Interactive image segmentation

Nguyen, Thi Nhat Anh

2012

Nguyen, T. N. A. (2012). Interactive image segmentation. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/48651>

<https://doi.org/10.32657/10356/48651>

INTERACTIVE IMAGE SEGMENTATION

NGUYEN THI NHAT ANH

School of Computer Engineering
Nanyang Technological University

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Master of Engineering

2012

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Associate Professor Cai Jianfei, for his persistent support, guidance and invaluable advices in my research and preparation of this report. His patience, motivation, enthusiasm, and immense knowledge have been of great value for me. I also would like to thank Assistant Professor Zheng Jianmin, for his invaluable guidance, advices, and comments to my research. Last but not the least, I would like to thank my fellow graduate students in the Center for Multimedia and Network Technology for the helpful discussions and collaboration.

Contents

1	Introduction	9
1.1	Background	9
1.2	Motivation and Objectives	10
1.3	Thesis Organization	13
2	Literature Review	14
2.1	Single-image Segmentation	14
2.1.1	Region-based Image Segmentation	14
2.1.2	Active Contours	17
2.1.2.1	Various Active Contour Models	17
2.1.2.2	Convex Active Contour Model	18
2.1.2.3	Split Bregman Solver	19
2.2	Multi-view image segmentation	21
3	Robust Interactive Image Segmentation	24
3.1	System Overview	24
3.2	Contour Initialization	25
3.3	Constrained Active Contour Model	27
3.3.1	Regional term formulation	27
3.3.2	Boundary term formulation	29
3.4	Experimental Results	30
3.4.1	Test on the benchmark data set	31
3.4.2	Visual results	32

4	Interactive Multi-view Image Segmentation	36
4.1	System Overview	37
4.2	Pre-segmentation and Initialization	39
4.3	3D Graph-cut Using Color and Silhouette Cues	41
4.3.1	Smoothness Term	42
4.3.2	Data Term	42
4.3.3	Gaussian Mixture Model Based Color Cue	44
4.3.4	Hard Constraint Enforcement	46
4.4	Silhouette Refinement Via 2D Segmentation	49
4.5	Local Editing and Refinement	51
4.5.1	Automatic selection of images for editing	51
4.5.2	Local editing and refinement	53
4.6	Experiments	54
4.6.1	Parameters	54
4.6.2	Segmentation Results	56
4.6.3	Limitations	59
5	Conclusions and Future Work	60
5.1	Conclusions	60
5.2	Future Work	61

List of Figures

1.1	The segmentation results of different algorithms, including Random Walks [18], Geodesic [1] and GrabCut [28], and our proposed method.	11
3.1	The segmentation results of our methods initialized by either Geodesic [1] or Random Walks [18].	26
3.2	Comparisons of the results using the two different g_b definitions in (2.2) and (3.5), respectively.	30
3.3	The segmentation results of different algorithms, including Random Walks [18], Geodesic [1] and GrabCut [28], and our proposed method, for three different images.	33
3.4	The segmentation results with different user inputs.	34
3.5	An example to show the ability of our method on handling topology changes. The images in the second row highlight erroneous parts for the segmentation results above them (red color indicates false-positive parts and blue color indicates false-negative parts).	35
4.1	The diagram of the proposed multi-view object segmentation system.	38
4.2	Illustration of hard-constraint enforcement for pre-segmented object pixels.	47

4.3	The constrained active contour model refines the initial silhouette contour and brings the contour closer and snapped to the actual object boundary in silhouette refinement step. The result is represented by the silhouette cue (third column), which is subsequently supplied to the next iteration of the 3D graph-cut.	51
4.4	Pseudocode for automatic selection of images for editing.	52
4.5	Local editing at an erroneous image.	54
4.6	Segmentation results of the “lion” sequence. The first row shows some of the original images. The second row is the results with only color cue. The fourth row is the results with both color and silhouette cues. The sixth row shows the final results after further local editing, where a few user strokes are added to image 10. The images in the third, fifth and seventh rows highlight erroneous parts for the segmentation results above them (red color indicates false-positive parts and blue color indicates false-negative parts).	55
4.7	Segmentation results of the “teapot” video sequence, which is casually captured by a hand-held camera. For the uniformly sampled 56 image frames, images 1, 16, 31 and 46 are pre-segmented in the first step. The first row shows some of the original images, and the second row shows the corresponding results of our system.	58
4.8	The 3D model reconstructed from the silhouettes of the “teapot” video sequence.	58

List of Tables

3.1	Error rate comparison using the MSRC dataset with exactly the provided trimaps.	31
4.1	Error rates of the segmentation results of different multi-view sequences.	59

Abstract

The state-of-the-art single-image interactive segmentation algorithms are sensitive to the user inputs and often not able to produce accurate cutting contour with one-shot user input. They frequently rely on laborious user editing to refine the segmentation boundary. In the first part of this thesis, we propose a robust and accurate interactive image segmentation method based on the recently developed continuous-domain convex active contour model. The proposed method exhibits many desired properties for a good interactive image segmentation tool, including the fast segmentation speed, the robustness to user inputs and different initializations and the ability to produce a smooth and accurate boundary contour. Experimental results on a benchmark data set show that the proposed tool is highly effective and significantly outperforms the state-of-the-art interactive image segmentation algorithms.

In the second part of this thesis, we extend our study to interactive segmentation of multi-view images, i.e., segmenting object regions from a sequence of calibrated images, which are taken on the same object from different viewing angles. In the case of a large number of images in the sequence, segmenting each image separately using interactive image segmentation techniques is time-consuming and requires a lot of user interaction and effort. The proposed method combines 2D interactive image segmentation and 3D object segmentation to exploit both the consistency constraint among different images in the sequence as well as the local information in each individual images to segment the sequence with a small amount of user interaction. Furthermore, we also introduce an editing tool for easily and arbitrarily

refining the segmentation result. Experimental results show that the proposed method produces good segmentation result even with challenging image sequences and the editing tool is effective in refining the segmentation result.

Chapter 1

Introduction

1.1 Background

Interactive image segmentation, which incorporates little user interaction to define the desired content to be extracted, has received much attention in the recent years. Many interactive image segmentation algorithms have been proposed in the literature. In general, interactive image segmentation algorithms can be classified into two categories: boundary-based approaches and region-based approaches.

In boundary-based approaches, the user is often asked to specify an initial area that is close to the desired boundary. The active contours/Snake method [21] evolves an initial contour to the desired boundary. The methods of intelligent scissors [26, 14] apply Dijkstra’s shortest path algorithm to find the cutting contour between neighboring boundary points that are specified by the user.

Considering that the boundary-based approaches require great care to specify the boundary area or the boundary points, especially for complex shapes, most recent interactive image segmentation algorithms take the regional information as the input. In particular, in region-based approaches, the user is often asked to draw two types of strokes to label some pixels as foreground or background pixels, after which the algorithm completes the labelling for other pixels. The state-of-the-art region-based interactive segmentation algorithms include the graph cut based methods [4, 28], the random walks based methods [18, 17], the geodesic methods [1,

12], etc. Most of these methods basically treat an image as a weighted graph with nodes corresponding to pixels in the image and edges being placed between neighboring pixels, and minimize a certain energy function on this graph to produce a segmentation.

1.2 Motivation and Objectives

In this thesis, we start with the problem of interactive segmentation of a single image with the input of foreground and background strokes, which requires least attention from the user. By carefully examining the state-of-the-art region-based approaches, we find that their performance is limited in terms of robustness and accuracy. First, the state-of-the-art region-based methods are sensitive to different user inputs (see Figure 1.1). As pointed out in [31], the graph cut algorithm is sensitive to the number of seeds, while the random walks and geodesic algorithms are sensitive to locations of seeds. This is mainly due to the different behaviors of the different energy functions. For example, graph cut tries to minimize the total edge weights along the cut. Thus, it may return very small segmentations (known as the “small cut” problem) in the case with small number of seeds provided. Random walks minimizes a Dirichlet energy and different boundary conditions (locations of seeds) always result in different harmonic functions.

Second, the cutting contours generated by the region-based approaches, especially by random walks and geodesic methods, are usually jaggy and do not snap to geometry features (see Figure 1.1). Additional refinement step is often needed to improve the segmentation performance of the existing region-based methods. Most of the state-of-the-art interactive image segmentation methods [28, 18, 1, 36] rely on additional user inputs to either globally or locally refine the cutting contour. However, when dealing with complex images, the user is often required to provide a lot of additional strokes or boundary points and thus struggles with labo-

Figure 1.1: The segmentation results of different algorithms, including Random Walks [18], Geodesic [1] and GrabCut [28], and our proposed method. The images in the second and fourth rows highlight erroneous parts for the segmentation results above them (red color indicates false-positive parts and blue color indicates false-negative parts). Note that GrabCut is an advanced version of graph cut. The three state-of-the-art algorithms are sensitive to different user inputs while our method is quite robust. In addition, the existing methods produce jaggy boundary contours while our constrained active contour method is able to smooth out the contours and make them snap to geometric edges without additional user input.

rious refinement/editing. Another way for boundary refinement is to use the active contours/Snakes model [21] to refine the initial boundary contour produced by a region-based segmentation approach as in [8]. However, the refinement based on Snakes is only able to change the contour locally for smoothness but incapable of evolving the entire contour to snap to geometry features/edges and incapable of handling topology changes of the evolving contour.

The above observations motivate us to design a new method for interactive image segmentation. The mathematical tool at the heart of the new method is the continuous-domain convex active contour model [5], which makes use of both the boundary and the regional information to find a global “optimal” solution. Continuous-domain convex methods have started to receive attention since they avoid the inherent grid bias in all discrete graph-based methods and also have fast and global numerical solvers through convex optimization [5]. However, the convex active contour model so far has mainly been applied for automatic image segmentation, which often results in over-segmentation with trivial solutions for complex images [5, 9]. On the other hand, it is not clear how to apply the convex active contour model [5] for interactive image segmentation. Directly incorporating user inputs as hard constraints into the model does not lead to better performance. The objective of this research is to develop a new interactive image segmentation method based on the convex active contour model [5], that possesses three desired properties including the fast segmentation speed, the robustness to user inputs and different initializations, and the ability to produce a smooth and accurate segmentation result reflecting user intention and geometric features.

