If we knew what it was we were doing, it would not be called research, would it?

– Albert Einstein, 1879 - 1955.

University of Alberta

ROBUST REAL-TIME BI-LAYER VIDEO SEGMENTATION USING INFRARED VIDEO

by

Qiong Wu

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Fall 2008

Your file Votre référence
ISBN: 978-0-494-47442-6
Our file Notre référence
ISBN: 978-0-494-47442-6

Abstract

Tele-immersive systems aim at creating the illusion that all participants at a remote location are in the same virtual room. As a new emerging technique, it is faced with many challenging problems, one of which is automatic, real-time foreground-background video segmentation, or called bi-layer video segmentation.

This thesis presents a novel method for the bi-layer video segmentation problem based on the fusion of infrared and color video. The method improves on previous algorithms by making the system independent of changes in ambient lighting and dynamic background, and it achieves the goal of automatic video segmentation in real time. Two possibilities for utilizing infrared sources are explored, namely foreground illumination and background illumination. Experimental results show that segmentation based on infrared and color video is a promising technique for real-time segmentation.

Acknowledgements

I would like to thank all people who have helped and inspired me during my master study.

I especially want to thank my supervisor, Dr. Pierre Boulanger. His wide knowledge and his logical way of thinking have been of great value for me. I am deeply grateful to my co-supervisor, Dr. Walter F. Bischof, for his detailed and constructive comments, and for his important support throughout this work. They both offered me great freedom on what I want to do and encouraged me to try different ways of solving problems. Their understanding, encouragement and personal guidance have provided a good basis for the present thesis.

I also would like to thank HP laboratories and TRLabs who offered financial support throughout my graduate studies.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Many tasks in computer vision involve robust and accurate foreground-background video segmentation. In many applications, including, for example, surveillance, video conferencing, motion capture, or gesture analysis for human-computer interaction (HCI), video foreground objects have to be separated from the background scene before proceeding with further analysis.

One prime application of foreground-background video segmentation is video conferencing systems, where there is a need to remove the background and replace it with a different one. For example, Microsoft's research group at Cambridge has developed a desktop videoconference system, i2i [17], which includes background substitution. The immersive video conferencing system proposed by HP [5, 4], called the Coliseum project, aims at providing realism in communication (see Figure 1.1). The idea of this next-generation videoconference system is to generate a virtual meeting environment, into which all participants at remote location will be inserted. The meeting room is then rendered for each participant from their viewpoint. In these applications, segmentation of the foreground/participant from the scene becomes an essential and crucial step.

There are many techniques and mature products on the market that provide foreground-background image segmentation. However, bilayer video segmentation remains a challenging problem. In many respects, it has higher demands than image

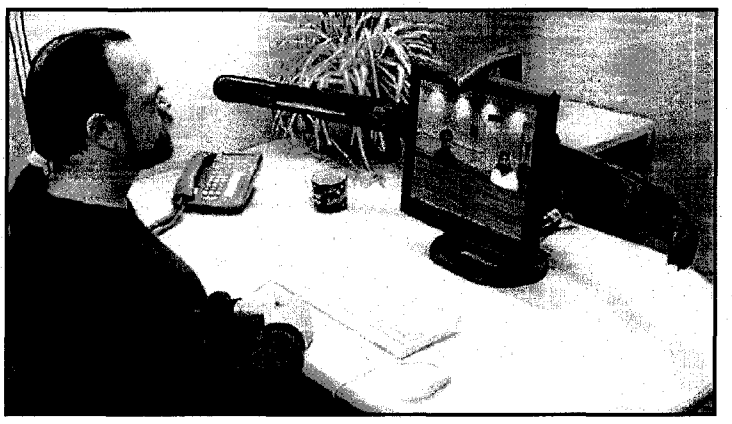


Figure 1.1: *HP* Coliseum immersive system [5].

Figure 1.2: iChat

segmentation, including real-time processing speed, consistency of segmentations between frames and minimal manual editing. Only a few video chat products on the market have successfully incorporated background substitution. For example, Apple introduced iChat in their release of Mac OS X version 10.3 "Panther" [1]. This product is capable of performing instant background substitution during video chatting. However, it only produces acceptable results in the situations where the background is static and uniformly colored, which largely constrains its usefulness in many common scenarios.

The challenges in foreground-background video segmentation, especially for

video conferencing system, include the following:

- Real-time processing: Segmentation should be performed at live processing speed to ensure the smooth progress of the video meeting/chatting.
- Robustness: Segmentation should be robust to environment changes, including illumination changes, camera shaking, dynamic backgrounds with presence of large moving objects and more.
- Automation: The foreground should be segmented automatically without user intervention.

We aim to solve this bi-layer video segmentation problem addressing the above challenges. Once the foreground objects have been detected and extracted by our system, they can be further processed for a variety of purposes.

1.2 Thesis Scope

This thesis explores mainly a new solution to the problem of bi-layer video segmentation by fusing infrared and color information. A powerful data acquisition unit has been developed using an optical technique that automatically gives synchronized infrared video and color video. We explore two possibilities for utilizing infrared illumination, foreground and background illumination. In the foreground illumination method, the resulting video sequences, infrared video and color video, are fed into the segmentation algorithm, Graph Cut or Relaxation Labeling, to optimize the segmentation results. The experimental results show the potential of both algorithms. Relaxation Labeling can utilize the fast parallel processing ability of the GPU and is more stable than Graph Cut in terms of changes in parameter values.

1.3 Contributions

The thesis makes two main contributions. First, we propose to adopt infrared information in the video segmentation system. The way we combined the infrared camera and the color camera automatically gives synchronized video sequences,

avoiding the disadvantages of conventional two-camera systems. Second, we explore two ways of utilizing IR illumination for bi-layer video segmentation. Third, in the foreground illumination method, we propose two different algorithms for video segmentation, achieving the goal of real-time processing speed, robustness and automatization. For Graph Cut, we simplify the graph construction by taking advantages of cue map. The efficiency of the Graph Cut algorithm is improved largely due to decreased number of graph nodes and edges. For Relaxation Labeling, it is a perfect parallel algorithm whose computation kernel can be implemented on a GPU, so that the segmentation can be done in microseconds.

1.4 Thesis Outline

Chapter 2 discusses the related work. Chapter 3 describes our proposed system by foreground IR illumination. In Chapter 4, we describe the other possibility of video segmentation by background IR illumination. In Chapter 5, we present our conclusions and future work.

Chapter 2

Literature Review

2.1 Video Segmentation

Video segmentation is an extension of image segmentation in the sense that video is a spatio-temporal sequence of images. Compared to the spatial nature of (static) images, video has both spatial and temporal characteristics. Segmenting a frame of video in the spatial domain is just like segmenting a static image [55]. In image segmentation the goal is to segment an image into spatially coherent regions, whereas in video segmentation the goal is to segment the image into spatio-temporally coherent regions [3]. That is, the segmentation of video sequence should be consistent among frames. Segmentation results usually serve for subsequent video/image analysis, such as object representation and description, feature measurement and even higher level tasks such as object classification, scene interpretation and coding purposes.

"Correct" video segmentation varies depending on the application. For example, for the purpose of achieving high video compression performance, the segmented objects might not be semantically meaningful to human observers [54]. Most of the current video segmentation research focuses on automatically segmenting images into semantically meaningful entities, such as people, ground, cars, sky or background etc. This object-based segmentation has wide applications in object-based media compression and coding (e.g., MPEG-4, codecs), visual-content retrieval (e.g., MPEG-7 related schemes) and object recognition.

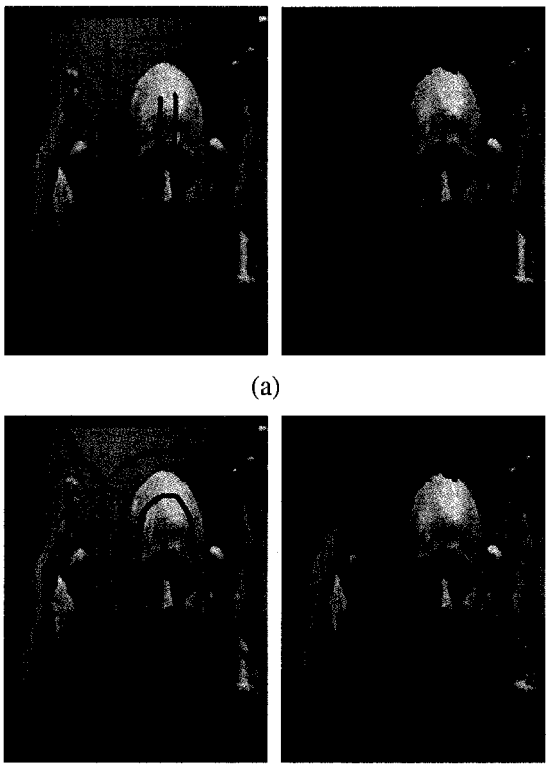
There is no general technical theory for video/image segmentation. Many research directions have been explored, but none of the developed segmentation algorithms are generally applicable to all images, and different algorithms are not equally suitable for particular applications [55]. As more and more segmentation algorithms are explored, classification of the various techniques for video segmentation becomes an essential task. Although there are a number of survey papers on image segmentation, very few discuss video segmentation, which only cover a small set of techniques developed and only focus on a specific application area. For example, Koprinska and Carrato [34] present a survey of techniques of temporal video segmentation. Temporal video segmentation divides a video stream into a set of meaningful and manageable shots, each of which is a sequence of frames and represented by selecting key frames. Zhang and Lu [54] present a review of motion-based segmentation algorithms. Correia and Pereira [15] present a classification criteria based on the application scenarios.

In video segmentation, more information can be used beyond color, edges and contrast information, which always play an important role in both video and image segmentation. A high-level understanding of image contents can also help to obtain automatic video segmentation. Acquiring such high-level cues, including frame difference, motion model, depth and stereo etc. at the pixel level or object level, often makes the video segmentation and tracking less challenging.

In this thesis, we focus on a specific application of video segmentation, video conferencing system, in which each frame is partitioned into two regions: foreground and background. The foreground (or object of interest) refers to the participant of the remote video meeting/chatting.

2.2 Previous Approaches

This section provides a general review of several state-of-the-art approaches in the field of bilayer video segmentation. The approaches can be classified into two general categories, segmentation by visual cues and segmentation by fusing visual with



(a)

(b)

Figure 2.1: Segmentation results by Interactive Graph Cuts [11].

other cues.

2.2.1 Segmentation by Visual Cues

Visual cues usually refer to the information observed in an image or a video frame, such as color, texture, contour and contrast information etc. It is unclear which cues to choose for a given segmentation problem and how to weight their importance, a problem known as the cue combination problem in computer vision [43]. Traditionally, supervised learning is used to label the data in an image and a classifier is trained on the labeled data to classify unlabeled pixels. Unsupervised segmentation methods can group pixels into regions of similar texture, but not able to segment foreground and background regions.

Recently, interactive segmentation techniques exploiting color/intensity and contrast cues have been demonstrated to be very effective for static images, as demonstrated in [44]. Usually color priors for the foreground and background are used to

classify pixels in the unknown area, and inaccurate segmentation caused by color noise is compensated by the contrast cues, which force the segmentation boundaries to be located where the contrast is locally maximum. Some techniques such as Active Contours [30] also use smoothness constraints to restrict the segmentation boundary. Unfortunately, all these methods require manual user input to provide segmentation cues. For example, Intelligent Scissors [38] and Live Wire [20] require boundary cues; the user has to accurately select some pixels the segmentation boundary should pass through. Then the segmentation boundary is determined by computing the "shortest path" between the marked pixels. In [11], the user needs to mark several foreground and background pixels, providing color cues on what the user intends to segment. In "GrabCut" [44], the user drags a rectangle around the desired foreground object. After initial segmentation, more editing may be needed to obtain the desired result.

There are two main disadvantages caused by manual input. First, the segmentation results seem to be very sensitive to user initialization. For example, in [11], it is important for the segmentation result that the user selects the most characteristic color region of the foreground and background. Figure 2.1 shows segmentation results obtained with the algorithm presented in [11]. We used the Matlab code [24] by Mohit Gupta and Krishnan Ramnath. The user input is shown in the left subimages of (a) and (b). The red brush stokes denote foreground seeds and the blue brush stokes denote background seeds. We can see from (a) and (b) that slightly different initial input results in very different segmentation result. Second, segmentation is beyond the capabilities of fully automatic methods. This makes them not applicable to video segmentation since manual editing of each frame is too expensive and impossible for video conferencing application.

2.2.2 Segmentation by Fusing Visual with Other Cues

In order to obtain the fully automatic segmentation, a more robust approach is required that explores fusions of several cues. Recently, many researchers have investigated the fusion of background model, motion or stereo with color and contrast,

as described in the following sections.

Background Model Based Segmentation

Several segmentation systems rely on prior knowledge about the background scene, which is represented by a histogram-based background model for gray images. The foreground is considered a region (in each new frame) that differs significantly from the background. This type of approach is commonly referred to as background subtraction, referring to the simple technique of subtracting the background model from a new frame and thresholding the result to find the foreground region.

The simplest background subtraction technique is based on the assumption that the background, as viewed by the camera, is static. In this simplified case, the background model is a background picture captured apriori. The foreground is detected by subtracting the current frame from the background picture and thresholding the result [35]. This process can be represented as: $|frame_i - background| > Th$, where i is the frame index and Th is a predefined threshold. This method obviously fails if the background is dynamic. More advanced techniques in this category use an adaptive background model to solve problems caused by regularly time-varying backgrounds, such as water waves, moving clouds, trees waving in the wind and so on. In the Pfinder system [52], each background pixel is modeled with the Gaussian described by a mean color value and a full covariance matrix, and the background model is continually updated. In [23, 29, 47], each pixel is modeled by a mixture of K Gaussians,

$$P(I_t) = \sum_{i=1}^{K} \omega_{i,t} \eta(I_t; \mu_{i,t}, \sum_{i,t})$$

where $\mu_{i,t}$ is the expectation of the ith Gaussian at time t, $\sum_{i,t} = \sigma_{i,t}^2 I$ is the covariance matrix. $\omega_{i,t}$, $\mu_{i,t}$ and $\sigma_{i,t}^2$ are updated according to whether I_t matches derived current Gaussian components. These methods can learn and adapt a background model, but they are based on the assumption that the color distributions of the foreground and background pixels are well separated.

There are several problems in developing a background subtraction algorithm.

