially it is RANSAC where we keep the N best proposals. In the last line a special hypothesis for outliers is added. This label should in the final segmentation only be assigned to pixels that are outliers to all the N most popular hypotheses. Note that for generating hypotheses, only the depth image and not the color image is used and later for regularization, the color image is used for $E_{hist}(S)$.

A number of details remain for the algorithm. Let d_i denote the depth of a pixel. The 3D-points (X_i, Y_i, d_i) are obtained from x_i, y_i and d_i by assuming that the camera matrix has the form

$$P = K \begin{bmatrix} I & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I & 0 \end{bmatrix}, \tag{2.39}$$

where the principal point (x_0, y_0) is supposed to be in the middle of the image and the focal length f is set to a fixed value. Specifically,

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \sim P \begin{bmatrix} X_i \\ Y_i \\ d_i \\ 1 \end{bmatrix} = \begin{bmatrix} fX_i + d_i x_0 \\ fY_i + d_i y_0 \\ d_i \end{bmatrix} \sim \begin{bmatrix} \frac{fX_i}{d_i} + x_0 \\ \frac{fY_i}{d_i} + y_0 \\ 1 \end{bmatrix} \tag{2.40}$$

is a system of equations where we can get X_i and Y_i by knowing all the other variables. Also, in Algorithm 4 inliers are only counted for the largest connected component.

Algorithm 4 Generate RGB-D image plane hypotheses

I_p = the best hypothesis for all pixels
Calculate (X_p, Y_p) from (x_p, y_p, d_p) for all $p \in \mathcal{P}$
for $k \in \{1, \dots\}$ **do**
    Select three random points $\{X_i, Y_i, d_i\}_{i=1}^3$
    Find the plane π through these points
    Count the number of inlier, i.e. points $\{X_p, Y_p, d_p\}_{p \in \mathcal{P}}$ that are closer than d_{min} to p.
    Refit the plane using the inliers, obtaining π'
    Again select the inliers using the new plane π'
    for all inliers q **do**
        if The current hypothesis k has more inliers than I_q **then**
            $I_q = k$
        end if
    end for
end for
Select the N most frequent values in I_p, denoted H_k, $k \in \{1, \dots, N\}$.
for $k \in \{1, \dots, N\}$ **do**
    for $p \in \mathcal{P}$ **do**
        $D_p(k)$ = distance from H_k to (x_p, y_p, d_p)
    end for
end for
Set $D_p(N+1) = \gamma d_{min}$ for all $p \in \mathcal{P}$

Chapter 3

Results

3.1 Binary segmentation

We start by comparing OneCut, GrabCut and fast trust region (Ftr). We vary the smoothness parameter and compare with a ground truth segmentation and also compare the energies of the resulting segmentations obtained by the different methods. The energy functions we consider are

$$\begin{aligned} \text{GrabCut} \quad & E(S) = h_\Omega(S) - \sum_b h_{\Omega_b}(S_b) + \lambda E_{smoothness}(S) \\ \text{OneCut} \quad & E(S) = 2\ln 2 E_{hist}(S) + \lambda E_{smoothness}(S) \\ \text{Ftr} \quad & E(S) = h_\Omega(S) + 4\ln 2 E_{hist}(S) + \lambda E_{smoothness}(S) \end{aligned} \tag{3.1}$$

The slope $2\ln 2$ is the average slope of $-h_\Omega(S)$ between 0 and $|\Omega|/2$, see Figure 2.3. For Ftr this was doubled since using $2\ln 2$ proved unreliable. Throughout this experiment we let $V(1,0) = V(0,1) = 1$ and $V(0,0) = V(1,1) = 0$ for the smoothness terms. Note that both the OneCut energy and Ftr energy are approximations of the GrabCut energy. We compare the segmentation result with a ground truth using the Jaccard score $|S \cap T|/|S \cup T|$ and we compare the energy values evaluated for the GrabCut energy.

In Figures 3.1 and 3.2 are the results using two different images. The energies obtained by Ftr are in most cases lower than the energies for GrabCut. For a low value of λ, the Ftr method gives a bias towards choosing roughly half of the pixels. This is because $h_\Omega(S)$ has a minimum at $|S| = |\Omega|/2$, see Figure 2.3. Also, for OneCut large and small λ tends to only include the seeds, which can be seen in the flower image, where the Jaccard score drops rapidly and the segmentation only includes the foreground seeds.

