# การปฏิเสธพื้นหน้าเพื่อกำจัดปรากฏการณ์ภาพเหลื่อมในการเชื่อมต่อลำดับวีดิทัศน์

นางสาวฐานิสสร พนารังสรรค์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554

# FOREGROUND REJECTION FOR PARALLAX REMOVAL IN VIDEO SEQUENCE STITCHING

Miss Thanissorn Panarungsun

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering Program in Electrical Engineering
Department of Electrical Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2011

| | |
|---|---|
| Thesis Title | FOREGROUND REJECTION FOR PARALLAX REMOVAL IN VIDEO SEQUENCE STITCHING |
| By | Miss Thanissorn Panarungsun |
| Field of Study | Electrical Engineering |
| Thesis Advisor | Assistant Professor Supatana Auethavekiat, Ph.D. |
| Thesis Co-advisor | Duangrat Gansawat, Ph.D. |

---

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

ฐานิสสร พนารังสรรค์ : การปฏิเสธพื้นหน้าเพื่อกำจัดปรากฏการณ์ภาพเหลื่อมในการเชื่อมต่อลำดับวีดิทัศน์. (FOREGROUND REJECTION FOR PARALLAX REMOVAL IN VIDEO SEQUENCE STITCHING) อ. ที่ปรึกษาวิทยานิพนธ์หลัก: ผศ. ดร.สุพัฒนา เอื้อทวีเกียรติ, อ. ที่ปรึกษาวิทยานิพนธ์ร่วม: ดร.ดวงรัตน์ แก่นสวัสดิ์, 86 หน้า.

การเกิดปรากฏการณ์ภาพเหลื่อมเป็นปัญหาสำคัญในกระบวนการเชื่อมต่อภาพเข้าด้วยกัน ปรากฏการณ์นี้ส่งผลให้เกิดความผิดพลาดในการปรับพิกัด และปรากฏการณ์ภาพหลอน ในงานวิจัยนี้ จึงนำเสนอวิธีแก้ปัญหาปรากฏการณ์ภาพเหลื่อมด้วยวิธีการปฏิเสธพื้นหน้า กระบวนการนี้เริ่มจาก การประมาณการเคลื่อนที่ของภาพระหว่าง 2 เฟรม โดยการจับคู่ลักษณะเด่นแบบ SIFT แล้วพิจารณาพิกเซลที่มีความแตกต่างระหว่างเฟรมที่มากที่สุดด้วยตัวดำเนินการตรรกะแบบ OR เพื่อสกัดพื้นหน้าออกมา พื้นหน้าที่สกัดได้แบ่งเป็น 2 กลุ่มคือ พื้นหน้าเฟรมใกล้ และพื้นหน้าเฟรมไกล ในการหาพื้นหน้าที่แท้จริงทำจากการโหวตเพื่อหาพิกเซลของพื้นหน้าเฟรมใกล้ที่ปรากฏบริเวณที่มีพื้นหน้าเฟรมไกล พื้นหน้าที่แท้จริงที่สกัดได้ ส่วนใหญ่เป็นขอบของวัตถุ อาศัยคุณสมบัติที่รูปร่างของวัตถุมีการเปลี่ยนแปลงน้อย ดังนั้นสามารถแปลงขอบวัตถุจากเฟรมอื่นมาเฟรมปัจจุบันตามโมเดลการเคลื่อนที่เส้นตรง 2 มิติที่หาได้จากขั้นตอนวิธี tracking ขอบในเฟรมอื่นถูกนำมาใช้ปรับปรุงขอบในเฟรมปัจจุบัน พื้นที่ของวัตถุพื้นหน้าถูกสร้างจากการระบายพื้นที่ 2 ครั้ง และปรับปรุงโดยใช้พื้นที่ของวัตถุพื้นหน้าในเฟรมอื่น ซึ่งถูกเคลื่อนมาในตำแหน่งเฟรมปัจจุบันด้วยโมเดลการเคลื่อนที่เส้นตรง 2 มิติ จากการทดลองกับสัญญาณวีดิทัศน์ 13 ชุด พบว่า การกำจัดปรากฏการณ์ภาพเหลื่อมด้วยวิธีที่นำเสนอ ส่งผลให้ (1) สามารถหาการเคลื่อนที่ได้ถูกต้องมากขึ้น (2) ลดปรากฏการณ์ภาพหลอน (3) ผลลัพธ์ของการเชื่อมต่อมีความคมชัดมากขึ้นและ (4) มีข้อมูลฉากหลังของผลลัพธ์เพิ่มมากขึ้น

| | |
|---|---|
| ภาควิชา วิศวกรรมศาสตร์ไฟฟ้า | ลายมือชื่อนิสิต |
| สาขาวิชา วิศวกรรมศาสตร์ไฟฟ้า | ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก |
| ปีการศึกษา 2554 | ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์ร่วม |

# # 547 01754 21 : MAJOR   ELECTRICAL ENGINEERING
KEYWORDS :  VIDEO SEQUENCE STITCHING / PARALLAX REMOVAL / MOTION ESTIMATION

THANISSORN PANARUNGSUN : FOREGROUND REJECTION FOR PARALLAX REMOVAL IN VIDEO SEQUENCE STITCHING.  ADVISOR : ASST. PROF. SUPATANA AUETHAVEKIAT, Ph. D., CO-ADVISOR : DUANGRAT GANSAWAT, Ph.D., 86 pp.

