VIDEO SEGMENTATION BY LEVEL SET

WANG PEILIN

(B.Sc., NATIONAL TAIPEI UNIVERSITY, 2010)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2014

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Wang Peilin

31 - Mar - 2014

Wang Peilin

31 March 2014

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor, Assoc. Prof. Ping Tan, for his instructive advice and useful suggestions on my thesis. I am deeply grateful of his help in the completion of this thesis. I am also deeply indebted to all my colleagues in Vision and Machine Learning Laboratory, National University of Singapore. I really enjoyed the pleasant stay with these talented and brilliant people for the past 2 years. Special thanks should go to my friends who have put considerable time and effort into their comments on my thesis draft. Finally, I am indebted to my parents for their continuous support and encouragement.

Contents

1 Introduction **1**

1.1 Overview of Video Segmentation 1

1.2 Thesis Organization 5

2 Literature Survey **7**

2.1 Continuous methods 7

2.1.1 Snakes 7

2.1.2 Level Set 8

2.2 Discrete methods 9

2.2.1 K-Means 9

2.2.2 Intelligent Scissor 10

2.2.3 Graph Cut 11

2.2.4 Lazy Snapping 14

3 Level Set Based Video Segmentation **17**

3.1 Preprocessing 18

3.2 Level Set with Probabilistic Classifier 18

3.2.1 Region Term 19

3.2.2 Boundary Term 22
3.3 Optimization 26

4 Guided User Interaction **30**
4.1 Segmentation Error Detection 31
4.1.1 Feature Design 31
4.1.2 Detection 35
4.2 Suggestive Auto-Correction 37

5 Evaluation & Experiment **41**
5.1 Evaluation 41
5.1.1 Limitation 41
5.1.2 Future Study 43
5.2 Experiment 46

6 Conclusion **59**

Abstract

Video segmentation is a process that partitions a video into multiple segments frame by frame. The goal of segmentation is to simplify the representation of the video into something that is more meaningful and easier to analyze. Different from image segmentation, it is more difficult to cut out an object from video sequences than from one still image. Because in a video, there are many more frames to be segmented. It is possible to propagate the segmentation from the first frame to the later frames by analyzing the motion information, while motion estimation is often fragile especially at occlusion boundaries or motion blurred regions. Extensive research has been done in video segmentation in recent years since it is fundamentally important and useful. This thesis develops a novel interactive video segmentation algorithm.

Recently, local and global classifiers are considered as the key components of a video cut out system such as Video SnapCut which is integrated into Adobe After Effect. Shape prior is proposed in Video SnapCut to further refine the segmentation boundary to make the probability map more reliable. The global classifier use global image statistics to roughly classify every pixel as background or foreground. In comparison, the local classifier focuses on a local window to obtain more precise statistic information for classification. In most of the cases, the segmentation boundary between two consecutive frames changes gradually. Shape priors enforce the segmentation contour to be similar to the previous frame, which is very important to deal with video

frames where the foreground and background have similar appearance and are difficult to be separated by local or global statistic information.

This thesis makes several contributions to the video segmentation problem. Firstly, level set segmentation method is proposed to estimate the object shape by integrating the local, global statistics, and also shape priors. In recent state-of-art method, graph cut, its objective function aims at finding a cut that could minimize the total energy cost with least penalty. However, there are some problems with this graph cut based method. For example, it usually cuts off the elongated object or can not deal with complex-structured object in order to minimize the energy. Also, it may wrongly connect foreground and background which the color space are similar or in blurred region. It is more difficult to capture arbitrary topology and model complicated objects for graph cut. Unlike recent state-of-art method, graph cut, level set handles elongate and branch-structure objects better and gives more reliable results after propagation. Level set techniques seek local optimizations in each frame and graph cut finds a global optimization in the whole video sequences. While in many applications, a global optimal is preferable than a local one. However, since the objective functions in segmentation can often be biased, a global segmentation might not be suitable. The proposed methods produce better results both quantitatively and qualitatively than existing approaches. In addition, the proposed method is generic and and can be incorporated into any existing techniques.

Secondly, guided user interaction is proposed to provide user a friendly graphic user interface and reduce human interaction in result correction. All existing interactive video segmentation techniques require the user to

manually identify segmentation errors and correct them after the initial automatic segmentation. The user typically needs to carefully examine every video frame after each automatic segmentation step. This process is often tedious considering there are often over hundreds of frames in a video block. Therefore, auto detection and correction system are proposed and discussed in detail. For this purpose, we train an automatic classifier to identify segmentation errors. Specially, we sample overlapping windows along the segmentation contour and classify them as correctly or incorrectly segmented according to various features. We designed 10 different features, including edge and boundary matching, global and local consistency, shape consistency, local model confidence, etc.

Normally, it is not possible that single feature can divide the positive and negative data precisely. It is very difficult to improve the quality of segmentation by simply adjust one single feature or apply a same adjustment for all the incorrect segmentation parts. Thus, ten designated features are combined into a 10 dimensional feature vector, and this feature vector is computed in a patch centered on the segment boundary. For each patch, a score is measured as the difference between the segmentation result and the ground truth. Moreover, we raise the weights for those mislabeled pixels which are far from ground truth boundary. For the patches with higher scores, they will be reported automatically with higher probability as segment errors. Detecting error segments automatically could save user's time for inspecting the automatic results to identify segmentation errors. Moreover, in order to make the modification more efficient and easier, after detecting error local segment parts, several modification options are provided based on

the designated feature vector analyzing. User can simply choose the most correct one from the options. The designated features are also generic and could be incorporated into any existing techniques.

Chapter 1

Introduction

1.1 Overview of Video Segmentation

Video segmentation is a process that partitions a video into multiple segments frame by frame and it is still a challenge in Computer Vision. It is a fundamental important research topic with the main purpose to simplify the representation of a video into something that is more meaningful and easier to analyze. Generally speaking, similar to image segmentation, the purpose is to distinguish foreground and background from a still image. But there are different difficulties between human and computers. Foreground and background could be intuitively and easily distinguished by human, but as for computers, they are all digital numbers with the range from 0 to 255. It is still a long way from computers to do segmentation work unassisted. It is difficult to do the segmentation work manually or automatically, the better way to achieve this goal is to do it semi-automatically.

Generally, image segmentation applies a data clustering method such as

K-Means [35] with foreground and background samples provided as input data. Then distance measurement such as Euclidean distance or Mahalanobis distance is used to computed the distance between the unknown sample to foreground/background centers. After obtaining the distance, the unknown sample would be assigned a score as the probability. After obtaining the probability map, the unknown samples would be further classified to foreground or background cluster. Different from image segmentation, video segmentation solves image segmentation dependently frame by frame. It inherits and propagates information from previous frame to next frame as a reference to further partition foreground and background.

Different from common segmentation task, this thesis focus on bi-layer segmentation. We update probability map locally in each frame instead of applying a global one like graph cut. Firstly, a global probability map is obtained from the initial ground truth. Secondly, local windows [37] are built along foreground boundary and propagated to next frame by Optical Flow method [34]. In each local window, shape prior and the independent classifier trained by designated features are merged into the original global probability map. In the end, this updated probability map is applied to level set method for final cut out optimization.

Traditionally, video segmentation is tedious because the user needs to browse the frame back and forth in order to identify incorrect or improper segmentations and interactively refine them. Tremendous methods have been achieved on image and video segmentation such as clustering methods, compression-based methods, region-growing methods, graph partitioning methods, histogram-based methods, split-and-merge methods, partial dif-

ferential equation-based methods, etc. These methods can be roughly divided as discrete or continuous based on their mathematic formulation.

Discrete methods often regard images as graphs where each pixel is considered as a vertex. Segmentation is generated by partitioning the graph into disconnected components by various optimization tools such as Normalized-Cut [33] or Graph-Cut [29, 5, 17]. These methods can be applied to videos by treating videos as a 3D graph directly. Bai et al. [37] sampled local windows along the segmentation boundary and build local models within these windows to better separate foreground and background in complicated scenes in a graph-cut based segmentation framework. They further improved the local color model by incorporating motion information in [3]. Zhong et al. [38] took a similar to [2] for the final segmentation.

Continuous methods such as the 'Snakes' [12] algorithm consider an image as a continuous 2D domain, and evolve a continuous contour to segment the image according to some predefined objective function. To handle topological changes of curves, Malladi et al. [19] introduced the level set method [23] to image segmentation. Late, Paragios and Deriche [25] combined geodesic active contours [7] and level set for supervised texture segmentation. These methods are widely used in medical image segmentation, where strong prior knowledge of shape, color and texture can be applied to separate a predetermined organ or tissue. Recently, there are a few works [31, 18] to apply level set methods for interactive image segmentation.

Video SnapCut is the most commonly used video segmentation cut out tool from Adobe After Effect. Graph-cut is its core algorithm. From our experiments, it could be discovered that graph cut often cuts out the elephant's

nose incorrectly in order to minimize its energy cost regardless the correct segments from previous frame. The compared result is shown in Fig. 1.1. The comparisons are given the first frame a ground truth for initialization. It could be seen that the results from graph cut and level set are still correct in Fig. 1.1 (a) and (b), but graph cut unexpectedly cuts off the elephant's nose in (c). Fig. 1.1 (d) shows that the user iteratively corrects the result by adding green and red strokes as foreground and background respectively. But it is incorrectly segmented by graph cut in Fig. 1.1 (e) again. Because graph cut looks for a global optimal in its energy optimization, the incorrect segmentation would be very difficult to recover back automatically in the following frames. Thus, once the elongated object such as the elephant's nose in this example was segmented wrongly, the user always needs to refine it back and forth repeatedly. Even the user interactively edits it back in current frame, graph cut would cut off the nose in the next frame again. But from the results of level set, user does not have to correct it after propagation.

In this example, it could be clearly found out that level set performs better than graph cut especially when handling elongated objects. Graph cut tends to converge to a global optimal which cuts off elongated object regardless of the original topologies and results in neighboring frames. It is the main reason that we use level set to segment video instead of using graph cut. In fact, level set based methods are well suited for video segmentation. First, it can capture arbitrary topology and model complicated objects easily. It is also possible to explicitly enforce the segmentation contour to be close to salient edges. Second, the segmentation of previous frames provides a nature initialization of the result on the current frame. It naturally enforces tem-

poral coherence and smoothness, which is difficult to achieve in graph-based optimization. Thus, level set is more suitable for video segmentation. Also, level set based method has already been used in [18] for image segmentation. We are motivated and further extend the basic framework of [18] for video segmentation. We also borrow local windows of [37] in this thesis. To achieve this goal we further combine probabilistic local window and depth information into propagation. Moreover, we combine shape prior and local models to refine and get more reliable results. As for better user interaction, we add many editing tools into our program for user to refine easily.

1.2 Thesis Organization

The remainder of this paper is organized as follows: in Chapter 2, we survey a variety of techniques and provided a tentative classification according to their properties; in Chapter 3, the proposed algorithm of level set based video segmentation is discussed in details; in Chapter 4, guided user interaction system which includes auto detection and correction is discussed in detail. Experiments, evaluation, and future study are given in Chapter 5. Chapter 6 concludes the thesis.

(a)

(b)

(c)

(d)

(e)

Figure 1.1: Comparison in video of elephant. Shows the shortcoming of graph cut that segments the elephant's nose incorrectly for minimizing energy cost. (a) and (b) The 2nd and 3rd frame respectively. (c) The 4th frame shows that graph cuts tends to converge to a global optimal without caring original topologies and neighboring frames. (d) The 4th frame after user's correction. (e) The 5th frame shows that graph cut incorrectly segments it again.