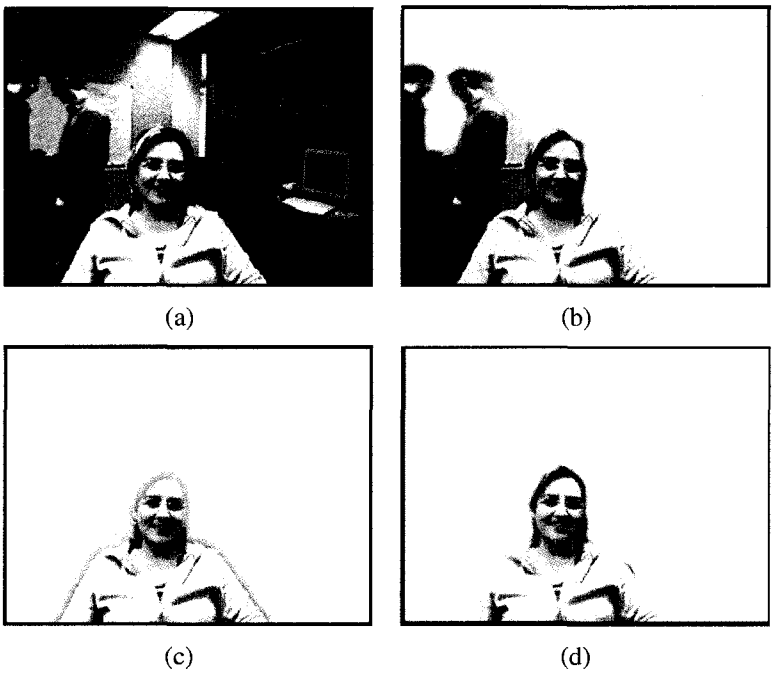


Figure 2.2: Comparison segmentation results (reproduced from [53]). (a) Original, (b) Result from motion-based segmentation in [18], where the large moving objects in the background are falsely classified as the foreground, (c) Result from motion-based segmentation in [53], where the moving background parts are removed, but some foreground boundary parts are missing, (d) Result from stereo-based segmentation in [32], with more accurate foreground boundaries.

First, the threshold is critical, resulting in a lot of noise if selected too low and suppressing significant changes if selected too high. Second, background subtraction techniques fail in the presence of large moving objects in the background. Third, they are very sensitive to illumination changes and image noise. Finally, they cannot discriminate foreground from background, if there is lack of color differences.

Motion-Based Segmentation

Traditional motion-based segmentation methods usually employ motion information only. Motions are typically represented as a 2D optic flow [6, 42, 46, 41, 49, 26]. The optic flow of a pixel is a motion vector calculated between a pixel in one frame and its corresponding pixel in the following frame. Motion-based segmentation finds groups of pixels in two or more frames where each group is a smooth motion field, or say, optic flow field. Motion segmentation alone cannot solve the segmentation problem of static scenes. Further, due to occlusion and disocclusion,

optic flow is not reliable at the boundaries of moving objects [31]. Finally, the estimation of per-pixel optic flow is computationally expensive because it involves extensive search for finding matching/corresponding pixels.

Very often the foreground object to be segmented has inconsistent motion and is often stationary, so adding other feature information can overcome this problem. The earliest work, to our knowledge, on combining motion with other image features for segmentation has been explored by Thompson [48]. Images were segmented based on regions with similar intensity and optic-flow values. The resulting, over-segmented regions were then merged using a heuristic approach. Black [10] combined intensity and motion for segmenting images based on Markov Random Fields. Tekalp et al. [2] presented a method for segmenting images separately by motion and color, and color segments were merged by regions belonging to the same motion region. All these methods segment each video frame into several layers for the purpose of video compression rather than for foreground-background segmentation; the segmented layers usually have holes and are not accurate at the boundaries [31]. Finally, with extensive optic flow computation, it is hard to achieve real-time processing speed.

Recent research on bi-layer video segmentation based on fusing motion, color and contrast cues has produced promising results. Criminisi et al. [18] present an algorithm that probabilistically fuses motion, color and contrast cues together with spatial and temporal priors. The main contribution of their paper is removing the need of computing pixel velocities, which is required for optic flow computation. However, it cannot cope with a large moving background or a stationary foreground. Yin et al. [53] improved motion-based segmentation results in the presence of large background motion and a stationary foreground. They showed a comparison result between their algorithm and the algorithm in [18] as well as a stereo-based segmentation algorithm [32], as shown in Figure 2.2. One can see that the motion-based algorithms can not beat stereo-based algorithms, which will be introduced in the next section. In addition, none of motion-based segmentation algorithms claimed real-time processing speed. In [53], a processing speed of between 1.2 and 2.7 fps

was reported in a Matlab implementation.

Stereo-Based Segmentation

Stereo vision usually refers to the problem of inferring the 3D structure of a scene from two or more images taken from different viewpoints. A two-camera stereo setup imitates the human vision system, which has two eyes looking into the scene. The major task in stereo is to exploit the correlation between stereo images, to compute disparity between corresponding image pixels, and to infer a depth map of image pixels from the disparity information. Stereo vision involves computing the depth of each image pixel, and thus can be applied to bilayer video segmentation due to the fact that the foreground is usually located on a depth plane with smaller depth than that of the background.

Conventional stereo algorithms have proven useful in depth computation [40, 16]. Recent research on addressing occlusion [21, 7, 33, 19] and depth discontinuities [9] are of particular interest for extracting a foreground object from the background. Williams [51] proposed a Gaussian process framework for the bi-layer segmentation of a scene, given a stereo pair of images, by estimating pixel-wise disparity. His approach classifies all pixels into three categories: foreground, background and occlusion, based on a disparity model used as the pixel classifier, which roughly estimates the disparity value for foreground (0.8D), background (0.2D) and occlusion (0.5D), where D is the maximum disparity value in the image. When the disparity model is violated, the application is bound to be incorrect. Further, Williams uses NSSD (normalized sum of squared differences) for matching when evaluating the correlation of pixels. Such depth computation from stereo is not only computationally expensive, it is also not accurate and not robust over low-textured or homogeneous regions.

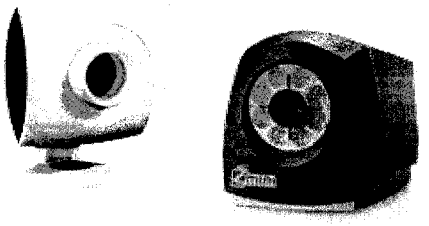
In our opinion, the most compelling real-time algorithm on bilayer video segmentation, which fuses stereo, color and contrast, was proposed by Kolmogorov et al. [32]. They investigated two different real-time algorithms to probabilistically fuse stereo, color and contrast cues, Layered Dynamic Programming (LDP) and

(a) (b) (c)

Figure 2.3: i2i stereo web-cam

Layered Graph Cut (LGC). In LDP, a Gibbs energy function $E(z, d, x; \Theta, \Phi)$ is defined explicitly incorporating priors on stereo disparity Φ and color/contrast information Θ. The segmentation is obtained by minimizing the energy independently over each epipolar line with a dynamic programming algorithm. In LGC, an energy function $E(z, x; \Theta)$ is globally minimized over an image, in which stereo disparity does not appear explicitly. Instead, the stereo match likelihoods are marginalized to compute a probability over the foreground and background hypotheses. The segmentation is determined by an expanded form of graph cuts. In both algorithms, stereo likelihoods are still measured by NSSD, which is computationally expensive. Color likelihood is modeled by Gaussian mixtures in RGB color space as in "GrabCut" [44]. The contrast term is computed by neighboring pixel-pairs in the cyclopean image. A real-time implementation of LGC was reported in their paper at around 10 frames per second for a 320×240 image on a 3-GHZ Pentium desktop machine.

Compared to other video segmentation methods, stereo-based segmentation needs a two-camera binocular stereo video setup, as shown in Figure 2.3. Stereo cameras can add additional computation for calibration and synchronization, which are both important for stereo match computation. Camera calibration is needed to rectify any pair of images to a "parallel camera geometry" so that two corresponding points line on the same horizontal line, called epipolar line, in the two images. The epipolar line constraints can reduce the search of corresponding points to a 1D search problem. Synchronization is necessary to ensure that the video sequences generated by two cameras are using the same timing, so that two corresponding frames



(a) (b)

Figure 2.4: Zcam (a) The new ZCam (b) Mini camera

are generated at the same time. The synchronization can be completed by either special digital circuit design, e.g. the synchronized USB 2.0 stereo web-cam developed by Microsoft research at Cambridge, as shown in Figure 2.3(c), or over the network. Nevertheless, both calibration and synchronization add extra time to the stereo computation kernel.

Time-Of-Flight Camera Based Segmentation

In general, time-of-flight (TOF) cameras are able to obtain depth information based on the time-of-flight principle. They consist of an amplitude-modulated infrared light source and obtain depth information by measuring the time taken by infrared light to travel to all objects in the scene and then return back to the camera. They can produce either a depth value for each captured pixel or a gray-scale depth map with an intensity proportional to the time of flight of the light reflected by a distant object [45]. A more detailed description of the time-of-flight principle can be found in [36, 39, 22]. Compared to the stereo-based methods, TOF cameras obtain depth information directly, avoiding the inconvenient and expensive computation, e.g. NSSD, for stereo matching.

In order to apply TOF cameras to real-time bilayer video segmentation, many researchers combine a TOF camera with a 2D regular web camera to produce both RGB and depth signals. Unfortunately, as in [45], the 2D camera image and the depth image do not correlate directly, caused by significant differences in camera lens and image characteristics of two cameras. Other problems discussed in [45]

include resampling if two cameras have different resolution, synchronization if two cameras have different frame rates, and calibration in order to map color image with depth image.

Today, several manufacturers have produced an integrated depth-video camera, e.g. the ZCam [28] shown in Figure 2.4, which is capable of producing perfectly synchronized RGB and depth signals. Such 3D cameras have many desirable properties, including robustness against illumination changes, fast and easy processing. Unfortunately these sensors are very expensive and difficult to integrate with normal video technology.

Chapter 3

Segmentation by Foreground Illumination

Traditional bi-layer segmentation methods based on the color and contrast information alone can only achieve desirable results with user interaction. In order to obtain automatic foreground-background video segmentation, one needs to infer more information than what is given in the original video. While some compelling algorithms have been proposed on combining stereo or motion information with original color video, they all have problems with illumination changes, with large moving object in the background, with high computational cost and so on. In this thesis, we seek a solution to bi-layer video segmentation problem by fusing infrared, color and contrast information. The challenges solved by our proposed system include real-time processing speed, automatic segmentation and robustness to illumination change and dynamic background.

We designed a data acquisition unit, involving an IR camera and a color camera, to automatically generate synchronized IR video and color video. An IR illuminator is used as an imperceptible light source besides ambient light in the visible spectrum. We investigate two ways of employing IR information by illuminating different parts of the scene with the IR illuminator: foreground IR illumination and background IR illumination. In the foreground IR illumination, the IR illuminator is put in front of the foreground/object, and it lights the foreground. In the background IR illumination, the IR illuminator is put behind the foreground and lights the background area. In either cases, the resulting video sequences will be fed to the

segmentation algorithms to complete foreground segmentation. Two segmentation algorithms are explored in this thesis on fusing infrared, color and contrast information: Graph Cut (GC) and Contrast Preserving Relaxation Labeling (CPRL). Both algorithms are suited for real-time implementation. Given the fact that CPRL can be computed by a parallel algorithm, its computation kernel can be implemented in the GPU with higher computation speed. The claims are verified in the experimental section.

In this Chapter we introduce how to achieve bi-layer video segmentation by foreground IR illumination. The background IR illumination method will be presented in the next Chapter.

3.1 Data Acquisition Unit

Figure 3.1 illustrates the structure of our data acquisition unit. An IR projector at 850nm is used as an imperceptible light source to illuminate the scene, besides the ambient light in the visible spectrum. We chose a ELMO CN42H three-CCD color camera (referred as color camera below) and a Sony XC-EI50 near IR CCD camera (referred as IR camera below) to capture the scene. The color camera is chosen to produce color video, and the IR camera sensitive at 400nm to 1000nm, is used for recording the IR video sequences. Since the XC-EI50 is also sensitive to visible light, an interference filter with a narrow-band (pass 850nm$\pm$25nm) is used to reject all light except the one produced by the IR illuminator. The color camera used in our system has no response to IR light, which is not true for all color video cameras.

As shown in Figure 3.2, the color and IR cameras are mounted at perpendicular angles on a cube beam splitter. A beam splitter is placed between the two cameras at a 45 degree angle, partitioning the incoming light into two perpendicular paths. Half of the light, including the IR light and ambient light, is reflected to the color camera, and the other half is transmitted to the IR camera. In the foreground IR illumination method, the foreground/object is in the field of IR illumination so that

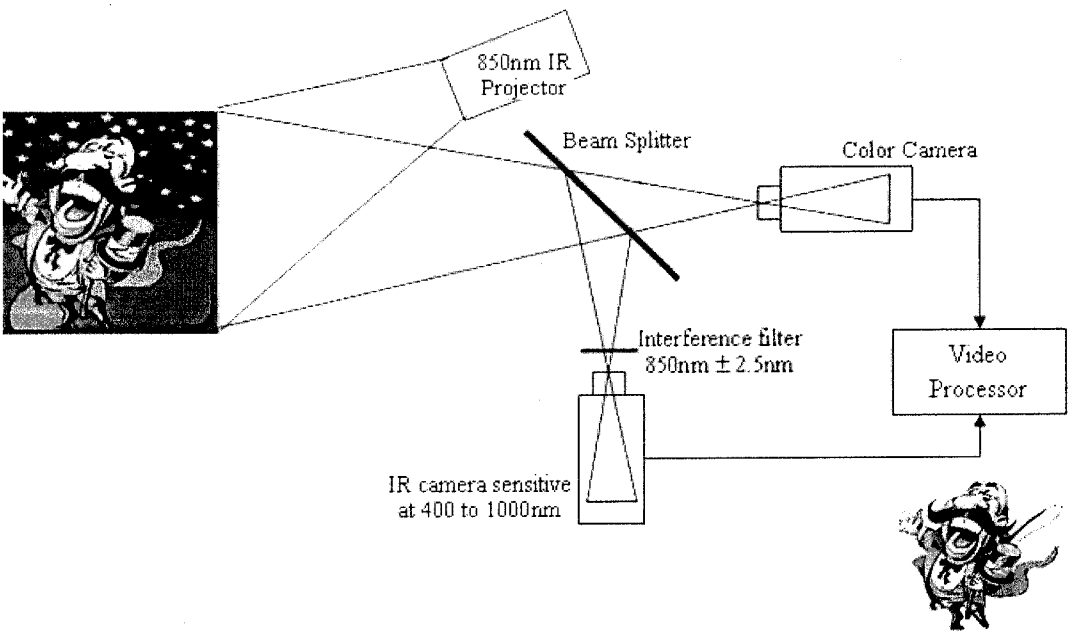


Figure 3.1: Structure of the data acquisition unit

the foreground can be illuminated by the IR illuminator.

