國 立 交 通 大 學

電子工程學系 電子研究所碩士班

碩　士　論　文

視訊會議適用並以建立背景模型為基礎之即時視訊影像切割

Real-Time Video Segmentation Based on Background Modeling for Videoconferencing

研 究 生：林岳賢

指導教授：林大衛 博士

中 華 民 國 九 十 三 年 六 月

視訊會議適用並以建立背景模型為基礎之即時視訊影像切割

Real-Time Video Segmentation Based on Background Modeling for Videoconferencing

研　究　生：林岳賢　　　　Student：Yueh-Hsien Lin

指導教授：林大衛 博士　　　　Advisor：Dr. David W. Lin

國　立　交　通　大　學

電子工程學系　　電子研究所碩士班

碩 士 論 文

A Thesis
Submitted to Institute of Electronics
College of Electrical Engineering and Computer Science
National Chiao Tung University
in Partial Fulfillment of Requirements
for the Degree of
Master of Science
in
Electronics Engineering
June 2004
Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 三 年 六 月

視訊會議適用並以建立背景模型為基礎之即時視訊影像切割

研究生: 林岳賢　　　　　　　　　　*指導教授: 林大衛教授*

國立交通大學　電子工程系 電子研究所碩士班

摘要

在本篇論文中，我們設計並實現一個在個人電腦上的視訊影像切割系統。此系統未來將被用來組成一個在個人電腦上的多點視訊會議系統。

此系統的基本概念是將畫面與其相對應的背景相減以得到移動的物件。首先，我們使用一個兩級的雜訊估計方法來估計攝影機的雜訊，並且把此結果拿來當做往後參數調整的參考。爲了取得一個最佳的背景，我們結合了兩個方法來消除因爲物體內部的平坦區域所造成的錯誤背景。首先，我們觀察六張連續畫面的變化情形來取得一個初步的背景。接著，我們利用影像變化加上填補及收縮的技巧來取的一個粗略的物件輪廓，並利用此資訊來修正初步的背景。當攝影機被移動時，我們會先估計出相對應的移動向量，並且由全景的儲存器中快速的找到相應的背景。最後的模擬結果顯示當我們得到足夠的背景後，所切割出來的物體邊界是相當精確的，且在攝影機被移動後，我們也能快速的重建背景。

在整體圖形應用及控制介面的實現上，我們採用了 Microsoft Windows SDK 來建構整個介面。在攝影機的控制上，我們引用了 VfW 的模組來達成。在 2.4-GHz CPU 及 256-MB RAM 的個人電腦上且攝影機不移動時，目前的執行速度是每秒約五張。若攝影機有移動，則是每秒約 1.7 張。

Real-Time Video Segmentation Based on Background Modelling for Videoconferencing

Student: Yueh-Hsien Lin *Advisor: Dr. David W. Lin*

Institute of Electronics
National Chiao Tung University

Abstract

We consider the design and implementation of video segmentation system on a personal computer (PC). The intended application is PC-based multipoint videoconferenceing system.

The basic idea of the system is a background subtraction technique. First, we use a two-stage noise estimation to estimate the camera noise and the result is used to decide the thresholds in the following steps. Due to the problem of flat inner regions, we combine two methods to obtain the final stationary background. First, the short-term background is used to obtain an initial background which usually includes many flat inner regions. Second, a temporary foreground mask is obtained using change detection, fill-in, and shrink techniques to remove the flat inner regions in the short-term background.

In order to deal with camera motion, we employ scene change detection, global motion estimation, and panorama background buffer. When camera motion occurs, the scene change detection starts the global motion estimation and then the stationary background buffer is recovered from panorama background buffer by the estimated global motion.

The simulation results show that when we gather enough information of the background, the obtained object mask is accurate and the accuracy of object mask can be quite consistent throughout the remainder of the sequence. The results also show that global motion estimation and background recovery can effectively improve the accuracy during rebuilding of new background.

In real-time PC-based software implementation of the segmentation algorithm, we employ a graphical user interface established using the Windows SDK. The capturing process is aided by the VfW (Video for Windows package). With a P-4 2.4-GHz CPU and 512-MB RAM, the current un-optimized implementation yields a speed of about 5 frames per second and for CIF (352x288) video when the camera is still. In the presence of camera motion, the processing time is about 1.7 frames per second.

誌謝

本篇論文的產生，最最感謝的就是我的指導教授－林大衛老師。謝謝老師給我這個之前攻讀土木系的門外漢一個機會，成為實驗室的一份子，並在這兩年的碩士生涯中給予各方面的協助，讓我快速的融入這個大環境，在短短的兩年中順利完成碩士論文。除了專業方面所需的知識外，在論文實作部份，老師更培養了我認真踏實的的研究態度，讓我獲益良多，而實驗室完善的資源，讓我可以順利的克服許多實作上所遭遇的的困難。感謝詹益鎬、吳俊榮、蔡家揚、劉夢遠和蔣宗書學長不吝提供其相關的研究經驗以及建議，讓我獲益良多。另外，感謝俊安和子良當我的模特兒，還有名彥、耀諄、振韋、建統、政偉、孝強、子瀚、明瑋、明哲、筱晴、盈縈等實驗室所有同伴，他們的勉勵與幫助，讓我擁有一個快樂的研究生生涯。最後，要特別感謝我的爸媽、哥哥與玉琳，他們給了我許多精神上的支持與鼓勵，陪伴我度過所有的艱難。在此，我要將我的論文獻給所有關心與幫助我的人。

2004 年夏天於交大

Contents

List of Figures

Chapter 1

Introduction

We consider the design and implementation of video segmentation system on a personal computer. The intended application is PC-based multipoint videoconferencing system.

The main objective in segmentation is to partition the data into meaningful and independent regions. According to the type of data, we can classify the processes into image segmentation and video segmentation. Here, we focus on the video segmentation. The general schemes for video segmentation can be seen as the following steps: pre-processing, feature extracting, decision, and post-processing. According to extracted features, many familiar techniques can be used to obtain the object mask, such as optical flow-based technique, change detection-based technique, edge detection-based technique and background subtraction-based technique.

The basic idea of the system is a background subtraction technique. The moving objects of current frame can be easily obtained by extracting the different region between current frame and background. To begin with, we need to build a background relative to current frame. The information of background can be obtained by many methods. The simplest way is to remove every moving objects and then obtain the background, but in many situations it is inconvenient to remove every moving objects. In this thesis, the background is obtained by gathering the stationary regions during the process of segmentation.

During the process of gathering information of background, flat inner regions in moving objects usually lead us to make wrong decisions since such regions usually look like stationary regions even though they move. In this system, a temporary object mask is used to reduce the influence of flat inner regions.

In general, the basic assumption of background subtraction technique is that the background should be stationary during whole process. When the background is changed due to camera moving, we should reset the background buffer and find a new background. In order to reduce the rebuilding time, we use the idea of sprite to recover most of the background buffer.

In many steps of our algorithm, we need a threshold to make decisions. In many situations, the threshold is adjusted to against the influence of noise and therefore we estimate the camera noise at first and the adjustment of threshold is based on camera noise.

This thesis is organized as follows. Chapter 2 is an overview of popular video segmentation techniques. Chapter 3 discusses the idea of image mosaicing. Chapter 4 gives a detailed description of proposed segmentation method. Chapter 5 describes the overall system architecture. Chapter 6 shows the simulation results of our system. Finally, chapter 7 contains the conclusion.

Chapter 2

Overview of Video Segmentation

2.1 Meaning of Segmentation

The main objective in segmentation is to partition the data into meaningful and independent regions. According to the type of data, we can classify the segmentation processes into image segmentation and video segmentation. In the case of image segmentation, the problem is two-dimensional in nature. While in case of video segmentation, in addition to the two-dimensional information we can also handle the problem with the aid of motion information.

The algorithm of segmentation depends to a large extent on the application and the data in which it is used. In one application, the ten partitions may be ideal but for other applications the two partitions may be desired. In the ideal case, we may develop a best algorithm for each application but it usually follows with large complexity. In practice, we often develop an algorithm best suitable to specific application not to general situation.

2.2 Basic Procedure of Segmentation

A general scheme for segmentation can be seen as the following steps: pre-processing, feature extracting, decision, and post-processing as show in Figure 2.1.

1. Pre-processing:

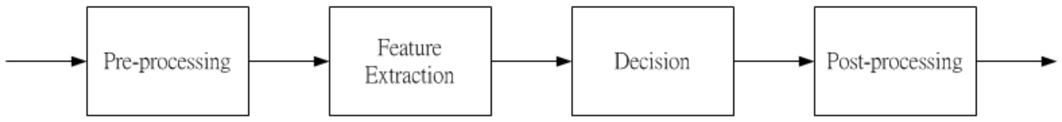


Figure 2.1: A basic segmentation system.

The original data may include lots of information but most of it is irrelevant to our application and sometimes influences our decision. In the stage, we remove the irrelevant information and keep the desired data.

2. Feature extraction:

   Depending on various segmentation algorithms, we need some specific features to achieve our goals. In this stage, we extract the desired features from original data.

3. Decision:

   The values of extracted features are usually more than two values and therefore we need to find a threshold to classify those data. In the stage, we can use many strategies to make a desired threshold set to find out the regions which meet our goal.

4. Post-processing:

   Although we can use many decision strategies to make a best threshold set, we may still make wrong decisions in some special regions. In the stage, we will try to correct those improper regions.

In the following sections, we introduce some popular segmentation techniques.

2.3 Optical Flow-Based Technique

The optical flow of a pixel is a motion vector represented by the motion between a pixel in current frame and its corresponding pixel in the following frame. In this method, we need to analyze the optical flow of every pixel and then partition a frame into different regions according to the different optical flows.

In the following we will show a system proposed in [1] in which the optical flow technique is used. The main assumption underlying this paper is the existence of a dominant global motion that can be assigned to the background. Areas that do not follow this background motion indicate the presence of independently moving objects. Therefore, the basic idea of the algorithm as shown in Figure 2.2 is to identify components that are moving differently from the background.

To be precise, the main block of the algorithm is the morphological motion filtering. It removes components that do not follow the dominant global motion, while perfectly preserving other parts of the image. The filter works on so-called flat zones in the image. A flat zone is the largest connected component where the gray-level is constant, possibly consisting of one pixel only, and the set of all flat zones is obviously a partition of the image. The morphological filter merges these flat zones according to a specified criterion, and so it does not introduce any new contours. The merging process is controlled by a filtering criterion that measures how well a flat zone conforms with the global motion.

The first step in this approach is to compute a dense optical flow field. The estimated dense motion field is then the starting point for calculating the global motion parameters. The global motion used in this approach is a six-parameter affine transformation:

$$u'(x,y) = a_1x + a_2y + a_3,$$

$$v'(x,y) = a_4x + a_5y + a_6,$$

where $(u'(x,y), v'(x,y))$ denotes the estimated optical flow vector at pixel (x,y). The parameters a_i $(1 \leq 6)$ are found by a robust least median of squares method, where by each independent flow vector $(u(x,y), v(x,y))$ of the estimated dense motion field provides one observation.

After calculating the global motion parameters, the transformation allows them to determine for each pixel (x,y) the motion vector $(u'(x,y), v'(x,y))$ according to the global motion. If the point belongs to the background, then $(u(x,y), v(x,y))$ and $(u'(x,y), v'(x,y))$ are expected to be very similar. Conversely, the more different $(u(x,y), v(x,y))$ and

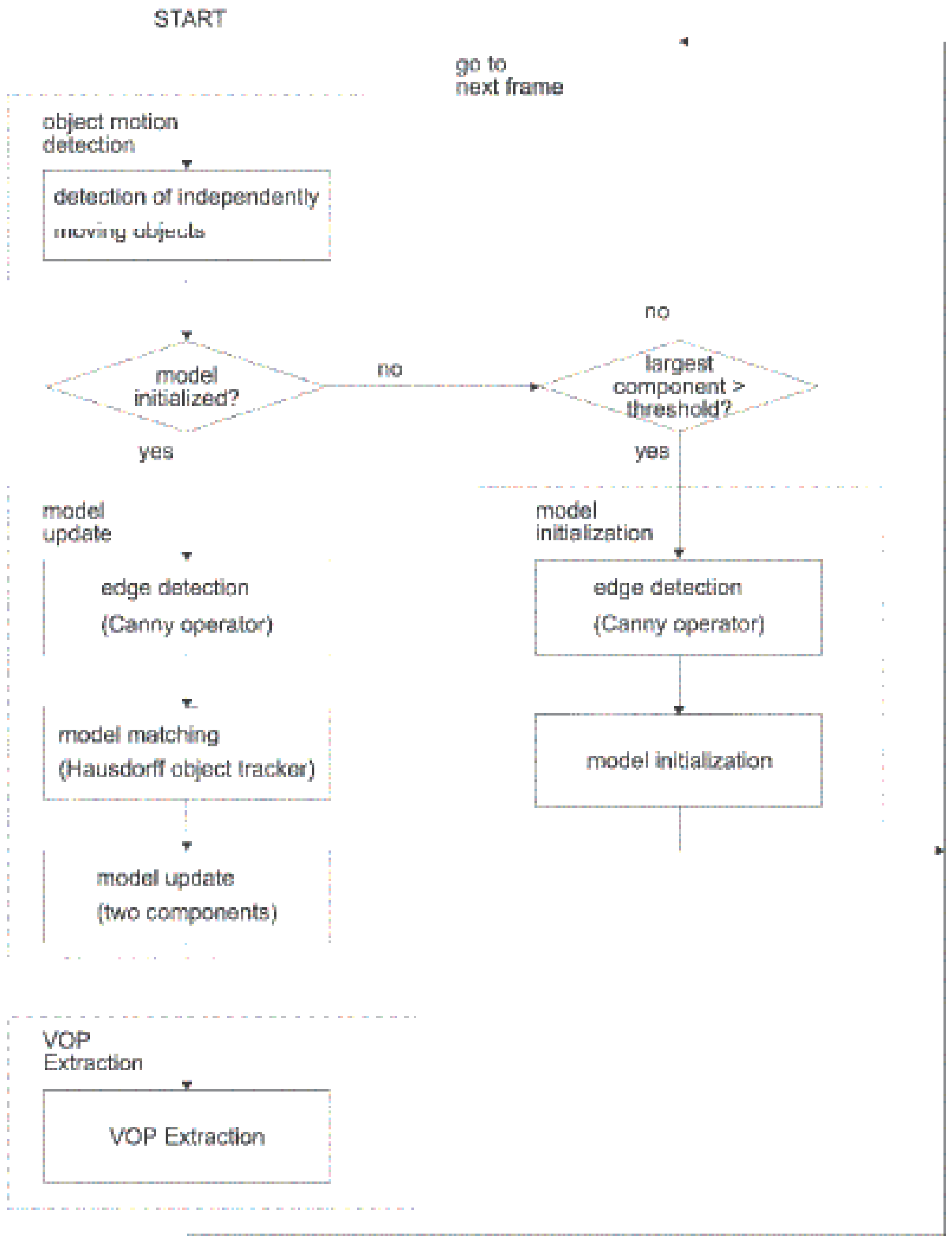

START
go to next frame
object motion detection
detection of independently moving objects
model initialized?
no
largest component > threshold?
no
yes
yes
model update
edge detection (Canny operator)
model matching (Hausdorff object tracker)
model update (two components)
model initialization
edge detection (Canny operator)
model initialization
VOP Extraction
VOP Extraction

Figure 2.2: Optical flow-based segmentation system (from [1]).

$(u'(x,y), v'(x,y))$ are, the more likely (x,y) belongs to an independently moving object. Therefore, the filter criterion $M(x,y)$ measures for each pixel (x,y) the difference between the synthesized global flow $(u'(x,y), v'(x,y))$ and the estimated local flow $(u(x,y), v(x,y))$, i.e.,

$$M(x,y) = (u'(x,y), v'(x,y))^2 + (u(x,y), v(x,y))^2,$$

where $M(x,y)$ is low for background pixels that conform with the global motion and high for pixels belonging to independently moving objects. The morphological filter requires a criterion for flat zones and not for individual pixels. This is now defined as the average of $M(x,y)$ over all pixels (x,y) belonging to a flat zone. Last, the residue or difference between original and filtered image indicates independently moving objects.

This technique has to estimate the motion of every pixel and find the dominant global motion. Usually, the related processes are very time-consuming. In many applications, the camera is always fixed and therefore the dominant global motion should be zero. For these reasons, another method called change detection-based technique is usually adopted to reduce the processing time of motion estimation when camera is fixed.

2.4 Change Detection-Based Technique

In the above method, optical flow is usually computed at every image point in the frame. Since the percentage of points in one frame having zero motion is usually large, it is possible to extract the independently moving objects from background by calculating frame difference between continuous frames. This technique is usually named change detection-based method. The main advantage of change detection is that it can avoid the computation in the estimation of optical flow.

Change detection algorithms usually start with the gray value difference map between the two frames considered. The local sum (or mean) of absolute difference is computed inside a small measurement window which slides over the difference map. At each location, this local sum of absolute differences is compared against a threshold. Whenever this threshold is exceeded, the center pixel of the current window location is marked as changed. The key issue of this technique is how to decide the threshold.

The algorithm in [2] provides some statistical methods to decide the threshold. It starts by computing the gray level difference image $D = d_k$, with $d_k = y_1(k) - y_2(k)$, between two considered pictures $Y_1 = y_1(k)$ and $Y_2 = y_2(k)$. The index k denotes the pixel location on the image grid. Under the hypothesis that no change occurs at location k, the corresponding difference d_k obeys a zero mean Gaussian distribution $N(0, \sigma)$ with variance σ^2, that is,

$$p(d_k|H_0) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-\frac{d_k^2}{2\sigma^2}\}.$$

Since the camera noise is uncorrelated between different frames, the variance σ^2 is equal to twice the variance of the assumed Gaussian camera noise distribution. H_0 denotes the null hypothesis, i.e., the hypothesis that there is no change at pixel k. The unknown parameter σ can be estimated offline for the used camera system, or recursively online.

In order to make the detection more reliable, they may use a region to evaluate the difference d_i instead of only a single pixel. They thus compute the local sum $\triangle_i^2$ of $(\frac{d_k}{\sigma})^2$ inside a small sliding window w_i, with i denoting the center pixel of the window.

The local sum $\triangle_i^2$ is proportional to the sample mean of $(\frac{d_k}{\sigma})^2$ as computed inside the window. Under the assumption that no change occurs inside the window when centered at location i, the normalized difference $(\frac{d_k}{\sigma})$ obeys a zero-mean Gaussian distribution $N(0, 1)$ with variance 1. Thus, the sum $\triangle_i^2$ obeys a χ^2-distribution with as many degrees of freedom as there are pixels inside the window. With the distribution $p(\triangle^2|H_0)$ known, the decision between "changed" and "unchanged" can be arrived at by a significance test. For this purpose, they usually specify a significance level α and compute a corresponding threshold t_α according to

$$\alpha = Prob(\triangle_i^2 > t_\alpha|H_0).$$

The statistic $\triangle_i^2$ is now evaluated at each location i on the image grid, and whenever it exceeds t_α, the corresponding pixel is marked as changed, otherwise as unchanged.