Chapter 2

Literature Survey

2.1 Continuous methods

2.1.1 Snakes

A 'snake' is an energy-minimizing active contour influenced by image forces and guided by external constraint forces that enforce it moving toward features such as edges or boundaries. Snakes are active contour because they can clamp nearby edges and localize them accurately. Snakes bring a new idea to visual problems such as edge and line detections, stereo matching, and motion tracking. But as mentioned before in [21], if the input is too blurred or not as sharp as enough, the forces that pushing snakes clamp onto edges may not strong enough to lock onto edges or boundaries accurately. As can be seen in Fig. 2.1, it could be discovered that if the edges/boundaries are too complex or blurred, snakes are fail to lock onto exact object user wants to cut out. And also, user may have to draw the snake curve may times in a

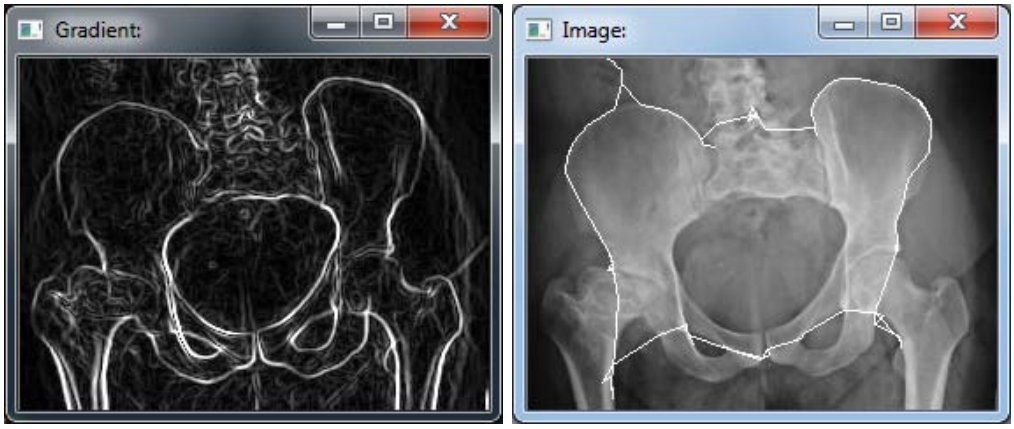


Figure 2.1: A result done by Snake, this active contour model would like to clamp onto the provided gradient map as accurate as possible.

complex region in order to get a better result which is too tedious.

2.1.2 Level Set

Level set based methods are well suited for video segmentation. It can capture arbitrary topology and can model complex objects. An introduction of and applications of level set method can be found in [23, 19, 18, 32, 24]. Level set method has been utilized to supervised image segmentation in [25]. Users are allowed to draw some strokes as hints of the foreground object. However, this work only relates to a simple statistical model and does not consider the edge costs of pixels. The drawbacks of previous work of level set based segmentation is not supervised. We will introduce and discuss about this approach in next chapter, and promote this work to a significant position with our methods.

2.2 Discrete methods

2.2.1 K-Means

K-means clustering method [35] is a popular method for cluster analysis in data mining. K-means clustering method aims at partitioning n points into k clusters in which each point belongs to the cluster with its nearest mean. For example, given a set of points $(x_1, x_2, \cdots, x_n)$, where each point is a d-dimensional real vector. K-means clustering method aims to partition the n points into k sets ($k \leq n$). $S = \{S_1, S_2, \cdots, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS):

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2 \tag{2.1}$$

where μ_i is the mean of points in S_i. Fig. 2.2 is an example for k-means of partitioning data points into 3 clusters illustrated by different colors. Fig. 2.3 shows the segmentation of k-means with k equals to 16.

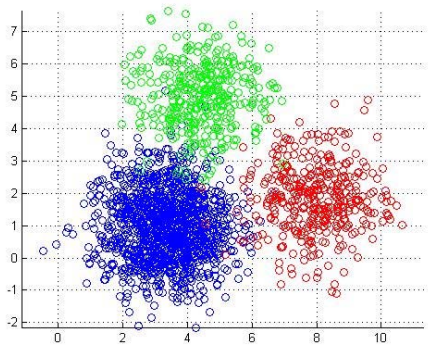


Figure 2.2: An illustration for K-means method of 3 clusters.

Figure 2.3: An example from Wikipedia with k = 16.

2.2.2 Intelligent Scissor

Intelligent scissors [21] are an interactive tool used for image segmentation and composition. It allows user to choose one point on an image and use it as a 'seed'. Once the seed is anchored, intelligent scissor will compute a global shortest path map by Dijkstra shortest path algorithm [8] and then find out a shortest path from current cursor position to the selected seed. Instant visual feedback is shown to user as in Fig. 2.4. Because Dijkstra algorithm [8] uses color gradient to build the shortest path map, once the edge is not sharp or clear enough, user may have to select more seeds for a better segmentation.

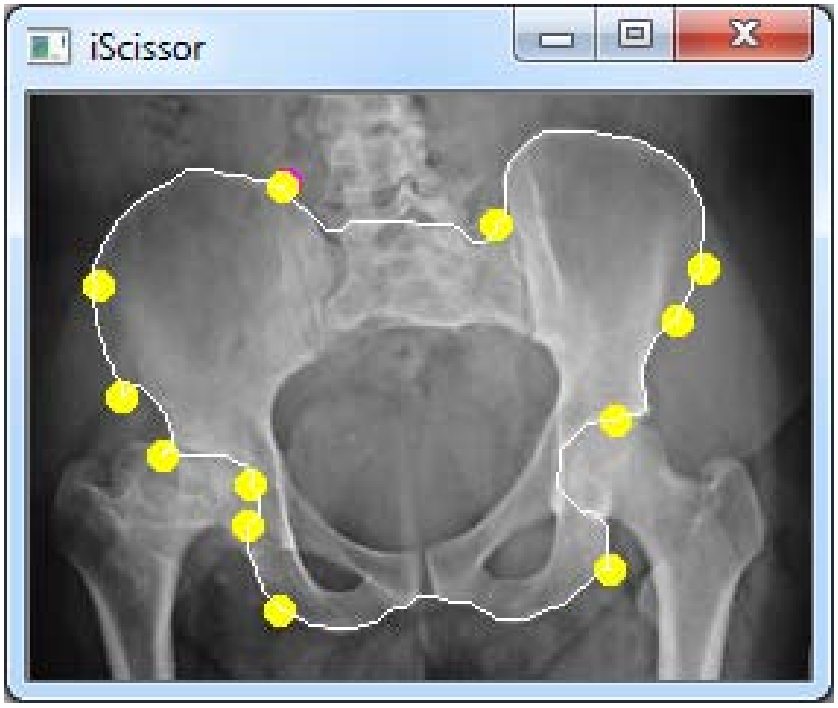


Figure 2.4: A segment of Pelvis bones cut out by Intelligent Scissor. The yellow points dedicate every seed that user has drawn, and the purple point dedicates the first seed.

2.2.3 Graph Cut

The theory of graph cut is first applied by Greig, Porteous, and Seheult in [10]. In the Bayesian statistical context of smoothing noisy (or corrupted) images, they show how the maximum a posterior (MAP) estimate of a binary image can be obtained by maximizing the flow through an image. Later on, this idea is further used in image segmentation. In graph theory, a cut could be referred as a partition of the vertices of a graph into two disjoint subsets that are joined by at least one edge. In an un-weighted undirected graph, the size or weight of a cut is the number of edges crossing the cut. In a weighted graph, the same term is defined by the sum of the weights of the

edges crossing the cut.

Graph cut can be applied to solve a wide variety of computer vision problems quickly, such as image smoothing, and many other computer vision problems that could be formulated in terms of energy minimization. Energy minimization problems could be reduced to instances of the maximum flow problem in a graph. In most of such problems in computer vision, the minimum energy solution corresponds to the MAP estimate of a solution. Although many computer vision algorithms involve cutting a graph (e.g., normalized cuts [33]), the term "graph cuts" is applied specifically to those models which employ a max-flow/min-cut optimization. Fig. 2.5 shows a toy example how graph cut finds a cut from those links.

On the other hand, graph cut is an algorithm that finds a globally optimal segmentation solution. Also known as min cut, and equivalent to max flow [36]. Suppose an image is a graph $G = \langle v, \varepsilon \rangle$, v is the set of all points, ε is the set of all arcs connecting adjacent points. It may be 4 or 8 connections between neighboring points. The goal is to assign a unique label x_i for each point $i \in v$. For example, $x_i \in \{foreground(= 1), background(= 0)\}$. $X = \{x_i\}$ can be obtained by minimizing $E(X) = \sum_{i \in v} E_1(x_i) + \lambda \sum_{(i,j) \in \varepsilon} E_2(x_i, x_j)$. Where $E_1(x_i)$ is the likelihood energy term which encodes the cost when the label of node i is x_i, $E_2(x_i, x_j)$ is the prior energy term which denotes the cost when the labels of adjacent points i and j are x_i and x_j, and λ is the weight of prior energy. E_1 encodes the 3-color similarity of a point, indicating if it belongs to foreground or background.

As for the definitions, $E_1(x_i)$ and $E_2(x_i, x_j)$ is defined as follows:

$$\begin{aligned} E_1(x_i = 1) = 1 \quad E_1(x_i = 0) = \infty \quad \forall i \in \mathcal{F} \\ E_1(x_i = 1) = \infty \quad E_1(x_i = 0) = 0 \quad \forall i \in \mathcal{B} \\ E_1(x_i = 1) = \frac{d_i^{\mathcal{F}}}{d_i^{\mathcal{F}} + d_i^{\mathcal{B}}} \quad E_1(x_i = 0) = \frac{d_i^{\mathcal{B}}}{d_i^{\mathcal{F}} + d_i^{\mathcal{B}}} \quad \forall i \in \mathcal{U} \end{aligned} \tag{2.2}$$

where $d_i^{\mathcal{F}}$ and $d_i^{\mathcal{B}}$ is the distance measurement of point i to foreground and background mean centers respectively.

$$\begin{aligned} E_2(x_i, x_j) = |x_i - x_j| \cdot g(C_{ij}) \\ g(\xi) = \frac{1}{\xi + 1}, \quad and C_{ij} = \|C(i) - C(j)\|^2 \end{aligned} \tag{2.3}$$

where E_2 is a function of the color gradient between two points i and j, on the other hand, it can also be considered as the penalty of assigning different labels to similar neighboring points.

Graph cut is further used in other application such as optimizing panorama in [26] or lazy snapping [17]. Fig. 2.6 shows the panorama result that is optimized by graph cut. There are 6 images in total, different colors of patches indicate which part is selected from those 6 images. Graph cut is also used and implemented into an advanced algorithm called Grab Cut [29]. Fig. 2.7 shows the results of grab cut in Open Source Computer Vision Library (OpenCV). It could be seen that the cut out segment is better than other approaches, and it is the reason that graph cut [10] is recently and widely used state-of-art algorithm in segmentation.

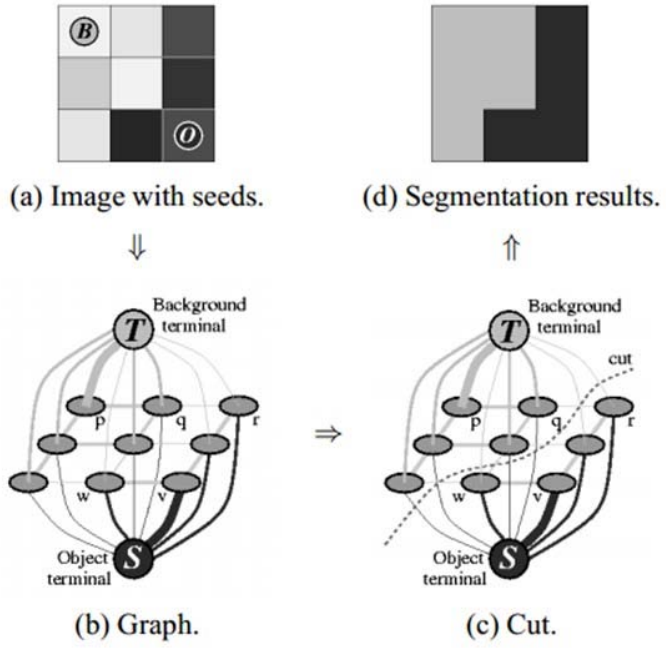


Figure 2.5: A simple 2D segmentation toy example for a 3x3 image. The seeds are $\mathcal{O} = \{v\}$ and $\mathcal{B} = \{p\}$. The cost of each pixel is reflected by the edge's thickness. The regional term (2) and hard constraints (4,5) define the costs of t-links. The boundary term (3) defines the costs of n-links. Inexpensive edges are attractive choices for the minimum cost cut.