The color and IR cameras have the same lens, and therefore yield the same video-resolution of 365×480 pixel images, which avoids the resampling computation. The two cameras are synchronized using a genlock mechanism, which is a technique to temporally synchronize video signals from different sources. The IR image is a mirror version of the color image due to the optical beam splitter. Also, the use of a beam splitter guarantees that the color image and IR image will be coplanar and coaxial because they have the same center of projection, so complex calibration is avoided to align images. A MacBeth calibration pattern [37], as shown in Figure 3.3, is used in our system to register two images. After selecting four corresponding point pairs in the IR and color images, the offset caused by the small differences in the position of the camera-lens centers can be estimated by computing the linear translation among these point pairs. This step can be done automatically by recognizing corresponding points of the MacBeth calibration pat-

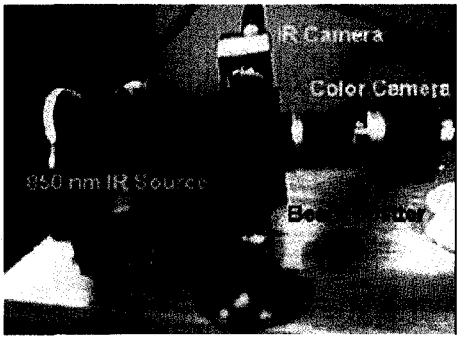


Figure 3.2: Data acquisition unit

Figure 3.3: MacBeth calibration pattern

tern.

The advantage of our system is that it gives full control of the illumination process because of the attributes of IR light, which is independent of ambient lighting in the visible spectrum. It solves the problem that video segmentation is sensitive to illumination changes. Furthermore, the inconvenient resampling and synchronization problem of conventional two-camera system in stereo based segmentation methods are also avoided, as described in the previous paragraph.

The basic outline of the process of our segmentation system is illustrated in Figure 3.4. The color image generated by the color camera will look the same as the view scene. The IR image is a mirror version of the color image due to the beam splitter and it can be easily registered with a simple image transposition.

Assume that after image registration, each IR image is an array $I = (I_1 ... I_i ... I_{|P|})$,

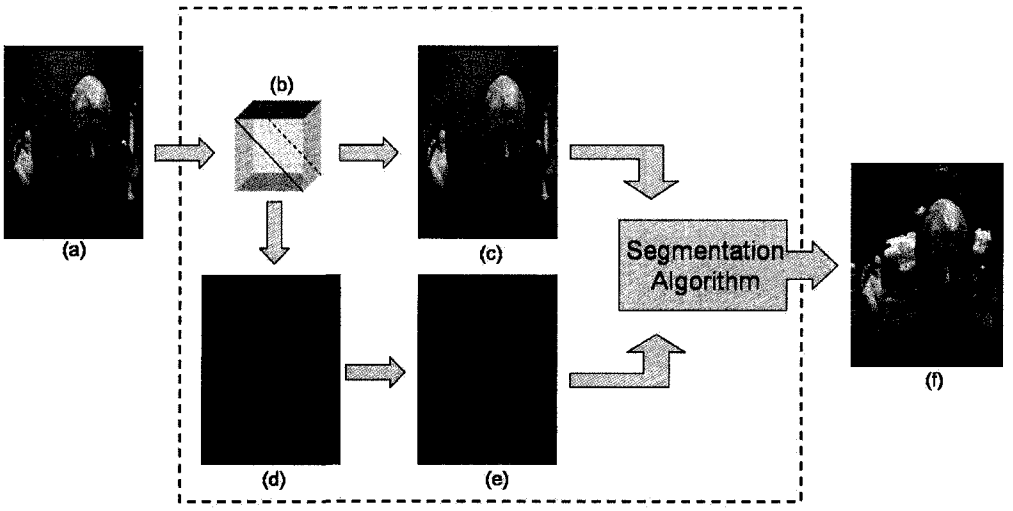


Figure 3.4: Process of operation (a) The view scene. IR light and normal illumination enters the camera. (b) A cube beam splitter splits the light into two perpendicular paths. (c) The color image generated by color camera. (d) The IR image generated by IR camera tuned to 850nm, with an interference filter (850nm$\pm$25nm) rejecting all light except IR light. (e) The registered IR image. (f) New composed image with background substitution.

and the color image is an array $Z = (Z_1...Z_i...Z_{|P|})$, where I_i is a gray scale value, Z_i is a RGB color vector, and P is the pixel set of an image. For the foreground-background segmentation, we seek a binary label vector $f = (f_1, f_2...f_{|P|})$, where $f_i \in \{0, 1\}$, with 1 denoting the foreground label and 0 the background label. The foreground is segmented as described below:

1. Cue Map Initialization. Information provided by the IR image allows estimation of the foreground, background and an unknown region. Further, it provides clues for building a Gaussian Mixture Model (GMM) for the foreground and background color spaces respectively.

2. Image Segmentation. For each video frame, the initialized cue map and the color image are fed to the segmentation algorithm to complete foreground segmentation. Two segmentation algorithms are investigated in our work:

   - Graph Cut (GC). Construct a graph for the color image with N-links reflecting contrast information and T-links reflecting color information. Each pixel in the unknown area is assigned a probability value in the maximum a posteriori sense given foreground and background GMM. The segmentation is determined by finding max-flow/min-cut of the constructed graph.

   - Contrast Preserving Relaxation Labeling (CPRL). Each pixel in the unknown area is assigned a probability vector with the first element representing the probability of belonging to the foreground and the second element the probability of belonging to the background. The probability vector is updated based on the neighborhood linear constraints. The segmentation is determined by the convergence of probability after a sufficient number of iterations.

3.2 Cue Map Initialization

A cue map can be defined according to the attributes of the IR image to improve the segmentation result. The cue map defined in our system is called the "pentamap".

The pentamap and color image are fed together to the segmentation algorithm to complete the foreground segmentation. We will present how to initialize a pentamap from an IR image in this section.

The attributes of the IR image provides clue for predicting foreground and background areas. The IR image is a gray scale image, in which brighter parts indicate the foreground (illuminated by IR source). Some foreground parts may be missing as they fall in the shadow area of IR illumination field. Missing foreground parts, however, must be within a certain distance from the illuminated parts as an integrated foreground object. Assuming this distance is τ (in numbers of pixels), one can predict that any dark area outside of this distance belongs to the background. The value of τ increases with the distance of the object to the IR source. That is, the further the object from the IR source, the more foreground parts are missing. Suppose the effective distance for the IR source is D_{max}, $\tau = \tau_{max} = f(D_{max})$, and the value of τ does not change during the video capture if the user does not change the camera and IR illumination configuration.

An estimate of the foreground area, which we call MASK, can be found from the IR image by simple thresholding: $MASK = \{p \in P | I_p \geq T\}$. A conventional trimap can be easily defined in terms of $MASK$ and τ. In many image segmentation algorithms, e.g. GrabCut [44], the input is a trimap. A trimap T partitions an image into three regions: foreground T_F, background T_B and unknown region T_U. Since we are sure that MASK belongs to the foreground, we can say $T_F = MASK$. In addition, since we know any area outside of distance τ from MASK belongs to the background, we can represent the predicted background by applying a dilation morphological operation $T_B = P - MASK.dilation(\tau)$. The remaining area is unknown, so $T_U = P - T_F - T_B = MASK.dilation(\tau) - MASK$.

In the traditional image segmentation algorithm, the trimap is used to build color GMMs for the foreground and background, which are derived from T_F and T_B respectively. Here, we propose the idea of a pentamap, which can derive more reliable color GMMs, leading to more accurate segmentation results.

We define a pentamap as follows:

Definition 1 *A pentamap Q partitions an image into five regions, certain foreground Q_{CF}, certain background Q_{CB}, local foreground Q_{LF}, local background Q_{LB} and unknown Q_U, represented as $Q : P \rightarrow \{Q_{CF}, Q_{CB}, Q_{LF}, Q_{LB}, Q_U\}$, with*

$$Q_{CF} = \{p|p \in MASK.erosion(s)\}$$
$$Q_{LF} = \{p|p \in MASK - Q_{CF}\}$$
$$Q_{CB} = \{p|p \in \sim (MASK.dilation(\tau + s))\}$$
$$Q_{LB} = \{p|p \in MASK.dilation(\tau + s) - MASK.dilation(\tau)\}$$
$$Q_U = \{p|p \in P - Q_{CF} - Q_{CB} - Q_{LF} - Q_{LB}\}$$

where τ and s are in terms of number of pixels.

Given these definition, one can transform a pentamap into a trimap as follows:

- $Q_{CF} + Q_{LF} = T_F$
- $Q_{CB} + Q_{LB} = T_B$
- $Q_U = T_U$

Figure 3.5 shows an example of a MASK, trimap and pentamap. In the pentamap model, $Q_{LF}(Q_{LB})$ is a narrow strip of width s that is separated from $T_F(T_B)$. In our approach, the color GMM of the foreground is derived from Q_{LF} rather than T_F (similarly, the color GMM of the background is derived from Q_{LB} rather than T_B), given that it is reasonable to assume that color in the unknown region is consistent with the color in its neighborhood regions rather than the whole map. That is, the color in Q_U should be consistent with the color of Q_{LF} or Q_{LB} rather than the whole region of T_F and T_B. In the experimental section, we show that our pentamap performs better than the trimap.

A pentamap can be automatically initialized from the IR image. The threshold T can be fixed since the intensity of the IR image does not change as the ambient light

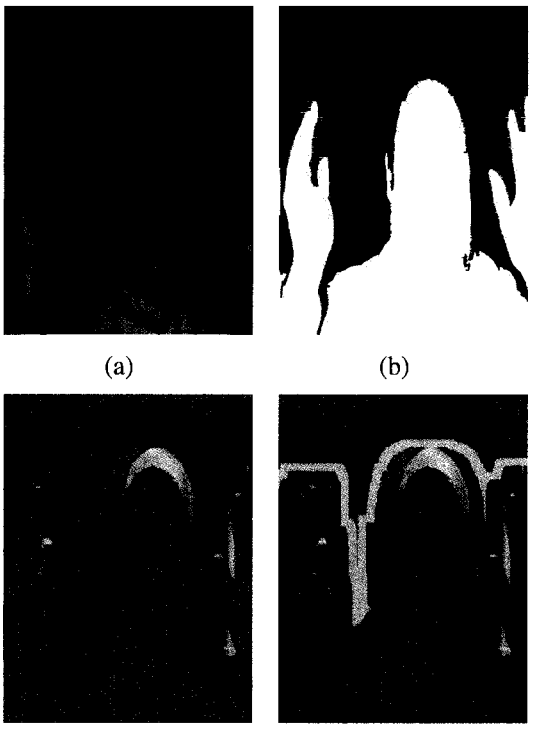


Figure 3.5: Pentamap initialization. (a) Registered IR image. (b) Foreground MASK. (c) Trimap with red=T_F, green=T_B and remaining=T_U. (d) Pentamap with red=Q_{CF}, green=Q_{CB}, blue=Q_{LF}, pink=Q_{LB}, and the remaining area=Q_U.

changes. The value of τ is determined by the configuration of the IR camera and the distance between the foreground object and the IR source, and it does not change during the video capture. In our experiments, we used the following parameter values: $T = 0.004$ and $\tau = 55$. The value of s (in the definition of Q_{LF} and Q_{LB} above) does not change the segmentation result much if $s \in [15, 25]$.

As in GrabCut [44], we use a Gaussian Mixture Model (GMM) to represent the foreground and background color spaces, which are derived from Q_{LF} and Q_{LB}. Each GMM, one for foreground and one for background, is a Gaussian mixture with M components (M=10 gives the best performance in the experiments), which can be interpreted as the number of RGB color clusters. An example of GMM representing foreground and background color models can be found in [44]. A vector $K = \{K_{1,f_1}...K_{i,f_i}...K_{|P|,f_{|P|}}\}$ is introduced to assign a GMM component to each pixel in the image, $K_{i,f_i} \in \{1...M\}$ representing a GMM component with mean μ_{i,f_i} and covariance matrix $\sum_{i,f_i} = \sigma^2_{i,f_i}$. The component is either from the foreground GMM or background GMM according to f_i, which is the label of the pixel. For each pixel in the unknown area Q_U, the probability that it belongs to the foreground is defined in the maximum a posteriori (MAP) sense, as in (3.1):

$$\forall p \in Q_U, Pr(p|f_p = 1) = \max_{K_{p,1}=1\cdots M} Pr(Z_p|K_{p,f_p=1}) \tag{3.1}$$

where $Pr(\cdot)$ is measured by a Gaussian probability distribution, so that:

$$Pr(Z_p|K_{p,f_p=1}) = |(2\pi)^d \sum_{p,1}|^{\frac{1}{2}} exp\{-\frac{1}{2}(Z_p - \mu_{p,1})^T \sum_{p,1}^{-1}(Z_p - \mu_{p,1})\} \tag{3.2}$$

d is the dimension of the measurement vector Z_p, which is the RGB value of pixel p. Similarly, the probability that a pixel belongs to the background is defined in (3.3) and (3.4):

$$\forall p \in Q_U, Pr(p|f_p = 0) = \max_{K_{p,0}=1\cdots M} Pr(Z_p|K_{p,f_p=0}) \tag{3.3}$$

$$Pr(Z_p|K_{p,f_p=0}) = |(2\pi)^d \sum_{p,0}|^{\frac{1}{2}} exp\{-\frac{1}{2}(Z_p - \mu_{p,0})^T \sum_{p,0}^{-1}(Z_p - \mu_{p,0})\} \tag{3.4}$$

3.3 Graph Cut

Boykov et al. [11, 12, 13] proposed the graph cut-based segmentation algorithm to perform various segmentation tasks. In this algorithm, a graph is constructed according to an energy function derived from the original image. The image segmentation is obtained by minimizing the energy function corresponding to the min-cut/max-flow of the constructed graph.