The method given above exhibits some shortcomings. First, there are inevitably decision errors. Typically, these errors appear as small isolated spots inside correctly labeled regions. Another drawback is that in some critical image areas the boundaries between differently classified regions tend to be somewhat irregular. To avoid these shortcomings, the MAP criterion can be used to get a better change mask. That is to say, they try to find the change mask $Q = q_k$ which maximizes the a posteriori density $p(Q|D)$, where D is the given difference image. The label q_k at location k can take either the value u for "unchanged" or c for "changed". After a series of deduction, we can get the finally decision rule

$$d_k^2 \overset{u}{\underset{c}{\gtrless}} 2\frac{\sigma_c^2\sigma^2}{\sigma_c^2-\sigma^2}(\ln\frac{\sigma_c}{\sigma} + (v_B(c)-v_B(u))B + (v_C(c)-v_C(u))C)$$

where B is a positive cost term of each horizontally or vertically oriented border pixel pair and C is that of diagonally pair. $v_B(q_k)$ and $v_C(q_k)$ denote the number of inhomogeneous cliques to which pixel k belongs when its label is q_k.

2.5 Edge Detection-Based Technique

Change detection for edges is another useful technique of segmentation. In the following, we explain how to employ the technique by the system in [4] and the block diagram is shown in Figure 2.3.

There are two major steps in this system.

1. Extraction of moving edge (ME) map:

   The automatic VO segmentation algorithm starts with edge detection which plays a key role in extracting the physical change of the corresponding surface in a real scene. However, such simple edge information usually suffers from a great deal of noise. Thus it results in slight changes of the edge locations in the successive frames. The difference of edges is defined as

   $$|\Phi(I_{n-1}) - \Phi(I_n)| = |\theta(\bigtriangledown G * I_{n-1}) - \theta(\bigtriangledown G * I_n)|,$$

   where the edge maps $\Phi(I)$ are obtained by the Canny edge detector [10], which is accomplished by performing a gradient operation on the Gaussian convoluted image, followed by applying the nonmaximum suppression to the gradient magnitude

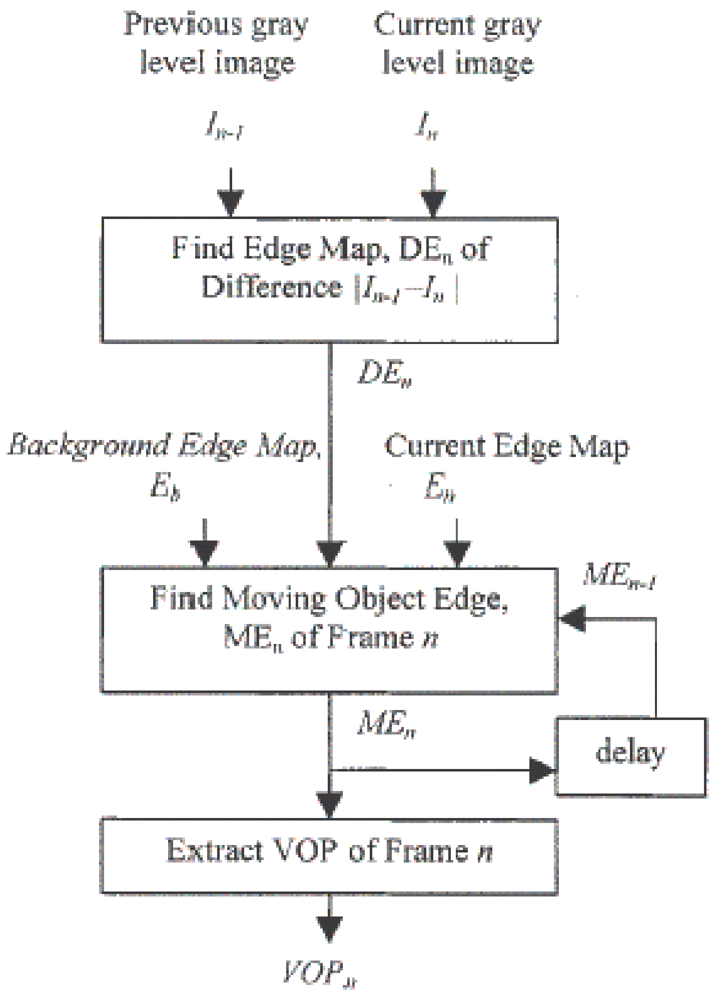

Previous gray level image
I_{n-1}
Current gray level image
I_n
Find Edge Map, DE_n of Difference $|I_{n-1} - I_n|$
DE_n
Background Edge Map, E_b
Current Edge Map E_n
Find Moving Object Edge, ME_n of Frame n
ME_{n-1}
ME_n
delay
Extract VOP of Frame n
VOP_n

Figure 2.3: Edge detection-based algorithm (from [4]).

to thin the edge and the thresholding operation with hysteresis to detect and link edges.

On the other hand, edge extraction from the difference image in successive frames results in a noise-robust difference edge map DE_n because Gaussian convolution included in the Canny operator suppresses the noise in the luminance difference

$$DE_n = |\Phi(|I_{n-1} - I_n|) = \theta(\bigtriangledown G * |I_{n-1} - I_n|).$$

After calculating the edge map difference of images using the Canny edge detector, they extract the moving edge ME_n of the current frame I_n based on the edge map DE_n of difference $|I_{n-1} - I_n|$, the current frame's edge map $E_n = \Phi(I_n)$, and the background edge map E_b. Note that E_b contains absolute background edges in case of a still camera and can be extracted from the first frame or by counting the number of edge occurrence for each pixel through the first several frames. They define the edge model $E_n = \{e_1, ..., e_k\}$ as a set of all edge points detected by the Canny operator in the current frame n. Similarly, they denote $ME_n = \{m_1, ..., m_l\}$ the set of l moving edge points, where $l \leq k$ and $ME_n \subseteq E_n$. The edge points in ME_n are not restricted to the moving object's boundary, and can be in the interior of the object boundary. If DE_n denotes the set of all pixels belonging to the edge map from the difference image, then the moving edge map generated by edge change is given by selecting all edge pixels within a small distance $T_{change}$of DE_n, i.e.,

$$ME_n^{change} = \{e \in E_b | \min_{x \in DE_n} ||e - x|| \leq T_{change}\}.$$

Some ME_n might have scattered noise which needs to be removed before proceeding to the next steps. In addition, a previous frame's moving edges can be referenced to detect temporarily still moving edges, i.e.,

$$ME_n^{still} = \{e \in E_n | e \notin E_b, \min_{x \in ME_{n-1}} ||e - x|| \leq T_{still}\}.$$

The final moving edge map for current frame I_n is expressed by combining the two maps

$$ME_n = ME_n^{change} \bigcup ME_n^{still}.$$

2. Extraction of VOP:

   With a moving edge map ME_n detected from DE_n, the VOPs are ready to be extracted. To obtain the foreground of image, they perform two steps. In the first step, they find the region which is between the first and the last edge points in each row. They may obtain some background region in the result. This is due to the fact that more than one object may exist in the sequence. Hence, in the second step, they obtain the region between the first and the last edge points in each column. The result of this step will also contain the background regions between the foreground objects. The final foreground region is obtain by applying *logical AND* operation on the horizontal and vertical candidate regions.

   In the case when the object boundary touches image boundary, they may obtain jagged moving edge in that location. This is because the actual moving edge points near the object boundary may lie outside the image boundary. The nearest edge points close to image boundary may be at various distances from the image boundary at various points. Hence they add the image points to moving edge points if either horizontal or vertical candidates touch image boundary points.

2.6 Background Subtraction-Based Technique

Thresholding the difference between two consecutive input frames is the basic concept of change detection-based segmentation. However, since the behavior and characteristics of the moving objects differ significantly, the quality of segmentation result depends strongly on background noise, object motion, and the contrast between the object and the background. Reliable and consistent object information is very difficult to obtain. Hence, instead of trying to get more information from the changing part of the scene, we concentrate on the stationary background where the characteristics are well known and more reliable.

The idea of background subtraction is to subtract the current image from the still background, which is acquired before the objects move in. After subtraction, only non-stationary or new object are left. The most straightforward way to separate background is to apply a simple difference and threshold method.

An example of background subtraction-based technique can be seen in [3]. This segmentation system consists of five major steps as shown in Figure 2.4.

1. Frame difference:

   The frame difference mask is generated simply by thresholding the frame difference. This data is sent to the background registration step where the reliable background is constructed from the accumulated information of several frame difference masks. Since the accumulated frame difference mask are used in the final decision for a reliable background, no filtering or boundary relaxation is applied on the frame difference.

   A significance test technique is used to obtain the threshold value. The test statistic is the absolute value of frame difference. Under the assumption that there is no change in the current pixel, the frame difference obeys a zero-mean Gaussian distribution and its probability density function is as follows:

$$p(FD|H_0) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-\frac{FD^2}{2\sigma^2}\},$$

   where FD is the frame difference and σ^2 is the variance of the frame difference. Note that σ^2 is equal to twice the camera noise variance σ_c^2. H_0 denotes the null hypothesis, i.e., the hypothesis that there is no change at the current pixel. The threshold value is decided by required significance level. Their relation is as follows:

$$\alpha = Prob(|FD| > TH|H_0),$$

   where α is the significance level and TH is the threshold value.

2. Background registration:

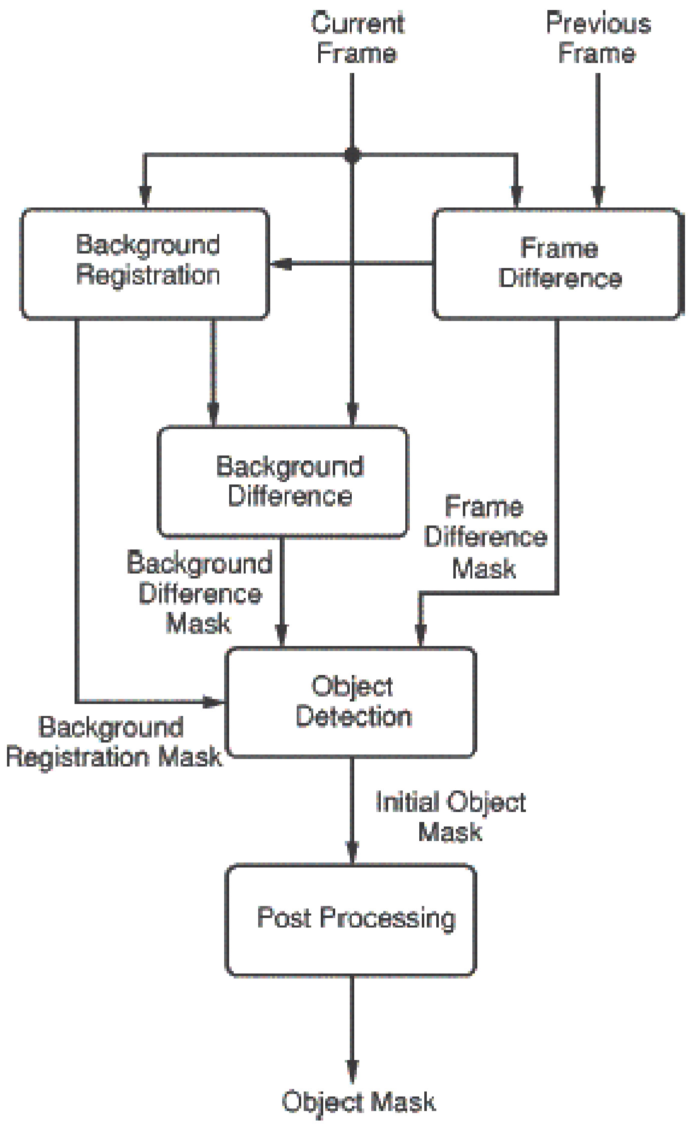


Figure 2.4: Background substraction-based segmentation system (from [3]).

The goal of background registration is to construct a reliable background information from the video sequence. In this application, it needs a reliable background information for change detection. An approximate background information is not helpful for object detection, and even worse, it will cause error in the later segmentation result until the background information is corrected. Therefore, for information that they are not very sure to be background, they tend to reject and leave the corresponding area in the background buffer empty.

In the background registration step, the history of frame difference mask is considered in constructing and updating the background buffer. A stationary map is maintained for this purpose. If a pixel is marked as changing in the frame difference mask, the corresponding value in the stationary map is cleared to zero; otherwise, if the pixel is stationary, the corresponding value is incremented by one. The values in the stationary map indicate that the corresponding pixel has not been changing for how many consecutive frames. If a pixel is stationary for the past several frames, then the probability is high that it belongs to the background region. Therefore, if the value in the stationary map exceeds a predefined value, then the pixel value in the current frame is copied to the corresponding pixel in the background buffer.

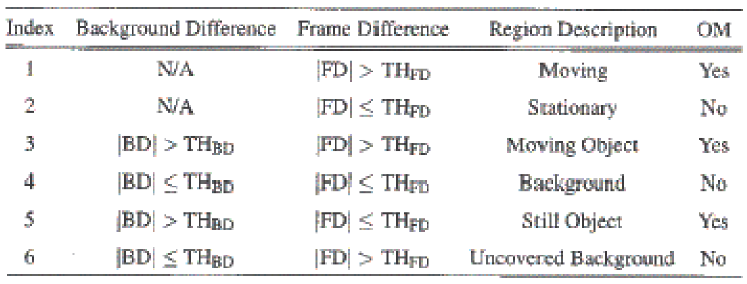
A background registration mask is also changed in this process. The value in the background registration mask indicates that whether the background information of the corresponding pixel exists or not. If a new pixel value is added into the background buffer, the corresponding value in the background registration mask is changed from nonexisting to existing.

3. Background difference:

   This step generates a background difference mask by thresholding the difference between the current frame and the background information stored in the background buffer. This step is very similar to the generation of frame difference mask.

4. Object detection:

   The object detection step generates the initial object mask from the frame difference mask and the background difference mask. The background registration mask,



Index	Background Difference	Frame Difference	Region Description	OM
1	N/A	$\lvert FD \rvert > TH_{FD}$	Moving	Yes
2	N/A	$\lvert FD \rvert \leq TH_{FD}$	Stationary	No
3	$\lvert BD \rvert > TH_{BD}$	$\lvert FD \rvert > TH_{FD}$	Moving Object	Yes
4	$\lvert BD \rvert \leq TH_{BD}$	$\lvert FD \rvert \leq TH_{FD}$	Background	No
5	$\lvert BD \rvert > TH_{BD}$	$\lvert FD \rvert \leq TH_{FD}$	Still Object	Yes
6	$\lvert BD \rvert \leq TH_{BD}$	$\lvert FD \rvert > TH_{FD}$	Uncovered Background	No

Figure 2.5: Initial object mask generation (from [3]).

frame difference mask, and background difference mask of each pixel are required information. Figure 2.5 lists the criteria for object detection, where BD means the absolute value of difference between the current frame and the background information stored in the background buffer, FD is the absolute value of frame difference, and the OM field indicates that whether or not the pixel is included in the object mask. TH_{BD} and TH_{FD} are the threshold values for generating the background difference mask and frame difference mask, respectively.

5. Post-processing:

   After the object detection step, an initial object mask is generated. However, due to the camera noise and irregular object motion, there exist some noise regions in the initial object mask. The approach to eliminate the noise region relies on an observation that the area of noise regions tend to be smaller than the area of the object. Regions with area smaller than a threshold value are removed from the object mask. In this way, the object shape information is preserved while smaller noise regions are removed. After removing noise regions, a close and an open operation with a 3×3 structuring element are applied on the object mask.

2.7 Summary

All techniques described above can obtain a desired object mask. Here, we explain the reason why we choose the background subtraction-based technique. In our application, the camera is usually fixed and therefore the optical flow-based technique is not under consideration.

The idea of change detection-based and edge detection-based techniques are very similar. In our experience, the object masks obtained by these two techniques are hard to resist the camera noise and some noise-removing filters are necessary. Besides, the accuracy of object mask is hard to keep for entire sequence.

The major disadvantage of the background subtraction-based technique is that it needs some time to gather enough information of background. The reason why we still choose this technique is that the accuracy of object mask is easier to keep when we collect enough information of background. Besides, the module for gathering background can be stopped to improve processing time when the whole background is obtained.

Chapter 3

Overview of Image Mosaicing

3.1 Introduction of Mosaic

The mosaic or sprite is an image constructed by all frames in a video sequence. The mosaic image can provide a panoramic view of scenes in the video sequence and there are many application based on this techniques. Since the successive images within a video sequence usually overlap by a large amount, image mosaicing technique can reduce the total amount of data needed to represent the scene. The sprite coding, for example, is one of the important components in MPEG-4 video coding and many related algorithms have been included in MPEG-4 video Verification Model [5].

In this thesis, we use this technique to deal with the camera motion. At first We estimate the camera motion and generate a panorama background buffer and then we can fast recover the stationary background buffer from panorama background buffer.

A mosaic generation system proposed in [9] is shown in Figure 3.1. There are three major steps in this system. First, choose some reliable region for alignment. Second, align all images in the sequence. Third, integrate aligned images to generate a mosaic image. The details are described in the following sections.

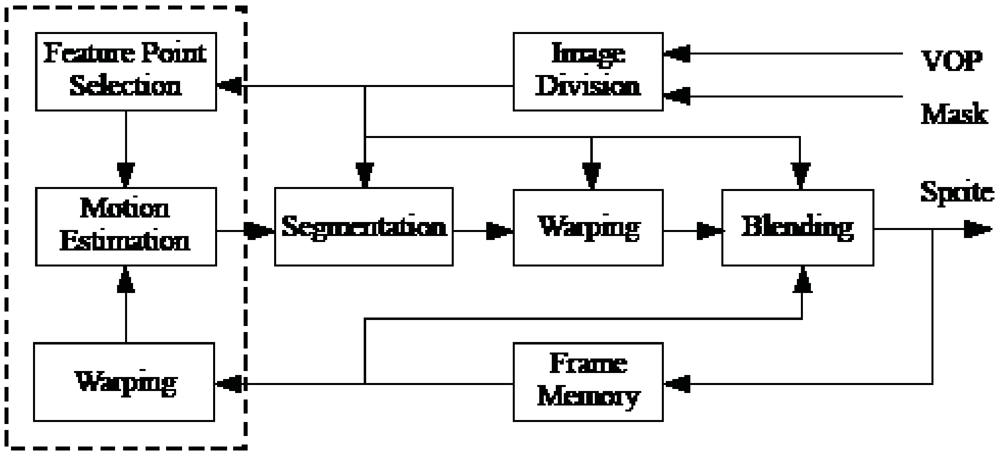


Figure 3.1: A sprite generation system (from [9]).

3.2 Reliable Region Selection

As we can see in [9], the main module of image alignment is the global motion estimation. In general, the accuracy of global motion estimation is usually influenced by local motion. Usually, the influence can be eliminated in a well designed global motion estimation algorithm. Besides, a simple way to deal with the problem is excluding those regions belonging to local motion before global motion estimation. Usually, the reliable regions can be obtained by the aid of initial segmentation mask as shown in Figure 3.2. The black region represents the reliable region, the gray region represents the unreliable region, and the white region represents moving objects.

3.3 Image Alignment

Image alignment depends on the chosen motion model and usually starts in global motion estimation (GME). The GME technique is designed to minimize the sum of squared difference E between the current frame I and the motion compensated previous frame I':

$$E = \sum_{i=1}^{N} e_i^2,$$

where $e_i = I'(x'_i, y'_i) - I(x_i, y_i)$, (x_i, y_i) denotes coordinates of ith pixel in current frame, and (x'_i, y'_i) denotes coordinates of corresponding pixel in previous frame.

