From Interactive to Semantic Image Segmentation

D. Phil Thesis

Robotics Research Group
Department of Engineering Science
University of Oxford

Supervisors:
Professor Andrew Zisserman
Professor Andrew Blake

Varun Gulshan
Brasenose College

Trinity Term, 2011

Varun Gulshan
Brasenose College

Doctor of Philosophy
Trinity Term, 2011

Abstract

This thesis investigates two well defined problems in image segmentation, viz. interactive and semantic image segmentation. Interactive segmentation involves power assisting a user in cutting out objects from an image, whereas semantic segmentation involves partitioning pixels in an image into object categories. We investigate various models and energy formulations for both these problems in this thesis.

In order to improve the performance of interactive systems, low level texture features are introduced as a replacement for the more commonly used RGB features. To quantify the improvement obtained by using these texture features, two annotated datasets of images are introduced (one consisting of natural images, and the other consisting of camouflaged objects). A significant improvement in performance is observed when using texture features for the case of monochrome images and images containing camouflaged objects. We also explore adding mid-level cues such as shape constraints into interactive segmentation by introducing the idea of geodesic star convexity, which extends the existing notion of a star convexity prior in two important ways: (i) It allows for multiple star centres as opposed to single stars in the original prior and (ii) It generalises the shape constraint by allowing for Geodesic paths as opposed to Euclidean rays. Global minima of our energy function can be obtained subject to these new constraints. We also introduce Geodesic Forests, which exploit the structure of shortest paths in implementing the extended constraints. These extensions to star convexity allow us to use such constraints in a practical segmentation system. This system is evaluated by means of a "robot user" to measure the amount of interaction required in a precise way, and it is shown that having shape constraints reduces user effort significantly compared to existing interactive systems. We also introduce a new and harder dataset which augments the existing GrabCut dataset with more realistic images and ground truth taken from the PASCAL VOC segmentation challenge.

In the latter part of the thesis, we bring in object category level information in order to make the interactive segmentation tasks easier, and move towards fully automated semantic segmentation. An algorithm to automatically segment humans from cluttered images given their bounding boxes is presented. A top down segmentation of the human is obtained using classifiers trained to predict segmentation masks from local HOG descriptors. These masks are then combined with bottom up image information in a local GrabCut like procedure. This algorithm is later completely automated to segment humans without requiring a bounding box, and is quantitatively compared with other semantic segmentation methods. We also introduce a novel way to acquire large quantities of segmented training data relatively effortlessly using the Kinect. In the final part of this work, we explore various semantic segmentation methods based on learning using bottom up super-pixelisations. Different methods of combining multiple super-pixelisations are discussed and quantitatively evaluated on two segmentation datasets. We observe that simple combinations of independently trained classifiers on single super-pixelisations perform almost as good as complex methods based on jointly learning across multiple super-pixelisations. We also explore CRF based formulations for semantic segmentation, and introduce novel visual words based object boundary description in the energy formulation. The object appearance and boundary parameters are trained jointly using structured output learning methods, and the benefit of adding pairwise terms is quantified on two different datasets.

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Varun Gulshan, Brasenose College

Acknowledgements

While the previous page states that this thesis is entirely my own work, the reality is that there are a number of amazing people behind my 'own work'. And here's my chance to thank them all. My advisors, **Prof. Andrew Zisserman** and **Prof. Andrew Blake** have not only been behind all the inspiration and ideas in this thesis, but have been instrumental in improving my thinking in all aspects of life and taking it to the next level. I will forever be their student. Thank you AZ and AB, you have made me a much better person.

I've been lucky to meet a lot of amazing people as part of this 4 year (4.25 years to be precise) journey of learning and fun. My labmates at VGG have been the best of people, and here's a personal thanks to them all (in no particular order): Thanks **Mukta** for advice on all things in life, Thanks **Patrick** for keeping the social life in the lab alive, Thanks **James** for being the best desk mate (and those cuff-links you lent me once, made me look smart :), Thanks **Emi** for bringing smiles to the lab, Thanks **Florian** for the philosophical discussions connecting Computer Science with life :). Then there are all the lovely juniors in VGG who have made it fun: Thanks **Relja, Arpit, Yusuf, Amr, Ken** and **Karen**. Good luck to my dearest first years at VGG: **Tomas, Omkar, Meelis** and **Mircea**. May you all get your PhD [but feel free to quit and join a startup, I almost did :)]. Special mention to **Nataraj**, an amazing thinker and beer drinker. And many more thanks to the non-vgg people who were so much fun: Thanks **Paul**, I'm sure we will end up in the same place again, its been now 10 years together! Thanks **Chris** (Mei), some amazing philosophical discussions I had with you, Thanks **Alex** (flint), for all our discussions about the future of mankind (we still need to work on our PhD show idea though :) and Thanks **Mark** (cummins), for lots of useful advice on startup stuff. And whoa, the post-docs get ignored everywhere :), but they are still lovely people: Thanks **Pawan**, I learnt a lot from you even in the few months we overlapped. Thanks **Andrea**, you are an amazing thinker, and I learn something new everytime I talk to you. Thanks **Victor**, you have always been the quiet genius. Thanks **Matthew**, you brought some coolness to the post-doc room :). Thanks **Marcin**, you were one hell of a hacker.

I have also been lucky to have some amazing friends, who have contributed a lot in my life and shaped my personality in many ways. Thanks **Rahul Garg**, you have been an inspiration to me, and will always be. Thanks **Grover**, you taught me how to dance and how to listen. Thanks **Singhal**, you taught me how to have fun in life. Thanks **Sahni**, for all the non-technical things I would not know without you. Thanks **Khudi**, for the many smiles you bring to everyone. Thanks **Gopal**, for teaching me how to not worry. Thanks **Chappu**, for teaching me how to be calm. Thanks **GR**, for being an amazing room mate at IIT. Thanks **Manav**, for broadening my knowledge and understanding of things.

There are many other un-thanked people who have contributed in so many ways in my life. The list would be longer than my thesis, thank you all the people who have touched me and made a difference. I would also like to thank some non-living things, because they have all been so amazing. Maximum thanks to my **Dell Laptop**, you have been rock solid, no data loss, no hardware failure, always there by my side. I would marry you if you were alive. Thanks to my **iPhone**, you have helped me keep in touch with people that matter to me. Thanks to my **bike**, for taking me around Oxford at any time of the day. Big thanks to my **work desk** and all the **toys** at my desk, you have been around all the time. And a final thanks to my amazing **ear muffs** for keeping me warm through the few harsh winters in Oxford.

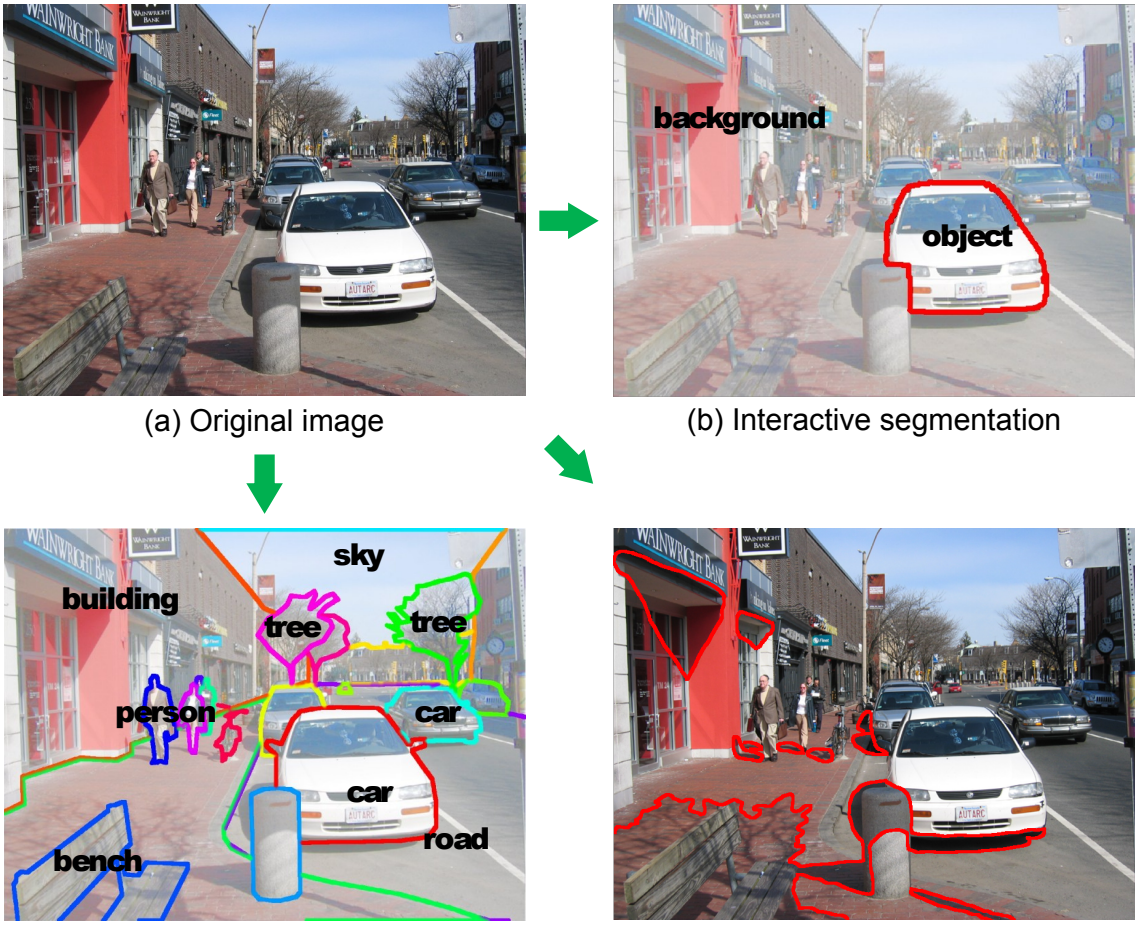
And you must be thinking, what about family, doesn't this guy have one? (especially if the reader is one of my brother, mom or dad). Yes I do, and a great one that. Thanks **Mom** and **Dad**, you defined me early on and then let me define myself, this thesis is for you (but don't try to read it, you won't understand :). Thanks to my brother **Karan Bhaiya**, he has been my advisor on everything outside academic. I won't be complete without him.

Contents

Chapter 1

Introduction

This thesis deals with the problem of image segmentation. The term segmentation, in general refers to the process of partitioning a set of entities into groups. Depending on the problem definition, these entities can correspond to words in a speech signal, pixels/contours in an image, point trajectories in a video etc. In this work, we deal with the problem of image segmentation – i.e. the grouping of pixels into meaningful regions. The problem of segmentation is ill-defined until a particular task is identified. For example, what are good segments for the image in Figure 1.1(a)? In this work, we answer this question with respect to two well defined tasks. The first task is interactive segmentation, where a human user desires to cut out a particular object from an image for the purpose of image editing. This task is well defined due to the presence of the human user who guides the segmentation algorithm. The performance at this particular task can also be evaluated quantitatively using metrics that measure the amount of effort spent in segmenting a particular object. This effort could be measured in units of user time, user experience ratings etc. The other task we tackle in this work is of segmenting object categories. The



(a) Original image (b) Interactive segmentation

(c) Semantic segmentation (d) Shadow segmentation

Figure 1.1: *Segmentation is task dependant.* The correct partitioning of an image depends on the task. Three different segmentation tasks are shown in (b), (c) and (d), with the corresponding partitionings outlined using coloured lines. (b) An interactively segmented car (the user might be interested in copy pasting the car onto another photo). (c) The semantic segmentation of the image into pre-defined object categories such as car, building, sky, tree etc. Such a segmentation is useful for scene understanding. (d) Segmenting out shadows in an image for the purpose of image editing. (Image in (c) courtesy: http://johnwinn.org/Presentations/presentations/ProbabilisticModelsForUnderstandingImages_ICML2008.ppt)

term 'object category' is used here to refer to a particular visual concept such as the concept of a car, horse, person, building etc. This task is well defined because specifying the object category defines which pixels should belong to a particular group. Segmentation outputs for both these segmentation tasks are illustrated for an image in Figure 1.1. The interactive segmentation task falls under the category of figure-ground segmentation, where the partitioning consists of a figure and a background (thus corresponding to a binary labelling of pixels), whereas in the second task it is possible to have more than two partitions depending upon the number of object categories.

1.1 Motivations and applications

Due to the increasing proliferation of digital photos and photo management/editing tools such as Picasa and Adobe Photoshop CS5, image editing is now a commonly used desktop application. Interactive segmentation is a fundamental part of such image editing tools, and has been integrated into two widely used software products, viz. Adobe Photoshop CS5 (called the quick-select tool) and Microsoft Powerpoint 2010 (called the background removal tool). Possible uses for interactive segmentation include retouching and adding special effects to certain parts of an image, removing an object from an image, copy-pasting an object from one image to another or simply removing the background from an image in order to highlight the object. Figure 1.2 shows some of the above discussed applications of interactive segmentation. Intelligent tools that allow users to select the desired pixels with minimal effort are key to making better image editing systems, as the brute force approach of making pixelwise selections (such as the simple Lasso selection tool in

(a) Original image (b) Image retouching

(c) Simple cutout (d) Object cut-paste

Figure 1.2: *Interactive segmentation applications.* (a) The original unedited image. (b) The image is retouched by converting the background to grayscale, thus highlighting the person in colour. (c) Simple cutout of the person, frequently of the type used in magazines and newspapers. (d) A cut and paste application of image segmentation for synthesising new images.

Photoshop) is tedious and error prone.

The second segmentation task, i.e. segmenting particular object categories from an image has applications from a scene understanding point of view. Understanding which parts of the image belong to what object category is a fundamental problem in perception and is useful towards the larger goal of building intelligent machines. Being able to extract such semantic information from images could also be useful in

organising photo collections (Picasa already allows users to filter photos containing faces), in building autonomously driving cars, parsing Google street view images to make them searchable etc. It can also be used to reduce user effort in interactive segmentation settings, by providing good initial segmentations when the object category is known – e.g. magazine covers and newspaper articles often contain cutouts of humans, so having a human specific interactive system can help reduce user effort.

1.2 Challenges in segmentation

First we discuss the challenges faced in interactive segmentation systems. An interactive segmentation system is fairly generic, in that it can be used to segment any object category that the user would like to segment. This means that interactive systems cannot make any assumptions about the object being segmented, and hence need to rely upon low-level cues such as pixel colours for segmentation. Natural images can have complex colour distributions which are not modelled easily by parametric functions, and often the colour distributions of the object and the background overlap significantly. This means that colour by itself is often not good enough to discriminate between the object and background. Figure 1.3 shows the complexity and overlap of colour distributions in a natural image by visualising 3D scatter plots of RGB values. Shadows and illuminations further add to the complexity of modelling object appearances using colour distributions. Other low level cues such as pixel contrasts on object boundaries can also be weak in certain parts of the image. Object boundaries often have mixed pixels, i.e. pixels that belong to both background and foreground as a consequence of motion blur and the point spread function of the camera. Segmenting out such pixels correctly is challeng-

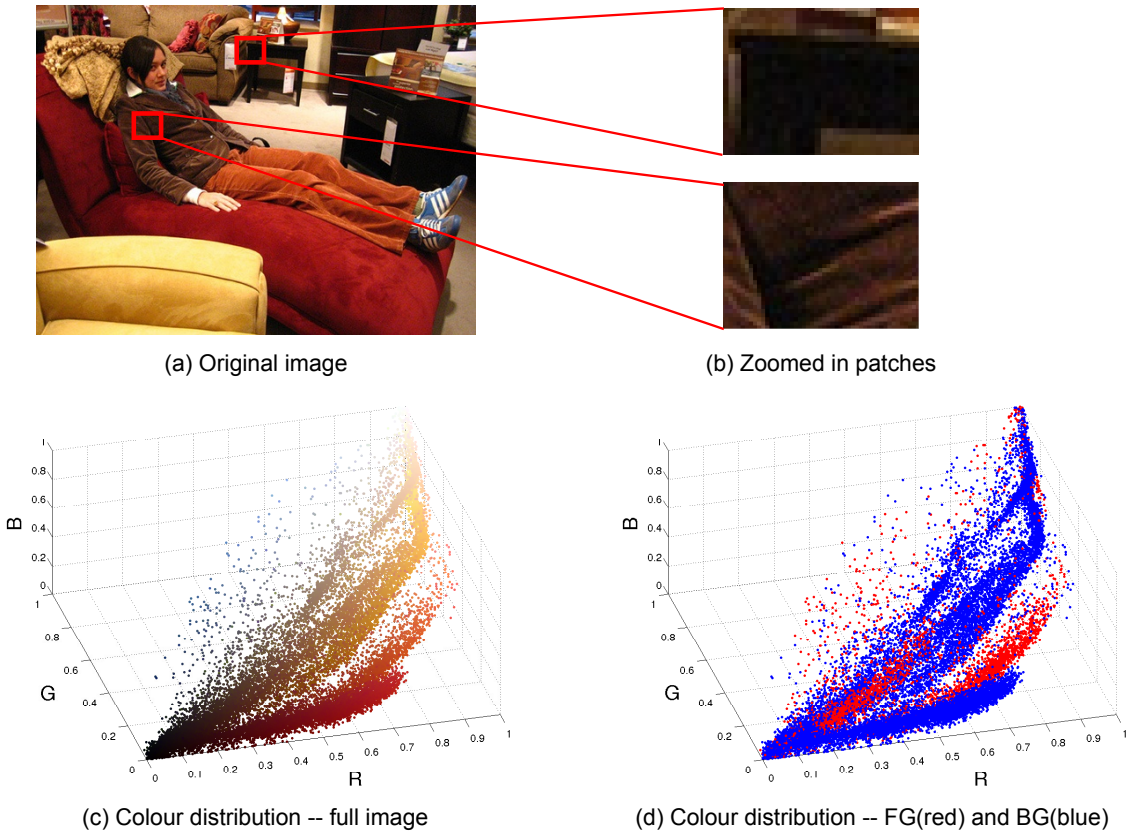


Figure 1.3: *Interactive segmentation challenges: Complex and overlapping colour distributions.* (a) Shows an image with a wide distribution of colours. The user would like to segment the girl sitting on the sofa. (b) Shows two patches from the foreground(FG) and the background(BG) that have very similar colours. (c) The global colour distribution of the image is visualised as a scatter plot of RGB values. The distribution spans a range of RGB values, signifying a complex distribution of colours. (d) Visualises the same scatter plot as (c), with the colours encoding FG and BG membership. The overlap between the FG and BG distributions can be seen clearly.

ing and involves estimating continuous transparency values as opposed to discrete membership labels of pixels. Objects can also have complex topologies with thin structures like hair, wires etc. which can be hard to select interactively. Figure 1.4 illustrates the challenges with weak boundaries, mixed pixels and thin structures in a natural image. These challenges can be overcome partially by imposing additional

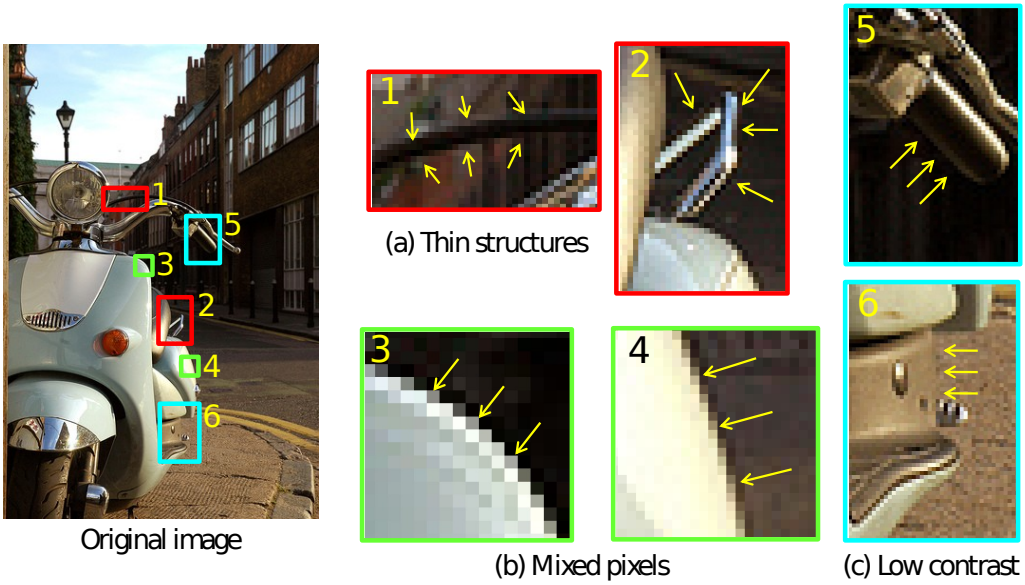


Figure 1.4: *Interactive segmentation challenges: Thin structures, mixed pixels and low contrast boundaries.* (a) Thin structures: Boxes 1 and 2 outlined in red highlight regions of the object which are very thin compared to the rest of the object. It can be hard to interactively select such thin parts. (b) Mixed pixels: Boxes 3 and 4 outlined in green highlight object boundaries where blending of foreground and background pixels can be observed. Continuous mattes need to be estimated for such pixels. (c) Low contrast: Boxes 5 and 6 outlined in cyan highlight low contrast boundaries of the object. Low contrast boundaries can be hard for algorithms to segment correctly due to lack of low level cues.

constraints on the segmentation such as consistency of labelling between neighbouring pixels, connectivity constraints, generic shape priors etc. Section 1.3 discusses our contributions in improving upon these low level cues using texture features and star convexity constraints.

In addition to the challenges involving low-level cues in interactive segmentation, semantic segmentation faces challenges often faced in higher level recognition of objects. It is almost impossible to classify pixels into image categories based on low-level cues such as colour and edges. Thus, semantic segmentation often needs complex models that can integrate cues over large regions of the image and also be

Figure 1.5: *Semantic segmentation challenges: Appearance variation within instances of same object.* Shown above are 9 images containing the object category 'boat' in the PASCAL VOC image dataset (Everingham et al., 2010). Note the variation in colour, size, viewpoint, geometric structure and appearance within these images of the same object category.

invariant to the geometric and appearance variations within instances of the same object. Figure 1.5 shows images from an object category 'boat' that exhibits a large intra-class variation. Variation within a class can come from various factors such as position and pose of the object relative to the camera, varying 3D structures and appearances of the object, occlusions, variation in illumination, shadows etc. Overcoming these challenges involves using better features to describe objects and powerful classifiers that can learn this variation within object categories. In this work, we explore several learning methods to segment object categories.

1.3 Contributions

1.3.1 Interactive segmentation

The first part of this thesis explores improvements in interactive segmentation systems based on low level cues. The first contribution is a comprehensive evaluation of texture based features in modelling object appearances for interactive segmentation. The performance gains obtained by adding texture features to existing interactive systems are evaluated and it is shown that texture features provide significant gain for the case of monochrome images. An additional dataset containing camouflaged objects is introduced to further demonstrate the utility of texture features. Further improvements in interactive systems are introduced by adding mid-level shape constraints in the form of star convexity priors. These shape constraints are very efficient to implement and help reduce interaction effort significantly by eliminating many false segmentations. The reduction in interaction effort in demonstrated quantitatively by simulating user interaction and measuring the effort required. To this end, an interactive segmentation dataset is also introduced which extends existing datasets with more real life images.

1.3.2 Semantic segmentation

The second part of this work is related to the problem of semantic segmentation. We slowly move away from interactive segmentation methods into more automated methods. The first contribution in this aspect is an algorithm that learns to segment humans given bounding boxes. In order to learn the large variability in human poses, an automatic method of capturing segmented humans using the Xbox Kinect is introduced. Automating the data collection enables us to acquire a large dataset and

train an efficient classifier to segment humans. We combine the trained classifier with low-level colour and edge cues used in interactive segmentation to build a final system that segments humans accurately given bounding boxes. This method is then fully automated by using people detectors to obtain the bounding boxes automatically. In the last part of this work, we learn to segment arbitrary object categories from bottom up segmentations. This method is fully automated and does not require any user interaction. Various methods to learn from multiple segmentations are explored in this work. We also explore learning boundary properties of segmentations jointly with object appearance in a structured output learning framework.

1.3.3 Publications

Following is the list of publications related to our work:

[1] A. Vedaldi, V. Gulshan, M. Varma and A. Zisserman. **Multiple Kernels for Object Detection**. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2009.

[2] V. Gulshan, C. Rother, A. Criminisi, A. Blake and A. Zisserman. **Geodesic Star Convexity for Interactive Image Segmentation**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[3] V. Gulshan, V. Lempitsky and A. Zisserman. **Humanising GrabCut: Learning to Segment Humans Using the Kinect**. In *Workshop on Consumer Depth Cameras for Computer Vision, ICCV*, 2011.

1.4 Thesis outline

Chapter 2 presents a review of existing interactive and semantic segmentation methods. The history of interactive segmentation methods and various formulations proposed till date are reviewed and discussed in detail. Special emphasis is given to the graphcut formulation used in this work. Separate sections are devoted to discussing pure bottom up segmentation methods based on super-pixelisation, followed by a discussion of semantic segmentation algorithms.

Chapter 3 discusses our first contribution towards improving interactive systems by using texture features for segmentation. The performance gains obtained by adding texture are evaluated quantitatively on two datasets. A new dataset containing images of camouflaged objects to demonstrate the effectiveness of texture features is introduced in this chapter.

Chapter 4 describes our next step in improving interactive systems by introducing a star convexity shape constraint into existing segmentation formulations. The theory of star convexity is discussed along with details of how to implement it in a practical system. A rigorous quantitative evaluation is provided by simulating user interaction on a large dataset of images. The significant reduction in user interaction effort is demonstrated through this evaluation.

Chapter 5 takes a step towards more automated segmentation of images by using higher level knowledge of object categories. It focuses on segmenting humans in images given a bounding box. The algorithm is trained on a large dataset of segmented humans obtained automatically using the Xbox Kinect. A fully automated system that uses people detectors to automatically obtain bounding boxes is demonstrated later in the chapter.

Chapter 6 presents fully automatic semantic segmentation algorithms that build upon low level super-pixelisations. Various methods for learning using multiple segmentations are introduced and discussed in the first part. In the second part, an algorithm that jointly learns appearances and pairwise terms in a structured output setting is introduced. All methods are evaluated quantitatively on two datasets.

Chapter 7 presents the conclusions and summarises the work done in this thesis.

Chapter 2

Literature Survey

This section reviews recent developments in interactive and semantic segmentation. In the first part of the review spanning Sections 2.1–2.2, we trace the history of various interactive methods, discuss the cost functions used to formulate the segmentation problem and give details of the various optimisation procedures used. A more in-depth review of the graph-cut energy minimisation framework is provided in Section 2.2 as this is the main optimisation engine used in most of our work. The second part of the review (Sections 2.3–2.4) deals with more automated segmentation methods which do not require any user interaction. Section 2.3 discusses various unsupervised and bottom-up super-pixelisation methods. These bottom-up methods are often used as a pre-process for various semantic segmentation methods that are discussed in detail in Section 2.4.

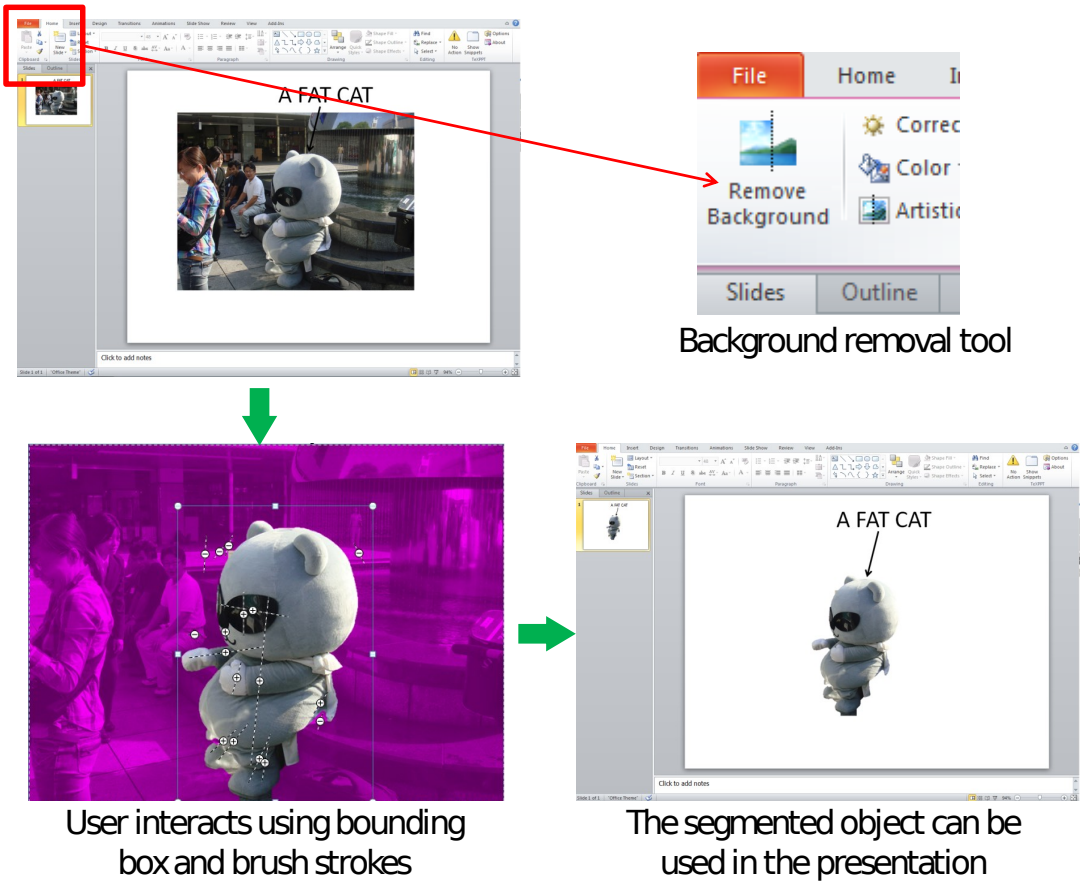


Figure 2.1: *Background removal tool in Microsoft Powerpoint 2010.* The tool lets users cut and paste objects of interest from images into in the presentation. The tool is based on the `GrabCut` paper by Rother et al. (2004). User interaction shown in the bottom left involves the user selecting a bounding box around the object, followed by further edits to correct the segmentation.

2.1 Interactive segmentation

Interactive segmentation is an essential image editing tool and now is an integral part of Adobe Photoshop CS5 and Microsoft Powerpoint 2010 (see Figure 2.1 for an example). Fairly mature implementations of interactive segmentation now exist in industry, and a significant amount of research has gone into building the products that exist today. Some of the earliest work on this problem can be traced back to

Active Contours (Kass et al., 1987) and Intelligent Scissors (Mortensen and Barrett, 1995). Most of these early models for interactive segmentation relied on finding good edges in images. A major weakness of these methods was the lack of representation of the object as the energy functions were only defined on the contour. To overcome this limitation, Chan and Vese (2001) defined a segmentation model based on the Mumford-shah functional (Mumford and Shah, 1989) that took into account appearances of objects alongside contour energies. In most of these methods, contours were represented using level sets as their representation for a segment. Section 2.1.1 provides a detailed discussion on such contour based methods.

More recently, pixel label based representations have become popular as a way of formulating the segmentation problem. These formulations are mostly based on random field models and can capture both region and boundary properties in a single cost function. Since the landmark work of Boykov and Jolly (2001) on using graph-cut inference for interactive segmentation, there has been a lot of research activity around the same representation but different cost functions (Criminisi et al., 2008; Duchenne et al., 2008; Grady, 2005, 2006; Rother et al., 2004; Sinop and Grady, 2007; Veksler, 2008; Li et al., 2004; Levin et al., 2004; Bai and Sapiro, 2009). These methods are reviewed in detail in Section 2.1.2. Section 2.2 is devoted to the discussion of optimisation of such cost functions using graph cuts.

2.1.1 Contour based methods

The Active contours work of Kass et al. (1987) was one of the earliest attempts at segmenting objects by measuring properties on the object-background boundary. User interaction in Kass et al. (1987) involved the user marking a rough curve around the object. This curve then snapped to the actual boundary by optimising a certain

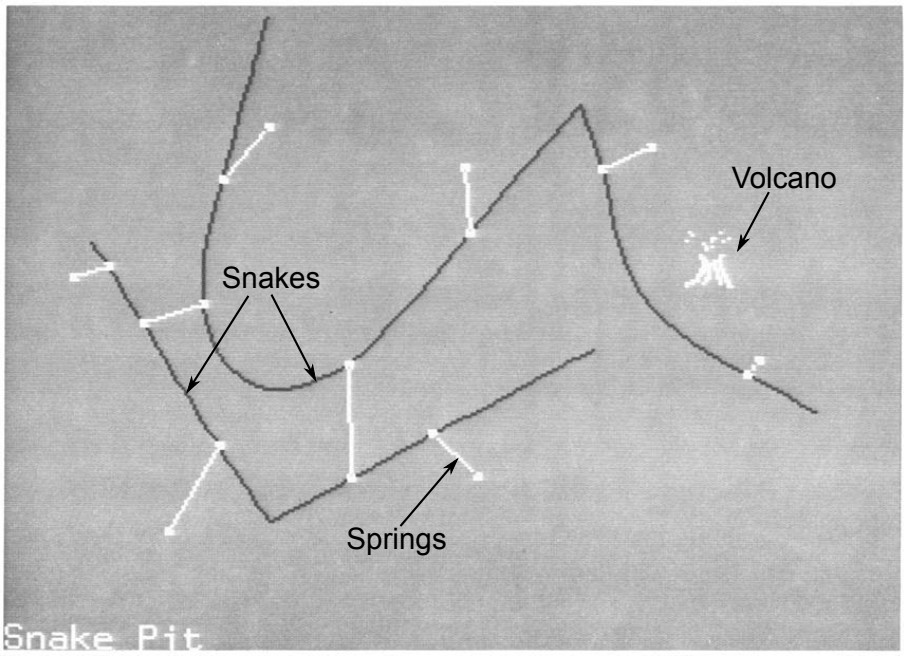


Figure 2.2: *Snake pit user interface for interacting with snakes.* The figure shows two snakes, with external forces such as springs (that attract the snake) and volcanoes (that repel the snake). Image taken from Kass et al. (1987).

cost function. Kass et al. (1987) formulated this cost function as follows. Let Ω be a bounded open subset of $\mathcal{R}^2$ denoting the spatial domain of the image. Let $u : \bar{\Omega} \rightarrow \mathcal{R}$ be a given monochrome image and $C(s) : [0,1] \rightarrow \mathcal{R}^2$ be a one dimensional curve parametrised by s. The energy of a particular curve C is described by the functional $J(C)$:

$$J(C) = \alpha \int_0^1 |C'(s)|^2 \mathrm{ds} + \beta \int_0^1 |C''(s)| \mathrm{ds} - \lambda \int_0^1 |\nabla u(C(s))|^2 \mathrm{ds} \quad (2.1)$$

$$C^* = \inf_C J(C) \quad (2.2)$$

where α, β and λ are positive parameters of the system that trade off different terms. $C'(s)$ denotes the first derivative of $C(s)$ wrt s, and $C''(s)$ the second

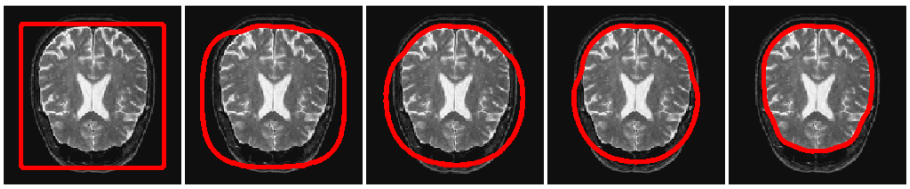


Figure 2.3: *Active contour evolution.* The initial contour provided by the user in the form of a bounding box snaps to the desired contour as the energy is minimised iteratively from left to right. Image courtesy http://www.ceremade.dauphine.fr/~peyre/cours/manifold/tp1.html

derivative (curvature) of the contour. The first two terms encourage the curve to be smooth while the last term encourages the curve to align itself to image edges. In Kass et al. (1987), this functional is minimised using an explicit representation for the contour using a few control points initialised from user interaction. The optimisation involves solving the Euler equations obtained by differentiating the functional in (2.1) to obtain local minima. They also allow for specifying further terms in the energy function such a spring force which attracts contours to a particular point and volcanoes which repel the contour from a particular point (see Figure (2.2) for an visualisation of the user interface). While Kass et al. (1987) used an explicit representation of the contour, implicit representations using level sets became popular for solving such functionals. In the level set formulation, the curve C is represented as the zero crossings of a level-set function $\phi(x, y) : \Omega \rightarrow \mathcal{R}$, i.e :

$$in(C) = \{(x, y) \in \Omega : \phi(x, y) < 0\} \tag{2.3}$$

$$out(C) = \Omega \setminus in(C) = \{(x, y) \in \Omega : \phi(x, y) > 0\} \tag{2.4}$$

$$C = \{(x, y) | \phi(x, y) = 0\} \tag{2.5}$$

Minimising functionals similar to (2.1) using level set based representations is

discussed in Caselles et al. (1993) (see Figure 2.3 for an example of curve evolution using such a method). Such methods are based on gradient descent and only find local minima of the energy functional making them sensitive to initialisation. There is related work on contour based tracking by Blake et al. (1993); Perez et al. (2001); Isard and Blake (1998). For example, the Jetstream (Perez et al., 2001) work expresses contour generation as a stochastic process, and uses particle filtering methods to find a contour that maximises the posterior probability of the contour. More specifically, if we denote the first $i+1$ points on a contour by $s_1, s_2 \cdots, s_{i+1}$ and the image data by $\mathbf{x}$, the posterior probability of the contour is given by the following recursive equations:

$$p_{i+1}(s_{1:i+1}|\mathbf{x}) \propto p_i(s_{1:i}|\mathbf{x})q(s_{i+1}|s_{i-1:i})l(x(s_{i+1}))$$

where the term $q(s_{i+1}|s_{i-1:i})$ captures the second order dynamics of the curve by penalising changes of direction in the contour. The term $l(x(s_{i+1}))$ indicates the likelihood of the contour passing through the point s_{i+1} based on the image gradient at that point. This posterior is then iteratively maximised using standard particle filtering based sampling techniques (Kitagawa, 1996). Such contour based methods work well if the edges are very clearly defined and the initial curve is close to the actual boundary. The weaknesses of the above formulations lie in not taking into account any appearance model of the object to be segmented. It can also be a very tedious task to edit the contour if it gets stuck in the wrong local minima using such cost functions.

To overcome these limitations, Chan and Vese (2001) describe an energy function based on the Mumford-Shah functional (Mumford and Shah, 1989) which does not

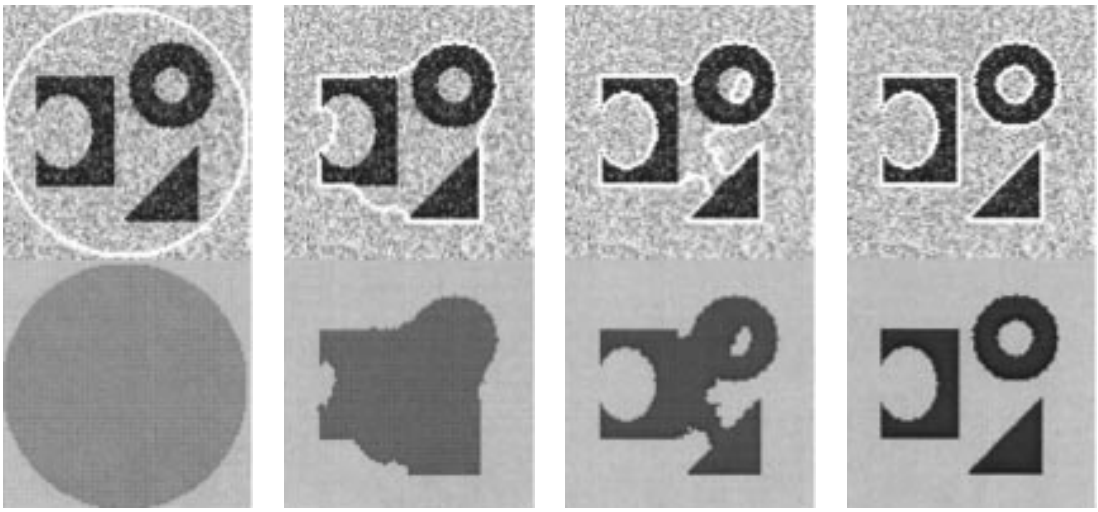


Figure 2.4: *Evolution of the contour while minimising* (2.7). The top row shows the evolution of the 0-level set of the function $\phi(x, y)$ used to represent the contour C in (2.7). The bottom row shows the piecewise reconstruction of the image that is also simultaneously optimised. Image taken from Chan and Vese (2001). The contour is able to delineate the objects in the image and adapt to the varied topologies.

rely on edges for evolving contours. It also takes into account region properties, and hence it would be appropriate to classify their method as *region based* rather than contour based (see Jermyn and Ishikawa (2001) for a discussion on how certain region based functionals can be written as contour based integrals by rewriting them as the flux of a vector field). Consider an image $u : \bar{\Omega} \to \mathcal{R}$ that consists of two regions of approximately piecewise constant intensities u^0 and u^1 as in Figure 2.4. Let C be a parametrised curve as before. A fitting term is defined as:

$$F_1(C) + F_2(C) = \int_{in(C)} |u(x, y) - c_1|^2 \mathrm{dx}\,\mathrm{dy} + \int_{out(C)} |u(x, y) - c_2|^2 \mathrm{dx}\,\mathrm{dy} \qquad (2.6)$$

where c_1 and c_2 are the averages of the regions inside C and outside C. The fitting term measures how well can the image be modelled with a piecewise constant

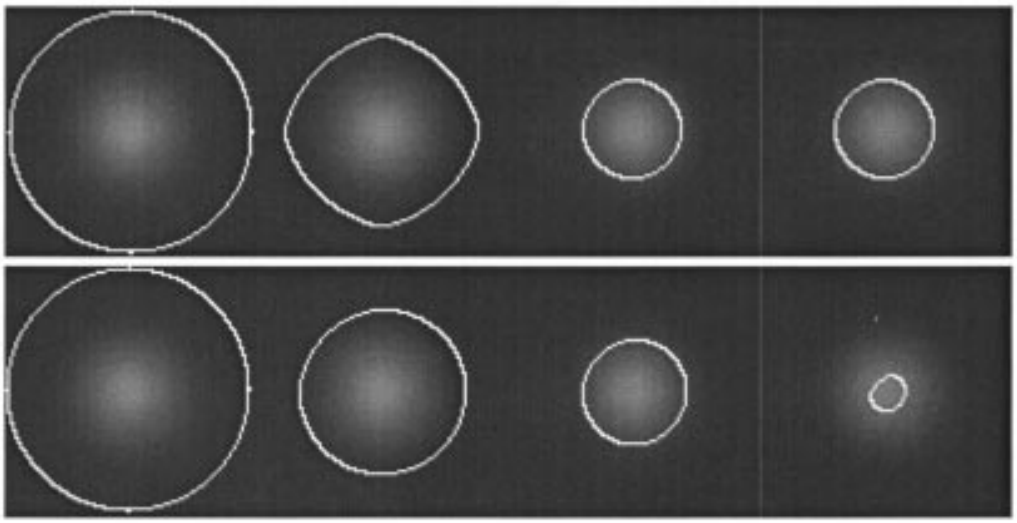


Figure 2.5: *Segmenting an object with a weak contour.* The top row shows curve evolution using the energy functional of Chan and Vese (2001) described in (2.7). The bottom row shows curve evolution using the energy functional of Caselles et al. (1993) (which relies only on edges). The Chan and Vese (2001) energy functional is able to segment objects with weak contours as it also models the appearance of the object. Image taken from Chan and Vese (2001).

approximation into two segments that are delineated by the contour C. For the particular case of image u mentioned above (i.e. consisting of two piecewise constant regions with intensities u^0 and u^1), one can see that the actual contour in u will be the minimiser of (2.6) with $c_1 = u^0$ and $c_2 = u^1$. A regularisation term is added to the fitting term to define the final energy functional that is optimised:

$$F(C) = \mu.Length(C) + \lambda_1 F_1(C) + \lambda_2 F_2(C) \tag{2.7}$$

This above energy functional is equivalent to the Mumford-Shah functional restricted to piecewise constant functions. It can be minimised again by representing the contour C as the zero crossing of a level set function $\phi(x, y)$. The evolution equations of this level set function are described in detail in Chan and Vese (2001).