Parallax is a key challenge that leads to inaccurate registration and ghosting effect of objects in the result of panorama image stitching. A novel foreground rejection method is proposed in this thesis to remove parallax in video sequence stitching. Firstly, the global motion is estimated between two frames using SIFT feature matching. The foreground is obtained by applying the logical OR operators to the pixels that have high displaced frame difference. There are two groups of foreground: the near-frames and the far-frames groups. Then, voting scheme is applied in the way that only the near-frame foreground inside the area of far-frame foreground is considered as the actual foreground. The extracted foreground at this stage is mostly the edges of objects. Since the change of a foreground shape is very small, 2D translation motion from tracking algorithm is used to project foreground (edges) from other frames to the current frame. The edges from other frames are then used to refine the edge in the current frame. The 2-stage rendering is then applied to the extracted edge for the foreground area. The foreground area is refined by the foreground area of other frames mapped to the current frames by 2D translation model. The experimental results of 13 test video sequences indicated that the removal of parallax objects by the proposed foreground rejection method led to (1) more accurate registration, (2) the reduction of the ghosting effect, (3) sharper stitched result and (4) more background information in the results.

| | |
|---|---|
| Department : Electrical Engineering | Student's Signature |
| Field of Study : Electrical Engineering | Advisor's Signature |
| Academic Year : 2011 | Co-advisor's Signature |

# ACKNOWLEDGEMENTS

In this acknowledgement, I would like to offer my sincerest gratitude to my major advisor, Asst. Prof. Dr. Supatana Auethavekiat, and my co-advisor, Dr. Duangrat Gansawat, for her guidance, invaluable advice and extensive support in this thesis. I would like to sincerely thank to my thesis committees, Asst. Prof. Dr. Supavadee Aramvith and Asst. Prof. Dr. Sukanya Phongsuphap for their constructive and helpful suggestions in this thesis.

I would like to express my sincere thanks to Assoc. Prof. Somchai Jitapunkul, Miss Parichat Sermwuthisarn and Miss Saranta Ponla for their supports and advices.

Finally, I am deeply thankful to my family for their love, encouragement, all understanding and supports through the duration of my studies.

# CONTENTS

CHAPTER Page

# List of Tables

# List of Figures

Figure Page

# CHAPTER I

# INTRODUCTION

## 1.1 Motivation and Problem Statement

Because of the parallax effect, when objects are located at different depths, they appear at different location under different viewing direction. In this thesis, parallax objects are defined as the objects located at different depths from the major part of the image (background). The inclusion of parallax objects in image stitching result leads to the shape distortion and/or ghosting effect (the partial appearance of an object). Therefore the foreground rejection algorithm to remove a parallax object is proposed in this project. The removal of the parallax object leads to the more accurate image registration. The image stitching is performed only to the background; thus, the stitched results appear more natural than the stitching with the inclusion of parallax objects.

## 1.2 Literature Review

Video sequence stitching is the process of creating a panorama image from a video sequence. Although there are no moving objects in each frame, objects may appear to be moved if they lie in different depths (parallax effect). There are two major approaches to solve the parallax effect in video stitching. The first one is to search for the merging points such that the distortion is not clearly visible [1-3]. The second one is to search for the effective blending method [4]. Hybrid method [5] is also proposed.

Since parallax objects change the position relatively with the video frame, the first solution leads to the distortion from the true image. The second solution requires accurate image registration to convert all video frames to the reference coordinates. The position change of the parallax object leads to a high error (outlier). The outlier impedes the accurate registration, since the popular cost functions in an image registration are the absolute and the squared errors, which are not robust against the outliers [6]. The registration based on features (Harris corner [7], Scale-Invariant Feature Transform (SIFT) [8][9], etc.) together with RANSAC [10] may reduce the error. Nevertheless, as suggested by [11], more accurate registration is acquired if the intensity based registration is applied after the feature based registration. The parallax effect of objects in intensity based registration cannot be avoided.

Since parallax objects change the location along the video frames, they can be considered as the foreground objects. Various foreground rejection algorithms have been proposed in [12]. However, the requirement of the parallax object removal is less strict than the one of conventional foreground rejection. Simpler algorithms can be applied. The only additional requirement is that the algorithm must be able to detect the foreground in the presence of a global motion.

## 1.3 Objectives

1. To propose the foreground rejection algorithm for background stitching in a video sequence.
2. To develop the algorithm that is able to stitch the background in video sequence so that the stitched result appears natural.

## 1.4 Scopes

1. The proposed foreground rejection method is for gray-scale video sequences.
2. The stitching is done only to the background.
3. The motion model of background is the affine model.
4. There is only one parallax object in the video sequence.
5. The global motion can be approximated from the feature based motion estimation.
6. The compositing surface is the flat plane.
7. The algorithm is performed off-line.
8. The merging filter is the median filter.

## 1.5 Research Procedure

1. Literature review on the topics that are relevant to the research works of the dissertation**.**
   - 1.1 Image stitching.
   - 1.2 Image stitching with parallax objects.
   - 1.3 Foreground rejection.
2. Develop the foreground rejection algorithm for video sequence stitching.
   - 2.1 Detection of the foreground edge by logical operator and voting.
   - 2.2 Simple rendering for the foreground object.
   - 2.3 Tracking of the foreground objects to refine the results in 2.1 and 2.2.
3. Evaluate the proposed algorithm by visual inspection and compare the results between the stitching with and without foreground rejection.
4. Check whether the conclusions meet all the objectives of the research work of the dissertation.
5. Write the dissertation.

## 1.6 Contributions

1. Propose the foreground rejection algorithm for background stitching of the video sequence.
2. Integrate the tracking algorithm into foreground rejection algorithm to improve the rendering.
3. Develop the background stitching algorithm on flat surface.

## 1.7 Thesis Organization

- **Chapter 1.** Motivation and problem statement, literature review, objectives, scopes, research procedure and contributions are discussed.
- **Chapter 2.** This chapter provides the survey on the algorithm for video stitching and foreground rejection techniques.

- **Chapter 3.** The proposed method is presented in this chapter.
- **Chapter 4.** This chapter gives the preliminary results of the proposed method. It also compares the results between with and without foreground rejection in video sequence stitching.
- **Chapter 5.** This chapter concludes the thesis with the evaluation of the proposed method and the recommendation for future work.