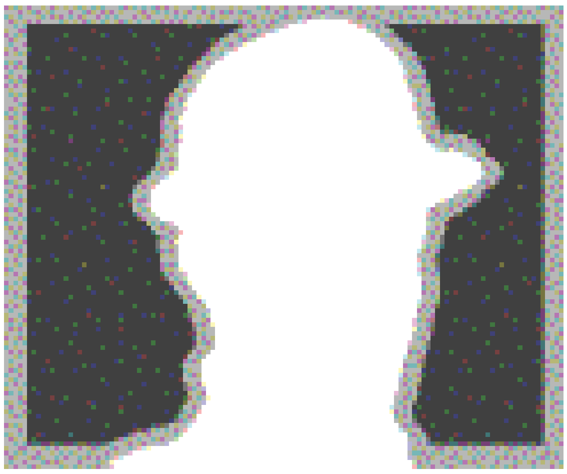


Figure 3.2: A reliability mask (from [9]).

There are many motion models that can be used to model camera motion. For example, in [6] the perspective motion model is defined as follows:

$$x_i' = \frac{a_0 + a_2 x_i + a_3 y_i}{a_6 x_i + a_7 y_i + 1}$$

$$y_i' = \frac{a_1 + a_4 x_i + a_5 y_i}{a_6 x_i + a_7 y_i + 1}$$

where $(a_0, ..., a_7)$ are motion parameters. An example of perspective transform is shown in Figure 3.3.

The perspective model is suitable when the scene can be approximated by a planar surface, or when the scene is static and the camera motion is pure rotation around its optical center. According to the perspective model, we can use other simpler models in some special situations. For instance, there are the affine model $(a_6 = a_7 = 0)$, the translation-zoom-rotation model $(a_2 = a_5, a_3 = -a_4, a_6 = a_7 = 0)$, the translation-zoom model $(a_2 = a_5, a_3 = a_4 = a_6 = a_7 = 0)$, and the translation model $(a_2 = a_5 = 1, a_3 = a_4 = a_6 = a_7 = 0)$. An example of affine transform is shown in Figure 3.4.

There are many ways to solve the motion parameters $\mathbf{a} = (a_0, ..., a_7)$. The gradient descent algorithm described in [6] and [7] is a popular method to solve for those parameters by the following iterative procedure:

$$\mathbf{a}^{t+1} = \mathbf{a}^t + \mathbf{H}^{-1}\mathbf{b}.$$

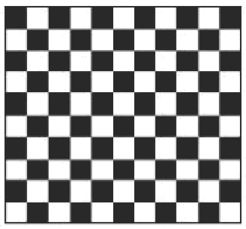
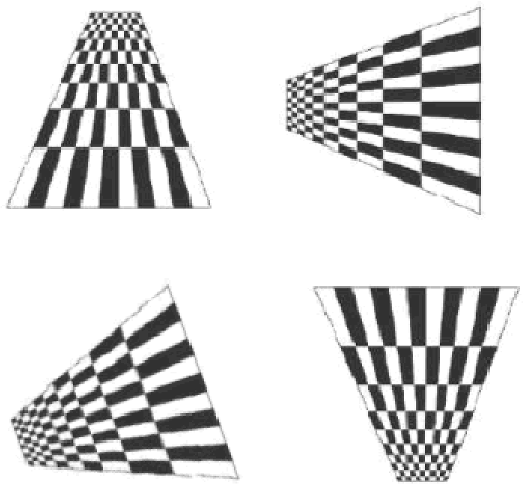
Original image	Images after perspective transform
	

Figure 3.3: Perspective transform.

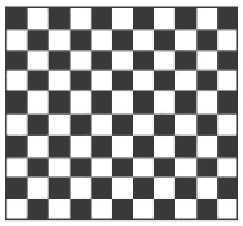
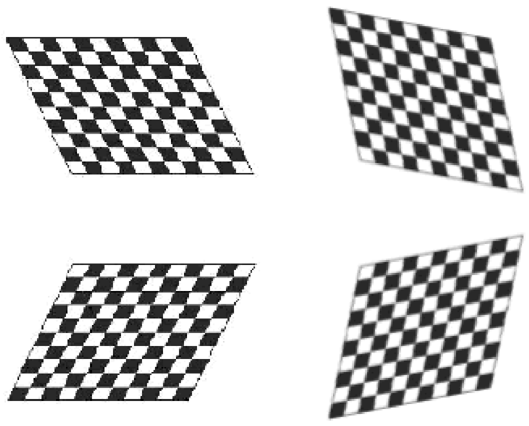
Original image	Image after affine transform
	

Figure 3.4: Affine transform.

Or equivalently

$$\delta\mathbf{a} = \mathbf{a}^{t+1} - \mathbf{a}^{t} = \mathbf{H}^{-1}\mathbf{b}$$

where $\mathbf{a}^{t+1}$ and $\mathbf{a}^{t}$ denote $\mathbf{a}$ at iterations t and $t+1$, respectively, $\mathbf{H}$ is 8×8 matrix equal to one-half times the Hessian matrix of E:

$$H_{kl} = \sum_{i=1}^{N} \frac{\partial e_i}{\partial a_k}\frac{\partial e_i}{\partial a_l},$$

and $\mathbf{b}$ is 8-element vector equal to minus one-half times the gradient of E:

$$b_k = -\sum_{i=1}^{N} e_i \frac{\partial e_i}{\partial a_k}.$$

Each iteration of the gradient descent consists of the following steps:

1. Compute the corresponding position (x'_i, y'_i) of (x_i, y_i) by adopted motion model.

2. Compute the error between corresponding pixels $e_i = I'(x'_i, y'_i) - I(x_i, y_i)$.

3. Compute the intensity gradient $\frac{\partial I'}{\partial x'}$ and $\frac{\partial I'}{\partial y'}$.

4. Compute the partial derivative of e_i w.r.t. the a_k using

$$\frac{\partial e_i}{\partial a_k} = \frac{\partial I'}{\partial x'}\frac{\partial x'}{\partial a_k} + \frac{\partial I'}{\partial y'}\frac{\partial y'}{\partial a_k}.$$

5. Repeat previous steps for all pixels within image boundaries and add its contribution to matrix $\mathbf{H}$ and vector $\mathbf{b}$.

6. Solve the equation $\delta\mathbf{a} = \mathbf{a}^{t+1} - \mathbf{a}^{t} = \mathbf{H}^{-1}\mathbf{b}$.

7. Update $\mathbf{a}$ by $\mathbf{a}^{t+1} = \mathbf{a}^{t} + \delta\mathbf{a}$.

8. Continue iterating until the stopping criterion is met.

3.4 Image Integration

Once the frames are aligned, they can be integrated to build a mosaic image. If two frames are mapped to different regions in the mosaic image it is easy to integrate by just putting them on proper locations, while it will need more consideration when the positions in mosaic image is mapped to by many frames. Several popular schemes can be used to deal with those positions that are mapped to by many frames:

1. Take the average value among those frames.
2. Take the median value among those frames.
3. Take the weighted median value or weighted average value among those frames.
4. Take the most recent value among those frames.

Another important issue is that an integer position in current frame may map into non-integer point in reference frame and therefore we have to deal with those non-integer points. Many popular interpolation techniques can be used to solve the problem:

1. The nearest neighborhood interpolation:

   This is a simplest and speediest interpolation algorithm in view of computational complexity. The value of non-integer point is replaced by the value of nearest integer point.

2. Bilinear interpolation:

   It uses a linear combination of four nearest integer points to produce a new value.

$$I(x_i, y_i) = a_0 + a_1 x_i + a_2 y_i + a_3 x_i y_i$$

   where a_0, a_1, a_2, a_3 can be solved by four nearest integer points.

3. Alternative techniques, such as using a larger resolution mosaic image proposed in [8].

3.5 Some Result of Mosaic Generation

In this section, we show a mosaic image using the algorithm described in previous section. Here, we use the 80th to 96th frames of Stefan sequence to generate mosaic image as

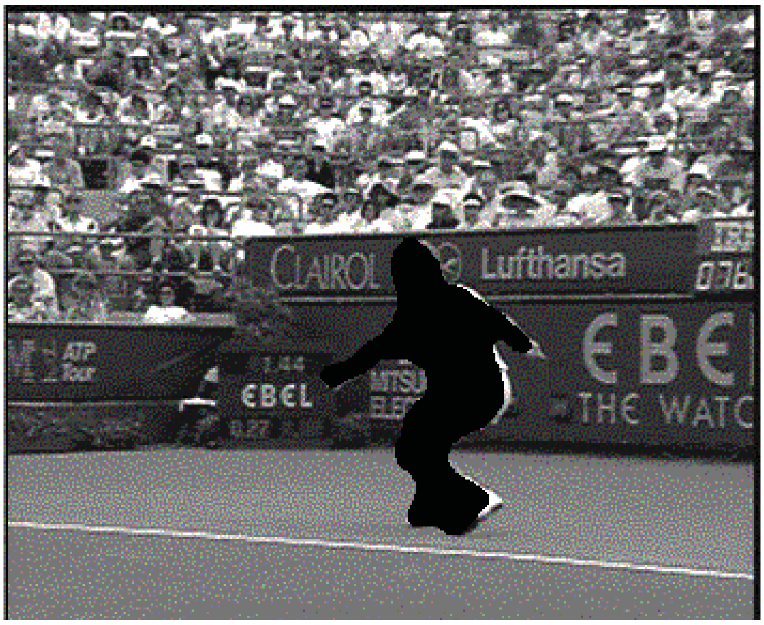


Figure 3.5: 80th frame.

shown in Figures 3.5 and 3.6, where we assume that we have an object mask already. The affine model is adopted to model camera motion and we use the bilinear interpolation to deal with non-integer point. The values of overlapped pixels are replaced by most recent values. The final result is shown in Figure 3.7.

With more parameters, the motion model can describe more camera motion while the complexity of finding parameters is also rising. The perspective model is the most accurate model, but it also has highest computational complexity. Besides, in our observation the a_6 and a_7 of perspective model are usually smaller than other parameters and therefore the values of a_6 and a_7 are usually less accurate. According to this observation, we tend to use the affine model in the procedure of global motion estimation. The translation model which only need two parameters is also under consideration, but according our experiment the accuracy of result is not acceptable.

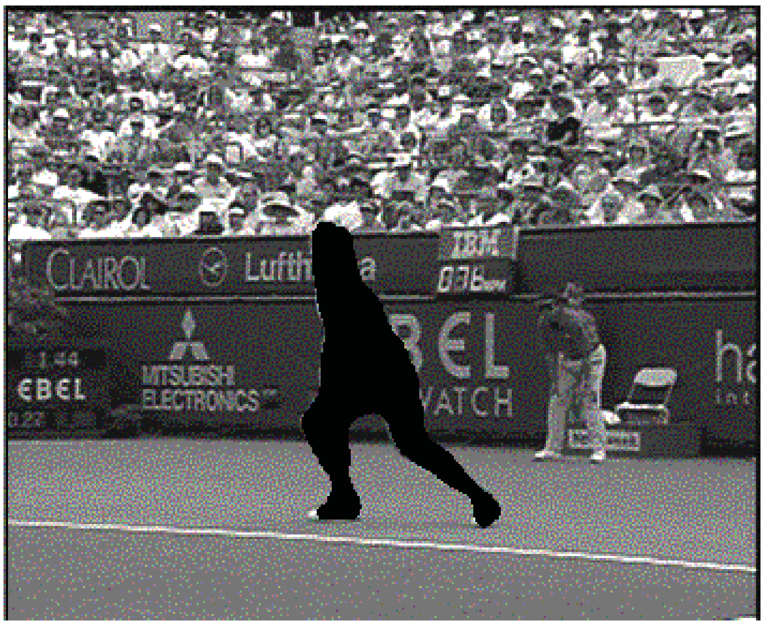


Figure 3.6: 96th frame.

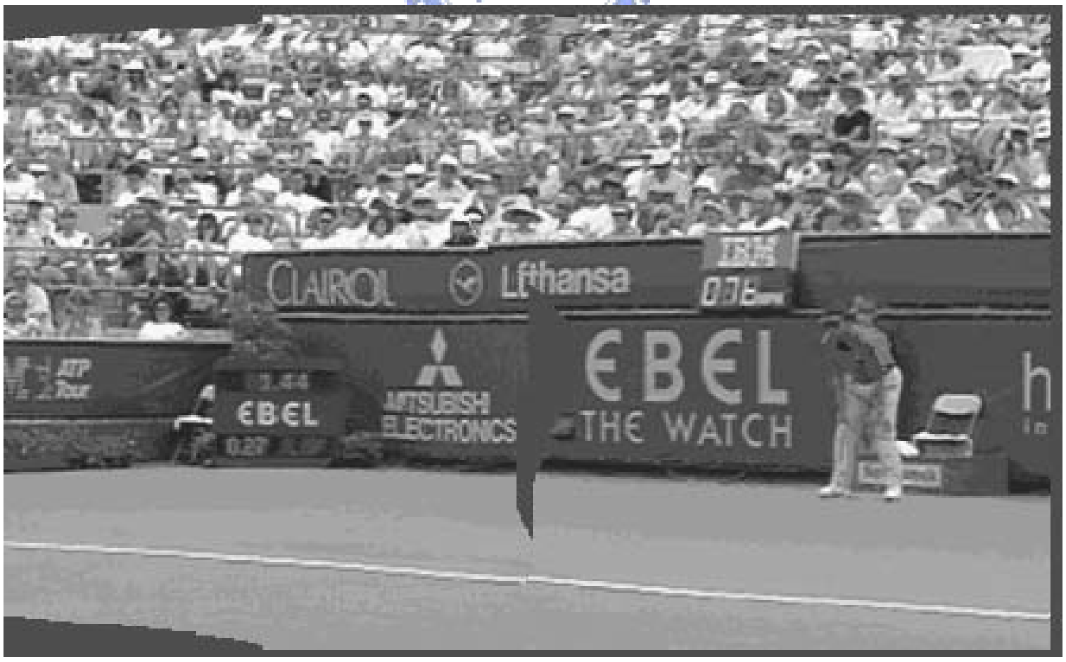


Figure 3.7: Mosaic image.

Chapter 4

The Proposed Segmentation Method

4.1 System Overview

Our segmentation system is a background subtraction-based scheme. The block digram is shown in Figure 4.1. To start, we estimate the camera noise and the following thresholds are decided according to estimated camera noise. We use the temporary foreground mask and short-term background to generate the stationary background buffer. The object mask can be obtained by finding the difference between current frame and stationary background buffer. If the scene change occurs, we have to apply global motion estimation to generate the panorama background buffer and recover the stationary background buffer.

In this system, most of these modules are used for processing each frame except global motion estimation. The global motion estimation works only when scene change occurs since the global motion estimation is a time-consuming process and it is useless when camera is fixed. The details of this system are discussed in the following sections.

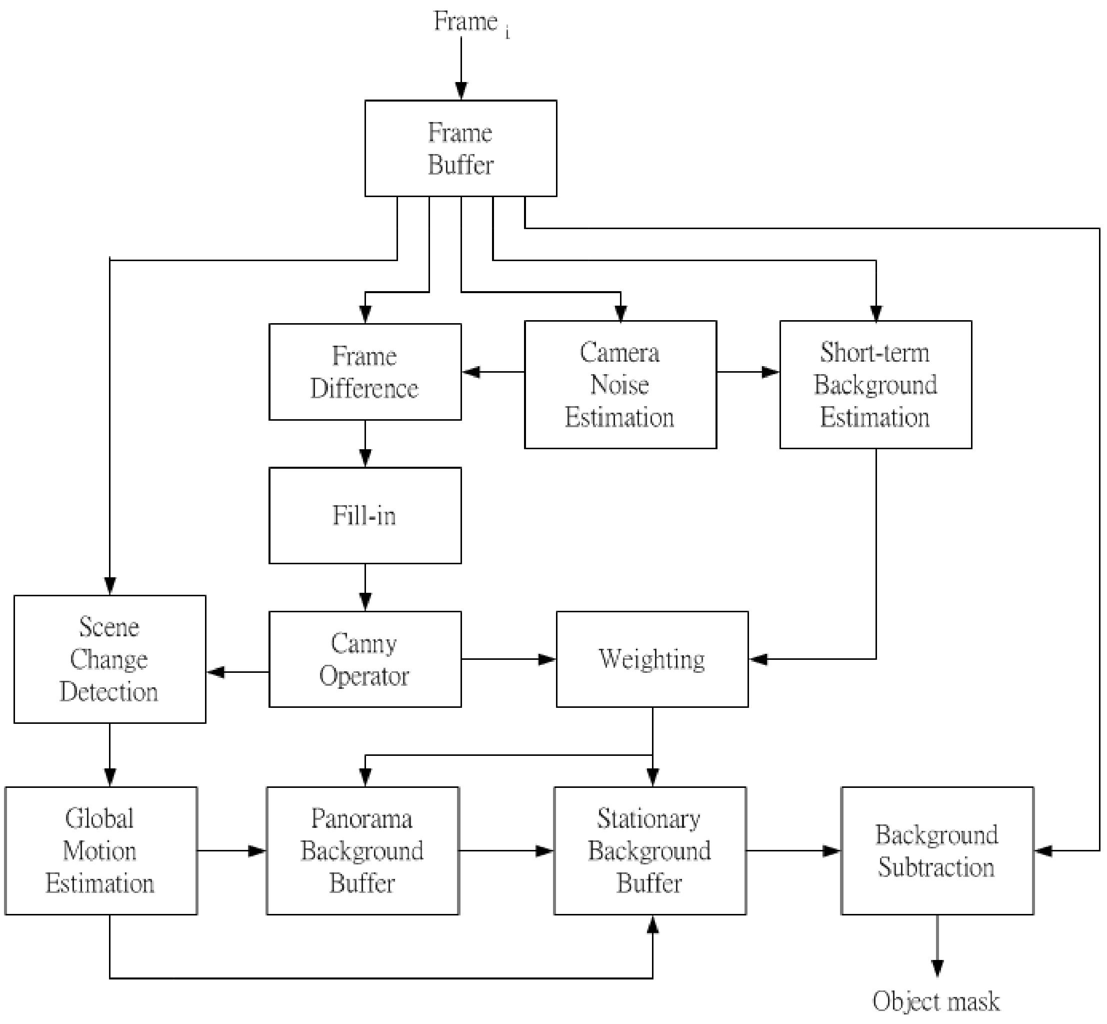


Figure 4.1: The block diagram of proposed method.

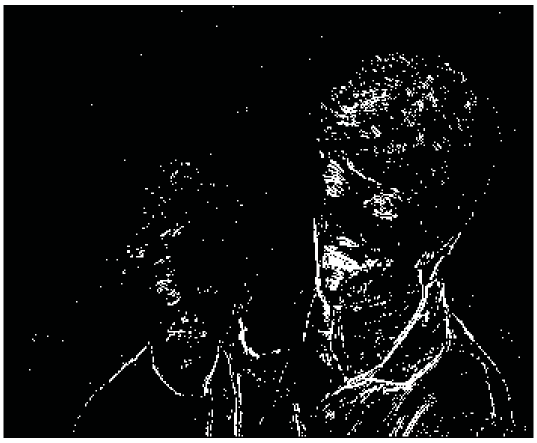


Figure 4.2: A thresholded frame difference map of the Mother-and-Daughter sequence.

4.2 Two-Stage Noise Estimation

4.2.1 Influence of Noise

In this system, the image is captured by camera and then we get the initial image from the output of camera. In the process of capturing, the image may suffer from camera noise and therefore the stationary background usually has some difference between successive capturing. In general, the larger camera noise makes the segmentation more difficult to achieve. For example, when change detection-based technique is applied, the frame difference map of larger noise sequence (Figure 4.2) includes more background pixels than smaller noise sequence (Figure 4.3). It is apparent that the larger noise sequence needs more processing to obtain a finer object mask.