See Figure 2.4 for a visualisation of the level set evolution on a toy image. The advantage of the above formulation compared to the previous contour models is that it is less sensitive to initialisation, as it relies on global appearance information and not just edge information. It can also find reasonable segmentations when the edges in the image are weak or not so well defined (see Figure 2.5).

Most of the methods discussed so far rely upon level-set methods that solve PDE's to evolve the contours. More recently, Boykov et al. (2006) proposed another surface evolution method that solves the same energy formulations using the geo-cuts combinatorial optimisation framework of Boykov and Kolmogorov (2003).

Another popular early work on interactive image segmentation is the Intelligent Scissors by Mortensen and Barrett (1995). It again relies only upon edge information to find object contours. Although no global energy function is defined in its formulation, it is similar in nature to the Active Contour work in that it latches onto the strongest edges in the image. As illustrated in Figure 2.6, the algorithm computes the shortest path from a seed pixel (which is a point marked by the user on the boundary) to the current position of the user's mouse. The user only needs to move the mouse crudely along the desired boundary, and the shortest path computation snaps it to a path around the object. The algorithm automatically inserts new seed points on the path based on their stability over time. The shortest path computation is done on the pixel grid, with the edge weight $l(p, q)$ between adjacent pixels p and q given by:

$$l(p, q) = \alpha_Z \cdot f_Z(q) + \alpha_D \cdot f_D(p, q) + \alpha_G \cdot f_G(q) \tag{2.8}$$

where f_Z is a quantity that measures the Laplacian Zero Crossing at q, f_G is

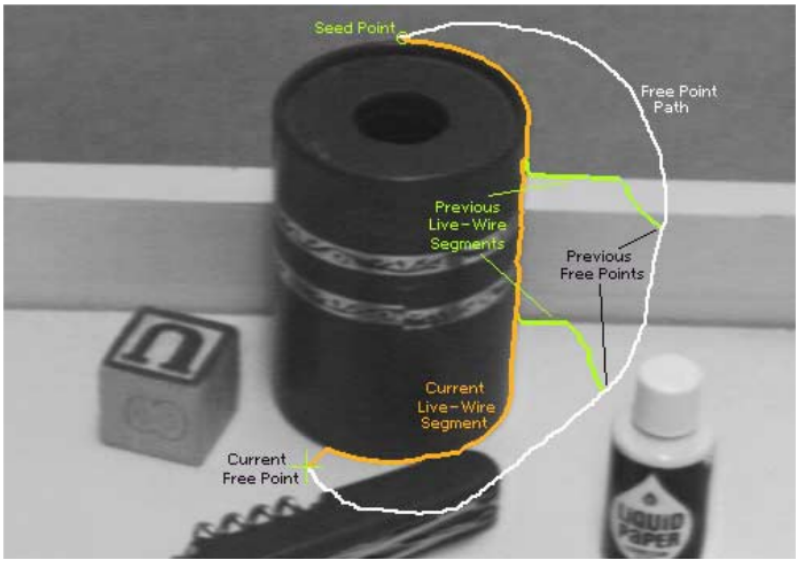


Figure 2.6: *The Intelligent Scissors at work.* The user specifies the seed point on the boundary and then drags the mouse along the free path (shown in white). The corresponding shortest paths (or *live wires*) are drawn in green at two intermediate points. The current live wire is shown in orange. Image taken from Mortensen and Barrett (1995).

inversely proportional to the gradient magnitude at q, and the f_D term measures how well the edge between p and q is aligned to the gradient direction. The α's are weights on the different terms. So $l(p, q)$ is small if there is a good edge orthogonal to the edge between pixels p and q. Dijkstra's shortest path algorithm is used to compute the shortest paths on a graph with the above defined weights. This formulation also suffers from limitations mentioned above of relying only on edge information. However, an advantage of this algorithm is that the shortest path computation is optimal, and hence any undesired solution can be attributed to a limitation of the model and not the optimiser.

2.1.2 Pixel labelling based methods

The previous section discussed the history of interactive segmentation formulations based on explicit contour or implicit level set representations. Most recent interactive segmentation algorithms (including the industry implementations in Adobe Photoshop CS5; Microsoft Powerpoint 2010) are instead based on formulating the segmentation task as a pixel labelling problem. Early work along this line was by Boykov and Jolly (2001) who proposed a globally optimisable energy function for binary labelling of pixels. Their formulation elegantly incorporates object representations such as colour histograms while taking interactions between pixels along the boundary into account. We introduce the following notation to describe their energy function: $\mathbf{x}$ denotes the vector of image pixels, where each element x_i of the vector is the RGB value of pixel i. The binary segmentation is denoted by the vector $\mathbf{y}$, with each element $y_i \in \{0, 1\}$ denoting the foreground$(= 0)$/background$(= 1)$ label. The appearance models for the foreground and background objects are parametrised by $\mathbf{w}$. These variables are combined in a Conditional Random Field (CRF) formulation over the segmentation $\mathbf{y}$, globally conditioned on the observed image $\mathbf{x}$ and the colour models $\mathbf{w}$ (see Figure 2.7 for the factor graph of the CRF). Each pixel is connected to its 8 neighbouring pixels to form a clique size of 2. The energy function given by this CRF is outlined below:

$$E(\mathbf{y}; \mathbf{w}, \mathbf{x}, \gamma) = \sum_{i=1}^{N} U_i(y_i|\mathbf{x}, \mathbf{w}) + \gamma \sum_{(i,j)\in\mathcal{N}} V_{ij}(y_i, y_j|\mathbf{x}) \tag{2.9}$$

where $\mathcal{N}$ denotes the set of neighbouring pixels, γ is a parameter of the system

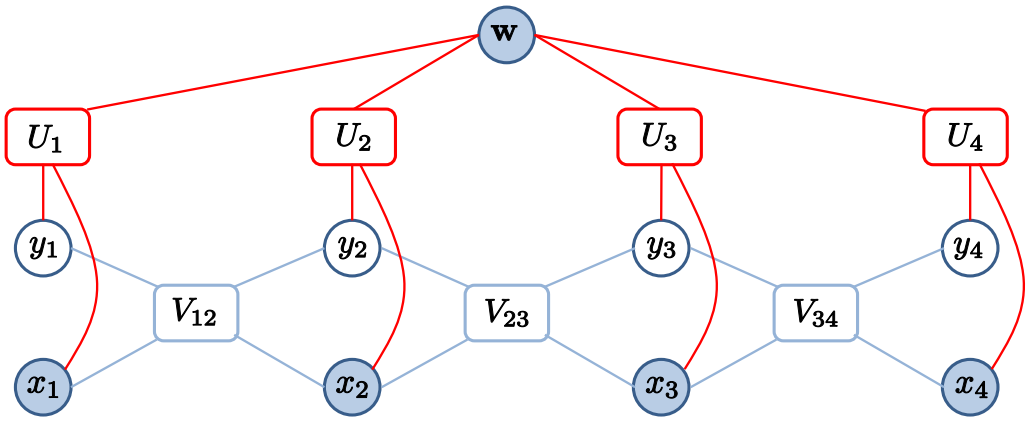


Figure 2.7: *Factor graph (Frey, 1998) for the boykov jolly energy function* in (2.9). The image shows a simple 1-D image with four pixels. The circles represent variables in the energy function (with the filled ones denoting observed variables). The factors are represented by rectangles, with the unary terms coloured red and pairwise terms coloured blue.

and the unary and pairwise terms are defined as:

$$U_i(y_i|\mathbf{x}, \mathbf{w}) = \begin{cases} -\log(p(x_i|\mathbf{w}_{fg})) & \text{if } y_i = 1 \\ -\log(p(x_i|\mathbf{w}_{bg})) & \text{if } y_i = 0 \end{cases} \tag{2.10}$$

$$V_{ij}(y_i, y_j|\mathbf{x}) = (y_i \neq y_j)e^{-\beta\|x_i - x_j\|^2} \tag{2.11}$$

where the terms $\mathbf{w}_{fg}$ and $\mathbf{w}_{bg}$ parametrise the foreground and background colour models respectively and $\mathbf{w} = \{\mathbf{w}_{fg}, \mathbf{w}_{bg}\}$. The unary term $U_i(y_i|\mathbf{x}, \mathbf{w})$ takes into account object based representations. It corresponds to negative log likelihood of observing the pixel value given the colour models. Boykov Jolly used histograms to represent the probability density functions used in (2.10), whereas Rother et al. (2004); Blake et al. (2004) use Gaussian mixture models. These unary terms may also be computed discriminatively as done by Kumar and Hebert (2006) using a logistic classifier. The pairwise term $V_{ij}(y_i, y_j|\mathbf{x})$ term captures the properties of

(a) Original image (b) User Annotation (c) Output segmentation

Figure 2.8: *User assisted segmentation with the Boykov and Jolly (2001) energy function.* (a) Original unannotated image. (b) The user provided seed brushes for the fg (white) and bg (red). (c) The output segmentation mask with the background coloured blue.

the segmentation boundary. It works by penalising label changes in regions of low contrast. The parameter β is set as in Boykov and Jolly (2001):

$$\beta = (2 \left\langle |x_i - x_j|^2 \right\rangle)^{-1} \tag{2.12}$$

where $\langle \cdot \rangle$ denotes an average over all neighbouring pixel pairs $i, j \in \mathcal{N}$. User interaction in this formulation consists of brush strokes to mark out foreground(fg) and background(bg) seed pixels (refer Figure 2.8). These brush strokes are then used to obtain the fg and bg colour models used in (2.10). The brush strokes are also additionally imposed as hard constraints in the energy function, and this is done easily by modifying the unary terms to take an infinite cost for violating the assigned label (from an implementation point of view, a finite cost that is big enough to overcome the effect of pairwise terms is sufficient, see Boykov and Jolly (2001)). Editing the obtained segmentation is also very easily allowed by the energy function. The user simply adds brush strokes in the incorrect regions of the segmentation, the algorithm then updates the the colour models and hard constraints and finds the segmentation that minimises the updated energy. One of the strengths of the energy

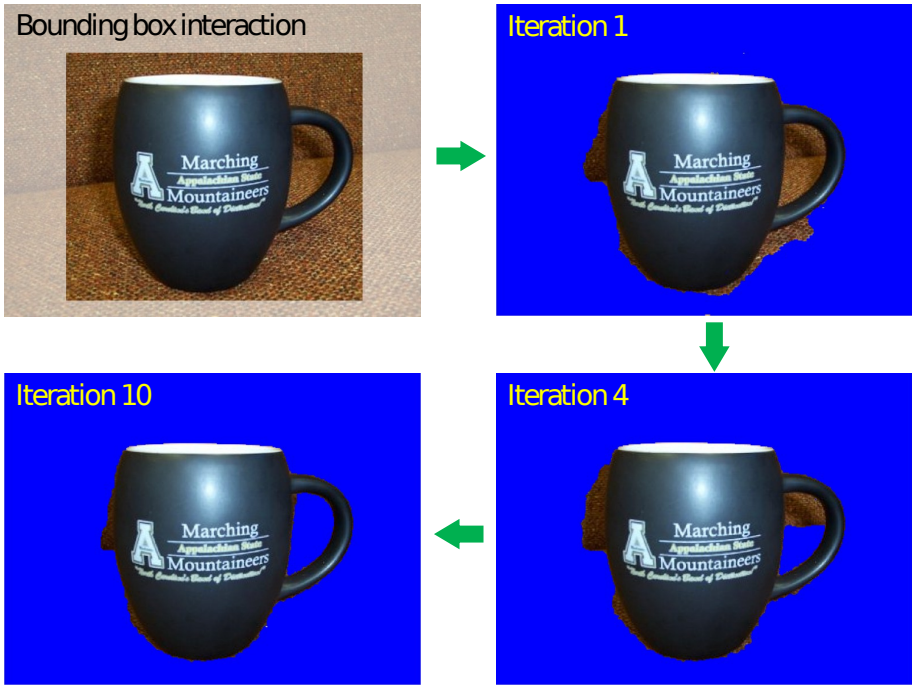


Figure 2.9: *Interactive segmentation using GrabCut.* The user initialises the segmentation with a bounding box. The algorithm then learns the colour models across several iterations and converges to the desired segmentation.

function in (2.9) is that it can be optimised globally as described later in Section 2.2. This means that any errors in the segmentation are due to deficiencies in the model and not the optimiser.

The landmark work of Boykov and Jolly (2001) was a major departure from previous works in interactive segmentation, as the proposed energy function allowed for arbitrary user interaction while providing globally optimal solutions at the same time. Since then, there have been several extensions that build upon this energy function. `GrabCut` (Rother et al., 2004) simplified user interaction further by requiring only a bounding box around the object as an initialisation (see Figure 2.9). `GrabCut` uses the same energy function as Boykov and Jolly (2001), but in addition to optimising over the segmentation labels $\mathbf{y}$, it also optimises the colour models $\mathbf{w}$

using a simple alternation (a good initialisation of $\mathbf{w}$ is obtained from the bounding box) (see Section 2.2.2 for more details). Lazy Snapping (Li et al., 2004) also uses the same energy function, but provides for a faster user experience by preprocessing the image into superpixel segments. A similar energy function extended to the time dimension has also been used for video segmentation by Li et al. (2005); Wang et al. (2005). More recent interactive segmentation work includes Paint selection (Liu et al., 2009), which improves upon the unary terms in (2.10) by using localised colour models based on recent brush strokes and also introduce a faster multi-core optimisation to provide instant feedback. Video Snapcut (Bai et al., 2009) is another recent video segmentation method that builds on top of this energy function but uses better unary terms computed over local windows tracked from the previous time frame. Extensions of this cost function are also used in semantic segmentation by Kumar et al. (2005); Kumar and Hebert (2006); Winn and Shotton (2006) and are described later in Section 2.4.

Grady (2006) proposed a similar formulation to Boykov Jolly that was motivated by the idea of random walks. Given seed pixels for foreground and background as in Boykov and Jolly (2001), for every pixel i it computes the probability of a random walker starting at i reaching one of the seed pixels. The random walker is allowed to make moves in a lattice like graph (the graph is similar to the CRF in Boykov and Jolly (2001)). The edge weights of the graph define the random walker biases (analogous to the contrast dependent terms of Boykov and Jolly (2001)). Though it would be very expensive to compute the probabilities of reaching the seed pixels by sampling from such a distribution, one can obtain the solution analytically by solving the analogous Dirichlet problem with boundary conditions at the seed points (Doyle and Snell, 1984). For the binary case, if we denote by y_i the probability

(a) Seed pixels (b) RW probabilities (c) Output segmentation

Figure 2.10: *Interactive image segmentation using random walks.* (a) Seed pixels provided by the user (blue = fg and pink = bg seed pixels). (b) The probabilities y_i for each pixel reaching the foreground seed (obtained by minimising (2.13)). (c) The output segmentation mask with the background coloured blue.

of a random walker starting at pixel i reaching a seed pixel labelled 1, then these y_i's can be computed as solutions of the following Dirichlet problem (obtained after some manipulation, see Grady (2006) for details):

$$\begin{aligned} \mathbf{y}^* &= \arg\min \frac{1}{2} \sum_{i,j \in \mathcal{N}} w_{ij}(y_i - y_j)^2 \\ \text{s.t} \quad & y_i = 1 \ \forall i \in F \\ & y_i = 0 \ \forall i \in B \end{aligned} \tag{2.13}$$

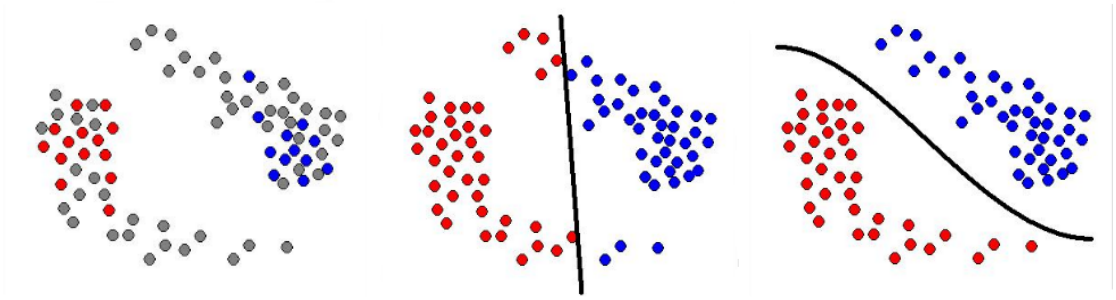
where the weights $w_{ij} = \exp(-\beta \|x_i - x_j\|^2)$ give the random walker biases (note the similarity to the edge weights in (2.11)) and F and B denote the foreground and background sets of seed pixels respectively. Grady also extends this model to include unary terms based on colour models in Grady (2005). The corresponding

optimisation problem is given by:

$$\begin{aligned} \mathbf{y}^* &= \arg\min y_i^2\lambda_i^0 + (1-y_i)^2\lambda_i^1 + \gamma \cdot \frac{1}{2}\sum_{i,j\in\mathcal{N}} w_{ij}(y_i - y_j)^2 \qquad (2.14) \\ \text{s.t} \quad & y_i = 1 \ \forall i \in F \\ & y_i = 0 \ \forall i \in B \end{aligned}$$

where λ_i^1 and λ_i^0 denote probabilities of the pixel i belonging to the labels 1 and 0 respectively and $\mathcal{N}$ denotes the neighbourhood system of the pixels. The above minimisation has a closed form solution which corresponds to solving a sparse linear system of equations (see Grady (2006) for a discussion on various sparse linear system solvers). To obtain a hard segmentation from the continuous probability values in y_i, simple thresholding around 0.5 is done. While the above cost function is motivated very differently to the Boykov Jolly energy function in (2.9), one can reinterpret the random walker cost function as a relaxation of the Boykov Jolly cost function into the continuous domain (the terms λ_i^1 and λ_i^0 in (2.14) can be seen as the unary terms $U_i(y_i = 0|x_i, \mathbf{w})$ and $U_i(y_i = 1|x_i, \mathbf{w})$ in (2.9) respectively, and the term $w_{ij}(y_i - y_j)^2$ in (2.14) corresponds to the pairwise term $V_{ij}(y_i, y_j|x_i, x_j)$ in (2.9)).

Duchenne et al. (2008) present another interpretation of the random walker cost function in the form of transductive learning (see Figure (2.11)). Their cost function is a generalisation of (2.13) to larger pixel neighbourhoods, which increases their computational expense significantly. Sinop and Grady (2007) further generalised the random walker cost function to the p-norm by replacing the pairwise term $\sum_{ij\in\mathcal{N}} w_{ij}(y_i - y_j)^2$ in (2.13) by its p-norm $\left(\sum_{ij\in\mathcal{N}}(w_{ij}|y_i - y_j|)^p\right)^{1/p}$. They show that Boykov and Jolly (2001) and Grady (2006) are special cases of their energy



(a) Seed pixels (b) Fully supervised classifier (c) Semi supervised classifier

Figure 2.11: *Image segmentation using transductive learning of Duchenne et al. (2008).* (a) Pixels visualised in feature space, where the blue and red pixels correspond to seed pixels obtained from brush strokes. The grey pixels represent the pixels to be segmented. (b) A fully supervised classifier does not take into account the grey pixels and hence makes errors. (c) A semi-supervised classifier, with the distance between pixel features defined using both spatial and RGB distances takes into account the unlabelled data while learning the classifier.

function corresponding to $p = 1$ and $p = 2$ respectively. They also obtain a new algorithm by setting $p = \infty$ and show that the geodesic matting (Bai and Sapiro, 2009) solution is one of the many degenerate solutions for the case $p = \infty$.

2.2 Graph-cuts optimisation

The previous section discussed various formulations for interactive segmentation, starting from the early contour based representations of Kass et al. (1987) to the more recent pixel labelling based formulations of Boykov and Jolly (2001); Rother et al. (2004); Liu et al. (2009). This section is devoted to the discussion on optimising the pixel labelling based energy function in (2.9) introduced in Section 2.1.2. The energy function is optimised by expressing it as an instance of a maxflow problem (Boykov and Kolmogorov, 2004) in a graph with suitable edge weights. Greig et al. (1989) were the first to use combinatorial min-cut/maxflow algorithms to min-

imise such energy functions. In the graph construction shown in Figure 2.12, each pixel corresponds to a node in the graph, with two additional nodes labelled as the source(s) and the sink(t). There are two kinds of edges in the graph: (i) The n-links, which go between nodes corresponding to pixels only and encode the pairwise terms (i.e. $V_{ij}(y_i, y_j|x_i, x_j)$) of the energy function and (ii) The t-links connect a pixel node to the s and t nodes and encode the unary terms (i.e. $U_i(y_i|\mathbf{x}, \mathbf{w})$) of the energy function. A $s-t$ cut is defined as the set of edges which when removed from the graph, disconnects the nodes s and t (i.e there does not exist a directed path from s to t). A minimal $s-t$ cut is defined as a $s-t$ cut such that adding back any edge from the cut leads to $s-t$ being connected. A minimal $s-t$ cut has the property that exactly one t-link is severed in the cut (if both links are severed, one of the t-links can be returned while still remaining a valid cut). Using this property, a minimal cut defines a unique corresponding labelling such that $y_i = 1$ if the t-link between (i, t) is broken and $y_i = 0$ if the t-link between (i, s) is broken. It can be shown the cost of a minimal $s-t$ cut is the same as the energy of the corresponding labelling up to an additive constant (Boykov and Jolly, 2001). As the minimum cut also belongs to the set of minimal cuts, finding the minimum cut leads to finding the minimum of the energy (with the optimal segmentation being the one corresponding to the minimum cut).

It is well known from combinatorial optimisation that the $s-t$ min-cut can be solved by computing the maximum flow from s to t in polynomial time (Ford and Fulkerson, 1962). Maxflow algorithms are well studied in the combinatorial optimisation literature and several algorithms such as the augmenting path based Dinic's algorithm (Dinic, 1970) and push-relabel (Goldberg and Tarjan, 1988) have been proposed for solving the maxflow problem. Boykov and Kolmogorov (2004)

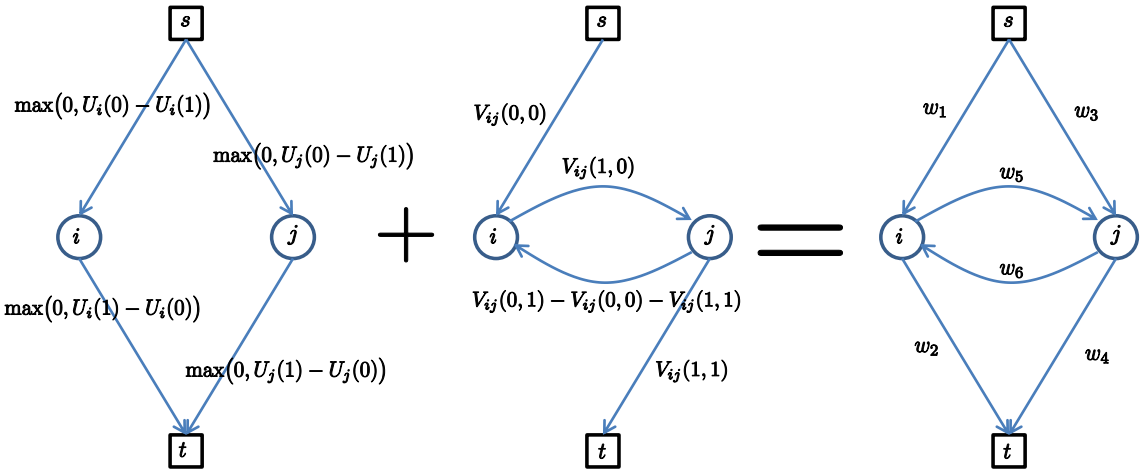


Figure 2.12: *Representing the energy function as a graph.* The leftmost graph shows how the unary terms are encoded into t-links, and the middle graph shows how the pairwise terms are encoded into the n-links and t-links. The final graph (on the right) sums the edge weights of the two graphs. Note, the above construction is one particular way of encoding the energy function. There are multiple equivalent re-parametrisations of the energy function that may be encoded by different edge weights, and correspond to the same energy function upto an additive constant. In order to have a feasible graph construction (i.e a graph with positive edge weights), the submodularity criteria needs to be satisfied (Kolmogorov and Zabih, 2004). The segmentation label y_i of a node is given by the label of the terminal it is disconnected from in the min-cut ($s = 0$ and $t = 1$ in the above notation).

provide a modification of Dinic's algorithm that works well in practice for many graphs encountered in computer vision (even though their theoretical complexity is worse than that of Dinic (1970), they provide empirical evidence of speed up using their algorithm). The graph-cuts algorithm is not just restricted to the minimising energy functions with potentials defined in (2.10) and (2.11). It is more general and can be used to minimise many other energy functions that have the same form as in (2.9). A necessary and sufficient condition for minimising energy functions of the form in (2.9) is that of submodularity (see Kolmogorov and Zabih (2004)), i.e. the

pairwise potentials should satisfy:

$$V_{ij}(0,0) + V_{ij}(1,1) \leq V_{ij}(1,0) + V_{ij}(0,1) \;\; \forall i,j \in \mathcal{N}$$

Intuitively, the submodularity criteria is needed to make sure the energy function is graph representable, i.e it can be transformed into a graph with positive edge weights (see Figure 2.12).

2.2.1 Graph-cuts extensions

The previous discussion was limited to optimisation of energy functions involving binary variables and pairwise terms. The maxflow based algorithms discussed above are however much more general and can be used to solve several extensions of the energy function in (2.9). One extension of the energy function is to handle multiple labels (as opposed to the binary labels discussed so far). Multi-label problems arise in the context of stereo depth estimation(Kolmogorov and Zabih, 2001), semantic segmentation (Shotton et al., 2006; Ladicky et al., 2009; Winn and Shotton, 2006) etc. A generic energy function for a CRF with multiple labels containing unary and pairwise terms can be written as :

$$E(\mathbf{y}) = \sum_i U_i(y_i) + \sum_{i,j} V_{ij}(y_i, y_j)$$

where $y_i \in \{1, \cdots, L\}$ and L is the number of labels. For the case of $L = 2$, this is the same as the energy function in (2.9) and can be globally optimised. For the general case of $L > 2$, Boykov et al. (2001) proposed the use of move making algorithms such as $\alpha - \beta$ swaps and $\alpha-$expansions. While the general multi-label

optimisation problem is NP-hard (Veksler, 1999), these move making algorithms are able to obtain local minima within fairly large move spaces. The general idea in Boykov et al. (2001) is to break the multi-label problem into smaller binary subproblems, where each subproblem corresponds to a particular move between two labels. These binary subproblems are submodular (under certain conditions on the pairwise terms described next) and can be solved exactly using graph-cuts. In order for the binary subproblems obtained in the $\alpha - \beta$ swap algorithm to be submodular, the pairwise potential $V_{ij}(y_i, y_j)$ needs to be a semi-metric – i.e $\forall \alpha, \beta \in \{1, \cdots, L\} : \; V_{ij}(\alpha, \beta) = V_{ij}(\beta, \alpha) \geq 0$ and $V_{ij}(\alpha, \beta) = 0 \Leftrightarrow \alpha = \beta$. For the α-expansion algorithm, $V_{ij}(y_i, y_j)$ needs to be a metric, i.e in addition to the semi-metric properties just described above, the triangle inequality also needs to be satisfied: $\forall \alpha, \beta, \gamma \in \{1, \cdots, L\} : \; V_{ij}(\alpha, \beta) \leq V_{ij}(\alpha, \gamma) + V_{ij}(\gamma, \beta)$. For the case of α-expansions, Boykov et al. (2001) prove that the local minimum achieved is within a factor of 2 of the global minimum.

There is also some work on other move making algorithms to solve the general multi-label case such as Woodford et al. (2008); Lempitsky et al. (2009) where moves are defined as fusions of proposal labellings. These algorithms generalise the space of moves in Boykov et al. (2001), but the subproblems encountered are not submodular in general. Hence they use more generic solvers such as QPBO (Kolmogorov and Rother, 2007; Boros and Hammer, 2002) to solve them. There are other special cases of multiple label problems that can be solved globally for certain classes of energy. For example, Ishikawa (2003) describes a graph construction to globally optimise multi-label problems where the label set is ordered and the pairwise terms are convex on that ordering. Another special case is presented in the Tiered scene labelling work of Felzenszwalb and Veksler (2010), where a geometric constraint

(a) Generic tiered labelling. (b) Example image with its tiered geometric labelling.

Figure 2.13: *Tiered labellings.* (a) Shows the generic structure of a tiered labelling. It consists of two labels T and B which satisfy the constraint that the T labels are always above the B labels. The other labels have to observe the column structure as shown in the image. These constraints allow the use of dynamic programming to optimise such multi-label problems. (b) An example implementation of tiered scene labelling for labelling different geometric classes such as sky, ground and surfaces above ground. Such scenes naturally satisfy the tiered labelling criteria. Images taken from Felzenszwalb and Veksler (2010).

is imposed on the labelling that allows obtaining globally optimal solutions using dynamic programming (see Figure 2.13 for an illustration).

Another extension of the binary labelling problem involves adding higher order terms that can model more complex interactions between pixels (such as Kohli et al. (2009a,b)). The energy function presented in (2.9) is restricted to pairwise terms. However, not all kinds of interactions can be expressed in the form of pairwise constraints (for example, constraints on the area of the segmentation involves all the pixels in a big clique). Some early work on modelling higher order interactions was presented in Kohli et al. (2009a,b). In this work, the authors introduce potentials over n-pixels at a time (with $n > 2$). The generic form of an energy with higher order potentials is written as follows:

$$E(\mathbf{y}) = \sum_i U_i(y_i) + \sum_{i,j} V_{ij}(y_i, y_j) + \sum_{c \in \mathcal{C}} V_c(\mathbf{y}_c) \tag{2.15}$$

where $U_i(y_i)$ and $V_{ij}(y_i, y_j)$ are the unary as pairwise potentials as before, c is clique of pixels with cardinality > 2 and $V_c(\mathbf{y}_c)$ is the corresponding higher order potential on the clique c. A specific case of $V_c(\mathbf{y}_c)$ termed as robust $\mathcal{P}^n$ potts model is discussed in Kohli et al. (2009b):

$$V_c(\mathbf{y}_c) = \begin{cases} \frac{\gamma_{\max} N(\mathbf{y}_c)}{Q} & \text{if } N(\mathbf{y}_c) \leq Q \\ \gamma_{\max} & \text{otherwise} \end{cases} \tag{2.16}$$

where $N(\mathbf{y}_c)$ counts the number of variables in the clique that do not take the dominant label (the dominant label in a clique c is the label that the maximum number of pixels have). This potential encourages all pixels within the clique c to take the same label. It penalises pixels that do not take the dominant label – this penalty increases linearly until there are more than Q pixels that do not take the dominant label, at which point the cost is truncated to a fixed value γ_{max}. Kohli et al. (2009b) show that by introducing auxiliary variables it is possible to rewrite this higher order potential using pairwise submodular potentials. The energy in (2.15) with $V_c(\mathbf{y}_c)$ as defined in (2.16) can then be optimised globally using graph cuts for the case of binary labels and using $\alpha - \beta$ swaps and $\alpha-$expansions for the case of multiple labels. There is also some recent work on optimising generic higher order functions in Komodakis and Paragios (2009); Rother et al. (2009); Ishikawa (2009).

2.2.2 Optimising the GrabCut model

The discussion in Sections 2.2 and 2.2.1 has been focused on optimising for the segmentation variables $\mathbf{y}$, with several extensions such as multiple labels for $\mathbf{y}$ and higher order cliques on $\mathbf{y}$ being discussed. This section focuses on optimising the

(a) Bounding box interaction (b) GrabCut output (c) Branch-and-mincut output

Figure 2.14: *Globally optimising colour models.* (a) Shows the bounding box interaction in green. (b) `GrabCut` gets stuck in a local minima with its colour models. (c) Branch and mincut (Lempitsky et al., 2008) is able to escape the local minima and find a better colour model and segmentation. Images taken from Lempitsky et al. (2008).

colour models $\mathbf{w}$ in the energy function (2.9). Having a good colour model is very important in getting high quality segmentations, as the colour models directly affect the unary terms. Optimising for colour models in addition to the segmentation labels was first discussed in the `GrabCut` work by Rother et al. (2004). Their optimisation problem can be written down formally as:

$$(\mathbf{y}^*, \mathbf{w}^*) = \arg\min_{\mathbf{y},\mathbf{w}} E(\mathbf{y}, \mathbf{w}; \mathbf{x}, \gamma) \tag{2.17}$$

where the energy function is the same as in (2.9). In `GrabCut`, this inference problem is solved by alternately estimating segmentations $\mathbf{y}$ and colour models $\mathbf{w}$. Minimising (2.17) with respect to $\mathbf{y}$ given $\mathbf{w}$ can be done with a maxflow computation as discussed in detail in Section 2.2. Minimising (2.17) with respect to $\mathbf{w}$ given $\mathbf{y}$ involves fitting Gaussian Mixture Models to the labelled pixels. This can be done using maximum likelihood fitting of parameters for the GMM using, for example, the EM algorithm (Dempster et al., 1977). This alternation requires a good initialisation which is obtained from the user provided bounding box. The user interaction in

addition to providing an initialisation, also provides hard constraints on the space of possible segmentations for $\mathbf{y}$ (as anything outside the bounding box is constrained to be background). To further speed up the algorithm, the minimisation over $\mathbf{y}$ can be done efficiently using the dynamic graph cuts framework of Kohli and Torr (2007) – as the colour models change slowly across iterations, it is more efficient to recycle the graph and flows used in the previous iteration than running graph-cuts from scratch. The alternation in Rother et al. (2004) is only able to achieve local optima due the nature of the optimisation. The branch and min-cut (Lempitsky et al., 2008) algorithm solves the problem of estimating $\mathbf{w}$ in a globally optimal way. The space of colour models is discretised and converted into a tree structure and an efficient branch and bound search is done in this space to find the best possible colour model (see Figure 2.14 for a qualitative comparison to `GrabCut`). Vicente et al. (2009) try to jointly optimise over $\mathbf{y}$ and $\mathbf{w}$ in (2.17) where they represent $\mathbf{w}$ as histograms over quantised colours. The variable $\mathbf{w}$ is eliminated from the optimisation by substituting its closed form solution in terms of the variable $\mathbf{y}$. This leads to higher order terms in the energy function which are NP-hard to solve. They compute approximate solutions to the energy function using Dual Decomposition based techniques (Bertsekas, 1999).

There is also some related work on parametric maxflow (Kolmogorov et al., 2007) that allows for optimising for several values of γ in (2.9) very efficiently. This allows for learning the optimal γ for a dataset very efficiently. Using parametric maxflow, it is also possible to solve a broader class of energy functions (such as certain ratio functionals).

2.3 Superpixel segmentation

Sections 2.1 and 2.2 discussed various interactive segmentation formulations and their optimisation in detail. Now we shift our attention to non-interactive methods, starting with bottom up groupings of pixels called superpixels. Ideally, these superpixels capture objects or parts of objects and do not cross object boundaries, thus providing for a sensible grouping of pixels upon which to apply top down learning methods. Figure 2.15 shows many different ways of obtaining superpixels for the same image.

The problem of super-pixelisation in general is ill-defined as there is no one correct way of super-pixelisation. As discussed in Chapter 1, the right segmentation depends on the task at hand. Most of the work on super-pixelisation is done with the aim of reducing the number of computational units from pixels to superpixels while also providing good regions of support for computing features. Martin et al. (2001); Arbeláez et al. (2011) introduced a way to quantify the quality of superpixels through the Berkeley Segmentation Datasets (BSDS-300 and BSDS-500) . Though their work is focused on contour detection and evaluating the quality of contours, super-pixelisations can also be evaluated in their framework as each super-pixelisation can be transformed into closed contours in the image. Ground truth contours for their dataset is obtained manually using human annotators. To make this task well defined, the annotators are asked to divide the image into certain number of segments, with each segment representing an object or a part of an object. Each image is labelled by 5–10 different annotators. While there are disagreements across the annotators, the authors observed a high degree of consistency for most of the contours. Defining the task precisely thus helped to nail down the evaluation

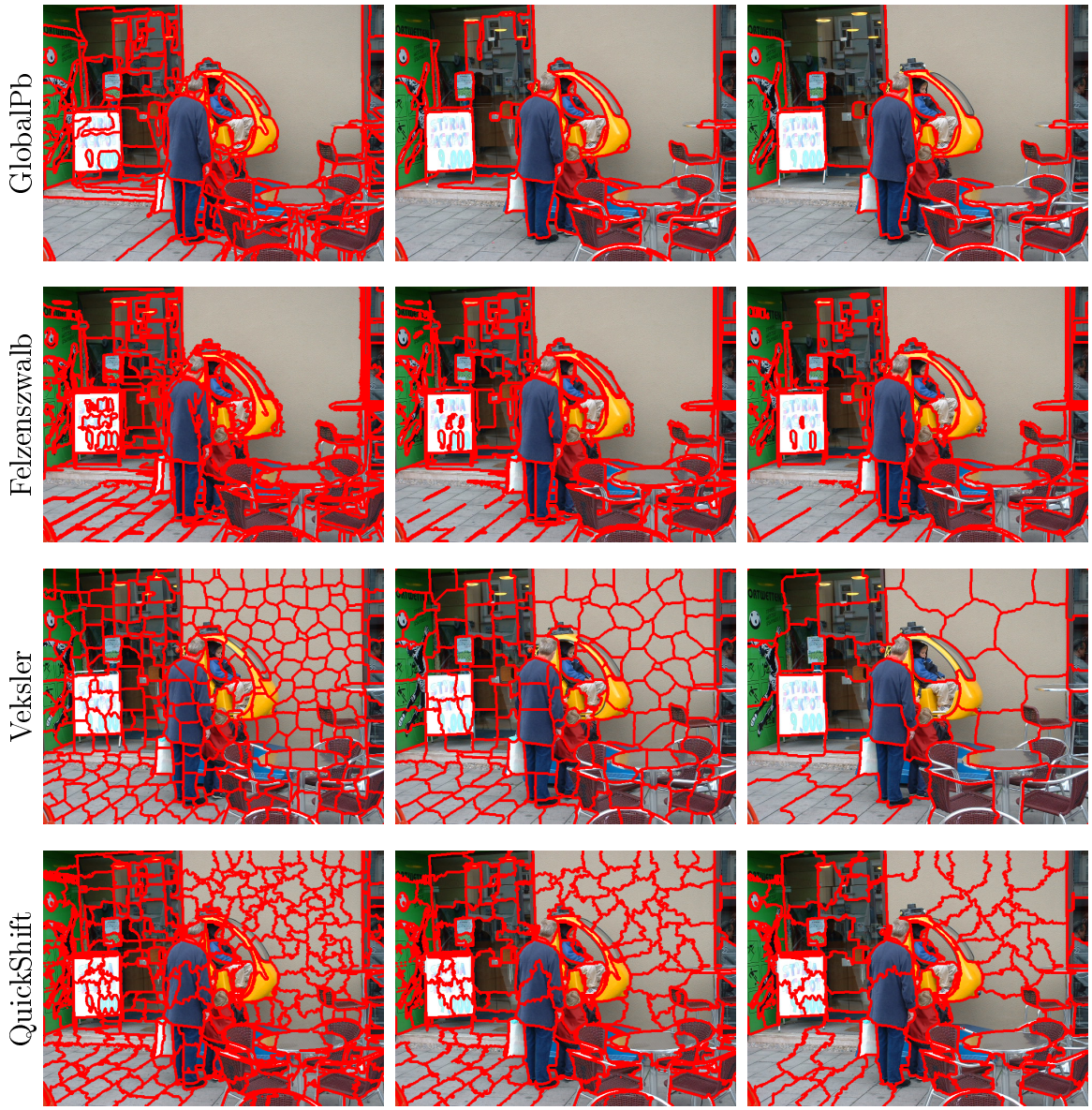


Figure 2.15: *Different superpixel methods at multiple scales.* Each row shows a different super-pixelisation method at three different scales. The superpixel boundaries are outlined in red. The leftmost column has many small superpixels whereas the rightmost column has much larger ones. The scale can be controlled by varying appropriate parameters for each super-pixelisation method. Key: *GlobalPb* refers to Arbeláez et al. (2011), *Felzenszwalb* refers to Felzenszwalb and Huttenlocher (2004), *Veksler* refers to Veksler et al. (2010) and *QuickShift* refers to Vedaldi and Soatto (2008). These methods are discussed in detail in Section 2.3.