# CHAPTER II

# BACKGROUND

Video stitching algorithms and foreground rejection methods are introduced in this chapter. The video stitching consists of two main parts: (1) registration as described in Section 2.1.1 and (2) merging as described in Section 2.1.2. In Section 2.1.1, 2D linear motion models and motion estimation techniques are described. In Section 2.1.2, composites surfaces and the overview of merging techniques are introduced. The foreground rejection methods are discussed in the next section: (1) pixel-based rejection in Section 2.2.1 and (2) block-based rejection in Section 2.2.2.

## 2.1 Video stitching

Figure 2.1 shows the block diagram of the video stitching algorithm. In a video sequence, each frame is taken at different viewing direction; therefore, their reference axes locate at different location. The first step of video stitching is to align (register) all frames to the same reference view. After that, information in all frames is merged (stitched) to form one large panoramic image. Currently, there are several applications of image stitching, such as digital maps, satellite photos and panorama images.

### 2.1.1 Registration

In computer vision, image alignment, image registration and motion estimation are related. The objective of the three algorithms is to find the motion (mapping) of a source image (frame) to a target image (frame). There are several mathematical motion models, both linear and non-linear. In this thesis, only 2D linear motion models are considered as described in Section 2.1.1.1. There are two major approaches for image registration: (1) direct or intensity based and (2) indirect or feature based methods. The direct and indirect methods are discussed in Sections 2.1.1.2 and 2.1.1.3, respectively.

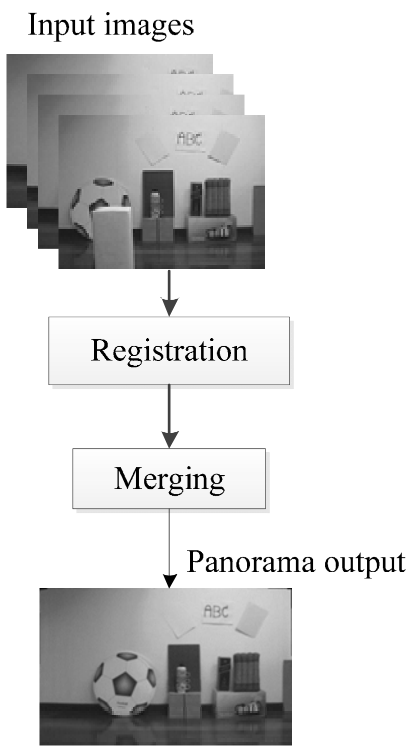


Figure 2.1: The block diagram of the video stitching algorithm.

#### 2.1.1.1 Motion model

The motion is described by the parameters in the motion models. There are many type of parametric motion models, such as 2D translation, planar perspective, 3D camera rotations, the mapping to non-planar (e.g., cylindrical or spherical) surfaces, etc. In this thesis, only the 2D linear models are considered. Three commonly used 2D linear models are 1) translation, 2) rigid body and 3) affine models.

**1) Translation model**

Translation model is used to represent the linear displacement in $x$ and $y$ directions. It can be used to approximate general motion as long as the motion is small. The translation model to transform the pixel at ($x$, $y$) to the one at $(x_{new}, y_{new})$ is defined as follows.

$$\begin{bmatrix} x_{new} \\ y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{2.1}$$

where $T_x$ and $T_y$ are the displacement in *x* and *y* direction, respectively.

**2) Rigid body model**

The rotation and scaling are added into the translation model to form the rigid body model. The rigid body model describes more general motion than translation model; however, it does not allow the change in object shape. It is defined as follows.

$$\begin{bmatrix} x_{new} \\ y_{new} \\ 1 \end{bmatrix} = S \begin{bmatrix} \cos\theta & \sin\theta & T_x \\ -\sin\theta & \cos\theta & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{2.2}$$

where *S* and $\theta$ are the scaling parameter and the rotating angle, respectively.

**3) Affine model**

Shearing is added to the rigid body model to form the affine model. The affine model allows the change in the shape of an object as long as the parallelism is preserved. It approximates the perspective model (pin holed camera), when the camera is located at infinity. It is defined as follows.

$$\begin{bmatrix} x_{new} \\ y_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{2.3}$$

where *a*, *b*, *c*, *d*, *e* and *f* are the affine parameters.

The affine model is selected as the transformation model in this thesis, because it can describe the widest range of the motion types among 2D linear motion models.

#### 2.1.1.2 Direct method

In the direct method, the optimum motion would provide the mapping with the least intensity difference between the source and the target images. The intensity may be described in spatial domain or transformed domain. The estimation accuracy is high, because it can use the (intensity) information of every pixel; however, its capture length is small, i.e. the estimation is accurate when the motion between frames is small. Furthermore, the intensity difference is often measured by either the absolute or the squared difference; therefore, its tolerance against outlier is low.

Three popular direct methods are presented in this section. They are (1) optical flow method, (2) block based method and (3) Bayesian method.

**1) Optical flow method**

Optical flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between images. Optical flow estimation methods use the Optical Flow Equation (OFE) for finding the motion parameters. The OFE provides the estimation of the motion by searching for the change in intensity between frames. This technique finds the motion parameters from the derivatives of the image intensity on space and time by considering the total temporal derivatives. The main idea is that the intensity of the same object is the same in every frame.

$$\frac{df_c\left(x,y,t\right)}{dt}=0, \tag{2.4}$$

where $f_c(x,y,t)$ is the continuous space-time intensity distribution;

$x$ and $y$ are the coordinate of the object and are varied according to the motion of the object;

$t$ is the time variable.

The optical flow equation or the optical flow constraint is obtained by applying the chain rule of differentiation to (2.4).

$$\left\langle \nabla f_c\left(x,y,t\right),(T_x,T_y)\right\rangle + \frac{\partial f_c\left(x,y,t\right)}{\partial t} = 0, \tag{2.5}$$

where $\nabla f_c(x,y,t) = \left[\frac{\partial f_c(x,y,t)}{\partial x} \quad \frac{\partial f_c(x,y,t)}{\partial y}\right]^T$ ;

$T_x$ and $T_y$ are the translated distance (motion) in $x$ and $y$ axes, respectively;

$\langle .,. \rangle$ denotes the vector inner product.