4.2.2 Motive of Noise Estimation

In the following steps of the system, we need many thresholds or parameters to make decisions and those thresholds are usually adjusted to against the influence of noise. We can adjust those parameter empirically but it is inconvenient since we have to tune them for different situations and it usually needs some experiences. In order to reduce the complexity of threshold decision, those parameters in the following steps are adjusted based on estimated camera noise.

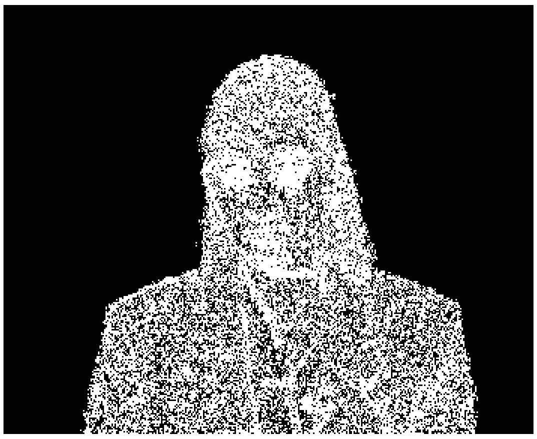


Figure 4.3: A thresholded frame difference map of the Akiyo sequence.

4.2.3 Camera Noise Model

In the thesis, we assume the difference d_k of stationary pixels between successive frames obeys a zero mean Gaussian distribution $N(0, \sigma)$ with variance σ^2, that is,

$$p(d_k|H_0) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\{-\frac{d_k^2}{2\sigma^2}\}$$

where H_0 denotes the null hypothesis. As in [2], since the camera noise is uncorrelated between different frames, the variance σ^2 is equal to twice the variance of the assumed Gaussian camera noise distribution.

4.2.4 Procedure for Noise Estimation

In order to estimate the variance σ^2, the sample space should include those pixels belonging to stationary background and exclude pixels belonging to moving objects. Our idea to discriminate the two kind of pixels is based on the observation which can be seen in Figure 4.4. The lighter pixels which represent larger difference are usually lumped together or are distributed like a strip when they are introduced by moving objects. On the other hand, the larger frame differences caused by camera noise are usually randomly distributed. Hence, we reject the pixels whose neighbors have larger frame difference from the sample space during noise estimation.

We use the similar mask of [11] to find out those pixels that belong to moving objects. For each pixel, we consider the four directional sums in frame difference map as shown

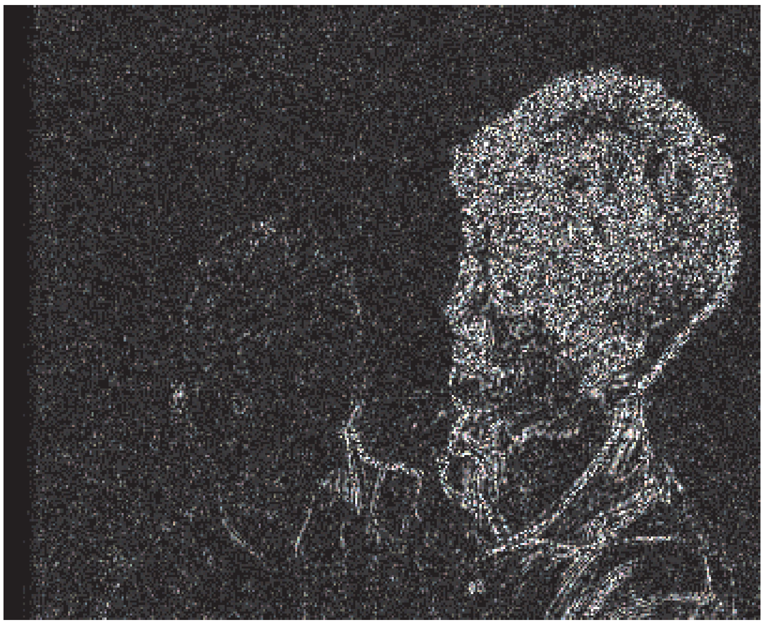


Figure 4.4: A frame difference map of Mother-and-Daughter sequence.

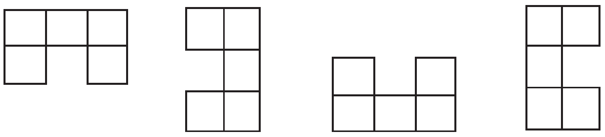


Figure 4.5: Four masks of directional sums.

in Figure 4.5. If one of the four directional sums is larger than certain threshold, we can assume that the pixel belongs to a moving object.

The following problem is how we choose the threshold. Up to the present, we can only calculate variance σ_G^2 of frame difference of entire frame, and therefore it is natural that we initially adjust the threshold based on σ_G^2. If one of the four directional sums of a pixel is larger than $\alpha\sigma_G^2$, the pixel is classified to pixels influenced by moving objects. After we remove those pixels influenced by objects, those remaining pixels are used to estimate σ^2. In order to verify the result of our method, we first manually choose the pixels belonging to stationary background to estimate the variance σ^2. In Figures 4.6 and 4.7, the white areas are chosen to estimate σ^2 and the estimation result is regarded as exact. As we can

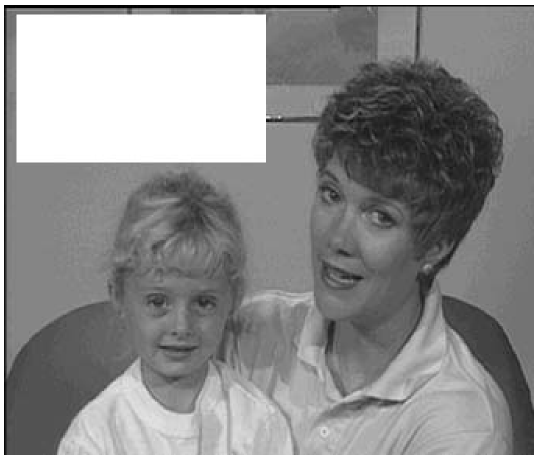


Figure 4.6: Mother-and-Daughter sequence.

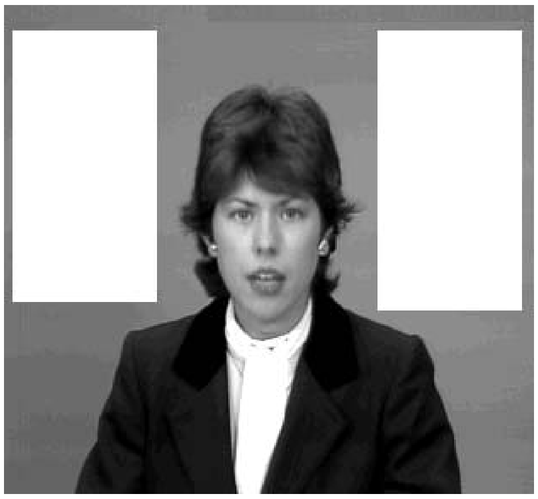


Figure 4.7: Claire sequence.

see in Figures 4.8 and 4.9, this method can effectively remove most pixels influenced by moving objects.

It is obvious that the results in Figures 4.8 and 4.9 are still influenced by moving objects, because we adjust the threshold based on variance σ_G^2 of entire frame which has high relationship with moving objects. In order to reduce this problem, we use a two-stage noise estimation in this system. In the first stage, we use the $\alpha\sigma_G^2$ to be the threshold and get the variance σ_1^2 of stage one. In the second stage, we use the $\beta\sigma_1^2$ as the threshold and then we can obtain the final result σ_2^2 of stage two. The final result is shown in Figures 4.10 and 4.11. It can be seen that the result of two stage method is closer to exact value.

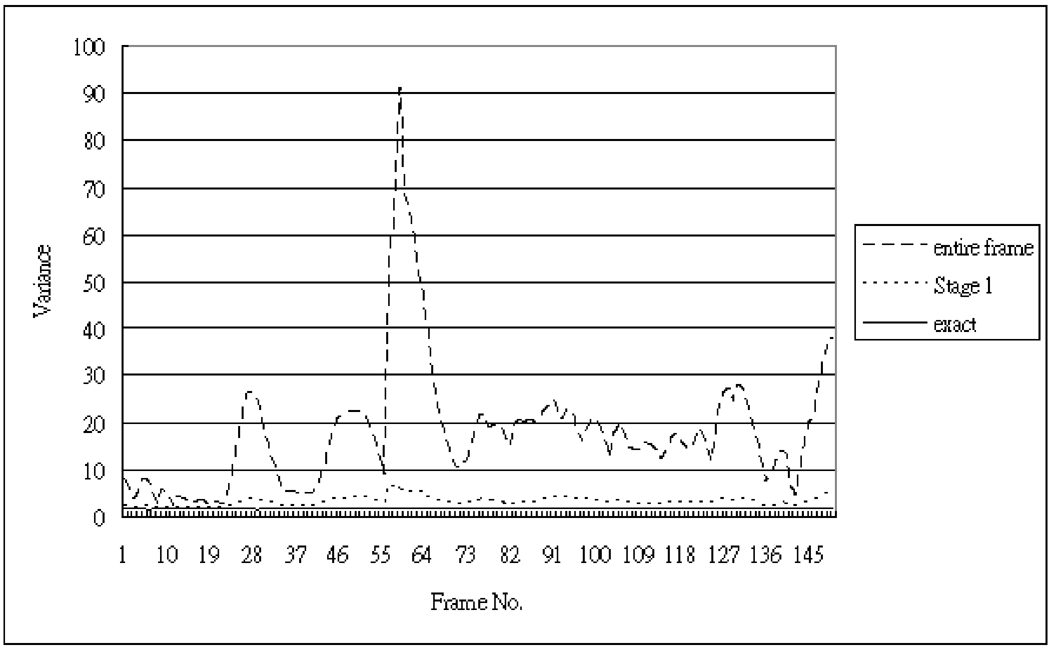


Figure 4.8: Noise estimation of Mother-and-Daughter sequence.

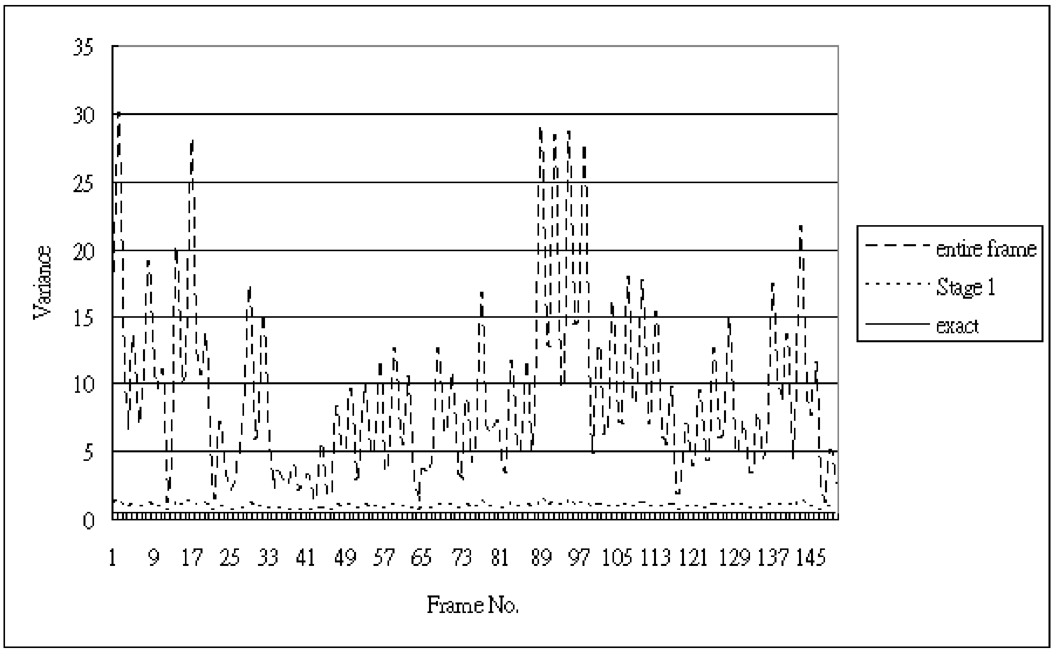


Figure 4.9: Noise estimation of Claire sequence.

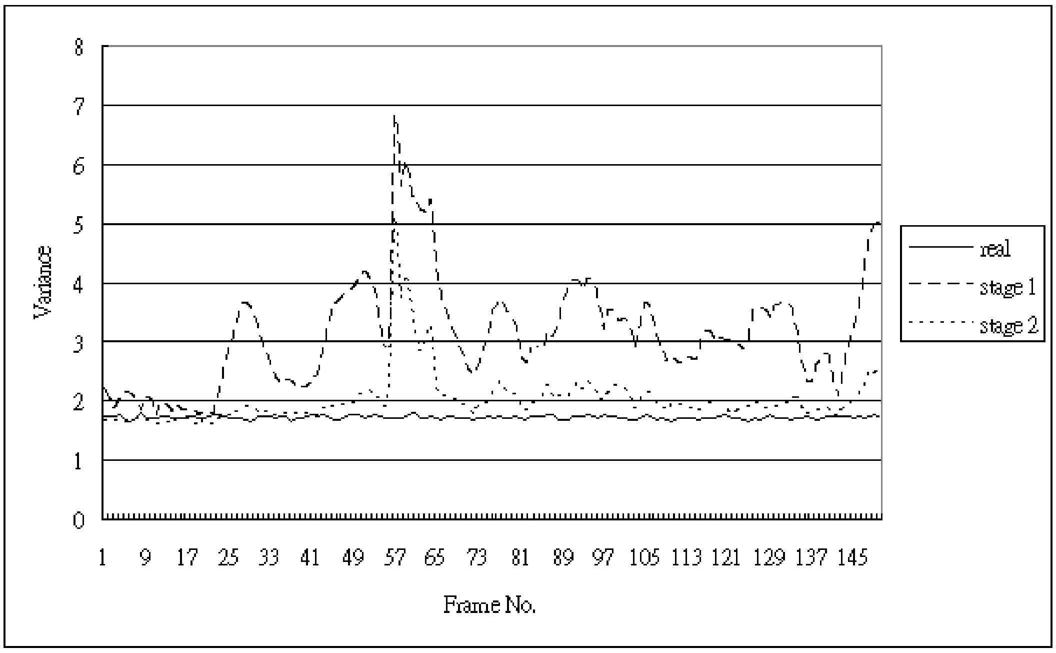


Figure 4.10: Noise estimation of Mother-and-Daughter sequence for the two-stage method.

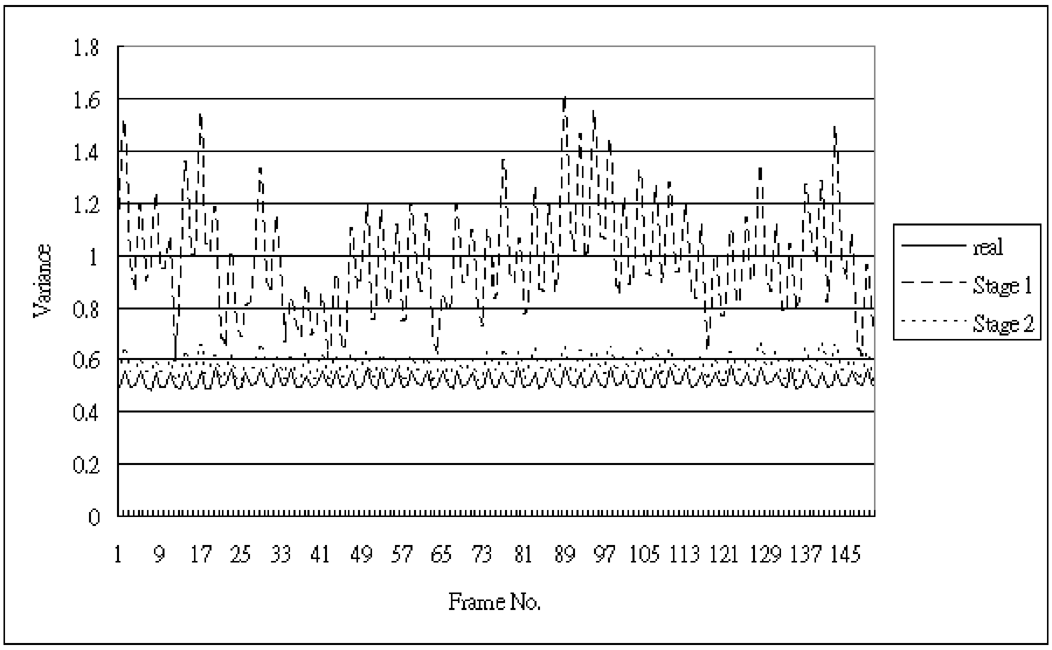


Figure 4.11: Noise estimation of Claire sequence for the two-stage method.

4.3 Temporary Foreground Mask

In this section, we will generate a temporary foreground mask and then the mask is used in the stationary background buffer, scene change, and global motion estimation. In this stage, we use change detection-based technique to obtain a rough mask. The major advantage of this technique is that the frame difference can be easily and fast gotten but many time-consuming methods for a more accurate object boundary usually make the whole system more complex and slower as shown in [1] and [2]. In this thesis, we only need to get a rough mask in this stage, and therefore those time-consuming methods can be neglected to keep the speed of whole system.

4.3.1 Get Initial Object Mask

At first, we use the 3×3 window to calculate the mean of squared frame difference for each pixel. If the result is larger than threshold, the pixel is classifed as in a moving object. On the other hand, a pixel is classified as background when the result is smaller than threshold. The threshold here is adjusted based on the camera noise, that is, $\gamma\sigma^2$. An example of thresholded frame difference map is shown in Figure 4.12. In the second step, we use the fill-in technique proposed in [1] to get a rough mask. At first they assign the pixels between the first and last white points of Figure 4.12 to white points for each row. This procedure is then repeatted for each column and once more for each row. The step-by-step results are shown in Figures 4.13, 4.14, and 4.15, respectively.

4.3.2 Refine Initial Object Mask

Frequently, a rough mask in previous section is enough for following stages while it may need more improvement in some cases. In Figure 4.16, for instance, there are two persons sitting side by side. Since the fill-in technique always marks the region between left and right boundaries, the background between the two persons is always regarded as objects. Although this problem can be mitigated in the following stages, it will be very helpful if the mask here is more accurate.

In this stage, we use the edge information to correct the initial mask and the Canny

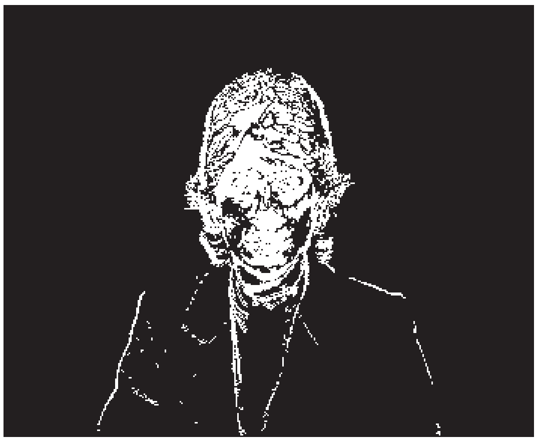


Figure 4.12: Threshold frame difference map of Claire sequence.