2.2.4 Lazy Snapping

Lazy Snapping [17]is an interactive image cutout tool which can partition an object from an image easily. The core algorithms of it are Markov Random Field (MRF) [16] and Graph Cut [10]. It provides instant visual feedback, once the user draws foreground and background strokes on the target image, instant feedback is made and shown by an image segmentation algorithm which combines graph cut with pre-computed over-segmentation. Lazy Snapping provides a better user experience, and the segmentation re-

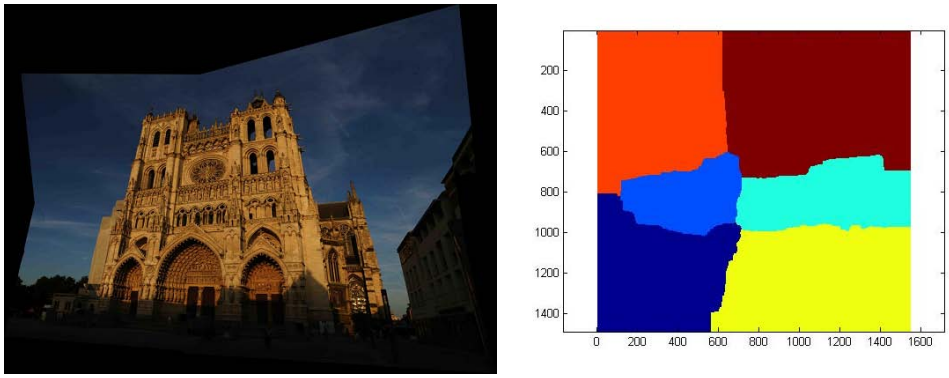


Figure 2.6: A result of panorama after optimizing by graph cut.

Figure 2.7: A test image of pelvis bones and its result by grab cut.

sults are also better than normal image cutout tool such as Magnetic Lasso in Adobe Photoshop. It could be seen that lazy snapping [17] based on graph cut [10] is efficient and accurate in cutting out still images. It is also further used in video segmentation in [37] and commercialized in Adobe After Effect renamed as Roto Brush. But Roto Brush also has many drawbacks such that it cannot deal with elongate objects or complex-structure models properly. It could be seen in Fig. 1.1, the elephant's nose is slight moving in these 3 consecutive frames, but Roto Brush cuts it out since it finds a global optimum.

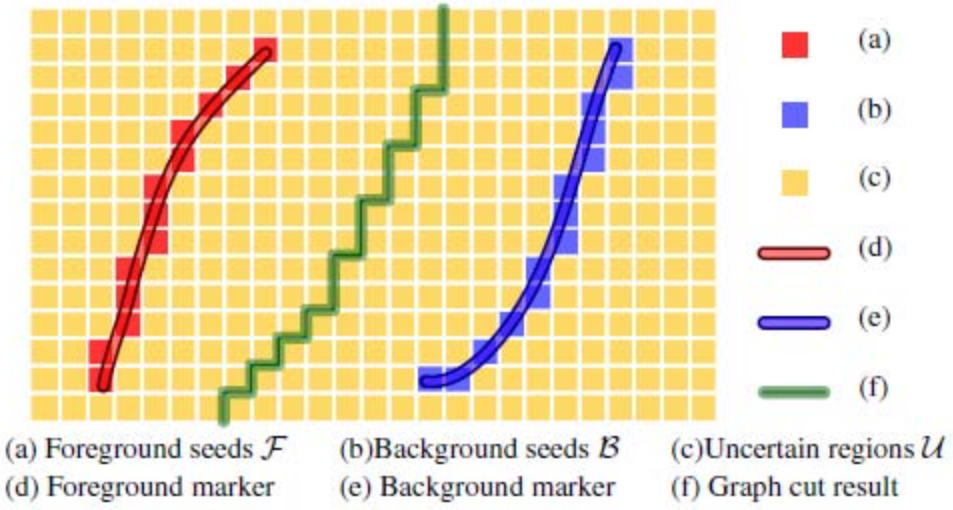


Figure 2.8: An illustration of lazy snapping. Usually, each yellow square stands for a pixel of an image.

Figure 2.9: A result done by lazy snapping. Fig. (a) shows the inputs, blue and red strokes stands for background and foreground inputs respectively; Fig. (b) shows the binary result after graph cut algorithm; Fig. (c) shows the visual instant feedback result to user.

Chapter 3

Level Set Based Video Segmentation

Level set based methods are well suited for video segmentation. First, it captures arbitrary topology and can model complicated objects easily. It is also possible to explicitly enforce the segmentation contour to be close to salient edges. Second, the segmentation of previous frames provides a nature initialization of the result on the current frame. It naturally enforces temporal coherence and smoothness, which is difficult to achieve in graph-based optimization. Because graph-based optimization computes a global optimal regardless of the result in neighboring frames. Third, the algorithm seeks for a local optimal near the initial, which makes results easy to control. It might be different from the user's expectation. When an elongated object is cut off, the user might need to correct it in all frame which is tedious and involved lots of human works.

3.1 Preprocessing

To facilitate the following level set based segmentation, we compute edge, texture and motion information at each pixel. We apply the Canny edge detection algorithm [6] to compute edges at each frame. In each frame, we detect Canny edges with different thresholds in order to obtain a much more detailed edge information. After that, a more reliable multi-threshold Canny edge result is obtained. We believe that level set could converge to the ground truth better by giving more promising edge detections. We compute a texture descriptor at each pixel by oriented filter banks. We employ 18 Gabor filters [20] within total 3 orientations and 2 scales at all color channels. Orientations and scales could be tuned with the video complexity. If there is a video with very high motion or in cluttered scenes with many foreground objects, more orientations and larger scales are required for parameter adjustments and feature extractions. We further compute the optical flow at all neighboring frames by the method described in [34]. Optical flow is implemented in CUDA for propagation optimization.

3.2 Level Set with Probabilistic Classifier

In level set based methods, the object boundary is implicitly represented as $\Phi(\mathbf{x}) = \mathbf{0}$, where $\mathbf{x}$ is an image coordinate. We refer Φ as the level set function in this paper. We segment video frames one by one, and at each frame, Φ is initialized by the propagated result with optical flow from its proceeding frame. We transfer segmented boundary of proceeding frame to

next frame optical flow. The zero level set is initialized on this propagated boundary. The values of Φ is updated according to the following equation,

$$\frac{\partial \Phi}{\partial t} = - \left[\underbrace{\alpha \cdot R(\mathbf{x}, t)}_{\text{region}} + \underbrace{\beta \cdot B(\mathbf{x}, t) + \gamma \cdot C(\mathbf{x}, t)}_{\text{boundary}} \right] \|\nabla \Phi\| \tag{3.1}$$

where t denotes the time of iteration. Roughly speaking, this equation evolves the segmentation boundary by the region force, edge force and curvature force.

3.2.1 Region Term

Region term includes statistic models of foreground and background regions. Most of the interactive segmentation methods such as [5, 37] build Gaussian Mixture Models (GMM) to separate foreground and background points. We train an Adaboost based classifier [30, 11] for such a separation. It is well known [22, 14] in the machine learning that discriminative models such as Adaboost outperform generative models such as GMM. Also, since we propagate the result of previous frame to next frame in our work. We believe that it is better to use discriminative models rather than using generative ones.

We train a global classifier for each video sequence. The key frame which segmented by user provides us a training data set of discriminative classifier. Both the 3-color channels and the Gabor filter responses are concatenated together to form a 4 dimensional vector. This vector would be used as the feature descriptor of a pixel, which is the input of discriminative classifier. We sample more pixels near the segmented boundary and less pixels outside in

order to capture the object with higher correctness. Also, to get a reasonable training data, there is a maximum number of sampled pixels in every bin of color histogram.

In most of the time, the information available from a local frame neighborhood is already sufficient to perform foreground/background classification on a pixel. Such information includes pixel colors and Gabor filter responses. To fully utilize local pixels' classification results in global image segmentation, our first goal is to make the global posterior probabilities close to the likelihoods estimated by a local pixel classifier. We use the following energy term E_R to measure the degree of inconsistency between the two pixels and then try to minimize this energy:

$$E_R = \iint_I (\Phi(\mathbf{x}) + 2(P(\mathbf{x}) - 0.5))^2 d\mathbf{x} \tag{3.2}$$

where $P((\mathbf{x}))$ denotes the likelihood of pixel $(\mathbf{x})$ being part of the target object according to a local discriminative probabilistic classifier which only gathers evidences available from a local neighborhood.

Similar to [37], we also train local models in local windows sampled along the segmentation contours. Local windows are propagated by optical flows between adjacent frames. The final probability of each pixel belong to foreground is computed as

$$P(\mathbf{x}) = \lambda_G P_G(\mathbf{x}) + \sum_i \lambda_i P_L^i(\mathbf{x}), \tag{3.3}$$

where $P_G(\mathbf{x})$ is the output of global classifier of pixel $\mathbf{x}$, $P_L^i(\mathbf{x})$ is the output

of ith local window ($\mathbf{x} \in LW_i$), and $\lambda_G + \sum_i \lambda_i = 1$. The weight of λ_i is computed as

$$\lambda_i = \frac{\lambda'_i(\mathbf{x})}{\lambda'_G(\mathbf{x}) + \sum \lambda'_i(\mathbf{x})}, \quad \lambda'_i(\mathbf{x}) = \exp(-\|\mathbf{x} - \mathbf{x}_c^i\|) \ , \tag{3.4}$$

where $\mathbf{x}_c^i$ is central pixel of ith local window.

We compute $R(\mathbf{x}) = \mathbf{\Phi} - \mathbf{2P}(\mathbf{x}) + \mathbf{1}$, where $P(\mathbf{x}) \in [\mathbf{0}, \mathbf{1}]$ is the probability that $\mathbf{x}$ belongs to foreground. This term minimizes inconsistency of Φ and output of classifiers $P_G(\mathbf{x})$. Updating Φ according to this region term essentially minimized the following integration in (3.2).

We design the following force term to reduce the energy defined in (3.2):

$$R(\mathbf{x}) \frac{\nabla \Phi}{\|\nabla \Phi\|} = (\Phi(\mathbf{x}) + 2(P(\mathbf{x}) - 0.5)) \frac{\nabla \Phi}{\|\nabla \Phi\|}, \tag{3.5}$$

where $\frac{\nabla \Phi}{\|\nabla \Phi\|}$ represents the unit outward normal vector of the zero level set.

Suppose $\mathbf{x}$ is a point on the zero level set. $P(\mathbf{x}) > \mathbf{0.5}$ means $\mathbf{x}$ is considered inside the target object by the local classifier. To make the posterior probability consistent with the local classifier, the zero level set should be expanded along the outward normal direction, which is consistent with the force term prescribed in (3.5). On the other hand, $P(\mathbf{x}) < \mathbf{0.5}$ means $\mathbf{x}$ is considered outside the target object by the local classifier. To make the posterior probability consistent with the output of classifiers, the zero level set should be shrunk along the reversed normal direction, which is also consistent with the force term prescribed in (3.5).

3.2.2 Boundary Term

Boundary term enforces the segmentation boundary to be close to salient image edges or some prior edges propagated from previous frames. We further some suppress edges ε_C near the boundary ε_B of propagated from previous frame. We use minimum distance as criteria to suppress Canny edges ε_C near ε_B. We select useful Canny and suppress the spurious edges as follows,

$$\mathbf{x}_i \in S_e, \quad \text{if} \begin{cases} \mathbf{x}_i \in \varepsilon_C \\ \arg\min_{\mathbf{x}_i} \|\mathbf{x}_i - \mathbf{x}_j\|, \exists \mathbf{x}_j \in \varepsilon_B \end{cases}, \tag{3.6}$$

That means there is a pixel in ε_B and $\mathbf{x}_i$ is its the nearest pixel. Then $\mathbf{x}_i$ is selected in set S_e.