Since the motion in OFE is described by two variables ($T_x$ and $T_y$), (2.5) alone is not enough to find the motion. Additional equation is added in term of the additional assumption regarding the motion. Examples of the additional assumption are as follows.

i. **Second-order differential method.** The second derivative of the spatial image gradient is zero.

ii. **Block motion model.** Only one motion exists in one block; thus, if there are *n* pixels in a block, there are *n* equations for $T_x$ and $T_y$. The accuracy depends on the number of objects and the amount of features (edges, corners) in a block. If the block consists of more than one object or has only a few features, the motion is inaccurate.

iii. **Horn and Schunck method.** There is no abrupt motion change among pixels. The assumption of Horn and Schunck method can be described as follows.

$$\min\left(\left|\nabla T_x(x,y)\right|^2 + \left|\nabla T_y(x,y)\right|^2\right), \tag{2.6}$$

where $|T|$ is the magnitude of *T*. The motion near the boundary of an object is smoothed and becomes less accurate.

The accuracy of OFE depends on the accuracy of the additional assumption. OFE is often solved by gradient descend method; so the result may be the local minima. Generally, the calculation of OFE requires a derivative operation, which leads to a noise enhancement.

### 2) Block-based method

Block-based motion estimation is among one of the most popular motion estimation method. The main idea of the block based method is that there is only one motion within a block. The block based method is applied in many techniques such as MPEG-1, MPEG-2, etc. There are several techniques in this method such as the phase-correlation method, the block-matching method, etc. In this thesis, only the most popular block-matching method is discussed.

In a block-matching method, a pixel-domain (intensity) search procedure is applied to estimate the motion. The source image is divided into blocks of $M \times N$ pixels. The method tries to find the block in the target image corresponding to the block in the source image. Figure 2.2 shows the example of mapping the block in the source frame (Frame# $k$) to the area in the target frame (Frame# $k'$). If the search area is not limited, the estimation must match the block in Frame# $k$ to every part in Frame# $k'$; consequently, the computing cost is high. To reduce the computing cost, the search window is used to limit the area in Frame# $k'$ (dashed box in Figure 2.2). The mapping is then done between the block in Frame# $k$ to only the area within the search window. In this method, there are two major considerations: 1) matching criteria and 2) search procedures.

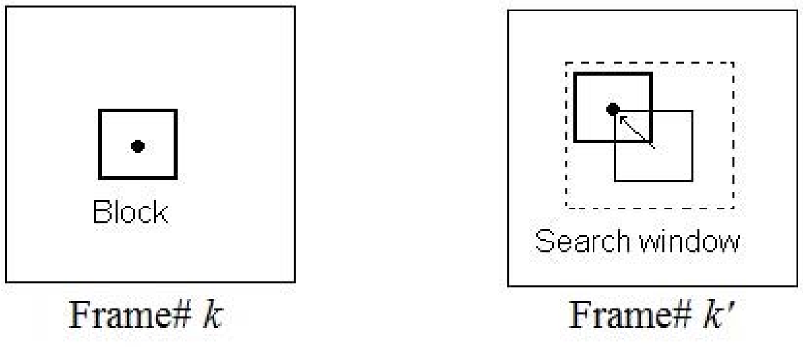


Figure 2.2: The basic idea of the block matching method.

***Matching criteria***

Matching criteria are used to measure the mapping quality. Most of them measure the normalized correlation. Examples of the matching criteria are the minimum Mean Square Error (MSE), the minimum Mean Absolute Difference (MAD), maximum maximum cross-correlation, Matching Pel Count (MPC), etc.

- **Minimum MSE**

MSE is defined as

$$MSE\left(T\right)=\frac{1}{N_1N_2}\sum_{(x,y)\in B}\left[f\left(x,y,k\right)-f\left(T(x,y),k'\right)\right]^2, \tag{2.7}$$

where $B$ is the given $N_1 \times N_2$ block;

$f\left(x,y,k\right)$ is the image intensity at (*x*, *y*) in the $k-$th frame;

$T(x,y)$ is the corresponding coordinate in the $k'-$th frame for (*x*, *y*) in the $k-$th frame. *T* is the function of the motion model.

The transformation acquired from minimum *MSE* is as follows.

$$T = \arg\min_T MSE(T) \tag{2.8}$$

- **Minimum MAD**

*MAD* is defined as

$$MAD\left(T\right)=\frac{1}{N_1N_2}\sum_{(x,y)\in B}\left|f\left(x,y,k\right)-f\left(T(x,y),k'\right)\right| \tag{2.9}$$

The transformation acquired from the minimum *MAD* is as follows.

$$T = \arg\min_T MAD(T) \tag{2.10}$$

- **Maximum phase-correlation**

Phase-correlation method makes use of the phase shifting property of Fourier transform. The phase of the signal in Fourier domain is shifted when there is the displacement in spatial domain. Given the relationship between the intensity in the $k-$th and the $k'-$th frame as follows.

$$f(x,y,k) = f(x+T_x, y+T_y, k') \tag{2.11}$$

The relation between the $k-$th and the $k'-$th frame in Fourier domain is

$$F(f_1, f_2, k) = F(f_1, f_2, k')\exp\left\{ j2\pi(f_1T_x + f_2T_y)\right\} \tag{2.12}$$

where $F(f_1, f_2, k)$ is the signal in the Fourier domain at the frequency $(f_1, f_2)$ in $k-$th frame.

The normalized correlation of the transformed signal in the $k-$th and the $k'-$th frame is as follows.

$$\tilde{C}(f_1, f_2) = \frac{F(f_1, f_2, k)}{F(f_1, f_2, k')}, \tag{2.13}$$

$$= \exp\left\{ j2\pi(f_1T_x + f_2T_y)\right\}, \tag{2.14}$$

where $\tilde{C}$ is the normalized cross correlation.