A good segmentation of an image should correspond to the minimum of an energy function in the form of:

$$E(f) = D(f) + V(f) \tag{3.5}$$

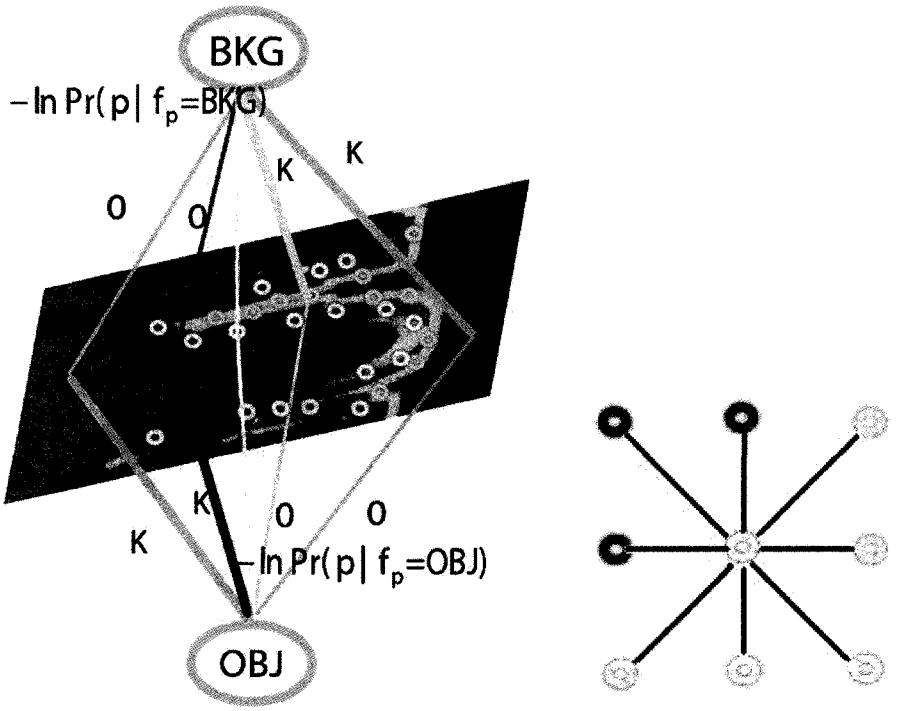
$D(f)$ and $V(f)$ are defined as:

$$D(f) = \lambda \sum_{p \in P} D_p(f_p) \tag{3.6}$$

$$V(f) = \sum_{\{p,q\} \in \mathbb{N}} [f_p \neq f_q] V_{p,q}(f_p, f_q) \tag{3.7}$$

where λ specifies the relative importance of $D(f)$, $[\cdot]$ is a delta function that gives 1 for $f_p \neq f_q$ and 0 otherwise, and $\mathbb{N}$ is the set of all pairs of neighboring pixels. $D(f)$, called the data term, gives a penalty for assigning different labels to each pixel, and $V(f)$, called the smoothness term, corresponds to the penalty of the edge/contrast information.

An undirected graph $G(V, E)$ is constructed according to the energy function. Each node in the graph corresponds to a pixel in the original image. There are two additional terminal nodes, one for the object, called OBJ, and the other for the background, called BKG. The weight of edges connecting nodes and terminals are given by the data term, and the weight of edges connecting neighborhood nodes are given by the smoothness term. The segmentation of the image is found by solving the min-cut/max-flow problem on the graph G, which should correspond to the minimum value of the energy function.



(a) An example of T-links

(b) An example of N-links

Figure 3.6: Graph construction from a pentamap. (a) T-links. Black node in red area=V_{CF}, black node in green area=V_{CB}, blue nodes=V_{LF} corresponding to each pixel in Q_{LF}, pink nodes=V_{LB} corresponding to each pixel in Q_{LB}, yellow nodes=V_U corresponding to each pixel in Q_U. Edges are constructed between each graph node V and terminal nodes, with edge weights reflecting the probability of classifying each node to each terminal. (b) N-links. Edges are constructed between node pairs (V_i, V_j), $V_i \in V_{LF}/V_{LB}/V_U$, $V_j \in V_U$, $\{i, j\} \in \mathbb{N}$ and $i \neq j$.

The graph constructed in our segmentation system is simplified because of the pentamap. In this section, we introduce how to construct a graph from an image to integrate the Graph Cut segmentation algorithm in our system.

One advantage of the pentamap is that it simplifies the complexity of the graph construction and thus improves the efficiency of the Graph Cut algorithm. An example of graph construction in our method is shown in Figure 3.6. For each frame to be segmented, an undirected graph $G = (V, E)$ is defined with a set of nodes V and a set of undirected edge E that connect nodes. Nodes are defined as follows:

- Terminal nodes: OBJ and BKG, representing the foreground and background respectively.

Table 3.1: Edge weight table

Edge Weight Table		
Edge	*Weight*	*For*
(V_i, OBJ)	K	$V_i \in V_{CF}/V_{LF}$
	0	$V_i \in V_{CB}/V_{LB}$
	$-\ln Pr(p\|f_p = 0) * \lambda$	$V_i \in V_U$
(V_i, BKG)	K	$V_i \in V_{CB}/V_{LB}$
	0	$V_i \in V_{CF}/V_{LF}$
	$-\ln Pr(p\|f_p = 1) * \lambda$	$V_i \in V_U$
(V_i, V_j)	$\alpha * \exp(-\|Z_i - Z_j\|^2/\beta)$ β is the expectation $2 \parallel Z_i - Z_j \parallel$	$\{i,j\} \in \mathbb{N}, i \neq j,$ $V_i \in V_{LF}/V_{LB}/V_U,$ $V_j \in V_U$

- Graph nodes V are classified into two categories:

  Certain Nodes: V_{CF} and V_{CB} correspond to Q_{CF} and Q_{CB} area respectively.

  Cut Nodes: $V_{cut} = V - V_{CF} - V_{CB} = \{V_{LF}, V_{LB}, V_U\}$. $V_i \in V_{LF}$ corresponds to each pixel in Q_{LF}, $V_i \in V_{LB}$ corresponds to each pixel in Q_{LB}, and $V_i \in V_U$ corresponds to each pixel in Q_U.

Edges are added between node pairs in the following cases:

- $(V_i, \text{OBJ/BKG}), V_i \in V$. Such edges are called T-links, as shown in Figure 3.6(a). The weight of T-links corresponds to the penalty of assigning a node to the corresponding terminal, which is given by the data term (3.8).

- (V_i, V_j), $V_i \in V_{LF}/V_{LB}/V_U$, $V_j \in V_U$, $\{i,j\} \in \mathbb{N}$ and $i \neq j$. Such edges are called N-links, as shown in Figure 3.6(b). The weight of N-links corresponds to the contrast/edge information, which is given by the smoothness term (3.9).

The edge weights are defined in Table 3.1, where $K = \infty$ (a very large value in practice), λ is the relative importance of data term, α controls smoothness. $D(f)$ and $V(f)$ in (3.5) now become

$$D(f) = \lambda \sum_{p \in P} -\ln Pr(p|f_p) \tag{3.8}$$

$$V(f) = \alpha \sum_{\{p,q\} \in \mathbb{N}} [f_p \neq f_q] \exp(-\|Z_i - Z_j\|^2/\beta) \tag{3.9}$$

Compared to previous segmentation techniques based on graph cuts, our graph construction is much simplified in terms of number of nodes and edges. Rather than creating a node for every pixel in the image, all pixels in Q_{CF} are represented by a single node, and the same principle holds true for pixels in Q_{CB}. Such representation prevents a cut from being made across the Q_{CF} (and Q_{CB}) area. In addition, we add neighborhood edges under very strict conditions: Since a cut can only happen in the unknown area, a contrast term (V_i, V_j) for predicting the object boundary is computed only in Q_U or between Q_U and Q_{LF}/Q_{LB}. The worst-case runtime complexity for solving a min-cut/max-flow problem is $O(mn^2)$, where n is the number of nodes and m is the number of edges in the graph [12]. In our approach, the number of nodes for the same image can, experimentally on average, be reduced to $n/5$ and the number of edges can be reduced to $m/2$, so the runtime complexity can be reduced to $1/50 * O(mn^2)$ on average.

3.4 Contrast Preserving Relaxation Labeling

Relaxation labeling can be used to reduce ambiguities and noise based on the parallel use of local constraints between labels [27]. In image segmentation by relaxation labeling, each pixel is first assigned a probability vector and a label based on the color information, and the probability vector is then updated iteratively based on the local constraints between labels [25].

We can apply the Relaxation Labeling technique to complete our foreground segmentation based on the initialized pentamap described in Section 3.2. We propose a new local constraint involving a contrast term, which we call contrast preserving relaxation labeling. As in [25], we proceed in three steps:

- *Step 1*: Initialization. For each pixel p, compute a probability vector

$$Pr^0(p) = \begin{bmatrix} Pr_1^0(p) & Pr_0^0(p) \end{bmatrix} \tag{3.10}$$

  where $Pr_1^0(p)$ is the probability of pixel p belonging to the foreground ($f_p = 1$), and $Pr_0^0(p)$ the probability of belonging to the background ($f_p = 0$).

Based on the pentamap, the probability vector is defined according to the following scheme:

$$Pr_1^0(p) = \begin{cases} 1, & \forall p \in Q_{CF}, Q_{LF}, \\ \dfrac{Pr(p|f_p = 1)}{Pr(p|f_p = 1) + Pr(p|f_p = 0)}, & \forall p \in Q_U, \\ 0, & \forall p \in Q_{CB}, Q_{LB} \end{cases} \tag{3.11}$$

and

$$Pr_0^0(p) = 1 - Pr_1^0(p) \tag{3.12}$$

$Pr(p|f_p = 1)$ and $Pr(p|f_p = 0)$ are previously defined in (3.1) and (3.3) respectively.

- *Step 2*: Iteration. In the nth iteration, the probability vector $Pr^n(p)$ for pixel p is updated based on the previous vector $Pr^{n-1}(p)$ and the neighborhood probability vector $Pr^n(q)$, $q \in \overline{N}(p)$ where $\overline{N}(p)$ is the 8-connected neighborhood about pixel p.

$$Pr_i^n(p) = \frac{Pr_i^{n-1}(p)(1 + Q_i^{n-1}(p))}{\sum_{j=0}^{1} Pr_j^{n-1}(p)(1 + Q_j^{n-1}(p))} \tag{3.13}$$

where

$$Q_i^{n-1}(p) = \frac{1}{card(\overline{N}(p))} \sum_{q \in \overline{N}(p)} C(p, q) * Pr_i^{n-1}(q) \tag{3.14}$$

with $i = \{0, 1\}$. $C(p, q)$ is the compatibility coefficient, and it uses contrast information as follows:

$$C(p, q) = \begin{cases} 1, & \text{if } \|Z_p - Z_q\| \leq \theta, \\ -1, & \text{otherwise} \end{cases} \tag{3.15}$$

where θ is the threshold of the contrast term.

- *Step 3*: Convergence and final labeling. As noted in [27], (3.13) converges to a consistent labeling as $n \to \infty$. After running for n_f iterations, each pixel is assigned a label with a larger probability component. We found $N_f = 10$ to be sufficient in our experiment.

3.4.1 CPRL Implementation on GPU

GPUs (graphics processing unit) are typically used only for computer graphics computations, which were traditionally handled by CPUs. GPUs add programmable vertex and fragment shaders to the graphics pipeline to increase graphics programming flexibility and to accelerate graphics pipeline processing. Vertex shaders allow vertex-based computation such as vertex's 3D position, and fragment shaders are used for per-pixel computation such as color processing.

GPUs can only process independent vertices and fragments, but can process many in parallel. This is especially effective when one wants to apply the same computation to many vertices or fragments. This computation is called kernel, and the programmer only needs to specify the kernel and data the kernel has to be applied to.

We claim that CPRL can be implemented in real-time based on the fact that it can be implemented in the GPU fragment shader. First, CPRL is a parallel algorithm, and the loop body in Step 2 is the computation kernel. Second, the updating process for each pixel is independent from the others in the same iteration, depending only on probability vectors of neighborhood pixels in the previous iteration.

3.5 Experimental Results

We captured two test video sequences for foreground IR illumination. Each video frame (either the color image or the IR image) has 365×480 pixels. All processes are running on a 2GHz Pentium desktop machine with 1G RAM. Real-time segmentation by GC was implemented in our laboratory, but in the meantime the alternative CPRL algorithm was implemented in Matlab.

3.5.1 Convergence of CPRL

We first show some segmentation results produced by CPRL. Figure 3.7 and Figure 3.8 are two examples of CPRL. At each iteration, we label pixels as foreground if the probability $Pr_0^n \in [0, 0.2)$, color pixels in pink if the probability

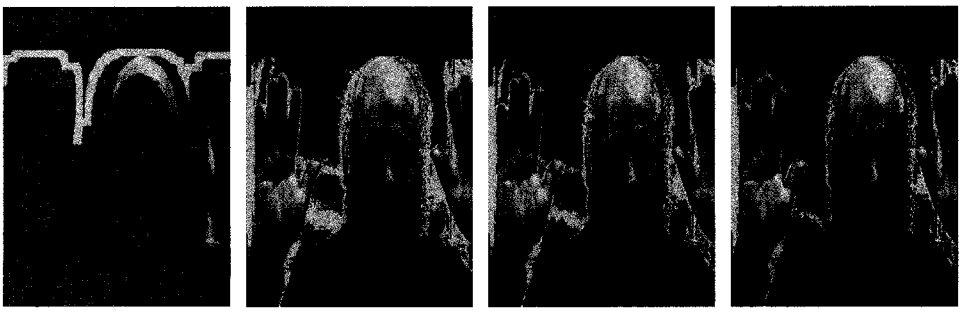
$Pr_0^n \in [0.2, 0.8]$, and otherwise in blue if the probability $Pr_0^n \in (0.8, 1]$. We can see that as more iterations are done, some pink pixels become either blue or labeled as foreground, which means the probability of those pixels converges to either 0 or 1. In the pentamap initialization, we used the following parameter values: $T = 0.004$ and $\tau = 55$. The value of s (in the definition of Q_{LF} and Q_{LB}) does not change the segmentation result much if $s \in [15, 25]$. We use $s = 25$ for video sequence 1 (referred as Seq1 below), and $s = 15$ for video sequence 2 (referred as Seq2 below). Throughout our experiments, we used the same parameter value for the pentamap initialization.

3.5.2 Comparison between GC and CPRL

Figure 3.9 shows the visual quality comparison of CPRL and GC. The segmentation results of CPRL and GC are not much different in visual quality, except that CPRL generates a smoother object boundary due to the fact that the computation of each pixel's probability of labeling is updated based on the neighborhood information. On the other hand, GC preserves better contrast information, which can be seen from the finger part of Seq1 frame 044 and frame 101 (see Figure 3.9). Some pixels between the fingers are background pixels but are labeled foreground in CPRL.