In addition to improving the single-image segmentation through using convex active contour, the other objective of this thesis is to extend the developed interactive single-image segmentation tool to multi-view image segmentation. Particularly, we consider the problem of segmenting a sequence of calibrated images, which are taken on the same object from different viewing angles. The acquirement of the ob-

ject’s silhouette in each view of a multi-view image sequence is very useful for many applications such as multi-view stereo reconstruction which produces 3D models from multi-view sequences. One possible approach to acquire the silhouettes of an object is to segment each image separately using the interactive image segmentation technique for a single image. However, in the case of a large number of images in the sequence, this approach will require a large amount of user interaction to segment the whole sequence, and the silhouettes obtained in each view might not meet the coherency constraint [19]. Another way is to perform the segmentation in 3D instead of 2D such as in [6] and [32], which guarantees the silhouette coherence at the cost of processing complexity. Our intention here is to combine both 2D and 3D segmentation techniques to develop an accurate and coherent multi-view cutting while still keeping the processing speed acceptable.

1.3 Thesis Organization

The remaining of the thesis is organized as follows. Chapter 2 reviews some of the basic concepts and pioneer work that either inspire or contribute to our work including several state-of-the-art region-based segmentation methods, various active contour models and how the Split Bregman method is used to solve convex active contour model, and the existing multi-view image segmentation work. Chapter 3 presents our proposed robust interactive image segmentation method and the experimental results to demonstrate the effectiveness of our proposed method. Chapter 4 presents our proposed interactive multi-view image segmentation method and the experimental results. Finally, Chapter 5 concludes the thesis and discusses some future research directions.

Chapter 2

Literature Review

In this chapter, we review some of the basic concepts and pioneer work that either inspire or contribute to our work. The first section reviews some of the techniques for single-image segmentation including the state-of-the-art region-based image segmentation methods and various active contour models. A more detail introduction for the convex active contour model [5], which our constrained active contour model is based on, and the usage of the Split Bregman method to solve convex active contour model are also presented in this section. The second section reviews some existing multi-view image segmentation work.

2.1 Single-image Segmentation

2.1.1 Region-based Image Segmentation

In region-based image segmentation approaches, the user is often asked to draw two types of strokes to label some pixels as foreground or background pixels, after which the algorithm completes the labeling for other pixels. Those pre-labelled pixels are also called foreground and background seeds. In this section, we give a brief review on several related state-of-the-art region-based image segmentation approaches.

One popular region-based image segmentation approach is the graph cut based methods introduced in [4], [25] and [28]. The graph cut based methods basically treat an image as a weighted graph with nodes corresponding to pixels in the image.

Besides, two virtual nodes are added to the graph: a “foreground” terminal and a “background” terminal. There are two types of undirected edges connecting nodes in the graph: neighborhood links connecting each pair of neighboring pixels and terminal links connecting each pixel to each of the terminals. The Graph Cut algorithm [4] models foreground and background pixel values according to the histograms, which are sampled from the foreground and background seeds. The fitness of each pixel value to the foreground or background model is reflected on the strength of the terminal links connecting this pixel to the foreground and background terminals. The max-flow/min-cut algorithm is employed to optimally classify each node in the graph as foreground or background. In the LazySnapping work [25], in addition to the foreground and background strokes, more intuitive user interface techniques are added to facilitate easier interactive image segmentation. It is the boundary polygon editing tool. The GrabCut method [28] extends the graph cut framework to segment color images, where the foreground and background colors are modeled by Gaussian mixture models. In addition, GrabCut method implements an iterative version of the graph cut optimisation to reduce the amount of user interaction needed to complete a segmentation task. GrabCut method also integrates a border matting algorithm. Besides, various types of user inputs are introduced in GrabCut, including a bounding box to enclose the foreground object, a lasso input for difficult images, foreground and background strokes for local editing, and a boundary brush for matting. The GrabCut method can achieve good performance in segmenting the images whose foreground and background colors are well separable, but its performance is often unsatisfactory for the images whose foreground and background share similar color distributions. One common problem for graph cut based methods is the “small cut” behavior. As graph cut tries to minimize the total edge weight along the cut, it may return very small segmentations in the case with small number of seeds provided and in the present of weak boundary.

Another popular interactive image segmentation approach is the random walks

algorithm [18, 17]. Similarly, the random walks algorithm requires the input of foreground and background seeds. This method also models an image as a graph. It calculates the probability that a random walker walking along the graph’s edges starting at each unlabeled pixel will first reach one of the pre-labeled pixels. The algorithm subsequently assigns each pixel to the label with the greatest probability. The random walks algorithm is more robust to a weak boundary and overcomes the “small cut” problem of graph cut. However, the random walks algorithm is essentially an approach that minimizes a Dirichlet energy with boundary conditions, where different boundary conditions (different input seeds) always result in different harmonic functions, therefore it is very sensitive to the positions and quantities of foreground and background seeds.

Recently, Bai et al. [1] proposed a geodesic framework based on the computation of weighted geodesic distances from each pixel to the foreground or background strokes for interactive image and video segmentation. The GeoS algorithm [12] extends the geodesic framework on improving the processing speed and relaxing the connectivity requirement, i.e., each segmented region needs to be connected to the corresponding user stroke. The advantage of the geodesic algorithms is that they are significantly faster than the graph cut or random walks algorithms. While the geodesic algorithms do not suffer the “small cut” problem like the graph cut method, their performance also depend on sufficient separation in the foreground and background color distributions. The major limitation of the geodesics algorithms is that they are very sensitive to the seed locations since different seed locations result in different geodesic distances for each pixels.

For the remaining of this thesis, we will refer to the original geodesic framework by Bai et al. in [1] as the Geodesic method and the original random walks method for image segmentation by Grady et al. in [18] as the Random Walks method.

2.1.2 Active Contours

In this section, we briefly review various active contour models and their application in interactive image segmentation. The convex active contour model recently introduced in [5], which our constrained active contour model is based on, is discussed in more details. Furthermore, we also summarize the usage of the Split Bregman method to solve convex active contour model in a fast manner as discussed in [16].

2.1.2.1 Various Active Contour Models

One of the first active contour models was proposed in [21] called Snakes. In [8] and [20], the authors proposed to use Snake model to refine the initial boundary contour produced by a region-based segmentation. However, as aforementioned, the refinement based on Snakes is only able to change the contour locally for smoothness but incapable of evolving the entire contour to snap to geometry features/edges and incapable of handling topology changes of the evolving contour.

Geodesic active contours (GAC) was introduced in [7] as an enhanced version of the Snake model. The GAC model allows topology changes and stable boundary detection when the gradients suffer from large variations, including gaps. However, GAC only use the boundary information therefore does not perform well on complex images with a lot of noisy edges or weak object boundary. Another well-known active contour model called Active contours without edges (ACWE) was proposed by Chan and Vese [10]. It seeks to approximate an image by a piecewise smooth function. This model focuses on the regional information of an image and therefore, removes the dependency on boundary information, but it is often trapped in local minimum due to its non-convex modelling. Therefore, in [5] and earlier in [9], the authors proposed several convex active contour models, which are to find the global minimum solution. Also in [5], the authors proposed to integrate the GAC model and the ACWE model to effectively exploit both boundary and regional information.

2.1.2.2 Convex Active Contour Model

The convex active contour model introduced in [5] can be generally expressed as

$$\min_{0 \leq u \leq 1} \left(\int_{\Omega} g_b |\nabla u| dx + \lambda \int_{\Omega} h_r u dx \right), \quad (2.1)$$

with the following symbol definitions:

- u is a function on image domain Ω , which receives a value between 0 and 1 at each pixel location x in the image. The segmented region is obtained by thresholding the function u .
- Function g_b is a boundary function, which is often an edge detection function such as

$$g_b(x) = \frac{1}{1 + |\nabla I(x)|^2}, \quad (2.2)$$

where $I(x)$ is the intensity of image pixel x .

- Function h_r is a region function that measures the inside and outside regions. Particularly, $h_r = h_r^{in} - h_r^{out}$, where h_r^{in} and h_r^{out} are the inside and outside region functions, respectively. They are often defined as

$$h_r^{in}(C_{in}, x) = (\mu_{in} - I(x))^2, \quad \mu_{in} = \frac{\int_{C_{in}} I(x) dx}{\int_{C_{in}} dx}$$

$$\text{and } h_r^{out}(C_{out}, x) = (\mu_{out} - I(x))^2, \quad \mu_{out} = \frac{\int_{C_{out}} I(x) dx}{\int_{C_{out}} dx},$$

where μ_{in} and μ_{out} are the mean intensities for inside and outside regions, C_{in} and C_{out} , respectively.

Basically, Eq. (2.1) consists of two terms balanced by a tradeoff factor λ , where the first term is a boundary term and the second term is a region term. The boundary term favors the segmentation along the curves that the edge detection function reaches minimum and also favors the segmentation with smooth boundary

curve. The second term ensures the segmentation complying with some region coherence criteria defined in function h_r .

Once the optimization problem of (2.1) is solved, the segmented region is found by thresholding the function u , i.e.,

$$C_{in} = \{x|u(x) > T\}; \quad C_{out} = \Omega \setminus C_{in}, \quad (2.3)$$

where typically $T = 0.5$.

The automatic segmentation problem based on the convex active contour model in (2.1) is usually solved by an alternate iterative approach depicted as follows.

1. Fix the segmentation, i.e., C_{in} and C_{out} , and update h_r .
2. Fix h_r to find the solution u for (2.1).
3. Update C_{in} and C_{out} according to (2.3).

The above three steps are repeated until convergence. It can be seen that the computation bottleneck of this iterative approach lies in step 2, i.e., solving the optimization problem of (2.1).

2.1.2.3 Split Bregman Solver

Several methods have been proposed to solve (2.1) for a given h_r . Chan *et al.* [9] proposed to either enforce the inequality constraint of $(0 \leq u \leq 1)$ using an exact penalty function, which is non-differentiable, or regularize the penalty function, which does not exactly enforce the inequality constraint. Bresson *et al.* [5] used a splitting/regularization approach to minimize (2.1). Their method “smears” the values of u near the object boundaries, and thus makes the segmentation results more dependent on the cutoff parameter T , which could eliminate the segmentation details.

Recently, Goldstein *et al.* [16] proposed to use the Split Bregman method to solve (2.1). The Split Bregman method is not only able to solve the convex active

CHAPTER 2. LITERATURE REVIEW

contour model but also a much more efficient solver. In the following, we briefly summarize the use of this Split Bregman solver as introduced in [16].

In particular, instead of solving (2.1) directly, Bregman introduced a new vectorial function d into the model as

$$\min_{u \in [0,1]} \int_{\Omega} g_b |d| + \lambda h_r u dx, \quad (2.4)$$

with the constraint of $d = \nabla u$. This constraint is enforced using the efficient Bregman iteration approach defined as

$$(u^{k+1}, d^{k+1}) = \arg \min_{u \in [0,1], d} \int_{\Omega} g_b |d| + \lambda h_r u + \frac{\mu}{2} |d - \nabla u - b^k|^2 dx \quad (2.5)$$

$$b^{k+1} = b^k + \nabla u^{k+1} - d^{k+1}, \quad (2.6)$$

where $k \geq 0$ is the iteration index. The computations of (2.5) and (2.6) are repeated until convergence.