In spatial domain,

$$C(x,y) = \delta(x-T_x, y-T_y), \tag{2.15}$$

where $C(x,y)$ is normalized cross correlation in spatial domain;
$\delta$ is the delta-dirac function.

$T_x$ and $T_y$ can be calculated as follows.

$$(T_x, T_y) = \arg\max_{x,y} C(x,y) \tag{2.16}$$

- **Maximum MPC**

Each pel (pixel) within the block $B$ is classified as either a matching pel or a mismatching pel according to

$$MP(x,y;T)=\begin{cases}1 & ; \quad if\left|f(x,y,k)-f(T(x,y),k')\right|\leq\tau \\ 0 & ; \quad otherwise\end{cases} \tag{2.17}$$

where $MP$ is the matching pel function which is 1 and 0 for matching and mismatching pel, respectively;

$\tau$ is a predetermined threshold.

Then the number of matching pels ($MPC$) within $B$ is given by

$$MPC(T)=\sum_{(x,y)\in B} MP(x,y;T) \tag{2.18}$$

and

$$T=\arg\max_{T} MPC(T) \tag{2.19}$$

The MPC criterion requires a threshold comparator. In case of the 2D translation model, it requires a $\log_2(N_1\times N_2)$ counter.

***Search procedures***

There are several optimization methods to search for parameters that give the best-matching block in motion estimation. Examples of popular block-based search algorithms such as full search, three-step search, and cross search are described as follows.

- **Full search**

The full search algorithm evaluates the matching by applying every possible motion parameter to $T$. For example, in 2D translation model for the image of $\mu \times \nu$ pixels, the possible $T_x$ and $T_y$ are $[-\mu - 1, \mu - 1]$ and $[-\nu - 1, \nu - 1]$, respectively. This algorithm is extremely time-consuming.

The "search window" can be applied to reduce the computing time. For the searching windows of $(2M_1+1) \times (2M_2+1)$ centered at the block's center, the possible $T_x$ and $T_y$ are $[-M_1, M_1]$ and $[-M_2, M_2]$, respectively. Though the computing cost is reduced, it is still high for it requires $(2M_1+1) \times (2M_2+1)$ mappings for one block.

In practice, search procedure with lower computing cost is required. Heuristic rule is often incorporated into the search procedure. Though the accuracy is reduced, it is still good enough for general applications. The popular heuristic search are three-step search and cross search.

- **Three-step search**

Three-step search is widely used because it is both simple and robust. It also gives a near optimal performance. It searches for the best motion parameters from a coarse to fine scale. Given that the block in the source frame, $B_S$, is centered at $(x, y)$. The algorithm of the three-step search to find 2D translation parameters are as follow.

1) Initialize the step sizes of the motion parameters ($\Delta T$), $Tx_{best}$ and $Ty_{best}$ to 4, 0 and 0, respectively.
2) Compute the matching criterion between $B_S$ and the block centered at $(x + T_x, y + T_y)$ in the target image; $T_x \in \{Tx_{best} - \Delta T, Tx_{best}, Tx_{best} + \Delta T\}$, $T_y \in \{Ty_{best} - \Delta T, Ty_{best}, Ty_{best} + \Delta T\}$. $T_x$ are $T_y$ are the motions that $B_S$ moves in $x$ and $y$ axes, respectively.
3) Assign $Tx_{best}$ and $Ty_{best}$ to $T_x$ and $T_y$ giving minimum error in 2). Reduce $\Delta T$ by half.
4) Resume Step 2)-3) until $\Delta T$ is less than 1.

5) The motions of $B_S$ from the source frame to the target frame in *x* and *y* axes are $Tx_{best}$ and $Ty_{best}$, respectively.

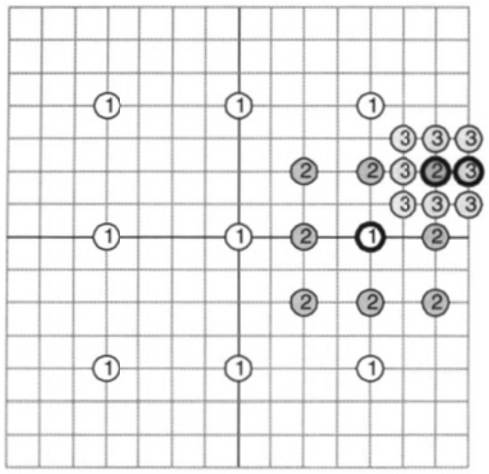


Figure 2.3: Illustration of three-step search [13].

Figure 2.3 illustrates the three-step search that can be described as follows.

In the first step, the matching criterion is calculated between $B_S$ centered at the central ① in the source frame and nine blocks centered at ① in the target frame. The nine blocks corresponds to the mapping of $B_S$ by ($T_x$, $T_y$) in Step (2) of the three-step search. Choose ① giving the best matching as the best motion parameter (shown as the bold **①** at the right edge) and the search continues with this motion as the search center. $\Delta T$ is then halved to 2.

In the second step, the matching criterion is calculated between the given block and the block whose center locates at ②. Choose ② giving the best matching as the new search center (shown as the bold **②** at the top right corner). The distance of the search is halved to 1.

In the third step, the matching criterion is calculated between the given block and the block whose center locates at ③. Choose ③ giving the best matching as the best motion (shown as the bold **③** at the right edge).

The number of iterations (steps) in the three-step search can be increased for better resolution. For example, if 0.25 pixel resolution is desired, the number of iterations can be increased to 5 (4, 2, 1, 0.5 and 0.25). Furthermore, the search is easily adapted to other motion models. The coordinate of the block in the target image is the transformed coordinate of the pixels in $B_S$.

- **Cross search**

A cross search is another logarithmic search strategy. In case of the 2D translation model, it differs from the three-step search as it searches only 5 positions and the motion parameter will be halved only when the best position is at the center or at the edge of the image. Given that the block in the source frame, $B_S$, is centered at $(x, y)$. The algorithm of the cross search for 2D translation parameters are as follow.