The result is called an *edge field*, $\Psi(\mathbf{x})$.

$$\Psi(\mathbf{x}_i) = \begin{cases} 0 & \text{for} \quad \mathbf{x}_i \in S_e; \\ \frac{\min(d_{max}, \min_{\mathbf{x}_j \in S_e} d(\mathbf{x}_i, \mathbf{x}_j))}{d_{max}} & \text{for} \quad \mathbf{x}_i \notin S_e, \end{cases} \tag{3.7}$$

where S_e is the set of edge pixels, and $d(\mathbf{x}_i, \mathbf{x}_j)$ is the Euclidean distance between two pixels $\mathbf{x}_i$ and $\mathbf{x}_j$.

We define $\Psi(\mathbf{x})$ as *edge field*. It equals to zero at remaining edges and gradually increases when moving away from an edge. We compute $B(\mathbf{x}) = -(\nabla \mathbf{\Psi}(\mathbf{x}) \cdot \frac{\nabla \mathbf{\Phi}(\mathbf{x})}{||\nabla \mathbf{\Phi}(\mathbf{x})||})$.

If there is no Canny edges detected in local windows or probabilities of pixels in a local window, we use mask transferred from previous segmented mask to be a shape prior of this local window. Shape prior is another force $S_p(\mathbf{x})$ combine in boundary term $B(\mathbf{x})$.

Our second goal is to make the zero level set snap to salient edges in the image because salient edges are likely to lie on the boundary of the target object. We rely on the edge field computed in the previous section to facilitate distant interactions between edges and level sets. Since the edge field reaches its minimal value at edge pixels and has a magnitude increasing monotonically with the distance from edge pixels, we design the following energy term E_B to measure the overall proximity between the zero level set and the set of detected edge pixels:

$$E_B(\Gamma) = \oint_\Gamma \Psi(\Gamma(s))\|\dot{\Gamma}(s)\|ds, \tag{3.8}$$

where $\Psi(\mathbf{x})$ is the edge field defined in (3.7), $\Gamma(s)$ is a 1D parametric representation of the zero level set and $s \in [0, 1]$, $\|\dot{\Gamma}(s)\|$ represents the derivative of the arc length of the zero level set with respect to the parameter. To minimize the energy E_B, we apply the Euler-Lagrange equations [39] to (3.8), and obtain the following equation for optimizing the position of points on the zero level set:

$$\frac{d\mathbf{x}}{dt} = -\left(\Psi(\mathbf{x}(s))\kappa(\mathbf{x}(s)) + \nabla\Psi(\mathbf{x}(s)) \cdot \mathbf{N}(\mathbf{x}(s))\right)\mathbf{N}(\mathbf{x}(s)) \tag{3.9}$$

where $\kappa(\mathbf{x})$ is the curvature of the zero level set at $\mathbf{x}$ and $\mathbf{N}(\mathbf{x})$ is the outward normal of the zero level set at $\mathbf{x}$. The first term in (3.9) tries to make the zero level set as straight as possible while the second term tries to move the zero level set towards relatively distant edges along the negative gradient of the edge field. Both terms can reduce the energy term defined in (3.8).

The first term in (3.9) concerns the smoothness of the detected object boundary, which we will be addressed in the next subsection. Therefore, here we only focus on the second term, which improves boundary localization. Since the normal of the zero level set can be formulated using the gradient of the level set function, we can replace the unit normal vector $\mathbf{N}(\mathbf{x})$ in (3.9) with the normalized gradient, $\frac{\nabla\Phi(\mathbf{x})}{\|\nabla\Phi(\mathbf{x})\|}$. Thus, we design the second force term for boundary localization as follows:

$$B(\mathbf{x})\frac{\nabla\Phi}{\|\nabla\Phi\|} = -\left(\nabla\Psi(\mathbf{x}) \cdot \frac{\nabla\Phi(\mathbf{x})}{\|\nabla\Phi(\mathbf{x})\|} - \delta S_p(\mathbf{x})\right)\frac{\nabla\Phi}{\|\nabla\Phi\|} \tag{3.10}$$

δ is computed from probabilities of local classifier as follows,

$$\delta = 1 - \frac{2\sum_{i=1}^{n}|P_L^i(\mathbf{x}) - 0.5|}{n}, \tag{3.11}$$

The curvature term is a standard force term in level set methods. It is primarily used for improving boundary smoothness. However, it is unnecessary to enforce boundary smoothness unconditionally, especially when the true object boundary has rough details. Our curvature term tries to provide a tradeoff between boundary smoothness and boundary faithfulness. That means in the neighborhood of unsuppressed edge pixels, boundary localization is still a more important goal than boundary smoothness. But in the absence of edge pixels, boundary smoothness serves as an effective prior to determine the shape and position of the local object boundary. Thus, we

define the curvature force term as follows:

$$C(\mathbf{x})\frac{\nabla\Phi}{\|\nabla\Phi\|} = -(\mu\kappa(\mathbf{x}) + (1-\mu)\Psi(\mathbf{x})\kappa(\mathbf{x}))\frac{\nabla\Phi}{\|\nabla\Phi\|}, \tag{3.12}$$

where $\kappa(\mathbf{x})$ denotes the curvature of the level set passing through $\mathbf{x}$, and $0 \leq \mu \leq 1$. Locally convex regions have $\kappa > 0$ on the boundary while locally concave regions have $\kappa < 0$ on the boundary. The second term in (3.12) modulates curvature with the edge field to weaken the curvature term in the neighborhood of edges. Nevertheless, the first term guarantees that the curvature term does not disappear completely as long as μ remains positive.

The curvature of a level set in a two dimensional level set method can be computed using the following equation, which has been proved in [15]:

$$\kappa = \frac{\Phi_{xx}{\Phi_y}^2 - 2\Phi_x\Phi_y\Phi_{xy} + \Phi_{yy}{\Phi_x}^2}{\left({\Phi_x}^2 + {\Phi_y}^2\right)^{\frac{3}{2}}}, \tag{3.13}$$

where Φ represents a signed distance transform of the zero level set in our method. Since it is extremely unlikely to have a circle with a radius smaller than the size of a pixel, we clamp any computed curvature to $[-\frac{1}{\Delta x}, \frac{1}{\Delta x}]$, where Δx represents the size of a pixel [24]. Since the default level set function in our method is defined using probabilities, the signed distance transform of the zero level set needs to be recomputed during every time step.

According to [7], this term essentially minimizes the following integration 3.8

3.3 Optimization

The equation in (3.1) can be efficiently solved using the Narrow Band Method in [32, 24] and the Fast Local Level Set Method in [27]. They restrict most computation to a narrow band of active pixels immediately surrounding the zero level set. In general, the narrow band is the following set of pixels, $\{\mathbf{x} : |\Phi(\mathbf{x})| < \theta_b\}$ where θ_b is a prescribed threshold (typically set to 6), and Φ is the signed distance transform of the zero level set. Note that we only need to update the signed distance transform within the narrow band during every time step.

Figure 3.1: Comparison. Upper: frames in videos. Middle: results of Roto Brush. Bottom: results of our algorithm. It could also be seen that even our algorithm is still incorrectly segmenting the grass beneath the player, the overall result is still better than graph cut. (left: Frame 82 of *Shrek*, right: Frame 114 of *Footballer*.)

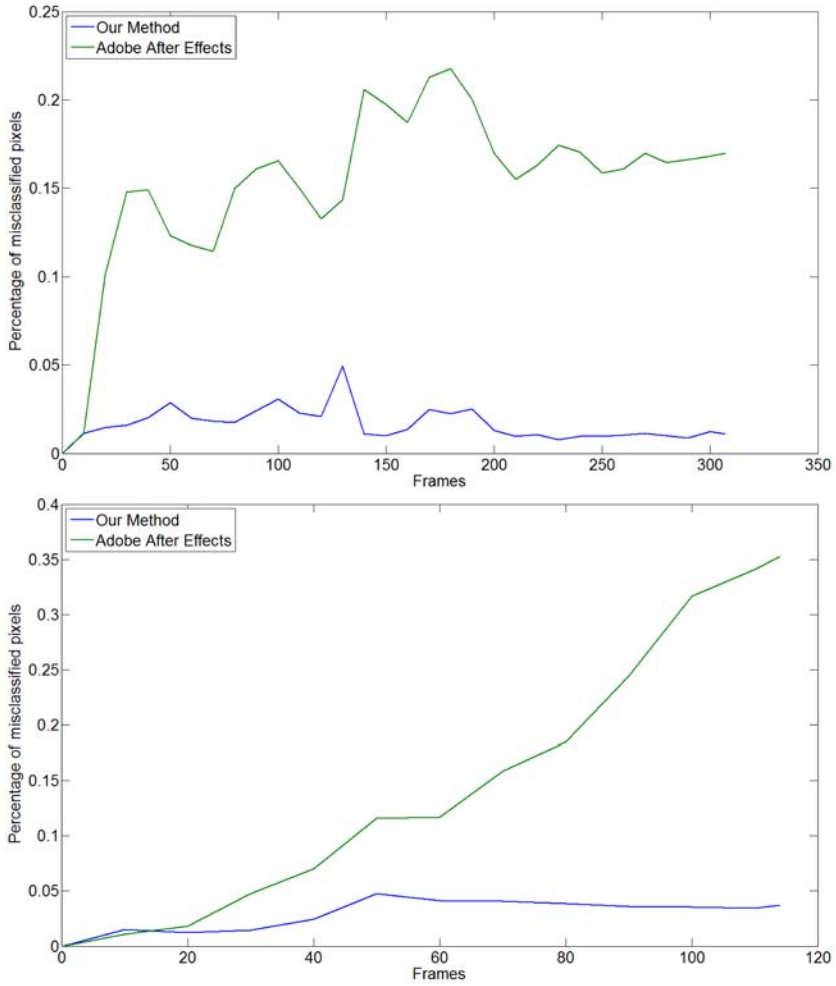


Figure 3.2: Comparisons of percentages misclassified pixels from Fig. 3.1. It shows our algorithm is more promising and seems Roto Brush usually requires constant initialization.

Because level set method is a local optimization tool and there is spatial coherence of frames in a video, level set based propagation in video segmentation is stronger than graph-cut in Snapcut. Fig. 3.1 and 3.2 show the comparisons of our algorithm and Roto Brush. If there is faster shape changes in video, the worse results are produced by Roto Brush. For example, the Shrek's left hand gets up quickly as shown in Fig. 3.1. Because

of local windows in previous frame can't follow this fast shape changes of Shrek, so the left hand is missed in the result of Roto Brush. Roto Brush is over dependent to boundaries and previous frame. For another example, a footballer is rolling in playground as shown in Fig. 3.1. There is a obvious straight-line edge in playground. Because a lot of local windows snap to this false and obvious boundary, a big patch of grassland becomes to foreground in result of Roto Brush. Roto Brush is liable to local minima in segmentation. Both Fig. 3.1 and 3.2 show that Roto Brush only cares the image patch in its local windows. Thus, local minimum cues, such as edges, become final segmented boundary and mislead Roto Brush to unreasonable results.

Fig. 3.3 shows the successful results from a video of a jumping cat. All of the results are only given a ground truth in 1st frame as a key frame. Although there is a motion blur occurred when the cat jumped, it could still be discovered that our method captures its hands in (c). It could also be seen that our method perfectly captured the cat's shape even when it played with a paper in (d), (e), (f), (g), and (h).