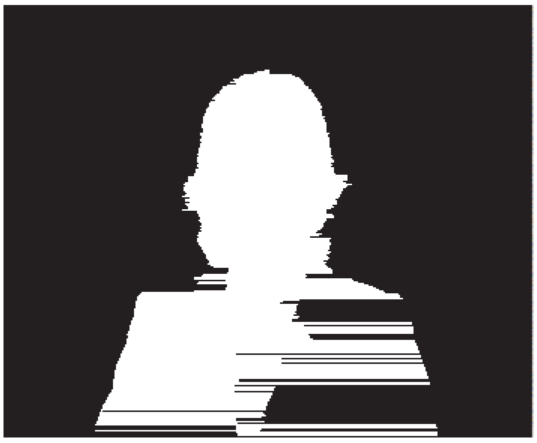


Figure 4.13: Fill-in for each row.

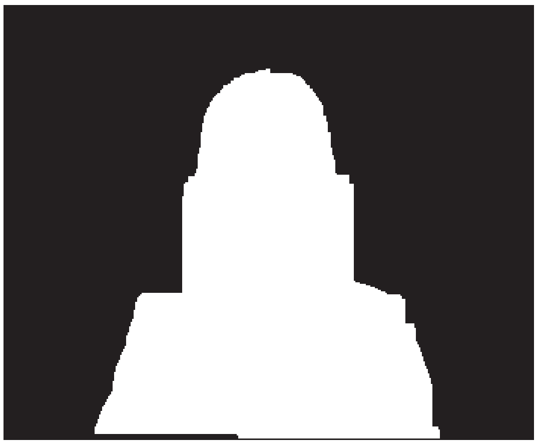


Figure 4.14: Fill-in for each column.

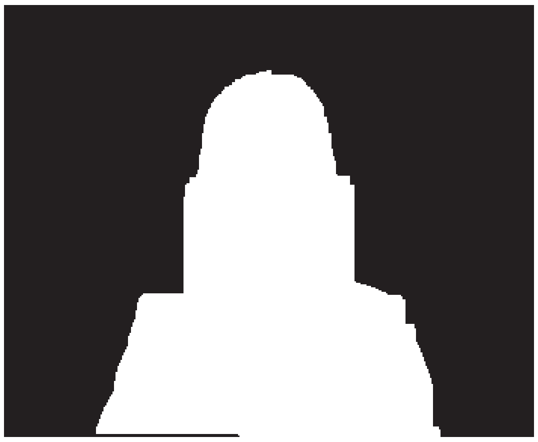


Figure 4.15: Second fill-in for each row.

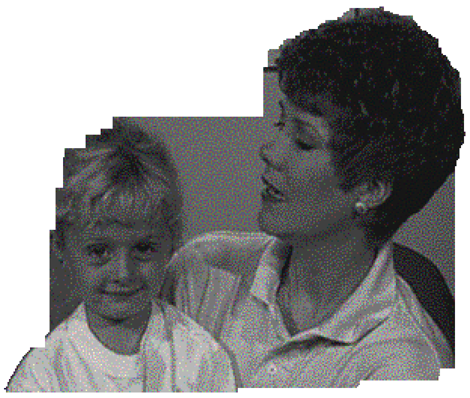


Figure 4.16: Initial object mask of Mother-and-Daughter sequence.

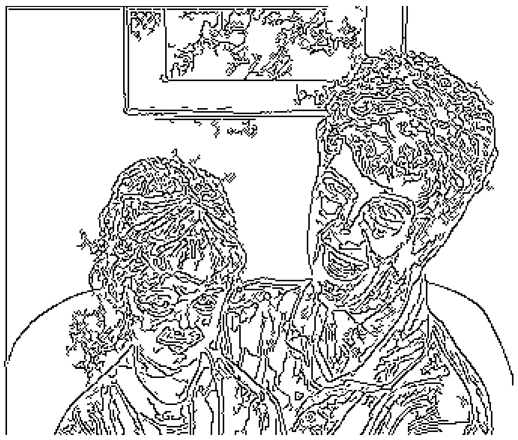


Figure 4.17: Edge map of Mother-and-Daughter sequence.

operator proposed in [10] is adopted to get edge information. The operator performs a gradient operation on the image which convoluted by gaussian filter and then nonmaximum suppression is applied to thin the edge. In the last step, the thresholding operation with hysteresis is used to find and link edges. The thresholding operation including two thresholds: high-threshold and low-threshold. Pixels whose gradient is larger than high-threshold are regarded as edges and pixels whose gradient is smaller than low-threshold are regarded as non-edges. Pixels whose gradient is between high-threshold and low-threshold need to check their neighbors. If one of its neighbors is regarded as edge, these pixels are classified to edge. The edge map after applying Canny operator is shown in Figure 4.17. The related code of canny operator is obtained from [12].

The way to refine the initial object mask is shrinking the initial mask to fit the edge map. The initial mask, edge map and shrunk mask are shown in Figures 4.16, 4.17 and 4.18, respectively, for Mother-and-Daughter sequence. In these figures, we can see that the edge map includes many background edges and those background edges usually interfere with the final result. To reduce the influence of background edge, we use a buffer to store those background edges. When a position of edge map always has edge, we assume that there is a background edge in the position. The result after removing the background edge can be seen in Figure 4.19 and the final object mask is shown in Figure 4.20. According to the Figures 4.16 and 4.16, we can see that the remaining background due to background edge can be effectively removed.

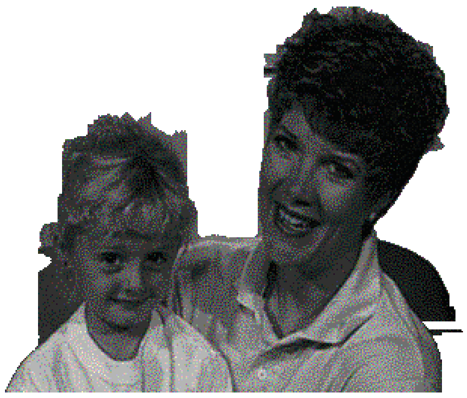


Figure 4.18: Refined mask of Mother-and-Daughter sequence.

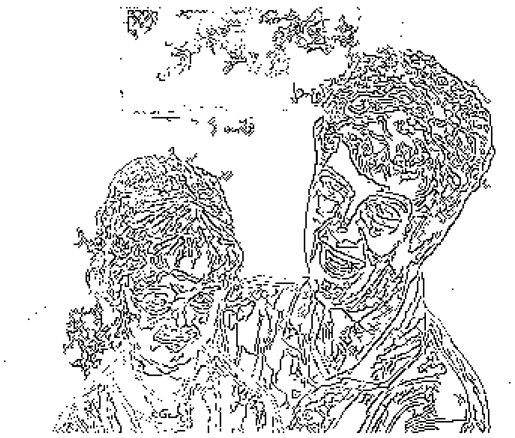


Figure 4.19: Edge map after removing background edges in Mother-and-Daughter sequence.

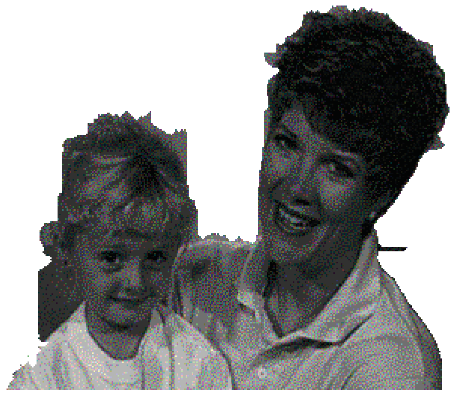


Figure 4.20: Final object mask of Mother-and-Daughter sequence.

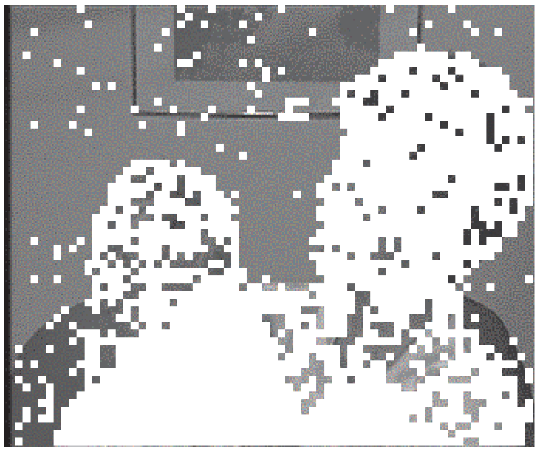


Figure 4.21: Result of short-term background estimation.

4.4 Short-Term Background Estimation

The simplest way to judge whether the value of a pixel is background is to check the frame difference at this location. Since the moving objects will cause larger frame difference, we can assume that the value of pixel belongs to background at this location when the frame difference at this location is very small from start to finish. For real-time application, it is impossible to make a decision after whole data is collected from beginning to end. In the system, we regard the value of pixel as background when its frame difference is samll for some consecutive frames. The major disadvantage of this method is that it is easier to make a wrong decision when the time of observation is not long enough and therefore the obtained background here is not reliable at some pixels.

We consider six consecutive frames $f_k(i)(1 \leq k \leq 6)$ as time of observation and a 3×3 windows is used to calculate frame difference $d_m(i) = f_6(i) - f_m(1 \leq m \leq 5)$ for each location i in a frame. For every location i, we caluculate the mean and variance of $d_m(i)(1 \leq m \leq 5)$. If the variance is smaller than threshold, it means the changes between the six frames are small and we can regard the value of pixel at location i of sixth frame as background. The threshold here is also based on camera noise, that is, $\lambda\sigma^2$. The result is shown in Figure 4.21 for the earlier example.

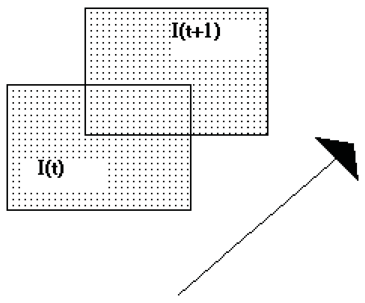


Figure 4.22: The influence of flat inner region.

4.5 Construct Stationary Background Buffer

In this stage, the information from short-term background estimation and temporary foreground is considered to generate the statioanry background buffer.

Most of wrong decisions in short-term are due to flat inner regions as shown in Figure 4.22. If an object has a large flat inner region, the overlap between successive moving objects is still stationary and is easily regarded as background. In order to reduce the influence of flat inner regions, we use the temporary foreground mask to weight every pixel before we put the short-term background into final background buffer.

A weighting mask is shown in Figure 4.23, where the black region represents reliable background and has higher weighting while the white region represents objects and has zero weighting. If a pixel is inside gray region which means the pixel is regarded as background in short-term background and its location is inside the temporary foreground mask, it is easier to suffer from flat inner region problem and we will give it a lower weighting. We accumulate the weighting for every position and the short-term background is put into stationary background buffer when the accumulated weighting meets threshold. The lower-weighting points can still become a real background when the these points are always regarded as short-term background for longer time to reduce wrong decisions due to flat inner region. The final background buffer after we have observed 280 frames is shown in Figure 4.24.

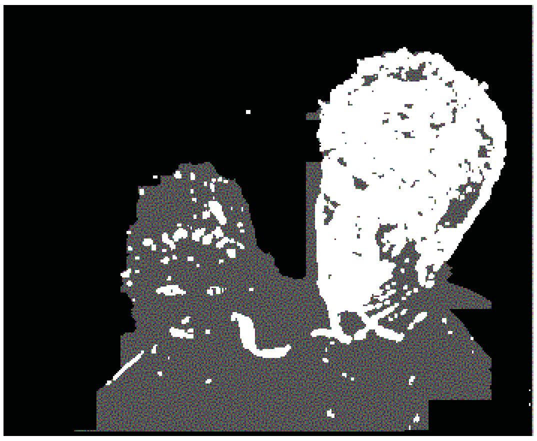


Figure 4.23: The weighting mask of Mother-and-Daughter sequence.

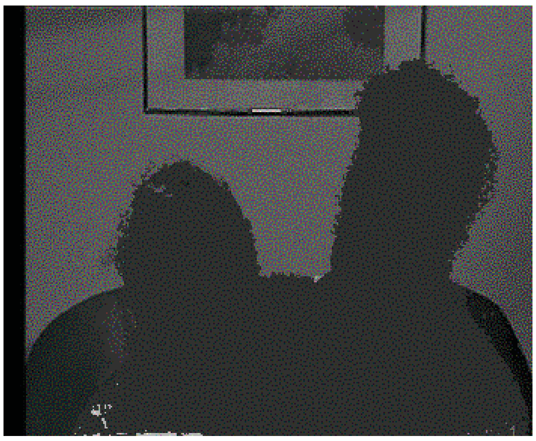


Figure 4.24: Final background buffer after observing 280 frames.

4.6 Deal with Camera Motion

In the background subtraction-based techniques, the background should be stationary. If the camera changes its position, the background buffer should be reset and information of new background should be gathered. In general, there are large overlapping regions between old and new backgrounds. In the thesis, we will use the overlapping regions by image mosaic technique to speed up background reconstruction.

4.6.1 Scene Change Detection

In the first step, we have to know whether camera motion occurs. Here, a scene change detection is used to detect the camera motion. When the frame difference between background at different times is large, we assume that the camera motion has occured. The background here is obtained by excluding objects with temporary foreground mask. Since a flat region usually has no frame difference when small camera motion occurs, we only consider the regions near edges.

4.6.2 Global Motion Estimation

After scene change occurs, we have to find the camera motion. The way to find the camera motion is using global motion estimation and the block diagram is shown in Figure 4.25. The hierarchical architecture of motion estimation of [6] and [9] is used in this system. The advantage of hierarchical architecture is that it can overcome large displacement and reduce computational complexity.

The goal of global motion estimation is to minimize the sum of squared differences E between current frame I and reference frame I':

$$E = \sum_{i=1}^{N} e_i^2$$

where $e_i = I'(x'_i, y'_i) - I(x_i, y_i)$. Below, we explain the method in detail.

1. Motion model

In this thesis, the affine model is adopted for camera moving, that is,

$$x_i' = a_0 + a_2 x_i + a_3 y_i,$$

$$y_i' = a_1 + a_4 x_i + a_5 y_i,$$

where $(a_0, ..., a_5)$ are motion parameters, and (x_i, y_i) and (x_i', y_i') are positions of current frame I and reference frame I', respectively.

2. Initial matching

   The gradient descent method needs an initial value of $a_k (0 \leq k \leq 5)$. If the initial value is far from final converged value, it is easy to get a local minimum solution. For that reason, the initial matching is needed to find a better initial value than arbitrary guess.

   In the thesis, we use the step search to fastly obtain the motion vector. The search range is ± 15 in both coordinates and therefore the range of full size is ± 60. After finding motion vector, we can obtain initial value of a_0 and a_1. The others are set as $a_2 = a_5 = 1$ and $a_3 = a_4 = 0$.

3. Gradient descent

   The detail of gradient descent is described in chapter 3. Besides, in the first iteration of each level, the histogram of $|e_i|$ is computed and find a threshold T such that the number of $|e_i|$ bigger than T are about 15% of considered pixels. In the following iterations, those pixels whose $|e_i|$ larger than T are excluded in gradient descent. In this thesis, we use the stopping criterion for gradient descent: at most 34 iterations are carried out at each level. The number of iterations is set by observing the speed of convergence in Stefan sequence and the related results from [6]. The transform between (x_i, y_i) and (x_i', y_i') is usually non-integer and therefore the bilinear interpolation is used here.

4. Projection

   The projection of motion parameters from one level to the next one is multiplying a_0 and a_1 by two, and others are keeping the same.

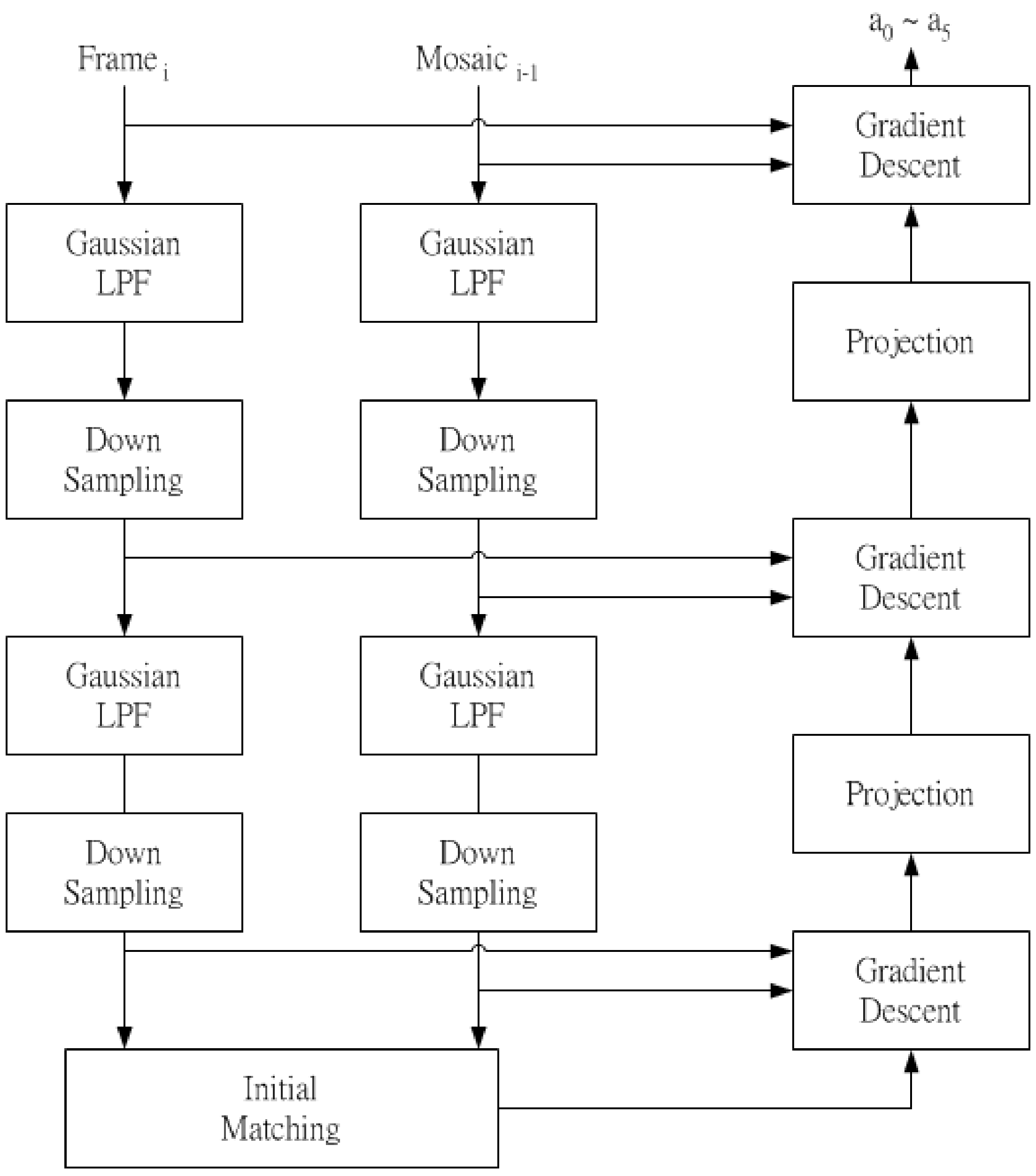


Figure 4.25: Global motion estimation.