Since (2.5) is differentiable, it can be solved using a simple alternating method. Specifically, the function is first differentiated with respect to u using Euler-Lagrange Differential Equation, which results in the following optimality condition for u :

$$\mu \Delta u = \lambda h_r + \mu \operatorname{div}(d^k - b^k), \quad u \in [0, 1], \quad (2.7)$$

where Δu is the Laplacian of u and $\operatorname{div}(d^k - b^k)$ is the divergence of $(d^k - b^k)$. Based on (2.7), u^{k+1} can be approximately obtained by a Gauss-Seidel iterative method [16]. After that, (2.5) is solved with respect to d . It has been shown in [34] that the minimizing solution d^{k+1} is given by soft-thresholding:

$$d^{k+1} = \frac{\nabla u^{k+1} + b^k}{|\nabla u^{k+1} + b^k|} \max(|\nabla u^{k+1} + b^k| - \mu^{-1} g_b, 0). \quad (2.8)$$

2.2 Multi-view image segmentation

In this section, we briefly review some existing multi-view image segmentation work. As mentioned previously, one possible approach to acquire the object silhouettes for a multi-view image sequence is to segment each image independently using the interactive image segmentation technique for a single image. However, considering that a multi-view image sequence often consists of $40 \sim 80$ images or more, this approach would require too much user effort to interactively segment each individual multi-view image [30].

Another way is to apply the recently developed image co-segmentation technique to jointly segment all the multi-view images. The image co-segmentation problem was first introduced in [29], which deals with automatically segmenting a similar foreground object from two images with unrelated backgrounds. Although the latest co-segmentation algorithms [27, 11] can achieve impressive segmentation results automatically, it is not designed for the considered multi-view object segmentation and is not able to achieve highly accurate segmentation.

One common problem with the previously mentioned methods is that the silhouettes obtained in each view might not meet the coherency constraint [19], which suggests that the correctly segmented silhouettes on the images that are formed from the projections of the same rigid 3D object must be the corresponding silhouettes. Several work has been proposed to incorporate this coherency constraint to improve the segmentation performance and reduce the demands placed on the user. In [33], Sormann et al. proposed a graph cut based multiple view segmentation method, where each image is firstly partitioned into a certain number of regions by the meanshift segmentation algorithm. The graph cut is carried out within these pre-segmented regions of the image. For each image, a shape prior is added to the graph cut's energy function to incorporate the coherency between the image's foreground object and that of the preceeding segmented image in the sequence. However, this scheme does not guarantee that the final segmented object regions

in the images satisfy the silhouette coherency constraint, which means that the coherency constraint within the image sequence is not fully utilized. In [23], Lee et al. proposed an automatic multi-view image segmentation scheme, which also employs a graph cut based segmentation in each individual image. The incorporation of the silhouette coherency constraint is realized by integrating the silhouette calibration ratio [3] into the data term of graph cut. The silhouette calibration ratio for each pixel calculates the probability of the pixel to be foreground from the silhouettes of the other views. It represents how much the other views agree that the pixel belongs to the foreground. The silhouette calibration ratios are updated through an iterative scheme. Also through this iterative scheme, the foreground and background color models are learned in a similar way as in the GrabCut method [28]. This method however also does not guarantee that the final segmented object regions satisfy the silhouette coherency constraint as the images are still segmented individually.

In order to enforce this silhouette coherency constraint, several work has been proposed to perform the segmentation in 3D instead of 2D [32, 6]. Most of those works employ an energy minimization scheme through 3D graph cut to label each 3D location (voxel) as object or background. In particular, the 3D graph-cut based approach in [32] minimizes an energy function to regularise the process of combining a series of imperfect silhouettes to estimate the voxel occupancy of a calibrated volume in space. These silhouettes are obtained by background subtraction against the known backgrounds. The energy function to be minimized contains a data term and a smoothness term. The data term is a sum over the individual voxels, where the penalty for a voxel is based on the observed intensities of the pixels that intersect it. The smoothness term is the number of empty voxels adjacent to filled ones. The limitation of this method is that the images are needed to be taken in a fixed, calibrated camera rig where the background of each image is known. In [6], Campbell et al. introduced an automatic 3D object segmentation scheme,

which also uses 3D graph cut to perform binary segmentation in 3D space. The object's silhouette in each image is found by projecting the resulting 3D object on the image. This method also targets to minimize an energy function containing data term and smoothness term. However, unlike [32], this method requires no prior knowledge of the object or environment. Instead it relies on a fixation constraint to initialize the solution. The fixation constraint assumes that the object of interest is always being focused upon by the camera and thus likely to be central to the viewpoint. Similar to the GrabCut method [28], this method employs an iterative approach to the segmentation task by learning colour models of the object and the background. In particular, a 3D graph is built with edge weights derived from the corresponding projections and foreground/background color models, which are initially sampled from the image pixels around the fixation points. Then, 3D graph-cut is performed to generate the 3D segmentation result. From the projection of the 3D segmentation result, an improved color model is updated, which is being used for the next round 3D segmentation. The whole process repeats until convergence. The limitation of this method is that as it employs an automatic scheme, which is mainly based on color information and is not supplied with additional guide from the user to distinguish between object and background, it only performs well in case the foreground and background colors are simple and well separated. In addition, this 3D object segmentation scheme fails to exploit a great amount of local information on each individual image such as the image's geometry features or the individual image's foreground and background color models. In addition, the essential fixation assumption that the scheme is based on is not always true. Another problem with 3D segmentation methods is that their processing speeds are relatively slow compared to the 2D segmentation methods.

Chapter 3

Robust Interactive Image Segmentation

In this chapter, we present the detail of our first work, a robust interactive single-image segmentation method based on the continuous-domain convex active contour model [5], which was reviewed in the previous chapter. The desired properties for the proposed method include fast segmentation speed, robustness to user inputs and different initializations, and the ability to produce a smooth and accurate boundary contour.

3.1 System Overview

The mathematical tool at the heart of our method is the continuous-domain convex active contour model [5], which makes use of both the boundary and the regional information to find a global “optimal” solution. Continuous-domain convex methods have started to receive attention since they avoid the inherent grid bias in all discrete graph-based methods and also have fast and global numerical solvers through convex optimization [5]. However, the convex active contour model so far has mainly been applied for automatic image segmentation, which often results in over-segmentation with trivial solutions for complex images [5, 9].

In this work, we propose to marry the powerful continuous-domain convex active contour with one of the state-of-the-art region-based methods, either Geodesic or

Random Walks (Geodesic is chosen due to its fast processing speed), where the region-based method is used in the first step to generate an initial contour and the convex active contour is then applied in the second step to optimize the contour. Note that here we not only use the region-based method to generate an initial contour, but also incorporate the information obtained in the pre-segmentation into the convex active contour model, which is non-trivial. Such an integration ensures that the contour evolving does not drift too far away from the initial contour, complies with the user input and reflects the user intention, while keeping the “strong” ability of contour evolving provided by the convex active contour model to absorb the non-robustness of region-based approaches and snap the contour to geometry features.

In addition, considering that the convergence speed for solving the convex active contour model is generally slow, we make use of the Split Bregman method, discussed in Section 2.1.2.3, to solve the proposed constrained convex active contour model at a faster speed.

3.2 Contour Initialization

For any active contour method, the contour needs to be initialized before the contour evolution process. Here, we use the segmentation result of the Geodesic method [1] for contour initialization due to its fast processing speed and the ability to avoid the “small-cut” problem.

In particular, we represent the result of the Geodesic algorithm by a probability map $P(x)$, whose value is within the range of $[0,1]$ indicating the probability that pixel x belongs to the foreground region. In the Geodesic algorithm, for a pixel x , its geodesic distances to the foreground or background seed regions are computed, which are denoted as $D_F(x)$ and $D_B(x)$ respectively. Then, the probability that

the pixel x belongs to the foreground is calculated as

$$P(x) = \frac{D_B(x)}{D_F(x) + D_B(x)}.$$

Once the probability map is available, we initialize the contour evolution by assigning $P(x)$ to the function $u(x)$ in (2.1).

It is worth mentioning that other region-based image segmentation algorithms such as Random Walks [18] can also be used for contour initialization. For example, for Random Walks algorithm, the resulting probability for a random walker to reach Foreground or Background label first from each pixel is used in the place of the resulting geodesic distance from Geodesic algorithm to calculate the probability map $P(x)$. The second column of Figure 3.1 shows the probability maps for both the Geodesic and Random Walks methods.

Figure 3.1: The segmentation results of our methods initialized by either Geodesic [1] or Random Walks [18]. First row: our method (initialized by Geodesic); second row: our method initialized by Random Walks.

3.3 Constrained Active Contour Model

As shown in (2.1), the convex active contour model consists of two terms: a regional term and a boundary term. Next, we discuss how to modify these two terms to incorporate the information from the user input and the initial segmentation result so as to ensure the refined contour complying with the user input.

3.3.1 Regional term formulation

The foreground and background seeds give an excellent description about the color distributions of the foreground and background regions. Foreground/background GMMs introduced in [35] are estimated from foreground/background seeds and are used to represent the color distributions of the foreground and background regions. Specifically, let $Pr(x|F)$ and $Pr(x|B)$ denote the probabilities that pixel x fits the foreground and background GMMs, respectively. The normalized likelihood that x belongs to foreground and background respectively are

$$P_F(x) = \frac{-\log Pr(x|F)}{-\log Pr(x|F) - \log Pr(x|B)} \text{ and } P_B(x) = \frac{-\log Pr(x|B)}{-\log Pr(x|F) - \log Pr(x|B)} . \quad (3.1)$$

We incorporate this regional information derived from foreground/background strokes into the regional term of the convex active contour model as

$$h_r(x) = P_B(x) - P_F(x) . \quad (3.2)$$

This definition of h_r ensures that the active contour evolves towards the one complying with the known GMM models. For instance, for a pixel x , if $P_B(x) > P_F(x)$ (resp. $P_B(x) < P_F(x)$) and $P_B(x) - P_F(x)$ is positive (resp. negative), $u(x)$ tends to decrease (resp. increase) during the contour evolution in order to minimize (2.1), which can lead to $u(x) \leq T$ (resp. $u(x) > T$) and the classification of the pixel belonging to the background (resp. the foreground).