1) Initialize the step sizes of the motion parameters ($\Delta T$), $Tx_{best}$ and $Ty_{best}$ to 2, 0 and 0, respectively.

2) Compute the matching criterion between $B_S$ and the block centered at $(x + T_x, y + T_y)$ in the target image; $(T_x, T_y) \in \{(Tx_{best}, Ty_{best}), (Tx_{best} - \Delta T, Ty_{best}), (Tx_{best} + \Delta T, Ty_{best}), (Tx_{best}, Ty_{best} - \Delta T), (Tx_{best}, Ty_{best} + \Delta T)\}$.

3) Assign $Tx_{best}$ and $Ty_{best}$ to $T_x$ and $T_y$ giving minimum error in 2). If $Tx_{best}$ and $Ty_{best}$ are both 0 or $x+Tx_{best} \pm\Delta T$ or $y+Ty_{best}\pm\Delta T$ is beyond the image boundary, reduce $\Delta T$ by half.

4) Resume Step 2)-3) until $\Delta T$ is less than 1.

In the above algorithm, the motion in Step 2) is set as the end of the (+)-shape cross; however, it can be changed to the end of the (x)-shape cross where $(T_x, T_y) \in \{(Tx_{best}, Ty_{best}), (Tx_{best} - \Delta T, Ty_{best}-\Delta T), (Tx_{best}+\Delta T, Ty_{best}-\Delta T), (Tx_{best}-\Delta T, Ty_{best}+\Delta T), (Tx_{best}+\Delta T, Ty_{best}+\Delta T)\}$.

The search with the (+)-shape cross is depicted in Figure 2.4 and can be described as follows.

In the first step, the matching criterion is calculated between $B_S$ centered at the central ① in the source frame and five blocks centered at ① in the target frame (at the four corners and at the center). Choose ① giving the best matching as the best motion parameter (shown as the bold ➀ at the right edge) and the search continues with this motion as the search center. Since the best ① is not at the center, Δ*T* is keep at 2.

In the second step, the matching criterion is calculated between the given block and the block whose center locates at ②. Note that the motion that has already been checked can be omitted. Choose ② giving the best matching as the new center (shown as the bold ➁ on the right). Since the best ② is not at the center, Δ*T* is keep at 2.

In the third step, the search continues in the same manner as in the second step. Because the best motion (shown as the bold ➂ on the bottom) is not at the center, Δ*T* for the fourth step is keep at 2.

In the fourth step, the best result (shown as the bold ④ at right) has the search area beyond the image boundary if Δ*T* is still kept at 2; thus, Δ*T* is halved to 1.

In the fifth step, the search continues in the same manner as the previous step but with Δ*T* = 1 (the center of the block is shown as ⑤). The search continues until the best motion is at the center or reaches the edge of the image.

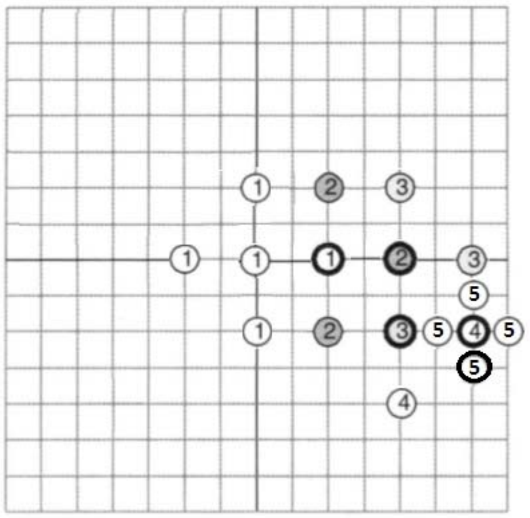


Figure 2.4: Illustration of the cross-search.

### 3) Bayesian method

In Bayesian method, the motion estimation is considered as a maximum *a priori* probability (MAP) estimation problem. The motion between the $k-\text{th}$ and $k'-\text{th}$ frame is

$$T = \arg\max_{T}(p(T \mid F_k, F_{k'})) \tag{2.20}$$

where $p(T \mid F_k, F_{k'})$ is a posterior pdf;

$F_k$ is the image matrix of the $k-\text{th}$ frame;

$F_{k'}$ is the image matrix of the $k'-\text{th}$ frame;

$T$ is the transformation operator to transform the coordinate between $k-\text{th}$ and $k'-\text{th}$ frame.

Bayes theorem is applied to Equation (2.20).

$$\begin{aligned} p(T \mid F_k, F_{k'}) &= \frac{p(T, F_k, F_{k'})}{p(F_k, F_{k'})} \\ &= \frac{p(F_k \mid T, F_{k'})\, p(T, F_{k'})}{p(F_k \mid F_{k'})\, p(F_{k'})} \\ &= \frac{p(F_k \mid T, F_{k'})\, p(T \mid F_{k'})}{p(F_k \mid F_{k'})} \end{aligned} \tag{2.21}$$

Because $F_k$ and $F_{k'}$ are given, $p(F_k \mid F_{k'})$ can be considered as a constant. The maximum of $p(T \mid F_k, F_{k'})$ is equivalent to the maximum of $p(F_k \mid T, F_{k'})\, p(T \mid F_{k'})$.

$$T = \arg\max_{T}(p(F_k \mid T, F_{k'})\, p(T \mid F_{k'})) \tag{2.22}$$

The first term in the right-hand side of (2.22) is the likelihood model. It measures the similarity between the $k-$th and $k'-$th frame under the transformation *T*. Gaussian distribution with zero mean is often assumed for the likelihood model. $p(F_k \mid T, F_{k'})$ is defined as follows:

$$p(F_k \mid T, F_{k'}) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{ \frac{-\sum_{(x,y)} [f(x,y,k) - f_{k'}(T(x,y,k'))]^2}{2\sigma^2} \right\} \tag{2.23}$$

where $\sigma^2$ is the distribution variance;

$f(x,y,k)$ is the intensity at $(x, y)$ in the $k-$th frame;

$f(x,y,k')$ is the intensity at $(x, y)$ in the $k'-$th frame.