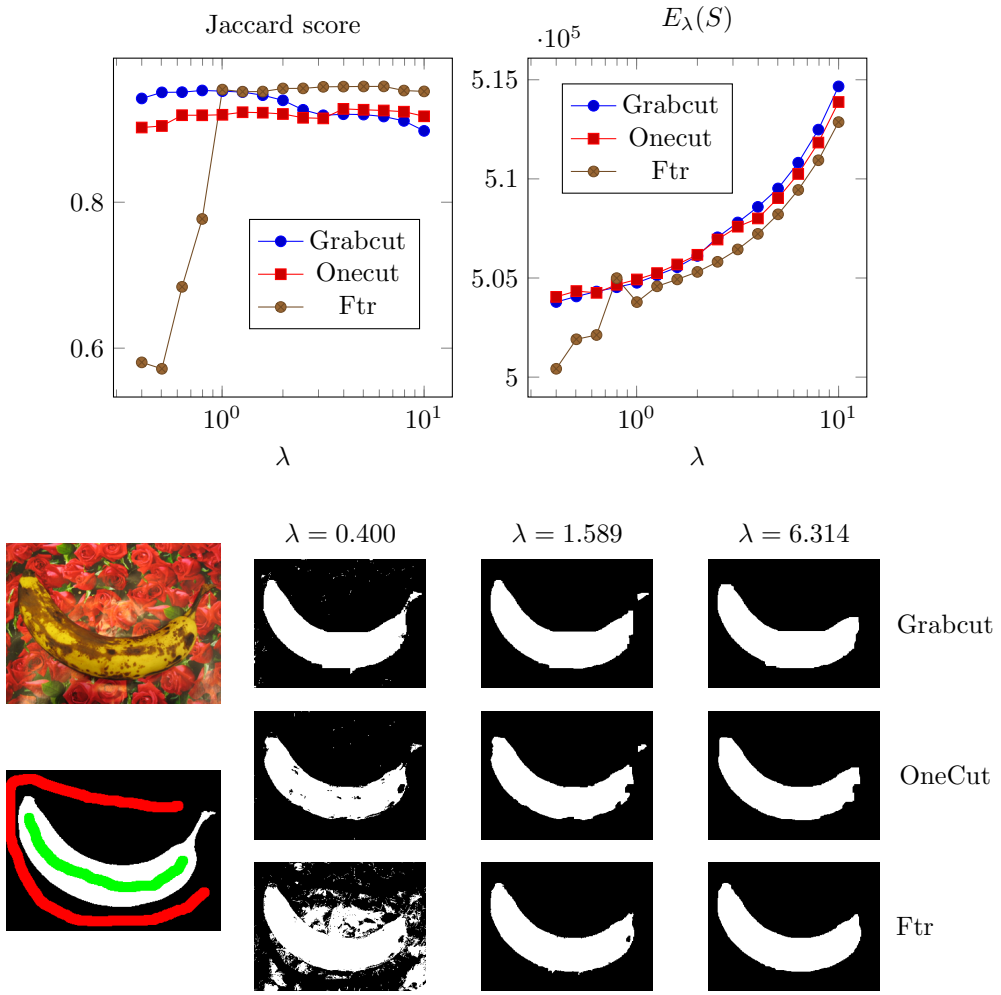

Figure 3.1: Comparison of GrabCut, OneCut and fast trust region (Ftr). The plots show a comparison of the Jaccard score and final energy (computed using the GrabCut energy, note dependence of λ) versus smoothness parameter λ. The left images shows the image to segment and an image with the ground truth and seeds. 32^3 bins were used. The image is taken from the GrabCut dataset [9].