The h_r definition of (3.2) fails in the case that the foreground and background color models are not well separated. Thus, to avoid this problem and also to exploit the segmentation result obtained by the Geodesic algorithm in step 1, we further propose to incorporate the probability map $P(x)$ into the region term h_r as

$$h_r(x) = \alpha(P_B(x) - P_F(x)) + (1 - \alpha)(1 - 2P(x)), \quad (3.3)$$

where $\alpha \in [0, 1]$ is a tradeoff factor. The second term $(1 - 2P(x))$ in (3.3) prevents the refined contour drifting too far apart from the initial segmentation. In particular, when $P(x) > 0.5$ and $(1 - 2P(x))$ is negative, $u(x)$ tends to increase in order to minimize (2.1), which favors classifying the pixel as a foreground pixel, and vice versa.

It is important to properly set the tradeoff factor α in (3.3). When the foreground and background colors are well separable, it is desired that the first term in (3.3) becomes dominating; otherwise, the second term in (3.3) should dominate. Thus, similar to the one suggested in [36], we set α to be the distance between the foreground and the background GMMs. We use the Monte-Carlo simulations to approximate the Kullback-Leibler divergence between the foreground and the background GMMs, i.e.,

$$\alpha = \frac{1}{n} \sum_{i=1}^n \left| \frac{\log Pr(x_i|F) - \log Pr(x_i|B)}{\log Pr(x_i|F) + \log Pr(x_i|B)} \right|. \quad (3.4)$$

In addition, it can be observed that when $h_r(x) \rightarrow +\infty$ (resp. $h_r(x) \rightarrow -\infty$), the regional term forces $u(x) = 0$ (resp. $u(x) = 1$) in order to minimize Eq. (2.1). This observation allows us to enforce some hard constraints in the contour evolution process. In particular, for those pixels that have no ambiguity in classification, including the pixels lying on the foreground/background strokes and the pixels having very large or very small $P(x)$ values ($P(x) > 0.9$ or $P(x) < 0.1$), we treat them as hard constraints in the contour evolution process. We directly assign a

negative h_r value and a positive h_r value, both with extremely large magnitude, to these confirmed foreground and background pixels, respectively. In this way, we guarantee that the refined result complying with the user input and also exploit more information from the initial segmentation result.

Note that unlike the h_r definition in Section 2.1.2.2, our proposed h_r model is fixed given the user input and the initial segmentation. Thus, there is no need for the three-step alternate iteration described in Section 2.1.2.2. Instead, only step 2 is needed, which can be solved by the Split Bregman method discussed in Section 2.1.2.3.

3.3.2 Boundary term formulation

The boundary term of $\int_{\Omega} g_b(x) |\nabla u| dx$ in (2.1) is essentially a weighed total variation of function u , where the weight g_b plays an important role. The definition of g_b in (2.2) is effective in the sense that it encourages the segmentation along the curves where the edge detection function is minimal. The problem with (2.2) is that at the locations with weak edges the boundary is likely to be smoothed out. Thus, in this research, we propose to incorporate the GMM probability map $P_F(x)$ defined in (3.1) to enhance the edge detection. Particularly, we define g_b as

$$g_b = \beta \cdot g_c + (1 - \beta) \cdot g_e, \quad (3.5)$$

where g_c and g_e are the results of applying the edge detection to the GMM probability map $P_F(x)$ and the original image, respectively, and $\beta \in [0, 1]$ is a tradeoff factor computed in a similar way as α given in (3.4). Note that the edge detection function returns values between 0 to 1 and a small value of g_b corresponds to a likely edge.

Figure 3.2 compares the results with and without incorporating the edge detection of the GMM probability map. It can be seen that incorporating g_c enhances

Figure 3.2: Comparisons of the results using the two different g_b definitions in (2.2) and (3.5), respectively. Note that some boundary problems due to using (2.2) are marked in (c).

the conventional edge detection result g_e , especially at the weak edges, which leads to a more accurate boundary contour.

3.4 Experimental Results

In this section, we verify the performance of our proposed method. We set the parameter λ in (2.1) to 1000, μ in (2.5) to 10000. The settings of other parameters have been discussed in the previous sections. All the parameters are being set in the same way for all the test images.

Table 3.1: Error rate comparison using the MSRC dataset with exactly the provided trimaps.

Method	Error rate
GMMRF [2]	7.9 (reported in [2])
Geodesic [1]	5.21 (our implementation)
Random Walks [18]	5.4 (reported in [13])
GrabCut [28]	5.66 (reported in [36])
Yang <i>et al.</i> [36]	4.08 (reported in [36])
Our method	3.768
Our method with Random Walks	3.77
Our method with Yang <i>et al.</i>	3.765

3.4.1 Test on the benchmark data set

The commonly used MSRC ground truth data set [28] is chosen for testing and comparison. The MSRC data set contains 50 test images, each of which are provided with trimaps and ground truth. Table 3.1 summarizes the achieved error rates (percentage of mislabelled pixels) by different state-of-the-art interactive image segmentation algorithms and our proposed method. We also test several variant versions of our method, where we replace the Geodesic method by the Random Walks or the Yang’s method [36]. For fair comparison, we use exactly the same trimaps provided by the MSRC data set as the user inputs for all the algorithms. The error rates for other state-of-the-art algorithms are either directly quoted from the best results reported in literature or obtained through our implementation.

Note that the MSRC data set is somewhat biased because the provided trimaps only contain small unknown regions, for which Geodesic and Random Walks perform well. We still choose it since it is the only publicly available data set with trimaps provided. In addition, we also test the performance on many images with large unknown regions, as illustrated in the visual results (see Section 3.4.2).

From Table 3.1, it can be seen that our proposed method achieves very low error rate, significantly outperforming the state-of-the-art interactive image segmentation algorithms, including Geodesic [1], Random Walks [18], GrabCut [28] and Yang’s approach [36]. In addition, we can see that our method is insensitive to the initial

contour since the initializations using different methods lead to almost the same error rate. This property can also be observed in the illustration in Figure 3.1, Section 3.2, where the initial contours produced by the Geodesic and Random Walks methods are totally different but final segmentation results in both cases are very similar. Moreover, the speed of convergence of our constrained active contour is very fast, taking less than half a second to optimize the segmentation result of an image with a resolution of 640×480 .

3.4.2 Visual results

Figure 3.3 shows the segmentation results of different algorithms for three different images. It can be seen that in these cases with large unknown regions, Geodesic and Random Walks perform poorly, producing inaccurate and jaggy boundary contours. Although the performance of GrabCut is much better, its results still contain some clearly visible artifacts, e.g., around the neck of the man, the right elbow of the boy and the horse head and leg regions. On the contrary, our method produces accurate and smooth contours and makes them snap to geometry edges.

As aforementioned, the Random Walks and Geodesic algorithms are sensitive to the seed locations. Figure 3.4 compares the segmentation results with different user inputs. It can be seen that, for Random Walks and Geodesic, different users inputs result in different segmentations. In contrast, our constrained active contour is able to fix the problem and generate stable results insensitive to the user input, as shown in Figure 3.4. Note that although the GrabCut algorithm is insensitive to the seed locations, it has the “small cut” problem as illustrated in Figure 1.1.

An important property of our constrained active contour model is the ability to handle topology changes of the boundary contour, which is not achievable by the classical Snakes model. As shown in Figure 3.5, while the Random Walks method produces two initial boundary contours that separate the object into two halves, our constrained active contour can evolve the boundary contours to one

(a) Input

(b) Random Walks

(c) Geodesic

(d) GrabCut

(e) Our method

(f) Input

(g) Random Walks

(h) Geodesic

(i) GrabCut

(j) Our method

(k) Input

(l) Random Walks

(m) Geodesic

(n) GrabCut

(o) Our method

Figure 3.3: The segmentation results of different algorithms, including Random Walks [18], Geodesic [1] and GrabCut [28], and our proposed method, for three different images.

Figure 3.4: The segmentation results with different user inputs.

outer contour around the object and one inner contour at the object's left hand. Similarly, the geodesic method produces only one initial closed boundary contour while our method further produces the additional inner contour.

Figure 3.5: An example to show the ability of our method on handling topology changes. The images in the second row highlight erroneous parts for the segmentation results above them (red color indicates false-positive parts and blue color indicates false-negative parts).

Chapter 4

Interactive Multi-view Image Segmentation

As discussed in Section 2.2, for a multi-view image sequence that contains a large number of images, segmenting each image separately using an interactive single-image segmentation technique will place a significant task on the user to segment the whole sequence. In order to reduce the demands placed on the user, several work has been proposed to incorporate coherency constraint that exists within the image sequence into segmentation process by performing segmentation in 3D instead of 2D [6, 32]. In [6], Campbell et al. introduced an automatic 3D object segmentation scheme, which uses 3D graph cut to performs binary segmentation in 3D space. The object’s silhouette in each image is found by projecting the resulting 3D object on the image. Although this 3D domain approach [6] ensures the silhouette coherence well, it has some limitations. First, it only makes use of the color prior in the data term of the 3D graph-cut, which can produce good results in the cases that the foreground and background color distributions are simple and well separated. However, for many real-world examples (e.g. Figure 4.6), the object is often captured in cluttered or camouflaged environment, for which the color model itself is insufficient to distinguish the foreground from the background. Second, it assumes that the object of interest is located at the center of each image, which is not always true.

Considering that each individual multi-view image in fact contains a great

amount of information, not just color, regarding the foreground and the background, in this paper we propose to integrate the 3D segmentation method with the existing 2D segmentation technique so as to combine their advantages, i.e. extracting more local prior information through 2D segmentation while exploiting the silhouette coherence through 3D segmentation. In particular, we first use interactive 2D image segmentation to segment a few images at high accuracy so as to introduce high-level prior into the system as hard constraints. Since only a few images need to be interactively segmented, the required user effort is of small amount. We then extend the iterative 3D volumetric graph-cut developed in [6] in two aspects: incorporating not only the color cue but also the silhouette cue, and enforcing the hard constraints extracted from the initial user interaction. The silhouette cue is generated and iteratively updated using an innovative 2D convex active contour model. In addition, a local editing and refinement step is also introduced to allow the user to edit the most erroneous image and quickly update the segmentation results of the entire multi-view sequence.

4.1 System Overview

The primary inputs of our system are a set of M multi-view color images, $\{I_m | m = 1, \dots, M\}$, and a set of associated projection matrices obtained through camera calibration. A projection matrix allows the center of any voxel $v_n \in R^3$ to be mapped to its corresponding location $x_{m,n} \in R^2$ in image I_m . The output of our system is the segmentation result for all the images, which can be represented by the object silhouettes or the foreground/background binary images.