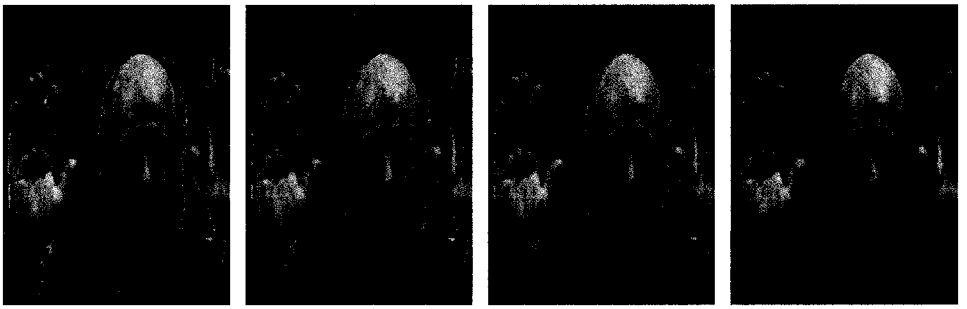
Evaluation Criteria

The performance of the GC and CPRL algorithms was evaluated in three respects: stability, accuracy and efficiency. Stability refers to changes in segmentation results as parameter values in the algorithms are changed. Is the result very sensitive to the parameter or is it very stable with respect to the variation of the parameter value. We test the stability by measuring the error rate of segmentation results given different sets of parameter values of the algorithm, and see how the variation of parameter values affects segmentation results. Accuracy is tested by measuring the error rate of segmentation results produced by different segmentation algorithms, given the best set of parameter values obtained from a stability test. Efficiency is measured by the average processing time for each video frame.



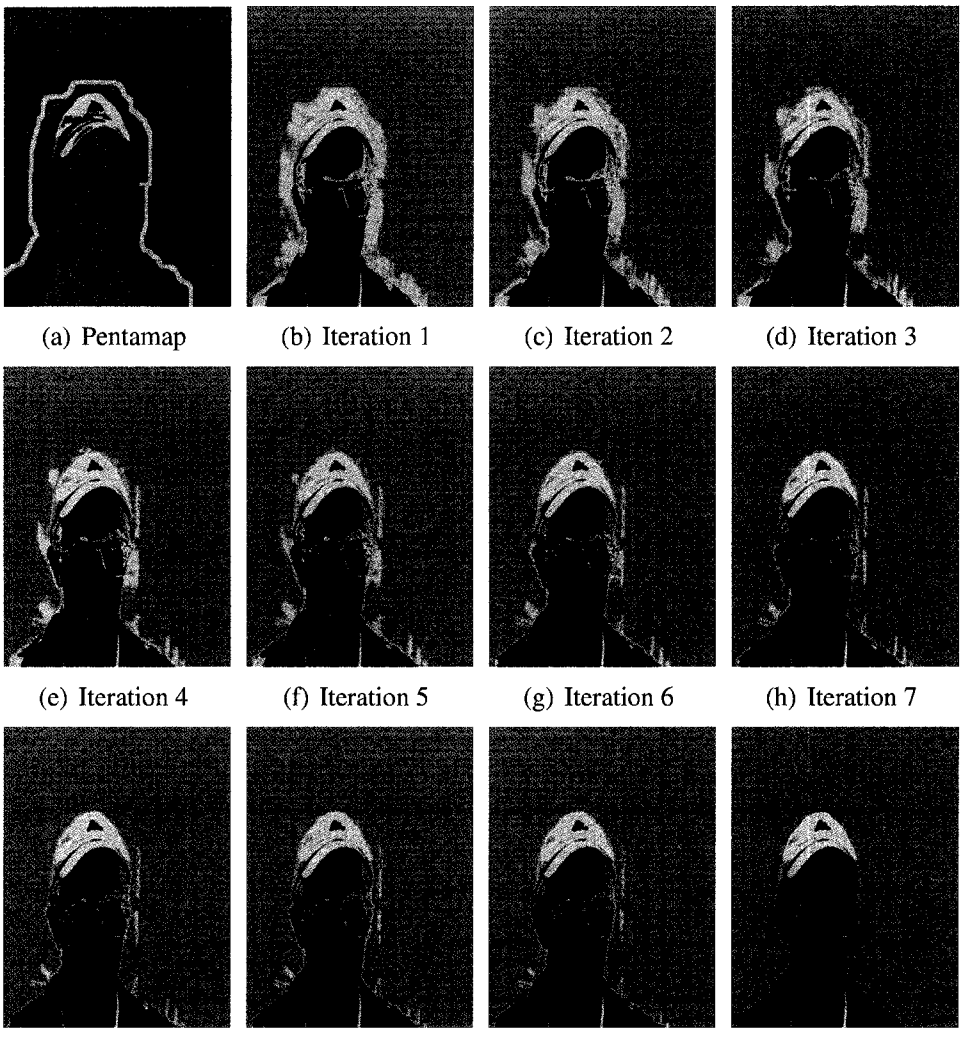
(a) Pentamap (b) Iteration 1 (c) Iteration 2 (d) Iteration 3

(e) Iteration 4 (f) Iteration 5 (g) Iteration 6 (h) Iteration 7



(i) Iteration 8 (j) Iteration 9 (k) Iteration 10 (l) Final Result

Figure 3.7: Convergence of CPRL on Seq 1 at frame 101. (a) Pentamap fed to CPRL. (b)-(k) show the probability of each pixel in each iteration. Pixels in pink color have the probability $Pr_0^n \in [0.2, 0.8]$. Pixels in blue color have the probability $Pr_0^n \in (0.8, 1]$. Pixels labeled as foreground have the probability $Pr_0^n \in [0, 0.2)$. Note that as n increases, $Pr_0^n \rightarrow 0/1$. Results are produced by θ=400.

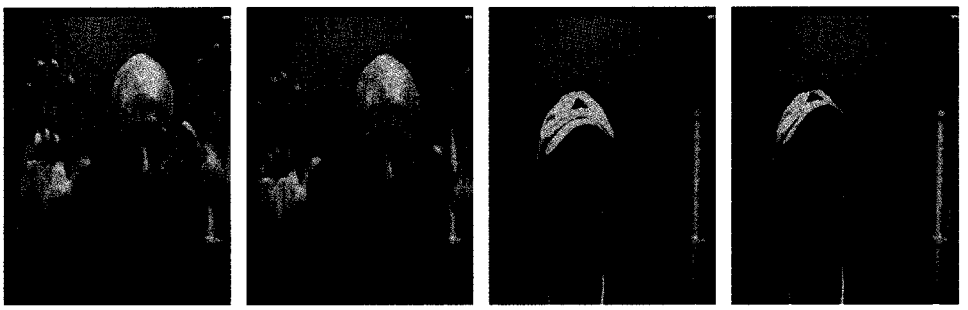


(a) Pentamap (b) Iteration 1 (c) Iteration 2 (d) Iteration 3

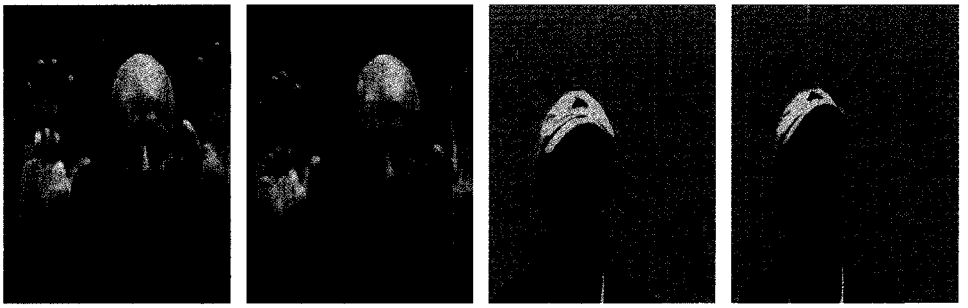
(e) Iteration 4 (f) Iteration 5 (g) Iteration 6 (h) Iteration 7

(i) Iteration 8 (j) Iteration 9 (k) Iteration 10 (l) Final result

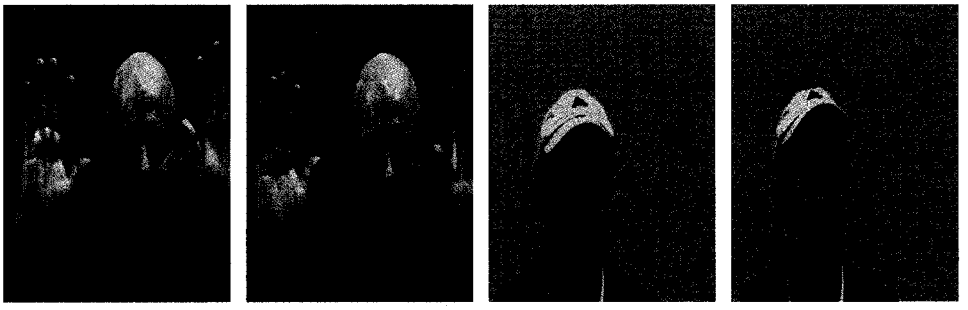
Figure 3.8: Convergence of CPRL on Seq2 at frame 027. (a) Pentamap fed to CPRL. (b)-(k) show the probability of each pixel in each iteration. Pixels in pink color have the probability $Pr_0^n \in [0.2, 0.8]$. Pixels in light blue color have the probability $Pr_0^n \in (0.8, 1]$. Pixels labeled as foreground have the probability $Pr_0^n \in [0, 0.2)$. Note that as n increases, $Pr_0^n \rightarrow 0/1$. Results are produced by θ=200.



(a) Seq1 044 (b) Seq1 101 (c) Seq2 027 (d) Seq2 057



(e) Seq1 044 CPRL (f) Seq1 101 CPRL (g) Seq2 027 CPRL (h) Seq2 057 CPRL



(i) Seq1 044 GC (j) Seq1 101 GC (k) Seq2 027 GC (l) Seq2 057 GC

Figure 3.9: Comparison of CPRL with GC. The first panel shows the original color images: we randomly select Seq1 frame044 and frame101, Seq2 frame027 and frame057 for experiment. The second panel shows segmentation results produced by CPRL, with $\theta = 400$. The third panel shows segmentation results produced by GC, with α=2, β=200, and λ=10.

To measure the error rate of segmentation results, the ground truth data was labeled manually, labeling each pixel as foreground, background or unknown. The unknown label was used to mark mixed pixels occurring along layer boundaries. Error is then measured as the percentage of incorrectly labeled pixels over the image area, ignoring "unknown" pixels. We acquired two video sequences containing 120 frames, and we measured the segmentation accuracy for every 10 frames starting with the 20th frame. In other words, we measured frames 20, 30, 40...110.

Stability

The algorithm GC has three parameters: α controls the smoothness, β is the expectation of the contrast term, and λ controls relative importance factor of the data term, which carries color information. We measured the error rate of segmentation results on Seq1 given four different sets of parameter values: $\{\alpha = 1, \beta = 200, \lambda = 20\}$, $\{\alpha = 10, \beta = 200, \lambda = 20\}$, $\{\alpha = 10, \beta = 200, \lambda = 2\}$ and $\{\alpha = 1, \beta = 200, \lambda = 2\}$. We only varied the value of parameter α and λ, since the segmentation result does not change much if $\beta \in [100, 400]$. The results are plotted in Figure 3.10(a), from which we can see that the segmentation result varies substantially as the parameter value changes. More specifically, the parameter sets $\{\alpha = 1, \beta = 200, \lambda = 20\}$ and $\{\alpha = 10, \beta = 200, \lambda = 20\}$ generally perform better than $\{\alpha = 10, \beta = 200, \lambda = 2\}$ and $\{\alpha = 1, \beta = 200, \lambda = 2\}$.

The CPRL algorithm has only one parameter, θ, the threshold of the contrast information. We also measured the error rate of segmentation results produced by three different parameter values: $\theta = 100$, $\theta = 200$ and $\theta = 400$, and we plotted the data in Figure 3.10(b). The data show that the segmentation result is robust as θ varies. The numerical error rate value in Figure 3.10(a) and (b) are listed in Table 3.2 and Table 3.3.

From our experiment, we can conclude that CPRL produces more stable segmentation result with respect to variations of parameter values. The segmentation results and spatial location of segmentation error, by different sets of parameter value, of Seq1 at frame 60 are shown in Figure 3.11.