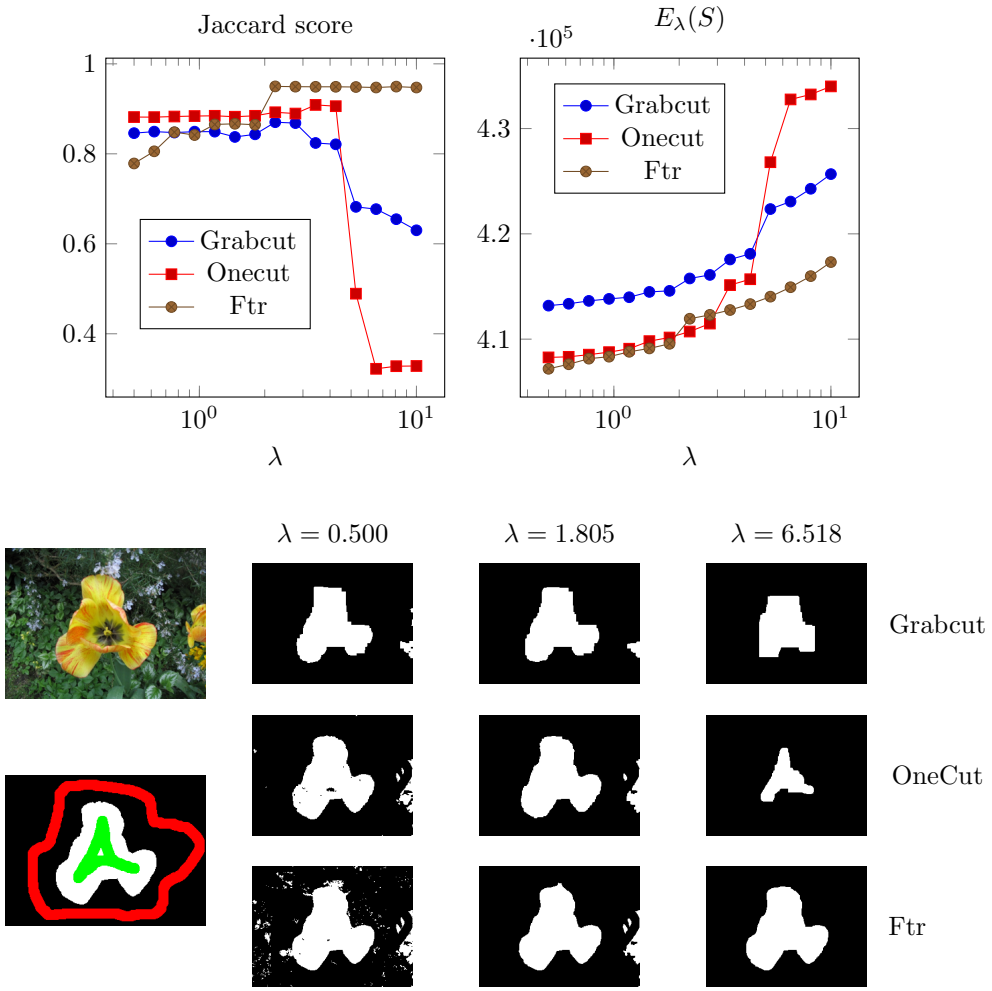

Figure 3.2: Comparison of GrabCut, OneCut and fast trust region (Ftr). The plots show a comparison of the Jaccard score and final energy (computed using the GrabCut energy, note dependence of λ) versus smoothness parameter λ. The left images shows the image to segment and an image with the ground truth and seeds. 32^3 bins were used. The image is taken from the GrabCut dataset [9].

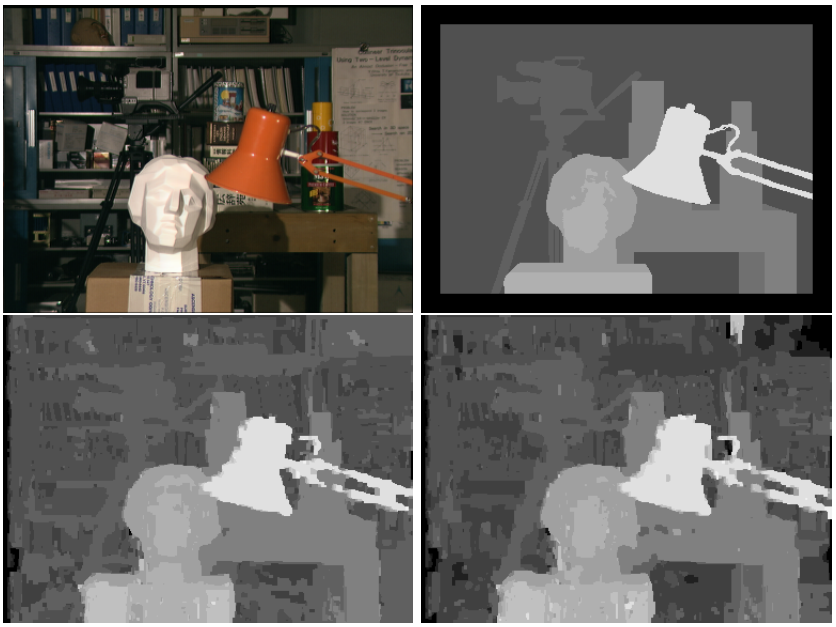

Figure 3.3: Top left: one of the images in the stereo pair. Top right: ground truth. Bottom left: using E_1. Bottom right: using E_2.