Figure 4.1 shows the diagram of the proposed multi-view object segmentation system. It can be seen that the entire system consists of five steps: pre-segmentation by interactive 2D cut (on a small subset of images), initialization by visual hull, silhouette refinement by 2D cut, 3D graph-cut and local editing and refinement.

Figure 4.1: The diagram of the proposed multi-view object segmentation system.

The first step is to segment a small set of images I_H (normally 3 to 4 images) sampled from the multi-view image sequence to obtain precise silhouettes using any of the existing interactive image segmentation methods such as Grabcut [28] or Geodesic [1], where the user typically needs to draw two types of colored strokes on an image to label some pixels as foreground and background seeds to guide the segmentation process. In our implementation, we use our proposed robust interactive image segmentation method discussed in the previous chapter to segment these images. The silhouettes of these images are used as hard constraints to guide the segmentation of the remaining images at the later steps.

The second step is to generate an initial silhouette for each of the remaining images, $I_m \notin I_H$. In particular, a visual hull containing the object is first generated from the pre-segmented silhouettes using the shape-from-silhouette method in [22]. The visual hull is then projected on each of the remaining images to generate an initial silhouette. The details of the first two steps can be found in Section 4.2.

Considering the initial silhouette obtained through 3D projection is not accurate, in the third step we extend a recently developed convex active contour model [5] to evolve the initial silhouette in each remaining image to snap to its nearby geometry features. The details of the developed convex active contour method will be discussed in Section 4.4.

In the fourth step, to ensure the segmentation coherence across different views, similar to [6], a 3D object segmentation via 3D graph-cut is performed. However, unlike [6], the 3D graph-cut developed in this research utilizes not only a color cue but also a silhouette cue as well as hard constraints obtained from the initial

user interaction. The details of the fourth step is described in Section 4.3. We would like to point out that step 3 and step 4 are tightly coupled and they together form an iteration process. In each iteration, the refined silhouette via the convex active contour provides better color and silhouette cues for the 3D graph-cut and in return the graph-cut produces better initial silhouette for the next-round silhouette refinement.

Finally, considering that for some complex multi-view image sequences there might still be some errors in the silhouettes after the iterative 2D and 3D segmentation, we introduce the last step to allow the user to easily and arbitrarily refine the segmentation result of any of the images locally with extra foreground or background strokes. After the local refinement, the silhouettes for the entire sequence will be updated accordingly. The proposed local editing scheme ensures a fast editing and updating speed. A smart image selecting scheme is also developed to automatically suggest the most erroneous image in the sequence to the user for local editing. The details of this last step can be found in Section 4.5.

4.2 Pre-segmentation and Initialization

The first step of our proposed method is the pre-segmentation step, where a small set of images I_H (normally 3 to 4 images) sampled from the multi-view image sequence is segmented to obtain precise silhouettes of these images using an interactive image segmentation method such as Grabcut [28] or Geodesic [1], where the user typically needs to draw two types of colored strokes on an image to label some pixels as foreground and background seeds to guide the segmentation process. In our implementation, we use our proposed robust interactive image segmentation method discussed in the previous chapter to segment these images. The silhouettes of the pre-segmented images are used as hard constraints to guide the segmentation of the remaining images at the later steps. At least three pre-segmented images are needed

to reconstruct the initial visual hull in step 2. We normally need to pre-segment from three to four images. However, the more images are pre-segmented, the more accurate the segmentation result will be.

The second step is to generate an initial silhouette for each of the remaining images, $I_m \notin I_H$. In particular, the initial visual hull containing the object is first generated from the pre-segmented silhouette images and their associated projection matrices using the shape-from-silhouette method introduced by Aldo Laurentini [22].

Back-projection of each silhouette image into 3D space using its associated projection matrix forms a generalized cone called a silhouette cone. Intersection of the set of silhouette cones forms the visual hull of an object. The geometrical meaning of the visual hull of an object is that it defines the upper bound of the approximation of 3D representation of the object such that substituting the 3D object with its visual hull does not affect any silhouette.

The accuracy of the visual hull reconstruction depends on both 3D sampling resolution of the reconstruction algorithm and the number of input silhouette images. A high sampling resolution together with a large number of silhouette images can achieve a good approximation of the real object. The pre-segmented images in the first step are chosen in such a way that they can capture the object from as many varying angles as possible so that the intersection of their silhouette cones forms a good visual hull approximating the real object.

The initial visual hull is then projected on each remaining image $I_m \notin I_H$ to generate the initial silhouette for that image. The actual object silhouette is entirely contained within the projected initial silhouette on each image.

4.3 3D Graph-cut Using Color and Silhouette Cues

Similar to the automatic 3D segmentation in multiple views introduced in [6], a volumetric graph-cut algorithm is employed in our system to segment the object in 3D space. In particular, a 3D array of voxels V is formed within the bounding box of the object’s visual hull. We create a 3D undirected graph with nodes corresponding to voxels, $v_n \in V$, and two additional nodes: an “object” terminal (a source S) and a “background” terminal (a sink T). There are two types of undirected edges in the graph: n-links (neighborhood links) connecting each pair of neighboring voxels in a six-connected sense, $\{v_i, v_j\} \in E$, and t-links, $\{v_i, S\}$ and $\{v_i, T\}$, connecting each voxel v_i to each of the two terminals. An min-cut algorithm is employed to label each voxel as inside the object visual hull ($O \subset V$) or outside ($B \subset V$), optimally dividing the graph into 2 separate parts: the Object and the Background.

Let $w(v_i, v_j)$ denote the weight of the link $\{v_i, v_j\}$, representing how strong the link is or how similar the connected two nodes are. It also represents the cost for disconnecting the link during graph cut and contributes to the energy function of graph cut that we try to minimise later.

Following [6], our 3D graph-cut operation is formulated as an energy minimization process:

$$O = \arg \min_{O \subset V} E_d(O, \Theta) + E_s(O), \quad (4.1)$$

where E_d is the data term measuring the energy cost of a segmentation O against the prior model Θ , E_s is the smoothness term measuring the energy cost of disconnecting the boundary links. Unlike [6], which only uses the color cue for Θ , we incorporate both color and silhouette cues.

4.3.1 Smoothness Term

As in [6], the smoothness term measures the energy cost associated with the surface area of the object, i.e. the summation of the weights of all the boundary links:

$$E_s = \sum_{\{v_i, v_j\} \in E, v_i \in O, v_j \in B} w(v_i, v_j), \quad (4.2)$$

where E the set of n-links, and O and B are the object and the background voxel sets, respectively.

The color discontinuities within the images project cutting planes through the voxel array which forms the boundaries of the visual hull. The n-link weight $w(v_i, v_j)$ is defined as the maximum pixel color difference of the projections of v_i and v_j over all the views:

$$w(v_i, v_j) = \lambda_1 \max_m e^{-\beta_m \|I_m(x_{m,i}) - I_m(x_{m,j})\|^2}, \quad (4.3)$$

where I_m is the m -th image, $x_{m,i}$ and $x_{m,j}$ are the projected pixels of v_i and v_j on image m , respectively, β_m is a parameter estimated from image m as in [28], and λ_1 is the trade-off factor between the smoothness term and the data term.

4.3.2 Data Term

The data term provides the preference for a voxel to be classified as object or background according to the prior model. It can be expressed as

$$E_d = y_n w(v_n, T) + (1 - y_n) w(v_n, S), \quad (4.4)$$

where y_n is the label of v_n and

$$\begin{cases} y_n = 1, & v_n \in O \\ y_n = 0, & v_n \in B, \end{cases}$$

and $w(v_n, T)$ and $w(v_n, S)$ are the weights of the t-links $\{v_i, T\}$ and $\{v_i, S\}$, respectively.

Considering that the actual object lies entirely within the initial visual hull, the set of voxels that is outside the initial visual hull, which is denoted as B_H , is confirmed to be background voxels. To enforce this hard-constraint for the 3D graph-cut, we set

$$\forall v_n \in B_H, \quad w(v_n, S) = 0, \quad w(v_n, T) = A, \quad (4.5)$$

where A is a constant whose value is large enough to force any $v_n \in B_H$ to be a background voxel (see Section 4.6.1 for details). For other voxels, their t-link weights are computed according to the color and silhouette cues.

In particular, $\forall v_n \notin B_H$, their t-link weights are defined as

$$\begin{aligned} w(v_n, S) &= 1 + \left[\left\{ \frac{1}{M} \sum_{m=1}^M L_m(x_{m,n}, \Theta) \right\} - \phi \right] \\ w(v_n, T) &= 1 - \left[\left\{ \frac{1}{M} \sum_{m=1}^M L_m(x_{m,n}, \Theta) \right\} - \phi \right], \end{aligned} \quad (4.6)$$

where $L_m(x_{m,n}, \Theta) \in [0, 1]$ is the likelihood that the projected pixel $x_{m,n}$ of v_n on image I_m belongs to foreground. In (4.6), we combine the probabilities from the individual images with a threshold parameter $\phi \in [0, 1]$, which encodes the level of robustness to noise of the algorithm in the same way as that in [6]. In this way, a voxel v_n is more likely to be classified as inside the object if $w(v_n, S) > w(v_n, T)$, which happens when the averaging probabilities across all the images $\left\{ \frac{1}{M} \sum_{m=1}^M L_m(x_{m,n}, \Theta) \right\}$ is larger than ϕ . Considering the special case of perfect object segmentation in all the images, i.e. the case of ideal binary classification where $L_m(x_{m,n}, \Theta)$ is either 0 or 1, a voxel would only be classified as an object voxel when all the silhouettes agree that the projection of that voxel is always a foreground pixel, i.e. $L_m(x_{m,n}, \Theta) = 1$ for all the images. Therefore, in this case,

$\phi \rightarrow 1$ since $\left\{ \frac{1}{M} \sum_{m=1}^M L_m(x_{m,n}, \Theta) \right\} \rightarrow 1$. In our algorithm, ϕ is set to a value between 0.8-0.9 to tolerate the imperfect classification in each image.

Now we show the key issue of how to calculate the probability $L_m(x_{m,n}, \Theta)$ for each individual image. Specifically, for an image I_m that does not belong to the set I_H of the pre-segmented images, the probability is computed as

$$L_m(x_{m,n}, \Theta) = \alpha_1 P_m(x_{m,n}) + (1 - \alpha_1) u_m(x_{m,n}), \quad (4.7)$$

where $P_m(x_{m,n})$ is the color cue that assesses the fitness of projected pixel $x_{m,n}$ of voxel v_n on image I_m to the foreground/background color models, which will be introduced in Section 4.3.3, $u_m(x_{m,n})$ is the silhouette cue representing the probability of pixel $x_{m,n}$ belonging to the foreground based on the 2D cut, which will be discussed in Section 4.4, and α_1 (empirically set as =0.5) is the tradeoff factor between the color cue and the silhouette cue.