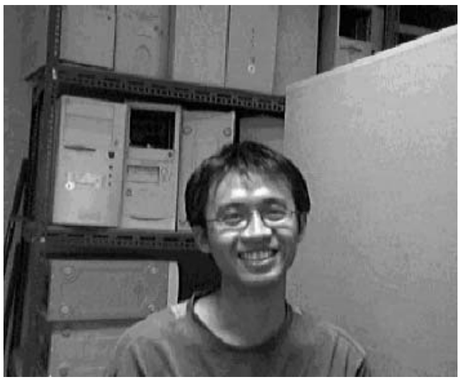


Figure 4.26: Image before camera moving.

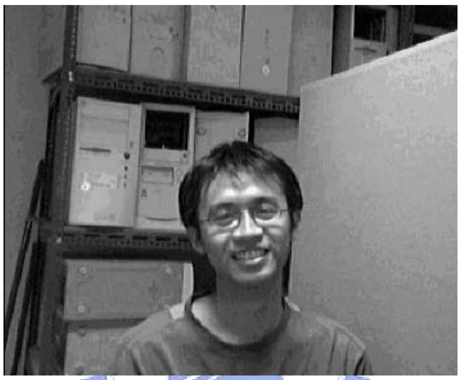


Figure 4.27: Image after camera moving.

4.6.3 Panorama Background and Background Recovery

After we have obtained the camera motion, the background can be stored in panorama background buffer by corresponding motion parameters. When camera motion occurs, the stationary background buffer can be rebuilt from panorama background quickly. Here the non-integer transform still exists and the bilinear interpolation is adopted to deal with this problem.

For example, Figures 4.26 and 4.26 represents images before and after camera motion occurs, respectively. The camera is horizontally rotated to left side of origin position. The panorama background buffer before camera moving is shown in Figure 4.28 and the recovered stationary background buffer after camera moving is shown in Figure 4.29. The stationary background buffer and panorama background buffer in the seventh frame since camera moved are shown in Figures 4.30 and 4.31, respectively.

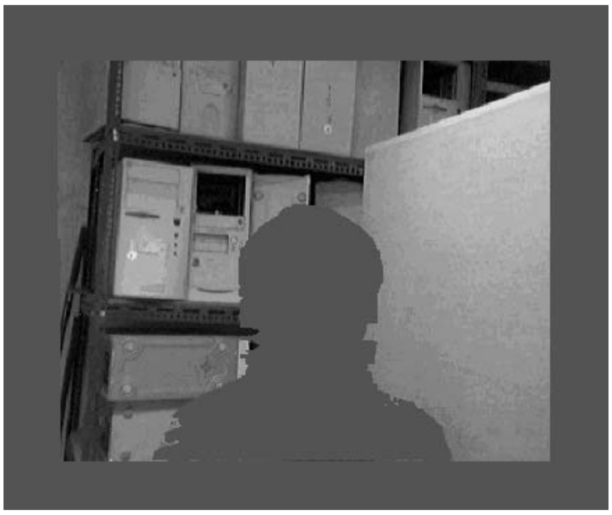


Figure 4.28: Panorama background buffer before camera moving.

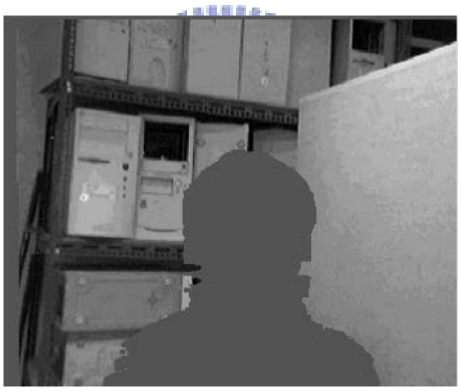


Figure 4.29: Recovered stationary background buffer after camera moving.

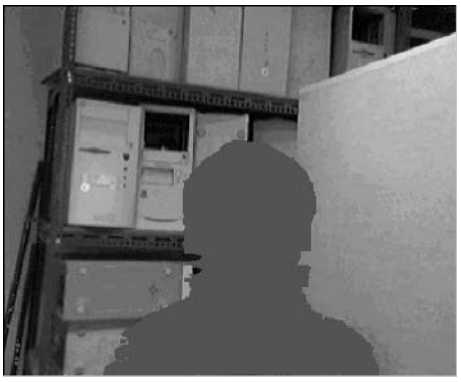


Figure 4.30: Stationary background buffer in the seventh frame since camera moving.

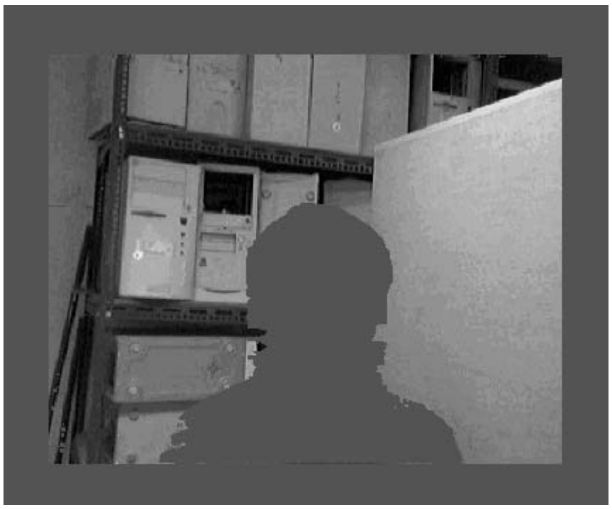


Figure 4.31: Panorama background buffer in seventh frame since camera moving.

4.6.4 Background Subtraction

The final object mask is obtained by finding difference between current frame and stationary background buffer. For a better result, both difference in luminance and difference in chromanace between the two frames are considered. In general, the background of current frame may suffer from light change and shadow and the stationary background may contain some wrongly identified background pixels. Therefore, using subtraction between current frame and background may still leave some background. For this reason, we have to remove the small region after subtraction. There are two steps to remove small regions. First, remove the small regions outside object mask. Second, remove the small regions inside object mask. In the first step, we check the connected length of object mask for each row and remove pixels whose connected length is less than threshold. Then, the processing in used for each column. After we remove the regions outside object mask, we have to fill the wrongly identified object pixels which usually looks like a hole inside object mask and the method is similar to first step.

An example of the current frame and background is shown in Figures 4.32 and 4.33. The result after subtraction and thresholding is shown in Figure 4.34. The final result with small region removing is shown in Figure 4.35.

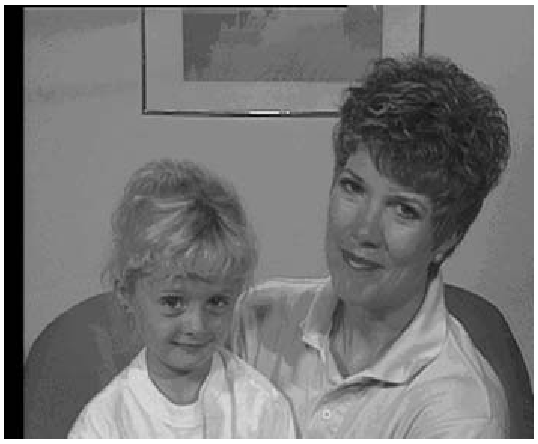


Figure 4.32: Frame 255 of mother and daughter sequence.

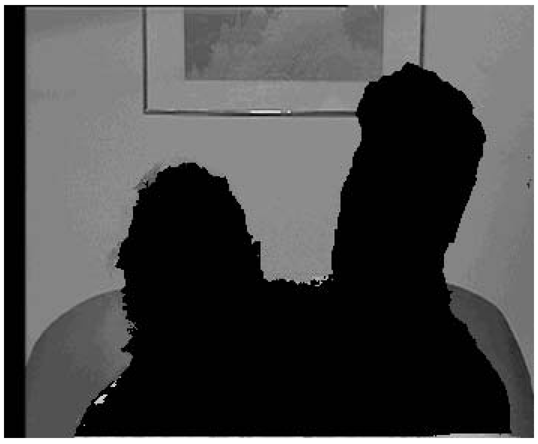


Figure 4.33: Stationary background buffer.

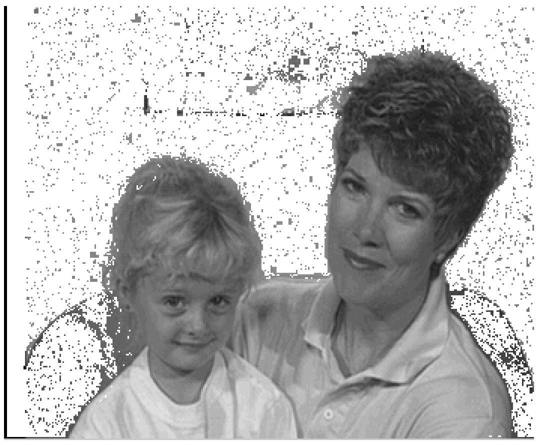


Figure 4.34: Mask after subtraction and thresholding.

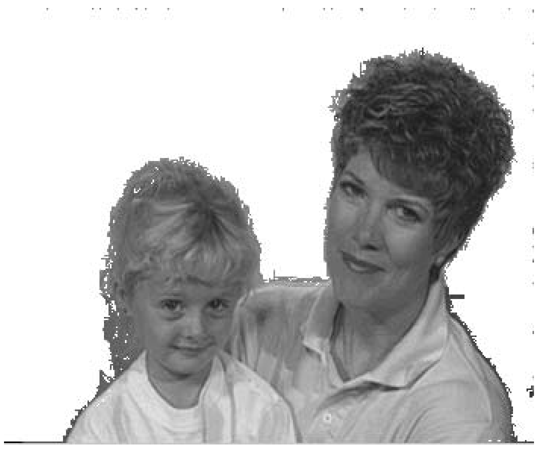


Figure 4.35: Final object mask.

4.6.5 Conclusion

In this section, we make a summary of proposed method described above. In the first step, the two-stage noise estimation is used to estimate the camera noise and as shown in chapter 4.2 this method can effectively remove the influence of moving objects. In the second step, we obtain a short-term background which usually suffers from flat inner region problem and therefore a temporary foreground mask is introduced to overcome this problem. For a better result, we use the edge information to refine the temporary foreground mask. After we obtain the short-term background and temporary foreground mask, we can use both of them to establish the stationary background buffer and then the object mask is obtained by subtraction between current frame and stationary background buffer.

If the camera is fixed, the procedure described above is enough to obtain a object mask. When camera motion occurs, we estimate the global motion and recover the stationary background from panorama background buffer by the estimated global motion.

Chapter 5

Overall System Architecture

The overall architecture of our segmentation system is shown in Figure 5.1. Here, we need a digital camera to capture images, a personal computer to control the system and to segment the image, and a displayer to show the final result.

There are two modes of application programming in Windows OS: Console mode which uses the file I/O and GUI (Graphical User Interface) mode. The GUI mode is suitable for our application and it is implemented by Windows SDK (Software development Kit) which is develope by Microsoft for high-level computer languages to easily implementation GUI mode.

The system block diagram is shown in Figure 5.2. The detail of segmentation is described in previous chapter and therefore we focus on video capturing, result displaying, and control in this chapter.

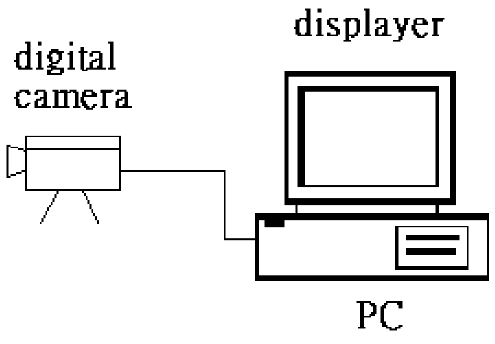


Figure 5.1: circumstance of implementation.

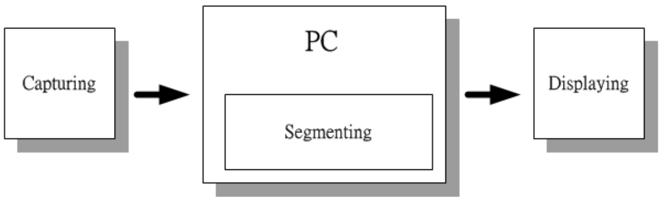


Figure 5.2: system block diagram.

5.1 Video Capturing

5.1.1 Video for Windows

In this system, the input image is captured by digital camera. To control the operation of capturing, a standard video capturing method, named VfW (abbreviation of Video for Windows), in the Microsoft OS is adopted. Video for Windows version 1.0 was released in November 1992 for the Windows 3.1 operating system and was optimized for capturing movies to disk [14] This SDK provides applications with a simple, message-based interface to access video and waveform-audio acquisition hardware and to control the process of streaming video capture to disk. Besides, VfW helps with connectivity to device driver and retrieve the capability and information of it.

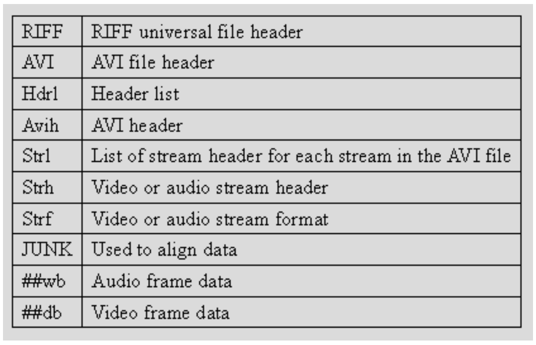
5.1.2 AVI Format

In VfW, AVI is the mostly used format. The captured raw frame is embedded in an AVI file which can be extracted for segmentation input.

AVI stands for Audio-Video Interleaved. Figure 5.3 shows the hierarchical structure. Refer to header file (vfw.h in Visual C++) for complete information about parsing AVI file. To extract video data, we use a file parser that simply locate $\sharp\sharp db$ and copy suitable length of data following that.

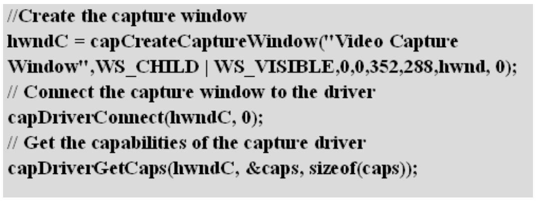
5.1.3 Implementation of Capture

The implementation of capture is aided by a free application called AVICap from [15]. It contains three steps:



RIFF	RIFF universal file header
AVI	AVI file header
Hdrl	Header list
Avih	AVI header
Strl	List of stream header for each stream in the AVI file
Strh	Video or audio stream header
Strf	Video or audio stream format
JUNK	Used to align data
##wb	Audio frame data
##db	Video frame data

Figure 5.3: AVI header.



```
//Create the capture window
hwndC = capCreateCaptureWindow("Video Capture
Window",WS_CHILD | WS_VISIBLE,0,0,352,288,hwnd, 0);
// Connect the capture window to the driver
capDriverConnect(hwndC, 0);
// Get the capabilities of the capture driver
capDriverGetCaps(hwndC, &caps, sizeof(caps));
```

Figure 5.4: Related code for creating a capture window.

1. Create capture handle:

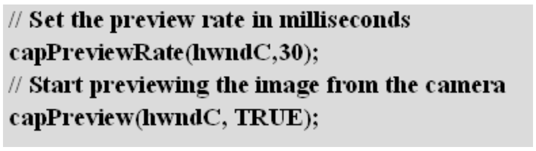
   An AVICap capture window handles the details of streaming audio and video capture to AVI files and it provides a flexible interface for applications. The video capture can be add to application by the code shown in Figure 5.4.

2. Parameter modification:

   After initializing driver window handler, some fundamental parameters should be confirmed to ensure captured data fit system requirement, such as the code shown in Figure 5.5.

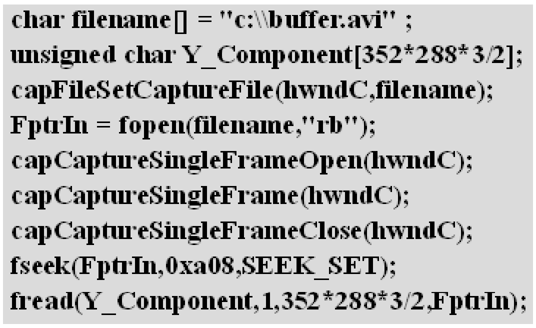
3. Capture operation:

   In this step, we start to capture image from digital camera and related code is shown



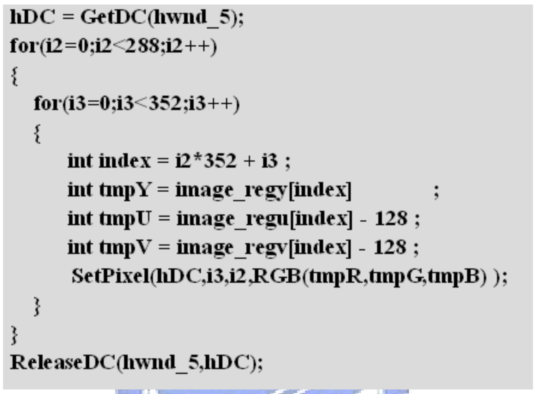
```
// Set the preview rate in milliseconds
capPreviewRate(hwndC,30);
// Start previewing the image from the camera
capPreview(hwndC, TRUE);
```

Figure 5.5: Related code for parameter modification.



```
char filename[] = "c:\\buffer.avi" ;
unsigned char Y_Component[352*288*3/2];
capFileSetCaptureFile(hwndC,filename);
FptrIn = fopen(filename,"rb");
capCaptureSingleFrameOpen(hwndC);
capCaptureSingleFrame(hwndC);
capCaptureSingleFrameClose(hwndC);
fseek(FptrIn,0xa08,SEEK_SET);
fread(Y_Component,1,352*288*3/2,FptrIn);
```

Figure 5.6: Related code for capture operation.



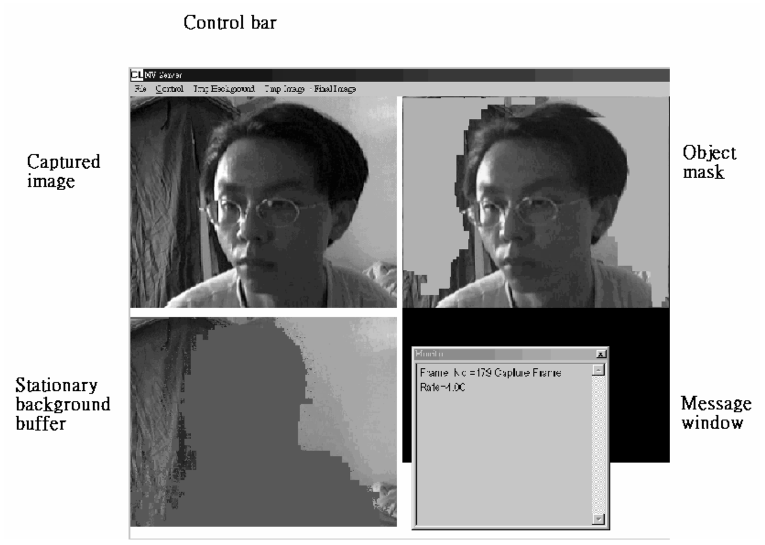
```
hDC = GetDC(hwnd_5);
for(i2=0;i2<288;i2++)
{
   for(i3=0;i3<352;i3++)
   {
        int index = i2*352 + i3 ;
        int tmpY = image_regy[index]            ;
        int tmpU = image_regu[index] - 128 ;
        int tmpV = image_regv[index] - 128 ;
         SetPixel(hDC,i3,i2,RGB(tmpR,tmpG,tmpB) );
   }
}
ReleaseDC(hwnd_5,hDC);
```

Figure 5.7: Related code for displaying.

in Figure 5.6. Here, the captured image which is AVI format is stored in a buffer and then the required video data is extracted from the buffer. Finally, the extracted data is sent to the module of video segmentation.