3.2 Stereo depth estimation

Here we experiment with the added histogram term for stereo. We will compare the two energies, see (2.34)

$$\begin{aligned} E_1(S) &= \mu E_{data}(S) + \lambda E_{smoothness}(S) + E_{hist}(S) \\ E_2(S) &= \mu E_{data}(S) + \lambda E_{smoothness}(S) \end{aligned} \tag{3.2}$$

to see if the added histogram term improves the result. We use images from the Middlebury dataset[1] and a quantitative comparison outlined in [10]. In Figures 3.3, 3.4, 3.5 and 3.6 are results using both E_1 and E_2, with the same parameters, $\mu = 8$ and $\lambda = 3$. Also, 16^3 bins were used. In Table 3.1 are quantitative results showing that the results are improved by the histogram term.

3.3 Geometric model fitting

This section will contain a few qualitative result of plane fitting in RGB-D images showing that it is possible. In Figure 3.7 are example segmentations.

[1] http://vision.middlebury.edu/stereo

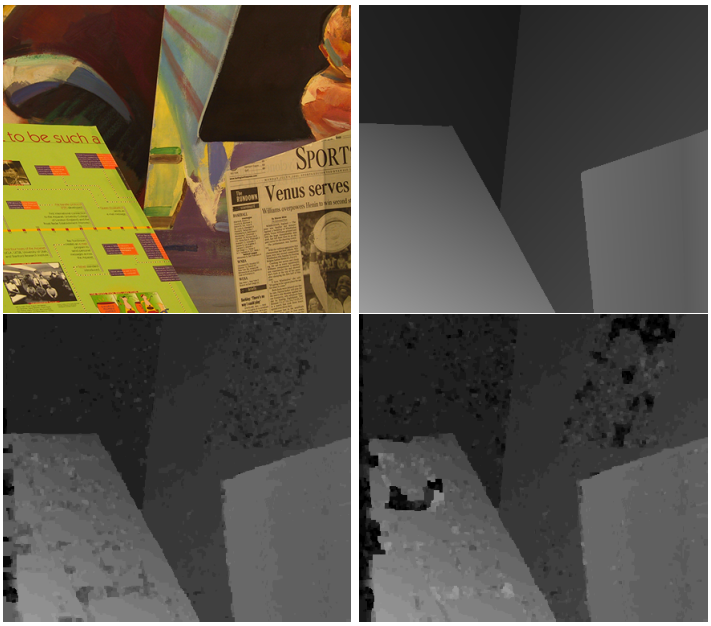

Figure 3.4: Top left: one of the images in the stereo pair. Top right: ground truth. Bottom left: using E_1. Bottom right: using E_2.

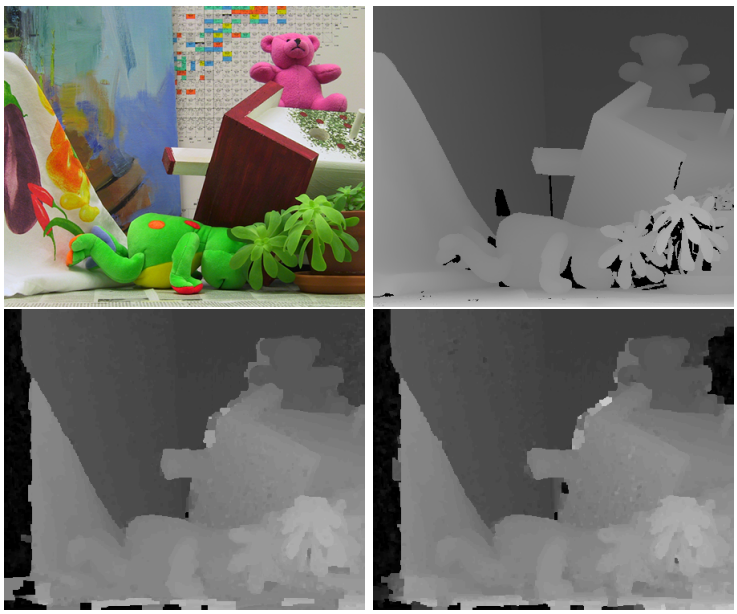

Figure 3.5: Top left: one of the images in the stereo pair. Top right: ground truth. Bottom left: using E_1. Bottom right: using E_2.