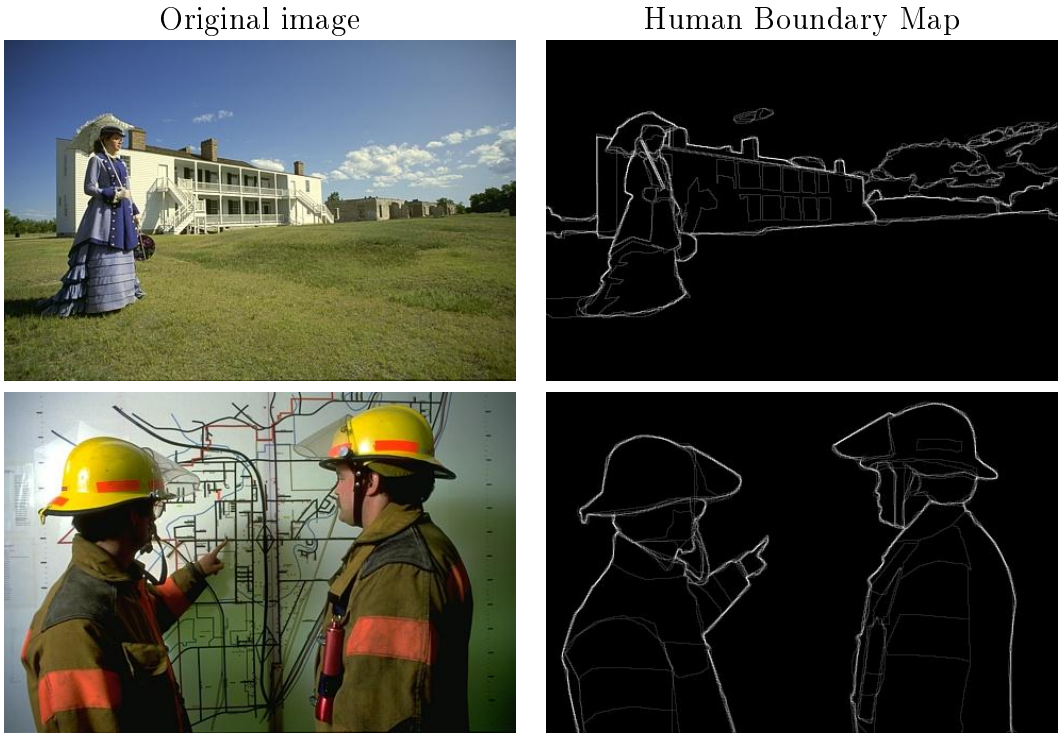


Figure 2.16: *Ground truth contours in the BSDS dataset.* The image on the right shows the averaged contour maps drawn by different human subjects for the images shown on the left. Higher intensity boundaries mean all humans agreed on the boundary whereas the lower intensity boundaries are those where fewer humans agreed. Image courtesy: http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/BSDS300/html/dataset/images.html

problem (with some ambiguity which is handled during the evaluation procedure). Figure 2.16 shows two such ground truth contour maps on images from the dataset. Using these ground truth contour maps, the precision and recall of boundaries can be computed for a particular super-pixelisation as described in Martin et al. (2004). Most super-pixelisation methods have a free parameter controlling the number of superpixels that can be varied to obtain precision recall curves. The F-measure ($F = 2PR/(P + R)$) is used to trade off precision(P) and recall(R) and the best F-measure along the curve is used to compare different methods.

We now discuss various super-pixelisation methods proposed in literature and used later in Chapter 6. Most of the popular and effective super-pixelisation methods are graph based (Shi and Malik, 2000; Arbeláez et al., 2011; Felzenszwalb and Huttenlocher, 2004; Veksler et al., 2010), and tend to have global energy formulations underlying them. One of the most influential work amongst graph based formulations is the normalised cuts work by Shi and Malik (2000). They formulate a cost function for partitioning nodes in a graph into two disjoint partitions. The main idea is to keep the cost of the partition properly normalised so as to avoid the shrinking bias associated with minimum cut partitions (Wu and Leahy, 1993). More formally, each pixel in an image corresponds to a node in the graph and an edge between two pixels u and v is weighted by their similarity denoted by $w(u, v)$. The edges are sparse and a pixel is only connected to pixels within a small spatial neighbourhood. The normalised cuts cost for partitioning the vertex set V into two disjoint sets A and B can then be written as:

$$\text{Ncut}(A, B) = \frac{\text{cut}(A, B)}{\text{assoc}(A, V)} + \frac{\text{cut}(B, A)}{\text{assoc}(B, V)} \tag{2.18}$$

where $\text{cut}(A, B) = \sum_{u \in A, v \in B} w(u, v)$ computes the cost of edges going across the partition and $\text{assoc}(A, V) = \sum_{u \in A, v \in V} w(u, v)$ is the total cost of edges going between A and all the nodes in V. The fraction $\frac{\text{cut}(A,B)}{\text{assoc}(A,V)}$ essentially computes the ratio of edges that is broken for set A to the total cost of edges that have one vertex in set A. The above cost can be written down as a function of binary indicator variables as shown in Shi and Malik (2000). Minimising (2.18) is NP-hard in general, and Shi and Malik (2000) show that if the binary variables are relaxed to be continuous, the cost can be globally optimised by solving an eigen value problem. The eigen vectors

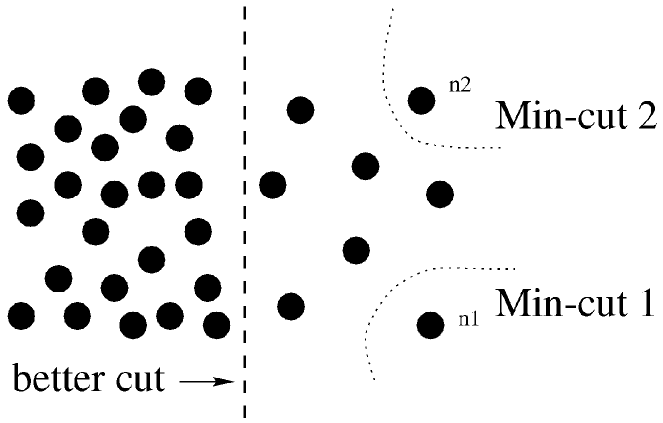


Figure 2.17: *Normalised cuts.* This toy example shows the shrinking problem with min-cuts, as they tend to also minimise the number of edges cut. A better cut can be obtained by minimising the normalised cut criteria in (2.18). Image courtesy Shi and Malik (2000).

of their problem can be interpreted as relaxations of binary indicator functions, and can be thresholded to get the partition. The above scheme can be extended from binary partitions to k-way partitions by recursively applying the same procedure, or other heuristics as described in Shi and Malik (2000).

One of the more recent state of the art super-pixelisation method on the BSDS dataset is the globalPb superpixels of Arbeláez et al. (2011). Their segmentation method builds upon the excellent Pb edge detector of Martin et al. (2004) and combines it with the global cost function of normalised cuts. The Pb detector of Martin et al. (2004) uses a learnt combination of colour and texture cues to obtain high quality contours in the image. These contours are then used to define the edge weights $w(u, v)$ for spectral clustering. The edge weights are computed using the intervening contour cue defined as:

$$w(u,v) = \exp\Big(-\frac{\max_{p\in\overline{uv}}\{mPb(p)\}}{\rho}\Big)$$

(a) Original image (b) gPb contours (c) Superpixels outlined

Figure 2.18: *GlobalPb superpixels.* (a) The original image. (b) Contour confidence values obtained after combining the *Pb* edge detector with the edges obtained from the eigen vectors. (c) Superpixel boundaries at a certain level of the hierarchy visualised. Superpixels are obtained from contours by applying the watershed transform to the contour image to get an over-segmentation and then iteratively merging those segments to generate a hierarchy.

where $mPb(p)$ is the contour detector output of Martin et al. (2004), $\overline{uv}$ is the line segment joining the pixels u and v and ρ is a parameter of the system. The term essentially gives a low similarity score when there is a strong edge crossing the line segment joining u and v. However, unlike Shi and Malik (2000) who threshold the eigen vectors to obtain the partitions, Arbeláez et al. (2011) use the gradients in the eigen vector image to actually obtain another set of edges. The edges obtained from the eigen vectors are combined linearly with the edges from the *Pb* detector to obtain the final contours. These contours are not necessarily closed – they are converted into a segmentation by applying a modification of the watershed transform (Beucher and Meyer, 1992) that they call the oriented watershed transform. These segments are then merged iteratively based on the strength of the shared contour to generate a hierarchy of segments. Figure 2.18 shows the output of the contour detector and the corresponding super-pixelisation.

Felzenszwalb and Huttenlocher (2004) provide an alternative formulation of the super-pixelisation problem. Their formulation attempts to find superpixels where

intra-region variation (i.e the variation within a superpixel) is less than the inter-region variation by a certain margin. This notion is formalised as follows. The image is represented as a graph, with nodes as the pixels and each pixel connecting to its immediate 8 neighbours. Let $w(u, v)$ be a measure of dissimilarity between two neighbouring pixels (such as the contrast). The intra-region variance of region R (denoted as $Int(R)$) is defined as the largest weight in the minimum spanning tree of the region R. The inter-region variance $Dif(R_1, R_2)$ between two adjacent regions R_1 and R_2 is defined as the minimum weight edge connecting the two regions. The predicate used to decide whether two neighbouring regions R_1 and R_2 should be merged is given as:

$$Merge(R_1, R_2) = \begin{cases} \text{true} & \text{if } Dif(R_1, R_2) \leq MInt(R_1, R_2) \\ \text{false} & \text{otherwise} \end{cases} \tag{2.19}$$

where the term $MInt(R_1, R_2) = \min\big(Int(R_1) + \tau(R_1), Int(R_2) + \tau(R_2)\big)$ chooses the minimum intra-region variance between R_1 and R_2 – an additional threshold $\tau(R)$ is added to $Int(R)$ to increase the margin between the inter-region variation and the intra-region variation. The threshold function used in Felzenszwalb and Huttenlocher (2004) is simply $\tau(R) = k/|R|$, where k is a parameter that effectively controls the number of superpixels (larger k means larger superpixels as merging regions is more likely). The algorithm initialises each pixel as a superpixel and keeps merging superpixels until no two adjacent superpixels can be merged according to the criteria in (2.19). In practice, this can be done very efficiently in time $O(n \log n)$ (where n is the number of pixels), using the union-find datastructure and exploiting the theoretical properties of Minimum spanning trees.

Veksler et al. (2010) recently introduces another graph based super-pixelisation

(a) Image with superpixel patches. (b) Stitched superpixels

Figure 2.19: *Veksler superpixels.* (a) Image shows three of the many overlapping superpixel bounding patches. The regions $R(l_1)$, $R(l_2)$ and $R(l_3)$ are colour coded and correspond to three different superpixel labels l_1, l_2 and l_3. (b) The corresponding stitching of the three superpixels after optimising the energy function in (2.20). While only three superpixels are visualised in the figure above, the optimisation is actually run on the entire image with all the labels.

formulation, that is expressed as a discrete global multi-label energy function. Each superpixel has its own label and is hard constrained to lie within a particular size square patch as visualised in Figure 2.19. These patches are uniformly distributed along a grid, and the size of the patch determines loosely the size of the superpixels. The energy function encourages these patches to stitch along boundaries with good contrast in a spirit similar to the texture synthesis work of Kwatra et al. (2003). The energy function is written as:

$$E(\mathbf{y}|\mathbf{x}) = \sum_{i=1}^{N} U_i(y_i) + \sum_{i,j \in \mathcal{N}} w_{ij}(\mathbf{x})(y_i \neq y_j) \tag{2.20}$$

where $\mathbf{x}$ denotes the image, $y_i \in \{1, \cdots, L\}$ denotes the superpixel label, L is the number of labels which is the same as the number of patches placed in the image and $w_{ij}(\mathbf{x}) = \exp(-\beta\|x_i - x_j\|^2)$ is the usual contrast dependant penalty. The

unary term $U_i(y_i)$ simply imposes the hard constraint that each superpixel label l lies within its pre-allocated patch $R(l)$:

$$U_i(y_i = l) = \begin{cases} 0 & \text{if } i \in R(l) \\ \infty & \text{otherwise} \end{cases}$$

Global optimisation for the above energy is NP-hard in general, but good solutions can be obtained using the alpha expansion algorithm discussed in Section 2.2.1, which give a factor of 2 approximation to the energy function.

Another category of super-pixelisation methods is the mode-seeking mean shift algorithm of Comaniciu and Meer (2002). The algorithm attempts to find peaks in a probability density function, given samples drawn from that distribution. It uses a parzen window estimator to approximate the probability density of features – for the case of image segmentation, these features are obtained by concatenating the position vector to the RGB vector at every pixel. It then iteratively finds the modes of this probability density by using a gradient ascent procedure. Each pixel is assigned to its nearest mode to generate a segment. The size of the segment can be controlled by the bandwidth parameter of the kernel used in parzen window estimator. Vedaldi and Soatto (2008) present an efficient algorithm called quick-shift which is based on a similar principal to mean shift. Rather than iteratively finding modes of the density function, the algorithm works by connecting each pixel to its nearest neighbour for which there is an increment in probability density. This leads to a tree like structure, whose modes can be recovered by breaking branches longer than a certain threshold.

2.4 Semantic segmentation

The previous section discussed purely bottom-up segmentation methods that were agnostic to the object categories in the image. This section discusses using top-down knowledge of the object categories to label all pixels in an image with their corresponding categories. The number of object categories depends on the problem definition. Some datasets (such as the Graz dataset by Opelt et al. (2006)) restrict it to a binary labelling problem where the task is to label pixels either as those of a particular object category and or as background. Other datasets have multiple classes (upto 20 in the PASCAL VOC dataset by Everingham et al. (2010)).

One of the earliest works on learning for segmentation is by Borenstein and Ullman (2002, 2004) who tackle the problem of segmenting a particular object category in an image. The training process in Borenstein and Ullman (2002) consists of selecting a large number of candidate image fragments from images containing the object category. Detection of these fragments is done by sliding them on an image and scoring using normalised cross correlation. The detection threshold for each fragment is set such that its false positive rate in non-object images is below a certain parameter α. Each fragment is also assigned a reliability value depending on how often its detected in object images. The corresponding figure-ground labels for these fragments are also stored as part of the training. At test time, each fragment is matched to the image and combined with a criteria for optimally covering the image with fragments (see Figure 2.20). The criteria maximises a trade-off between individual match quality, fragment reliability and consistency between overlapping fragments. Borenstein and Ullman (2004) extended this work by learning fragments and their corresponding figure-ground labels in an unsupervised fashion. Leibe and

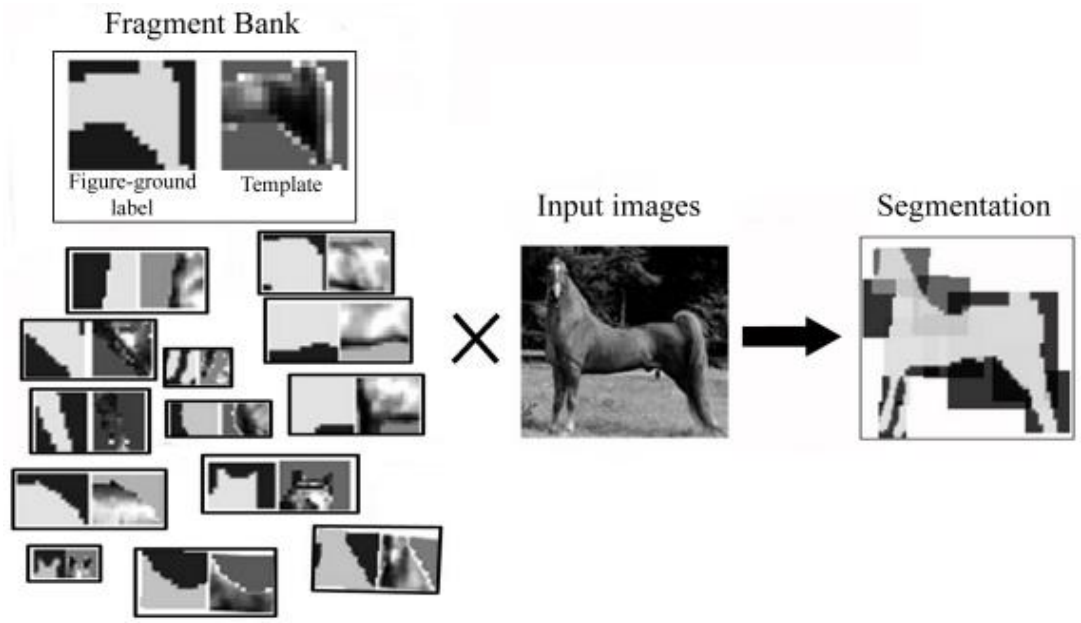


Figure 2.20: *Learning to segment of Borenstein and Ullman (2002).* The fragments in the trained fragment bank are convolved with the test image to give individual fragment matches. These matches are then combined with their reliability scores and consistency with overlapping fragments to give a covering for the test image shown on the right. Image courtesy Borenstein and Ullman (2002).

Schiele (2003) propose a similar system where they learn a codebook of image fragments using clustering and object detection is performed using a hough transform. The segmentation is then generated by overlaying segmentation masks of matched codebook entries. Levin and Weiss (2006) extend this idea to learn a set of optimal fragments in a CRF framework by combining bottom up and top-down information.

`ObjCut` (Kumar et al., 2005) also combines top-down cues with bottom up cues in a single energy function. The top down cues are provided by samples from a Layered pictorial structure model (Kumar et al., 2004). These samples impose a shape prior in the form of a unary term in an energy function that combines the shape prior with contrast dependent pairwise terms (see Figure 2.21). Another related work is `PoseCut` (Bray et al., 2006) where an estimate of the human body pose is used

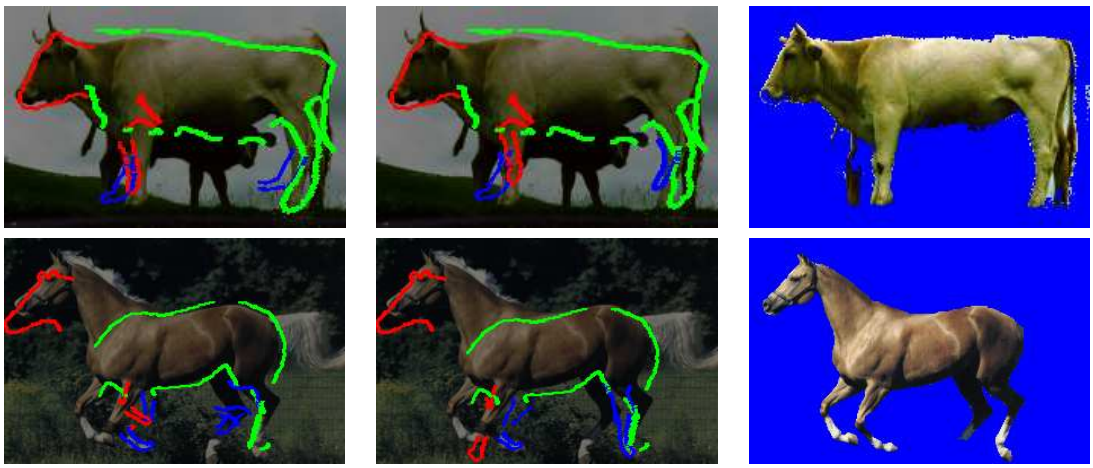


Figure 2.21: *ObjCut segmentation.* The left two columns show samples from the Layered pictorial structure model used in `ObjCut` (the colours encode different parts of the pictorial structure). These samples are used to provide a shape prior, and combined with colour models from the image to give the final segmentation in the rightmost column. Image courtesy Kumar (2008).

to provide a shape prior term for a CRF energy function across multiple registered views.

Most of the work discussed above focused on segmenting a single category from an image. Since the introduction of the PASCAL VOC 2007 Dataset (Everingham et al., 2007) and the MSRC segmentation dataset (Shotton et al., 2006), there has been significant work on the problem of segmenting multiple categories from the same image as well. One line of work has been towards adapting object detectors to guide segmentation. For example, Yang et al. (2010) use the output of the part based detector of Felzenszwalb et al. (2009) to give rough shape priors for object segmentation (see Figure 2.22). These shape priors are combined with image specific colour models and object layering priors based on detection scores. They also exploit low level grouping by using superpixels obtained from Arbeláez et al. (2011) as their labelling units. Another related work on using object detectors for segmentation is

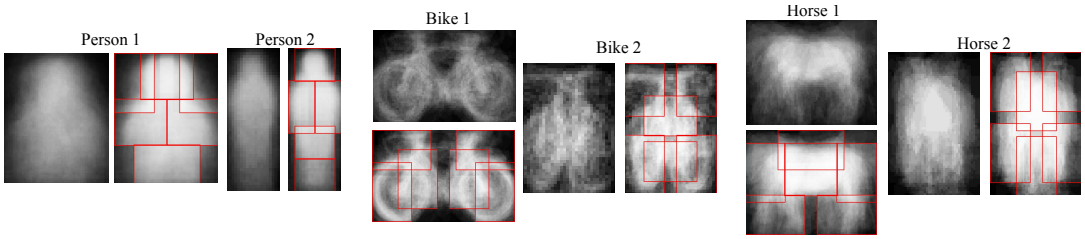


Figure 2.22: *Shape priors obtained from object detection.* The object detector of Felzenszwalb et al. (2009) trains different models for different aspects of the object. Each model consists of a deformable mixture of parts – each part learns its own soft mask which can then be overlaid to obtain a soft mask for the entire object detection. Image courtesy Yang et al. (2010).

by Brox et al. (2011) that builds upon the `Poselets` (Bourdev and Malik, 2009; Bourdev et al., 2010) detector. `Poselets` can be thought of as object parts that are tightly clustered in appearance and pose space. This makes individual poselets easier to detect, and individual detections of different poselets can be combined together to give full object detections as discussed in Bourdev et al. (2010). Brox et al. (2011) uses the detected poselets to predicts soft segmentation masks in a similar spirit to Yang et al. (2010). These soft masks are then aligned to the object boundaries detected using the globalPb detector of Arbeláez et al. (2011). This algorithm is extended to handle multiple classes by competing for pixels across multiple poselet detections.

Another line of work on segmenting multiple object categories is based on CRF formulations and attempts to label pixels directly with corresponding categories as opposed to using object detections to initialise segmentations. The TextonBoost (Shotton et al., 2006) work proposes a CRF based formulation for semantic segmentation, where the unary potentials are obtained by combining appearance, shape and context based features. One of the main contributions of their work is the introduction of Texture-layout features for computing unary terms (see Figure 2.23).

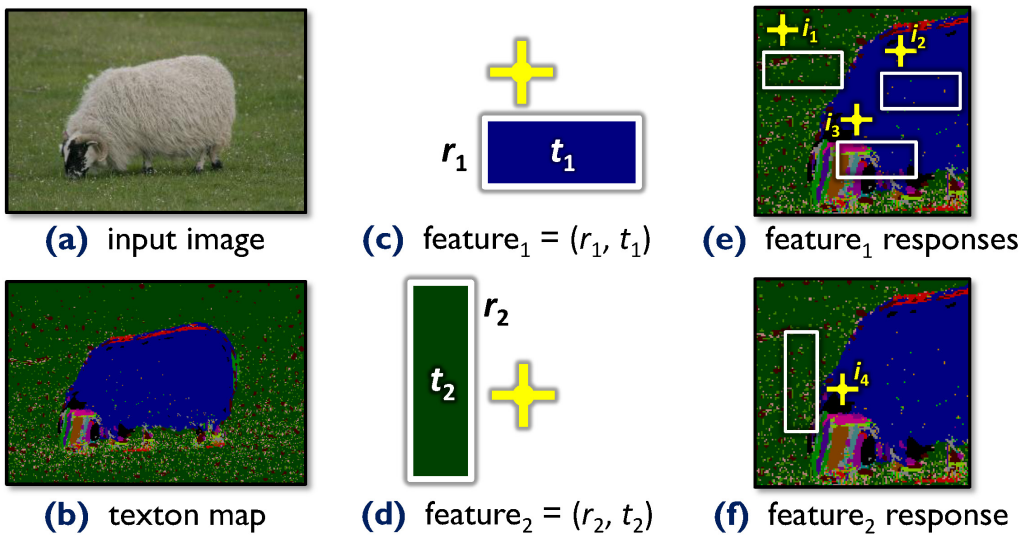


Figure 2.23: *Texture-layout features of Shotton et al. (2006).* The texton map shown in (b) is an indexed image that maps the texture descriptor computed at each pixel onto its nearest visual word. Each entry in the texture-layout feature is computed as the proportion of a particular texton in a rectangular region that is offset by a certain distance from the pixel. Two such rectangles are shown in (c) and (d). (e) shows the feature computation at three different locations indexed by i_1, i_2 and i_3 (the corresponding feature$_1$ responses are roughly $0, 1$ and 0.5 respectively). Image courtesy Shotton et al. (2006).

The texture layout features are simply counts of vector quantised texture descriptors at varying offsets from the pixel. As these counts are computed at various offsets from the central pixel, they are able to capture context information effectively also. Discriminative classifiers are learnt from these texture features using JointBoost (Torralba et al., 2007) to give unary terms for segmentation. These potentials are then combined with per-image colour models and location potentials in the following energy formulation:

$$E(\mathbf{y}|\mathbf{x},\mathbf{w}) = \sum_i U^i_{\text{texture}}(y_i|\mathbf{x},\mathbf{w}_{\text{texture}}) + \sum_i U_{\text{colour}}(x_i,y_i|\mathbf{w}_{\text{colour}}) + \\ + \sum_i U_{\text{position}}(i,y_i|\mathbf{w}_{\text{position}}) + \sum_{i,j\in\mathcal{N}} V(y_i,y_j|x_i,x_j,\mathbf{w}_{\text{edge}})$$

where $y_i \in \{1,\cdots,L\}$ is the output class label at each pixel, $\mathbf{x}$ is the image, $U_{<\text{cue}>}$ are the various unary terms for different cues such as texture, colour, position and $V(y_i,y_j|x_i,x_j,\mathbf{w}_{\text{edge}})$ are pairwise terms that combine the standard Ising potential with contrast dependant cost. $\mathbf{w}_{<\text{cue}>}$ corresponds to the parameters for each cue, and these parameters are trained separately using piecewise training (except for $\mathbf{w}_{\text{colour}}$ that is estimated on a per-image basis). At test time, inference is done using the alpha expansion algorithm (Boykov et al., 2001) described in Section 2.2.1. Shotton et al. (2008) later proposed another set of more efficient texture features called semantic texton forests. Semantic texton forests are random forests (Amit and Geman, 1997) trained using simple pixel wise comparison features. Each pixel in an image is then assigned to a leaf node in each tree of the random forest and a corresponding class distribution can be obtained by averaging the class distributions of its assigned leaf nodes. The leaf node indices along with the class distribution replace the textons used in Shotton et al. (2006). Counts of these features aggregated over local regions in the same way as Shotton et al. (2006) are then used as features into another random forest classifier that outputs class probabilities for these features. Replacing the use of filter banks, visual word assignment and boosting based classifiers with random forests makes their algorithm computationally very efficient, and this enables them to even run their segmentation at real-time speeds on videos. Another semantic segmentation work using a CRF formulation is the Layout CRF

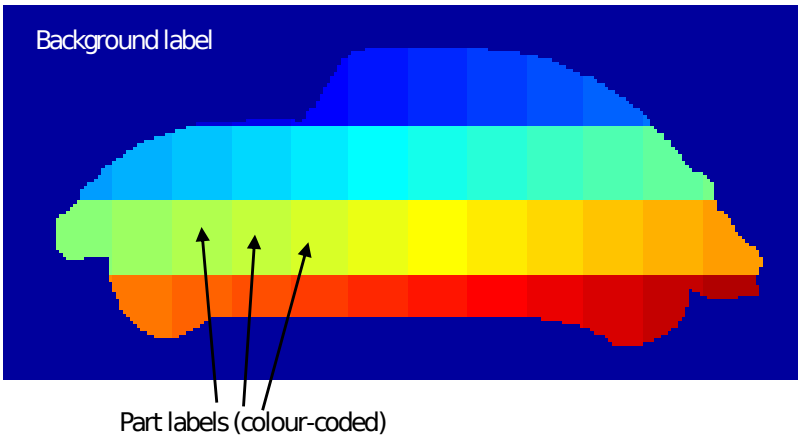


Figure 2.24: *Spatial parts in a Layout CRF*. Different colours denote different spatial parts of the object. The parts above are obtained by laying over a uniform grid on the object shape. The training procedure allows for deformation of these parts to match the particular training instance. The pairwise potentials in the Layout CRF ensure that the relative ordering of these parts remains the same (for e.g the yellow labels remain below the blue labels, the red labels remain to the right of orange labels etc). Image courtesy http://johnwinn.org/Presentations/presentations/LayoutCRF_Expanded.ppt

by Winn and Shotton (2006), that adds a loose spatial model of the object using latent variables in a CRF (see Figure 2.24). These latent variables are used to model spatial parts of the object, and their relative layout is enforced using asymmetric pairwise terms in the CRF. The Layout CRF also models multiple instances of the same object by allowing for different labels for different instances.

There have also been a number of successful semantic segmentation methods based around classifying superpixels. One of the simplest methods by Pantofaru et al. (2008) learns a discriminative classifier to predict the label of each superpixel. Features such as colour histograms, quantised SIFT descriptors and image position are used to train non-linear SVM classifiers to predict class labels for each superpixel. Multiple super-pixelisations per image are used and a separate classifier trained for each super-pixelisation. At test time, the classification output for each pixel is

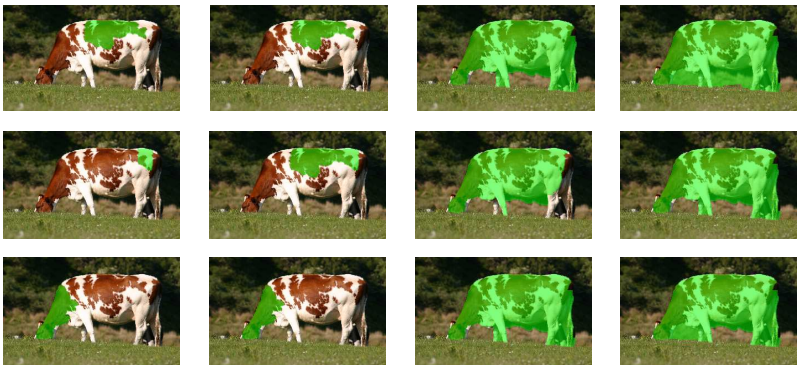


Figure 2.25: *Figure ground hypothesis generated using Carreira and Sminchisescu (2010).* Multiple hypothesis are generated by placing seeds randomly in the image and varying the foreground bias to generate segments of various sizes. Image courtesy Carreira and Sminchisescu (2010).

obtained by averaging the class probabilities of each superpixel it belongs to. Having multiple regions of support for each pixel improves labelling accuracy significantly by providing better regions of support for features and also reducing the chances of superpixels crossing object boundaries. The Associative CRF work of Ladicky et al. (2009) presents a more sophisticated formulation on top of superpixels. Their energy formulation spans the entire hierarchy of segmentations. Different features are computed for different levels in the hierarchy and consistency of labels across the hierarchy is imposed using higher order potentials.

One of the more recent state of the art methods by Li et al. (2010) also uses bottom-up segmentations as a pre-process to obtaining semantic labellings for the whole image. However, unlike Pantofaru et al. (2008); Ladicky et al. (2009), they focus on obtaining holistic figure-ground segmentation hypothesis that try to capture the entire object in a single segment. The constrained parametric mincuts algorithm of Carreira and Sminchisescu (2010) is used to generate a small set of such figure-ground segments (see Figure 2.25). Having such good quality segments reduces

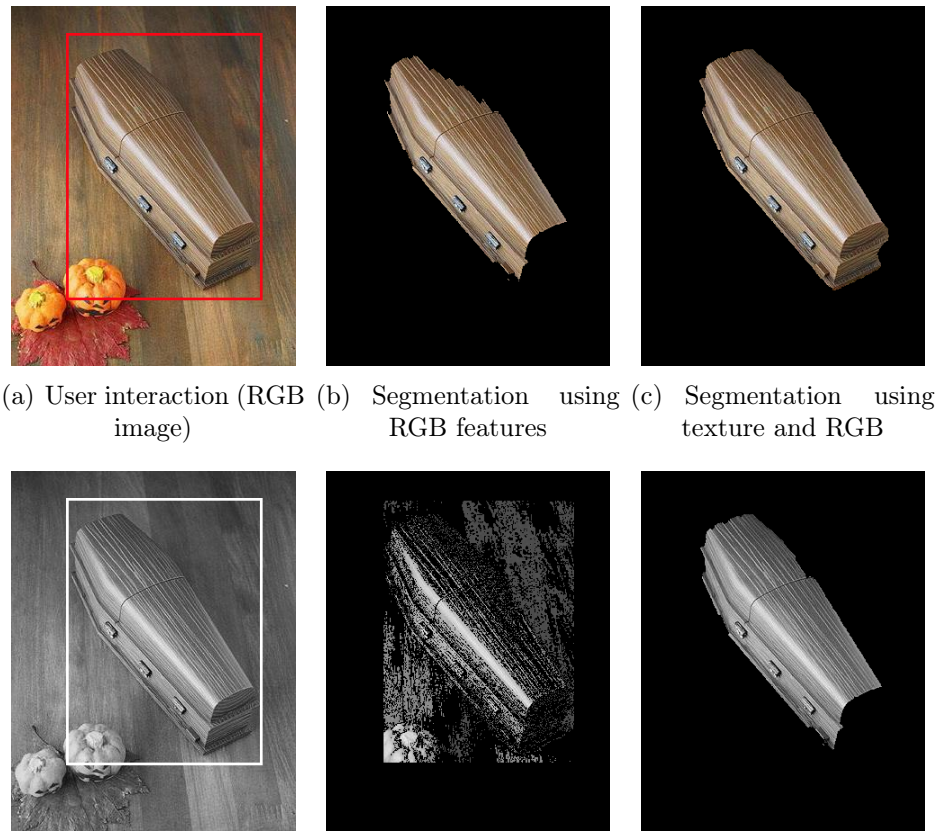
the search space of hypothesis significantly, enabling them to use powerful features and non-linear classifiers. Multiple features based on bag of sift words, HOG and the global Pb boundary detector (Arbeláez et al., 2011) are extracted and used to train a regressor to predict segmentation quality for each hypothesis. The highly ranked image segments are then combined in a final post-processing step to generate semantic labels for the entire image.

Chapter 3

Features for interactive segmentation

This chapter deals with improving upon the low-level cues used in interactive segmentation. In most recent work on interactive segmentation by Boykov and Jolly (2001); Rother et al. (2004); Grady (2005); Criminisi et al. (2008), the image representation is restricted to simple localised pixel based features, such as the RGB colour or monochrome intensity. This is both interesting – that such successful segmentations can proceed from such simple features – and also, as is demonstrated in this chapter, not always sufficient. Figure 3.1, for example, shows a case where texture based features help to generate a more accurate segmentation than just colour pixels. In this chapter, the fusion of texture features with pixel features is explored as a means of increasing the performance of segmentation algorithms. It is shown first that, for many images, pixel colour alone is sufficient to support segmentation, but that including texture features does not diminish this performance. Furthermore, for a certain class of test data termed 'camouflage' images, the inclusion of texture features procures a significant improvement in segmentation performance.

Our texture features are inspired by the the texture patch classifiers of Varma



(a) User interaction (RGB image) (b) Segmentation using RGB features (c) Segmentation using texture and RGB

(d) User interaction (Gray image) (e) Segmentation using Gray features (f) Segmentation using texture and Gray

Figure 3.1: *Segmentation of a 'camouflaged' coffin.* (a) The user specifies a bounding box around the object as in `GrabCut`, and the algorithm segments out the object of interest. (b) Segmentation obtained using RGB values of pixels as features. (c) Segmentation obtained using texture features combined with RGB values. Note the top right corner of the coffin is segmented correctly using the texture feature, but is missed with the RGB features. (d),(e) and (f) Same results using monochrome images and features. More results of such images are shown later in Figure 3.7.

and Zisserman (2003) and Local Binary Patterns (LBP) (Ojala et al., 2002), rather than using filter bank representations used in Leung and Malik (1999); Shotton et al. (2006). Of course, texture has been used for segmentation before: Malik and Perona (1989) used filter responses to find texture boundaries, and texture is one of the cues used for image segmentation within the framework of Normalised cuts by Malik et al. (2001). Galun et al. (2003) also used aggregated filter responses in a multiscale problem for the case of unsupervised image segmentation. More recently, texture features have been used for the task of semantic (i.e. class based) image segmentation, e.g. Shotton et al. (2006) use textons as weak classifiers, and Schroff et al. (2006) use sliding windows over texton maps.

The benefits of combining feature types are well recognised. In image classification, a significant performance boost has been obtained by combining multiple features, e.g. by using a linear sum of kernels (Zhang et al., 2007), and in segmentation, features such as colour and textons have been combined using random forest classifiers (Schroff et al., 2008; Shotton et al., 2008). Here we explore the effect of fusing pixel and texture features in the unary terms of graph-cuts segmentation.

Our findings are supported by quantitative evaluations with tests of statistical significance. To that end two image datasets are introduced for evaluating segmentation performance with various different features. In Section 3.1, a simple extension of the `GrabCut` model to allow for arbitrary features is described. The features used in this work are described in Section 3.2. Datasets and quantitative evaluations are given in Section 3.3 and results are presented in Section 3.4.

3.1 GrabCut model with arbitrary features

Section 2.2 described the energy function for `GrabCut` that uses RGB values as unary features. In this section, notation is extended to allow for inclusion of arbitrary features in the unary terms. Recalling the energy function used in (2.9):

$$E(\mathbf{y}, \mathbf{w}; \mathbf{x}, \gamma) = \sum_{i=1}^{N} U_i(y_i|\mathbf{x}, \mathbf{w}) + \gamma \sum_{(i,j)\in\mathcal{N}} V_{ij}(y_i, y_j|\mathbf{x}) \tag{3.1}$$

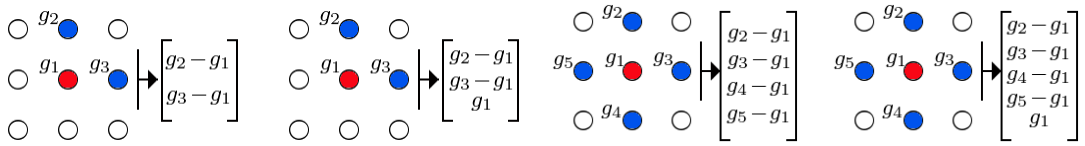
We redefine the unary term as:

$$U_i(y_i|\mathbf{x}, \mathbf{w}) = \begin{cases} -\log(p(f_i(\mathbf{x})|\mathbf{w}_{fg})) & \text{if } y_i = 1 \\ -\log(p(f_i(\mathbf{x})|\mathbf{w}_{bg})) & \text{if } y_i = 0 \end{cases} \tag{3.2}$$

where the term $f_i(\mathbf{x})$ denotes the feature extracted at pixel i. Different feature functions are described in the following section.

3.2 Texture features

Foreground and background appearances are modelled as distributions over features $f_i(\mathbf{x})$, which are then used to compute the feature likelihoods $p(f_i(\mathbf{x})|y_i, \mathbf{w})$. Boykov and Jolly (2001) use the gray scale value of the pixel i for modelling object appearance. We denote this simple feature by *p-gray*. `GrabCut` (Rother et al., 2004) uses the RGB value x_i as the feature $f_i(\mathbf{x})$, termed *p-RGB* here. This set of features is expanded to include features with texture information, based on local patch appearance features of Varma and Zisserman (2003). There, the feature vectors consisted



(a) L-patch (b) L-patch & p-gray (c) '+'-patch (d) '+'-patch & p-gray

Figure 3.2: *Texture features visualised.* The g_i's here denote the gray scale values at the pixels. (a) *L-patch*: This feature is a vector constructed by taking the difference of the gray scale value of non-central pixels with the central pixel. (b) *L-patch & p-gray*: concatenates the *L-patch* feature with the gray value g_i of the central pixel. This fused texture feature is a linear transformation of the feature vector $[g_2; g_3; g_1]$, obtained by subtracting g_1 from the elements. The RGB value of the central pixel may also be used to give the *L-patch & p-RGB* feature. (c) and (d) Similar features on a '+' rather than an L shape.

of the gray values at neighbouring pixels. Our features are chosen to clearly separate texture information from colour information, in order to study how information from these features is fused. Figure 3.2 illustrates some of the texture features and explains the similarity between our fused features and some used previously by Varma and Zisserman (2003). Texture features (like *L-patch* and *'+'-patch*) contain pure pixel-difference information to capture local patterns in the image. Related features have been used in random forest classifiers of Shotton et al. (2008) and also in the Local Binary Pattern (LBP) feature (Ojala et al., 2002). To fuse colour information with those texture features, the central pixel value is simply concatenated with the texture feature to give combined features which are termed as *L-patch & p-gray*, *L-patch & p-RGB*, *'+'-patch & p-gray* and so on. An alternative way of fusing colour information with these texture features would be to treat them as independent features and multiply their likelihoods. Our concatenation of features is more general, subsuming the "naive Bayes" product of likelihoods. In addition to *L-patch* and *'+'-patch*, a feature consisting of the pixels in a whole 3×3 patch is also considered, and termed a *patch* feature. Our motivation to experiment with these different

features was to explore whether texture information could be captured using the lower dimensional features such as *L-patch* and *'+'-patch* without having to take into account the full high dimensional *patch* feature.

3.3 Evaluation

3.3.1 Datasets

For the purpose of evaluation, two image datasets have been set up. The first *Natural image dataset* has 100 images collected from the GrabCut Dataset and the Caltech-256 dataset (Griffin et al., 2007). Note that *p-RGB* is a powerful feature on this dataset, as in most of the images the colour palette of the foreground is well contrasted with that of the background – see Figure 3.3. The other *Camouflage dataset* has 50 images consisting mostly of animals camouflaged with their surroundings. Foreground and background colour palettes are only weakly contrasted in this dataset, so that it provides a much more stringent test of the value of texture as a cue for segmentation – see Figure 3.4. Ground truth for each dataset was marked out by hand. User interaction for these images are specified in annotation files,using two types of annotations: *ribbons* and *boxes*. Ribbon annotations correspond to a thick marker kind of annotation in which the user marks out the boundary using a thick brush. These annotations can be obtained automatically by eroding and dilating the ground truth segmentation. Box annotations, as used in `GrabCut`, consist of a bounding box fitted loosely to the object, and are obtained manually. Figure 3.5 shows these annotations on an image from the *Natural image dataset*.

Figure 3.3: *Natural image dataset* – 12 images out of a total of 100 are shown. The ground truth segmentation is outlined in red. Observe that in all most of these images, the RGB distributions of the object and the background are well contrasted, so that p-RGB is a powerful feature on this dataset.