For an image I_m that belongs to the set I_H of the pre-segmented images, if $x_{m,n}$ is a background pixel, then $v_n \in B_H$, the set of voxels that are confirmed to be outside of the object, for which the t-link weights are set as (4.5). Otherwise, if $x_{m,n}$ is a foreground pixel (labelling function $y_{m,n} = 1$), theoretically, $L_m(x_{m,n}, \Theta)$ should be equal to 1. However, we will show in Section 4.3.4 that $\forall I_m \in I_H, \forall y_{m,n} = 1$, $L_m(x_{m,n}, \Theta)$ needs to be carefully set in order to meet the hard constraints.

4.3.3 Gaussian Mixture Model Based Color Cue

The color cue assesses the fitness of a pixel's color to the foreground/background color models. The color models represent the color distributions of the foreground and background regions. Similar to [6], we use Gaussian Mixture Model (GMM) for color modelling. Specifically, at each iteration, foreground and background GMMs are learnt from foreground and background seeds, which are derived from the current silhouette at each image.

We first discuss how to generate the foreground and background seeds in each image. For the images that belong to the set of the pre-segmented images I_H , the foreground and background seeds are the entire segmented foreground and background regions. For the images that do not belong to I_H , it can be observed that pixels that are far away from the object boundary are more likely to be correctly classified. Thus, we obtain the foreground and background seed regions in $I_m \notin I_H$ by shrinking the foreground and the background regions by a safe distance D . Considering that the voxels outside the initial visual hull are confirmed background voxels, the background seed region is further updated as the union of the shrunked background region and the initial background region resulted from the initial visual hull in the first step of our framework.

We next discuss how to build the foreground and background color models. The foreground seed region in each image may not contain all the color of the object since one image only captures one view of the object. Thus, to build up a global color model for the object, we use the pixels from the generated foreground seeds of all the views. On the other hand, as the background varies over different views, a separate background GMM color model is learnt for each view. Considering that the background seed region at one view may not contain all the color information of the background at that particular view due to occlusion, we also sample the nearby views, whose camera directions are less than 45 degree away from the particular view.

Let $Pr_m(x|F)$ and $Pr_m(x|B)$ denote the probabilities that pixel x in image I_m fits the foreground and background GMM color models, respectively. The color cue $P_m(x)$ representing the normalized likelihood that pixel x is a foreground pixel is calculated as

$$P_m(x) = \frac{Pr_m(x|F)}{Pr_m(x|F) + Pr_m(x|B)}. \quad (4.8)$$

4.3.4 Hard Constraint Enforcement

In the first step of our framework, with sufficient user interaction, it is reasonable to deem that for the set I_H of the pre-segmented images we obtain perfect silhouettes, which are used as hard constraints for the subsequent segmentation. However, the 3D graph-cut does not guarantee that the projection of the segmented 3D object on I_H matches the hard constraints of the pre-generated silhouettes. Specifically, let $y_{m,n}$ and $\hat{y}_{m,n}$ respectively denote the segmentation results obtained in the first step of the pre-segmentation and the fourth step of the 3D graph-cut for pixel $x_{m,n}$ with $I_m \in I_H$ and $y_{m,n} = 1$. The issue here is how to set $L_m(x_{m,n}, \Theta)$ so as to ensure $\hat{y}_{m,n} = y_{m,n} = 1$.

In particular, $\forall I_m \in I_H, \forall x_{m,n}, y_{m,n} = 1$ implies that at least one of the 3D voxels lying in the optical line of pixel $x_{m,n}$ must be an object voxel. Let v_{n_0} denote such an object voxel for pixel $x_{m,n}$. Considering that the t-link weights for v_{n_0} is calculated according to (4.6), simply setting $L_m(x_{m,n}, \Theta)=1$ does not guarantee that the average likelihood over all the M images is big enough to make v_{n_0} being classified as an object voxel in the 3D graph-cut. On the other hand, setting $L_m(x_{m,n}, \Theta)$ to a value large enough to guarantee $\hat{y}_{m,n} = y_{m,n}$ might have the problem of making background voxels lying on the optical line of $x_{m,n}$ being classified as object voxels.

Figure 4.2 illustrates this dilemma on the “sculp face” image sequence. This sequence consists of 14 images but only a few images of this sequence are shown in the figure. In this figure, images numbered 1 and 5 are the pre-segmented images. The second row in this figure shows the projected segmentation results of a few images in the sequence when $L_m(x_{m,n}, \Theta)$ is fixed to 1, with $x_{m,n}$ is a foreground pixel on a pre-segmented image $I_m \in I_H$ as aforementioned. It can be observed that the resulted foreground regions are significantly smaller than the actual ones in all of the images, including the pre-segmented images, which means that the hard-constraint on those pre-segmented images are not guaranteed.

Figure 4.2: Illustration of hard-constraint enforcement for pre-segmented object pixels. The segmentation results for a few images in “sculp face” sequence using different settings for the likelihood $L_m(x_{m,n}, \Theta)$ of a pre-segmented foreground pixel $x_{m,n}$. In second and fourth rows, $L_m(x_{m,n}, \Theta)$ is fixed to 1 and 1.5, respectively. In sixth row, $L_m(x_{m,n}, \Theta)$ is calculated using the proposed iterative scheme. The images in the third, fifth and seventh rows highlight erroneous parts for the segmentation results above them (red color indicates false-positive parts and blue color indicates false-negative parts).

On the other hand, the fourth row of Figure 4.2 shows the projected segmentation results when $L_m(x_{m,n}, \Theta)$ is fixed to a larger value, which is 1.5. This assignment makes the object regions in the pre-segmented images become larger and nearly overlap with the actual object regions. However, this also makes background voxels lying on the optical line of $x_{m,n}$ being classified as object voxels, which causes overshooting in other non-pre-segmented images, $I_i \notin I_H$ and results in bigger foreground regions than the actual ones in those images.

Thus, instead of assigning a fixed constant value to the foreground likelihood $L_m(x_{m,n}, \Theta)$, $\forall I_m \in I_H$, $\forall x_{m,n}$ with $y_{m,n} = 1$, we propose to adaptively find a value for $L_m(x_{m,n}, \Theta)$ through iterations. In particular, for a fixed set of color and silhouette cues which are generated by the third step: Silhouette refinement by 2D cut, several iterations of 3D graph-cut are executed, where the value of $L_m(x_{m,n}, \Theta)$ is updated after each iteration until the resulting silhouette is exactly the same as the pre-segmented silhouette for each image $I_m \in I_H$. $\forall I_m \in I_H$, $\forall x_{m,n}$ with $y_{m,n} = 1$, we initialize $L_m(x_{m,n}, \Theta) = 1$. After each iteration of the 3D graph-cut, if $\hat{y}_{m,n} \neq y_{m,n}$, we increase $L_m(x_{m,n}, \Theta)$ by a small constant value ϵ and then repeat the 3D graph-cut. The iteration stops when $\hat{y}_{m,n} = y_{m,n}$, $\forall I_m \in I_H$, $\forall x_{m,n}$ with $y_{m,n} = 1$. Note that the iterations of 3D graph-cut as described above forms an inner loop within the loop consisting of 2 steps: Silhouette refinement by 2D cut and 3D graph-cut illustrated in Figure 4.1. These inner iterations of 3D graph-cut runs quite fast since there is no need to update the color and the silhouette cues after the first iteration. The constant ϵ , which is the increment of $L_m(x_{m,n}, \Theta)$ in each iteration, is set to $0.3 * M$ in all of our experiments, where M is the number of images in the sequence.

The sixth row in Figure 4.2 shows that when $L_m(x_{m,n}, \Theta)$ is calculated using the proposed iterative scheme, the resulting silhouettes of the segmented object are good. We also observe that the projected silhouettes on images numbered 1 and 5 comply with the hard-constraint of the pre-segmentation.

4.4 Silhouette Refinement Via 2D Segmentation

The step of silhouette refinement serves for two purposes: one is to refine the initial contour in each image produced by either the visual hull or the 3D graph-cut, and the other is to generate better color and silhouette cues for the subsequent 3D graph-cut. In this research, we propose to apply our proposed constrained active contour model which was described in Chapter 3 for the silhouette refinement on each image $I_m \notin I_H$. The reason we adopt the convex active contour model as a 2D segmentation method for the silhouette refinement lies in its strong capability to evolve the initial contour to snap to the geometry features/edges in an image and its fast processing speed due to convex optimization.

As mentioned in Section 3.2, we first need to generate an initial probability map $P(x)$ to initialize the active contour in each image $I_m \notin I_H$. However, unlike the interactive image segmentation framework described in Chapter 3, there are no user input strokes on those images $I_m \notin I_H$ in order to use existing interactive image segmentation methods like Geodesic or Random Walks to generate the probability map $P(x)$. Therefore, we propose to generate the initial probability map $P(x)$ from the current silhouette as follows.

Let $S(x)$ denote the current segmentation result at a particular image, where a value of 0 or 1 indicates a background or foreground pixel at location x , respectively. Considering the fact that the areas far away from the boundary are likely to be correctly classified, we create an initial probability map $P(x)$ over the image domain to represent the probability that a pixel belongs to foreground region as follows.

$$P(x) = \begin{cases} 1, & \text{if } d(x) \geq D \text{ and } S(x) = 1 \\ 0.5 + 0.5(d(x)/D)^2, & \text{if } d(x) < D \text{ and } S(x) = 1 \\ 0.5 - 0.5(d(x)/D)^2, & \text{if } d(x) < D \text{ and } S(x) = 0 \\ 0, & \text{if } d(x) \geq D \text{ and } S(x) = 0, \end{cases}$$

where $d(x)$ denotes the Euclidean distance from pixel x to its nearest boundary

point, and D , a threshold value on $d(x)$, is empirically set to $\frac{1}{10}\sqrt{R_{obj}}$ with R_{obj} representing the size of the current foreground region. An example of the initial probability map $P(x)$ is shown in Figure 4.3(b).

In this application of the constrained active contour model for 2D silhouette refinement, instead of sampling the foreground and background color models from the foreground and background strokes as in Chapter 3, we build up these color models from the foreground and background pixels according to the current silhouette. In particular, the sets of pixels with $y(x) = 1$ and $y(x) = 0$ are treated as foreground and background seeds, respectively. Note that the GMM models in Section 4.3.3 are different from the models here, where the previous GMM models are trained across different views and for the purpose of the 3D segmentation while the GMM models here are the local models obtained from each individual image and for the purpose of the 2D segmentation.