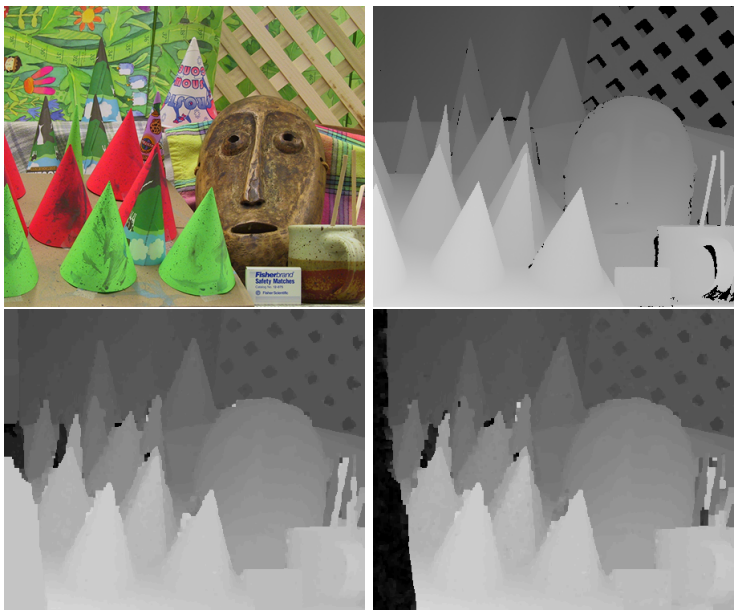

Figure 3.6: Top left: one of the images in the stereo pair. Top right: ground truth. Bottom left: using E_1. Bottom right: using E_2.

	Rank	Average percent bad pixels
E_1	138.6	12.2
E_2	149.2	14.7

Tsukuba, figure 3.3

	Nonocc	All	Disc
E_1	2.95	4.73	13.1
E_2	5.21	7.17	16.9

Venus, figure 3.4

	Nonocc	All	Disc
E_1	7.74	9.25	18.4
E_2	12.5	14.0	24.3

Teddy, figure 3.5

	Nonocc	All	Disc
E_1	11.4	20.2	27.8
E_2	12.6	21.6	26.2

Cones, figure 3.6

	Nonocc	All	Disc
E_1	4.32	14.2	12.4
E_2	5.34	15.7	14.3

Table 3.1: Results for the different images compared to ground truth. Nonocc means the image is only compared to the ground truth on non-occluded pixels. All means all pixels are tested. Disc means that the errors at discontinuities in the ground truth are evaluated. Pixels are considered bad if the disparity is wrong with more than 1. The presented errors are the percentage of wrong pixels in the considered regions. Rank is an aggregate measure of all the results.