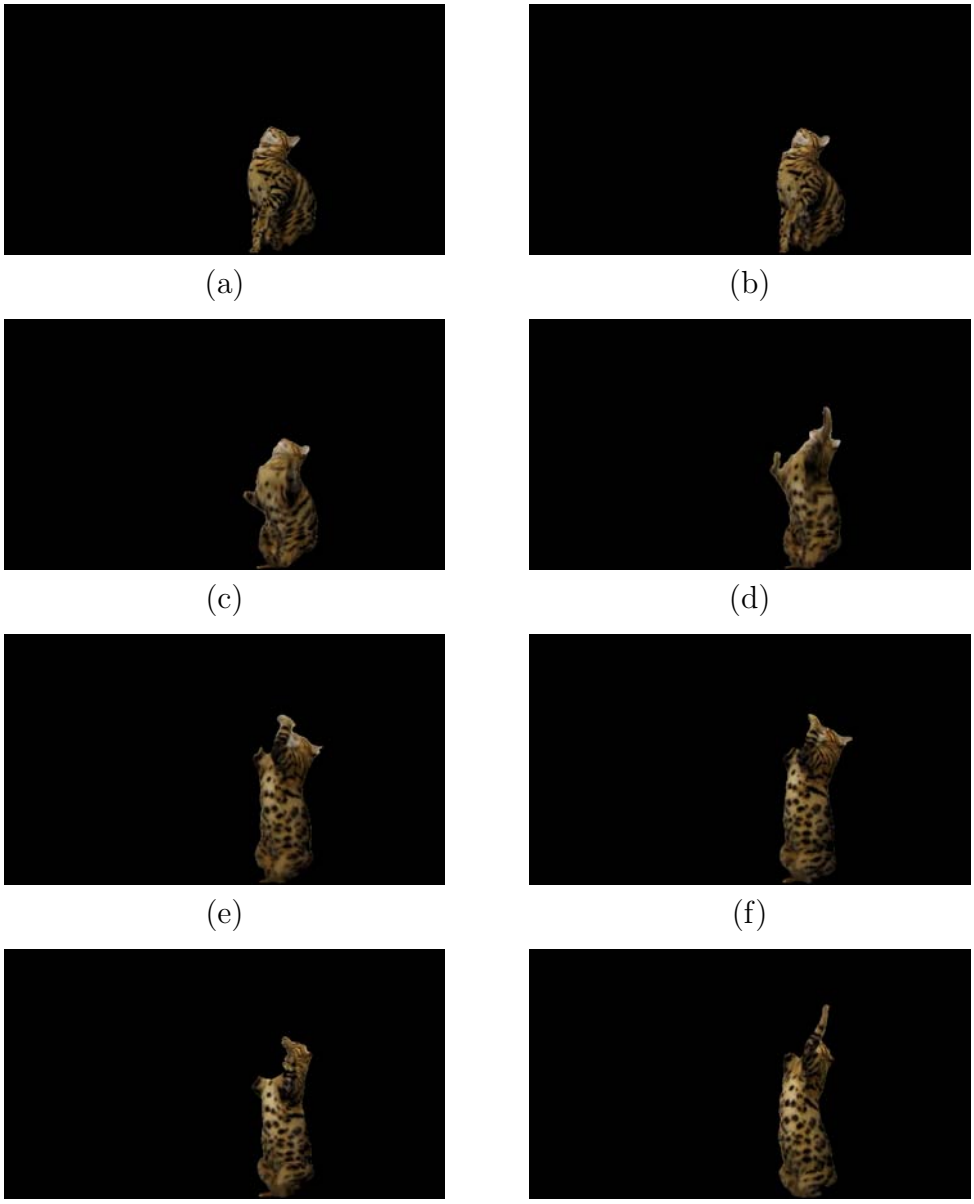


Figure 3.3: Successful results in video of Cat with our method. (a) The 51st frame, (b) the 56th frame, (c) the 60th frame, (d) the 62nd frame, (e) the 64th frame, (f) the 67th frame, (g) the 72nd frame, (h) the 81st frame.

Chapter 4

Guided User Interaction

Even there are tremendous researches about video segmentation and it has been achieved in many applications. The performance of video segmentation is still far from perfect. Users need to locate segmentation errors by retrieving frames back and forth and then draw additional strokes to correct them frequently. It is the most tedious part of video segmentation to browse the segmented frames back and forth and add new strokes repeatedly. Intuitively, if a system can automatically locate error segments, the users will be relieved from carefully checking the segmentation results. Furthermore, if the system can provide possible corrections for the user to select, the users do not even need to draw more strokes while correcting error segments. In order to achieve such way, interactive video segmentation would be more efficient and friendly.

Auto detection and auto correction could be one of the solutions to reduce these tedious refinements by the user. If we could collect the error segments and train a patch based classifier to identify incorrect segmenta-

tions. Furthermore, if the program could not only detect errors but also provide many possible segments and let the user select the ideal one. It will be more convenient since that the users do not need to do corrections anymore. The users only need to select the most correct segment patches which are provided automatically by the system. Toward this goal, we build a patch based classifier to identify incorrect segmentations. This classifier is trained from manually labeled ground truth data, and is applied to local windows of each frame. We define feature according to the property of video and segmentation parameters.

4.1 Segmentation Error Detection

4.1.1 Feature Design

Given a patch, P, centered on the boundary of the segmentation in frame F_i, and $p_j = (X_j, F_i) \in P$ are pixels inside the patch P, the features for a patch is defined below.

1. Optical Flow. The propagation of local model and shape prior all depends on optical flow. Hence, we take its accuracy measures as features to detect incorrect segmentation.

    - Disocclusion often causes problems for video segmentation. It can be measured by the number of non-matched pixels inside a patch. We define the set $N(F_i)$ as all the pixels in the frame F_i that are not destination of any pixel in the frame F_{i-1} according to the optical flow. We count the number of non-matched pixels as the

Non-Matched Point feature.

$$f = Size(\{p_j \in P\} \cap \{p_j \in N(F_i)\}) \tag{4.1}$$

- The length of optical flow measures the speed of moving object in image. When in case with high speed motion, the motion blur may cause ambiguity for segmentation. Also the reliability of the optical flow would also be impaired. We compute the average length of optical flow for pixels inside a patch as the **Optical Flow Length** feature.

$$f = \sum_{p_j \in P} ||X_j - X_j^B||^2 \tag{4.2}$$

2. Statistic models. We can analyze global and local statistic models to detect incorrect segmentations.

   - The global model and the local model measures the probability of foreground for each pixel. The probability evaluated at foreground (and background) pixels should be 1 (and 0). A pixel is ambiguous if its probability is close to 0.5. So we define the confidence of the global and local model as the **Global Model Confidence** and **Local Model Confidence** features.

$$f = \sum_{p_j \in P} |Global(p_j) - 0.5|, f = \sum_{p_j \in P} |Local(p_j) - 0.5| \tag{4.3}$$

   in which $Global(p_j)$ and $Local(p_j)$ are the probabilities of pixel p_j

under the global model and local model respectively.

- The foreground probability of a pixel is evaluated by both the global model and the local model. If these two models are inconsistent, the segmentation could be optimized according to the wrong one. We define a feature according to this consistency as the **Global & Local Consistency** feature.

$$f = \sum_{p_j \in P} |Global(p_j) - Local(p_j)| \tag{4.4}$$

3. Image features. We also analyze image features such as edge and blur to evaluate segmentation quality.

   - Intuitively, segmentation is difficult at image regions with less contrast caused by low resolution or motion blur. So we define the image blurriness following [1]. Specifically, for a pixel p_j, we select a $5 * 5$ patch centered at it and compute eigenvalues of the image intensity matrix, which are $\lambda_j^1, ..., \lambda_j^5$ in descend order. The **Blurriness** feature are defined as

$$f = \sum_{p_j \in P} \left(\frac{\sum_{m=1}^{2} \lambda_j^m}{\sum_{n=1}^{5} \lambda_j^n} \right) \tag{4.5}$$

   - The segmentation would be more likely to be correct if the boundary matches well the edges in the image. We compute shape context [4] descriptor at the center of each patch for both the segmentation boundary and image edges. Suppose $S(P)$ and $E(P)$ as the descriptors of the segmentation boundary and image edges

for patch P respectively. We measure the consistency of these two descriptors as the **Edge & Boundary Matching** feature.

$$f = \frac{S(P) \cdot E(P)}{||S(P)|| \cdot ||E(P)||} \tag{4.6}$$

Besides shape matching, we also measure the distance between canny edge and the boundary of our segmentation. For a pixel p_j, let $d(p_j)$ be the minimal distance from the pixel to any point on boundary, and let $c(p_j)$ be the strength of the canny edge on pixel p_j. The **Edge & Boundary Distance** feature could be computed as

$$f = \sum_{p_j \in P} d(p_j) * c(p_j) \tag{4.7}$$

Moreover, the segmentation contours at neighboring frames should be similar. Hence, we also evaluate the consistency of the shape context descriptor of segmentation contours at neighboring frames as the **Shape Consistency** feature.

$$f = \frac{S(P) \cdot S(P')}{||S(P)|| \cdot ||S(P')||} \tag{4.8}$$

Here, P' is the corresponding position of patch P in the frame F_{i-1} located by the optical flow of the center pixel of the P.

- The segment boundary would be wrong if it was far away from the center of the patch. To measure the closeness between segment boundary and the patch center, we compute the **Boundary**

Center Pass feature.

$$f = \sum_{p_j \in P} max(0, \frac{-d(p_j)^2 + 100}{100}) \tag{4.9}$$

4.1.2 Detection

In this section, we will introduce the training and detection of the automatic error report detector.

We collect training data from a set of videos with variety of topics. For a clip of video, we label the ground truth segmentations for all frames and run the result of our segmentation system. Training samples are collected as the local windows defined before. We consider each local window as a patch, typically with $31 * 31$ pixels. The 10 dimensional feature of a patch is computed as introduced above. To obtain the label for a patch, we first compute a score for each patch to measure the segmentation error inside the patch. The score is computed as the difference between our segmentation result and the ground truth in the patch. Moreover, we give larger weights for those mislabeled pixels far from ground truth boundary. For patches with higher scores, they should be reported by auto error report system with higher probability, and thus be positive training data(with label +1), and vice versa for the negative training data. Intuitively, the label of the training data could be obtained by assigning a threshold on the mislabel score. However, since the ground truth may contain noise, there exists a range of score which is hard for making decision of wrong or correct segmentation. So while training, we exclude patches with scores drop in such an confused range, and use only

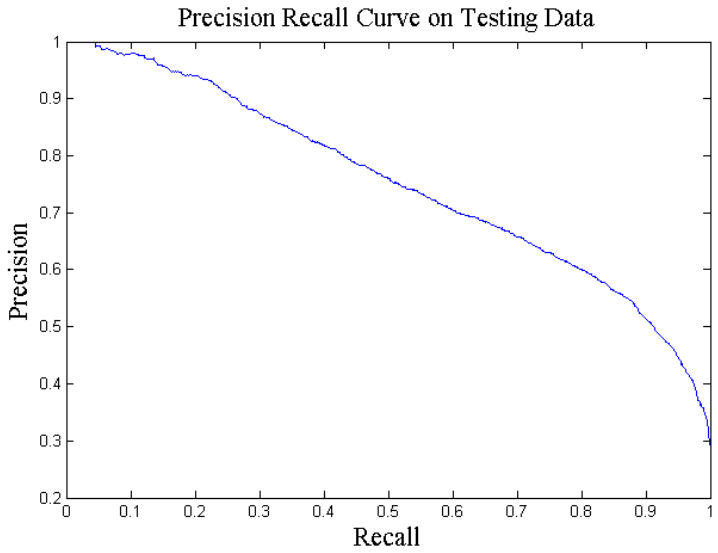


Figure 4.1: Precision Recall Curve on testing data.

those could be definitely classified by the mislabel score.

Considering the quite acceptable performance of our propagation system, it turns out that our training data is apparently unbalanced–the number of negative data(correct local segmentation) is more than that of the positive data. Also, it is not proper to arbitrarily predefine a model for fitting feature space to the label. We use the most traditional Gentle Adaboost as the classifier.

Figure 4.1 shows a precision-recall curve of our error detector on testing data. The average precision is 77.7%. In practice, we may choose a point on the precision recall curve as the working point based on our requirement on the quality of the reports. In our case, we want to ensure most of the reports to be really incorrect segmented parts, so we apply a comparatively high threshold on the output score of the Adaboost detector.

4.2 Suggestive Auto-Correction

Detecting error segmentations automatically could save users' time for trying to find error everywhere on segmentation boundary. Moreover, in order to make the modification more efficient and easier, after detecting error local segmentation parts, we try to provide several modification options. Users could simply choose one of the options (if any one modify the error correctly), but no longer need to consider about how to give supportive information, such as drawing strokes.

To provide modification choices, we first analyze the error detector. In Adaboost, a feature would be frequently selected by weak classifiers if it is informative in splitting the training data. In other words, the error segmented parts are much easier to be detected due to these features; and hence these features would have a large probability of leading to error.