After the initial probability map $P(x)$ and the foreground and background color models are generated, we solve the following convex active contour model as in Chapter 3.

$$\min_{0 \leq u \leq 1} \left(\int_{\Omega} g_b |\nabla u| dx + \lambda \int_{\Omega} h_r u dx \right),$$

with h_r and g_b calculated according to formula (3.3) and (3.5), respectively.

As mentioned previously, our constrained active contour model has the strong capability to evolve the contour entirely to snap to the geometry features/edges of the image while complies with the foreground/background color model and the initial silhouette. Once the optimization of the convex active contour model (2.1) is solved, we obtain the solution of $u(x)$, which represents the probability of a pixel x belonging to the foreground and is used as a silhouette cue in (4.7) for the next iteration of the 3D graph-cut.

Figure 4.3 illustrates how the constrained active contour model evolves the initial silhouette contour and brings the contour closer and snapped to the actual object boundary in silhouette refinement step. Figure 4.3(d) shows the resulting silhouette

cue.

Figure 4.3: The constrained active contour model refines the initial silhouette contour and brings the contour closer and snapped to the actual object boundary in silhouette refinement step. The result is represented by the silhouette cue (third column), which is subsequently supplied to the next iteration of the 3D graph-cut.

4.5 Local Editing and Refinement

Considering that for some complex multi-view image sequences, there might still be some errors in the silhouettes after the iterative 2D and 3D segmentation, we introduce this local editing and refinement step to allow the user to provide more strokes to edit the erroneous silhouettes in one or more images, which will then be used to automatically refine the entire sequence.

4.5.1 Automatic selection of images for editing

In the case that the sequence consists of a large number of images, it will be troublesome and time-consuming for the user to browse through the whole image sequence to select an image for editing. In this section, we introduce an automatic selecting scheme to suggest the most erroneously segmented images for user editing. In particular, inspired by the co-segmentation idea in [29], i.e. the same foreground object in two different images shares similar histograms of image features, we compare the color histogram of the segmented foreground region in each non-pre-segmented image $I_m \notin I_H$ with that of the color histograms in the pre-segmented images I_H to figure out the most erroneous segmentation.

In our implementation, we use only color histogram to identify the images with the most erroneous segmentation result but it can easily be extended to use other image features. Figure 4.4 presents the pseudocode for the algorithm to suggest the images for editing.

Calculate the color histogram of the segmented object region in each image.

Normalize the histogram by the size of the corresponding object region.

The normalized color histogram of each of the pre-segmented images is used as the reference.

for each non-pre-segmented image $I_m \notin I_H$

Calculate the correlation between the normalized histogram of I_m and that of each pre-segmented image.

Calculate the maximum correlation among the calculated correlations for I_m .

This maximum correlation represents the similarity between the segmented object region of I_m and the true object.

end for

Images are in turn suggested for editing in ascending order of the similarity between their segmented object region and the true object.

Figure 4.4: Pseudocode for automatic selection of images for editing.

Note that the algorithm only arranges the set of non-pre-segmented images in the order from the most to the least erroneously segmented image and in turn presents those images in the arranged order to the user. It is up to the user to choose to edit or skip a particular image and how many images he wants to edit before starting the silhouette refinement for the whole sequence, which is described in the next section.

4.5.2 Local editing and refinement

Similar to the first step where user needs to draw colored strokes on the selected images to guide the segmentation, in this step user needs to draw extra colored strokes to edit the segmentation result of each selected image. Note that even for the most erroneously segmented image, the erroneous areas are typically small. Thus, the user only needs to add strokes at the erroneous areas to edit the segmentation result locally and there is no need to segment the entire image again. In particular, we use the same robust interactive image segmentation method proposed in Chapter 3 to refine the user edited image here. A region R with radius r around the newly added user strokes are defined and only the pixels within this local region need to be relabelled, which is realized by setting all other pixels as foreground or background seeds according to their current labels. Figure 4.5 illustrates the editing process for the segmentation result of an image in the “lion” sequence (image numbered 10). Initially, there are segmentation errors at the leg area of the lion in the presented image (Figure 4.5(b)). A few user strokes are added to those areas, where red color strokes indicate foreground seeds and blue color strokes indicate background seeds. Figure 4.5(c) presents the foreground and background seeds for this image, where black color indicates background seeds, white color indicates foreground seeds and grey color indicates the pixels that could be relabelled (region R). All pixels in the image except for the pixels within the region R are set as foreground or background seeds. The new segmentation result is shown in Figure 4.5(d).

After editing and refining the most erroneously segmented image, we then perform the iterative 3D segmentation again to update the segmentation results of other images. Now the pixels in region R of the edited image also become hard-constraints for the 3D graph-cut. To speed up the process, only the t-link weights of the voxels that have projected pixels lying within region R need to be recalculated. Since only a small number of nodes are changed, the 3D graph-cut converges very fast, typically within 2 iterations. The fourth row and the sixth row of Figure 4.6

Figure 4.5: Local editing at an erroneous image.

show the segmentation results for the “lion” sequence before and after the local editing and refinement step, respectively. Image 9 is one of the pre-segmented images in the sequence. Image 10 is the selected image for editing. It is added with a few user strokes. Eventually not only image 10 but also images 14 and 15 are refined.

4.6 Experiments

4.6.1 Parameters

There are quite a few parameters in our proposed system. Most of the parameters have been discussed previously except λ_1 in (4.1), A in (4.5), and λ_2 in (2.1). Parameters λ_1 and λ_2 are empirically set to 0.2 and 100, respectively.

Parameter A in (4.5) needs to be a relatively large value to enforce the hard constraint. According to [4], the constant A is chosen such that

$$0 \leq w(v_n, S), w(v_n, T) \leq A \quad (4.9)$$

and

$$A > \max_{v_i \in P} \sum_{v_j: (v_i, v_j) \in E} w(v_i, v_j), \quad (4.10)$$

Equation (4.10) guarantees that if $v_n \in B_H$, then the min-cut algorithm will choose the cut through the edge $\{v_n, S\}$ and as a result, v_n will be classified as a

Figure 4.6: Segmentation results of the “lion” sequence. The first row shows some of the original images. The second row is the results with only color cue. The fourth row is the results with both color and silhouette cues. The sixth row shows the final results after further local editing, where a few user strokes are added to image 10. The images in the third, fifth and seventh rows highlight erroneous parts for the segmentation results above them (red color indicates false-positive parts and blue color indicates false-negative parts).

background voxel. Otherwise, if the algorithm chooses to sever the edge $\{v_n, T\}$ with the cost of $w(v_n, T) = A$, we would construct a smaller cost cut by restoring $\{v_n, T\}$ and severing all n-links from v_n (costs less than A) as well as the opposite t-link $\{v_n, S\}$ (zero cost $w(v_n, S) = 0$). From Equation (4.6) and (4.3), the t-link weights $w(v_n, S)$ and $w(v_n, T)$ are smaller than $2 - \phi$ and the n-link weight $w(v_i, v_j)$ is smaller than λ_1 , which is set to 0.2 as previously mentioned. Therefore, we set K to 2 to satisfy both conditions in Equation (4.9) and (4.10).

4.6.2 Segmentation Results

We first test two real-world multi-view image sequences: the 17-view 704×528 “lion” sequence and the 14-view 704×528 “sculpt face” sequence, whose camera parameters are pre-computed. Figure 4.6 shows the results of the “lion” sequence. The “lion” sequence is a very challenging one because the foreground and background colors are very similar and the shape of the lion is very complex. It can be seen from Figure 4.6 that with only the color cue, the multi-view object segmentation system is unable to produce an accurate object boundary in each image (see the second row of Figure 4.6). In contrary, with both the color cue and the silhouette cue, the system can generate much more accurate segmentation results where in general the segmentation boundaries are smooth and snapped to the geometry features (see the fourth row of Figure 4.6). In addition, with a little additional user input to image 10, we are able to generate almost perfect segmentations (see the sixth row of Figure 4.6) through the local editing and refinement, as mentioned in Section 4.5. The sixth row of Figure 4.2 shows the segmentation results of the “sculpt face” sequence with images 1 and 5 are two pre-segmented images. Similarly, the proposed system is able to produce very accurate silhouette for each view.

Next, we consider segmenting relatively large-scale multi-view image sequences. Instead of the time-consuming process of capturing large number of multi-view images through placing the camera at different locations, we capture a video sequence

using one camera moving around the rigid object and the video sequence is then uniformly sampled to generate a multi-view image sequence. The projection matrices for the sequence are then obtained using the method proposed in [15]: camera calibration from multi-view stereo. Despite the convenience in data acquisition, the multi-view image sequence generated from video recording brings in more challenges. In particular, as the camera is moving during the recording, many of the images in the sequence are blurred, which results in blurred boundaries between foreground and background regions and also makes the automatic camera calibration less accurate, compared to the case of static multi-view image acquisition. Figure 4.7 shows the generated “teapot” sequence, which consists of 56 frames with a resolution of 640×480 and four of them are pre-segmented. It can be seen that the blurring and the shadow at the lower part of the teapot makes it difficult to identify the object boundary even for human being (see the first row of Figure 4.7). Thus, the pre-segmented silhouettes may not be perfect and may provide a poor hard-constraint for the multi-view segmentation. We therefore relax the hard constraints in (4.5) and Section 4.3.4 by directly setting the likelihood $L_m(x_{m,n}, \Theta)$ to respectively be 1 and 0 for the foreground and background pixels within a small band around the boundary of the pre-segmented silhouette. Surprisingly, without strong hard constraints, the proposed multi-view segmentation framework can still produce accurate silhouettes without using any additional user input (see the second row of Figure 4.7). This is mainly because of the relatively large number of multi-view images available, which provides stronger 3D coherence and more hints among different views. It is worth mentioning that the fact that the 3D graph-cut collects information from a large number of images in different views also make it more robust to the inaccuracy in the projection matrices of small number of images. After segmenting the “teapot” video sequence, we use the resulted silhouettes to construct a 3D model for the teapot using the bundled depth-map merging method [24] as shown in Figure 4.8.

Figure 4.7: Segmentation results of the “teapot” video sequence, which is casually captured by a hand-held camera. For the uniformly sampled 56 image frames, images 1, 16, 31 and 46 are pre-segmented in the first step. The first row shows some of the original images, and the second row shows the corresponding results of our system.

Figure 4.8: The 3D model reconstructed from the silhouettes of the “teapot” video sequence.

Besides the above real-world multi-view sequences, we also test our system on three commonly used Middlebury datasets ¹: “DinoSparseRing”, “DinoRing” and “TempleSparseRing”. Table 4.1 summarizes the segmentation performance of all the sequences, with and without the silhouette cue, in terms of the error rate, i.e. the number of the mislabelled pixels over the object size. From Table 4.1, we can see that except for the “sculp face” sequence, where the error rate does not change much, for all the other sequences the error rates reduce significantly with the additional silhouette cue.