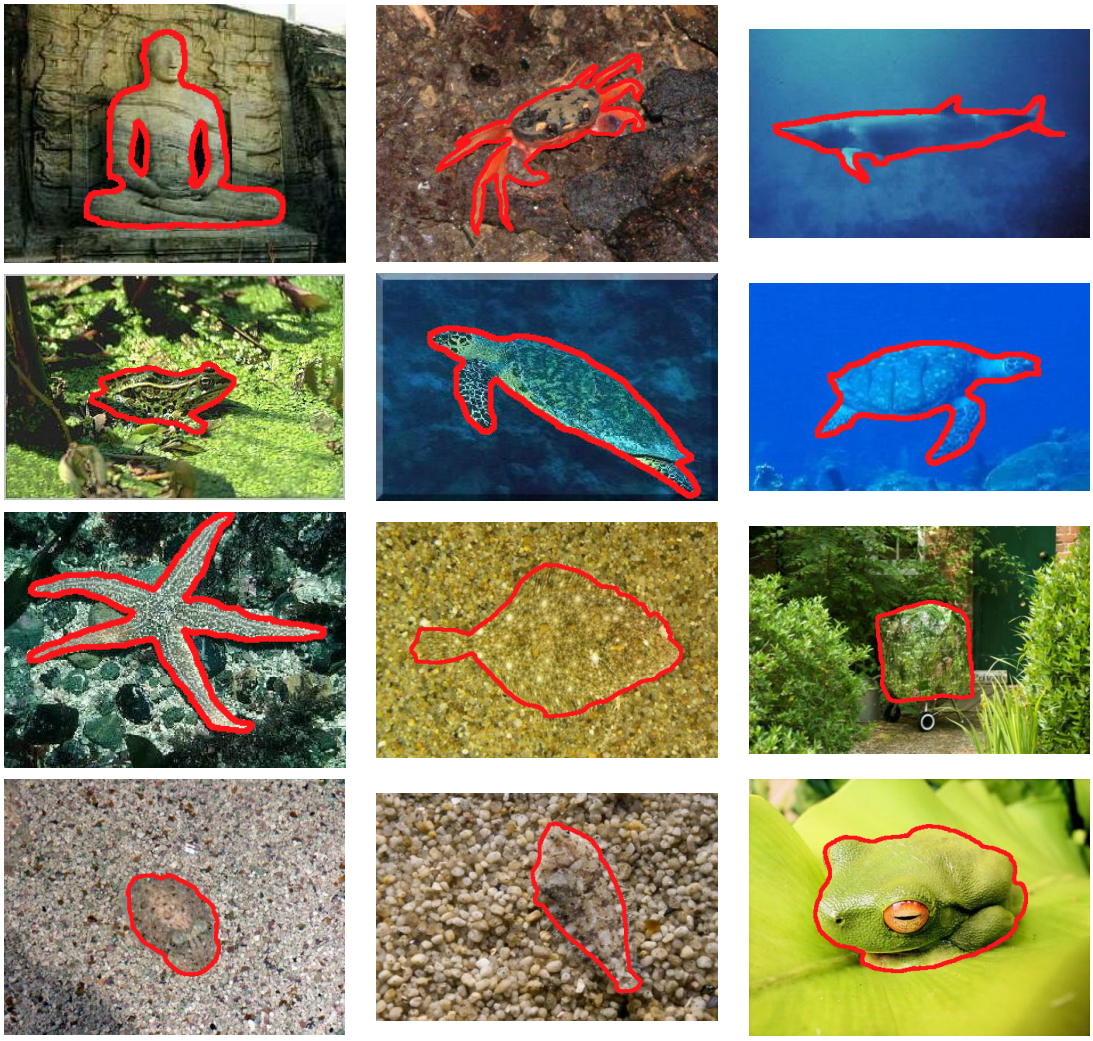
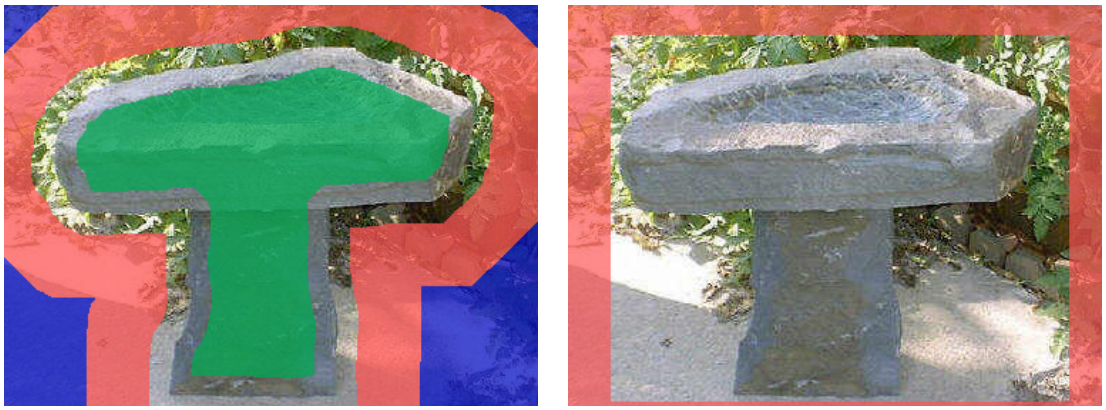


Figure 3.4: *Camouflage image dataset* – 12 images out of a total of 50 are shown. The ground truth segmentation is outlined in red. Note that the RGB distributions of the object and the background overlap to quite some extent in these images.



(a) Ribbon Annotation (b) Box Annotation

Figure 3.5: *Two forms of user annotation.* In the ribbon annotation (a), the light green label is clamped to be foreground. The pink region is clamped to be the background and the blue region is clamped to be background but is not used to learn background appearance. The unlabelled region is the region where the segmentation needs to be inferred. In the box annotation (b), all pixel labels outside the bounding box are clamped to background. Labels inside the box need to be inferred.

3.3.2 Parameter learning

In this section we discuss how to set parameters in the segmentation model, such as the coherence parameter γ which weighs pairwise coherence terms against unary data terms. Other hyper-parameters include the number of Gaussian mixtures, in the case where GMM's are used to model class posteriors in (3.2). To set these parameters, the dataset is partitioned into training and testing sets, each containing an equal number of images. Parameters are estimated on the training set in a coordinate ascent fashion, varying one parameter while keeping others fixed, and choosing the values that maximise performance on the training set. Once the best parameters are found on the training set, they are used to compute test set performance. To get an estimate of the variance in performance on the test set and the sensitivity of parameters, parameter learning and testing is repeated on 10 random

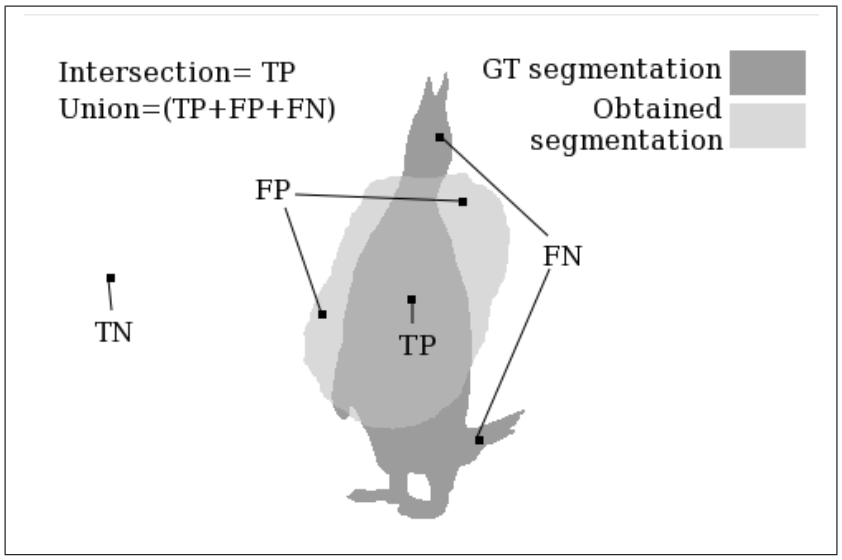


Figure 3.6: *Geometric meaning of overlap score.* Let **gt** denote ground truth labels, **y** denotes segmentation whose score is to be measured. The overlap score is defined geometrically as $\frac{Intersection(\mathbf{y},\mathbf{gt})}{Union(\mathbf{y},\mathbf{gt})}$. From the figure, it is clear that this is equal to $\frac{tp}{tp+fp+fn}$.

training/testing splits of the dataset. This gives 10 test performances from which mean performance together with standard error can be computed.

3.3.3 Performance measure

The performance measure used to evaluate the quality of a segmentation **y** is its overlap score with the ground truth segmentation **gt**. It is defined as:

$$Score(\mathbf{y}, \mathbf{gt}) = \frac{tp}{tp + fp + fn} \tag{3.3}$$

where tp, tn, fp, fn denote true positives, true negatives, false positives and false negatives respectively (positive refers here to the foreground class) — see also Figure 3.6. The use of overlap score, rather than simply evaluating the proportion $\frac{tp+tn}{tp+tn+fp+fn}$ of correct pixels, avoids the problem of undue domination by the la-

belling of background pixels (term tn), many of which are labelled explicitly by the user. The overlap score is more sensitive to segmentation errors in the foreground, as it does not depend on tn. This performance score is related to how much effort the user needs to put it to get the segmentation (s)he wants. A higher score means less effort in further editing as the segmentation is almost correct, whereas a lower score would mean additional interaction from the user is required to get the segmentation correct. The Overlap score is also used as the performance measure in the Pascal VOC2010 Segmentation Challenge (Everingham et al., 2010).

3.4 Results

3.4.1 Natural image dataset

We first discuss results on the *Natural image dataset*, using ribbon annotations. Figure 3.3 shows a few example images from this dataset. The first column of numbers shown in Table 3.2 corresponds to using the ribbon shaped annotations on this dataset. The first thing to note from this column is that there isn't much to be gained in performance by using texture-based features (with either the monochrome feature or the RGB feature). The reason for this is that such ribbon shaped annotations already place strong constraints on segmentation, leaving only a modest degree of ambiguity to be resolved by the segmentation algorithm. For these kinds of annotations, it is even possible to get reasonable segmentations without using any unary terms at all — i.e setting $p(x_i|\mathbf{w}_{fg}) = p(x_i|\mathbf{w}_{bg})$ and using only the contrast-dependent pairwise terms. The segmentation algorithm then finds the boundary with the lowest sum of contrasts, subject to the user clamping constraints specified in the annotation. The performance using such a method is $79.2 \pm 0.4\%$, compared

Feature	D	*Natural Dataset* Ribbons	*Natural Dataset* Boxes	*Camouflage Dataset* Ribbons	*Camouflage Dataset* Boxes
L-patch	2	81.5 ± 0.4%	52.5 ± 0.7%	66.3 ± 0.4%	34.3 ± 0.4%
'+'-patch	4	81.5 ± 0.3%	61.2 ± 0.5%	66.5 ± 0.5%	38.2 ± 0.5%
patch	8	81.7 ± 0.4%	63.3 ± 1.3%	65.9 ± 0.5%	40.6 ± 1.0%
p-gray	1	81.2 ± 0.4%	60.9 ± 0.9%	62.4 ± 0.6%	32.6 ± 0.3%
L-patch & p-gray	3	82.8 ± 0.3%	73.5 ± 1.1%	66.7 ± 0.3%	40.4 ± 0.6%
'+'-patch & p-gray	5	83.1 ± 0.4%	75.1 ± 0.4%	67.1 ± 0.3%	53.3 ± 1.1%
patch & p-gray	9	82.7 ± 0.3%	72.0 ± 1.0%	66.5 ± 0.3%	58.9 ± 2.0%
p-RGB	3	88.3 ± 0.3%	82.6 ± 0.6%	69.4 ± 0.4%	62.5 ± 1.5%
L-patch & p-RGB	5	88.8 ± 0.4%	82.5 ± 0.5%	71.0 ± 0.4%	66.6 ± 1.7%
'+'-patch & p-RGB	7	**88.8 ± 0.4%**	**86.1 ± 0.4%**	**71.2 ± 0.3%**	**69.3 ± 1.2%**
patch & p-RGB	12	88.5 ± 0.3%	81.8 ± 0.5%	69.5 ± 0.5%	69.3 ± 0.8%

Table 3.2: *Comparison of performances across features.* Performance figures are in terms of percentage area overlap. The features are grouped into 3 categories. The first group consists of the first three rows representing the texture features introduced in Section 3.2. The next group consists of the conventional *p-gray* feature, followed by its fusion with the 3 texture features. The final group consists of the conventional *p-RGB* feature followed by its fusion with the 3 texture features. The second column (labelled D) shows the dimensionality of the feature. Results use GMM's to compute the unary data term.

to $81.2 \pm 0.4\%$ using *p-gray* and $88.3 \pm 0.3\%$ using *p-RGB*.

The next set of experiments on the *Natural image dataset* is run with box annotations as opposed to the ribbon annotations. The box annotations provide for harder and less supervised segmentation problems because the unlabelled region is larger, and the the initialisation of the appearance models becomes an issue. Results are reported in the second column of Table 3.2. Now we begin to see some performance difference between conventional pixel features and the fused texture features. The difference is clearly visible for the monochrome features (performance of *p-gray* goes up from $60.9 \pm 0.9\%$ to $75.1 \pm 0.4\%$ for *'+'-patch & p-gray*). In the case of RGB features, a small gain in performance is obtained (performance of *p-RGB* goes up from $82.6 \pm 0.6\%$ to $86.1 \pm 0.4\%$ for *'+'-patch & p-RGB*). Observe that the

reason for not gaining much performance over simple RGB features is that *p-RGB* by itself is a sufficiently powerful feature to obtain good separation of foreground and background for most images in this (well contrasted) dataset.

There are a few images in the *Natural Image dataset* where the fused texture features outperform conventional *p-RGB* features. We observed that in these images colour was not a very strong feature due to the similarity in colours between the foreground object and the background. However, such images are very few (roughly 3 out of 100) and thus the performance gain does not signify in the aggregated results.

3.4.2 Camouflage image dataset

This dataset contains images in which there is significant overlap in colour distributions between the foreground and background. Thus the colour information is weak in these images and that allows texture features to provide information complementary to the colour information. A natural place to look for such images is images of animals camouflaged with their surroundings, and they constitute most of the 50 images in this dataset. Figure 3.4 shows a few examples from this dataset. Results are tabulated in columns 3 and 4 of Table 3.2. First observe that the average performance on this dataset is significantly worse than on the *Natural image dataset*, reflecting the overall level of difficulty of the camouflage dataset (for example, *p-RGB* using box annotations performs at 82.6% on the *Natural image dataset* compared to 62.5% on this dataset). The next observation is that, for the case of monochrome features, the fused texture feature gains significantly over *p-gray*. This is clearly noticeable for the case of box annotations (*p-gray* goes up from 32.6% to 53.5% for *'+'-patch & p-gray*). As mentioned before, box annotations make for a

Feature	D	*Natural Dataset* Ribbons	*Natural Dataset* Boxes	*Camouflage Dataset* Ribbons	*Camouflage Dataset* Boxes
'+'-patch	4	81.5 ± 0.3%	61.2 ± 0.5%	66.5 ± 0.5%	38.2 ± 0.5%
LBP	1	80.1 ± 0.3%	31.1 ± 0.8%	60.1 ± 0.7%	35.6 ± 0.4%

Table 3.3: *LBP performance figures.* This table compares performance of LBP to the *'+'-patch* feature. The *'+'-patch* performances are taken from Table 3.2. The performance of LBP on the *Natural Dataset* with Box annotations is much inferior to the *'+'-patch* feature. However, on the *Camouflage Dataset* with Box annotations, LBP performance is comparable to the the *'+'-patch.*

Feature	*Natural set* Boxes	*Camouflage set* Boxes
p-gray	57.9 ± 0.6%	31.3 ± 0.4%
'+'-patch & p-gray	64.3 ± 0.6%	38.2 ± 0.6%
p-RGB	80.5 ± 1.0%	59.0 ± 1.2%
'+'-patch & p-RGB	81.0 ± 0.6%	67.2 ± 1.2%

Table 3.4: *Performances using logistic classifier.* Performances are only reported for the box annotations in this table. The logistic classifier does not perform as well as the generative unaries – compare with corresponding entries in Table 3.2. On the pixel features – i.e *p-gray* and *p-RGB* it is lower on average by about 2% than the generative method. On the fused features it performs much worse than the GMM unaries. We hypothesise that this is due to the limited quadratic feature mapping which is not good enough to discriminate between the object/background features.

more stringent test than ribbon annotations, and a more sensitive test for the added value of texture features. Even with RGB features, the gains in using the fused texture feature are noticeable now (*p-RGB* goes up from 62.5% to 69.3% for *'+'-patch & p-RGB*). A qualitative comparison of segmentations obtained using *p-RGB* vs. *'+'-patch & p-RGB* features is provided in Figure 3.7.

3.4.3 LBP and Logistic classifier

We also run experiments using the Local Binary Pattern (LBP) feature on the two datasets. Table 3.3 summarises the LBP results and compares it with the *'+'-patch*

feature. The performance of LBP is observed to be noticeably inferior to the '+'-*patch* feature. The thresholding process in LBP throws away a lot of information in order to achieve a high degree of illumination invariance. For our images, such a degree of invariance is not required and causes a loss of discriminative power, thus lowering the performance. Section 3.4.4 outlines the implementation details of LBP.

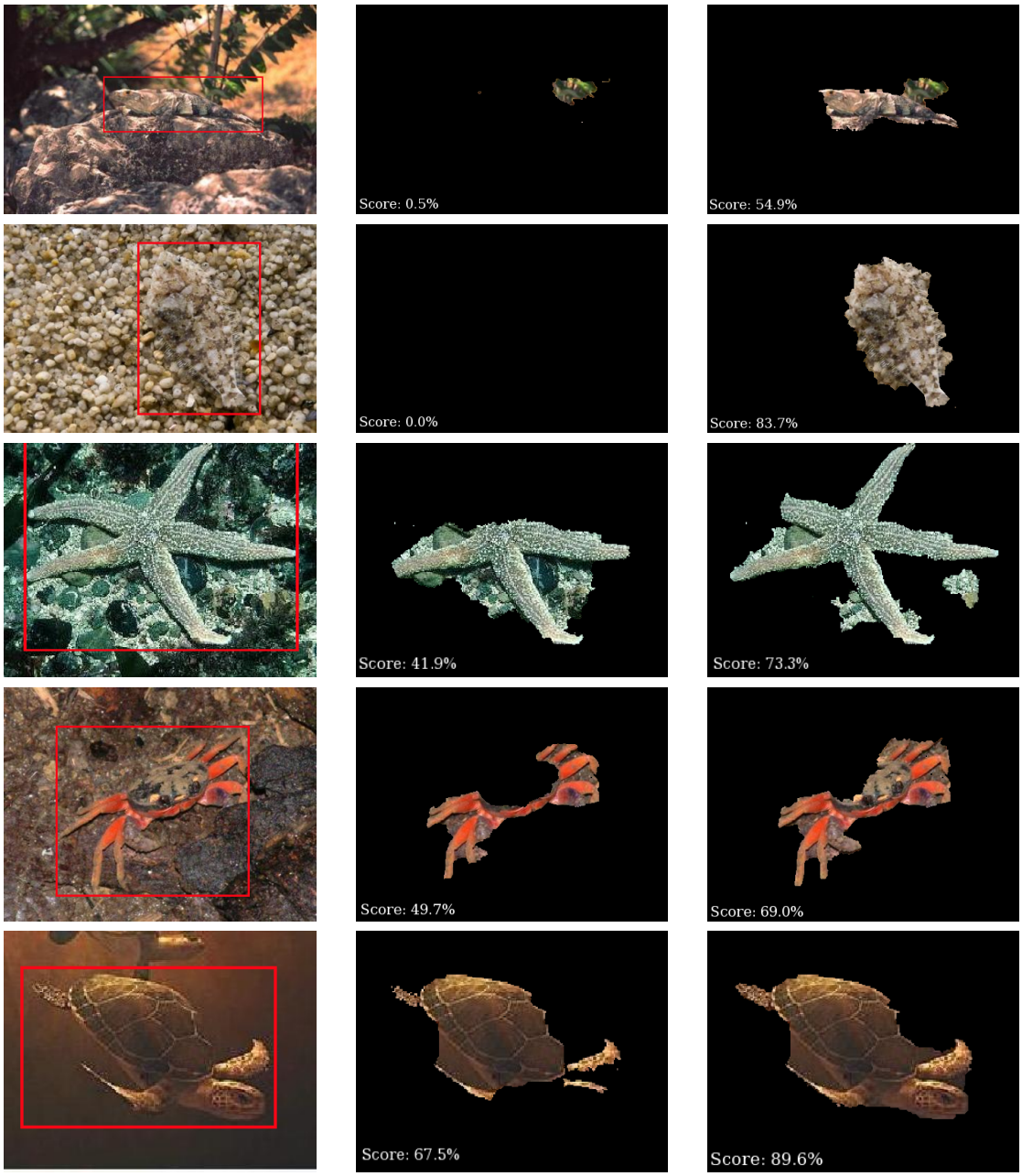
We also report results using the logistic regression classifier to compute the class posteriors. A logistic classifier is used by Kumar and Hebert (2006) for modelling unary terms for the task of segmenting and detecting man-made structures. The unary terms for this case are given by:

$$U_i(y_i|\mathbf{x},\mathbf{w}) = \begin{cases} -\log\Big(\sigma\big(\mathbf{w}^T\phi(f_i(\mathbf{x}))\big)\Big) & \text{if } y_i = 1 \\ -\log\Big(1-\sigma\big(\mathbf{w}^T\phi(f_i(\mathbf{x}))\big)\Big) & \text{if } y_i = 0 \end{cases}$$

where ϕ is a feature space mapping and σ is the logistic sigmoid function. We use a quadratic feature space mapping, which for a two dimensional feature looks like:

$$\phi(\begin{bmatrix} l_1 \\ l_2 \end{bmatrix}) = [1 \;\; l_1 \;\; l_2 \;\; l_1.l_1 \;\; l_2.l_2 \;\; l_1.l_2] \tag{3.4}$$

The parameter $\mathbf{w}$ is learnt using the IRLS algorithm (Rubin, 1983). Results using a logistic classifier are tabulated in Table 3.4. The logistic classifier does not perform as well as the GMM models. We wish to experiment with adding regularisation and exploring other feature space mappings to improve performance of the logistic regression classifier as part of future work.



Original image | Segmentation with *p-RGB* | Segmentation with *'+'-patch & p-RGB*

Figure 3.7: *Qualitative results for p-RGB vs. '+'-patch & p-RGB.* Note that the *p-RGB* segmentation in the second row is empty because the unary terms are not discriminative enough and the pairwise terms cause the segmentation to collapse. The scores indicated on the bottom left of the segmentations are percentage overlap accuracies. These results were obtained on images from the *Camouflage image dataset*. Even though the final segmentations are not totally correct, obtaining a higher overlap score implies less editing effort on part of the user to correct the segmentation.

3.4.4 Implementation details

Parameter Learning

For the *Natural image dataset*, γ is optimised over the discrete set of values $\{20, 50, 75, 100, 150, 200\}$. The number N of Gaussian mixtures is varied over the discrete set $\{3, 5, 7, 9, 11\}$. For the ribbon annotations, there are no additional iterations to re-estimate the colour models as the initial user annotation is already sufficient. For box annotations, at most 10 iterations are used. For the *Camouflage image dataset* γ is optimised over the discrete set $\{0, 20, 50, 75, 100, 150, 200\}$, where 0 is included because, for this set, the foreground can sometimes vanish completely with $\gamma > 0$, in which case $\gamma = 0$ may be optimal.

Gaussian mixture models

GMM's are used to model the feature likelihoods $p'(f_i(\mathbf{x})|y_i, \mathbf{w})$. Given a set of seed pixels to learn the GMM from, the features from the seed pixels are vector quantised into N_{mix} clusters using the binary split algorithm of Orchard and Bouman (1991), and a gaussian fit to each cluster (N_{mix} is the number of gaussian mixtures). The weight of the gaussian mixture is taken proportional to the number of points in the cluster. As we iterate to learn new colour models, the GMM for the current iteration is updated by applying a standard EM iteration (Dempster et al., 1977) to the GMM in the previous iteration.

LBP feature

The LBP feature mentioned in Section 3.4.3 is obtained by thresholding the gray values in a 3x3 patch about the gray value of the central pixel and then encoding the

binary patch as a number. For a 3×3 patch, this corresponds to $2^8 = 256$ possible numbers. The object models are represented as histograms of length 256 and used to compute the feature likelihoods.

3.5 Conclusion

Through our comprehensive quantitative evaluation, we have observed that the inclusion of texture features conveys a decided advantage in the segmentation of monochrome images – i.e when fusing intensity and texture. This is particularly significant with applications where there is no colour, such as working with a monochrome photo stock or with medical images. For many colour images, such as those in the *Natural image dataset*, texture adds an insignificant improvement in performance over simple pixel colour alone. However texture features give substantial gains on colour images where there is significant camouflage between the object and the background, and we demonstrated this quantitatively on the *Camouflage image dataset*.

Chapter 4

Star Convexity and Extensions

While the last chapter deals with improving low-level features for interactive segmentation, here we look at introducing mid-level cues such as shape constraints into segmentation. Our shape constraint is an extension of Veksler (2008)'s star convexity prior, in two ways: from a single star to multiple stars and from Euclidean rays to Geodesic paths. Global minima of the energy function are obtained subject to these new constraints. We also introduce Geodesic Forests, which exploit the structure of shortest paths in implementing the extended constraints. The star convexity prior is used in a interactive system based on a brush stroke interface as opposed to the bounding box interaction in the previous chapter. In a brush stroke interface, the user marks out regions that belong to the foreground and background with brush strokes of adjustable thickness, thus providing constraints for the segmentation and pixels to learn colour models from. Figure 4.1 shows such kind of interaction on two images.

Introducing shape constraints leads to a considerable reduction in user effort. We quantify this reduction in user effort by evaluating our system by means of

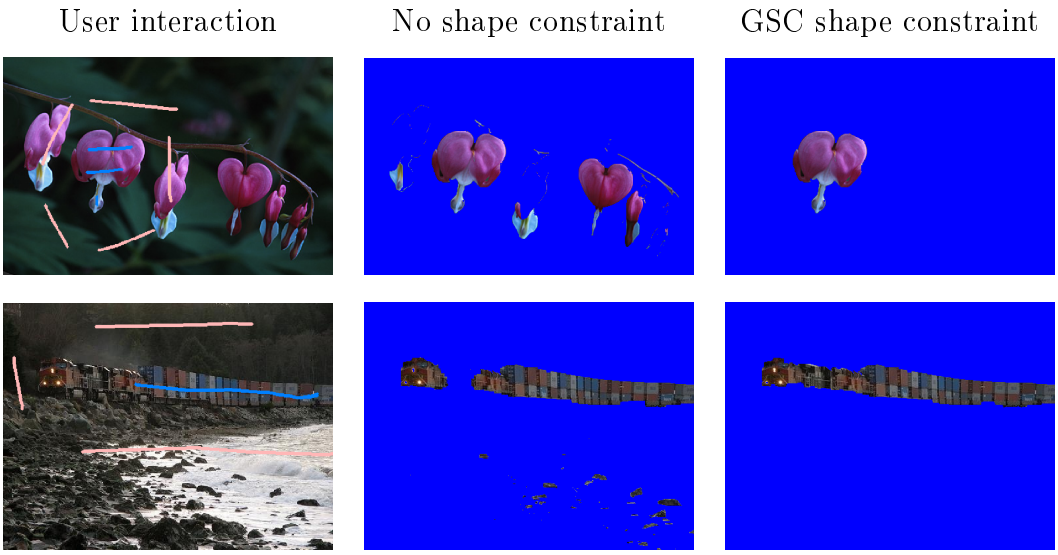


Figure 4.1: *Shape constraints for Interactive Segmentation.* The left most column shows the user interaction, with the blue and pink strokes representing foreground (FG) and background (BG) brushes respectively. The output segmentations obtained using the Boykov Jolly energy formulation (Boykov and Jolly, 2001) are shown in the middle column. The rightmost column shows segmentations obtained by minimising the same energy function subject to Geodesic Star Convexity (GSC) constraints. Notice how the star convexity constraint helps to remove disconnected FG islands, and also to connect up FG islands into a single component.

a "robot user" that measures the amount of interaction required in a precise way. We also introduce a new and harder dataset which augments the existing GrabCut Dataset with images and ground truth taken from the PASCAL VOC segmentation challenge (Everingham et al., 2009).

Figure 4.1 also demonstrates the power of having shape constraints in an interactive segmentation system. Shape constraints such as connectivity can restrict the space of possible segmentations to a smaller subset, helping to eliminate false segmentations. This is the main focus of this chapter: to introduce shape constraints in interactive segmentation, by means of a powerful extension of the star convexity prior proposed by Veksler (2008). Figure 4.2 shows how such constraints also

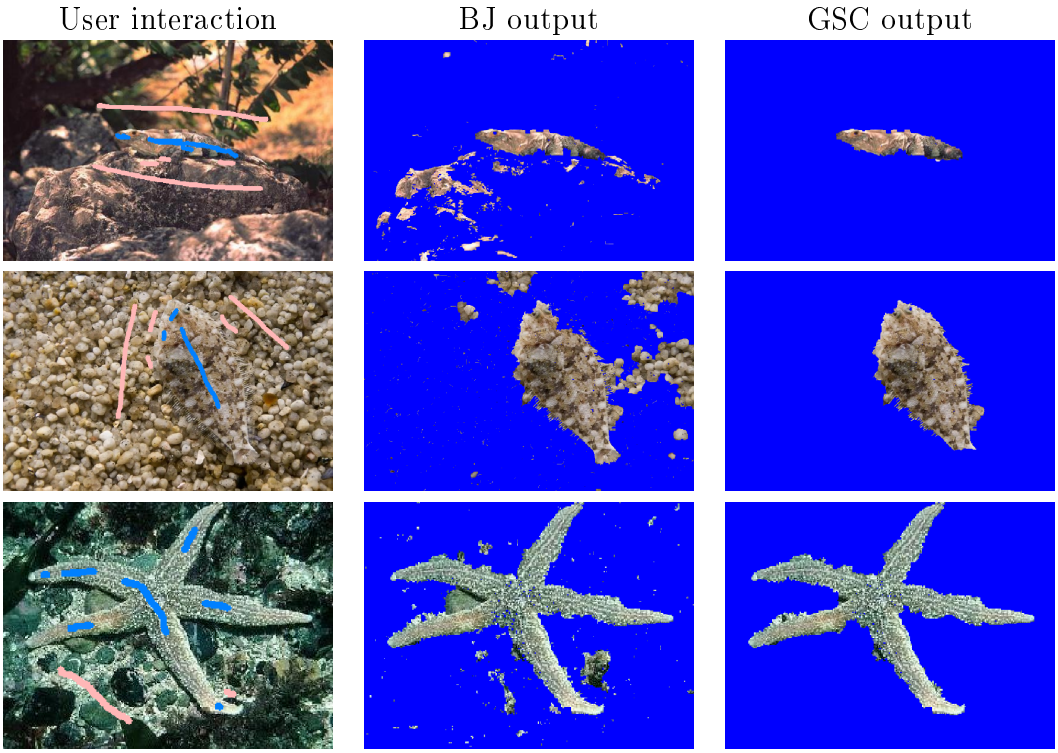


Figure 4.2: *Shape constraints on the camouflage image dataset.* On this dataset, colour models overlap significantly between the foreground and background, thus providing for noisy unaries (unaries are computed using gaussian mixture models over the simple p-RGB feature described in Section 3.2). Shape constraints help to regularise the unaries by restricting the set of allowed segmentations to a sensible set. Interaction effort is thus reduced as the user can achieve the desired segmentation with lesser brush strokes.

improve segmentations on some images from the *Camouflage image* dataset. For the case of camouflage images, the unary terms are noisy and this leads to many regions being segmented incorrectly. With a shape constraint, many of those false segmentations are eliminated while still allowing the desired object to be segmented.

It is well known that shape is a powerful cue for object recognition. Shape models have been used for category specific segmentation (Kohli et al., 2008; Kumar et al., 2005) and help in making the problem well posed. In such applications, shapes are represented as some kind of distance transform from a template, but

such representations are limited and need good initialisations. There has been some work in generic connectivity constraints for segmentation by Vicente et al. (2008) and Nowozin and Lampert (2009). They show that obtaining globally optimal solutions under the connectivity constraint is NP-hard, and then try to find approximate solutions. Connectivity can also be imposed as a post-processing step as done in paint-select (Liu et al., 2009), but that is not principled and cannot be expressed as an energy function. The shape constraint proposed here is closest to Veksler (2008), where a star convexity constraint is imposed on the segmentation, and globally optimal solutions are achieved subject to this constraint. The star convexity constraint also ensures connectivity to brushes, and is a stronger assumption than plain connectivity. In this chapter, we extend the idea of star convexity in a way that gets rid of the limitations of Veksler (2008), and yet being more meaningful than just plain connectivity (Nowozin and Lampert, 2009; Vicente et al., 2008). Note that we still obtain globally optimal solutions while breaking free from Veksler's limitations.

4.1 Star convexity and extensions

This section reviews the ideas of star convex sets (Section 4.1.1) and how this concept is extended to multiple stars (Section 4.1.2) and Geodesic stars (Section 4.1.3).

4.1.1 Single star convexity

Star convex sets have been defined in the geometry and math community (Smith, 1968; Valentine, 1964). Recently Veksler (2008) used such sets as a shape prior in image segmentation. To define star convexity, we first define the idea of visibility. A point p is said to be visible to c via a set $\mathbf{y}$ if the line segment joining p to c lies

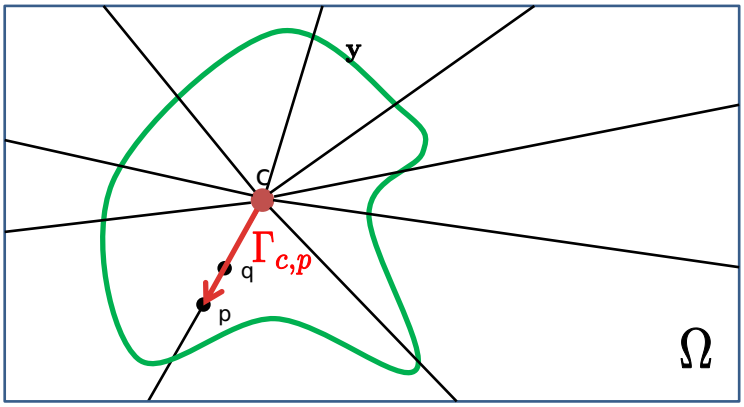


Figure 4.3: A star convex object $\mathbf{y}$ (in green) wrt star centre c. p is any point in the image domain Ω and $\Gamma_{c,p}$ is the straight line segment connecting c to p. An object is star convex wrt centre c if for every point p in the object, all points on the line segment $\Gamma_{c,p}$ also lie inside the object.

in the set $\mathbf{y}$. A set $\mathbf{y}$ is star convex with respect to centre c, if every point $p \in \mathbf{y}$ is visible to c via $\mathbf{y}$ (ref. Figure 4.3). Denote by $S^*(\{c\})$, the set of all shapes which are star convex wrt to centre c. The star constraint is expressed as an energy:

$$E^*(\mathbf{y}|c) = \begin{cases} 0 & \text{if } \mathbf{y} \in S^*(\{c\}) \\ \infty & \text{if } \mathbf{y} \notin S^*(\{c\}) \end{cases} \tag{4.1}$$

The set $\mathbf{y} \subseteq \Omega$ can also be represented as a function $y : \Omega \to \{0, 1\}$, where Ω is the domain of the image and $\forall p \in \Omega : p \in \mathbf{y} \Leftrightarrow y_p = 1$. The energy $E^*(\mathbf{y}|c)$ can then be expressed using pairwise terms on $\mathbf{y}$. As shown in Fig. 4.3, for the line segment $\Gamma_{c,p}$ joining p to c, the star constraints can be written as done in Veksler (2008):

$$\forall q \in \Gamma_{c,p},\ E^*_{pq}(y_p, y_q) = \begin{cases} \infty & \text{if } y_p = 1 \text{ and } y_q = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$E^*(\mathbf{y}|c) = \sum_{p\in\Omega} \sum_{q\in\Gamma_{c,p}} E^*_{p,q}(y_p, y_q) \tag{4.2}$$

It is easy to see that such an energy is submodular as the labelling $(y_p = 1, y_q = 0)$ has an ∞ energy, and will always satisfy the submodularity criteria (ref. Section 2.2.2 for the submodularity criteria). In practice, the domain Ω is discrete, points p and q correspond to pixels and the paths $\Gamma_{c,p}$ are rasterized versions of continuous lines. As discussed in Veksler (2008), the constraint need not be implemented $\forall q \in \Gamma_{c,p}$, but needs to be imposed only between 8-connected neighbouring pixels (p, q), which makes the constraint efficient to implement. It is also possible to define the constraint directly in the discrete domain (see Section 4.1.3) which avoids rasterizing and other implementation issues discussed in Veksler (2008).

The star convexity constraint is quite flexible in the set of shapes it allows as opposed to fixed geometries like ellipses and boxes in Slabaugh and Unal (2005). It includes the set of all convex shapes containing the point c. Figure 4.4 shows examples taken from Veksler (2008) which depict fairly complex objects as being star convex. The same figure also shows limitations of star convexity, which restricts its applications to real images. The next two sections discuss how to extend the star convexity constraint to overcome these limitations.

4.1.2 Multiple stars

A natural extension of single star convexity is to use multiple stars to define a more general class of shapes. This section explores two alternative ways of defining multiple star convexity and explains why the second way is more practical than the first. Consider for simplicity star convexity wrt to two star centres $\{c_1, c_2\}$. The

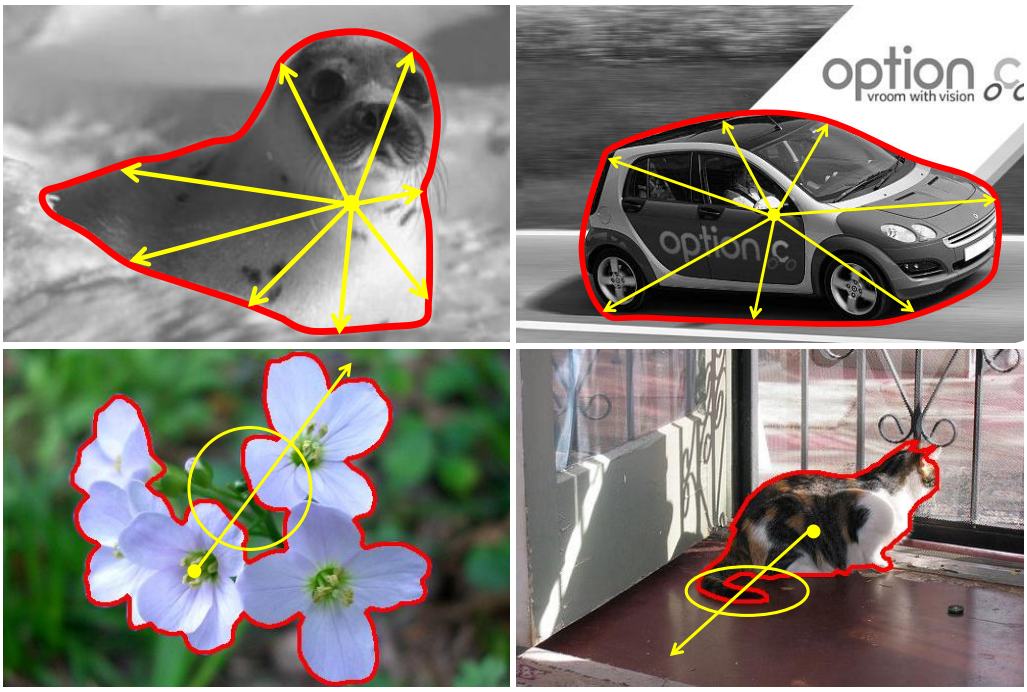


Figure 4.4: *Star convex and objects that are not star convex.* The object of interest is outlined in red. The top row depicts examples of star convex objects, with a plausible star centre marked. The bottom row shows deviations from star convexity, either in the fine details or gross deviations due to multiple object instances. A plausible star centre is shown for such objects with the region that prohibits visibility being encircled.

first definition is an extension of the visibility argument presented in Section 4.1.1 – A set $\mathbf{y}$ is star convex wrt $\{c_1, c_2\}$ if every point $p \in \mathbf{y}$ is visible to at least one of the star centres via $\mathbf{y}$ (ref. Figure 4.5). This set can also be characterised as a finite union of starshaped sets (Toranzos and Cunto, 2004) – A set $\mathbf{y}$ is star convex wrt $\{c_1, c_2\}$ if $\exists \mathbf{y}_1, \mathbf{y}_2 \subseteq \Omega$ *s.t.* $\mathbf{y} = \mathbf{y}_1 \cup \mathbf{y}_2$ and $\mathbf{y}_1 \in S^*(\{c_1\})$ and $\mathbf{y}_2 \in S^*(\{c_2\})$. This definition directly translates to an implementation as:

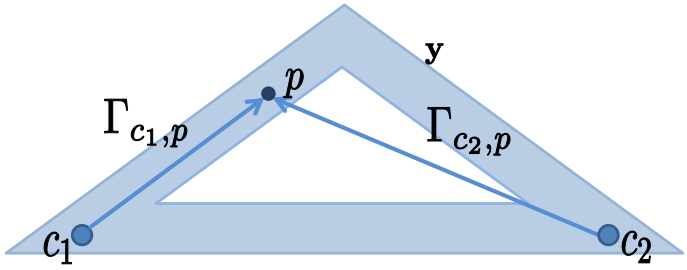


Figure 4.5: *Multiple star semantics – visibility.* $\mathbf{y}$ (shaded blue) is an object which is star convex wrt. centres $\{c_1, c_2\}$, p denotes an arbitrary point in $\mathbf{y}$ and $\Gamma_{c_1,p}$ and $\Gamma_{c_2,p}$ are the two straight lines joining p to the star centres. Every point $p \in \mathbf{y}$ should be visible to at least one star centre for the object to be star shaped wrt $\{c_1, c_2\}$.

$$E^*(\mathbf{y}|\{c_1,c_2\}) = \min_{\mathbf{y}_1,\mathbf{y}_2} E^*(\mathbf{y}_1|c_1) + E^*(\mathbf{y}_2|c_2) + \delta_{\mathbf{y}}(\mathbf{y}_1 \cup \mathbf{y}_2) \tag{4.3}$$

$$\delta_{\mathbf{y}}(\mathbf{y}_1 \cup \mathbf{y}_2) = \begin{cases} 0 & \text{if } \mathbf{y} = \mathbf{y}_1 \cup \mathbf{y}_2 \\ \infty & \text{otherwise} \end{cases}$$

where $E^*(\mathbf{y}_1|c_1)$ and $E^*(\mathbf{y}_2|c_2)$ are as defined in (4.2) and ensure that $\mathbf{y}_1$ and $\mathbf{y}_2$ are star convex wrt to c_1 and c_2 respectively. The $\delta_{\mathbf{y}}(\mathbf{y}_1 \cup \mathbf{y}_2)$ constraint ensures that $\mathbf{y}$ is a union of the two star shapes $\mathbf{y}_1$ and $\mathbf{y}_2$. It is possible to express this energy using pairwise terms between $\mathbf{y}_1$ and $\mathbf{y}_2$ and triple order cliques going across $\mathbf{y}, \mathbf{y}_1$ and $\mathbf{y}_2$. (see Figure 4.6 for a construction). However, this energy is not submodular as the $\delta_{\mathbf{y}}(\mathbf{y}_1 \cup \mathbf{y}_2)$ term causes the submodularity to break, which causes problems for optimisation when the constraint is combined with the segmentation energy later in Section 4.3 (see Appendix Section 8.1 for proof of non-submodularity). This is not surprising as these visibility semantics are closely related to the classical NP-complete "Art Gallery Problem" – the problem of placing the minimum number of guards which together can observe the whole gallery (O'Rourke, 1987). Also it is

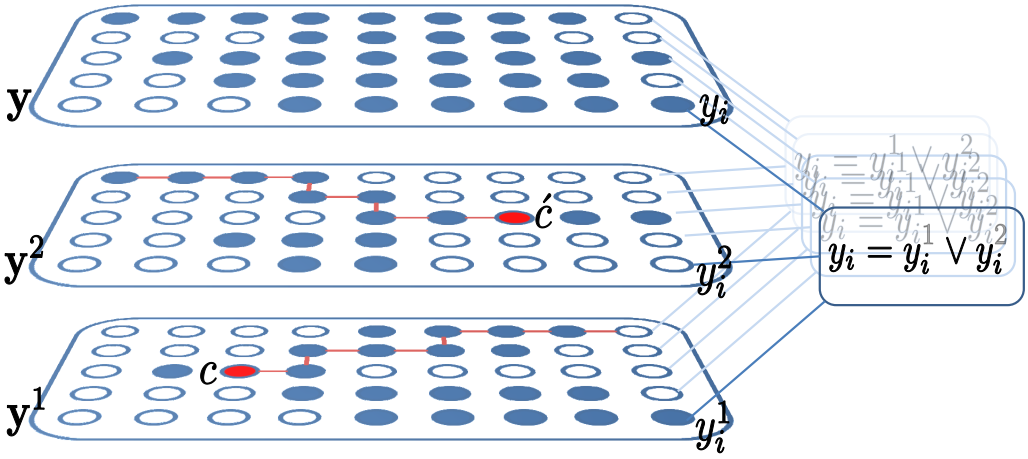


Figure 4.6: *Implementing multiple star semantics.* Graphical representation of the energy in (4.3) using pairwise and triple order cliques. Three layers of variables, one each for $\mathbf{y}$, $\mathbf{y}_1$ and $\mathbf{y}_2$ are shown. The star convexity constraints are imposed using pairwise terms on $\mathbf{y}_1$ and $\mathbf{y}_2$ as described in (4.2) (the star centres are coloured red). The constraint $\mathbf{y} = \mathbf{y}_1 \cup \mathbf{y}_2$ is imposed by a triple clique connecting each pixel i across the three layers (i.e y_i, y_i^1 and y_i^2). The triple clique takes 0 energy if $y_i = y_i^1 \vee y_i^2$ and ∞ energy otherwise. The coloured nodes are those for which the label is 1. The above figure shows a labelling that obeys the constraints in (4.3).

not obvious how to extend these semantics from a finite discrete set of star centres to an infinite continuous set of star centres – e.g a brush stroke.