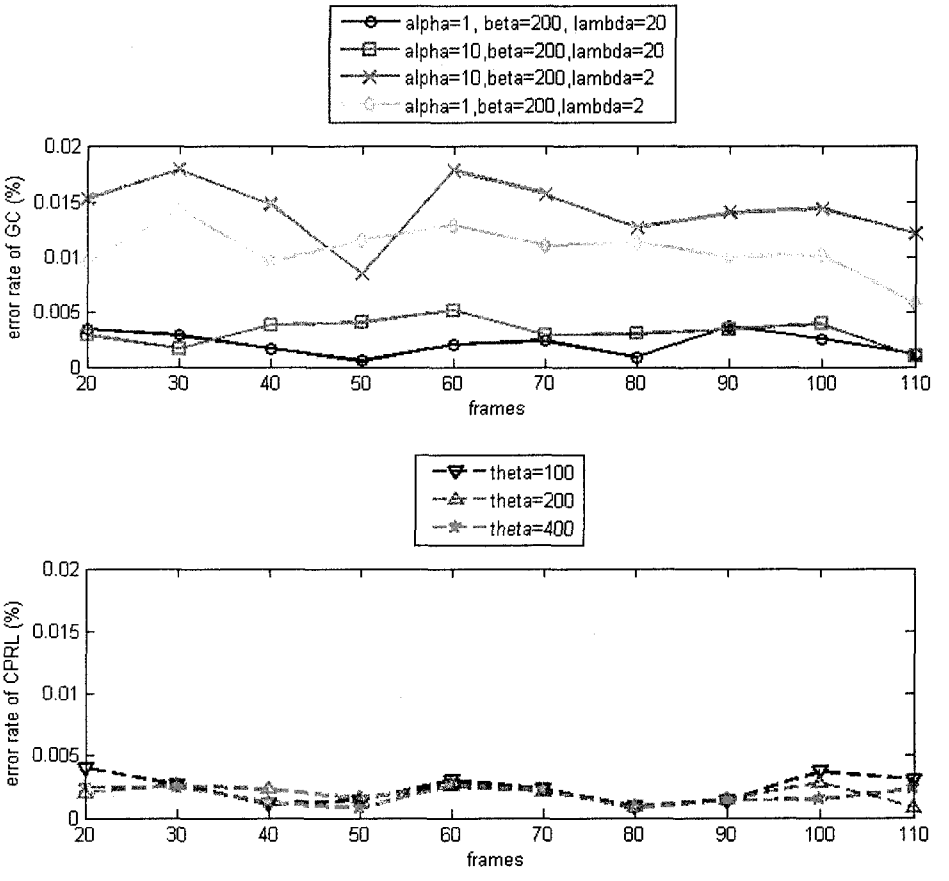


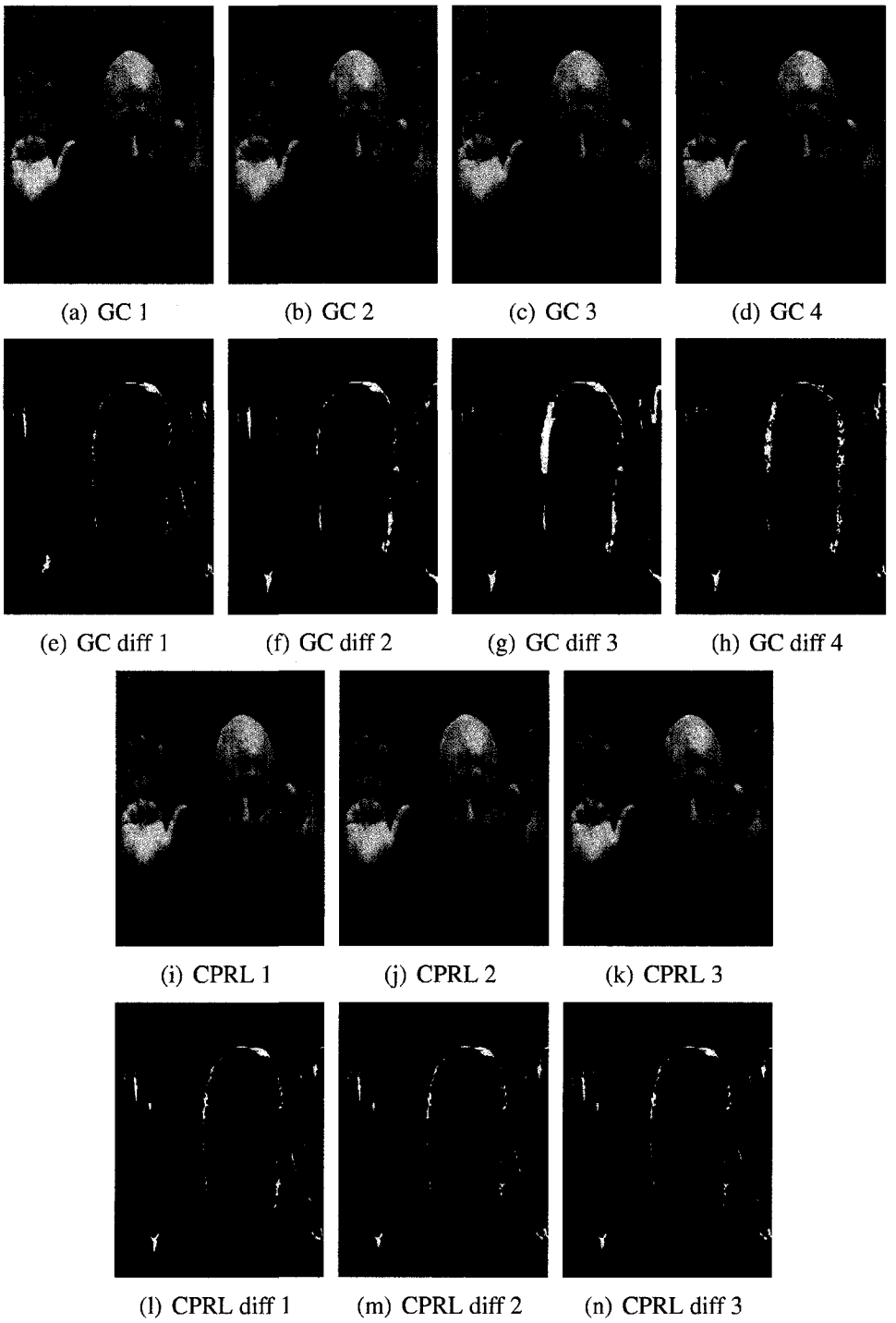
Figure 3.10: Error rate of GC and CPRL. The segmentation error is computed, with respect to ground truth, on Seq1 at every 10 frames starting at frame 20 till frame 110. (a) Error rate of segmentation results produced by GC. Tested on four different sets of parameter value: $\{\alpha = 1, \beta = 200, \lambda = 20\}$, $\{\alpha = 10, \beta = 200, \lambda = 20\}$, $\{\alpha = 10, \beta = 200, \lambda = 2\}$, $\{\alpha = 1, \beta = 200, \lambda = 2\}$. Note that segmentation results vary largely on different set of parameter value, and the parameter set $\{\alpha = 1, \beta = 200, \lambda = 20\}$ produces lower segmentation error on average. (b) Error rate of segmentation results produced by CPRL. Tested on three different sets of parameter value: $\theta = 100$, $\theta = 200$ and $\theta = 400$. Note that the segmentation results do NOT vary largely on different sets of parameter values, and the parameter set $\theta = 400$ produces lower segmentation error on average.

Table 3.2: Error rate table for segmentation results produced by GC

Error rate table										
	Frame									
Parameter Value	20	30	40	50	60	70	80	90	100	110
$\{\alpha = 1, \beta = 200, \lambda = 20\}$	0.0035	0.0029	0.0017	0.0006	0.0021	0.0024	0.0009	0.0038	0.0026	0.0011
$\{\alpha = 10, \beta = 200, \lambda = 20\}$	0.0029	0.0016	0.0039	0.0041	0.0052	0.0030	0.0031	0.0035	0.0039	0.0010
$\{\alpha = 10, \beta = 200, \lambda = 2\}$	0.0153	0.0179	0.0148	0.0084	0.0178	0.0157	0.0127	0.0140	0.0144	0.0121
$\{\alpha = 1, \beta = 200, \lambda = 2\}$	0.0096	0.0143	0.0096	0.0116	0.0129	0.0111	0.0113	0.0101	0.0102	0.0058

Table 3.3: Error rate table for segmentation results produced by CPRL

Error rate table										
	Frame									
Parameter Value	20	30	40	50	60	70	80	90	100	110
$\theta = 100$	0.0039	0.0026	0.0011	0.0014	0.0030	0.0023	0.0009	0.0013	0.0037	0.0030
$\theta = 200$	0.0019	0.0026	0.0022	0.0015	0.0025	0.0021	0.0009	0.0014	0.0028	0.0008
$\theta = 400$	0.0024	0.0024	0.0012	0.0008	0.0026	0.0022	0.0007	0.0014	0.0014	0.0022



(a) GC 1 (b) GC 2 (c) GC 3 (d) GC 4

(e) GC diff 1 (f) GC diff 2 (g) GC diff 3 (h) GC diff 4

(i) CPRL 1 (j) CPRL 2 (k) CPRL 3

(l) CPRL diff 1 (m) CPRL diff 2 (n) CPRL diff 3

Figure 3.11: Spatial distribution of segmentation error for Seq1 at frame 60. (a)-(d) are the segmentation results produced by GC. (a) is produced by $\{\alpha = 1, \beta = 200, \lambda = 20\}$ (b) is produced by $\{\alpha = 10, \beta = 200, \lambda = 20\}$ (c) is produced by $\{\alpha = 10, \beta = 200, \lambda = 2\}$ (d) is produced by $\{\alpha = 1, \beta = 200, \lambda = 2\}$. (e)-(h) are the spatial distributions of the segmentation error corresponding to (a)-(d). (i)-(k) are the segmentation results produced by CPRL. (i) is produced by $\theta = 100$ (j) is produced by $\theta = 200$ (k) is produced by $\theta = 400$. (l)-(n) are the spatial distributions of segmentation error corresponding to (i)-(k).

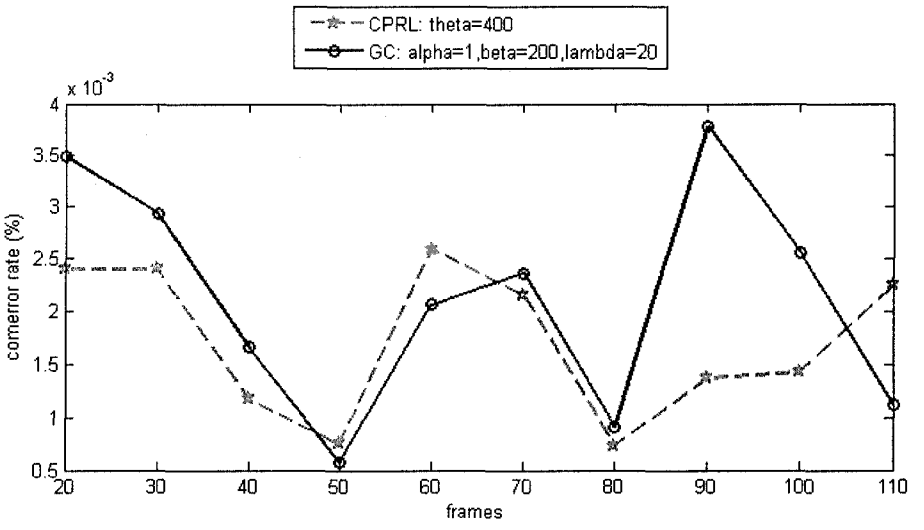


Figure 3.12: Comparison error rate of GC with CPRL. Results are produced on Seq1. Error rate of GC is produced by $\{\alpha = 1, \beta = 200, \lambda = 20\}$ and error rate of of CPRL is produced by $\theta = 400$.

Accuracy

We compare the accuracy of CPRL with GC by measuring the error rate of segmentation results produced by two algorithms. We used the parameter values which performed best in the stability test. For GC, we used the parameter set with best overall performance in the stability test section, which is $\{\alpha = 2, \beta = 200, \lambda = 20\}$. The same is true for CPRL, with overall best performance parameter value in the stability test $\theta = 400$. The comparison of the error rates of CPRL and GC is shown in Figure 3.12. One can see that, given the specific parameter value, CPRL segments images with lower error rate on average comparing to GC. We compared error rate on 20 different sets of parameter value for GC and CPRL, the overall performance of CPRL is still superior to GC because the segmentation results produced by GC has larger variation depending on the parameter value.

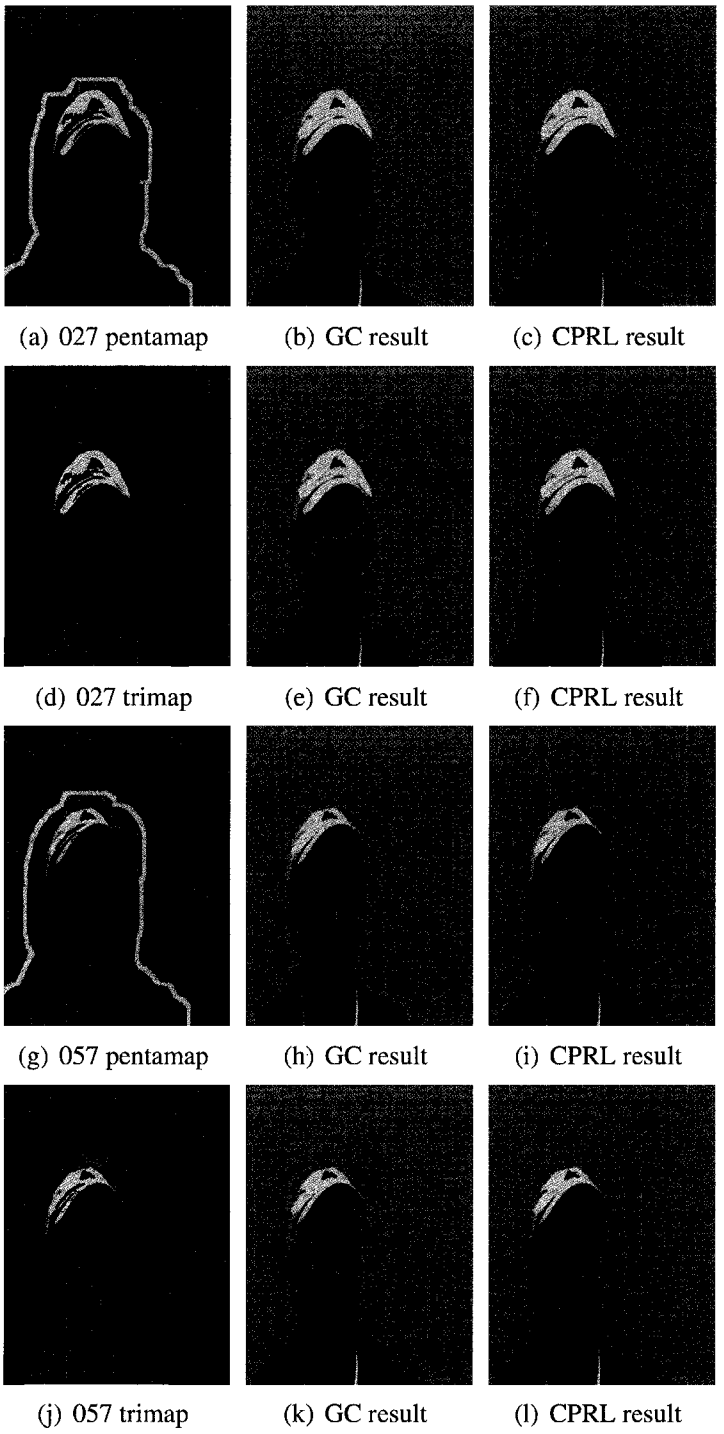
Efficiency

We implemented GC in C++, using the min-cut/max-flow algorithm code provided by Vladimir Kolmogorov [12]. Since the graph construction, in terms of the number of edges and nodes, is much simplified due to the pentamap, the computation is speeded up to the $1/50$ percent of original speed (explained in section 3.3). On average, it took only 0.1 seconds for processing a 365×480 image on our computer.

We implemented CPRL in Matlab, as an alternate algorithm for foreground segmentation. The claim we made that CPRL can be implemented in real-time is based on the GPU implementation. If we have 10 iterations in CPRL, the image needs to be processed in the GPU frame shader ten times. Given that the computation of each iteration can be finished in a few microseconds, the processing of each image can be done in real-time.