The proposed system is implemented in C++ and tested on a quad-core Intel 3.33 GHz Xeon Processor with 16 GB RAM. To achieve a decent running speed, we

¹<http://vision.middlebury.edu/mview/data/>

Table 4.1: Error rates of the segmentation results of different multi-view sequences.

Sequence	# of Views	Error Rate	
		with only color cue	with color & silhouette cues
Lion	17	5.2	3.8
sculp face	14	0.58	0.6
Tea Pot	56	2.1	0.9
Dino Sparse Ring	16	1.7	0.66
Dino Ring	48	1.72	0.6
Temple Sparse Ring	16	1.4	1.1

use the Open Multi-Processing (OMP) API to parallelize several major processes. Particularly, in step 3 of the silhouette refinement, up to 8 threads are used to process multiple images in parallel. In step 4 of the 3D graph-cut, multiple voxels in the voxel array are also processed in parallel. The parallelized program spends about 4 minutes to segment the “lion” sequence with the voxel array size of 150^3 and 1 minute to update the entire segmentation results after the local editing with the additional user strokes.

4.6.3 Limitations

In general, the proposed system can handle most of the common objects, but it would not be able to perform well on the objects with a lot of trivial boundaries such as trees and hair, which is due to the inherent limitation of the adopted 2D and 3D segmentation methods. In addition, our current system only utilizes the color cue and the silhouette cue. It is possible to make use of other cues such as the stereo-matching cue to further improve the segmentation performance at the cost of increasing the difficulty in trading off among multiple cues.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis, we have studied two problems of image segmentation: interactive single-image segmentation and interactive multi-view image segmentation. For single-image segmentation, the state-of-the-art interactive image segmentation algorithms are sensitive to the user inputs and often not able to produce accurate cutting contour with one-shot user input. They frequently rely on laborious user editing to refine the segmentation boundary. Therefore, in the first part of this thesis, we proposed a robust and accurate interactive image segmentation method based on the recently developed continuous-domain convex active contour model. We have demonstrated that our method significantly outperforms the state-of-the-art interactive segmentation methods. It exhibits many desired properties for a good segmentation tool, including the robustness to user inputs and different initializations, the ability to produce a smooth and accurate boundary contour, and the ability to handle topology changes. Our method runs very fast, taking less than three seconds in total to segment an image with a resolution of 640×480 , due to the fact that the proposed constrained active contour model can be solved quickly by a fast Split Bregman method and the adoption of the Geodesic algorithm for initialization.

For multi-view image segmentation, we have proposed a multi-view object seg-

mentation system, which is able to accurately segment a foreground object out of a set of multi-view images with a small amount of user input and acceptable processing speed. The system itself is a nice and coherent integration of several existing techniques including interactive image segmentation, 3D graph-cut and the convex active contour. The experimental results have demonstrated that the proposed system is a practical and effective tool that can perform well for accurate multi-view object segmentation even for very challenging multi-view image sequences.

5.2 Future Work

There are several exciting research directions that could extend or further improve the performance of our current work for both single-image and multi-view image interactive segmentation.

For interactive single-image segmentation, our robust segmentation method proposed in Chapter 3 is essentially a hard segmentation method. It cannot handle transparent or semi-transparent boundaries such as semi-lucent hair. It is interesting to extend the proposed method to soft segmentation by integrating an effective boundary matting tool. In addition, in the proposed single-image segmentation method, there are several free parameters that we choose heuristically by assigning them with different values and selecting the value that gives the best average error rates over all the testing images, such as λ in (2.1) and μ in (2.5). It is worth to develop a learning algorithm to learn those parameters in a supervised way using a larger dataset. Lastly, since our algorithm is highly robust to the user input's position, the users do not need to carefully place the strokes on the image. However, it would still be useful to conduct an analysis about how the scribbles should be given to achieve good segmentations.

For multi-view image segmentation, with the usage of OMP API to parallelize many major components of the multi-view image segmentation process as intro-

duced in Chapter 4, we have achieved a relatively fast execution speed. However, we believe that the processing time can be significantly reduced by re-implementing the program on GPU, which could possibly lead to a real-time multi-view image segmentation tool. Moreover, it is interesting to use the scale-invariant feature transform (or SIFT) matching across multi-view images to further enhance the performance of our interactive multi-view image segmentation method.

Bibliography

- [1] X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *IEEE ICCV*, pages 1–8, 2007.
- [2] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive GMMRF model. *Computer Vision-ECCV 2004*, pages 428–441, 2004.
- [3] E. Boyer. On using silhouettes for camera calibration. *Computer Vision-ACCV 2006*, pages 1–10, 2006.
- [4] Y. Boykov and M.P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in ND images. In *IEEE ICCV*, pages 105–112, 2001.
- [5] X. Bresson, S. Esedoglu, P. Vandergheynst, J.P. Thiran, and S. Osher. Fast global minimization of the active contour/snake model. *Journal of Mathematical Imaging and Vision*, 28(2):151–167, 2007.
- [6] NDF Campbell, G. Vogiatzis, C. Hernandez, and R. Cipolla. Automatic 3D object segmentation in multiple views using volumetric graph-cuts. *Image and Vision Computing*, 28(1):14–25, 2010.
- [7] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *International journal of computer vision*, 22(1):61–79, 1997.

BIBLIOGRAPHY

- [8] FHY Chan, FK Lam, PWF Poon, H. Zhu, and KH Chan. Object boundary location by region and contour deformation. *Vision, Image and Signal Processing, IEEE Proceedings*, 143(6):353–360, 2002.
- [9] T.F. Chan, S. Esedoglu, and M. Nikolova. Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM J. Appl. Math.*, 66, no. 5:1632–1648, 2006.
- [10] T.F. Chan and L.A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001.
- [11] K. Chang, T. Liu, and S. Lai. From co-saliency to co-segmentation: an efficient and fully unsupervised energy minimization model. In *CVPR*, pages 2129–2136, 2011.
- [12] A. Criminisi, T. Sharp, and A. Blake. Geos: Geodesic image segmentation. *ECCV*, pages 99–112, 2008.
- [13] O. Duchenne, J.Y. Audibert, R. Keriven, J. Ponce, and F. Ségonne. Segmentation by transduction. In *IEEE CVPR*, pages 1–8, 2008.
- [14] A.X. Falcao, J.K. Udupa, and F.K. Miyazawa. An ultra-fast user-steered image segmentation paradigm: live wire on the fly. *IEEE Transactions on Medical Imaging*, 19(1):55–62, 2002.
- [15] Y. Furukawa and J. Ponce. Accurate camera calibration from multi-view stereo and bundle adjustment. *International journal of computer vision*, 84(3):257–268, 2009.
- [16] T. Goldstein, X. Bresson, and S. Osher. Geometric applications of the split Bregman method: Segmentation and surface reconstruction. *Journal of Scientific Computing*, 45, no. 1-3:272–293, 2010.

BIBLIOGRAPHY

- [17] L. Grady. Multilabel random walker image segmentation using prior models. In *IEEE CVPR*, volume 1, 2005.
- [18] L. Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006.
- [19] C. Hernandez, F. Schmitt, and R. Cipolla. Silhouette coherence for camera calibration under circular motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 343–349, 2007.
- [20] D.P. Jang, D.S. Lee, and S.I. Kim. Contour detection of hippocampus using dynamic contour model and region growing. In *Engineering in Medicine and Biology Society, 1997. Proceedings of the 19th Annual International Conference of the IEEE*, volume 2, pages 763–766. IEEE, 2002.
- [21] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *IJCV*, 1(4):321–331, 1988.
- [22] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 150–162, 1994.
- [23] W. Lee, W. Woo, and E. Boyer. Identifying foreground from multiple images. *Computer Vision–ACCV 2007*, pages 580–589, 2007.
- [24] J. Li, E. Li, Y. Chen, L. Xu, and Y. Zhang. Bundled depth-map merging for multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010*, pages 2769–2776. IEEE.
- [25] Y. Li, J. Sun, C.K. Tang, and H.Y. Shum. Lazy snapping. *ACM Transactions on Graphics (TOG)*, 23(3):303–308, 2004.
- [26] E.N. Mortensen and W.A. Barrett. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60(5):349–384, 1998.

BIBLIOGRAPHY

- [27] L. Mukherjee, V. Singh, and J. Peng. Scale invariant cosegmentation for image groups. In *CVPR*, pages 1881–1888, 2011.
- [28] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH*, 2004.
- [29] Carsten Rother, Thomas P. Minka, Andrew Blake, and Vladimir Kolmogorov. Cosegmentation of image pairs by histogram matching - incorporating a global constraint into MRFs. In *CVPR*, pages 993–1000, 2006.
- [30] S.M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. 2006.
- [31] A.K. Sinop and L. Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *IEEE ICCV*, pages 1–8, 2007.
- [32] D. Snow, P. Viola, and R. Zabih. Exact voxel occupancy with graph cuts. In *IEEE Conference on Computer Vision and Pattern Recognition, 2000*, volume 1, pages 345–352. IEEE, 2000.
- [33] Mario Sormann, Christopher Zach, and Konrad Karner. Graph cut based multiple view segmentation for 3D reconstruction. In *Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, pages 1085–1092. IEEE Computer Society, 2006.
- [34] Y. Wang, W. Yin, and Y. Zhang. A fast algorithm for image deblurring with total variation regularization. *Rice University CAAM Technical Report TR07-10*, 2007.
- [35] C. Yang, R. Duraiswami, NA Gumerov, and L. Davis. Improved fast Gauss transform and efficient kernel density estimation. In *IEEE ICCV*, pages 664–671, 2003.

BIBLIOGRAPHY

- [36] W. Yang, J. Cai, J. Zheng, and J. Luo. User-friendly interactive image segmentation through unified combinatorial user inputs. *IEEE Transactions on Image Processing*, 19(9):2470–2479, Sept. 2010.

Publications

- [1] N.T.N. Anh, W. Yang, and J. Cai. Seam carving extension: a compression perspective. In *Proceedings of the 17th ACM International Conference on Multimedia*, pages 825-828. ACM, 2009.
- [2] N.T.N. Anh, J. Cai, J. Zhang and J. Zheng. Robust Interactive Image Segmentation Using Convex Active Contour. Accepted at *IEEE Transactions on Image Processing*.
- [2] N.T.N. Anh, J. Cai, J. Zhang and J. Zheng. Constrained Active Contours for Boundary Refinement in Interactive Image Segmentation. Accepted at *IEEE International Symposium on Circuits and Systems 2012*.