The first definition of star convexity for multiple centres, above, is arguably a natural extension. The second definition, described next, is both computationally more tractable and also extends naturally to an infinite set of star centres. The definition of the line segment joining star centre c to p (denoted $\Gamma_{c,p}$) is extended to the line segment joining the set of star centres $\mathbf{c}$ to point p (denoted $\Gamma_{\mathbf{c},p}$). Observing that $\Gamma_{c,p}$ is the shortest path between point p and centre c, we define $\Gamma_{\mathbf{c},p}$ as the shortest path between point p and set $\mathbf{c}$:

$$c(p) = \arg\min_{c \in \mathbf{c}} d(c,p) \quad , \quad \Gamma_{\mathbf{c},p} = \Gamma_{c(p),p} \tag{4.4}$$

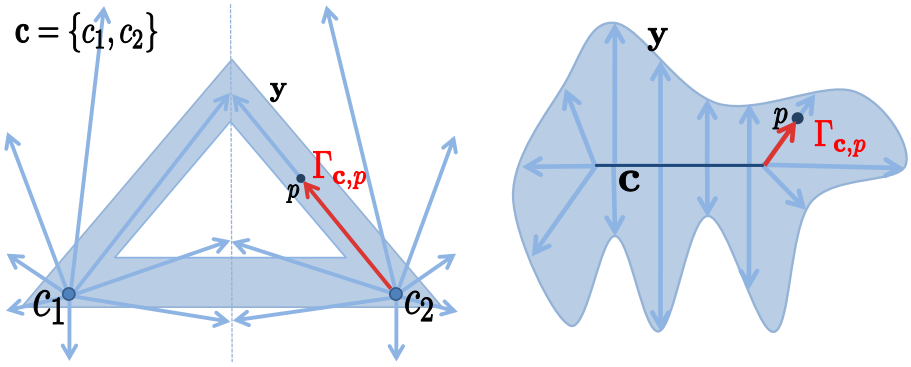


Figure 4.7: *Alternative multiple star semantics – visibility to nearest centre.* $\mathbf{c}$ denotes the set of star centres, $\mathbf{c} = \{c_1, c_2\}$ in the triangle on the left, and $\mathbf{c}$ is a line segment in the shape on the right. For every point $p \in \mathbf{y}$, star convexity is imposed on $\Gamma_{\mathbf{c},p}$ – the segment joining p to the nearest star centre.

where $d(c, p)$ is the Euclidean distance between c and p and $c(p)$ denotes the closest star centre to point p. Figure 4.7 visualises these shortest paths for the case of discrete and continuous star centres. This construction is a restriction of the previous visibility semantics – here each point $p \in \mathbf{y}$ should be visible to its nearest star centre $c(p)$. The star energy $E^*(\mathbf{y}|\mathbf{c})$ can then be written exactly as in (4.2) with the shortest paths now defined as in (4.4). Note that the star energy remains submodular which keeps things tractable as explained in section 2.1. Also these semantics extend nicely to having brush strokes as star centres. To our knowledge, such a definition of star convexity has not yet been proposed in the literature.

4.1.3 Geodesic stars

The previous section defined $\Gamma_{\mathbf{c},p}$ as the shortest path between star centres $\mathbf{c}$ and point p. In this section we generalise the notion of shortest path from Euclidean to geodesic, and also define this directly in the discrete domain. Geodesic paths can bend and adapt to image data as opposed to straight Euclidean rays, thus

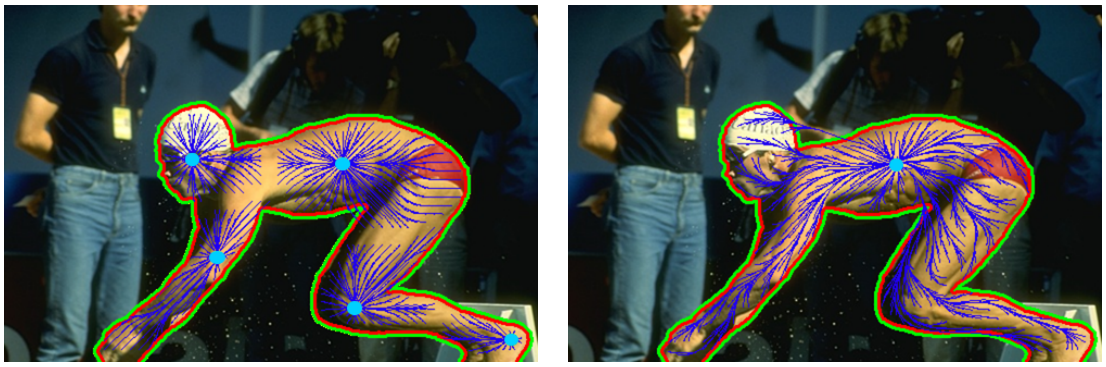
extending visibility and reducing the number of star centres required. In the case of image segmentation the gradients in the underlying image provide information to compute such paths (ref. figure 4.8). To define the geodesic distance, we first define the length of a discrete path:

$$L(\Gamma) = \sum_{i=1}^{n-1} \sqrt{(1-\gamma_g) d(\Gamma^i, \Gamma^{i+1})^2 + \gamma_g \|\nabla I(\Gamma^i))\|^2} \tag{4.5}$$

where Γ is an arbitrary parametrised discrete path with n pixels given by $\{\Gamma^1, \Gamma^2, \cdots, \Gamma^n\}$, $d(\Gamma^i, \Gamma^{i+1})$ is the Euclidean distance between successive pixels, and the quantity $\|\nabla I(\Gamma^i))\|^2$ is a finite difference approximation of the image gradient between the points (Γ^i, Γ^{i+1}). The parameter γ_g weights the Euclidean distance with the geodesic length. It is also possible to use gradients in the likelihood image to get rid of texture edges as done in Bai and Sapiro (2009). Using the above definition, one can define the geodesic distance as in Criminisi et al. (2008):

$$d_g(a,b) = \min_{\Gamma \in \mathcal{P}_{a,b}} L(\Gamma) \ , \ \Gamma_{a,b} = \arg\min_{\Gamma \in \mathcal{P}_{a,b}} L(\Gamma) \tag{4.6}$$

where $\mathcal{P}_{a,b}$ denotes the set of all discrete paths between two grid points a and b. Note that unlike Bai and Sapiro (2009), we are interested in the geodesic path $\Gamma_{a,b}$ rather than geodesic distance $d_g(a,b)$. The above definition of geodesic distance between two points also extends to distance between a set of points $\mathbf{c}$ and a point p exactly as in (4.4). Notice that everything is now defined in the discrete domain, including the shortest paths $\Gamma_{a,b}$. Thus there is no need of any rasterization step while setting up the star energy, this saves computation time. If one were to connect up every point $p \in \Omega$ to the star centre $\mathbf{c}$ using the shortest paths $\Gamma_{\mathbf{c},p}$, the structure obtained would be a collection of trees rooted at $\mathbf{c}$, and we call this structure a *Geodesic Forest* (see

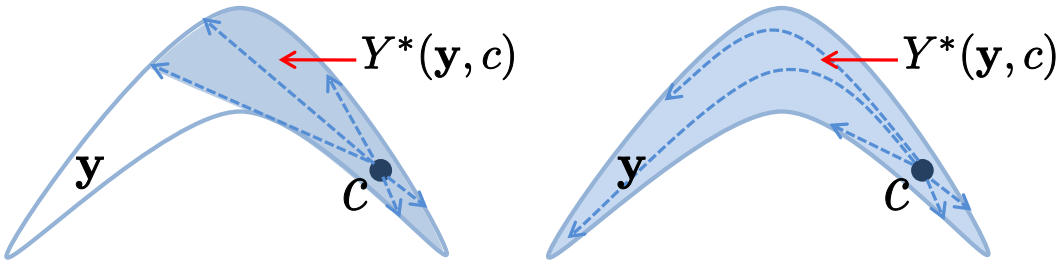


(a) Euclidean rays (b) Geodesic paths

Figure 4.8: *Geodesic stars improve visibility.* (a) Shows how the object outlined can be seen by multiple Euclidean star centres. (b) Shows how most of the object (except the head) can be seen by just one geodesic star centre. Notice how the forest bends around image gradients.

Figure 4.8 for a visualisation). Geodesics are efficiently computed using shortest path algorithms ($O(n \log n)$, with n being the number of pixels). Also note that the star energy $E^*(\mathbf{y}|\mathbf{c})$ now needs to be written as $E^*(\mathbf{y}|\mathbf{x}, \mathbf{c})$ as it now depends on the underlying image. This energy can be again expressed as in (4.2) with the shortest paths being given by (4.6).

It is important to emphasise the role of the parameter γ_g, as it controls the weighing between geodesic and Euclidean. Indeed setting $\gamma_g = 0$ results in a Euclidean shape (albeit in discrete domain), and $\gamma_g = 1$ relies purely on image gradients. We also rescale the gradients such that the average gradient is the same as the average Euclidean distance between neighbouring pixels – this is necessary to give γ_g a sensible meaning. This parameter is discussed in detail later in section 4.5. We use the shorthand GSC (Geodesic Star Convexity) to refer to this shape constraint for $\gamma_g > 0$ and the shorthand ESC (Euclidean Star Convexity) to refer to the case $\gamma_g = 0$.



(a) Euclidean $Y^*(\mathbf{y}, \mathbf{c})$ (b) Geodesic $Y^*(\mathbf{y}, \mathbf{c})$

Figure 4.9: *Illustrating the* $\mathbf{c}$*-stars of a boomerang shape* $\mathbf{y}$. The shaded region of the shape denotes the region $Y^*(\mathbf{y}, \mathbf{c})$. (a) The $\mathbf{c}$-star for the Euclidean case can only see part of the object. (b) The $\mathbf{c}$-star for the geodesic case has better visibility and can see the entire object.

4.2 Visibility experiments

In order to capture the extent of the visibility of an object from a star centre, we define $Y^*(\mathbf{y}, \mathbf{c})$, the $\mathbf{c}$-star of set $\mathbf{y}$. It is the set of all points in $\mathbf{y}$ which are visible to $\mathbf{c}$ via $\mathbf{y}$ (Valentine, 1964) (see Figures 4.9 and 4.10). Denoting the ground truth object as $\mathbf{y}_{gt}$, the extent of its visibility $Y^*(\mathbf{y}_{gt}, \mathbf{c})$ can be computed as the following optimisation:

$$\begin{aligned} &\min_{\mathbf{y}} \Delta(\mathbf{y}, \mathbf{y}_{gt}) \quad \Leftrightarrow \min_{\mathbf{y}} \Delta(\mathbf{y}, \mathbf{y}_{gt}) + E^*(\mathbf{y}|\mathbf{x}, \mathbf{c}) \\ &s.t\ \mathbf{y} \in S^*(\mathbf{c}) \qquad s.t\ \mathbf{y} \subseteq \mathbf{y}_{gt} \\ &\quad \mathbf{y} \subseteq \mathbf{y}_{gt} \end{aligned} \tag{4.7}$$

where $\Delta(\mathbf{y}, \mathbf{y}_{gt})$ measures the Hamming distance between $\mathbf{y}$ and $\mathbf{y}_{gt}$:

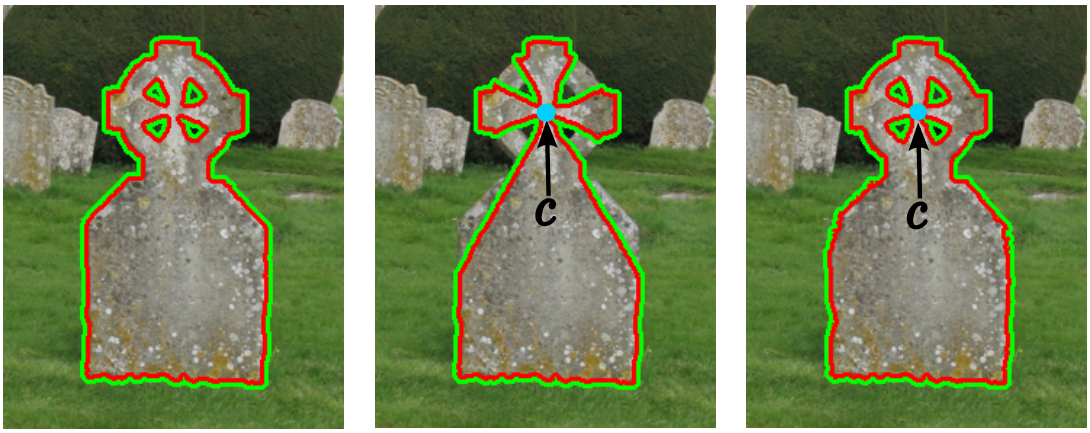
$$\Delta(\mathbf{y}, \mathbf{y}_{gt}) = \sum_{i \in \Omega} (y_i \neq y_i^{gt}) \tag{4.8}$$

Note the constraint $\mathbf{y} \subseteq \mathbf{y}_{gt}$ is equivalent to the hard constraint $\forall i : i \notin \mathbf{y}_{gt},\ y_i = 0$

and essentially means that one cannot see through the background pixels. The cost function in (4.7) can be optimised globally with graph cuts, as all the terms are submodular. The **c**-stars for a single star centre c are visualised for the case of Euclidean and geodesic star convexity in Figure 4.10. A similar cost function (but without the star constraint), was minimised for the purpose of parameter learning in Szummer et al. (2008).

In our quantitative evaluation, we find the **c**-stars for the case of 1 and 2 star centres on a dataset of images introduced later in section 4.4.2. The question here is how to choose the star centres **c**. In our experiments, the star centres were chosen manually to be favourable to the Euclidean case, so as to maximise the visibility of the star centre with Euclidean rays. It would also be possible to find the globally optimal location of star centres using Branch-and-mincut (Lempitsky et al., 2008) by optimising (4.7) wrt **c** (in addition to **y**). The occlusion rates for this *Visibility Experiment* are reported in Table 4.1. We make the following three observations from this table: (i) It is beneficial to have multiple stars as that improves the visibility of the star centres. (ii) Geodesic stars have better visibility than Euclidean stars, and thus lower occlusion rates. (iii) Geodesics improve visibility more for more complex shapes (such as those in the top row of Fig. 4.14).

One must remember that lower occlusion rates in Table 4.1 do not automatically mean the shape constraint works better in a practical system. Indeed, having no shape constraint, or just a simple connectivity constraint, would give an occlusion rate of zero in this experiment. In practice the shape constraint must be combined with other considerations, as discussed in the next section.



(a) Ground truth (b) ESC c-star (c) GSC c-star

Figure 4.10: **c**-*stars for Euclidean and geodesic star convexity.* (a) The ground truth object outlined. (b) The Euclidean **c**-star of the ground truth. (c) The corresponding geodesic **c**-star with the same star centre. The sets $Y^*(\mathbf{y}, \mathbf{c})$ are overlaid as a two-coloured boundary, with the red boundary towards the object side and green towards the background.

	Percentage Occlusion Rates			
	Full dataset		Complex Shapes	
Method	1 star	2 stars	1 star	2 stars
ESC	4.50±0.58	2.85±0.39	10.95±1.26	6.88±0.89
GSC	4.16±0.54	2.22±0.30	9.48±1.15	4.79±0.66

Table 4.1: *Visibility experiment.* The occlusion rates denote the percentage of pixels of the object not visible to the star centre. Left two columns report performance on the whole dataset and the right two columns on a subset of complex shapes in the dataset. Observing along the rows, GSC has much lower error rates than ESC, suggesting that geodesic improves the visibility of the star centres. We also see that 2 star centres have better visibility than just one. Also note the error rates are higher for complex shapes as one would have expected.

4.3 Star convexity in a segmentation system

The star convexity constraints introduced above are combined with the energy formulation of Boykov and Jolly (2001) as done by Veksler (2008). Recalling the energy function of Boykov Jolly as described in (2.9):

$$E(\mathbf{y}|\mathbf{x}) = \sum_{i \in \Omega} U(y_i|\mathbf{x}) + \lambda \sum_{(i,j) \in \mathcal{N}} V(y_i, y_j|\mathbf{x}) \tag{4.9}$$

$$\bar{\mathbf{y}} = \arg\min_{\mathbf{y} \in \mathcal{Y}} E(\mathbf{y}|\mathbf{x})$$

where $\mathbf{x}$ denotes the image, Ω is the set of all pixels, λ is a weighting on the pairwise terms and $\mathcal{N}$ represents the set of neighbouring pixel pairs. The data term $U(y_i|\mathbf{x})$ is computed using the negative log likelihood of colour models learnt from the user provided brush strokes, and the pairwise terms are the usual contrast dependent terms $V(y_i, y_j|\mathbf{x}) = \exp(-\beta\|x_i - x_j\|^2)$ (Rother et al., 2004). The colour models use GMMs with 5 components each for FG/BG. Also as the brush strokes are sparse, the GMMs are mixed with uniform colour models as suggested in Das et al. (2009) to explain colours not observed under the brush strokes. In the final system, the energy in (4.9) is minimised subject to $\mathbf{y}$ being star convex:

$$\begin{array}{ll} \min_{\mathbf{y}} & E(\mathbf{y}|\mathbf{x}) \\ s.t & \mathbf{y} \in S^*(\mathbf{c}) \end{array} \Leftrightarrow \min_{\mathbf{y}} E(\mathbf{y}|\mathbf{x}) + E^*(\mathbf{y}|\mathbf{x}, \mathbf{c}) \tag{4.10}$$

It is possible to obtain global minima of (4.10) efficiently as both $E(\mathbf{y}|\mathbf{x})$ and $E^*(\mathbf{y}|\mathbf{x}, \mathbf{c})$ are submodular. The star convexity constraint above reduces the cardinality of allowed segmentations significantly. For an $n \times n$ pixel image, the cardinality of the unconstrained output space is 2^{n*n}, whereas with a single Euclidean

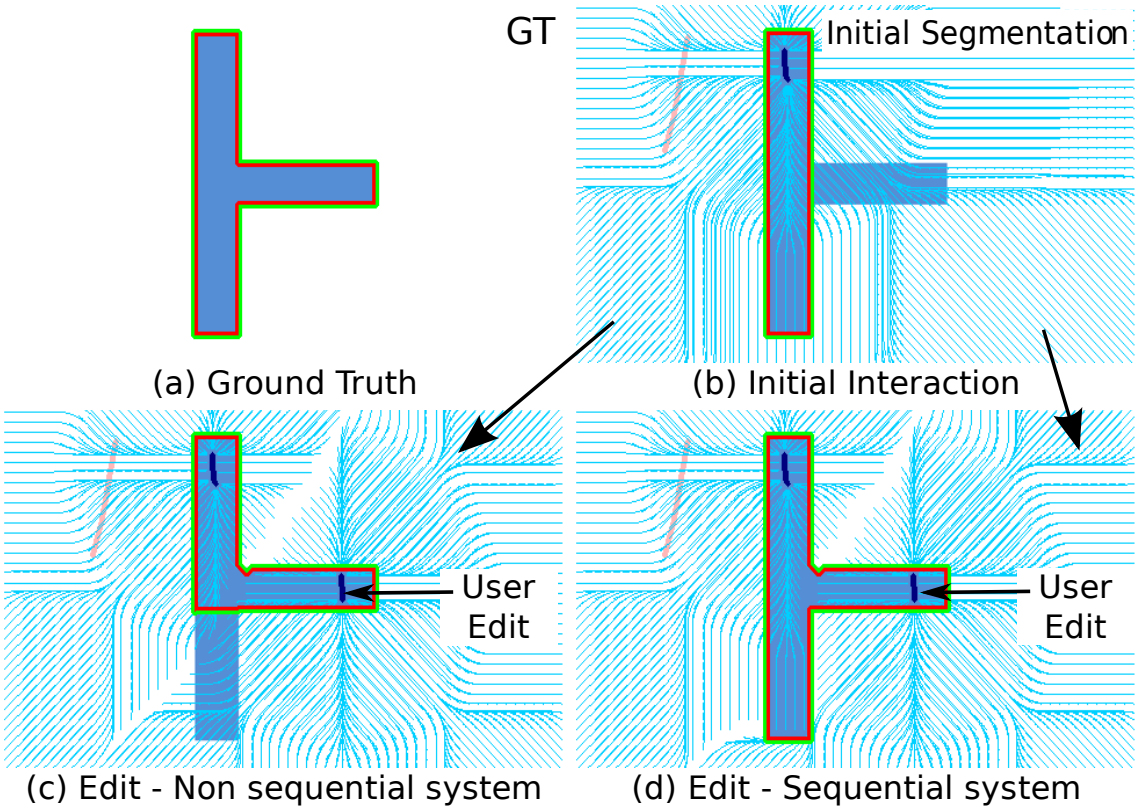


Figure 4.11: *Sequential system.* (a) Ground truth shape is outlined. (b) User places initial brush strokes, blue for FG and pink for BG. The output segmentation is shown outlined, with the underlying geodesic forest (note that $\gamma_g = 0$ here, so that the forest is actually Euclidean). (c) User makes an edit and the shape constraint changes non-intuitively. The bottom of the T-shape which was visible to the star centre on the top now is closer to the newly added star centre and not visible to it. (d) The sequential system ensures that visibility is incremental, that things which were visible before do not lose visibility. The geodesic forest is not permitted to enter the previous segmentation, so it bends/terminates around it.

star, the output space is reduced significantly to $O(n^n)$ (see Appendix Section 8.2 for the intuition). A good regulariser would allow the ground truth object to lie within its set of constraints while restricting the output space as much as possible.

Finally we consider the choice of star centres $\mathbf{c}$. In an interactive system, the FG brush strokes are a natural choice for star centres, since that avoids the need for any separate interaction to select the centres. In that way, unlike Veksler (2008), the set

of allowed shapes changes as the user adds brush strokes, and this can sometimes cause shape constraints to change un-intuitively – see Fig. 4.11. To resolve this issue, a sequential system is developed which ensures that the shape constraints change progressively as the user interacts. This is done by enforcing two properties at every edit: (i) The current segmentation should always be valid under the new star convexity constraint obtained after the user edit and (ii) the shape constraint should only change locally, around the newly placed brush stroke. (i) can be implemented by ensuring that no paths are allowed to enter the FG segmentation, in the geodesic forest computed after the edit. This can be enforced by adding an ∞ cost on edges going from BG to FG in the geodesic computation. (ii) is implemented by only incrementally computing geodesics from the newly placed brush strokes.

4.4 Quantitative evaluation

We evaluate interactive segmentation systems quantitatively by means of a "robot user" (Nickisch et al., 2009). Existing evaluation of interactive segmentation is either qualitative (Bai and Sapiro, 2009; Grady, 2006) or is restricted to a fixed set of user interactions with seed points obtained by eroding and dilating the ground truth segmentation (GrabCut Dataset; Duchenne et al., 2008). The robot user generates a flexible sequence of user interactions, according to well-defined rules, that model the way in which residual error in segmentation is progressively reduced in an interactive system.

An ideal evaluation system would measure the amount of effort required to segment an image, in a user study. Alternatively, the robot user simulates user interaction by placing brushes automatically. It starts with an initial set of brush-strokes

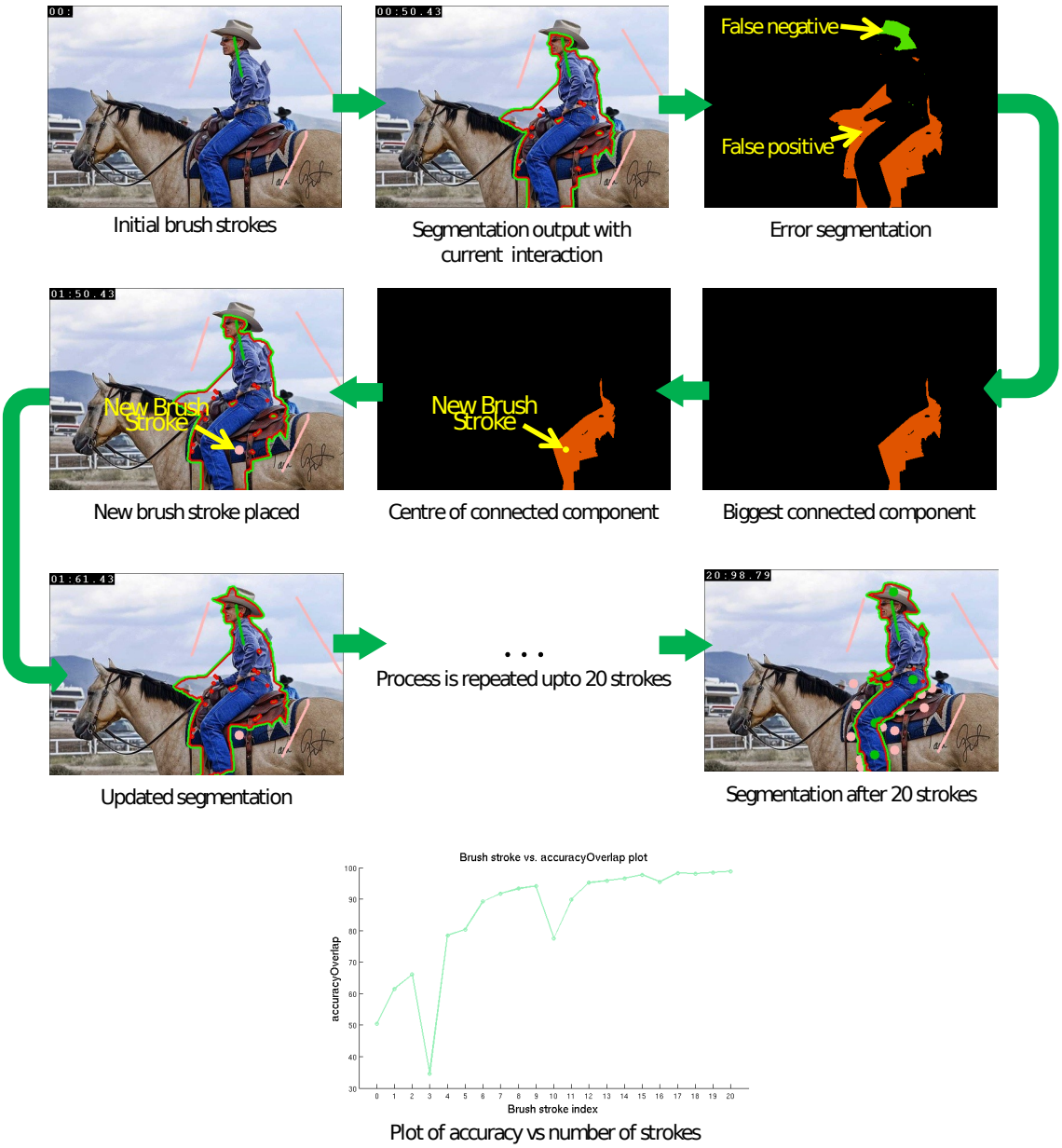


Figure 4.12: *Robot user in action.* The robot user starts with a minimal amount of initial interaction which consists of 1 fg stroke and 3 bg strokes. The initial interaction is done manually for the dataset and stored. It then computes the segmentation with the current interaction, and places a new brush stroke in the centre of the biggest connected component in error. The segmentation is then updated and this process is repeated for 20 strokes to generate a accuracy vs number of brush strokes plot.

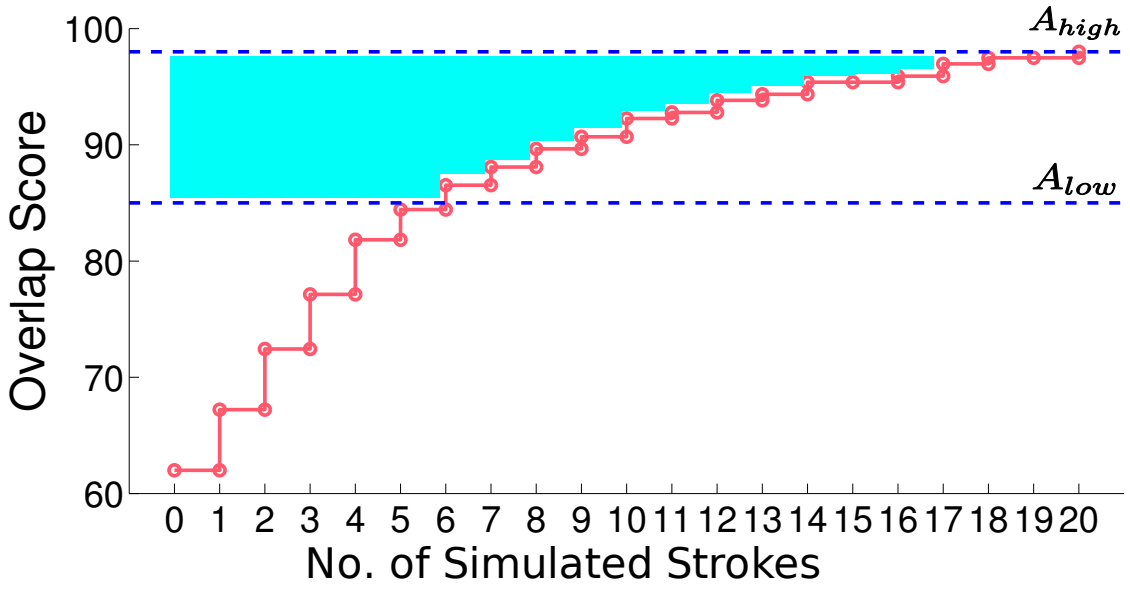


Figure 4.13: *Average overlap score vs. no. of strokes plot.* The above plot is obtained by averaging the overlap score vs. no. of strokes plot of all the images in the dataset. The area above the curve gives a measure of the average number of strokes required for user interaction. Since we are interested in the degree of interaction required to achieve high segmentation accuracy, the average is restricted to the band $[A_{\text{low}}, A_{\text{high}}]$, as illustrated (shaded in blue). When the segmentation accuracy is below A_{low}, a penalty of 1 stroke is added to the Avg. effort score. No penalty is paid if the segmentation accuracy is above A_{high}. The penalty varies linearly between 1 and 0 when the score lies between $[A_{\text{low}}, A_{\text{high}}]$.

(chosen manually with one stroke for FG and 3 strokes for BG) and computes a segmentation. It then places a circular brush stroke (diameter 17 pixels) in the largest connected component of the segmentation error area, placed at a point farthest from the boundary of the component. The process is repeated up to 20 times, generating a sequence of 20 simulated user strokes. In this way a custom sequence of brush strokes is generated for each algorithm. Figure 4.12 describes this process on an example image from the dataset.

4.4.1 Evaluation of interactive segmentation quality

Interactive system quality is evaluated as the average number of strokes (termed as Avg. Effort) required to achieve segmentation quality within a certain band. This is illustrated in Figure 4.13. The graph of overlap score v. no. of brush strokes captures how the accuracy of the segmentation varies with successive user interactions, and the average no of strokes summarises that in a single score. Here overlap score, the measure used to evaluate segmentation quality in the VOC segmentation challenge Everingham et al. (2009), is given by $100 \cdot \frac{\mathbf{y} \cap \mathbf{y}_{gt}}{\mathbf{y} \cup \mathbf{y}_{gt}}$ (with $\mathbf{y}$ denoting output segmentation and $\mathbf{y}_{gt}$ denoting ground truth). The average is computed over a certain range of scores, and we take $A_{\text{low}} = 85$, $A_{\text{high}} = 98$.

4.4.2 Dataset

The dataset (see Figure 4.14) consists of the GrabCut Dataset(49 images) augmented with images from the PASCAL VOC'09 segmentation challenge (99 images) (Everingham et al., 2009) and 3 images from the alpha-matting dataset (Rhemann et al., 2009). The VOC images come with ground truth labelling of object classes, and are cleaned up for the task of figure-ground segmentation. Images from the GrabCut dataset contain complex shapes but the foreground and background tend to have disjoint colour distributions. The VOC dataset on the other hand has simpler shapes (car, bus) but more complex appearances, in which the colour distributions of foreground and background overlap.

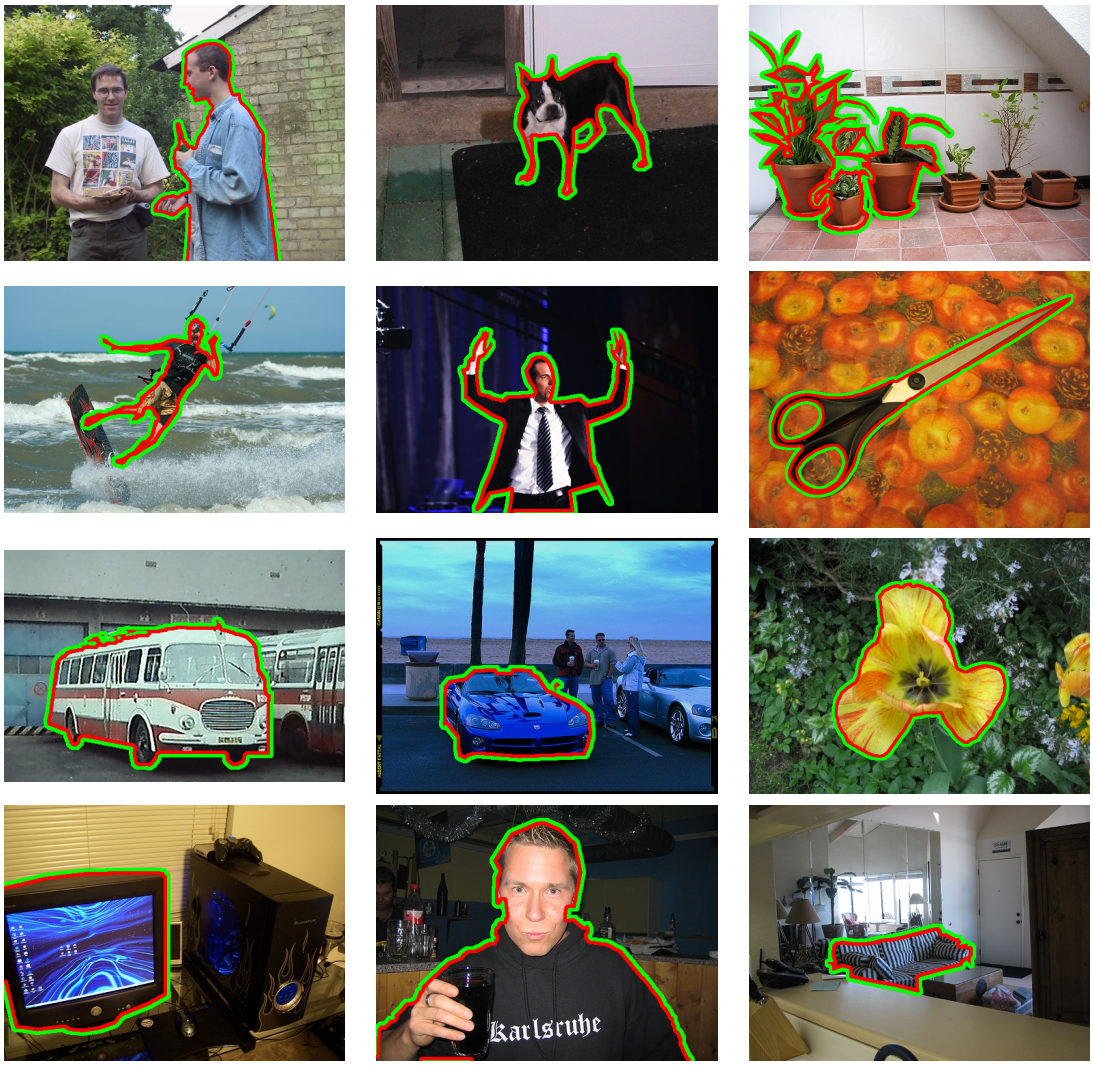


Figure 4.14: *Dataset.* 12 out of 151 images are shown. The top two rows show images with complex shapes; the bottom two rows have images with relatively simple shapes that can be modelled with just one Euclidean star centre.

4.4.3 Parameter cross validation

To set free parameters such as γ_g in (4.5) and λ in (4.9), the image set is split into validation and test sets, and line search for the best parameters is performed on the validation set. Parameters are selected to minimise the area above the accuracy overlap vs. brush strokes curve. This is repeated for 10 splits, to estimate the stability of parameters.

4.5 Results

This section analyses results obtained using the simulated user interaction on the introduced dataset. We first compare our different shape constraints with the Boykov Jolly energy function, and later compare our best system with existing methods such as Bai and Sapiro (2009); Grady (2006).

The first set of results compares: (i) *BJ* – Boykov Jolly with no shape constraint (Boykov and Jolly, 2001) (ii) *PP* – Post-Processing output of *BJ* to remove disconnected foreground islands (Liu et al., 2009). (iii) *ESC* – Boykov Jolly with multiple star Euclidean Star Convexity, implemented using geodesic forests with $\gamma_g = 0$. (iv) *GSC* – Boykov Jolly with Geodesic Star Convexity implemented with geodesic forests computed on the likelihood image. The parameter γ_g is set by validation as described in section 4.4.3. (v) *ESCseq* – Sequential system (as described in sec. 2.1) on top of *ESC* (vi) *GSCseq* – Sequential system on top of *GSC* and (vii) *GSCseq(opt)* – Same as *GSCseq*, but with γ_g being set optimally for each image.

Before moving on to the discussion of these results, the system *GSCseq(opt)* needs further explanation. In our experiments, we observed that the optimal choice of the parameter γ_g can be image dependent. This is because of the varying difficulty

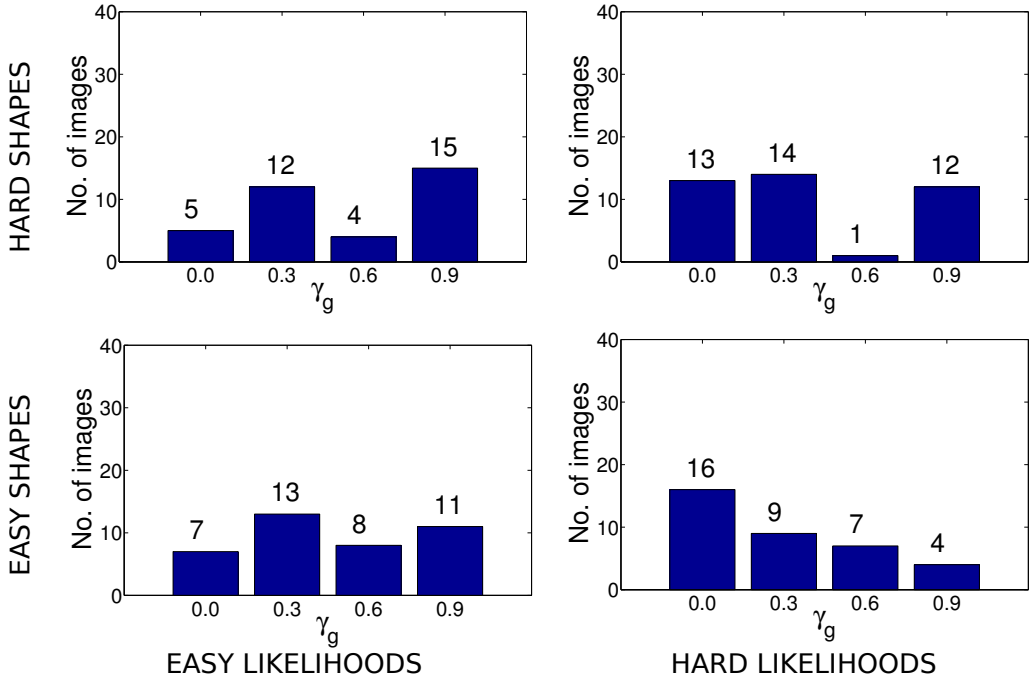


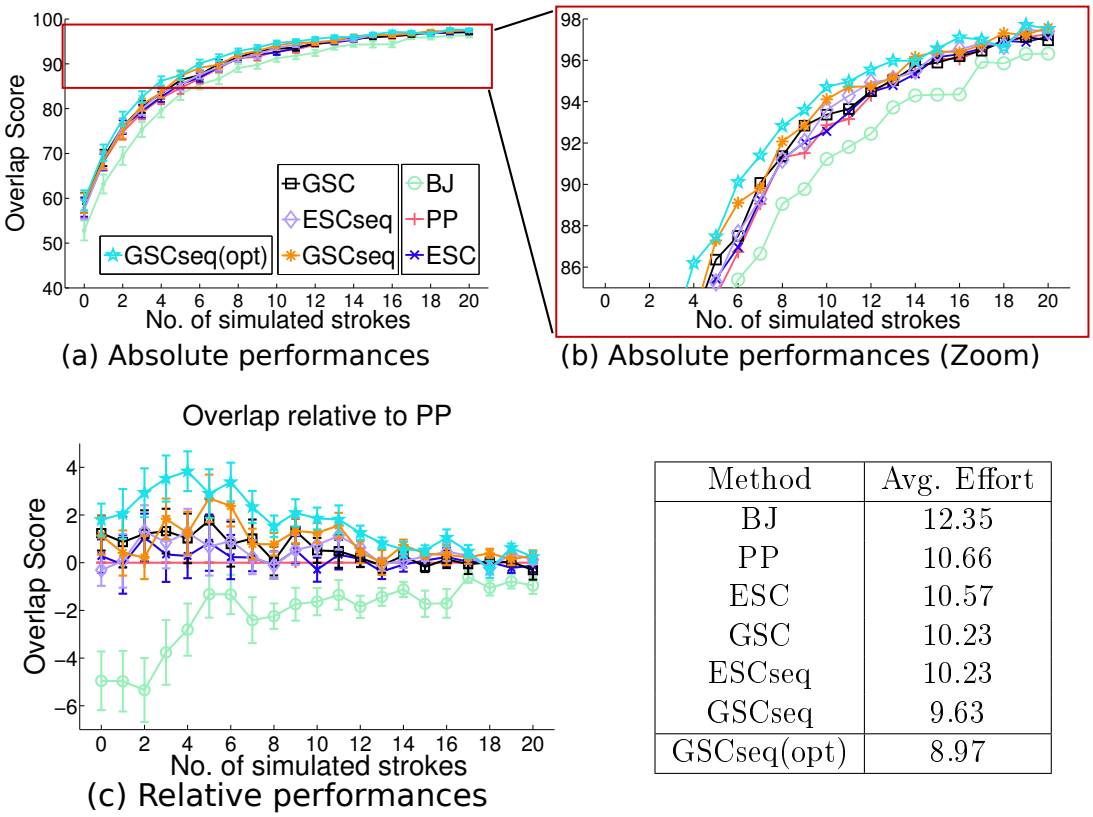
Figure 4.15: *Optimal γ_g as a function of shape and likelihood difficulty.* The above histograms visualise how often a particular γ_g is chosen as a function of likelihood and shape difficulty. Shapes are classified as easy or hard by sorting the per image occlusion rates in the visibility experiment and splitting at the median. The likelihood difficulty sorting is done using performances of a segmentation system that uses only likelihoods (i.e $\lambda = 0$ in (4.9)). With easy shapes and hard likelihoods (bottom right), $\gamma_g = 0$ is the majority choice and with hard shapes and easy likelihoods (top left), $\gamma_g = 0.9$ is the clear choice. For hard shapes+hard likelihoods and easy shapes+easy likelihoods, it's a trade off between shape and likelihood difficulty and neither is the clear winner.

of likelihoods and shapes in our dataset. We know from the visibility experiment (section 4.2) that for complex shaped objects, it is better to have a high γ_g – i.e be close to geodesic. However, that experiment is purely theoretical – in a real system the cost function also includes the likelihoods. The images in our dataset have complex appearances, and that can lead to noisy likelihoods. Such likelihoods can be a problem with high γ_g, as the geodesic constraint then connects up all the noisy likelihoods resulting in poor segmentations. In such cases, Euclidean

($\gamma_g = 0$), can behave better by ensuring that the shape does not bend arbitrarily and stays reasonable. Thus the choice of an optimal γ_g depends on the quality of the likelihoods and the complexity of the object shape. Our quantitative experiment (ref. Figure 4.15) confirms that – for complex shapes geodesic is preferred, and for complex likelihoods Euclidean (see Figure 4.15 caption for precise definitions of complex shapes and likelihoods). Thus, one could potentially let the user choose γ_g and this is what happens in *GSCseq(opt)* – the simulated user varies and chooses the γ_g that minimises effort. Choosing γ_g requires additional effort, which means *GSCseq*(*opt*) should not be compared with other systems. However, *GSCseq(opt)* still provides a useful benchmark as it lower bounds the effort required by any system that sets γ_g automatically.