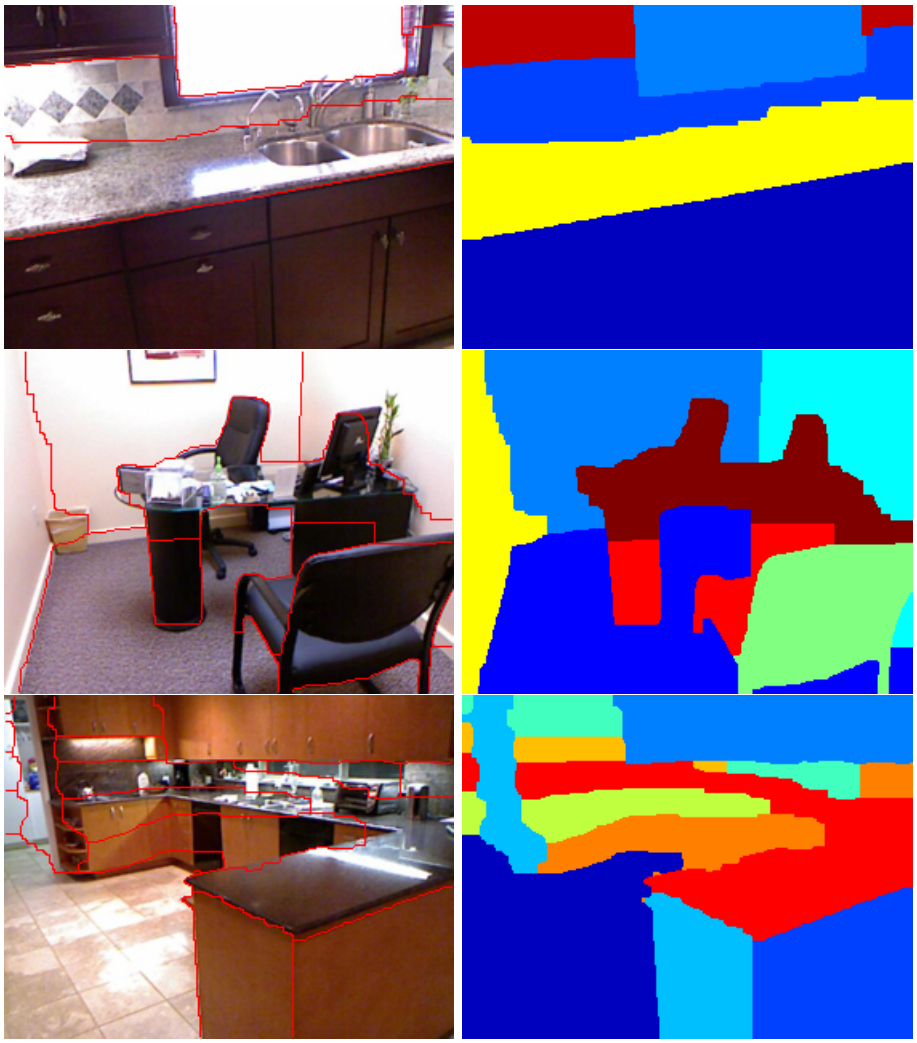

Figure 3.7: Results of segmenting RGB-D images into planar regions.

Chapter 4

Conclusions

For the binary case, we have showed how to add a volume constraint to the OneCut method using the Fast trust region method. The resulting energy was lower with the volume constraint than for OneCut and GrabCut. The sensitivity for parameter choice in OneCut was somewhat mitigated by adding a volume constraint, see Figure 3.2. It should be kept in mind that OneCut is solved by one graph cut and Fast trust region requires solving multiple graph cut problems.

For the multi-class problem, a term similar to the OneCut term has been described and we have showed how to optimize this term using α-expansions. Two applications have been studied: stereo depth estimation and geometric model fitting. For stereo the result on the Middlebury dataset was somewhat improved by including the histogram term. It should be noted that we start with the method described in [2] and there are many known improvements to this method. Testing the histogram term for any of these methods has not been done. For geometric model fitting the qualitative results showed that the walls and major surfaces were found. However, it is unclear how the different terms in the energy function contribute.

Bibliography

[1] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, 2004.

[2] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[3] Lena Gorelick, Frank R Schmidt, and Yuri Boykov. Fast trust region for segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1714–1721. IEEE, 2013.

[4] Lena Gorelick, Frank R Schmidt, Yuri Boykov, Andrew Delong, and Aaron Ward. Segmentation with non-linear regional constraints via line-search cuts. In *Computer Vision–ECCV 2012*, pages 583–597. Springer, 2012.

[5] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[6] Hossam Isack and Yuri Boykov. Energy-based geometric multi-model fitting. *International journal of computer vision*, 97(2):123–147, 2012.

[7] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? In *Computer VisionECCV 2002*, pages 65–81. Springer, 2002.

[8] S.J.D. Prince. *Computer Vision: Models Learning and Inference*. Cambridge University Press, 2012.

[9] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314, 2004.

[10] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002.

[11] Meng Tang, Lena Gorelick, Olga Veksler, and Yuri Boykov. Grabcut in one cut. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1769–1776. IEEE, 2013.

Master's Theses in Mathematical Sciences 2015:E12
ISSN 1404-6342

LUTFMA-3272-2015

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

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