The second term in the right-hand side of (2.23), $p(T \mid F_{k'})$ is *a priori* model. Since prior information such as smoothness constraint can be included in this term, Bayesian method is one of the most flexible and the most accurate motion estimation; however, it has the high computing cost, because MAP estimation is often solved by global optimization, such as simulated annealing, mean field annealing, highest confidence first, etc. Furthermore, it requires the training phase in order to estimate the parameters for the likelihood model and *a priori* model.

#### 2.1.1.3 Indirect method

In the indirect method, features such as edges, corners, etc., that are extracted from the images are used for matching. An estimation technique is applied to find the motion parameters such that the position of the feature in one image agrees with the position of the corresponding feature in the other image.

The accuracy of the indirect method depends on the quality of the features and the accuracy of the feature matching. Its capture length is larger than the indirect method; however, the accuracy is less, because the accuracy is limited by the accuracy of the feature extraction. The problem of inaccurate feature can be reduced by having the features distribute all over the image. Furthermore, the features should not changed position when the image is viewed at different direction (robust against transform) or corrupted by noise [11]. There are various feature extraction techniques, such as Harris [7], Scale Invariant Feature Transform (SIFT) [8][9], etc. SIFT is implemented in this thesis, so its algorithm is described in more detail.

- **Scale Invariant Feature Transform (SIFT)** [8][9]

SIFT is a method for extracting distinctive invariant features from images. Its underlying assumption is that good features are distinct across the scale space. The SIFT feature point, defined as a keypoint, is robust to affine transformation and noise; thus, it provides the reliable matching between images.

The scale space in SIFT is divided into octave. Images in the same octave have the same size, but with different level of blurring. The image is blurred by the convolution with the Gaussian function according to the following function.

$$L(x,y,\sigma) = G(x,y,\sigma) * f(x,y) \tag{2.24}$$

where $L(x,y,\sigma)$ is the pixel at (*x*,*y*) in the scale space at the blurring level ($\sigma$).

$G(x,y,\sigma)$ is a variable-scale Gaussian filter and is defined as follows.

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{2.25}$$

The blurring level is thus determined by the Gaussian variance, $\sigma$. At the next higher octave, the size of the image is halved by subsampling. The SIFT scale space is shown at the left column of Figure 2.5.

SIFT keypoints and their descriptions are extracted according to the following four stages.

### i. Scale-space extrema detection

The extrema is defined as the local maximum of the Different of Gaussian (DoG) in scale space. DoG is the difference between the two nearby scale space as shown in the right column of Figure 2.5. The pixel is the extrema if it has the highest DoG among its neighborhood. Figure 2.6 shows the example of neighborhood where pixels in the red square are the neighbor of the red pixel.

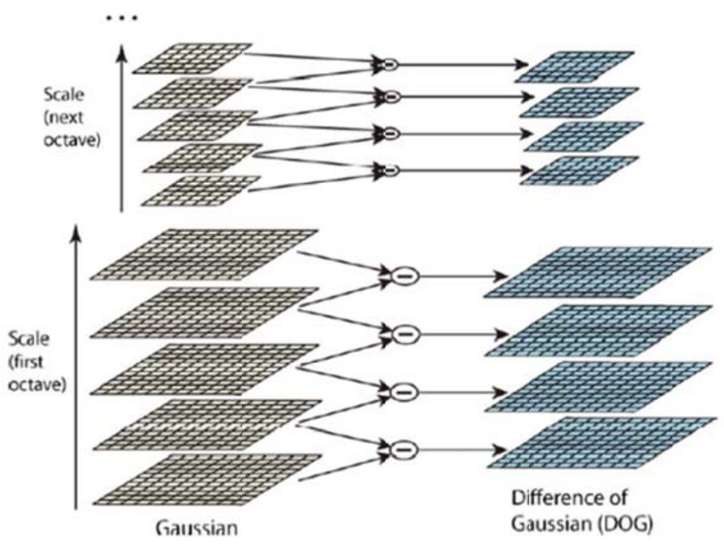


Figure 2.5: Illustration of Difference of Gaussian (DoG) of scale space in each octave [8].

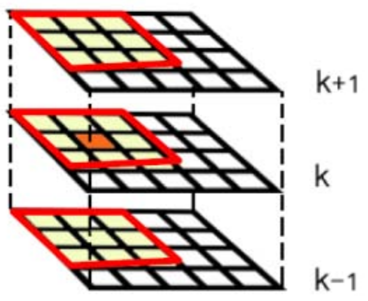


Figure 2.6: Illustration of the neighborhood of the red pixel. The neighbor pixels are located inside the red square [9].

**ii. Keypoint localization**

The extrema satisfying the following criterion is defined as a keypoint.

$$\frac{(D_{xx}+D_{yy})^2}{D_{xx}D_{yy}-(D_{xy})^2} < \frac{(r+1)^2}{r} \tag{2.26}$$

where $D_{mn}$ is the second derivatives of DoG in *m* and *n* direction; $D_{mn}$ is calculated at the extrema found in Step i.;

*r* is a constant which is defined by user. According to the experiment in [8], the optimal *r* is 10.

**iii. Orientation assignment**

The orientation of a keypoint is approximated from the orientation histogram which is constructed from the magnitude and the angle of the intensity gradient as follows. First, the magnitude and the angle of the pixel of *L* are estimated from the following equations.

$$m(x,y) = \sqrt{(L_x)^2 + (L_y)^2} \tag{2.27}$$

$$\theta(x,y) = \arctan(\frac{L_y}{L_x}) \tag{2.28}$$

where $L_x$ and $L_y$ are the vertical and the horizontal gradient of *L*, respectively.