Figure 4.16 visualises results for these systems. We make the following observations: (i) The best system on the whole dataset is *GSCseq* – taking 9.63 brush strokes to reach an accuracy of 98%. (ii) All of our shape constraints perform better than simple post-processing. (iii) The geodesic system (*GSC*) improves over Euclidean (*ESC*) as demonstrated by the reduction in effort from 10.57 strokes to 10.23. (iv) The sequential system *GSCseq* has a definite advantage over the non-sequential system *GSC*, reducing effort from 10.23 strokes to 9.63 strokes. On a closer examination of Fig. 4.16, one would say that all methods except for *GSCseq(opt)* and *BJ* are perhaps too close and std-errs too large to draw definite conclusions. The Std-errs are an indication that our dataset is still very small and has a large variability on a per image basis. However, the performance order matches our intuition and theory, which is encouraging.

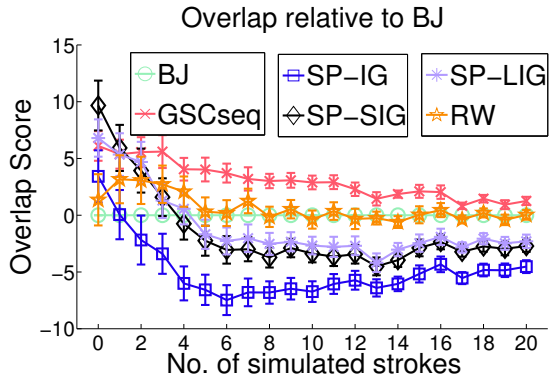
Our system is also compared with other algorithms. *GSCseq* is chosen as our system and compared with Bai and Sapiro (2009) and Random Walker (Grady,



Method	Avg. Effort
BJ	12.35
PP	10.66
ESC	10.57
GSC	10.23
ESCseq	10.23
GSCseq	9.63
GSCseq(opt)	8.97

Figure 4.16: *Overlap score vs. No. of strokes plots.* (a) Overlap scores for various methods are plotted as a function of no. of strokes with error bars for each stroke. (b) Zoom in of the region $[A_{low} = 85, A_{high} = 98]$ in the plot, error bars removed for clarity. (c) Overlap score measure relative to *PP*. The table next to the plots shows our measure – the interaction effort required to reach an accuracy of 98, in units of brush strokes. *GSCseq(opt)* is noted separately in the table as it involves additional effort in choosing γ_g.

2006). In Bai and Sapiro (2009), geodesics are computed on the likelihood image and each point assigned the label of its nearest stroke. In addition to that we also try variations of this method with geodesics computed on RGB gradients of raw and smoothed images. We use the following shorthands: (i) SP-LIG - geodesics computed on Likelihood Image Gradients. These likelihood image gradients use the same likelihood image as used for *GSC* and *GSCseq*, (ii) SP-IG - geodesics computed on raw image gradients. (iii) SP-SIG - geodesics computed on smoothed image



Method	Avg. Effort
SP-IG	17.78
SP-SIG	15.77
SP-LIG	15.14
BJ	12.35
RW	12.31
GSCseq	9.63

Figure 4.17: *Comparison with other methods.* The plot compares *GSCseq* with existing methods. The performance differences are more remarkable here compared to Fig. 4.16 because this plot compares algorithms with different cost functions, whereas Fig. 4.16 compares algorithms with the same cost function subject to different constraints. *GSCseq* takes the least avg. effort compared to other methods.

gradients and (iv) RW - Random walker. The results are plotted in Figure 4.17. This evaluation really brings out a significant difference in performance across these methods. Shortest paths methods (*SP-LIG,SP-IG,SP-SIG*) take around 15 brush strokes and fail due to lack of any boundary regularisation and sensitivity to brush location (Sinop and Grady, 2007). *RW* only performs as well as BJ (≈12 brush strokes). Though *RW* is based on only pairwise terms, there is an extension which incorporates colour models (Grady, 2005). That extension however is a relaxation of BJ into the continuous domain, and is expected to have similar shortcomings to BJ.

We also show a qualitative comparison with *DijkstraGC* (Vicente et al., 2008) in Fig. 4.18. This example clearly demonstrates the benefit of having more than just a simple connectivity constraint. *DijkstraGC* only connects up the segments with a narrow path, but GSC imposes a more sensible constraint and generates a better segmentation.

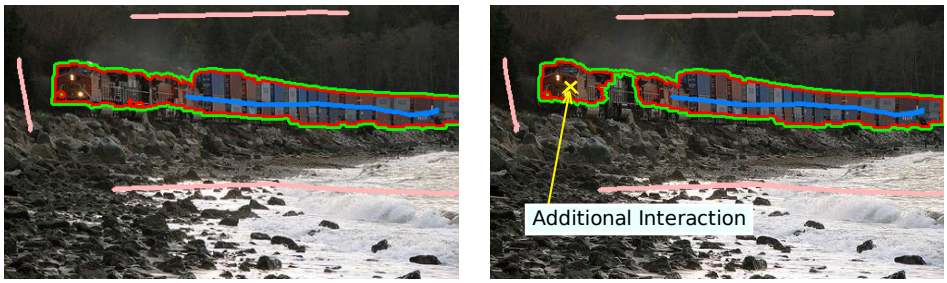


Figure 4.18: *Comparison with DijkstraGC* (Vicente et al., 2008). Both *GSC*(left) and *DijkstraGC*(right) are run with exactly the same energy function and user interaction. To connect the engine of the train to the rest, *DijkstraGC* needs further interaction as shown above. However, it only finds a very narrow path connecting the two sections of the train

4.6 Conclusions

We summarise our contributions as follows. We extended the notion of star convexity from single to multiple centres in a tractable way, and further generalised this notion from Euclidean to Geodesic. To our knowledge, such definitions of star convexity do not exist in the literature and we are the first to define them and apply them in a real system. We demonstrate using a rigorous evaluation system, how such extensions reduce interaction effort, defining the *state-of-the-art* system on a newly introduced dataset.

It would be also interesting to explore star convexity constraints to represent shapes. Indeed, we saw in this chapter that the shape constraint can model arbitrarily complex shapes and could be extended to object specific constraints as opposed to interactive constraints. These shape constraints can also be exploited in other ways suggested in Veksler (2008) – such as reducing the shrinking bias using negative edge weights, ratio optimisation and area constrained segmentation.

The datasets and code used in the quantitative evaluation (including all the segmentation methods) can be found at: `http://www.robots.ox.ac.uk/~vgg/research/iseg/`.

Chapter 5

Learning to segment humans

The previous two chapters focused on the interactive setting for segmentation, and the methods could be used to segment arbitrary object categories. Our objective in this chapter is to reduce user interaction by bringing in higher level knowledge about the object category being segmented. To that end, this chapter focuses on automatically segmenting the human object category in images. We first present an algorithm that segments humans given a bounding box around them, and then later show how it can be fully automated. This is cast as a learning problem and linear classifiers are trained to predict segmentation masks from sparsely coded local HOG descriptors. These classifiers introduce top-down knowledge to obtain a crude segmentation of the human which is then refined using bottom up information from local colour models in a SnapCut (Bai et al., 2009) like fashion. Obtaining a large dataset of images with segmented humans is crucial to capture and learn the possible variations in human poses and backgrounds. We show that a large dataset of roughly 3500 humans can be automatically acquired very cheaply using the Kinect. The method is quantitatively evaluated on images of humans in cluttered scenes, and a

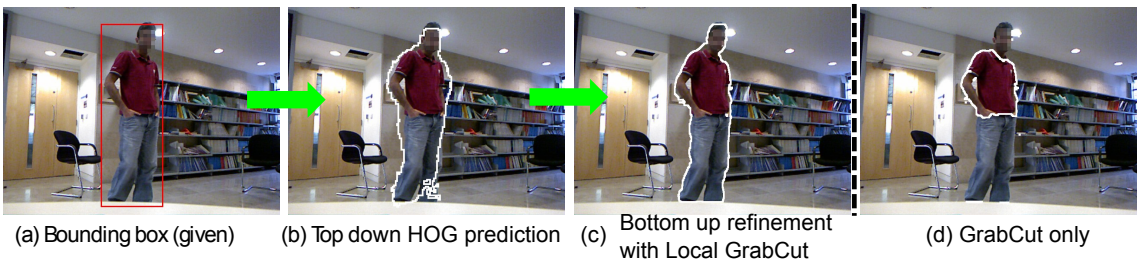


Figure 5.1: *Overview of our method.* (a) Bounding box around the human. (b) Predicted segmentation using local HOG descriptors within the bounding box. (c) Output of Local GrabCut initialised with predicted segmentation in (b). (d) Purely bottom up `GrabCut` output for comparison. `GrabCut` uses global colour distributions to distinguish between foreground and background and cannot handle overlapping colour distributions very well, hence misses the legs and the face. Our method relies on top-down information obtained from hog-descriptors which segments the body parts crudely. This initialisation can then be refined reliably using local colour models.

very high performance obtained.

Humans are the most commonly occurring subject in images, and obtaining their silhouette has many useful applications, with cut and paste image editing as in `GrabCut` (Rother et al., 2004) being one of the most popular. Other applications include 3D pose estimation, e.g. Agarwal and Triggs (2006) where human silhouettes are used to estimate pose of the body, and estimating body parameters such as height, waist etc, e.g. Balan and Black (2008) who use human segmentations as a pre-process for such body parameter estimation. All these applications need unoccluded people, but highly accurate segmentations. This is the scenario we focus on, which differs from the PASCAL segmentation challenge (Everingham et al., 2010) and the H3D dataset (Bourdev and Malik, 2009), where occlusions are numerous and current methods fall short of providing accurate enough segmentations.

Our approach is similar to the fragment based work of Borenstein and Ullman (2002, 2004); Levin and Weiss (2006), however instead of obtaining fragments by

nearest neighbour matching and overlaying their masks, we discriminatively learn segmentation masks for local regions directly from their HOG descriptors. We also tackle the harder problem of segmenting humans who have much more variation in appearance (different tight/loose clothing, pose) compared to the horses and cows used in Borenstein and Ullman (2002, 2004); Levin and Weiss (2006). As in Borenstein and Ullman (2002, 2004); Levin and Weiss (2006) we train and test our algorithm on bounding boxes around humans. Figure 5.1 shows the top-down and bottom-up stages of the algorithm. The top-down learning of segmentation is described in Section 5.1, and Section 5.2 describes how bottom-up information is utilised by running a local colour model based GrabCut (Bai et al., 2009) initialised from the predicted top-down segmentation. The learning part of the algorithm requires training data and to this end, we acquire a dataset of about 3500 images with segmented humans automatically using the Xbox Kinect. The dataset acquisition process is described in Section 5.3, and a quantitative evaluation of the segmentation algorithm is given in Section 5.4. We also test a fully automated version of the algorithm (from images to segmentation masks without requiring a bounding box), that uses trained detectors of Felzenszwalb et al. (2009) to automatically determine the human bounding box (Section 5.4.3).

5.1 Learning problem: Using top-down information

The goal of the learning problem is to predict a segmentation mask given an image and a bounding box. This segmentation mask is then fed into a Local GrabCut (Bai et al., 2009) like procedure (Section 5.2) that combines the prediction with low level cues such as pixel colours and edges to obtain the final segmentation. To train a

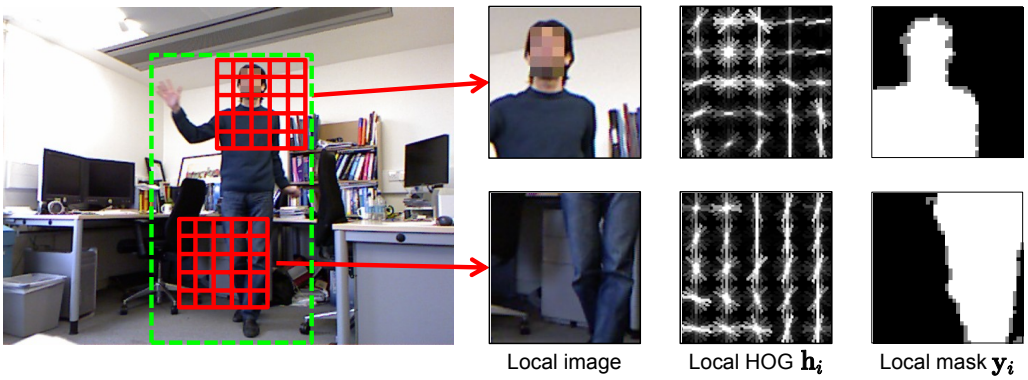


Figure 5.2: *Local sliding window features and segmentation masks.* The image inside the bounding box (shown dotted green) is resized to a fixed height of 128 pixels, thus setting the scale of the features. Local HOG features (denoted $\mathbf{h}_i$) and corresponding segmentation masks $\mathbf{y}_i$ are extracted from sliding windows sampled at an interval of one HOG cell. The gray pixels in the mask $\mathbf{y}_i$ denote pixels whose segmentation is not known and are ignored in learning. The size of the local region depicted in the figure is the one used in the experiments in Section 5.4.

predictor, a fully supervised training set consisting of images with bounding boxes and corresponding binary segmentation masks is used. The bounding box is divided into dense overlapping local regions, and independent per-pixel classifiers are trained to predict the label of every pixel within the local region (e.g. if the local region mask is of size 30×30 then 900 different classifiers are learnt). At test time, the predicted segmentations in the overlapping local regions are combined using majority voting to obtain a segmentation mask for the entire bounding box. We use linear SVMs as our classifiers and non-linearity is added by sparse coding the feature vectors onto a pre-trained dictionary (Section 5.1.1). Weak spatial information is also incorporated into the predictor by learning position dependent classifiers (Section 5.1.2).

Fig. 5.2 shows the local image regions and the corresponding features and segmentation masks extracted from a training image. The local image appearance is encoded as a HOG (Dalal and Triggs, 2005) descriptor. Every training image is sampled densely to give many such descriptors and corresponding segmentation masks.

Let i denote an index over these local descriptors and denote the i^{th} descriptor by the vector $\mathbf{h}_i$. Denote the corresponding local segmentation mask by $\mathbf{y}_i$ and a specific pixel at location l within the mask by $y_{i,l}$ ($y_{i,l} \in \{-1, +1\}$ where -1 refers to background class and $+1$ to the foreground class). A separate linear classifier $f(\mathbf{h}; \mathbf{w}_l) = \text{sign}(\mathbf{w}_l^T \mathbf{h})$ is trained for every location l (where $\mathbf{w}_l$ denotes the parameters of the classifier). These local predictors can be thought of as part detectors that predict a segmentation mask as opposed to detecting the presence or absence of parts. Section. 5.4.1 gives the implementation details of HOG, mask sizes etc.

5.1.1 Non-linear mapping via dictionary

A linear classifier cannot possibly capture the amount of variation seen in our training data as the local window slides over different parts of the body. One solution is to use non-linear classifiers that have an arbitrary discriminative power (such as an SVM with an RBF kernel) . However the large amount of training data (roughly $180,000$ HOG vectors in our training set) makes the cost of non-linear classifiers prohibitive as the kernel matrix does not fit in memory and SVM training is very slow. Instead, we keep the benefits of fast large scale linear SVM solvers (Fan et al., 2008; Shalev-Shwartz et al., 2007), but introduce non-linearity in our feature vector by sparse coding the input feature into a dictionary (Mairal et al., 2009). The idea behind sparse coding is to describe the input space (of dimensionality d) by a dictionary $D_{d \times K}$ consisting of K representative elements. Each feature vector $\mathbf{h}_i$ is then approximated by a vector of sparse coefficients $\mathbf{x}_i$ over the dictionary D s.t $\mathbf{h}_i \approx D\mathbf{x}_i$. Learning the optimal dictionary D and reconstructions $\mathbf{x}$ is formulated

as the following matrix factorisation problem Mairal et al. (2009):

$$\min_{D,\mathbf{x}_1,\cdots,\mathbf{x}_N} \sum_{i=1}^{N} \|\mathbf{h}_i - D\mathbf{x}_i\|_2^2 + \lambda\|\mathbf{x}_i\|_1 \tag{5.1}$$

where the $l1$ regularisation (i.e $\|\mathbf{x}_i\|_1$) induces sparsity on $\mathbf{x}$. The above optimisation problem is non-convex with respect to D and $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$. We use the sparse coding toolbox provided by the authors of Mairal et al. (2009) to learn the dictionary D and to map $\mathbf{h}_i$ to $\mathbf{x}_i$ at test time. Note that the mapping from $\mathbf{h}_i$ to $\mathbf{x}_i$ is non-linear due to the additional sparsity regulariser. The classifier $f(\mathbf{h}; \mathbf{w}_l)$ is now trained using $\mathbf{x}_i$ as the input vector instead of the raw HOG descriptor $\mathbf{h}$ (henceforth denoted as $f(\mathbf{x}; \mathbf{w}_l)$). The implementation details of the SVM and dictionary learning are explained in Section 5.4.1.

5.1.2 Adding spatial position

Weak spatial information is added by learning separate classifiers for different vertical locations in the bounding box. As shown in Fig. 5.3, the bounding box is divided vertically into 16 HOG cells, and the local HOG window spans 5×5 cells. This means a total of 12 different vertical positions for the local descriptor. These 12 vertical positions are divided into 4 levels, and a separate classifier is trained for each level. Learning separate classifiers for different locations makes the task of learning easier as there is less variation in the feature vectors (e.g. HOG descriptors for the upper body look quite different to those extracted from the lower body). Even though having non-linearity in our feature space can cope with this variation to a limited extent, explicitly building this into our model does show some quantitative improvement (Section 5.4.2).

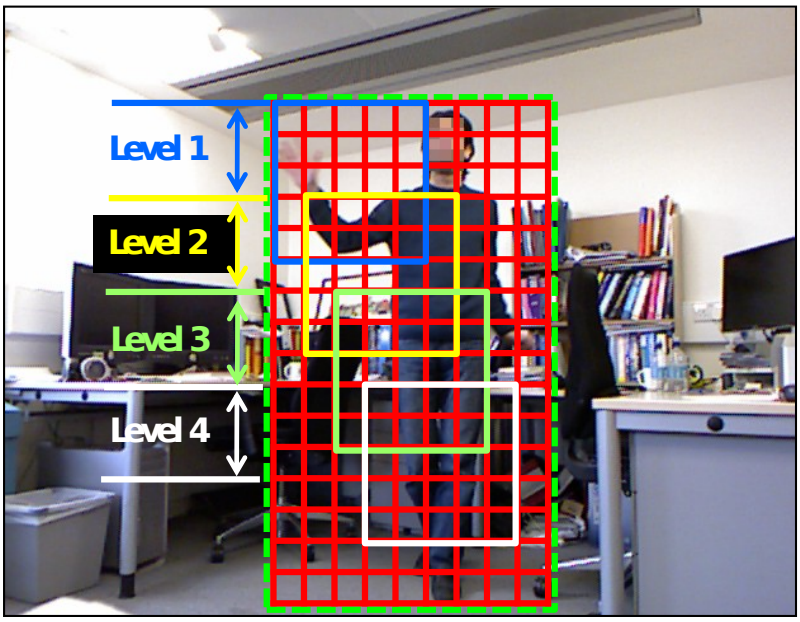


Figure 5.3: Spatial position in SVM. The bounding box is divided vertically into 16 HOG cells. As each local HOG window spans 5×5 cells, there are a total of 12 possible vertical positions for this local window. These vertical positions are divided into 4 levels as shown above, and a separate classifier is learnt for each level.

5.2 Local GrabCut: using bottom-up information

The top-down segmentation obtained from the local HOG based classifiers gives a good starting point for running GrabCut based segmentation (see Fig. 5.4 (a)). This initialisation can be exploited to learn location specific colour models (as opposed to learning a single colour model for foreground and background as in `GrabCut`). Such local colour models were shown to be very useful in video segmentation (Bai et al., 2009) where a good initial segmentation was obtained from the previous frame and also in alpha matting (Levin et al., 2008). Having local colour models greatly improves their discriminative power and this effect is shown quantitatively later in Section 5.4.

The grabcut energy formulation is the same as in Rother et al. (2004) except

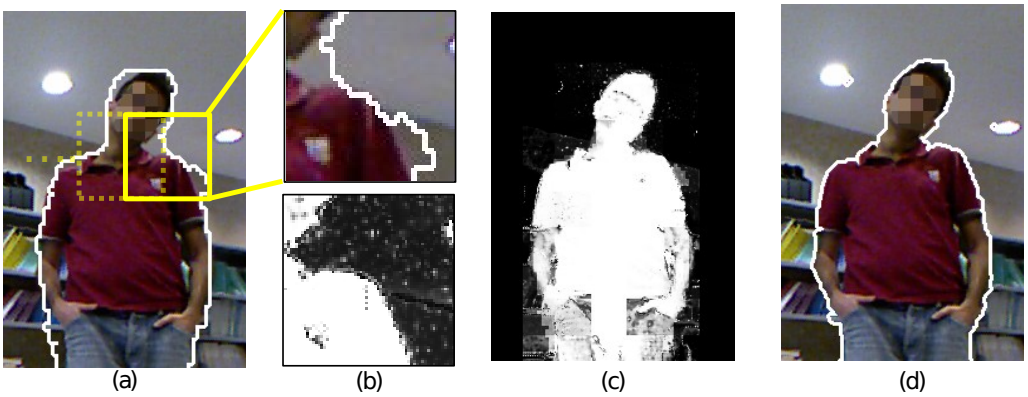


Figure 5.4: *Local GrabCut initialised from HOG prediction.* (a) The top-down segmentation predicted using the SVM. LocalGC windows are slid over the image at an interval of half their width. Only those windows that intersect the segmentation boundary are used to estimate local unaries. (b) Local window colour models are initialised from the segmentation within it, and the corresponding unary image is visualised for the specific window. (c) Unary visualised for entire image by averaging overlapping windows. (d) Segmentation obtained after applying graphcut on these local unaries. (Note: Only the upper body is shown in the image for clarity, the algorithm is actually run on the entire bounding box.)

that the unary terms computed using local colour models as opposed to global colour models. Recycling the variables used before to denote new terms, let $\mathbf{x}_i$ denote the RGB value of pixel i, $y_i \in \{-1, 1\}$ denote the output segmentation label of pixel i and y'_i denote the predicted label of pixel i (obtained from the top-down segmentation). The energy function for segmentation is given by:

$$E(\mathbf{y}|\mathbf{x}, \mathbf{y}') = \sum_{i=1}^{N} U(y_i, \mathbf{x}_i) + \gamma' \sum_{i=1}^{N} (y_i \neq y'_i) + \gamma \sum_{i,j \in \mathcal{N}} V(y_i, y_j, \mathbf{x}_i, \mathbf{x}_j) \qquad (5.2)$$

where the term $\sum_{i=1}^{N} (y_i \neq y'_i)$ penalises deviations from the top-down segmentation. The unary terms $U(y_i, \mathbf{x}_i)$ are given by negative log likelihoods of local colour model distributions (ref. Fig. 5.4 for a visualisation). Note that the local windows used for colour models are different from the local windows used for HOG prediction

described in Section 5.1. To prevent confusion, we refer to these new local windows as localGC windows. As every pixel is covered by more than one localGC window, the negative log likelihoods from each window are averaged to obtain the unary term for each pixel. The pairwise terms are the usual contrast dependant terms from Rother et al. (2004): $V(y_i, y_j, \mathbf{x}_i, \mathbf{x}_j) = (y_i \neq y_j) \exp\left(-\beta \|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$. As in SnapCut (Bai et al., 2009), the process of estimating the colour models and segmentation is iterated a few times (segmentations usually converge within 4 iterations). The energy in (5.2) is optimised globally using the graph-cut code of Boykov and Kolmogorov (2004). The two most important parameters for this system are γ and the localGC window size. These are optimised on the training data set as described in Section 5.4.1.

5.3 Dataset

We use the Xbox Kinect to obtain a dataset of segmented humans. The Kinect device consists of an infra-red projector camera pair that computes a depth image, and a calibrated RGB camera. Publically available drivers and libraries provided at OpenNI are used to automatically obtain segmentations of moving objects from the kinect. The OpenNI libraries obtain pixel accurate segmentations from the depth images using background subtraction like algorithms and project them onto the RGB camera. Figure 5.5 shows the kind of data that is obtained using the Kinect after post-processing to give human segmentations. Both the depth image and the RGB image are captured at a resolution of 640×480 pixels. The raw output of the Kinect RGB camera is a Bayer pattern image (Bayer Filter), and needs to be subsequently debayered to get a RGB value at every pixel. Debayering code from the ROS Kinect package (ROS OpenNI) is used here.

(a) RGB camera image (b) Labels obtained from kinect (c) Cleaned up Ground truth

Figure 5.5: *Training data from Kinect.* (a) The 640×480 RGB image. (b) Automatically segmented human projected from the depth camera to the RGB camera. The red labels denote the segmented object. The white labels denote regions that are non IR-reflective (such as hair, monitors etc), shadow regions of the stereo camera or regions lost due to registration of the cameras. (c) Cleaned up ground truth is obtained by marking the regions lost due to registration as unknown (gray). Also, any white labels in the vicinity of red labels are marked as unknown to prevent labelling the hair incorrectly, as we observed that hair was often non-IR reflective for certain people. We do not use the body part layout during training, thus its possible to capture data and train our algorithm for other classes such as cats, dogs etc.

To collect data, we record a sequence of approximately 100 frames for each human subject at various indoor locations (the locations are restricted to be indoors as the Kinect does not work outdoors due to IR interference from sunlight). As the RGB and depth images are not synchronised in time, we only record those frames whose time-stamps differ by less than 11ms to prevent motion from introducing a significant error in the segmentation. The first 10 acquired frames are ignored to allow the OpenNI segmentation algorithm to burn in (the OpenNI segmentation uses a background subtraction like algorithm that takes a few frames to start segmenting accurately).

Most images in the dataset consist of unoccluded humans (unlike Everingham et al. (2010); Bourdev and Malik (2009)) and this fits well with the use cases mentioned in the introduction to the chapter. A limitation of this dataset is that it

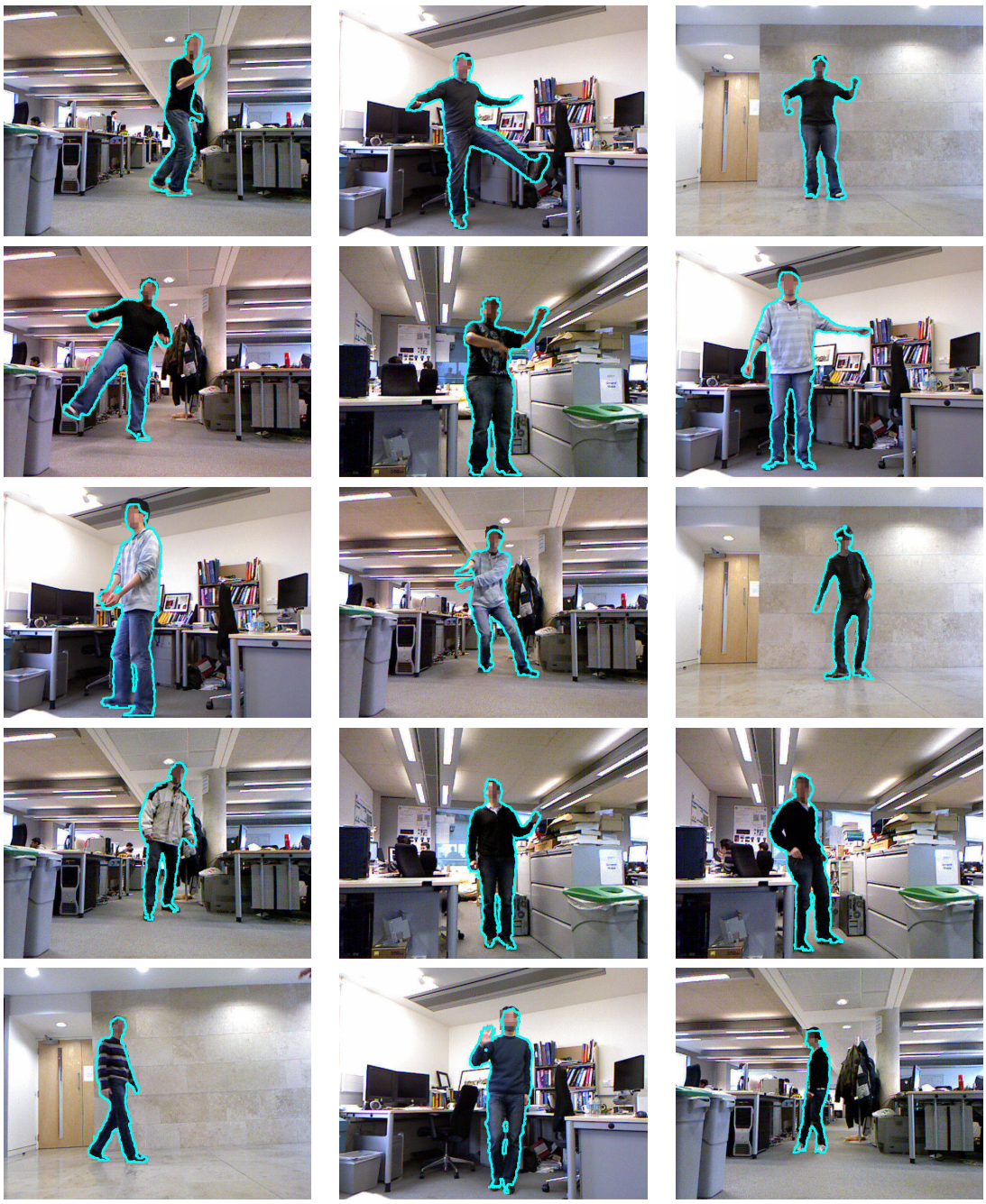


Figure 5.6: *Example training images from our dataset.* The ground truth segmentation is outlined in light blue. Note the variation in pose and complex background appearances.

consists mostly of standing humans (the automatic OpenNI segmentation algorithm makes errors when humans interact with objects such as chairs when they sit). Compared to the Pascal VOC (Everingham et al., 2010) and the H3D dataset (Bourdev and Malik, 2009) which require expensive manual annotation, Kinect based dataset acquisition is very cheap and is also much faster. We obtain 3386 segmented images using the Kinect, of which 1930 are used in training and 1456 used in test (compared to 925 segmented humans in Everingham et al. (2010) and 1005 in Bourdev and Malik (2009)). To prevent overlap between train and test data, both the locations and people are kept disjoint between the two sets. 10 human subjects across 4 different locations are used for training, and 6 subjects across 4 different locations are used for testing. Some images and ground truth segmentations from the dataset are shown in Fig. 5.6.

5.4 Results

5.4.1 Implementation details and parameter settings

For computing HOG, we use the implementation of Felzenszwalb et al. (2009). The image inside the bounding box is resized to a height of 128 pixels and divided vertically into 16 HOG cells of size 8×8 pixels (the width is set to preserve the aspect ratio of the bounding box). Each HOG cell gives a 13 dimensional descriptor (9 bins for orientation and 4 normalisation constants). A local HOG window spans 5×5 HOG cells (ref. Fig. 5.2), giving rise to a $13 \times 5 \times 5 = 325$ dimensional descriptor. This window is slid at an interval of 1 HOG cell, providing for an overlap of 4×5 HOG cells ($= 80\%$ of window size) between two neighbouring windows. These 325 dimensional descriptors obtained from each local HOG window are mapped to a

dictionary of size 2500. $\lambda = 0.15$ is used for dictionary learning in (5.1). The local segmentation masks are resized to a resolution of 40×40, meaning that 1600 classifiers (one per pixel) are learnt in total. LibLinear (Fan et al., 2008) is used to train these linear classifiers. The training data is doubled by flipping every training image around the vertical axis. Roughly $180,000$ pairs of local HOG descriptors and segmentation masks are extracted for training. LibLinear scales well to such large amounts of training data and is efficient especially when the features are sparse (as in our case). The cost function used in training LibLinear is:

$$\min_{\mathbf{w}_l} \frac{1}{2}\|\mathbf{w}_l\|^2 + C\sum_{i=1}^{N} \max(0, 1 - y_{i,l}\mathbf{w}_l^T\mathbf{x}_i)^2 \tag{5.3}$$

Note that we choose the L2-hinge-loss (i.e the term $\max(0, 1 - y_{i,l}\mathbf{w}_l^T\mathbf{x}_i)^2$) in (5.3) as opposed to the L1-hinge-loss (i.e $\max(0, 1 - y_{i,l}\mathbf{w}_l^T\mathbf{x}_i)$) because LibLinear has an efficient implementation for solving the L2-hinge-loss in the primal. The L2-hinge-loss cost function performed empirically similar to the L1-hinge-loss cost function while being about three times faster to train for a range of C values. Summarising the magnitude of the training task, 1600 independent SVM classifiers are trained, with each classifier learning 2500 parameters (for a total of $1600 \times 2500 = 4$ million parameters). Each SVM is trained using $180,000$ training points, and it takes approximately 1 hour in total to train all these 1600 SVM classifiers (with C set to the optimal value described below). The training procedure is not parallelised in our implementation, the run-time reported is for serial code on a 2.5GHz CPU. In the case where spatial information is also used (Section 5.1.2), 4×1600 SVM classifiers are learnt, and each classifier is trained using $180,000/4 = 45,000$ training points. The training time is similar to the time taken for training classifiers without the

spatial information.

To tune parameters such as dictionary size, local HOG window size, the regularisation parameter C in (5.3) and γ' in (5.2) the training set is divided into three train and validation folds. Parameters are chosen to maximise the mean overlap score on the validation sets. The dictionary size is varied over the set $\{500, 1000, 1500, 2000, 2500\}$, with 2500 performing the best (we didn't go further as computational expense increases with increasing dictionary size). C was varied in the set $\{0.01, 0.05, 0.1, 0.35, 0.7, 1, 10\}$ with 0.1 being the best. γ' is varied in $\{0, 0.5, 1, 10\}$ with $\gamma' = 1$ being the best. The local HOG window size was varied in the set $\{4 \times 4, 5 \times 5, 6 \times 6\}$ (in units of HOG cells) with 5×5 being the best. For parameters such as γ in (5.2) and localGC window size, a line search is done on the entire training dataset to optimise the mean overlap score. γ is varied in the set $\{10, 20, 25, 30, 50, 100\}$ with $\gamma = 20$ being the best. The localGC window size is varied in the set $\{61 \times 61, 81 \times 81, 101 \times 101, 121 \times 121\}$ with 61×61 pixels being the best. For learning local colour models, Gaussian mixture models with 3 components are used. The top-down prediction takes roughly 2 seconds per image and Local GrabCut takes about 6 seconds per image (these are using unoptimised implementations in matlab).

5.4.2 Quantitative evaluation

The performance measure used for evaluation is the overlap score between the predicted segmentation and the ground truth segmentation (this is also the measure used in Everingham et al. (2010)). The overlap score between two binary segmentations $\mathbf{y}_1$ and $\mathbf{y}_2$ is given by $\frac{\mathbf{y}_1 \cap \mathbf{y}_2}{\mathbf{y}_1 \cup \mathbf{y}_2}$. Mean overlap score over all test images is reported along with the standard-error of the mean (the standard-error is the

Methods	Train(%)	Test(%)
Box+GC	76.5 ± 0.4	72.5 ± 0.6
Box+LocalGC	78.0 ± 0.4	74.4 ± 0.6
LinSVM	73.9 ± 0.2	76.1 ± 0.3
SpSVM	86.1 ± 0.1	80.6 ± 0.2
SpSVM+Pos	89.8 ± 0.1	82.6 ± 0.2
SpSVM+Pos+GC	87.3 ± 0.2	86.5 ± 0.3
SpSVM+Pos+LocalGC	**91.8 ± 0.1**	**88.5 ± 0.2**

Table 5.2: *Average overlap scores on the train and test sets along with std-error.* The abbreviations are detailed in Section 5.4.2. The top two rows are bottom-up methods based on colour models only, the middle rows are purely top-down methods based on prediction from local HOG's. Best performance is achieved by combining both (last two rows).

standard-deviation divided by the square root of the test set size).

We compare variations of different learning methods presented in Section 5.1 (linear SVM, sparse coded HOG descriptors and position dependant SVM) and the effect of adding bottom up information on the output of these methods. The following shorthands are used to denote the different algorithms: (i) *Box+GC:* Denotes `GrabCut` initialised from the bounding box, with the parameter γ optimised on the training set. (ii) *Box+LocalGC:* Denotes the Local GrabCut implementation described in Section 5.2, initialised from the bounding box. (iii) *LinSVM:* Denotes the output of a Linear SVM learnt on the HOG descriptors. (iv) *SpSVM:* Denotes the output of a linear SVM learnt on sparsely coded HOG descriptors. (v) *SpSVM+Pos:* Extension of *SpSVM,* with a separate SVM for each vertical position (Section 5.1.2) (vi) *SpSVM+Pos+GC:* Uses the top-down segmentation obtained using *SpSVM+Pos* to initialise colour models for `GrabCut`. The top-down segmentation is also used as a weak unary term. (vii) *SpSVM+Pos+LocalGC:* Analogous to *SpSVM+Pos+GC,* but using the Local GrabCut implementation described in Section 5.2.

Table 5.2 compares performances of the methods introduced above. *Box+GC* does not perform well on our dataset as most images in our dataset have background clutter and complex appearances, causing global colour models to fail. *Box+LocalGC* does not do well because bounding boxes do not give good enough initialisations for local colour models. Purely top-down methods (i.e. the middle rows of Table 5.2) perform better than purely bottom up methods (the top two rows). Amongst the purely top-down methods, a noticeable performance gain is seen in going from *LinSVM* = 76.1% to *SpSVM* = 80.6%, suggesting that sparse coding the HOG descriptors is indeed beneficial in improving the power of the linear classifier. Adding weak spatial information also helps in increasing performance by 2% (i.e. *SpSVM+Pos* = 82.6% compared to *SpSVM* = 80.6%). There is a clear benefit in combining top-down and bottom up information as demonstrated by the best performing methods in the bottom two rows of Table 5.2 (e.g., performance of *SpSVM+Pos* goes up from 82.6% to 88.5% for *SpSVM+Pos+LocalGC*). Also, local colour models in *SpSVM+Pos+LocalGC* (88.5%) perform better than global colour models in *SpSVM+Pos+GC* (86.5%). A qualitative comparison between *SpSVM+Pos, SpSVM+Pos+LocalGC* and *Box+GC* is shown in Figure 5.7.

5.4.3 Full automation

To this point, segmentations have been obtained given a bounding box around the person. In practice this bounding box could be provided manually, and in the quantitative comparison above, the bounding boxes were computed using the ground truth segmentations. However, it is possible to use trained person detectors to obtain these automatically. This then enables humans to be segmented directly given the image, without requiring a human (or ground truth) to specify a bounding box (see

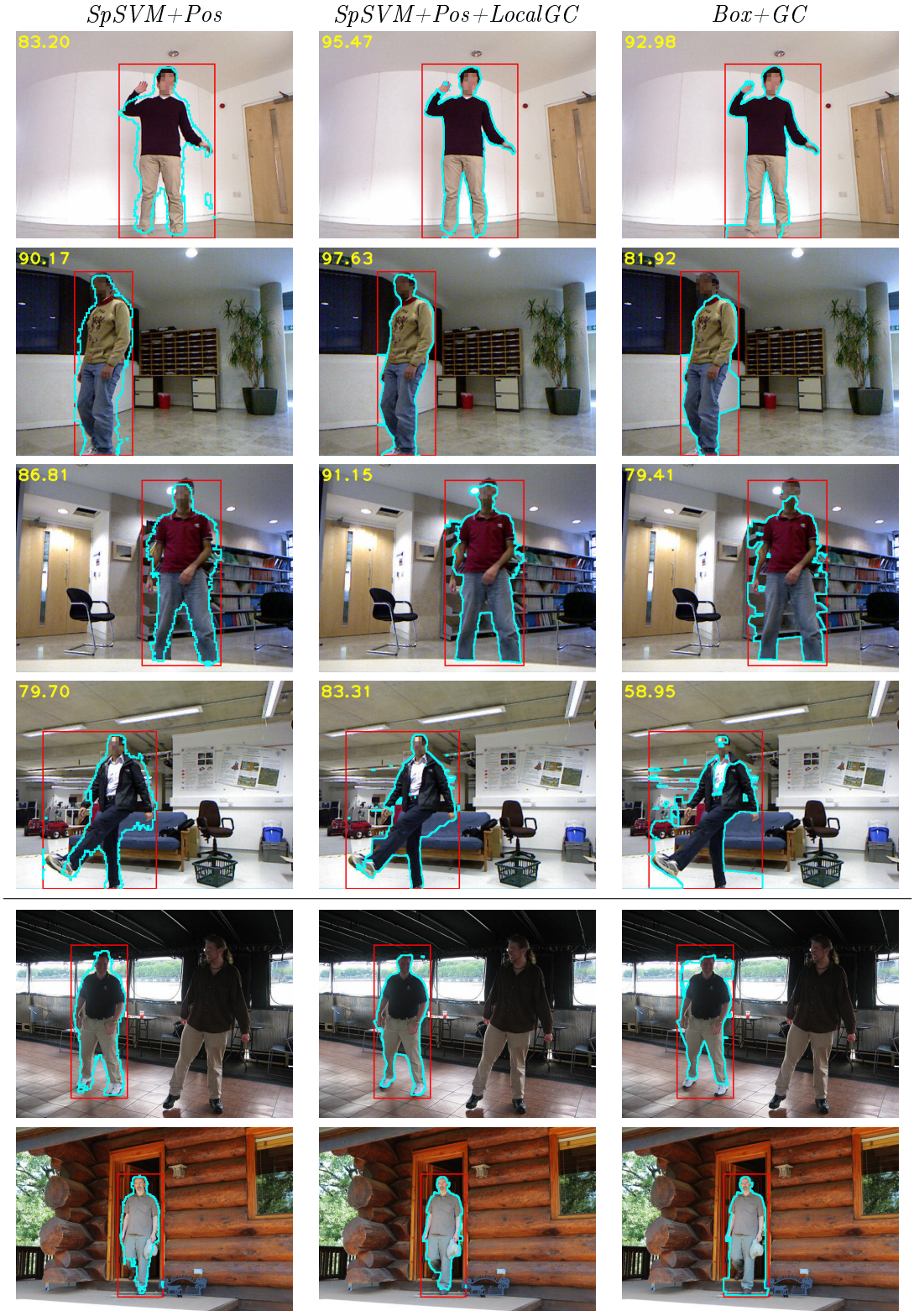


Figure 5.7: *Qualitative results.* Output segmentations outlined for various methods (with the overlap score printed on the top-left). Top four rows are from our test set and bottom two rows are images from outside our dataset (PASCAL VOC and H3D respectively). The errors in the segmentations can be corrected with further user interaction such as brush strokes.