5.2 Result Displaying

After we finish the video segmentation, we need to display the result on the displayer. The procedure to create a diaplay window is simliar to previous section. After creating a window, we can show the result on displayer according to the related code in Figure 5.7.


Control bar
NV Server
File Control Tmp Background Tmp Image Final Image
Captured image
Object mask
Stationary background buffer
Message window
Monitor

Frame No =179 Capture Frame Rate=4.00

Figure 5.8: The entire application program interface.

5.3 Control System

There are two major control unit: capture-control unit and threshold-adjustment unit. The capture-control unit controls every option needed for digital camera, such as start, stop, image size, and luminance. The threshold-adjustment unit is used to adjust the related threshold in temporary foreground mask, short-term background, and background subtraction.

The entire application program interface is shown in Figure 5.8.

5.4 Conclusion

In the section, we show the required processing time in this system. First, we consider the situation of zero camera motion. For quickly obtaining an apparent improvement, we stop some special modules which only useful in special situation, such as shrink initial object mask to edge map. Besides, we reduce the excuting frequence of some time-consuming modules. According the discussion in chapter 4, the module for noise estimation need not to work for every frame since the camera noise is usually keep the same. Here, we only estimate the noise of first frame and then the camera noise of following four frames is equal to that of the first frame. Besides, the module for displaying the final result is also

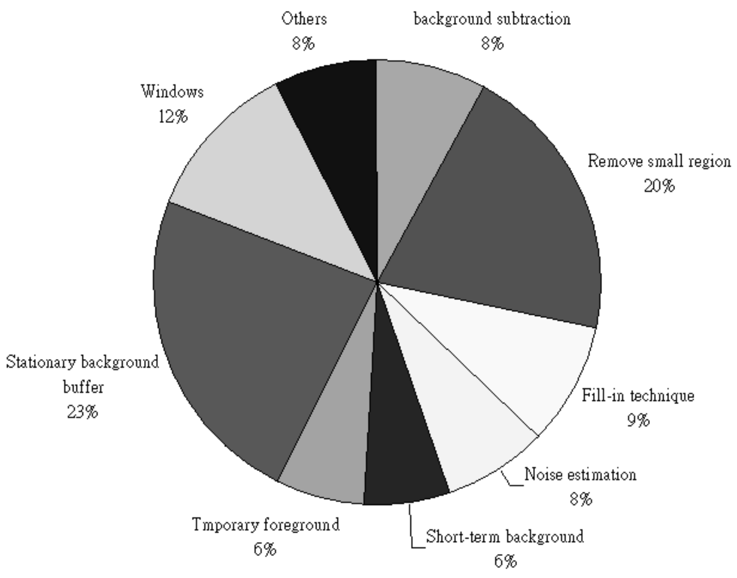


Figure 5.9: Relative computing time of every module when camera is fixed.

a time-consuming process, and therefore the stationary background buffer is displayed every ten frames and the object mask is displayed every two frames.

The image size in this system is CIF (352 × 288) and the PC is P-4 2.4-GHz with 512-MB RAM. The Relative computing time of every module is shown in Figure 5.9 and the current implementation yields a speed of about 5 frames per second. The higher time-consuming modules are stationary background buffer and remove small region. When we obtain the background, we may stop many modules to improve the efficiency, such as stationary background buffer, temporary foreground mask, fill-in technique, and short-term background. After we stop those modules, we can save about 44% of entire processing time. The module for remove small region is used to refine the final object mask and in the future it will be the major target for optimization.

Now, we discuss the efficiency when camera motion occurs. The relative computating time is shown in Figure 5.10 and the frame rate is about 1.7 frames per second. Since our system will be combined with MPEG-4 encoder, many identical modules can joint toghther, such global motion estimation which is about 21in our system.

The most time-consuming module is bilinear interpolation. The interpolation is used in gradient descent and warp the current background to panorama background buffer.

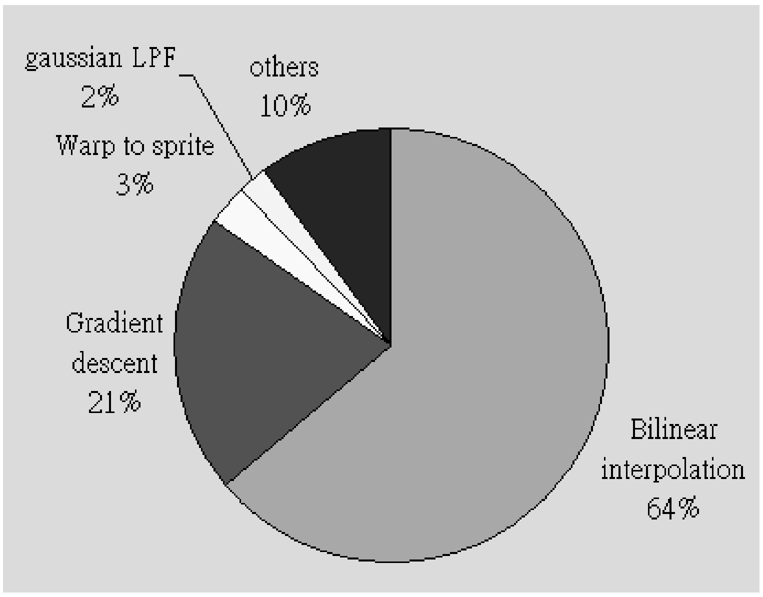


Figure 5.10: Relative computing time of every module when camera motion occurs.

Here, the nearest neighborhood algorithm which is less accurate but fast is under consideration. In our experiment, the accuracy of mosaic image is too low to accept when we use the nearest neighborhood algorithm. The required accuracy of interpolation in gradient descent is very high and therefore we should keep the use of bilinear interpolation in gradient descent. The required accuracy of interpolation for warping background to panorama buffer depend on the camera motion. When the image is enlarged, the block effect is very apparent. Here, we prefer to keep the overall accuracy and therefore the nearest neighbor algorithm is not adopted in our system.

Chapter 6

Simulation Results

6.1 Segmented Image Masks

In this section, we show some simulation result of Mother-and-Daughter, Claire, and Akiyo sequences. The Claire and Akiyo sequences are common cases in videoconferencing and Mother-and-Daughter is the case which include two major objects. The background subtraction needs some time to gather information of background and therefore the object masks of initial frames are less accurate. It can be seen that we can get more accurate image masks with the more information in the stationary background buffer.

Some results of Mother-and-Daughter sequence, Claire sequence, and Akiyo sequence are shown in Figures 6.1, 6.2, and 6.3, respectively. According to our observation the required time to obtain enough background for Mother-and-Daughter sequence, Claire sequence, and Akiyo sequence is about 260 frames, 150 frames, and 10 frames, respectively.

As we can see, the boundary of mask is very accurate when the related background is obtained and therefore the accuracy of our method is highly dependent on the amount of obtained background. There are two major factors to influence the required time for gathering enough background. First, if the background is always covered by moving objects, the related background is difficult to obtain and therefore the required time depends on the time which the covered background become uncovered. When these covered background become uncovered, the related boundary is usually incorrect, such as Figures 6.1(b). Sec-

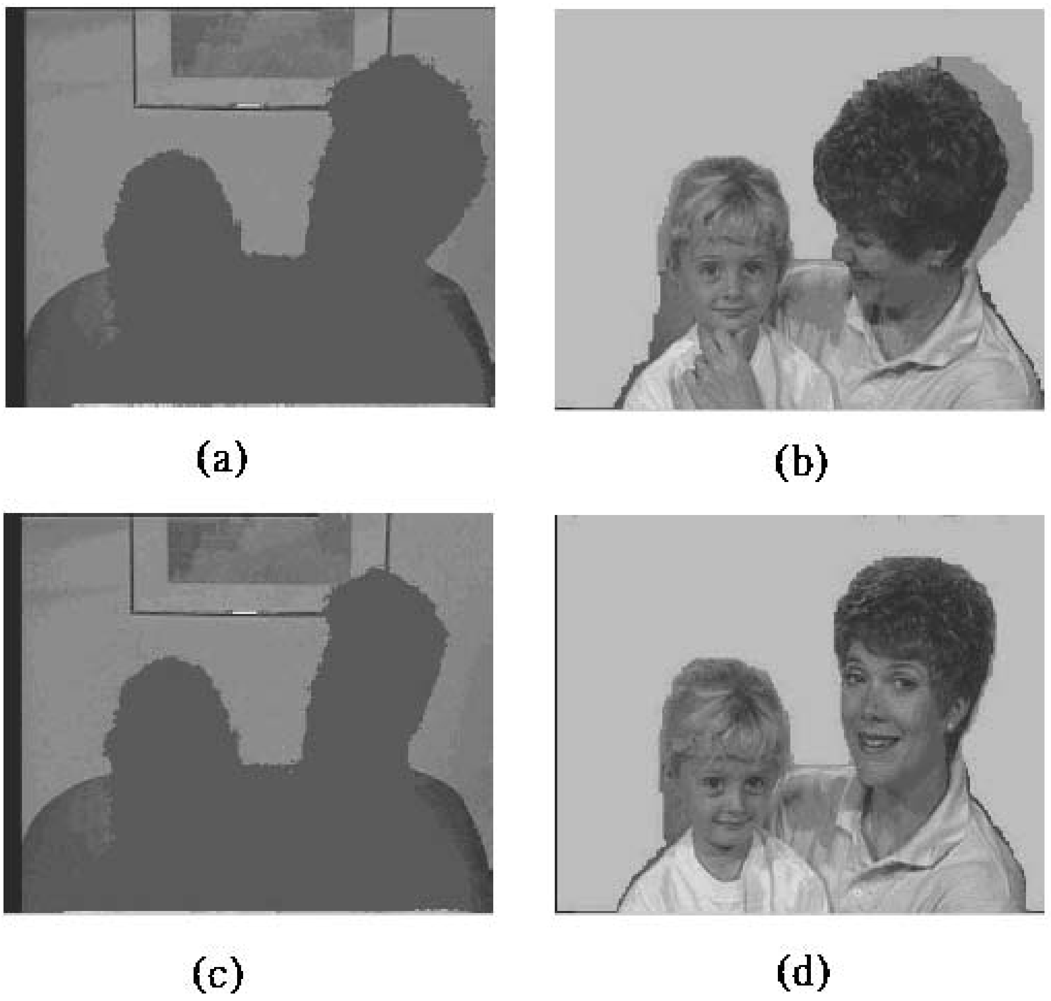


Figure 6.1: (a) Stationary background buffer in 140th frame. (b) Image mask in 140th frame. (c) Stationary background buffer in 260th frame. (c) Image mask in 260th frame.

ond, the required time to gather enough background also depends on the camera noise. In the case of low camera noise sequence, we can set more critical thresholds in short-term background and temporary foreground. The more critical thresholds can lead the shorter gathering time in stationary background buffer. The camera noises of the three sequences from high to low are Mother-and-Daughter, Claire, and Akiyo. According to the figures given above, the required time to obtained enough background from long to short is also in this order.

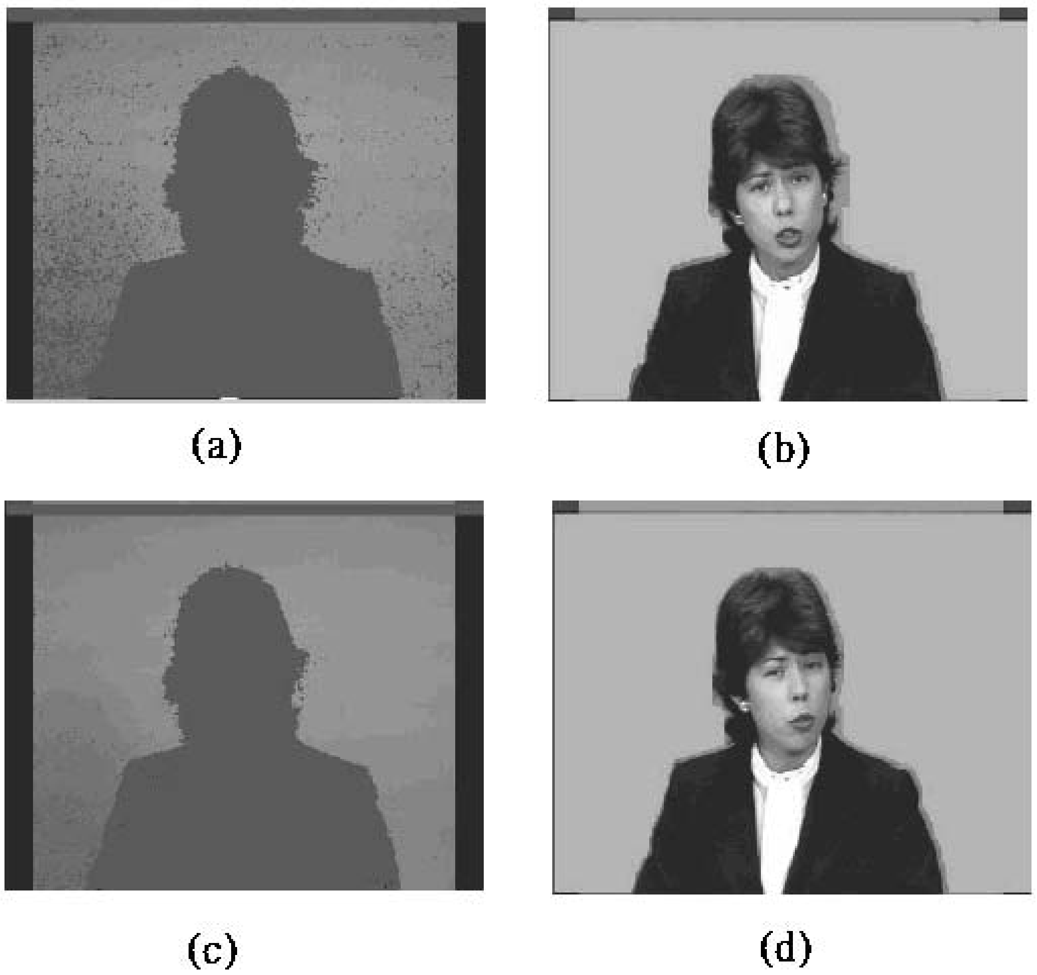
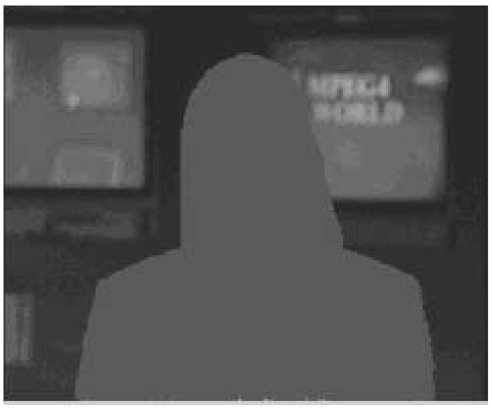
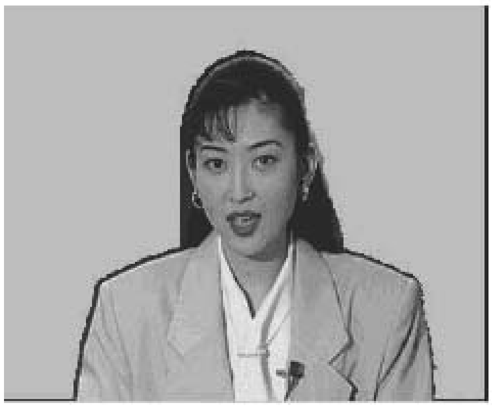


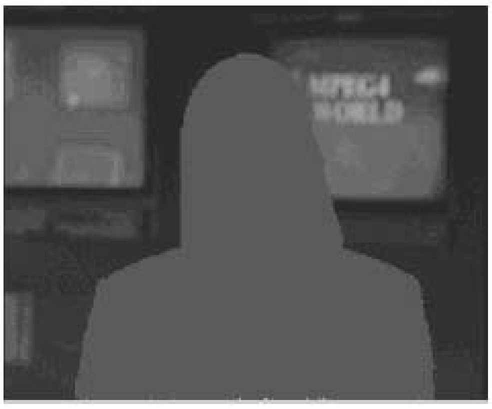
Figure 6.2: (a) Stationary background buffer in 60th frame. (b) Image mask in 60th frame. (c) Stationary background buffer in 150th frame. (c) Image mask in 150th frame.



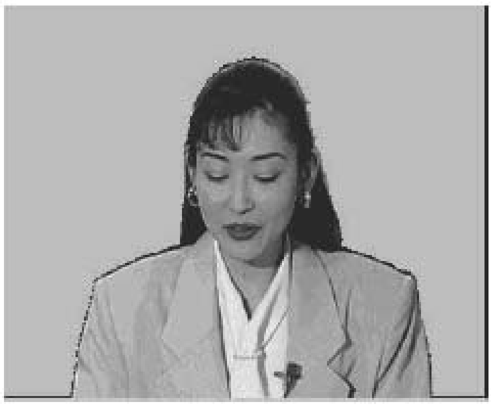
(a)



(b)



(c)



(d)

Figure 6.3: (a) Stationary background buffer in 10th frame. (b) Image mask in 10th frame. (c) Stationary background buffer in 165th frame. (c) Image mask in 165th frame.

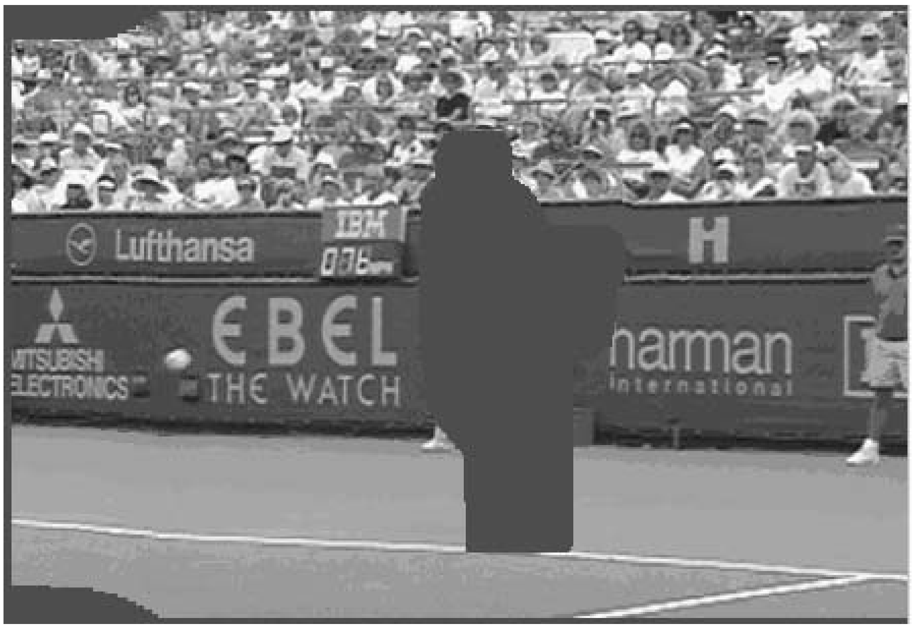


Figure 6.4: Mosaic result from initial to 13th frames.