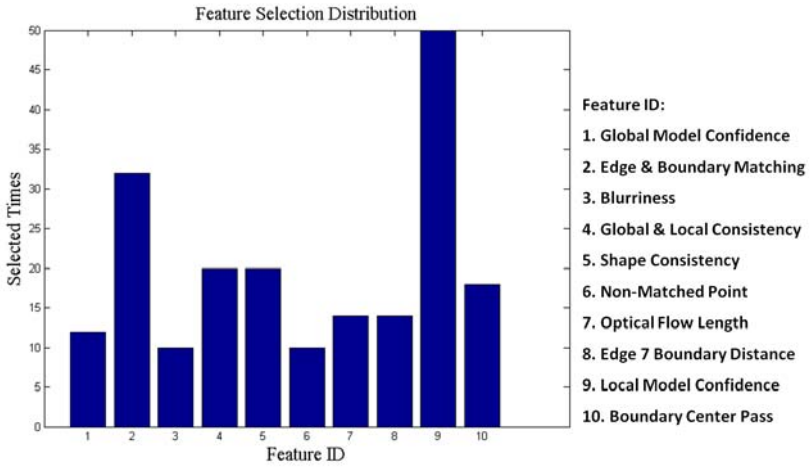


Figure 4.2: Feature Selection by Adaboost detector.

Figure 4.2 shows the times of each feature been selected by weak classifiers. It shows that **Edge & Boundary Matching**, **Global & Local**

Consistency, **Shape Consistency**, and **Local Model Confidence** are some most informative features. Accordingly, these discriminative features provide some suggestions in modifying the segmentation result as following.

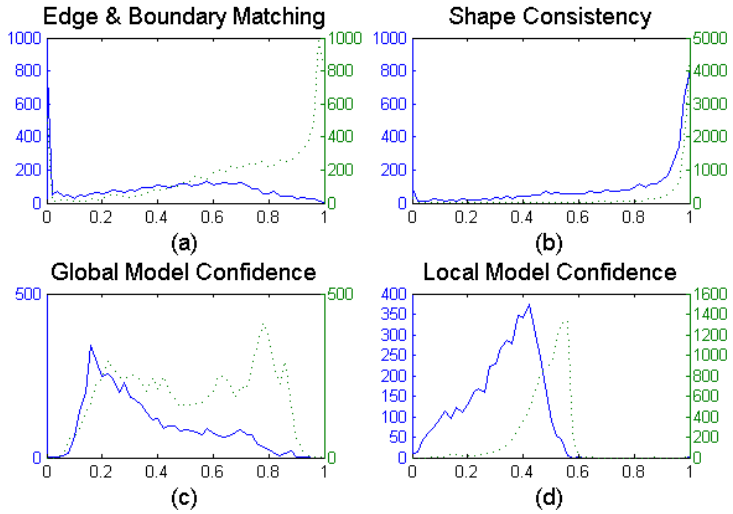


Figure 4.3: Feature distribution of 4 most discriminant features. The solid line shows the distribution of wrong segmentation parts, while the dash line is for correct segmentation parts.

1. **Edge & Boundary Matching**. Figure 4.3(a) shows the distribution of this feature on positive and negative samples, in which correct segment patches fits canny edge closer. This suggest that we should increase the force towards canny edges.

2. **Shape Consistency**. Figure 4.3(b) shows the distribution of this feature. Almost the same as **Edge & Boundary Distance**, incorrectly segmented parts should try to increase the weight of constraint from shape prior.

3. **Local Model Confidence** and **Global & Local Consistency**. It

is expectable that our algorithm would be weak when local classifier cannot clearly divide the pixel in patch. While local classifiers are weak or strongly conflict with global model, we may change to borrow the information from the global model, and hence increase the weight of global model. Figure 4.3(c)(d) shows the feature distribution of global and local model confidence. Both features show that the segmentation would be more likely to be correct if with higher color model confidence.

From the feature distributions, we see that there is no one feature which can precisely divide the positive and negative data, so it is hard to improve the quality of segmentation by simply adjust a single feature or apply a same adjustments for all the wrong segmentation parts. We pass the job of deciding rectification methods from several options to the users. After the algorithm automatically segment the video, the error detector will run on every local window and assign large scores to those windows with high probabilities to be error. Then we provide 4 choices of rectification, which emphasis on local model probability, global model probability, canny edge, or shape prior respectively, for users to make decision. To make the prediction precisely, for every 50 frames, we report the top 20 windows with the highest scores for users to process. After this, the system will run auto segmentation throughout the 50 frames again. This loop will stop until there are very few windows to report or most of the reported windows are actually correctly segmented.

With this system, it could be obviously discovered that the user could obtain better refined results by given fewer corrections especially in small and

detailed boundary. Normally the user needs to manually correct errors with carefulness for complicated boundary. Zoom in might be needed for precise corrections. But it could be seen from Fig. 4.4 and 4.5, our system already provides the most possible refinements with the most reliable and significant features trained by AdaBoost. Our system helps the user to describe complex object boundary automatically without human's supervision. We believe that the user will no longer need to deal with complex and cluttered errors such as Fig. 4.5. With more training experiences, our system will provide more precise and much higher reliable auto correction results.

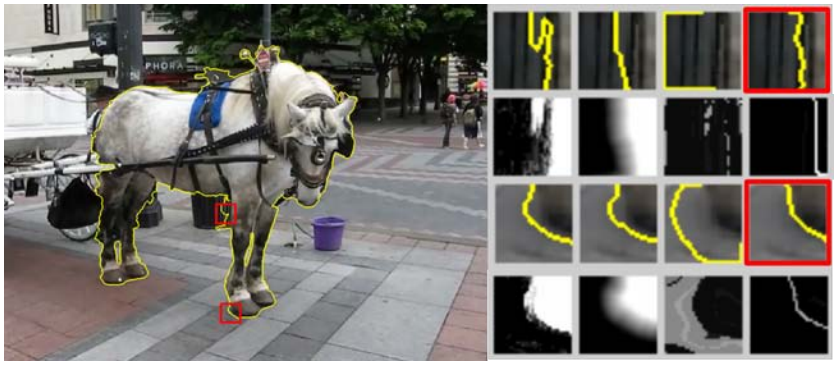


Figure 4.4: A case of auto correction system.

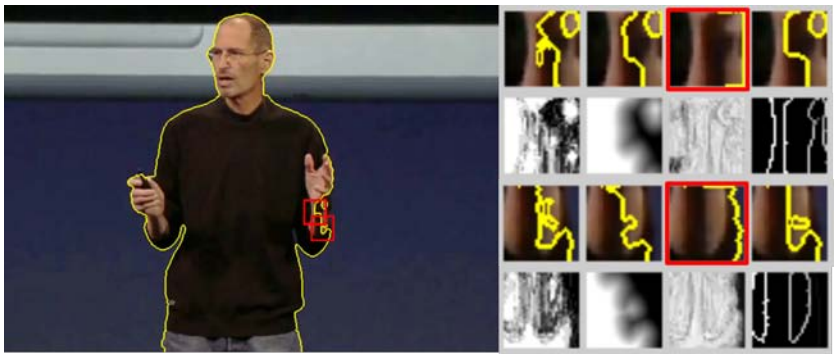


Figure 4.5: Another case of auto correction system.

Chapter 5

Evaluation & Experiment

5.1 Evaluation

5.1.1 Limitation

Although in most of cases, our segmented results are better than Video SnapCut. There are still many limitations of our approach such as computational time, memory problem, matting optimization, and hardware requirement, etc.

Computational Time

In our program, the average time of propagating one frame is $3 \sim 5$ seconds for the image with resolution 640x480. As for images with larger resolution such as 1024x720, the time may increase to $7 \sim 15$ seconds depends on the number of local windows. Since local windows are fully covered the entire boundary of the object, if the area of the object was very huge then the

computational time will increase proportionally. If the feature with those local windows is more complicated, then the computational time may increase exponentially. As for the computational time in Video SnapCut, the average time of each frame is under 1 second regardless image resolution since finding a global optimum is much faster than finding many local optimums.

Although we encode and speed up optical flow and level set optimization by GPU, the computational time is still longer than Video SnapCut. This limitation could be solved by a hard way which using clusters servers or a computer with higher hardware requirement and more powerful capacities. It could also be solved by a soft way that optimizing the source code to speed up classifiers' training time and propagations progress. It is believed that better data structures and memory pooling access could reduce time cost and make the program more efficient and less buggy.

Memory Problem

In the propagation progress, result of Canny edges, optical flow, histogram, Gabor feature, original video sequences are used during propagation. Especially in the preprocessing stage, pre-computed features such as edges, optical flow vectors, and Gabor feature response are saved into a mat file in MATLAB for further using. Once the video sequences are too long, for example, over 300 frames, the memory will overflow due to the storage limitation. It could be solved by increasing memory space but it is not the best solution. It is better if we could compute each feature whenever we need to use it and release its' memory while it is no longer used. But on the other hand, the computational time will increase and be shared to each frame since we have

to compute all the features we need for propagation in each frame.

So far we have not come up with a better solution for this problem. To use Solid Storage Disk (SSD) or to handle these feature with better data structure arrangement may be workable. But apparently, more related research or studies are needed to address and conquer this problem reasonably and properly.

Hardware Requirement

We run our level set based video segmentation program on a powerful computer with 24.0 GB RAM, 64-bit operating system, 3.33GHz processor, and NVIDIA Quadro 2000 independent graphic card. Because we use GPU to speed up optical flow and level set algorithm, a specific graphic card is needed for our program which is not general enough. And the memory size in the specific graphic card has to be large enough to handle optical flow buffering. From our experiment, overflow occurs when the image resolution is too large such as a raw image data with resolution 1600x900. Rewrite and encode the algorithm to a simpler one may be helpful. Because it is more convincing and general for normal users if this program could be generally used without such great hardware requirements.

5.1.2 Future Study

Specific Issue

Although most of the experimental results obtained by our presented methods are better and more accurate than by graph cut, the performance in the

scenes with very high motion or in cluttered scenes with many foreground objects is still far from perfect. To make the proposed methods work with such scenarios, we might have to come up with other new approaches. This is still an important issue worthy we spending time doing research on it. We do believe that this thesis would inspire not only us but also other researchers who are working on these topics and fields.

Additional Feature

Additional feature can be added into classifier training process such as depth map information, hard color constraint depends on the environment, or some other textures which could be specific to input data. It could be seen in Fig. 5.1 that the result of the human head is correctly segmented by integrating depth information into classifier. It yields that by adding additional and reasonable features into original feature space, performance can be promoted into a more significant position.

Bi-Direction Propagation

Most of the current video cut out systems, even ours, are segmenting object in a forward propagation. But it could be observed that sometimes the incorrectness occurs in forward propagation but could be correctly obtained in backward propagation. For an example, if a human waves his arm from background scene into his body, then new background which was covered by his arm will appear in sample spaces. It may cause some errors because new appeared area is not trained before and may be classified into wrong clusters. From another point of view, if a human waves his arm from inside his body

to outside, his arm may be tracked correctly by local windows.

So far bi-direction propagation is not widely used in any video segmentation tool. It may be useful somehow, but it may also be impractical since the computational time will be doubled. Therefore, how to find a balance between bi-direction propagation and merge it properly and efficiently could be one of the future study options.

Guided User Interaction

Although we have presented a guided user interaction system in chapter 4, but the performance is still far from perfect. We trained the Adaboost classifier only based on the features observed from our experiments. The performance could be further improved by adding more discriminative features to training space. Although the system pops up 4 the most possible results from our classifier, but there is still a chance that all of these provided results are unexpected and unreasonable. Further study is still needed to increase the correctness of this system and make it more reliable. It is believed that the accuracy and correctness could improved by better feature design.

Matting Optimization

Unlike Video SnapCut, we did not apply any Matting algorithm into our program. The boundaries of segmented objects sometimes are indented and not smooth enough. Since we did not apply any Matting algorithm to smooth the final cut out boundary, the boundary with more motion blur may return unexpected result to the user. Once an improper or unexpected result was obtained, user may have to correct it frame by frame which is too tedious

and unacceptable.

Nowadays, tremendous research on matting algorithm such as [9, 28, 13] has been achieved. In SnapCut, they already integrated matting algorithm to optimize the final cut outs. In the future, we would like to integrate matting algorithm and make the final result smoother and more reasonable.