3.5.3 Comparison between Pentamap and Trimap

In our experiments, we verified that our proposed pentamap outperforms the trimap. We initialized the pentamap and the trimap according to our definition in Section 3.2, fed them to the segmentation algorithm and compared the segmentation results. According to the pentamap definition, the color model for foreground and background should be derived from Q_{LF} and Q_{LB}, which are colored as blue and pink region in the pentamap image, as shown in Figure 3.13(a) and Figure 3.13(g). In the trimap definition, however, the foreground and background color model are derived from T_F and T_B, which are colored red and green in the trimap image, as shown in Figure 3.13(d) and Figure 3.13(g). The difference of color model leads to differences in the segmentation results. The experimental results show that, in general, the pentamap produces more accurate segmentation results than the trimap. This supports our assumption in Section3.2 that "the color in unknown region should be consistent with the color of neighborhood region rather than the whole image".



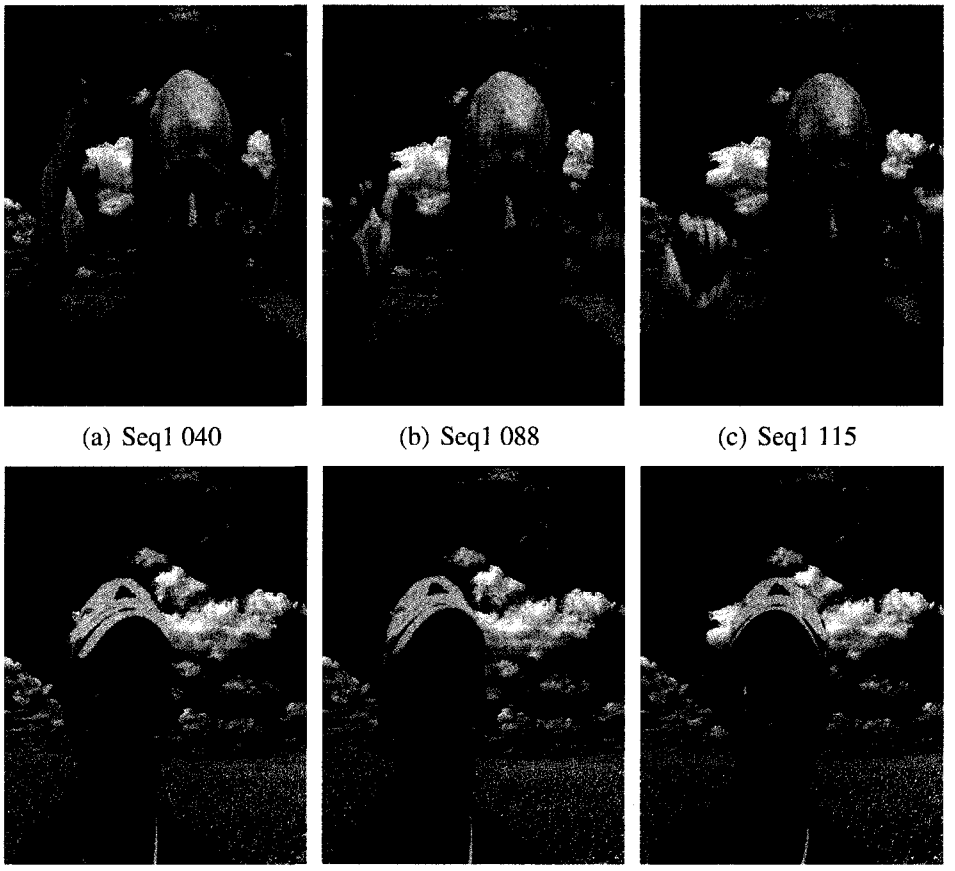
(a) 027 pentamap (b) GC result (c) CPRL result

(d) 027 trimap (e) GC result (f) CPRL result

(g) 057 pentamap (h) GC result (i) CPRL result

(j) 057 trimap (k) GC result (l) CPRL result

Figure 3.13: Pentamap vs. Trimap for Seq2 at frame 027 and frame 057. Segmentation results produced by GC use α=2, β=200, and λ=10. Segmentation results produced by CPRL use $\theta = 400$.



(a) Seq1 040 (b) Seq1 088 (c) Seq1 115

(d) Seq2 030 (e) Seq2 050 (f) Seq2 118

Figure 3.14: Video segmentation and background substitution. We show the segmentation results produced by GC for two experimental video sequences.

3.5.4 Comparison with Other Segmentation Methods

We compared our system with other video segmentation systems using the criteria we proposed for video conferencing system (see Section 1.1): real-time processing speed, robustness to dynamic background and automation. These are summarized in Table 3.4.

3.5.5 Background Substitution In Sequences

Figure 3.14 demonstrates the application of foreground/background segmentation in a video conferencing system, which substitutes the background in real-time. The segmentation result of our segmentation process is a binary label map where each pixel is classified as belonging to either the foreground or the background. Since

Table 3.4: Comparison table

Criteria	Source	Real-time processing	Robustness to dynamic background	Automation
Image-based segmentation		no	fail	user interaction needed
Motion-based segmentation	Yin et al. [53] Table 3	no	range of error rate [0.09 0.24]	yes
Stere-based segmentation	Yin et al. [53] Table 3	10 fps on 320×240 image	range of error rate [0.05 0.17]	yes
IR-based segmentation	Wu Table 3.3	10 fps for GC 100 fps for CPRL on 365×480 image	range of error rate [0.001 0.005]	yes

human vision is very sensitive to segmentation boundary artifacts, border blurring is applied to the object borders in order to blend the foreground with the new background. We thus achieve the equivalent of α-matting effects without really computing α values at each pixel but instead applying the process of Gaussian blurring process.

α-matting actually calculates a weighted average of foreground color and background color for each pixel, and a Gaussian blurring filter pre-calculates a weighted average of neighborhood colors for each pixel. These two definitions are very similar, especially when a pixel is at the border between foreground and background.

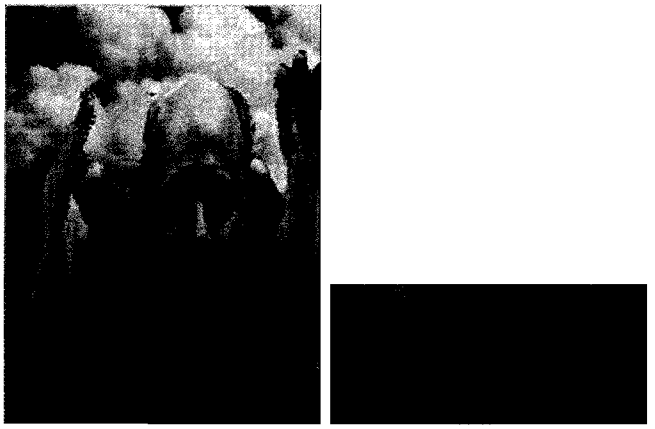
Border blurring begins with the "hard" segmentation produced by the GC algorithm (CPRL produces smoother object boundaries, and therefore we do not apply blurring process for CPRL). A blurred boundary contour is defined by the foreground area found by GC algorithm and morphological operations. The defined boundary contour should contain all pixels of boundary artifacts. A Gaussian filter is then applied to the boundary contour, so that there is a smooth transition between foreground and background, eliminating obvious artifacts at the boundaries. Figure 3.15 illustrates the segmentation results before and after border blurring.

Conventional matting techniques, such as Bayesian matting [50], compute α values based on the color of neighborhood pixels. This computation is very slow and does not perform well for objects with relatively smooth boundaries. We compared results produced by Bayesian matting and border blurring (both in Matlab code), as shown in Figure 3.16. For Bayesian matting, we used the Matlab code provided by Rucheek Sangani [14]. For an image of size 365×480, Bayesian matting took more than 40 minutes and border blurring took only less than 0.1 seconds. Our results presented here look very similar to the border matting results of GrabCut [44]. Our method is, however, much simpler and more efficient.

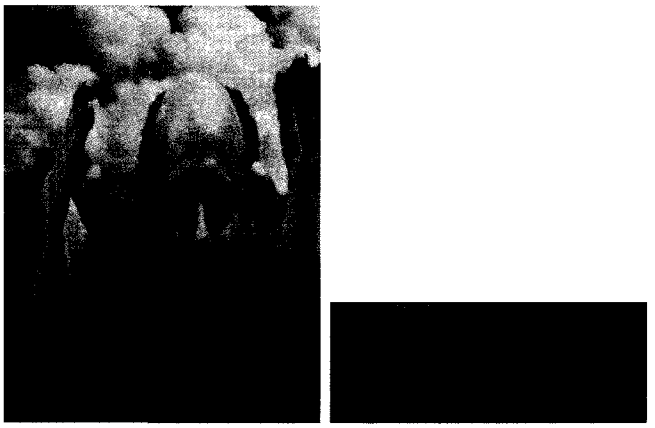
(a) Boundary Before Segmentation

(b) Boundary After Segmentation

Figure 3.15: Comparison of border blurring. Suppose the foreground area in (a) is F. The blurred boundary contour is defined as $Blur_{mask} = \{F - F.dilation(s_1).erosion(s_2).dilation(s_3)\}$. We used $s_1 = 4$, $s_2 = 12$, and $s_3 = 6$.



(a) Bayesian matting



(b) Border blurring

Figure 3.16: Comparison of border blurring with Bayesian matting. (a) is generated by Bayesian matting and (b) is generated by border blurring

3.6 Discussion

In this chapter, we presented a method for bi-layer segmentation of natural video in real-time using foreground IR illumination. By illuminating the foreground with an IR illuminator, the IR image has the following attributes:

- IR image is a gray-level image.
- The foreground object is illuminated by the IR illuminator and therefore has high intensity in IR image.
- The background has lower intensity in the IR image.

Two algorithms are presented to complete foreground segmentation: Graph Cut and Contrast Preserving Relaxation Labeling. We demonstrated the process of foreground segmentation by GC and CPRL with pentamaps, presented segmentation results with different algorithms, and analyzed the properties of GC and CPRL.

- Both GC and CPRL are capable of real-time processing. In theory, CPRL has more potential to achieve high speed by implementing the kernel in GPU due to the fact that it is a parallel algorithm.
- Both GC and CPRL produce good quality segmentation results. The segmentation results produced by GC preserve better contrast information, and ones produced by CPRL preserve better smooth segmentation boundary. As analyzed in accuracy section, on average, CPRL produces more accurate segmentation results with lower error rate.
- CPRL is more stable than GC with respect to the variation of parameter value. In other words, CPRL is less sensitive to the variation of parameter value.

Tradeoff between GC and CPRL: Given that both GC and CPRL produce good quality segmentation results, which one should we choose? At this point, we have a real-time GC implementation running in our laboratory. In the long run, however, once we have GPU implementation of CPRL, CPRL will exceed GC with respect

to processing time and stability. Furthermore, CPRL produces comparable, or even more accurate, segmentation results than GC. In a word, CPRL is more economical in computational speed.

There are some other important differences between GC and CPRL. Currently, we are looking into a parallel algorithm solution for GC using a GPU, without which we can only improve the computational efficiency of GC by taking benefit from pentamaps. The idea behind this is that a pentamap can simplify the graph construction in terms of number of nodes and edges, and therefore the run time complexity is reduced as it is proportional to the graph complexity.

There are many advantages for the foreground IR illumination design. First, the foreground object can be automatically recognized in the IR image given that it is illuminated by the IR illuminator and therefore appears bright in the IR image. Second, due to many good attributes of the IR image, the pentamap can be initialized robustly and automatically. This plays an important role in achieving the capability of automatic foreground segmentation. Third, pentamap initialization is independent of ambient lighting because of IR characteristics. Hence, the foreground-background segmentation will not be affected by changes in illumination (of ambient light). Fourth, the foreground object can be recognized if it is within the effective distance (D_{max} defined in Section 3.2) of the IR illuminator, and this distance acts like a plane dividing the foreground and background. Therefore, any moving objects presented in the background will not be segmented if it appears beyond this effective distance.

There are, however, two shortcomings with our foreground IR illumination method.

1. If the foreground object is too far away from the IR source, it will not be detected. This problem can be avoided by moving the IR source around and find the best position of the IR source by observing whether the IR image yields a good foreground MASK.

2. Any object behind the foreground object will be captured and segmented as foreground if it is very close to the foreground and within the effective distance of the IR source. This problem can be solved by using the background IR illumination method discussed in the next chapter.

Chapter 4

Segmentation by Background Illumination

In the previous chapter, we presented how to obtain automatic real-time foreground-background video segmentation by foreground IR illumination. The foreground illumination method has shortcomings caused by the constraints on the effective distance. In addition, missing foreground parts can only be found by a segmentation algorithm, which involves increased computational complexity. In this chapter, we investigate another way of utilizing IR information for foreground segmentation, background IR illumination. It can compensate, to a degree, for the disadvantages caused by the foreground IR illumination method.

4.1 Data Acquisition

In the background IR illumination method, the same data acquisition unit is used as in the foreground IR illumination method, as shown in Figure 3.1, except that the IR illuminator is moved behind the foreground and illuminate the background. An example of captured IR and color images is shown in Figure 4.1.

One can see that the IR image produced by background IR illumination is very different from the one produced by foreground illumination method: The background area appears bright in the IR image because of illumination of IR illuminator, and the foreground appears dark since all IR light hitting the foreground object is blocked from the IR camera.

Figure 4.1: A frame of IR image and color image produced by background IR illumination

We found that the background illumination method has several advantages. First, in the IR image, the foreground object is separated from the background area with high intensity contrast. Sharp and clear foreground boundary is well preserved (see Figure 4.1). Second, with background illumination one has more freedom regarding the positioning and orienting of the IR source (or sources). Even background objects that are close to the foreground object can be illuminated with an IR source and thus are easily classified as background. This reduces errors due to spurious foreground objects, which can occur with the foreground illumination method.

One can take advantage from IR images produced by background IR illumination. We segment the foreground object from the background by simple thresholding technique, without assistance of any segmentation algorithm. Since the IR image preserves sharp and smooth foreground boundary, matting or boundary blurring becomes unnecessary. However, as one may notice, some parts of the background also appear dark in the IR image. This is because those background parts are beyond the illumination field of the IR illuminator. If we have multiple IR illuminators to illuminate the background, there will be no "blind spots" in the background. At this point, we only have one IR illuminator, we will consider only the region of the illumination field for the purpose of comparison this method with foreground illumination method.

Since foreground segmentation is completed by only thresholding and no complex segmentation algorithm is needed, the processing is simple and very fast.