(a) Original detection (b) *SpSVM+Pos* (c) *SpSVM+Pos+LocalGC*

Figure 5.8: *Fully automatic segmentation.* (a) Bounding box detection output. (b), (c) Segmentation output of our algorithms. The bounding box in (b) and (c) is obtained by transforming the detection in (a) as explained in Section 5.4.3.

Figure 5.8).

We use the person detector of Felzenszwalb et al. (2009) that is trained on the PASCAL VOC 2009 dataset to obtain bounding boxes around humans. To convert the detection to a bounding box more accurately, the median offset and scaling of the detections to the ground truth bounding box is estimated on the training set. At test time, the detection is transformed to a bounding box using these estimated parameters. Using full automation, the average overlap scores achieved is: 78.6±0.6. The lowering of performance is mainly due to missed detections (which get scored as 0) and partial detections (such as detecting the upper body instead of the whole human).

We compare this performance with the semantic segmentation of Ladicky et al. (2009) whose method is quite competitive on the PASCAL VOC segmentation challenge (Everingham et al., 2010). Their method gives an average overlap score of 42.3 ± 0.5 on this dataset (we train and test their algorithm using publically available code provided by the authors of Ladicky et al. (2009)). Clearly, our algorithm has a significantly better performance. Note, however, that the method of Ladicky et al. (2009) does not make use of the person detection bounding box (though con-

versely this means that it is not hurt when the detector fails, as we are), and is also not able to benefit from the fact that there is only one person per image by choosing the most confident detection per image. For a more fair comparison, we also try clipping the segmentation output of Ladicky et al. (2009) to the bounding box detections used for our method – this improves the performance of Ladicky et al. (2009) to 66.2 ± 0.5. This is still much lower than the performance obtained by our fully automated method as reported above (i.e 78.6 ± 0.6).

5.5 Conclusion

We have introduced a method for easily and automatically obtaining training data for segmentation learning algorithms using the Kinect, together with a novel algorithm for learning from such data that scales to a large training dataset. We also combine this learning based method with bottom-up information to further improve segmentation performance. Although we have concentrated here on humans, since these are arguably the most important object category, the framework is general and can be used for any other object category for which the depth background subtraction applies, e.g. objects which move such as cats, dogs, cows etc. The dataset used in this chapter is available at `http://www.robots.ox.ac.uk/~vgg/data/humanSeg/`.

Chapter 6

Semantic segmentation

In this chapter, we move towards fully automatic methods for segmenting object categories in images. The learning algorithm described in the previous chapter makes a step towards that, but it was designed assuming a bounding box detection around the object category, and focused only on segmenting humans in images. Segmentation methods that rely upon bounding box detectors are not suitable for all kinds of object categories (e.g. 'stuff' like things such as sky, grass, objects that are not very well enclosed by boxes such as various human poses, giraffes etc.). In this chapter, we build classifiers from the ground up that do not rely on the output of object detectors and experiment with different object categories such as bikes, cars etc. Several methods for semantic segmentation were covered in the literature review (e.g. Pantofaru et al. (2008); Shotton et al. (2006, 2008); Ladicky et al. (2009); Li et al. (2010)). We build upon the work of Pantofaru et al. (2008) who propose a simple superpixel based semantic segmenter that combines classifier outputs across multiple super-pixelisations. In the first part of this chapter (Sections 6.1–6.4), we extend their method and look at various ways of learning from multiple segmen-

tations, and discuss their computational efficiency vs. performance gain trade-offs. The methods are evaluated quantitatively on the Graz (Opelt et al., 2006) and the PASCAL VOC datasets (Everingham et al., 2010). In the second part of the chapter (Sections 6.6–6.7), we explore learning boundary properties of segmentations in a structured output setting, and show that learning pairwise terms for segmentation helps improve performance over independent superpixel classification. A global histogram of visual words along the boundary of the segmentation is used to describe the boundary of a segmentation. This boundary description is implemented using pairwise terms in a CRF formulation for object category segmentation. The parameters for the boundary as well as superpixel appearance are learnt jointly in a structured output learning setting. We demonstrate that the boundary descriptions based on a histogram of visual words perform better than the commonly used Ising model based pairwise terms.

6.1 Superpixel classification

Bottom up super-pixelisations provide good regions to aggregate feature responses that can be used to infer class labels for the superpixel. These aggregated responses can then be used as feature vectors to train powerful classifiers. Several bottom up super-pixelisation methods were covered in detail in the Literature survey in Section 2.3. In this chapter, superpixels obtained from QuickShift (Vedaldi and Soatto, 2008), Veksler (Veksler et al., 2010) and gPb (Arbeláez et al., 2011) are used (see Figure 2.15 in the Literature Survey for a visualisation of these super-pixelisations on the same image). First, we introduce some notation to describe a simple superpixel classifier assuming a single super-pixelisation per image, and later

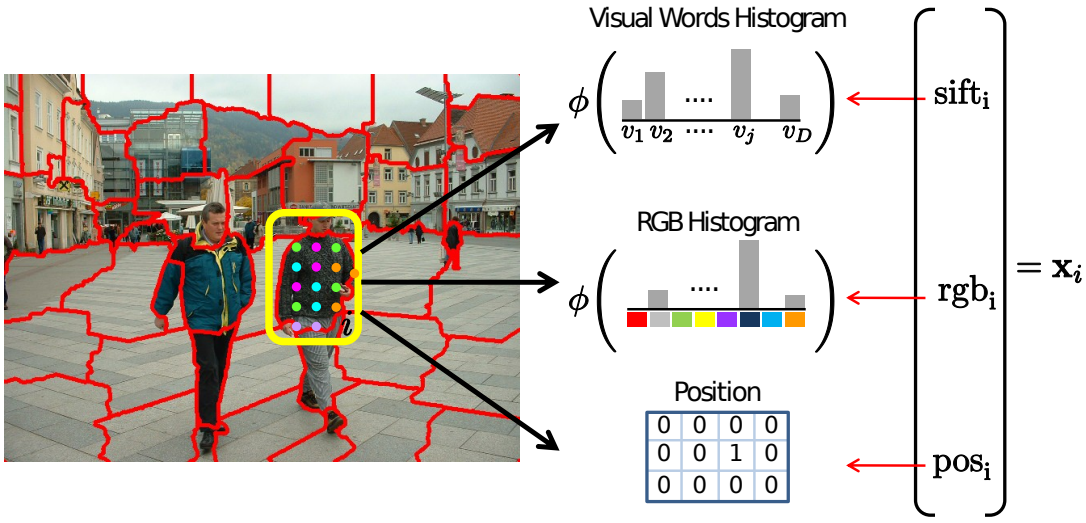


Figure 6.1: *Superpixel features.* The figure shows a superpixel indexed by i highlighted in a yellow box. For each superpixel, three different features are extracted, viz. (i) Visual words histogram: This is a histogram of quantised SIFT descriptors (Lowe, 2004) computed densely within the superpixel. (ii) RGB histogram: Histogram of discretised RGB colours in the superpixel and (iii) Position: A sparse vector encoding the position of the superpixel centroid relative to the image. A χ^2 feature map is applied to the histograms before concatenating them to give the feature vector $\mathbf{x}_i$ for superpixel i. The label y_i of superpixel is given by the majority label within the superpixel. In the image above $y_i = 1$ as the object category is person and majority of the superpixel lies within a person.

extend it to multiple super-pixelisations (Section 6.2.1). Denote by $\mathbf{x}_i$ the feature response extracted from a superpixel indexed by i, and by y_i the ground truth label of the superpixel i. As superpixels sometimes straddle object boundaries, they can contain more than one object class. In such cases, the majority label within superpixel i is chosen as its label y_i (an alternative is to ignore superpixels with more than one class, we observed empirically that this choice doesn't make much difference in performance). The feature response $\mathbf{x}_i$ is obtained by concatenating histograms of visual words, histograms of RGB colours, and location within the image (see Figure 6.1 for a visualisation, and Section 6.2.2 for implementation details). We

deal with only a single object category at a time, hence the label y_i is binary, i.e $y_i \in \{-1, +1\}$ with -1 denoting background and $+1$ denoting the object category.

To train classifiers to predict the label y_i of a superpixel given its feature response $\mathbf{x}_i$, a fully supervised training dataset is used. The Graz dataset (Opelt et al., 2006) provides for such a training set with three different object categories corresponding to people, bikes and cars. Each category has about 300 images with corresponding pixelwise segmentations of each object category. This dataset can be used to extract training points $(\mathbf{x}_i, y_i)$ which are used to train SVM classifiers. In this work, simple linear classifiers are used, as they are very efficient and scale to large amounts of training data. The vector quantisation of SIFT/RGB features to obtain visual words already introduces significant non-linearity in our feature space. Explicit feature maps (Vedaldi and Zisserman, 2010) are further used to introduce non-linearity at the classifier level. In particular, the approximate χ^2 kernel map introduced in (Vedaldi and Zisserman, 2010) is used to map the visual words and RGB histograms before concatenating them into the feature $\mathbf{x}_i$ (see Section 6.2.2 for details). For a binary classifier, the discriminant function is denoted by $f(\mathbf{x}, \mathbf{w}) : \mathcal{X} \times \mathcal{W} \rightarrow \mathbb{R}$, where $\mathbf{w}$ denotes the parameters of the classifier, $f(\mathbf{x}, \mathbf{w}) > 0$ signifies that the superpixel with feature vector $\mathbf{x}$ is foreground and background if $f(\mathbf{x}, \mathbf{w}) < 0$. The corresponding output label can thus be simply written as $\text{sign}(f(\mathbf{x}, \mathbf{w}))$. For a linear classifier, the discriminant function is defined as $f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T\mathbf{x}$. The linear classifier is trained using the SVM cost function described later in Section 6.2.2.

6.2 Multiple segmentations

The previous section discussed classifying superpixels obtained from a single super-pixelisation only. Often, a single super-pixelisation is not enough to accurately delineate the object boundaries. Using multiple segmentations has been proposed as a solution to generate more robust bottom up segmentations (Russell et al., 2006; Malisiewicz and Efros, 2007; Pantofaru et al., 2008). With multiple segmentations, it is much more likely that one of the segmentations snaps to the correct object boundary. Multiple segmentations also provide multiple regions of support for classifying each pixel, thus increasing the reliability of classification. Having multiple superpixel features for each pixel can also be thought of as having multiple kernels for learning (see Section 6.4), and the benefits of multiple kernel learning have been discussed in detail in Vedaldi et al. (2009); Gehler and Nowozin (2009). In the next section, we explore different ways of learning from multiple super-pixelisations.

6.2.1 Combining multiple segmentations

Before going into the details of the different methods of learning from multiple super-pixelisations, we define the notion of IofR's (i.e. **I**ntersection **of** **R**egions) as introduced by Pantofaru et al. (2008). An IofR is a region obtained as a result of intersecting multiple segmentations. These are easily understood from the illustration in Figure 6.2. An IofR corresponds to a region where all the super-pixelisations agree, hence they become the new atomic units for classification in the methods described below. Some additional notation is introduced to describe multiple segmentations. The variable s is used to index over different super-pixelisations, $s \in \{1, \cdots, S\}$. Also, an additional indexing function $spix(p, s)$, that gives the index of the super-

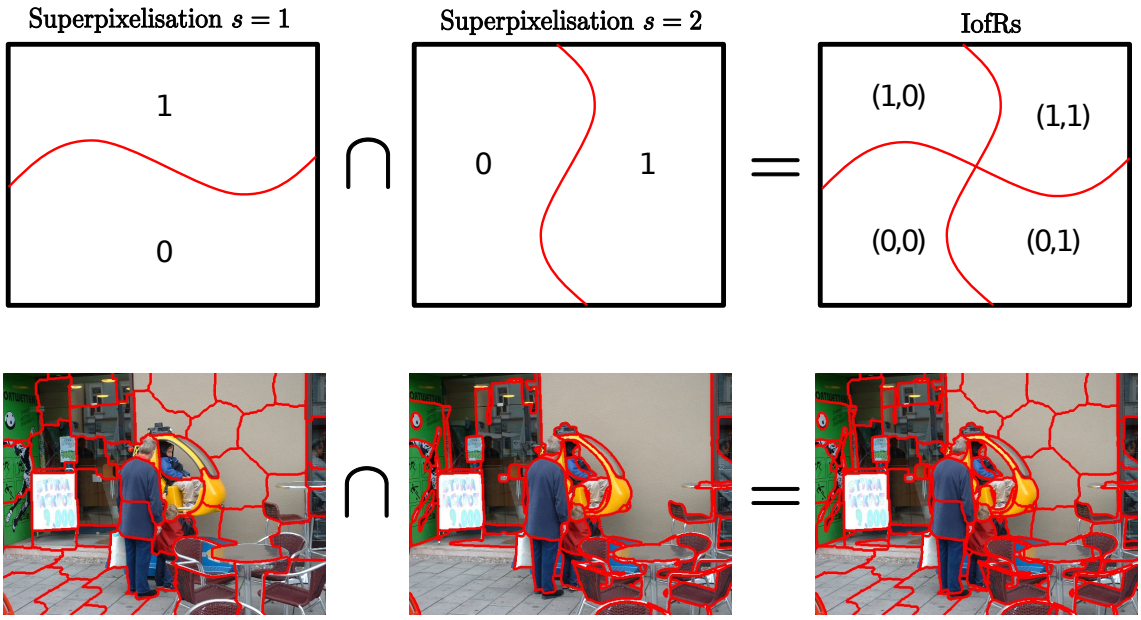


Figure 6.2: *Intersections of Regions (IofR).* Top row: A toy super-pixelisation explaining the concept of IofR as introduced by Pantofaru et al. (2008). The IofR's are shown by intersecting two very simple segmentations of the rectangular box. The superpixels in each of the toy super-pixelisations are indexed by the numbers 0 and 1. An IofR can be thought of as atomic regions where all the super-pixelisations agree. The same idea of IofR's extends to the intersection of more than 2 segmentations. Bottom row: IofR's obtained by intersecting two real super-pixelisations of an image.

pixel containing pixel p in the super-pixelisation indexed by s is used for notational convenience. Thus the feature vector of the superpixel to which pixel p belongs in the super-pixelisation s is given by $\mathbf{x}_{spix(p,s)}$. Using this notation, four different ways of learning from multiple segmentations are described as follows:

1. *Avg-Indep*: This method is similar to the one introduced by Pantofaru et al. (2008). A separate classifier $f(\mathbf{x}, \mathbf{w}_s)$ is learnt independently for each super-pixelisation $s \in \{1, \cdots, S\}$ as described in Section 6.1 ($\mathbf{w}_s$ denotes the parameters learnt for the super-pixelisation s). At test time, the corresponding classifier outputs for each super-pixelisation are simply averaged at each pixel.

More formally, the classifier output at pixel p denoted by $f_p(\mathbf{w}_1, \cdots, \mathbf{w}_s)$ is defined as: $f_p(\mathbf{w}_1, \cdots, \mathbf{w}_s) = \frac{1}{S}\sum_{s=1}^{S} f(\mathbf{x}_{spix(s,p)}, \mathbf{w}_s)$. This method only differs slightly from Pantofaru et al. (2008) in that they convert the classifier outputs into probabilities before averaging them, as opposed to simply averaging the classifier outputs in this case.

2. *Avg-Union*: This method is similar to *Avg-Indep* in that the classifier outputs of different super-pixelisations are simply averaged at test time. However, during training, instead of learning a separate classifier $f(\mathbf{x}, \mathbf{w}_s)$ for every super-pixelisation s, a single classifier $f(\mathbf{x}, \mathbf{w})$ is trained by taking of union of all the training data across all super-pixelisations and learning a single classifier. This is equivalent to replicating each image S times, and treating the multiple segmentations of the image as single super-pixelisations of copies of the image. At test time, it uses the same formulation as *Avg-Indep*, i.e. the classifier output at pixel p is given by $f_p(\mathbf{w}) = \frac{1}{S}\sum_{s=1}^{S} f(\mathbf{x}_{spix(s,p)}, \mathbf{w})$.

3. *LPβ-Indep*: This method is a generalisation of the *Avg-Indep* method. A separate classifier $f(\mathbf{x}, \mathbf{w}_s)$ is learnt for each super-pixelisation s in the same way as *Avg-Indep*. But rather than simply averaging the classifier outputs at test time, a linear combination of these classifiers is learnt. The classifier output at pixel p is given by: $f_p(\mathbf{w}_1, \cdots, \mathbf{w}_S, \beta_1, \cdots, \beta_S) = \sum_{s=1}^{S} \beta_s f(\mathbf{x}_{spix(s,p)}, \mathbf{w}_s)$. The weights β_s are normalised (i.e $\sum_{s=1}^{S} \beta_s = 1$) and are learnt using a discriminative training procedure as in Gehler and Nowozin (2009) (see Section 6.2.2 for details). Unlike the weight vectors $\mathbf{w}_s$ which are trained over superpixelisations, the weights β_s are trained over IofR's because each IofR gets a different classification score.

4. *IofR-Joint*: As with *Avg-Indep* and *LPβ-Indep,* this method learns a separate classifier for each super-pixelisation s. However, these classifiers are learnt together in a single joint optimisation as opposed to learning them independently. In this method, a feature descriptor is extracted for each IofR, as opposed to each superpixel for the case of *Avg-Indep* and *LPβ-Indep*. The feature descriptor of each IofR is formed by concatenating the feature vectors of all the superpixels it belongs to (see Figure 6.3 for an illustration). This can be formally written down as $\mathbf{x}_{\text{IofR}(p)} = [\mathbf{x}_{spix(p,1)}; \mathbf{x}_{spix(p,2)}; \cdots; \mathbf{x}_{spix(p,S)}]$ where $\mathbf{x}_{\text{IofR}(p)}$ denotes the feature vector of the IofR that pixel p belongs to. A linear classifier $f(\mathbf{x}, \mathbf{w})$ is now trained over these features as for the case of a single super-pixelisation. Note that the dimensionality of the $\mathbf{w}$ vector is now S times the dimensionality of the $\mathbf{w}$ vectors in the previous three methods. The $\mathbf{w}$ vector for this case can be interpreted as a concatenation of $\mathbf{w}$ vectors for all the super-pixelisations, i.e. $\mathbf{w} = [\mathbf{w}_1; \cdots; \mathbf{w}_S]$, and this simple concatenation leads to the joint optimisation across all the $\mathbf{w}_s$ vectors. At test time, the classifier output at pixel p is given by: $f_p(\mathbf{w}) = f(\mathbf{x}_{\text{IofR}(p)}, \mathbf{w})$.

6.2.2 Implementation details

For generating multiple super-pixelisations, we use three different methods by Arbeláez et al. (2011); Veksler et al. (2010); Vedaldi and Soatto (2008). A total of five different super-pixelisations for each image are generated, and abbreviated as follows. The first segmentation is generated using the gPb superpixels of Arbeláez et al. (2011) (termed *Spixel-gPb*), two using QuickShift (Vedaldi and Soatto, 2008) (termed *Spixel-qShift-1* and *Spixel-qShift-2*) and two using Veksler et al. (2010)

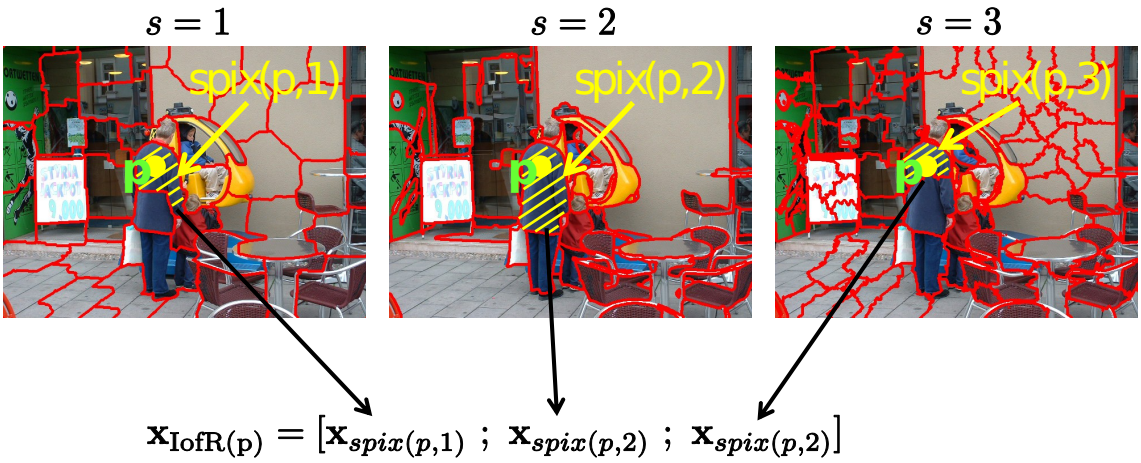
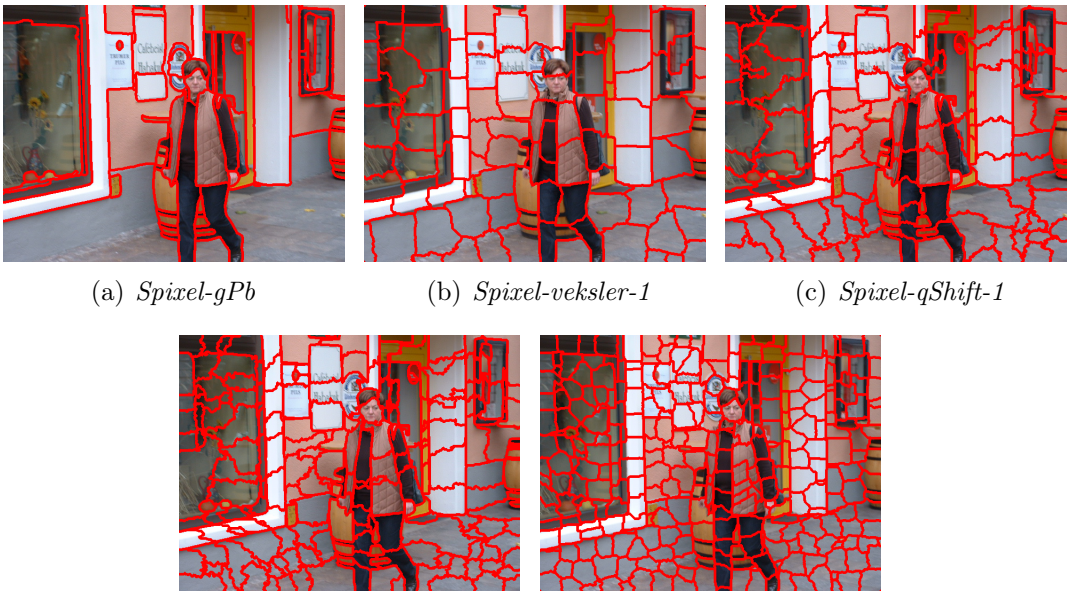


Figure 6.3: *IofR-Joint features.* Three different super-pixelisations indexed by $s = 1, 2$ and 3 are shown above. The superpixel corresponding to pixel p is shaded in yellow in each image. The feature vector for the IofR containing pixel p (denoted by $\mathbf{x}_{\mathrm{IofR}(p)}$) is obtained by concatenating the feature vectors of all the superpixels to which p belongs. The IofR itself is not visualised in the figure, it can be obtained by intersecting the shaded yellow regions. One feature vector is extracted for each IofR (and not per pixel, pixels that belong to the same IofR share the same feature vector).

(termed *Spixel-veksler-1* and *Spixel-veksler-2*). The parameters for each method are set so that each super-pixelisation captures different scales (see Figure 6.4). E.g., for the cars object category in the Graz dataset, on average, *Spixel-gPb* has 43 super-pixels per image, *Spixel-veksler-1* has 105, *Spixel-qShift-1* has 156, *Spixel-qshift-2* has 229 and *Spixel-veksler-2* has 331 superpixels per image. These superpixels give roughly 4600 IofR's per image. For *IofR-Joint*, with 150 training images per category (as is the case on the Graz dataset), that means a total of $150 \times 4600 = 690,000$ training points, as each IofR gives one training point.

The feature vector $\mathbf{x}$ as illustrated in Figure 6.1, is a concatenation of three different descriptors. The first descriptor used is a bag of visual words. Visual words are obtained by computing SIFT (Lowe, 2004) vectors sampled densely in a 5×5 grid across the image, and assigning each descriptor to its nearest element in



(a) *Spixel-gPb* (b) *Spixel-veksler-1* (c) *Spixel-qShift-1*

(d) *Spixel-qShift-2* (e) *Spixel-veksler-2*

Figure 6.4: *Multiple super-pixelisations.* Superpixelisations for the five different methods discussed in Section 6.2.2 are outlined above for an image from the Graz-people dataset. The superpixels span different scales, starting with large superpixels for *Spixel-gPb* to many smaller superpixels pixels for *Spixel-veksler-2*.

a vocabulary. The dense SIFT implementation in VlFeat (Vedaldi and Fulkerson, 2010) is used to compute the SIFT descriptors. The descriptors are computed at four different scales corresponding to sizes $[4, 6, 8, 10]$ as documented in VlFeat. The vocabulary is learnt in an unsupervised way using k-means clustering on SIFT descriptors sampled from the entire dataset, with $k = 1600$. The final descriptor extracted from each superpixel is a histogram of visual words contained within it. An explicit χ^2 feature map (Vedaldi and Zisserman, 2010) is applied to the histogram to introduce non-linearity in the classifier. If we denote by $\phi(\mathbf{h})$ the χ^2 mapped feature of a histogram denoted by $\mathbf{h}$, with h_k denoting the k^{th} entry in the histogram, then

each entry h_k is mapped to three corresponding entries in $\phi(\mathbf{h})$:

$$\phi(h_k) = \begin{bmatrix} 0.8\sqrt{h_k} \\ 0.6\cos\big(0.6\log(h_k)\big) \\ 0.6\sin(0.6\log(h_k)\big) \end{bmatrix}$$

The second descriptor extracted from each superpixel is a histogram of RGB values. This histogram is compute analogously to the histogram of visual words – each pixel colour is mapped to its nearest colour in a vocabulary and a histogram of word occurrences is used to describe the superpixel (a vocabulary of size 500 is used for the RGB histogram). A χ^2 feature mapping is applied to this histogram in the same way as the visual word histogram. The third feature is the position of the superpixel relative to the image. The image is divided into a 6×8 grid, and an indicator variable is used to denote the location of the centroid of the superpixel (see Figure 6.1 for a visualisation). With all these three features concatenated, the total dimensionality of the feature vector $\mathbf{x}$ equals $1600 \times 3 + 500 \times 3 + 48 = 6348$.

Once the features vectors $\mathbf{x}_i$ and the corresponding labels y_i are extracted from all images, LibLinear (Fan et al., 2008) is used to train the linear classifiers $f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T\mathbf{x}$. The cost function used to train LibLinear is:

$$\mathbf{w}^* = \arg\min \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N} \max(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i)$$

where N is the total number of training points extracted from all images. The parameter λ is set by cross validation on a subset of the training data. For the case of *LPβ-Indep*, once the parameters $\mathbf{w}_s$ for each super-pixelisation have been learnt using LibLinear, learning the β parameters involves another training stage.

The following cost function suggested in Gehler and Nowozin (2009) is used for this second training stage:

$$\begin{aligned} \min_{\beta,\xi,\rho} \quad & -\rho + \frac{1}{\mu N}\sum_{p=1}^{N}\xi_p \\ s.t. \quad & y_p \sum_{s=1}^{S}\beta_s f(\mathbf{x}_{spix(p,s)}, \mathbf{w}_s) + \xi_p \geq \rho \quad \forall p = 1, \cdots, N \\ & \sum_{s=1}^{S}\beta_s = 1, \quad \beta_s \geq 0, \quad s = \{1, \cdots, S\}, \quad \xi_p \geq 0, \quad \forall p = 1, \cdots, N \end{aligned}$$

The parameter μ is a meta-parameter of this system (like the regularisation parameter λ in the SVM cost function), and is again set using cross validation. The cost function above is similar to the SVM cost function, where ρ corresponds to the margin on correctly classified instances and ξ_p is a slack variable which allows for violations in the training data. The weights β_s are constrained so as to learn a convex combination of classifiers $f(\mathbf{x}, \mathbf{w}_s)$. The above optimisation problem is an instance of a linear programming problem and is optimised using Mosek: Optimization Toolkit (2010).

6.3 Experiments on multiple segmentations

The different methods for learning from multiple segmentations described in Section 6.2.1 are evaluated against the segmentation tasks on the Graz (Opelt et al., 2006) dataset and the PASCAL VOC 2010 Dataset (Everingham et al., 2010). The Graz dataset consists of three separate collections, each containing a different object category, viz. people, bikes and cars. Each image in the dataset contains an instance of the object, i.e. there are no images containing only the background class in the

Method	Graz-people	Graz-bikes	Graz-cars
Spixel-gPb	54.51	56.11	57.66
Spixel-veksler-1	57.11	62.61	62.43
Spixel-qShift-1	58.04	58.96	62.06
Spixel-qShift-2	57.10	59.04	61.44
Spixel-veksler-2	55.49	61.68	59.09
Avg-Indep	62.17	66.77	67.02
Avg-Union	62.79	66.73	67.64
LPβ-Indep	62.43	66.77	67.21
IofR-Joint	**64.46**	**67.81**	**68.49**

Table 6.1: *Segmentation results on the Graz dataset.* The first five rows show performances of individual super-pixelisation methods and the bottom four rows list the performance of the four superpixel combinations discussed in Section 6.2.1. There is a clear advantage in combining multiple super-pixelisations on the Graz object categories, where the single best super-pixelisation performs about $6-8\%$ lower than the best superpixel combination (*IofR-Joint*) across all three categories.

dataset, thus providing for a balanced training set of positives and negatives. The PASCAL VOC Dataset, on the other hand, is a single collection that contains images labelled with 20 object categories. In this work, we train and test on three object categories from this dataset, viz. Aeroplane, Car and People. We also restrict the dataset to images containing at least one instance of the object category, keeping the setting similar to the Graz dataset.

To evaluate quantitative performance, the overlap score of the predicted segmentations with the ground truth segmentation is computed across all images as in the PASCAL VOC Challenge (Everingham et al., 2010). The results for single super-pixelisations as well as different ways of combining multiple super-pixelisations are summarised in Tables 6.1 and 6.2, and Figure 6.5 shows some qualitative results. Recalling the abbreviations used for the different methods, the five single super-pixelisations are denoted by *Spixel-gPb*, *Spixel-qShift-1, Spixel-qShift-2, Spixel-veksler-1* and *Spixel-veksler-2*. The different methods for combining multiple

Method	Aeroplane	Cars	People	People-full
Spixel-gPb	61.08	41.34	40.62	21.89
Spixel-veksler-1	52.66	35.89	36.27	17.72
Spixel-qShift-1	55.75	37.87	34.37	14.69
Spixel-qShift-2	55.63	36.57	35.37	14.40
Spixel-veksler-2	53.11	33.75	33.86	14.55
Avg-Indep	62.51	41.23	38.00	21.27
Avg-Union	61.99	40.71	39.46	18.43
LPβ-Indep	64.02	43.33	39.26	**24.13**
IofR-Joint	**67.10**	**43.56**	**43.32**	—

Table 6.2: *Segmentation results on the PASCAL dataset.* On this dataset, the simple averaging methods for combining super-pixelisations perform slightly worse than the single best super-pixelisation, but *IofR-Joint* performs noticeably better than any single super-pixelisation. The performance on the PASCAL dataset is reported for the validation set, as the ground truth for the test data is not made publically available. The last column (People-full) represents the case where the entire PASCAL set (i.e. images both with and without people) is used for training and testing. The missing entry for *IofR-Joint* in this column is because it is not feasible to run it on such a large dataset due to memory constraints.

segmentations introduced in Section 6.2.1 are abbreviated as *Avg-Indep, Avg-Union, LPβ-Indep* and *IofR-Joint.*

For the Graz object categories, the first clearly visible trend in Table 6.1 is that combining the multiple super-pixelisations in any of the four ways mentioned above is better than just using a single super-pixelisation (e.g. for the case of cars, the best performing single super-pixelisation scores 62.43 compared to 68.49 for the *IofR-Joint* method). Amongst the different methods of learning from multiple segmentations, the first three (i.e *Avg-Indep*, *Avg-Union* and *LPβ-Indep*) all perform very similarly to each other. A minor gain in performance is achieved by learning across all the segmentations jointly using *IofR-Joint* (e.g. for the people category, the performance goes up from 62.79 for *Avg-Indep* to 64.46 for *IofR-Joint*). However, this gain is minor compared to the large computational expense of *IofR-Joint.* As

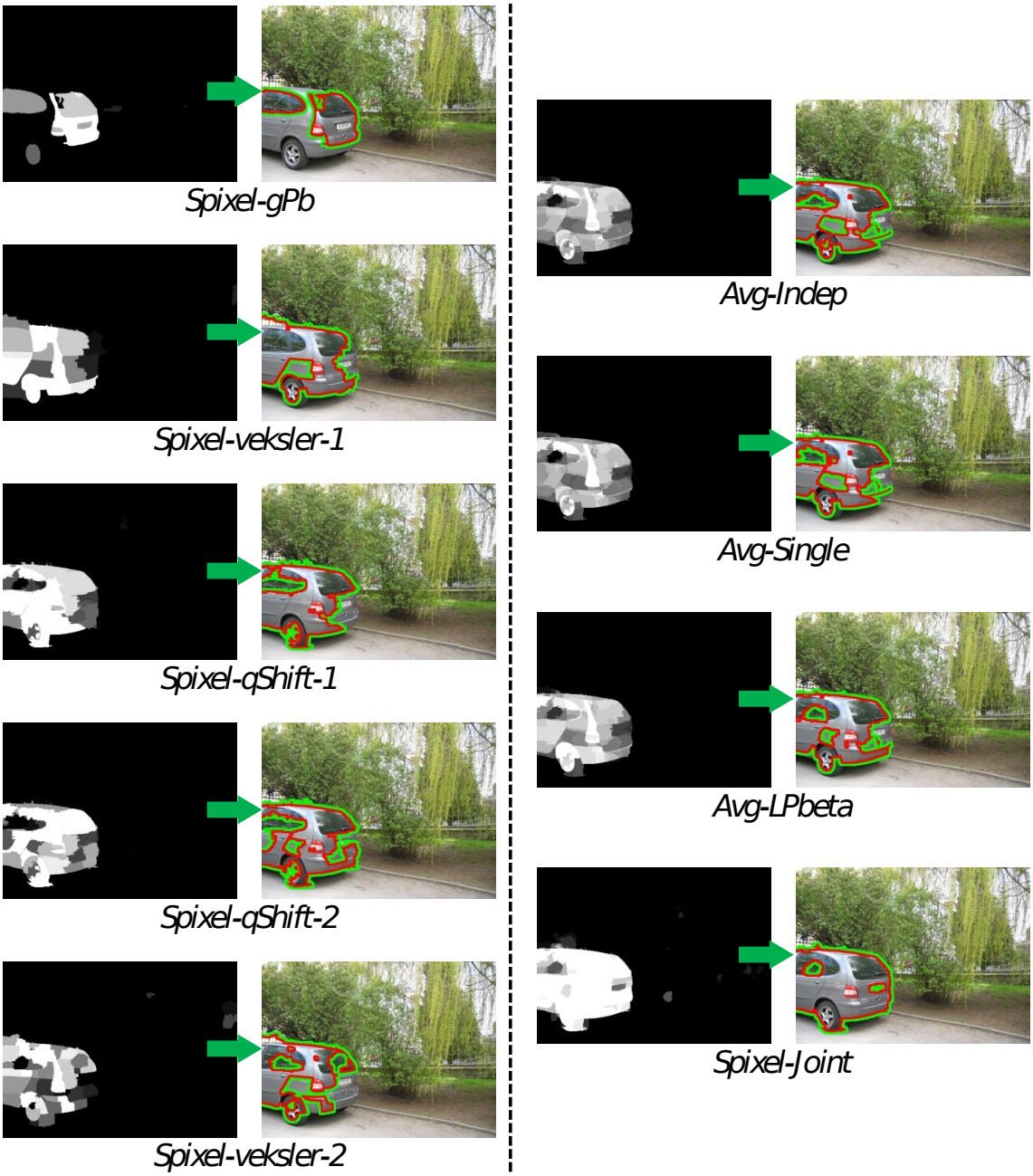


Figure 6.5: *Qualitative results for combining multiple segmentations.* The left column shows classifier outputs on single super-pixelisations, and the right column shows classifier outputs for the various methods to combine multiple super-pixelisations. The classifier outputs are truncated to the range $[-1, +1]$ and visualised as gray scale images. The corresponding segmentations (obtained by thresholding the classifier scores at 0) are shown as a two coloured boundary, with red pointing to the inside of the segmentation and green to the outside. The image above is taken from the test set in the Graz-cars object category.

explained in Section 6.2.1, *IofR-Joint* computes a feature vector for each IofR in an image. The number of IofR's is significantly more than the number of superpixels in any given super-pixelisation (e.g. for the case of cars, the number of IofR's in the training set is around $690,000$ compared to $50,000$ total number of superpixels for the single super-pixelisation *Spixel-veksler-2*). Thus, a simple averaging of classifier outputs across the different super-pixelisations as done in *Avg-Union* seems to be the most practical way of learning using multiple super-pixelisations.

On the PASCAL dataset, the overall performance across all methods is lower than that on the Graz dataset. This is because the PASCAL dataset has much more intra-class variation across the training and testing sets, providing for a harder learning task. Also, on this dataset, simple averaging methods (i.e. *Avg-Indep*, *Avg-Union* and *LPβ-Indep*) perform slightly worse than the single best super-pixelisation *Spixel-gPb*. This is because on the PASCAL dataset, the other four single super-pixelisations perform much worse than *Spixel-gPb*, and thus a simple averaging drags down the performance of *Spixel-gPb*. The *IofR-Joint* method however is still able to combine the five different super-pixelisations effectively, leading to an increase in performance (e.g. performance in the People category goes up from 40.62 for *Spixel-gPb* to 43.32 for *IofR-Joint*).

6.4 Relation to multiple kernel learning

The *IofR-Joint* and *LPβ-Indep* methods of learning from multiple segmentations discussed above can also be interpreted in the light of multiple kernel learning. For *IofR-Joint*, the feature vector computed for each IofR is a concatenation of features obtained from each super-pixelisation. This can be equivalently thought of

as taking the average of multiple linear kernels, where each kernel is obtained from the feature vector of a single super-pixelisation, and the concatenation of feature vectors is equivalent to averaging kernels for the case of linear classifiers. The *LPβ-Indep* method is inspired from the LPboost learning method discussed in Gehler and Nowozin (2009), where instead of learning across multiple kernels jointly, a simple linear combination of classifiers learnt from each kernel is trained using a boosting like cost function.

The other two methods of learning, i.e. *Avg-Indep* and *Avg-Union* do not correspond to any kind of multiple kernel learning. They simply average the outputs of multiple classifiers. One difference in multiple kernel learning using the *IofR-Joint* method, compared to multiple kernel learning across different features as done in Gehler and Nowozin (2009) is that *IofR-Joint* doesn't scale very well as the number of super-pixelisations increase. Adding another super-pixelisation leads to a rapid increase in the number of IofR's, as the number of intersections between superpixels increases. Thus the number of training points increases rapidly with the increasing number of super-pixelisations. The *LPβ-Indep* method scales better with increasing number of super-pixelisations, as the complexity of learning the classifier parameters $\mathbf{w}_s$ depends on the number of superpixels and not upon the number of IofR's. The complexity of the second learning stage of *LPβ-Indep* does indeed depend on the number of IofR's, but the optimisation problem is much smaller, as for every IofR, only S number of classifier outputs are needed (as opposed to S feature vectors for *IofR-Joint*).

6.5 Learning pairwise terms

The various learning methods uptil now treat superpixels as independent entities, i.e. the dependence between labels of neighbouring pixels is not modelled. In Chapters 3–5, we saw that Conditional Random Field (CRF) formulations were used to add pairwise dependencies between labels of neighbouring pixels. These pairwise terms helped model simple properties of object boundaries such as the total contrast along the boundary, Euclidean boundary length etc. Such simple pairwise terms have also been used in CRF formulations for semantic segmentation by Winn and Shotton (2006); Shotton et al. (2006); Ladicky et al. (2009). The pairwise terms in these formulations involve only a few parameters that control the strength of the pairwise term and are often set by hand or cross validation. In the following sections, we introduce novel pairwise terms that capture more complex boundary statistics based on histograms of visual words along the boundary. Introducing these terms leads to significantly more parameters for the pairwise terms, and thus we resort to more principled structured output learning methods (Tsochantaridis et al., 2004) for setting these parameters.