5.2 Experiment

In the experiment, we compare our results with Video SnapCut [37] only since it is the current state-of-art video segmentation tool. We try to find the reasons of some awkward performances of Video SnapCut. In the comparisons of Video SnapCut and our algorithm based on level set, Video SnapCut is a segmentation tool which has less stronger ability of propagation than we thought at first. Especially in the situation of fast motions and foreground/background shape changes. However, we could not obtain the explanations of it from the source code of Video SnapCut. Therefore, we have to make some assumptions based on the paper [37] and various experiments. Finally, we think that local optimized solutions is better than global ones in performance of propagation from segmented results of key frames might be one of the possible reasons.

At present, the unique video cutout system to be used in commercial production is Video SnapCut [37] (renamed Roto Brush) which has been transferred to Adobe After Effects CS5. This method adopts a key frame-based forward propagation workflow, which means that the accurate segmentations in key frames provided by users are propagated to the adjacent frames based

on a set of local classifiers. Although this system has been put into practical use, it does not conform to user's habit in object selection. Besides, since the interactions only happen in key frames, the final results depend on accurate selections in key frames and stable propagations. If the video scenes are complicated, a lot of tedious manual corrections are still needed to obtain good results in every frame. We list the failure cases of Video Snapcut as follows.

Failure case 1: nose of elephant. The nose of elephant is a slim part of object. The elephants initialized to be foreground as shown in Fig. 5.2(a). However, the nose of elephant was classified to be background in next frames immediately in Fig. 5.2(b) and (c). We re-initialized segmented results in the 23rd frame as a key frame manually in Fig. 5.2(d). However, video SnapCut continued to do error segmentation in Fig. 5.2(e). It seems that there is only a slight movement of nose of elephant in these frames. But users have to do a lot of tedious segmentation work in every frame manually in order to correct it. So we guess that it caused by graph cut likes to cut slim object in foreground.

Failure case 2: Shrek's waving arm. In this experiment, the left arm of Shrek was wrongly classified when waving his arm. Firstly, we set the size of local window to be 30 by 30 pixels. So we guess that it caused by accuracy of optical flow. Then we set the size of local window to be 80 by 80 pixels. However, the left arm of Shrek was still missed in final result when waving his arm as shown in Fig 5.3 (b). Because this hand is contained in the search region as shown in Fig. 5.3 (c) and (d) separately, we think it is caused by graph cut in Video SnapCut wants to find global optimization and get rid of

the slim part of foreground.

Failure case 3: Nemo's tail. Nemo swings its tail quickly in the water as shown in Fig. 5.4. Segmentation result of Video SnapCut snaps to a wrong edge in Nemo's tail. This edge seems be the strongest edge in Nemo's tail. Finally, Video SnapCut missed Nemo's tail immediately from segmented key frame Fig. 5.4(a).

Failure case 4: tail of parrot. It seems to be similar to failure case 3 as shown in Fig. 5.5. Video SnapCut missed the slim tail of parrot when parrot swings its tail quickly in the video.

In the experiments, Video SnapCut often cuts out or connect regions incorrectly in order to minimize its energy cost. It could be discovered from the following results that level set handles elongated objects and complex scene better than Video SnapCut. The compared results are shown in Fig. 5.6, 5.7, 5.8, and 5.9. All of those comparisons are given only the first frame as ground truth to initialize the entire propagation. It could be clearly seen that without any user interaction, our video cut out system can provide better results for such elongated objects.

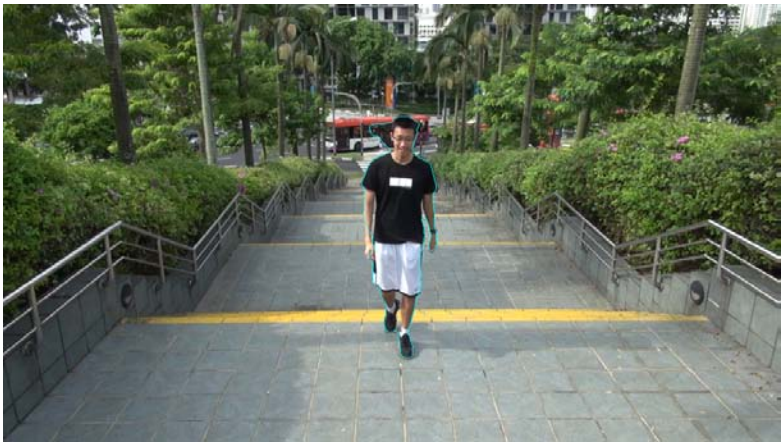
Case 1: left hand of the man. The shadow area occurs while the man puts down his hands. In the progress of seeking global optimum in Video SnapCut, it cuts off his shadow-cover left hand. It could be seen in Fig. 5.6 (b)$\sim$(d) that the left hand is cut off more and more in the propagation. In Fig. 5.6 (d) only the thumb is correctly segmented. User needs to refine the very hand almost frame by frame.

Case 2: left armpit area of the man. The original correct cut out region converges in the propagation. It shows another drawback of Video SnapCut

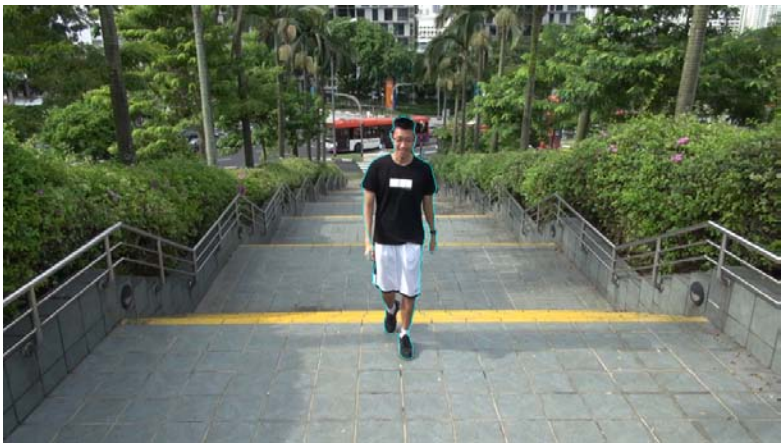
that it cannot handle elongated object well compare to our result. In could be discovered that the area is nearly disappeared in Fig. 5.7 (c). In this case, user also needs to refine that converged area frame by frame by hand.

Case 3: the blue t-shirt is wrongly cut out together with the black jacket. As mentioned before, Video SnapCut finds a global optimum. It is the reason that the blue t-shirt is considered as part of foreground. The incorrect segment did not correctly cut out even in Fig. 5.8 (d).

Case 4: both hands of the man are wrongly classified as part of background. When the hands move inside his body, they are considered as a foreground in order to optimize the energy function of graph cut. Even the right hand is recovered in Fig. 5.9 (d), but the left hand is still error. Instead of finding a global optimum, it could be seen that local optimum gives better result as shown in left-hand side images.



(a)



(b)

Figure 5.1: Comparison in video of walking man. (a) The 23rd segmented frame with original features, (b) the 23rd segmented frame with additional depth information added into training space.

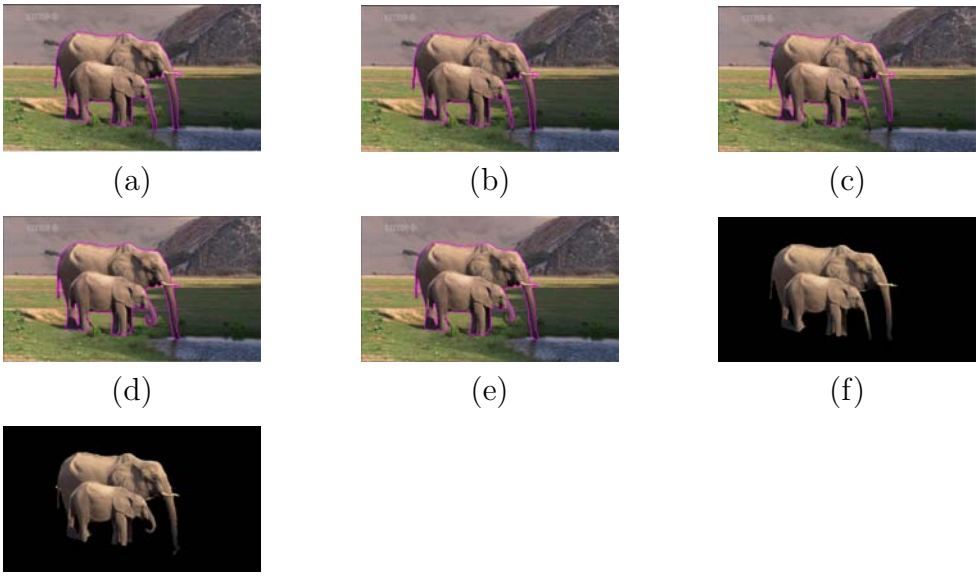


Figure 5.2: Comparison in video of elephant. (a) The initial segmented 1st frame, (b) the 2nd segmented frame with Video SnapCut, (c) the 3rd segmented frame with Video SnapCut, (d) user refined the 23rd as a new key frame, (e) segmented result of 25th frame propagated from (d), (f) the 3rd segmented frame with our method, (g) the 25th frame propagated from (a) with our method.

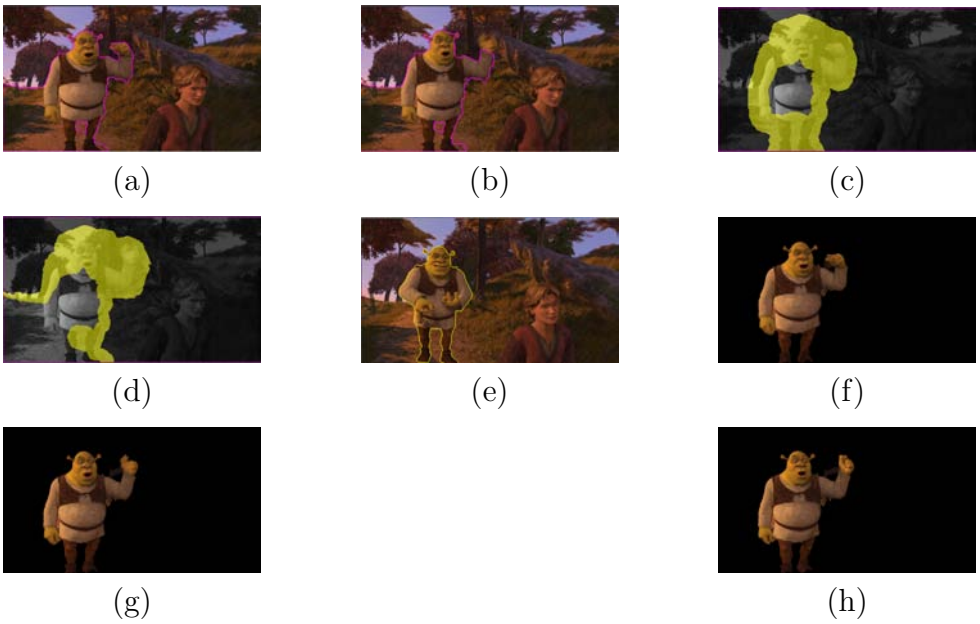
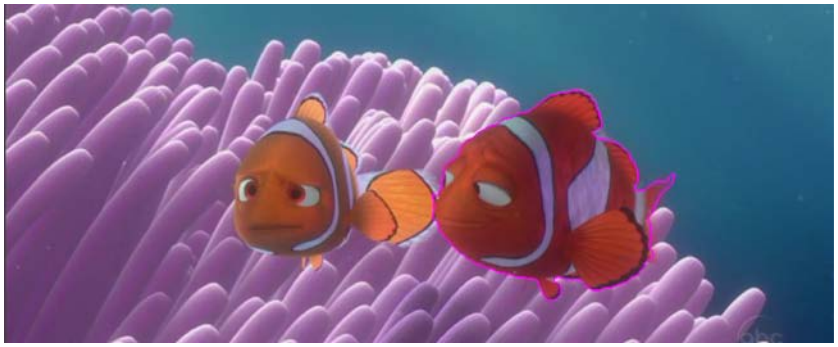
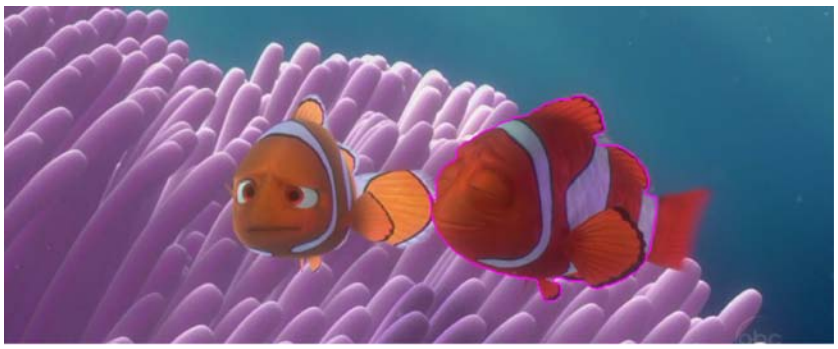


Figure 5.3: Comparison in video of Shrek. (a) The re-initialized segmented the 42nd frame, (b) the 43rd segmented result with Video SnapCut, (c) the search region of the 42nd frame with Video SnapCut, (d) the search region of the 42nd frame with Video SnapCut, (e) initialized 1st frame as key frame for propagation, (the segmentation results of (f)-(h) are propagated from (e)), (f) the 42nd segmented frame with our method, (g) the 43rd segmented frame with our method, (h) the 44th segmented frame with our method.