4.2 Experimental Results

We captured one test video sequence with background IR illumination. We cropped the video frame size to limit the segmenting area within the illumination field of the IR illuminator. As shown in Figure 4.2, in the view scene (images on left hand side), a computer behind the foreground object is turned on, playing a dynamic video sequence of desktop screen saver, which acts as the dynamic object presented in the background. Images on the right hand side of Figure 4.2 are segmentation results corresponding to the original video frame on the left. Our experiment results show that the foreground object can be well segmented from the background with simple thresholding, and there is no need to turn to any segmentation algorithm for help. Even in the presence of moving objects in the background, the segmentation results are not affected. The threshold we used in segmentation is $threshold = 0.2$.

4.3 Discussion

In this Chapter, we demonstrated a method of background IR illumination for foreground video segmentation, and presented segmentation results. By illuminating the background by IR illuminator, we can obtain an IR image with following attributes:

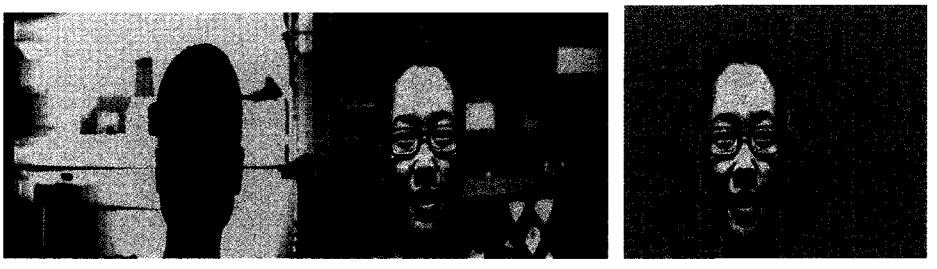
- The IR image is a gray-level image.
- The background area is illuminated by the IR illuminator, and therefore has high intensity in the IR image.
- The foreground has low intensity in the IR image.
- The foreground boundary highly contrasts in intensity against background, and is very sharp and clear in the IR image.

(a) Frame 100 (b) segmentation result

(c) Frame 130 (d) segmentation result

(e) Frame 160 (f) segmentation result

(g) Frame 190 (h) segmentation result



(i) Frame 220 (j) segmentation result

Figure 4.2: Segmentation result by background IR illumination

4.3.1 Comparison with Foreground Illumination

The background illumination method is compared with the foreground illumination method using the following criteria.

- Distance constraint: Background illumination method eliminates the distance constraint that is required in the foreground illumination method. No matter how close the background object is to the foreground object, it will not be falsely labeled as the foreground as long as it is illuminated by the IR illuminator.

- Accuracy: In the background illumination IR image, the foreground has high contrast with background at the boundaries. Therefore it has, on average, a lower error rate.

- Efficiency: There is no need for complex algorithms in background illumination. Simple thresholding can yield real-time processing speed.

- Cost: Background illumination has higher demands on the illumination. Adding more IR illuminators increases the cost of the system.

If the background IR illumination method is superior to foreground IR illumination method in so many aspects, should we simply go with background illumination method for the application of background substitution in video conferencing system? In fact, this is still an open question. The background illumination method has other problems, such as higher demands on illumination. Only if we have a sufficient number of IR illuminators and the if background is well illuminated without any "blind spots" , then the simple thresholding technique can produce good segmentation results.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This thesis presents a system solution to the problem of automatic bi-layer video segmentation in real-time in the application of background substitution for a video conferencing system. The proposed system includes a design for combining IR information with normal video information, and different methods are proposed for segmentation for different IR illumination technique.

In our design, an IR illuminator gives an imperceptible light source in addition to ambient light. The way we combine the IR camera and the color camera enables the system to automatically generate synchronized IR video sequences. We explored two ways of illuminating the scene with IR illuminators. Different foreground background IR illumination leads to different methods for foreground-background segmentation.

In foreground IR illumination, it is the foreground that is illuminated by the IR illuminator. A foreground MASK can be found by thresholding the IR image, and the missing foreground parts are completed by two alternative segmentation algorithms, GC and CPRL. In the background IR illumination method, it is the background that is illuminated by the IR illuminator. The complete foreground region can be found by thresholding the IR image, requiring that the whole background area in the view scene be well illuminated, which can only be achieved with multiple IR illuminators.

This thesis presented these two options for applying IR information in video segmentation. The experimental results show that they are both promising. Advantages and disadvantages of both two methods are analyzed in the corresponding chapter. It would be interesting to continue experimenting with both ways of IR illumination and to compare which one is more suitable for the application of video conferencing system.

As many systems specialized for the application in certain circumstances, our proposed system also has limitations. The use of IR mainly determines the constraints of the system. First, in the foreground illumination method, there is a distance constraint. The IR camera can automatically detect the foreground object only if it is within the effective distance of the IR source, and this distance is used to divide foreground and background. Therefore, the user may need to move the IR source around and find the best position by observing whether the IR image yields a good foreground region. The distance constraint also requires that background objects can not be too close to the foreground object, otherwise if the background object is captured by the IR camera it is also labeled as foreground. Second, in the background illumination method, the number of IR illuminators constrains the illumination field, the field of view that our video segmentation system is effective in. The field of view of the cameras has to be inside the illumination field. Otherwise, there will be blind spots in the illuminated background. It would be interesting to continue exploring our system to develop a method of eliminating the these constraints.

5.2 Future Work

There are mainly two directions in our future work. First, CPRL needs to be implemented on a GPU for real-time bi-layer video segmentation. With further experimental results, we will be in a better position to compare GC with CPRL. It is also interesting to investigate how the compatibility coefficients in CPRL affect the segmentation results. Other than contrast-preserving relaxation labeling, we should investigate other suitable relaxation labeling models for video/image segmentation.

Second, other means of utilizing IR for video segmentation should be explored. There are many ways of combining IR with normal video technique. For example, the combination of foreground and background IR illumination may give a more robust initialization of cue map. With more regions determined by the IR, we could spend less effort and cost on the segmentation algorithms. On the downside, we would have higher cost for buying IR sources. Introducing polarized IR may make the detection of IR easier. As described by Ben-Ezra [8], polarized light from the background makes the background black in the image because its polarization is "out of stage". Ben-Ezra also tried out two illumination sources, the foreground was illuminated by unpolarized light, and the background was illuminated by the polarized light. It would be interesting to explore how it works on IR.

The problem of video segmentation itself has a wide range of applications. The approach proposed in the thesis is not only a new solution to real-time bi-layer video segmentation, but also to motion tracking and many other video applications. The idea of fusing IR with normal video technique provides us a new direction for video technology. The promising results presented in the thesis not only prove the usability of the system, but also reveal a new direction in vision in general.

Bibliography

[1] http://www.apple.com/macosx/features/ichat.html.

[2] Yucel Altunbasak, P. Erhan Eren, and A. Murat Tekalp. Region based parametric motion segmentation using color information. *Joural of Graphical Models and Image Processing*, 60(1):13–23, January 1998.

[3] N. E. Apostoloff and A. W. Fitzgibbon. Automatic video segmentation using spatiotemporal t-junctions. In *Proceedings of the 17th British Machine Vision Conference, Edinburgh*, 2006.

[4] H. Harlyn Baker. The coliseum immersive teleconferencing system. HP Laboratories: Palo Alto Ca, 2002.

[5] H. Harlyn Baker. Computation and performance issues in coliseum, an immersive videoconferencing system. HP Laboratories: Palo Alto Ca, 2003.

[6] S. S. Beauchemin, D. J. Fleet, and J. L. Barron. Performance of optical flow techniques. In *CVPR92*, pages 236–242, 1992.

[7] Peter N. Belhumeur. A bayesian approach to binocular stereopsis. *International Journal of Computer Vision*, 19(3):237–260, 1996.

[8] Moshe Ben-Ezra. Segmentation with invisible keying signal. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:1032, 2000.

[9] Stan Birchfield and Carlo Tomasi. Depth discontinuities by pixel-to-pixel stereo. In *ICCV*, pages 1073–1080, 1998.

[10] Michael J. Black. Combining intensity and motion for incremental segmentation and tracking over long image sequences. In *European Conference on Computer Vision*, pages 485–493, 1992.

[11] Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. *In Proc. IEEE International Conference on Computer Vision*, 1:105–112, 2001.

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

[13] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, November 2001.

[14] Yung-Yu Chuang, Brian Curless, David H. Salesin, and Richard Szeliski. A bayesian approach to digital matting.

[15] P. L. Correia and F. Pereira. Classification of video segmentation application scenarios. *IEEE Trans. Circuits Syst. Video Techn.*, 14(5):735–741, 2004.

[16] Ingemar J. Cox, Sunita L. Hingorani, Satish B. Rao, and Bruce M. Maggs. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3):542–567, 1996.

[17] A. Criminisi. i2i: 3d visual communication. Microsoft Research: Cambridge.

[18] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov. Bilayer segmentation of live video. In *CVPR06*, pages 53–60, Washington, DC, USA, 2006. IEEE Computer Society.

[19] A. Criminisi, J. Shotton, A. Blake, and P. H. S. Torr. Gaze manipulation for one-to-one teleconferencing. In *ICCV03*, page 191, Washington, DC, USA, 2003.

[20] Alexandre X. Falcao, Jayaram K. Udupa, Supun Samarasekera, Shoba Sharma, Bruce Elliot Hirsch, and Roberto de A. Lotufo. User-steered image segmentation paradigms: live wire and live lane. *Graph. Models Image Process.*, 60(4):233–260, 1998.

[21] Davi Geiger, Bruce Ladendorf, and Alan L. Yuille. Occlusions and binocular stereo. In *European Conference on Computer Vision*, pages 425–433, 1992.

[22] S. Burak Gokturk, Hakan Yalcin, and Cyrus Bamji. A time-of-flight depth sensor - system description, issues and solutions. In *CVPRW04*, page 35, Washington, DC, USA, 2004. IEEE Computer Society.

[23] W. E. L. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities in a site. In *CVPR*, page 22, Washington, DC, USA, 1998. IEEE Computer Society.

[24] Mohit Gupta and Krishnan Ramnath. Interactive segmentation tool-box.

[25] Michael W. Hansen and William E. Higgins. Relaxation methods for supervised image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(9):949–962, 1997.

[26] Ellen Catherine Hildreth. *Measurement of Visual Motion*. MIT Press, Cambridge, MA, USA, 1984.

[27] R. A. Hummel and S. W. Zucker. On the foundations of relaxation labeling processes. pages 585–605, 1987.

[28] G. Iddan and G. Yahav. 3d imaging in the studio (and elsewhere). In *Proc. Of SPIE 4298: Videometrics and Optical Methods for 3D Shape Measurements*, pages 48–55, 2001.

[29] Y. Ivanov, C. Stauffer, A. Bobick, and E. Grimson. Video surveillance of interactions. In *Proc. of the CVPR'99 Workshop on Visual Surveillance*, Fort Collins, Colorado, November 1998.

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

[31] S. Khan and M. Shah. Object based segmentation of video using color motion and spatial information. 2:746–751, 2001.

[32] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother. Bi-layer segmentation of binocular stereo video. In *CVPR*, pages 407–414, Washington, DC, USA, 2005. IEEE Computer Society.

[33] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. In *ECCV02*, pages 82–96, 2002.

[34] I. Koprinska and S. Carrato. Temporal video segmentation: A survey. *Image Communication*, 16(5):477–500, 2001.

[35] Maylor K. Leung and Yee-Hong Yang. Human body motion segmentation in a complex scene. *Pattern Recognition*, 20(1):55–64, 1987.

[36] X. Luan, R. Schwarte, Z. Zhang, Z. Xu, H.-G. Heinol, B. Buxbaum, T. Ringbeck, and H. Hess. Three-dimensional intelligent sensing based on the pmd technology. In *Proc. SPIE*, volume 4540, pages 482–487, Dec 2001.

[37] Tomoo Mitsunaga and Shree K. Nayar. Radiometric self calibration. *cvpr*, 01:1374, 1999.

[38] Eric N. Mortensen and William A. Barrett. Interactive segmentation with intelligent scissors. *Graph. Models Image Process.*, 60(5):349–384, 1998.

[39] T. Oggier, M. Lehmann, R. Kaufmann, M. Schweizer, M. Richter, P. Metzler, G. Lang, F. Lustenberger, and N. Blanc. An all-solid-state optical range camera for 3d real-time imaging with sub-centimeter depth resolution (swissranger). In *Optical Design and Engineering. Proceedings of the SPIE*, volume 5249, pages 534–545, Feb 2004.

[40] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(2):139–154, March 1985.

[41] I. Overington. Gradient-based flow segmentation and location of the focus of expansion. In *Proc. 3rd Alvey Vision Conference*, pages 169–177, 1987.

[42] Jerry L. Potter. Scene segmentation using motion information. *Computer Graphics and Image Processing*, 6:558–581, 1977.

[43] Andrew Rabinovich, Serge Belongie, Tilman Lange, and Joachim M. Buhmann. Model order selection and cue combination for image segmentation. *cvpr*, 1:1130–1137, 2006.

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

[45] Neven Santrac, Gerald Friedland, and Raul Rojas. High resolution segmentation with a time-of-flight 3d-camera using the example of a lecture scene. Technical Report B-06-09, Freie Universität Berlin Department of Mathematics and Computer Science, September 2006.

[46] A. Spoerri and S. Ullman. The early detection of motion boundaries. In *Proc. 1st International Conference on Computer Vision*, pages 209–218, 1987.

[47] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. *Computer Vision and Pattern Recognition*, 2:246–252, 1999.

[48] William B. Thompson. Combining motion and contrast for segmentation. *PAMI*, pages 543–549, November 1980.

[49] William B. Thompson, Kathleen M. Mutch, and Valdis A. Berzins. Dynamic occlusion analysis in optical flow fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7(4):374–383, 1985.

[50] Oliver Wang, Jonathan Finger, Qingxiong Yang, James Davis, and Ruigang Yang. Automatic natural video matting with depth. In *PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, pages 469–472, Washington, DC, USA, 2007. IEEE Computer Society.

[51] Oliver Williams. A switched gaussian process for estimating disparity and segmentation in binocular stereo. In *Advances in Neural Information Processing Systems 19*, pages 1497–1504. MIT Press, Cambridge, MA, 2007.

[52] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[53] P. Yin, A. Criminisi, J. Winn, and I. A. Essa. Tree-based classifiers for bilayer video segmentation. In *CVPR07*, pages 1–8. IEEE Computer Society, 2007.

[54] D. Zhang and G. Lu. Segmentation of moving objects in image sequence: A review. *Circuits, Systems, and Signal Processing*, 20(2):143–183, March 2001.

[55] Yu-Jin Zhang. *Advances in Image and Video Segmentation*. IRM Press, May 2006.