In particular, we use the margin rescaled SVM formulation introduced in Tsochantaridis et al. (2004) for learning parameters (described in detail in Section 6.6.1). The margin rescaled formulation provides a globally optimisable cost function to train CRF parameters assuming inference is tractable. Szummer et al. (2008) used the same SVM formulation for learning parameters for an interactive segmentation system. Their unary terms were obtained from colour models estimated from user specified regions and the pairwise terms in their CRF were restricted to Ising models and pairwise contrasts. Thus, the number of parameters learnt in their system was

small, and the appearance models were not learnt jointly with the pairwise terms. In this work, more versatile appearance models and pairwise terms are used and learnt jointly. The appearance terms are outputs of linear classifiers trained upon the superpixel feature vectors as discussed in Section 6.1. The pairwise terms consists of histograms of visual words computed at the boundary of the object and are described in Section 6.6. We only use single super-pixelisations to learn the CRF in this section (the super-pixelisation *Spixel-veksler-1* is used for the Graz dataset, and *Spixel-gPb* used for the PASCAL dataset, as these are the best performing ones as discussed in Section 6.3). It would be possible to use multiple super-pixelisations by treating the IofR's as superpixels and using the same CRF model. As the number of IofR's is significantly more than the number of superpixels in any single super-pixelisation, we use single super-pixelisations to keep the computational complexity under control.

6.6 CRF model

In this section, the energy function for labelling a single super-pixelisation for an image is described. The previous notation is extended to introduce the pairwise terms in our CRF formulation. Let $\mathbf{x}_i$ denote the feature vector for superpixel i as defined previously and y_i denote its label. As before, we consider the problem of segmenting a single object category in this section, therefore $y_i \in \{-1, +1\}$, where -1 denotes the background class and $+1$ denotes the foreground class. Consider two neighbouring superpixels i and j (two superpixels are considered to be neighbouring if they share a common boundary). Let $\mathbf{x}_{ij}$ denote the feature describing the boundary between two neighbouring superpixels i and j. The feature $\mathbf{x}_{ij}$ is computed as

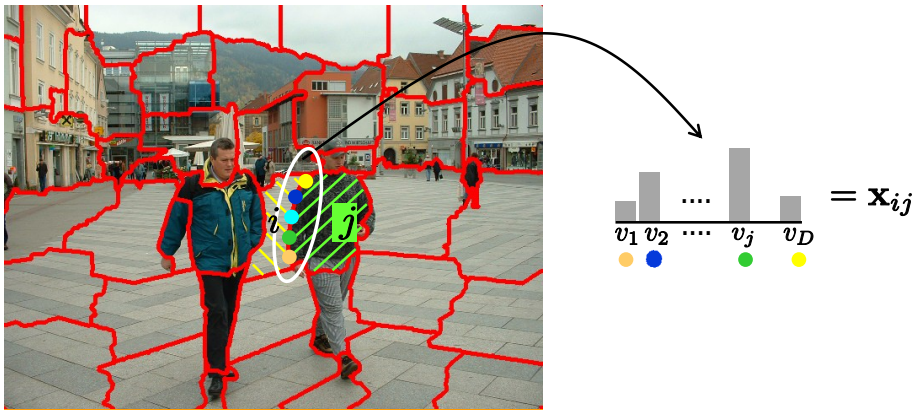


Figure 6.6: *Edge features.* The figure above visualises the feature $\mathbf{x}_{ij}$ between two adjacent superpixels indexed by i and j. The feature consists of a unnormalised histogram of quantised SIFT vectors (visual words) along the shared boundary between the two superpixels. The histogram is unnormalised, so that summing up these individual histograms along the boundary (i.e. the term $\sum_{ij\in\mathcal{N}} \mathbf{x}_{ij}(y_i \neq y_j)$ in (6.1)) gives the histogram of the entire object boundary.

an unnormalised histogram of visual words along the boundary between superpixels i and j (see Figure 6.6 for an illustration). The energy for a labelling $\mathbf{y}$ given an image $\mathbf{x}$ is then given by:

$$E(\mathbf{y}|\mathbf{x}, \mathbf{w}_{\text{app}}, \mathbf{w}_{\text{edge}}) = \sum_{i=1}^{N} -y_i \mathbf{w}_{\text{app}}^T \mathbf{x}_i + \mathbf{w}_{\text{edge}}^T \sum_{ij\in\mathcal{N}} \mathbf{x}_{ij}(y_i \neq y_j) \tag{6.1}$$

where $\mathbf{w}_{\text{app}}$ denotes the appearance parameters and $\mathbf{w}_{\text{edge}}$ the pairwise parameters. These parameters are learnt using the training procedure described in Section 6.6.1. The unary terms in the above energy function are the same as any standard linear SVM classifier would have, they favour the label $y_i = 1$ if $\mathbf{w}_{\text{app}}^T \mathbf{x}_i > 0$ and the label $y_i = -1$ when $\mathbf{w}_{\text{app}}^T \mathbf{x}_i < 0$. The pairwise terms $\sum_{ij\in\mathcal{N}} \mathbf{x}_{ij}(y_i \neq y_j)$ compute an unnormalised histogram of visual words along the segmentation boundary, and the term $\mathbf{w}_{\text{edge}}^T \sum_{ij\in\mathcal{N}} \mathbf{x}_{ij}(y_i \neq y_j)$ then simply scores the entire histogram. The parameters $\mathbf{w}_{\text{edge}}$ are trained so that the penalty for breaking an edge that does

not lie on the object boundary is high and is low for breaking an edge on the object boundary. $\mathbf{w}_{\text{edge}}$ is constrained to be ≥ 0 so that the energy remains submodular. Submodularity guarantees exact inference which is needed for training CRF using structured output methods as explained in Section 6.6.1. The parameters $\mathbf{w}_{\text{app}}$ and $\mathbf{w}_{\text{edge}}$ are trained together jointly as described in the following section.

6.6.1 Structured output learning of CRF

To train the parameters $\mathbf{w}_{\text{app}}$ and $\mathbf{w}_{\text{edge}}$ in the CRF formulation in (6.1), a discriminative training procedure that maximises the margin between the energy of the ground truth and all the other segmentations is used. Before describing the training cost function, we rewrite the energy function in (6.1) in a more general form as done in Tsochantaridis et al. (2004):

$$E(\mathbf{y}|\mathbf{x}, \mathbf{w}_{\text{app}}, \mathbf{w}_{\text{edge}}) = \begin{bmatrix} \mathbf{w}_{\text{app}}^T & \mathbf{w}_{\text{edge}}^T \end{bmatrix} \begin{bmatrix} \sum_{i=1}^{N} -y_i \mathbf{x}_i \\ \sum_{ij \in \mathcal{N}} (y_i \neq y_j) \mathbf{x}_{ij} \end{bmatrix} = \mathbf{w}^T \psi(\mathbf{x}, \mathbf{y})$$

Writing the energy function as above makes explicit the linear dependence of the energy function on the parameter vector $\mathbf{w} = [\mathbf{w}_{\text{app}}; \mathbf{w}_{\text{edge}}]$. To introduce the max-margin training cost function, some additional notation for the training set is introduced. Let the subscript t index over the training data, and the set $\mathcal{K} = \left\{(\mathbf{x}_t, \mathbf{y}_t)\ ,\ t = \{1, \cdots, T\}\right\}$ denote the training data. Here the term $\mathbf{x}_t$ denotes the image t and $\mathbf{y}_t$ denotes the ground truth labels of all the superpixels in image t. We use the margin rescaled cost function for training the parameters $\mathbf{w}$ as described in

Tsochantaridis et al. (2004):

$$\begin{aligned}
\min_{\mathbf{w},\xi} \quad & \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{T}\sum_{t=1}^{T}\xi_t \\
\text{s.t.} \quad & \forall \bar{\mathbf{y}}_1 \in \mathcal{Y}_1 \ : \mathbf{w}^T[\psi(\mathbf{x}, \bar{\mathbf{y}}_1) - \psi(\mathbf{x}, \mathbf{y}_1)] \geq \Delta(\mathbf{y}_1, \bar{\mathbf{y}}_1) - \xi_1 \\
& \vdots \\
& \forall \bar{\mathbf{y}}_T \in \mathcal{Y}_T \ : \mathbf{w}^T[\psi(\mathbf{x}, \bar{\mathbf{y}}_T) - \psi(\mathbf{x}, \mathbf{y}_T)] \geq \Delta(\mathbf{y}_T, \bar{\mathbf{y}}_T) - \xi_T \\
& \mathbf{w}_{\text{edge}} \geq 0, \xi_1 \geq 0, \cdots, \xi_T \geq 0
\end{aligned}$$

where the term $\Delta(\mathbf{y}_t, \bar{\mathbf{y}}_t)$ is the loss function used to rescale the margin and the set $\mathcal{Y}_t$ denotes the set of all possible labellings for image t. The loss function $\Delta(\mathbf{y}_t, \bar{\mathbf{y}}_t)$ measures the penalty paid for labelling the image t as $\bar{\mathbf{y}}_t$ given the ground truth labelling $\mathbf{y}_t$. Like in Szummer et al. (2008), we use the simple hamming loss, i.e. $\Delta(\mathbf{y}_t, \bar{\mathbf{y}}_t) = \sum_{i \in spix(t)} y_{t,i} \neq \bar{y}_{t,i}$ (where $spix(t)$ is the set of superpixel indices for image t). The above cost function is minimised using the cutting plane algorithm of Joachims et al. (2009). Joachims et al. (2009) first rewrite the above optimisation problem as an equivalent problem with only a single slack (called the 1-slack formulation):

$$\begin{aligned}
\min_{\mathbf{w},\xi} \quad & \frac{\lambda}{2}\|\mathbf{w}\|^2 + \xi \qquad (6.2) \\
\text{s.t.} \quad & \forall (\bar{\mathbf{y}}_1, \cdots, \bar{\mathbf{y}}_T) \in \{\mathcal{Y}_1 \times \cdots \times \mathcal{Y}_T\} : \\
& \frac{1}{T}\mathbf{w}^T \sum_{t=1}^{T}[\psi(\mathbf{x}, \bar{\mathbf{y}}_t) - \psi(\mathbf{x}, \mathbf{y}_t)] \geq \frac{1}{T}\sum_{t=1}^{T}\Delta(\mathbf{y}_t, \bar{\mathbf{y}}_t) - \xi \\
& \mathbf{w}_{\text{edge}} \geq 0, \xi \geq 0
\end{aligned}$$

Algorithm 6.1 *Structured SVM training with margin rescaled cost function.*

1. Input training data: $\mathcal{K} = \{(\mathbf{x}_1, \mathbf{y}_1), \cdots, (\mathbf{x}_T, \mathbf{y}_T)\}, \lambda, \epsilon$ where ϵ is the convergence threshold
2. Initialise constraint set $\mathcal{C}$ to be empty, i.e. $\mathcal{C} = \phi$
3. **repeat**
4. $$(\mathbf{w}, \xi) = \arg\min_{\mathbf{w},\xi} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \xi$$
$$\text{s.t.} \quad \forall(\bar{\mathbf{y}}_1, \cdots, \bar{\mathbf{y}}_T) \in \mathcal{C} :$$
$$\frac{1}{T}\mathbf{w}^T \sum [\psi(\mathbf{x}, \bar{\mathbf{y}}_t) - \psi(\mathbf{x}, \mathbf{y}_t)] \geq \frac{1}{T}\sum_{t=1}^{T} \Delta(\mathbf{y}_t, \bar{\mathbf{y}}_t) - \xi$$
$$\mathbf{w}_{\text{edge}} \geq 0, \xi \geq 0$$
5. **for** $t = 1, \cdots, T$ **do**
6. $\hat{\mathbf{y}}_t = \arg\min_{\mathbf{y}} \{\mathbf{w}^T \psi(\mathbf{x}_t, \mathbf{y}) - \Delta(\mathbf{y}_t, \mathbf{y})\}$
7. **end for**
8. $\mathcal{C} = \mathcal{C} \cup \{(\hat{\mathbf{y}}_1, \cdots, \hat{\mathbf{y}}_T)\}$
9. **until** $\frac{1}{T}\sum_{t=1}^{T} \Delta(\mathbf{y}_t, \hat{\mathbf{y}}_t) - \frac{1}{T}\mathbf{w}^T \sum [\psi(\mathbf{x}, \hat{\mathbf{y}}_t) - \psi(\mathbf{x}, \mathbf{y}_t)] \leq \xi + \epsilon$
10. return $\mathbf{w}$

The number of constraints in the above equation is exponential ($= 2^{N_1 + \cdots + N_T}$ where N_t is the number of superpixels in image t) and thus cannot be solved in the above form. The cutting plane algorithm works by iteratively constructing a subset of these constraints that lower bound the original problem and then improves upon those lower bounds by adding more constraints. These constraints are generated by solving an optimisation problem to find the most violated constraints given a vector $\mathbf{w}$ (see Algorithm 6.1 for a formal explanation). At any given time, only a small subset of constraints is active thus keeping the problem tractable. The optimisation problem in (6.2) is a standard convex quadratic optimisation problem

and can be solved using off the shelf solvers (such as `quadprog` in `matlab` or Mosek: Optimization Toolkit (2010)). The cutting plane algorithm has convergence guarantees that are linear in the number of training samples, i.e. the number of iterations taken to converge is independent of the number of training samples, and the cost of each iteration is linear in the number of training samples. The cutting plane algorithm to minimise (6.2) is detailed in Algorithm 6.1. Note, Step 6 in Algorithm 6.1 corresponds to an inference step and is optimised using graph-cuts.

6.7 Experiments on pairwise terms

The CRF learning is evaluated with single super-pixelisations on different categories in the Graz and the PASCAL VOC datasets. As in Section 6.3, the task is to predict binary labels for object categories in each test image, and performance is evaluated using the overlap score of the predicted segmentations with the ground truth segmentations. Simple variations on the pairwise terms are explored, these are abbreviated as follows:

1. *UnaryOnly*: No pairwise terms are used in this formulation. This degenerates to learning a simple SVM classifier given a single super-pixelisation as in Section 6.1 and is our baseline for evaluating the effect of adding pairwise terms.

2. *CRF-Ising*: Simple Potts model penalties are used as pairwise terms. This can be easily implemented in the energy function in (6.1) by setting the edge feature $\mathbf{x}_{ij}$ to the scalar value 1.

3. *CRF-Contrast*: Contrast dependant penalties in the same spirit as in (2.11)

Method	Graz-people	Graz-bikes	Graz-cars	Pascal-people
UnaryOnly	56.93	62.11	62.47	40.01
CRF-Ising	58.37	64.72	67.03	39.77
CRF-Contrast	58.91	64.76	67.49	39.15
CRF-Sift	62.41	65.06	**68.67**	**41.16**
CRF-Sift-Contrast	**62.55**	**65.40**	68.62	39.49

Table 6.3: *CRF learning results.* The abbreviations for the different methods are listed in Section 6.7. The first row (*UnaryOnly*) does not use any pairwise terms. The CRF formulations in the bottom three rows all perform significantly better than *UnaryOnly*. The best performance is obtained by *CRF-Sift* and *CRF-Sift-Contrast* that use our visual words based pairwise terms.

are used as pairwise terms. This is easily implemented in the energy function in (6.1) by setting the edge feature $\mathbf{x}_{ij}$ to a scalar value $\exp(-\beta c_{ij})$ where c_{ij} is the contrast between two superpixels i and j.

4. *CRF-Sift*: The unnormalised visual word histogram is used as a descriptor of the boundary. This is the feature vector illustrated in Figure 6.6.

5. *CRF-Sift-Contrast*: The edge feature for this particular edge descriptor is simply a concatenation of the edge features of *CRF-Sift* and *CRF-Contrast*. The visual word feature vector $\mathbf{x}_{ij}$ as shown in Figure 6.6 is simply concatenated with the scalar value of the contrast to give the new feature vector $\mathbf{x}'_{ij} = [\mathbf{x}_{ij}; \exp(-\beta c_{ij})]$.

The performances for these different edge features are summarised in Table 6.3 and some qualitative results shown in Figure 6.7. On the Graz dataset, the benefit of adding pairwise terms is clearly seen in the difference between *UnaryOnly* and the other *CRF* based edge features (e.g. on the Graz-cars dataset, the performance of *UnaryOnly* goes up from 62.47 to 68.67 for *CRF-Sift*). The benefit of using visual words based pairwise terms in the CRF over simple Potts model and Con-

(a) *UnaryOnly* (b) *CRF-Sift*

Figure 6.7: *Qualitative results for CRF learning.* Segmentation outputs on two test images from the Graz-people dataset. To the left are outputs with superpixel classification only, and to the right are segmentation outputs of our CRF based formulation.

trast dependant potentials is also evident from the quantitative evaluation – e.g., on the Graz-people dataset, the performance of *CRF-Contrast* goes up from 58.91 upto 62.55 for *CRF-Sift-Contrast*. Amongst the three different classes in the Graz dataset, the improvement using *CRF-Sift* over *UnaryOnly* is more significant for Graz-people and Graz-cars (about 6%) than for the Graz-bikes (about 3%). On the PASCAL-people dataset, a minor improvement is seen on adding the visual words based edge terms (i.e. performance goes up from 40.01 using *UnaryOnly* to 41.16 using *CRF-Sift*). In summary, the visual words based pairwise features are helpful in classifying object boundaries and provide a significant boost in performance compared to simply learning independent superpixel classifiers. There is not much difference in the performance between *CRF-Sift* and *CRF-Sift-Contrast* suggesting that concatenating a contrast dependant potential to the visual words based edge vector doesn't help much.

6.8 Conclusions

In this chapter, we have explored different ways of learning to segment from bottom up super-pixelisations. In the first part, we explored different ways to learn from multiple segmentations. Combining classifiers across multiple super-pixelisation gave a significant performance boost compared to using single super-pixelisations. We also found that a simple averaging of classifier outputs across multiple segmentations performs almost as well as jointly learning across multiple segmentations, while being much more simple to compute. In the second part, we introduced a CRF based formulation for segmentation with novel pairwise terms that model visual word statistics of the object boundary. In this formulation, the appearance and the pairwise parameters were learnt jointly, and we observed a significant benefit in using pairwise terms based on visual words as opposed to commonly used Potts potential based pairwise terms.

It would be natural to combine the two ideas presented in this chapter, i.e. combine learning from multiple super-pixelisations with the novel pairwise terms. This is left as future work as discussed in Section 7.1.2.

Chapter 7

Conclusions

In this thesis, we explored two particular segmentation tasks, those of interactive selection of objects in an image and second, the task of automatically segmenting object categories in an image. We started by improving upon the low level cues used in interactive segmentation, and finished with learning based methods for segmentation using bounding box detections and different superpixel classification methods. This journey is summarised in the following.

In Chapter 3, we demonstrated how texture features can be integrated into existing interactive segmentation formulations, and followed it up with a rigorous quantitative evaluation. One of the main conclusions from the quantitative evaluation was that adding texture features gives a significant performance boost when segmenting monochrome images. This improvement is useful for applications that deal with stocks of monochrome or medical images. We also demonstrated the effectiveness of texture features formulations colour images on a newly introduced dataset of images with camouflaged foreground objects.

In Chapter 4, we demonstrated the efficacy of adding mid-level cues such as

shape constraints into interactive segmentation systems. We generalised the star convexity prior idea to allow for multiple star centres while keeping the optimisation problem tractable, and also allowed for the star convexity constraint to adapt to the image data via the use of geodesics. These constraints were added to existing interactive segmentation systems, and the reduction in user effort was quantified with an evaluation system that simulates user interaction. A significant reduction in user effort was observed for interactive systems with the star convexity constraints.

In the second part of our work (Chapters 5 and 6), we explored more automated methods for image segmentation by bringing in object category level information into our models. The first contribution towards these goals was a novel learning based method to segment object categories given bounding boxes. An efficient and highly scalable learning method to predict segmentation masks from local HOG descriptors was presented. We also introduced an effective method for integrating bottom up information into these predictions using a Local GrabCut like procedure. This algorithm was trained and tested on a large dataset of segmented people that was captured automatically using the Kinect. This segmentation procedure was later fully automated using bounding boxes obtained from pre-trained person detectors, and good quantitative performance compared to other semantic segmentation methods was demonstrated.

In Chapter 6, we explored fully automated ways to segment object categories from bottom up super-pixelisations. The first contribution was an investigation into various methods for learning using multiple super-pixelisations. We found a significant performance gain in using multiple super-pixelisations over single super-pixelisations, and also observed that the surprisingly simple method of averaging classifier outputs across multiple segmentations performed almost as well as com-

plex joint learning methods. In our second contribution towards superpixel based learning, we explored adding novel pairwise terms based on histograms of visual words in a CRF based formulation. We observed that adding pairwise terms significantly improved performance over individual superpixel classification, and that the visual words based edge features performed noticeably better than simple Ising model based pairwise terms.

7.1 Future Work

7.1.1 Interactive segmentation

There are some interesting extensions to be explored for the geodesic star convexity constraint discussed in Chapter 4. It might be possible to make the star convexity constraint object specific, by adapting the star centres and their corresponding geodesics to a particular object category, and introducing additional constraints for restricting the set of shapes to be object like. This could eventually be useful for adding object category specific shape priors into both interactive and semantic segmentation. It would also be interesting to explore how to automatically learn such shape priors from training data. Another natural extension of the star convexity constraint is to generalise it to videos. As the time dimension in a video is not homogeneous with the spatial dimension, a trivial extension of the star convexity constraint by treating the time dimension as just another spatial dimension would not do very well. One alternative to treat the time dimension properly would be to extend the constraint by tracking star centres over time, enforcing connectivity along the tracks, and then treating the spatial domain in the same way as in Chapter 4.

7.1.2 Semantic segmentation

We would also like to explore several extensions for the semantic segmentation work discussed in this thesis. The first extension would be to combine the two methods introduced in Chapter 6, i.e. integrate multiple super-pixelisations with CRF based learning. The other obvious extension to the algorithms presented in Chapter 6 is allowing for multiple object categories to be segmented in the same image (the methods presented in Chapter 5 only tackle the problem of segmenting a single object category at a time). For the superpixel classifiers, this could be achieved by either combining multiple binary classifiers and choosing the most confident classification for each pixel, or by jointly learning a single classifier for all object categories. We would also like to introduce more global level cues into the superpixel classification work discussed in Chapter 6. In our current formulations, feature vectors are limited to a single superpixel, and thus local by nature. The recent Pylon segmentation model introduced by Lempitsky et al. (2011) (which is a special case of the Associative CRF model of Ladicky et al. (2009)) allows for integrating features over segmentation hierarchies, by automatically selecting the appropriate scale of the superpixels during inference. We would like to explore extending their model with multiple segmentation hierarchies and our novel CRF features. Another possibility for adding global level information into semantic segmentation is to integrate the superpixel based classifiers more tightly with the bounding box detectors that are trained on instances of the full object.

Yet another way in which we would like to bring in global level information into semantic segmentation is using image level priors. In the experiments discussed in Chapter 6, there was always an instance of the object category present in the image, and the classifiers job was only to select the right pixels. In a harder real

life application scenario (as in the VOC PASCAL challenge), often it is not known whether an instance of the object is present or not. Using image level priors in such scenarios can help provide that information, thus reducing the load of the individual superpixel classifiers.

Chapter 8

Appendix

8.1 Why $\delta_{\mathbf{y}}(\mathbf{y}_1 \cup \mathbf{y}_2)$ is not submodular?

In Section 4.1.2 of Chapter 4, we mentioned that the term $\delta_{\mathbf{y}}(\mathbf{y}_1 \cup \mathbf{y}_2)$ in (4.3) is not submodular. Here we give a more detailed explanation of the non submodularity. The term $\delta_{\mathbf{y}}(\mathbf{y}_1 \cup \mathbf{y}_2)$ implements the constraint $\mathbf{y} = \mathbf{y}_1 \cup \mathbf{y}_2$. As visualised in Figure 4.6 (Chapter 4), this constraint can be broken down into a sum of triple cliques across pixels, i.e:

$$\delta_{\mathbf{y}}(\mathbf{y}_1 \cup \mathbf{y}_2) = \sum_{i=1}^{N} \text{or}(y_i, y_i^1, y_i^2)$$

$$\text{or}(y_i, y_i^1, y_i^2) = \begin{cases} 0 & \text{if } y_i = y_i^1 \vee y_i^2 \\ \infty & \text{otherwise} \end{cases}$$

where the $\vee$ operator corresponds to the binary operator for the 'or' operation. One can verify that the function $\text{or}(y_i, y_i^1, y_i^2)$ is non-submodular using its truth

y_i	y_i^1	y_i^2	$\text{or}(y_i, y_i^1, y_i^2)$
0	0	0	0
0	0	1	∞
0	1	0	∞
0	1	1	∞
1	0	0	∞
1	0	1	0
1	1	0	0
1	1	1	0

Table 8.1: Truth table for $\text{or}(y_i, y_i^1, y_i^2)$. The projection $f(y_i^1, y_i^2) = \text{or}(y_i = 1, y_i^1, y_i^2)$ corresponds to the last four rows of the table, and is easy to see that it is non-submodular.

table shown in Table 8.1. For the function $\text{or}(y_i, y_i^1, y_i^2)$ to be submodular, it is necessary for all its projections onto two variables to be submodular. Consequently, to show non submodularity of $\text{or}(y_i, y_i^1, y_i^2)$, it is enough to construct a projection of $\text{or}(y_i = 1, y_i^1, y_i^2)$ onto two variables that is not submodular. If we look a particular slice of the function $f(y_i^1, y_i^2) = \text{or}(y_i = 1, y_i^1, y_i^2)$, we can see that: $f(0,0) + f(1,1) = \infty + 0 = \infty$ and $f(0,1) + f(1,0) = 0 + 0 = 0$. Thus $f(y_i^1, y_i^2)$ is not submodular as $f(0,0) + f(1,1) \nleq f(0,1) + f(1,0)$. The original function $\delta_{\mathbf{y}}(\mathbf{y}_1 \cup \mathbf{y}_2)$ is also not submodular, as it is a sum of non-submodular functions (and each term of the sum depends on independent variables).

8.2 Proof of O(n^n)

This section outlines how we can analytically obtain a cardinality of $O(n^n)$ for the output space of segmentations under the Euclidean star convexity constraint as mentioned in Section 4.3 of Chapter 4. Consider an image of size $n \times n$ pixels as shown in Fig. 8.1. A way of representing the star convex shape as a function $\rho(\theta)$ is shown in the figure, where the variable θ indexes the rays from the star centre and

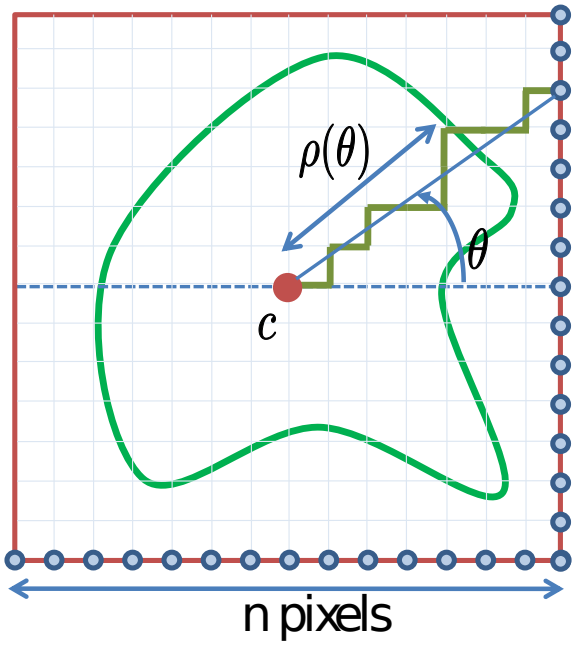


Figure 8.1: *Representing a star convex shape as a function.* The star convex shape is outlined in green, with the corresponding star centre c in red. The shape can be encoded as a function $\rho(\theta)$ as shown in figure. In a discrete image, one only needs to consider rays from the star centre to the every pixel on the boundary. Thus there are atmost $4n$ such rays. Each ray has atmost $n\sqrt{2}$ pixels.

$\rho(\theta)$ denotes how far along the ray does the object extend to. Note that while the function $\rho(\theta)$ is actually continuous, when implemented, there will only be a finite discrete number of θ's. For the case of $n \times n$ image, there are a maximum of $4n$ such θ's (one θ for each pixel on the boundary). For simplicity, we have considered the star centre to be exactly in the centre of the image. Thus $\rho(\theta) \in [0, n\sqrt{2}]$, as $n\sqrt{2}$ is the maximum length of the ray (in pixels) along the diagonal. A simple upper bound on all the possible configurations is thus given by $\left(n\sqrt{2}\right)^{4n}$ which is $O(n^n)$. Note that this might be a loose upper bound because in practice a lot of the rays intersect near the centre, and are merged into one ray.

Bibliography

Adobe Photoshop CS5. Image editing software from adobe. URL http://www.adobe.com/products/photoshop.html. 3, 14, 23

A. Agarwal and B. Triggs. Recovering 3D human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006. 104

Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 1997. 53

P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011. 39, 40, 42, 43, 44, 50, 51, 56, 123, 129

X. Bai and G. Sapiro. Geodesic matting: A framework for fast interactive image and video segmentation and matting. *International Journal of Computer Vision*, 2009. 15, 30, 85, 92, 97, 99, 100

X. Bai, J. Wang, D. Simons, and G. Sapiro. Video SnapCut: Robust video object cutout using localized classifiers. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, 2009. 27, 103, 105, 109, 111

A. Balan and M. Black. The naked truth: Estimating body shape under clothing. In *Proceedings of the European Conference on Computer Vision*, 2008. 104

Bayer Filter. http://en.wikipedia.org/wiki/Bayer_filter. URL http://en.wikipedia.org/wiki/Bayer_filter. 111

D. Bertsekas. Nonlinear programming. *Athena Scientific*, 1999. 38

S. Beucher and F. Meyer. Mathematical morphology in image processing. *Marcel Dekker*, 1992. 44

A. Blake, R. Curwen, and A. Zisserman. A framework for spatiotemporal control in the tracking of visual contours. *International Journal of Computer Vision*, 1993. 18

A. Blake, C. Rother, M. Brown, P. Perez, and P. H. S. Torr. Interactive image segmentation using an adaptive GMMRF model. In *Proceedings of the European Conference on Computer Vision*, 2004. 24

E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *Proceedings of the European Conference on Computer Vision*, 2002. 48, 49, 104, 105

E. Borenstein and S. Ullman. Learning to segment. In *Proceedings of the European Conference on Computer Vision*, 2004. 48, 104, 105

E. Boros and P. Hammer. Psuedo-boolean optimization. *Discrete Applied Mathematics*, 2002. 34

L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *Proceedings of the International Conference on Computer Vision*, 2009. 51, 104, 112, 114

L. Bourdev, S. Maji, T. Brox, and J. Malik. Detecting people using mutually consistent poselet activations. In *Proceedings of the European Conference on Computer Vision*, 2010. 51

Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *Proceedings of the International Conference on Computer Vision*, 2001. 15, 23, 25, 26, 27, 29, 30, 31, 57, 60, 76, 90, 97

Y. Boykov and V. Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *Proceedings of the International Conference on Computer Vision*, 2003. 21

Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004. 30, 31, 111

Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001. 33, 34, 53

Y. Boykov, V. Kolmogorov, D. Cremers, and A. Delong. An integral solution to surface evolution pdes via geo-cuts. In *Proceedings of the European Conference on Computer Vision*, 2006. 21

M. Bray, P. Kohli, and P. H. S. Torr. Posecut: Simultaneous segmentation and 3d pose estimation of humans using dynamic graph-cuts. In *Proceedings of the European Conference on Computer Vision*, 2006. 49

T. Brox, L. Bourdev, S. Maji, and J. Malik. Object segmentation by alignment of poselet activations to image contours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011. 51

J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 55

V. Caselles, F. Catté, T. Coll, and F. Dibos. A geometric model for active contours in image processing. *Numerische Mathematik*, 1993. 18, 20

T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 2001. 15, 18, 19, 20

D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002. 47

A. Criminisi, T. Sharp, and A. Blake. GeoS: Geodesic image segmentation. In *Proceedings of the European Conference on Computer Vision*, 2008. 15, 57, 85

N. Dalal and B. Triggs. Histogram of Oriented Gradients for Human Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005. 106

P. Das, O. Veksler, V. Zavadsky, and Y. Boykov. Semiautomatic segmentation with compact shape prior. *Image Vision Computing*, 2009. 90

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 1977. 37, 73

E. A. Dinic. Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation. *Soviet Math Doklady*, 1970. 31, 32

P. Doyle and L. Snell. Random walks and electric networks. *Washington, D.C.: Mathematical Association of America*, 1984. 27

O. Duchenne, J.-Y. Audibert, R. Keriven, J. Ponce, and F. Segonne. Segmentation by transduction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 15, 29, 30, 92

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results, 2007. 50

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results, 2009. 76, 95

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results, 2010. 8, 48, 67, 104, 112, 114, 116, 120, 123, 133, 134

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 2008. 107, 115, 132

P. Felzenszwalb and O. Veksler. Tiered scene labeling with dynamic programming. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 34, 35

P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 2004. 40, 42, 44, 45

P. F. Felzenszwalb, R. B. Grishick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009. 50, 51, 105, 114, 120

L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, 1962. 31

B. Frey. *Graphical Models for Machine Learning and Digital Communication*. MIT Press, 1998. 24

M. Galun, E. Sharon, R. Basri, and A. Brandt. Texture segmentation by multiscale aggregation of filter responses and shape elements. In *Proceedings of the International Conference on Computer Vision*, 2003. 59

P. V. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *Proceedings of the International Conference on Computer Vision*, 2009. 126, 128, 133, 138

A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM*, 1988. 31

GrabCut Dataset. URL `http://research.microsoft.com/en-us/um/cambridge/projects/visionimagevideoediting/segmentation/grabcut.htm`. 62, 76, 92, 95

L. Grady. Multilabel random walker image segmentation using prior models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005. 15, 28, 57, 101

L. Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006. 15, 27, 28, 29, 92, 97, 99

D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1989. 30

G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007. URL http://authors.library.caltech.edu/7694. 62

M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 1998. 18

H. Ishikawa. Exact optimization for markov random fields with convex priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2003. 34

H. Ishikawa. Higher-order clique reduction in binary graph cut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 36

I. H. Jermyn and H. Ishikawa. Globally optimal regions and boundaries as minimum ratio weight cycles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001. 19

T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 2009. 143

M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *Proceedings of the International Conference on Computer Vision*, 1987. 15, 16, 17, 30

G. Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 1996. 18

P. Kohli and P. H. S. Torr. Dynamic graph cuts for efficient inference in markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007. 38

P. Kohli, J. Rihan, M. Bray, and P. Torr. Simultaneous segmentation and pose estimation of humans using dynamic graph cuts. *International Journal of Computer Vision*, 2008. 77

P. Kohli, M. P. Kumar, and P. H. S. Torr. P3 & beyond: Move making algorithms for solving higher order functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009a. 35

P. Kohli, L. Ladický, and P. H. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 2009b. 35, 36

V. Kolmogorov and C. Rother. Minimizing nonsubmodular functions with graph cuts-a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007. 34

V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions via graph cuts. In *Proceedings of the International Conference on Computer Vision*, 2001. 33

V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004. 32

V. Kolmogorov, Y. Boykov, and C. Rother. Applications of parametric maxflow in computer vision. *Proceedings of the International Conference on Computer Vision*, 2007. 38

N. Komodakis and N. Paragios. Beyond pairwise energies: Efficient optimization for higher-order mrfs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 36

M. P. Kumar. *Combinatorial and Convex Optimization for Probabilistic Models in Computer Vision*. PhD thesis, Oxford Brookes University, 2008. 50

M. P. Kumar, P. H. S. Torr, and A. Zisserman. Learning layered pictorial structures from video. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, 2004. 49

M. P. Kumar, P. H. S. Torr, and A. Zisserman. OBJ CUT. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005. 27, 49, 77

S. Kumar and M. Hebert. Discriminative random fields. *International Journal of Computer Vision*, 2006. 24, 27, 71

V. Kwatra, A. Schodl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, 2003. 46

L. Ladicky, C. Russell, P. Kohli, and P. H. S. Torr. Associative hierarchical crfs for object class image segmentation. In *Proceedings of the International Conference on Computer Vision*, 2009. 33, 55, 120, 121, 122, 139, 152

B. Leibe and B. Schiele. Interleaved object categorization and segmentation. In *Proceedings of the British Machine Vision Conference*, 2003. 48

V. Lempitsky, A. Blake, and C. Rother. Image segmentation by branch-and-mincut. In *Proceedings of the European Conference on Computer Vision*, 2008. 37, 38, 88

V. Lempitsky, C. Rother, S. Roth, and A. Blake. Fusion moves for markov random field optimization. Technical report, 2009. URL http://research.microsoft.com/apps/pubs/default.aspx?id=80790. 34

V. Lempitsky, A. Vedaldi, and A. Zisserman. A pylon model for semantic segmentation. In *Advances in Neural Information Processing Systems*, 2011. 152

T. Leung and J. Malik. Recognizing surfaces using three-dimensional textons. In *Proceedings of the International Conference on Computer Vision*, 1999. 59

A. Levin and Y. Weiss. Learning to combine bottom-up and top-down segmentation. In *Proceedings of the European Conference on Computer Vision*, 2006. 49, 104, 105

A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 2004. 15

A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008. 109

F. Li, J. Carreira, and C. Sminchisescu. Object recognition as ranking holistic figure-ground hypotheses. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 55, 122

Y. Li, J. Sun, C. K. Tang, and H. Y. Shum. Lazy snapping. *ACM Transactions on Graphics*, 2004. 15, 27

Y. Li, J. Sun, and H. Shum. Video object cut and paste. *ACM Transactions on Graphics*, 2005. 27

J. Liu, J. Sun, and H.-Y. Shum. Paint selection. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, 2009. 27, 30, 78, 97

D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004. 124, 130

J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the International Conference on Machine Learning*, 2009. 107, 108

J. Malik and P. Perona. A computational model of texture segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1989. 59

J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 2001. 59

T. Malisiewicz and A. A. Efros. Improving spatial support for objects via multiple segmentations. In *Proceedings of the British Machine Vision Conference*, 2007. 126

D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the International Conference on Computer Vision*, 2001. 39

D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004. 41, 43, 44

Microsoft Powerpoint 2010. Presentation software from Microsoft. URL http://office.microsoft.com/en-us/powerpoint/. 3, 14, 23

E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, 1995. 15, 21, 22

Mosek: Optimization Toolkit. http://www.mosek.com, 2010. 133, 145

D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Comm. Pure Appl. Math.*, 1989. 15, 18

H. Nickisch, P. Kohli, and C. Rother. Learning an interactive segmentation system. Technical report, 2009. URL http://www.citebase.org/abstract?id=oai:arXiv.org:0912.2492. 92

S. Nowozin and C. H. Lampert. Global connectivity potentials for random field models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 78

T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002. 59, 61

A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic object recognition with boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006. 48, 123, 125, 133

OpenNI. Open source libraries for interfacing with Natural Interaction devices. URL http://www.openni.org/. 111

M. T. Orchard and C. A. Bouman. Color Quantization of Images. *IEEE Transactions on Signal Processing*, 1991. 73

J. O'Rourke. *Art gallery theorems and algorithms*. Oxford University Press, Inc., New York, NY, USA, 1987. 82

C. Pantofaru, C. Schmid, and M. Hebert. Object recognition by integrating multiple image segmentations. In *Proceedings of the European Conference on Computer Vision*, 2008. 54, 55, 122, 126, 127, 128

P. Perez, A. Blake, and M. Gangnet. Jetstream: Probabilistic contour extraction with particles. In *International Conference on Computer Vision*, 2001. 18

Picasa. Fast and easy photo sharing from google. URL http://picasa.google.com/. 3, 5

C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott. A perceptually motivated online benchmark for image matting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 95

ROS OpenNI. Open source project focused on the integration of the primesense sensors with ROS. URL http://www.ros.org/wiki/openni_kinect. 111

C. Rother, V. Kolmogorov, and A. Blake. GrabCut: interactive foreground extraction using iterated graph cuts. *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, 2004. 14, 15, 24, 26, 30, 37, 38, 57, 60, 90, 104, 109, 111

C. Rother, P. Kohli, W. Feng, and J. Jia. Minimizing sparse higher order energy functions of discrete variables. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 36

D. Rubin. Iteratively reweighted least squares. *Encyclopedia of Statistical Sciences*, 1983. 71

B. C. Russell, A. A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006. 126

F. Schroff, A. Criminisi, and A. Zisserman. Single-histogram class models for image segmentation. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, 2006. 59

F. Schroff, A. Criminisi, and A. Zisserman. Object class segmentation using random forests. In *Proceedings of the British Machine Vision Conference*, 2008. 59

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the International Conference on Machine Learning*, 2007. 107

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000. 42, 43, 44

J. Shotton, J. Winn, C. Rother, and A. Criminisi. *TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Proceedings of the European Conference on Computer Vision*, 2006. 33, 50, 51, 52, 53, 59, 122, 139

J. Shotton, M. Johnson, and R. Cipolla. Semantic Texton Forests for Image Categorization and Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 53, 59, 61, 122

A. K. Sinop and L. Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *Proceedings of the International Conference on Computer Vision*, 2007. 15, 29, 101

G. Slabaugh and G. Unal. Graph Cuts Segmentation Using an Elliptical Shape Prior. In *Proceedings of the IEEE International Conference on Image Processing*, 2005. 80

C. Smith. A characterization of star-shaped sets. *American Mathematical Monthly*, 1968. 78

M. Szummer, P. Kohli, and D. Hoeim. Learning CRFs using graph cuts. In *Proceedings of the European Conference on Computer Vision*, 2008. 88, 139, 143

F. Toranzos and A. Cunto. Sets expressible as finite unions of starshaped sets. *Journal of Geometry*, 2004. 81

A. Torralba, K. Murphy, and W. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007. 52

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the International Conference on Machine Learning*, 2004. 139, 142, 143

F. Valentine. *Convex sets*. McGraw-Hill, 1964. 78, 87

M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003. 57, 60, 61

A. Vedaldi and B. Fulkerson. Vlfeat - an open and portable library of computer vision algorithms. In *Proceedings of the 18th annual ACM International Conference on Multimedia*, 2010. 131

A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In *Proceedings of the European Conference on Computer Vision*, 2008. 40, 47, 123, 129

A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 125, 131

A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Proceedings of the International Conference on Computer Vision*, 2009. 126

O. Veksler. PhD thesis, Cornell University, 1999. 34

O. Veksler. Star shape prior for graph-cut image segmentation. In *Proceedings of the European Conference on Computer Vision*, 2008. 15, 75, 76, 78, 79, 80, 90, 91, 102

O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and supervoxels in an energy optimization framework. In *Proceedings of the European Conference on Computer Vision*, Berlin, Heidelberg, 2010. Springer-Verlag. 40, 42, 45, 123, 129

S. Vicente, V. Kolmogorov, and C. Rother. Graph cut based image segmentation with connectivity priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 78, 101, 102

S. Vicente, V. Kolmogorov, and C. Rother. Joint optimization of segmentation and appearance models. In *Proceedings of the International Conference on Computer Vision*, 2009. 38

J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. *ACM Trans. Graph.*, 2005. 27

J. Winn and J. Shotton. The Layout Consistent Random Field for Recognizing and Segmenting Partially Occluded Objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006. 27, 33, 54, 139

O. J. Woodford, P. H. S. Torr, I. D. Reid, and A. W. Fitzgibbon. Global stereo reconstruction under second order smoothness priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 34

Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1993. 42

Xbox Kinect. Full body game controller from Microsoft. URL `http://www.xbox.com/kinect`. 9, 11, 105, 111

Y. Yang, S. Hallman, D. Ramanan, and C. Fowlkes. Layered object detection for multi-class segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 50, 51

J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: a comprehensive study. *International Journal of Computer Vision*, 2007. 59

THE END