6.2 Global Motion Estimation and Mosaic

In order to deal with camera motion, we use the global motion estimation to align the image with panorama background buffer and recover the stationary background buffer. In this section, we first use the Stefan sequence to show the result of panorama background buffer. The temporary mask here has obtained by others from http://cwww.ee.nctu.edu.tw. The result from initial frame to 13th frame is shown in Figure 6.4. The result from 40th frame to 73th frame is shown in Figure 6.5. The result from 130th frame to 161th frame is shown in Figure 6.6.

As shown in [9] and [6], the more accurate mosaic of Stefan sequence results require a perspective model. The advantage of perspective model is that it can handle the transform as shown in Figure 6.7. If the displacement of camera is small, those transform can be approximated by affine transform. A example is shown in Figure 6.8 The results given above meet small displacement requirement and therefore the affine model is enough to obtian an acceptable mosaic. In the case of larger displacements, such as 246th to 247th frames, the affine model fails as shown in Figure 6.9.

Now, we show the benefit of background recovery using a sequence captured in our

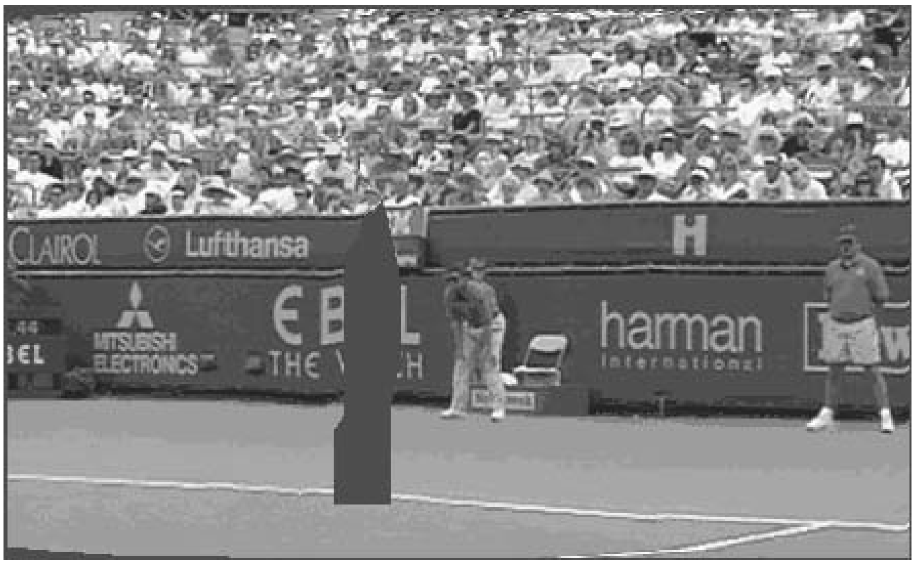


Figure 6.5: Mosaic result from 40th to 73th frames.

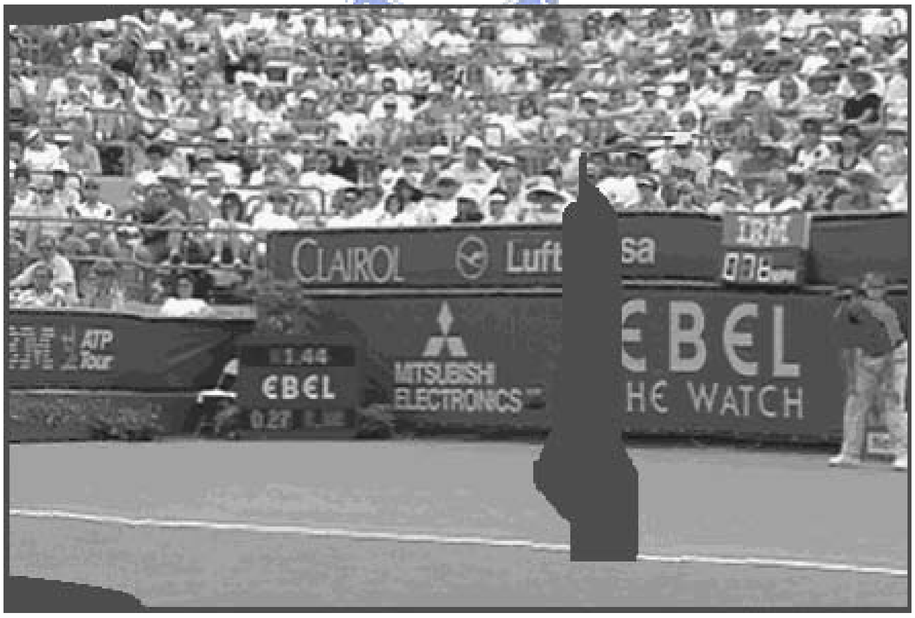


Figure 6.6: Mosaic result from 130th to 161th frames.

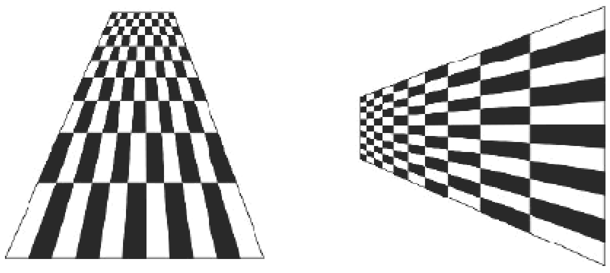


Figure 6.7: Perspective tramsform.

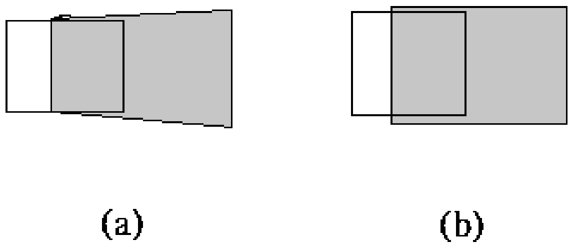


Figure 6.8: (a) Perspective tramsform. (b) Approximated by affine transform.

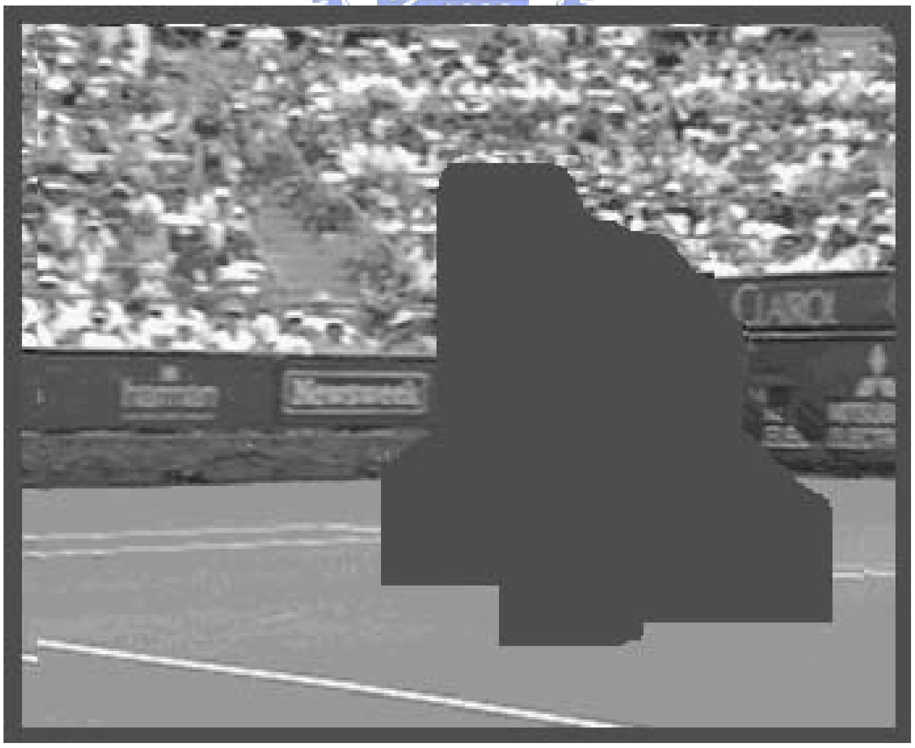


Figure 6.9: Mosaic result from 246th to 247th frames.

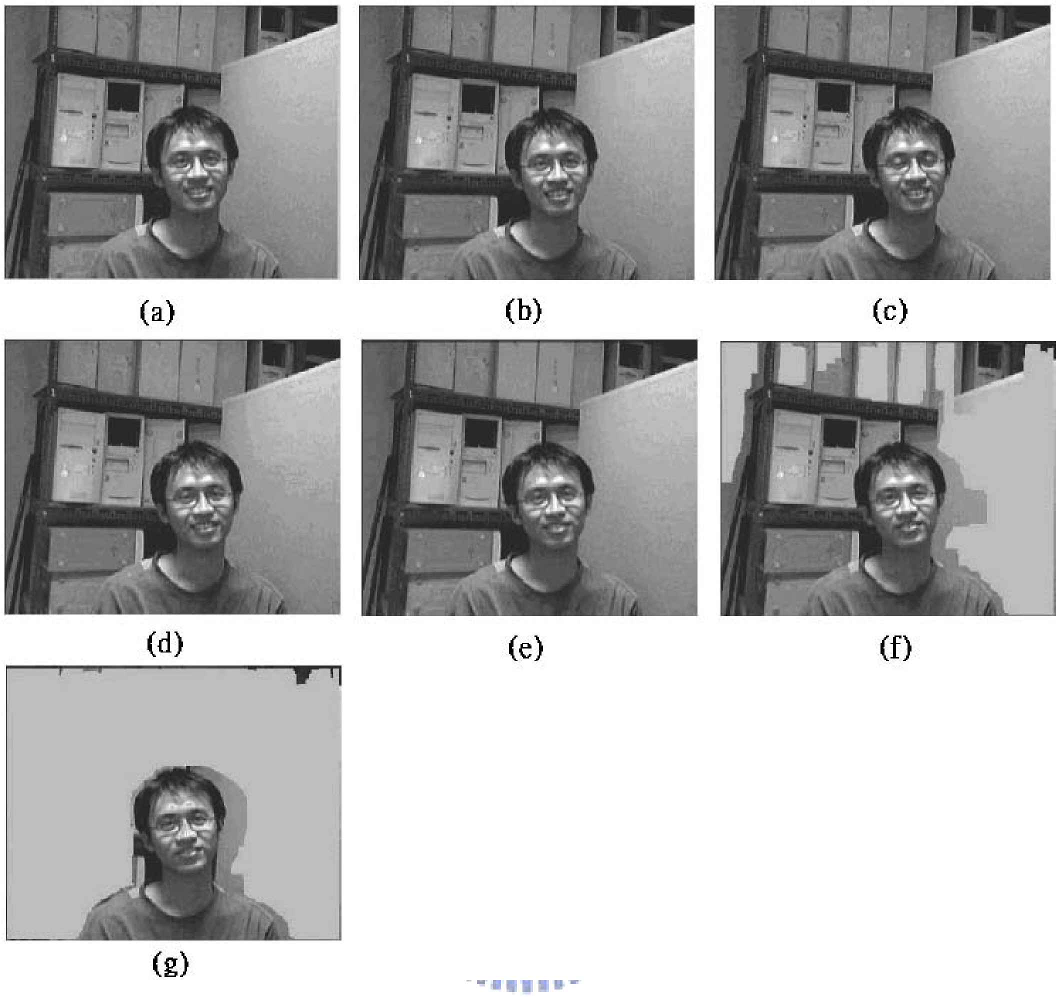


Figure 6.10: Image mask without background recovery. (a) The 145th frame which camera motion is detected. (b) 146th frame. (c) 147th frame. (d) 148th frame. (e) 149th frame. (f) 150th frame. (g) 151th frame.

lab.. If we remove the function of background recovery and just reset the background when camera motion occurs, the image masks at consecutive time step safter camera moving are shown in Figure 6.10. If we use background recovery instead of reseting background, the image masks at the same time steps are shown in Figure 6.11. It is obvious that the result with background recovery can get more accurate mask during rebuilding of new background.

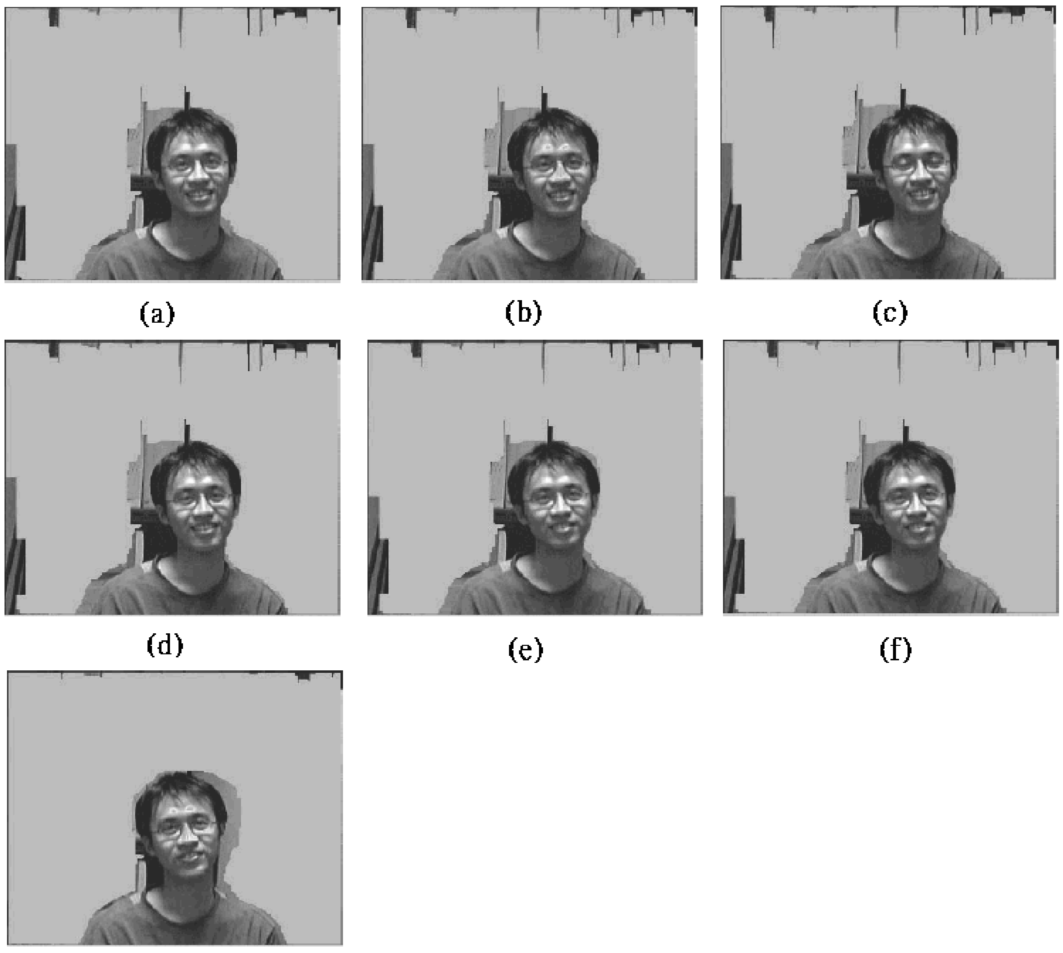


Figure 6.11: Image mask with background recovery. (a) The 145th frame which camera motion is detected. (b) 146th frame. (c) 147th frame. (d) 148th frame. (e) 149th frame. (f) 150th frame. (g) 151th frame.

Chapter 7

Conclusion and Future Work

We developed and implemented of an video segmentation system on personal computer. TThe intended application is PC-based multipoint videoconferencing system.

The core of our system is the background subtraction technique. We use a temporary foreground mask to reduce the influence of inner flat region in background construction and use a panorama background buffer to improve the accuracy of image mask during camera moving. For easier obtaining the relative thresholds of each module a two staged method for camera noise estimation is introduced to reduce the effect of moving objects and those thresholds are adjusted based on the estimated camera noise.

The relative simulations in previous chapters show the system can get an accurate image mask and deal with the camera moving. Besides, the two staged noise estimation also effectively reduces the effect of moving objects.

For quality improvement we can do some improvements for the main projects, in the future.

1. Adding the module to deal with shadow and light change.

   The position of shadow is controled by the position of light and therefore the shadow effect greatly depends on the position of light. If the great shadows appear in background, the shadows are also regarded as moving objects in the module of background subtraction. Hence, a module to recude the shadow effect can improve the accuracy of final image mask when the shadow is great.

2. Using a more robust motion model.

   In this system, a affine model is used and it still can not handle every camera moving. In the module of background recovery, the more accurate recovery can obtain a more accurate image mask when camera moving occurs and therefore a more robust model, such as perspective model may effectively improve the auuracy of background recovery by handling more camera moving.

3. Combine the segmentation system with MPEG-4 encoder.

   The image mask will be used as alpha plane in MPEG-4 encoder and therefore some modules, such as global motion estimation and initial matching which are also used in MPEG-4 encoder can be combined to improve the speed.

4. More optimization.

   The current processing time is not fast enough and a better optimization is need to improve the efficiency. The optimization may focus on interpolation and removing small regions.

Bibliography

[1] T. Meier and K. N. Ngan, "Video segmentation for content-based coding," *IEEE Trans. Circuits Syst. Video Technol.*,vol. 9, no. 8, pp. 525–538, Dec. 1999.

[2] T. Aach, A. Kaup ,and R. Mester, "Statistical model-based change detection in moving video," *Signal Processing*, vol. 31, pp. 165–180, Mar. 1993.

[3] S. Y. Ma, S. Y. Chien, and L. G. Chen, "Efficient moving object segmentation algorithm using background registration technique," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 7, pp. 577–586, July 2002.

[4] C. Kim and J. N. Hwant, "Fast and automatic video object segmentation and tracking for content-based application," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 2, pp. 122–129, Feb. 2002.

[5] MPEG-4 Video Group, "MPEG-4 video verification model version 16.0," *ISO/IEC JTC1/SC29/WG11, MPEG00/N3312*, Noordwijkerhout, Netherlands, Mar. 2000.

[6] F. Dufaux and J. Konrad, "Efficient, robust, and fast global motion estimation for video coding," *IEEE Trans. Image Porcessing*, vol. 9, pp.497–501, Mar. 2000.

[7] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing, 2nd ed.*. Cambridge University Press, Cambridge, England, 1992.

[8] A. Smolic and T. Wiegand, "High-resolution video mosaicing," in *Proc. ICIP2001, IEEE Int. Conf. Image Processing*, Oct. 2001, pp. 1–4.

[9] Y. Lu, W. Gao, and F. Wu, "Fast and robust sprite generation for MPEG-4 video coding," in *Proc. IEEE Pacific-Rim Conf. Multimedia*, Beijing, China, Oct. 2001, pp. 118–125.

[10] J. F. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 6, pp. 679–698, Nov. 1986.

[11] Y. H. Jan and D. W. Lin, "Video segmentation with extraction of overlaid objects via multi-tier spatio-temporal analysis," to appear in *J. Chinese Institute Elec. Eng.*,2004.

[12] "Canny operator code," http://ouray.cudenver.edu/ na0alber/DataCompressionPaper.htm.

[13] "Video for Windows," http://msdn.microsoft.com/library/psdk/multimed/avifile 8dgz.htm.

[14] "VidCap: Full-Featured Video Capture Application," http://msdn.microsoft.com/library/devprods/vs6/visualc/vcsample/vcsmpvidcap.htm.