(a)



(b)

Figure 5.4: Failure case of Video SnapCut. (a) The 37th frame is segmented as a key frame, (b) propagated result of the 38th frame from (a).

(a)

(b)

Figure 5.5: Failure case of Video SnapCut. (a) The 81th frame is segmented as a key frame, (b) propagated result of the 82th frame from (a).

(a)

(b)

(c)

(d)

Figure 5.6: Comparison in video of cloth fitting. The figures from left to right are results from Video SnapCut and our program respectively by giving only the first ground truth as initialization without any user interaction. (a)∼(d) The results of 412th, 413th, 416th, and 418th frame respectively.

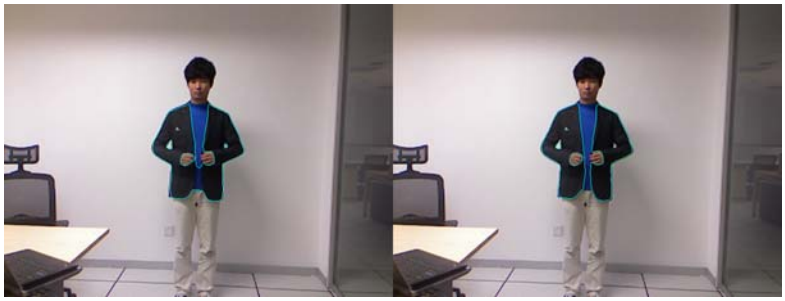
(a)

(b)

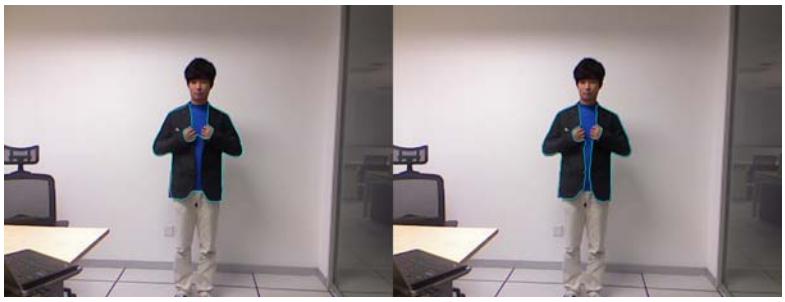
(c)

Figure 5.7: Comparison in video of cloth fitting. Shows the shortcoming of Video SnapCut that the left armpit part converges during the propagation. (a)∼(c) The results of 451th, 452th, and 453th frame respectively.

(a)



(b)

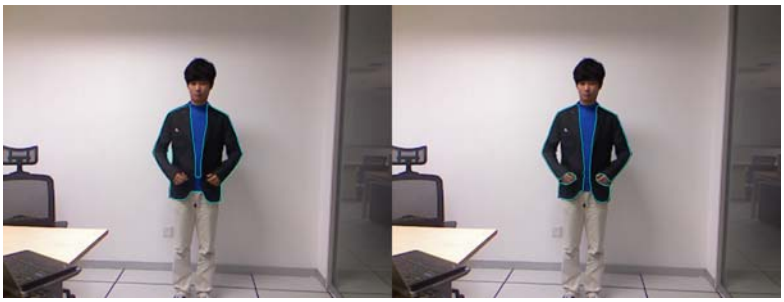


(c)

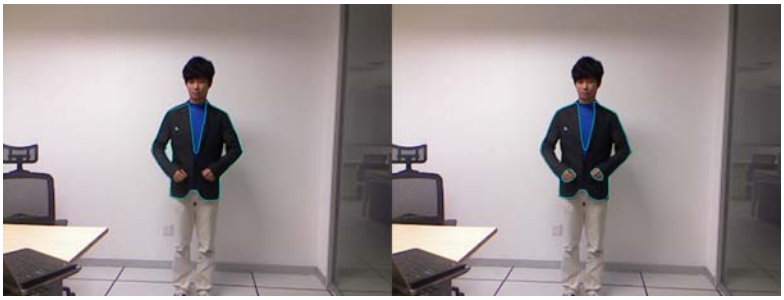
(d)

Figure 5.8: Comparison in video of cloth fitting. Shows the shortcoming of Video SnapCut that connects the black jacket incorrectly during the propagation. (a)∼(d) The results of 454th, 464th, 482th, and 515th frame respectively.

(a)



(b)



(c)

(d)

Figure 5.9: Comparison in video of cloth fitting. Shows Video SnapCut cuts off both of the hands during the propagation. (a)∼(d) The results of 459th, 450th, 595th, and 597th frame respectively.

Chapter 6

Conclusion

Video segmentation is a fundamentally important research topic and is widely used in many applications. In this thesis, existing video segmentation algorithms are reviewed and another kind of video segmentation related algorithm is proposed.

In chapter 3, we have presented a new framework for video segmentation that combines global/local classifiers and level set algorithm. Our approach firstly connect adjacent frames by optical flow to maintain color consistency. In the propagation phase, our predefined feature is also considered as a constraint to retain object topology between neighboring frames. The predefined feature evolves Gabor feature to retain textures welly. From the feature descriptor, shape prior is also utilized to improve the propagation performance and make the result more reasonable. In addition, our approach is particularly flexible to incorporate with other features or constraints. For example, depth information could be integrated into the feature descriptor as well to avoid results with unexpected object boundary.

In chapter 4, we have introduced an auto detection and correction system based on multiple features trained by Adaboost classifier. We also evaluated the influences of each feature from our experiments. Our approach advocates using less human interaction and getting expectable and qualitative results. In the meanwhile, our approach is generic and can be used with existing global/local classifiers and feature descriptors.

In chapter 5, evaluation and more experiment results are introduced. In the evaluation section, the limitations of our system and future study for further improvement are discussed in detail. In the experiment section, more results are shown to further compare the performance of our system and others. Although in most of cases our results are better and with less human interaction than current state-of-art algorithm, but there are still many limitations in our approach. Further research is still needed to be developed in the future.

Bibliography

[1] Su B, S Lu, and C L Tan. Blurred image region detection and classification. *ACM Multimedia*, November 2011.

[2] Xue Bai and Guillermo Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *IEEE Proc. of ICCV*, 2007.

[3] Xue Bai, Jue Wang, and Guillermo Sapiro. Dynamic color flow: A motion adaptive color model for object segmentation in video. In *IEEE Proc. of ECCV*, 2010.

[4] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 24(24):509–522, 2002.

[5] Yuri Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *IEEE Proc. of ICCV*, pages 105–112. IEEE, 2001.

[6] John F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, November 1986.

[7] Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. *Int'l Journal of Computer Vision*, 22(1):61–79, February 1997.

[8] E. W. Dijkstra. A note on two problems in connexion with graphs. *NUMERISCHE MATHEMATIK*, 1(1):269–271, 1959.

[9] Leo Grady, Thomas Schiwietz, Shmuel Aharon, and Rudiger Westermann. Random walks for interactive alpha-matting. In *IN PROCEEDINGS OF VIIP 2005*, pages 423–429, 2005.

[10] D.M. Greig, B.T. Porteous, and A.H. Seheult. Exact maximum a posteriori estimation for binary images. *Royal Statistical Society Series B*, 51:271V279, 1989.

[11] Etienne Grossmann. Adatree: Boosting a weak classifier into a decision tree. In *Proceedings of the Computer Vision and Pattern Recognition Workshop*, volume 6, pages 105–112. IEEE, June 2004.

[12] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *Int'l Journal of Computer Vision*, 1(4):321–331, 1988.

[13] Rolf Kohler, Michael Hirsch, Bernhard Scholkopf, and Stefan Harmeling. Improving alpha matting and motion blurred foreground estimation.

[14] Julia Lasserre and Christopher M. Bishop. Generative or discriminative? getting the best of both worlds. *BAYESIAN STATISTICS*, 8:3–24, 2007.

[15] Michael E. Leventon, W. Eric L. Grimson, Olivier Faugeras, and William M. Wells III. Level set based segmentation with intensity and curvature priors. In *Proceedings of the IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*, pages 4–11. IEEE, 2000.

[16] S. Z. Li. Markov random field models in computer vision, 1994.

[17] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. *ACM Trans. on Graph.*, 23(3):303–308, 2004.

[18] Yugang Liu and Yizhou Yu. Interactive image segmentation based on level sets of probabilities. *IEEE Transactions on Visualization and Computer Graphics*, 18(2):202–213, 2012.

[19] Ravikanth Malladi, James A. Sethian, and Baba C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17:158–175, 1995.

[20] B.S. Manjunath and W.Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(8):837–842, 1996.

[21] Eric N. Mortensen and William A. Barrett. Intelligent scissors for image composition. In *In Computer Graphics, SIGGRAPH Proceedings*, pages 191–198, 1995.

[22] Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes, 2001.

[23] Stanley Osher and James A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.

[24] Stanley J. Osher and Ronald P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, first edition, 2003.

[25] Nikos Paragios and Rachid Deriche. Geodesic active regions and level set methods for supervised texture segmentation. *Int'l Journal of Computer Vision*, 46(3):223–247, 2002.

[26] Shmuel Peleg and Moshe Ben-Ezra. Stereo panorama with a single camera, 1999.

[27] Danping Peng, Barry Merriman, Stanley Osher, Hongkai Zhao, and Myungjoo Kang. A pde-based fast local level set method. *Journal of Computational Physics*, 155(2):410–438, 1999.

[28] Christoph Rhemann, Carsten Rother, and Margrit Gelautz. Improving color modeling for alpha matting.

[29] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut - interactive foreground extraction using iterated graph cuts. *ACM Trans. on Graph.*, 23(3):309–314, August 2004.

[30] Robert E. Schapire. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear estimation and classification*, pages 149–171, 2002.

[31] Björn Scheuermann and Bodo Rosenhahn. Interactive image segmentation using level sets and dempster-shafer theory of evidence. In *Proc. of Scandinavian conference on Image analysis (SCIA)*, pages 656–665, 2011.

[32] J. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[33] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[34] Deqing Sun, Stefan Roth, and Michael J. Black. Secrets of optical flow estimation and their principles. In *IEEE Proc. of CVPR*, pages 2432–2439. IEEE, 2010.

[35] Ming syan Chen, Unsupervised Learning, M. s. Chen Ntu, M. s. Chen Ntu, and Qj Oi. k-means method.

[36] Zhenyu Wu and Richard Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1993.

[37] Bai Xue, Wang Jue, Simons David, and Sapiro Guillermo. Video snapcut: robust video object cutout using localized classifiers. *ACM Trans. on Graph.*, 28(3):1–11, 2009.

[38] Fan Zhong, Xueying Qin, and Qunsheng Peng. Transductive segmentation of live video with non-stationary background. In *IEEE Proc. of CVPR*, 2010.

[39] Songchun Zhu and Alan Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):884–900, September 1996.