The histogram has 36 bins, where each bin corresponds to the orientation span of 10°. Its input is pixels in the neighborhood of a keypoint. The neighborhood in this step is located only in the same blurring level. The magnitude of each pixel to put into the histogram is the weighted *m*. The weight is calculated according to the position of the pixel to the keypoint and defined as the circular Gaussian function with the mean at the position of the keypoint and the standard deviation of $1.5\sigma$ (the blurring level of

*L*). The orientation of the keypoint is defined as the angle of the bin having the magnitude within 80% of the highest magnitude. Because the keypoint is defined not only at the bin with the highest magnitude; therefore, some keypoints may have more than one orientation.

**iv. Construction of the keypoint descriptor**

The keypoint descriptor is used to match keypoints between images. It provides the information of the area around the keypoint in the form of the set of 16 orientation histograms. With the help of Figure 2.7, the construction of the descriptor is as follows.

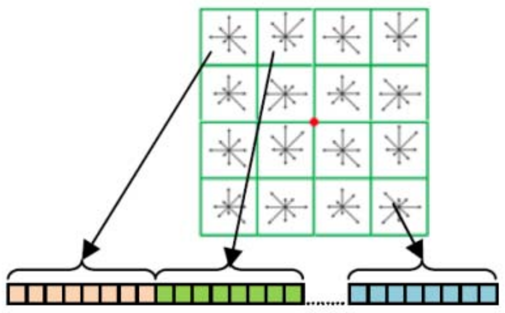


Figure 2.7: The construction of the keypoint descriptor. The keypoint is shown as a red point. [9]

i. Let the descriptor window be the 16×16 block with the center at the keypoint (the red point in the figure). Find the orientation and the magnitude of the pixels by (2.27) and (2.28). Then, the orientation of each pixel is rotated by the orientation of the keypoint.

ii. Divide the descriptor window to the 16 blocks of 4×4 pixels (shown as the green block in the figure). The 8-bin histogram is formed in the same manner as Step iii with the standard deviation of the Gaussian weight of 24.

iii. Order the histogram of each area into 128-D feature vector (see the figure) and normalize the feature vector to reduce the effect of illumination change.

The matching between SIFT keypoints is performed on the descriptors. The matching accuracy can be improved by statistic method such as RANSAC [10]. In this thesis, the matching criterion of SIFT keypoints is the minimum Euclidean distance, which is proposed as a good matching criterion in [8].

### 2.1.2 Merging

Merging is the process to stitch the registered images to create a panoramic image. When image registration is already done, a final compositing surface will be chosen. The registered images are placed on the compositing surface. At the overlapped region (among images), intensities are merged together to form an intensity for the panoramic image.

#### 2.1.2.1 Compositing surface

A panoramic image is created by placing the registered image on the compositing surface. The compositing surface can be flat, cylindrical, spherical, etc. Flat compositing surface is the simplest surface, but the stitching is then limited to the images with the field of view not exceeding 90° [11]. Cylindrical compositing surface is commonly used for stitching the larger panorama and is the natural choice for the rotating camera.

The objective of this thesis is to remove parallax objects for background stitching. Flat compositing surface is used so that the effect of parallax will be clearly seen.

#### 2.1.2.2 Merging technique

The merging techniques can be categorized into three groups: 1) optimal seam selection [1-3][14][15], 2) intensity blending [4][16][17] and 3) hybrid method [5].

**1) Optimal seam selection**

In the optimal seam selection, the algorithm searches for the line that gives the least distortion when images are stitched along this line. The intensity information is derived from one image and intensity blending is not required. Thus, the stitched image does not contain the ghosting effect, which is the partial appearance of an object (see Figure 2.8 for one such example.). However, the parallax objects may be distorted (the building in the yellow oval in Figure 2.9), because they are located at different position, when the position of the camera is changed.

**2) Intensity blending**

In the intensity blending technique, the algorithm provides the rules to blend the intensity in the overlapped region [18][19]. The average intensity is the easiest blending rule; however, it neglects the effect of objects' displacement (due to motion and parallax effect) leading to the ghosting effect. Weighted average intensity, which has the higher weight at the center of the image, can also be used as the blending rule. However, the weighted averaged intensity still faces the problem of the ghosting effect. Median intensity can be used to reduce the effect of the ghosting effect but only when the motion of the parallax or moving object is high [20]. Other complex blending methods, such as multi-resolution blending on Gaussian or Laplacian pyramid [16], gradient domain blending [17], etc., are also proposed. However, the problem of the ghosting effect is not entirely solved.

**3) Hybrid method**

In the hybrid method [5], the seam is selected and then the blending is applied in the overlapped region where the intensity between images agrees with one another. Since the seam and blending are applied to the area where intensity among frames is similar, the ghosting effect can be avoided. However, the object may appear incorrectly. Figure 2.10 shows one example of the stitched result from the hybrid method. The person holding the balloons appeared twice.

In this thesis, the median filter is selected as the merging method to evaluate the proposed system. Error due to the parallax effect will appear, when the parallax objects are shown in more than half of the available data. The error is suppressed, otherwise. This is under the assumption that the error found in more than half of the available data is hard to reject, while the error appears only in a few data can be easily rejected.

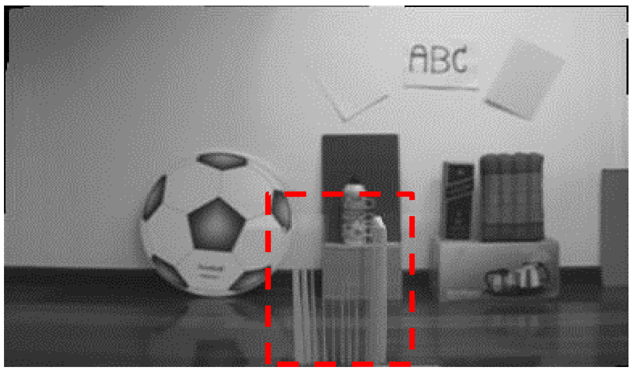
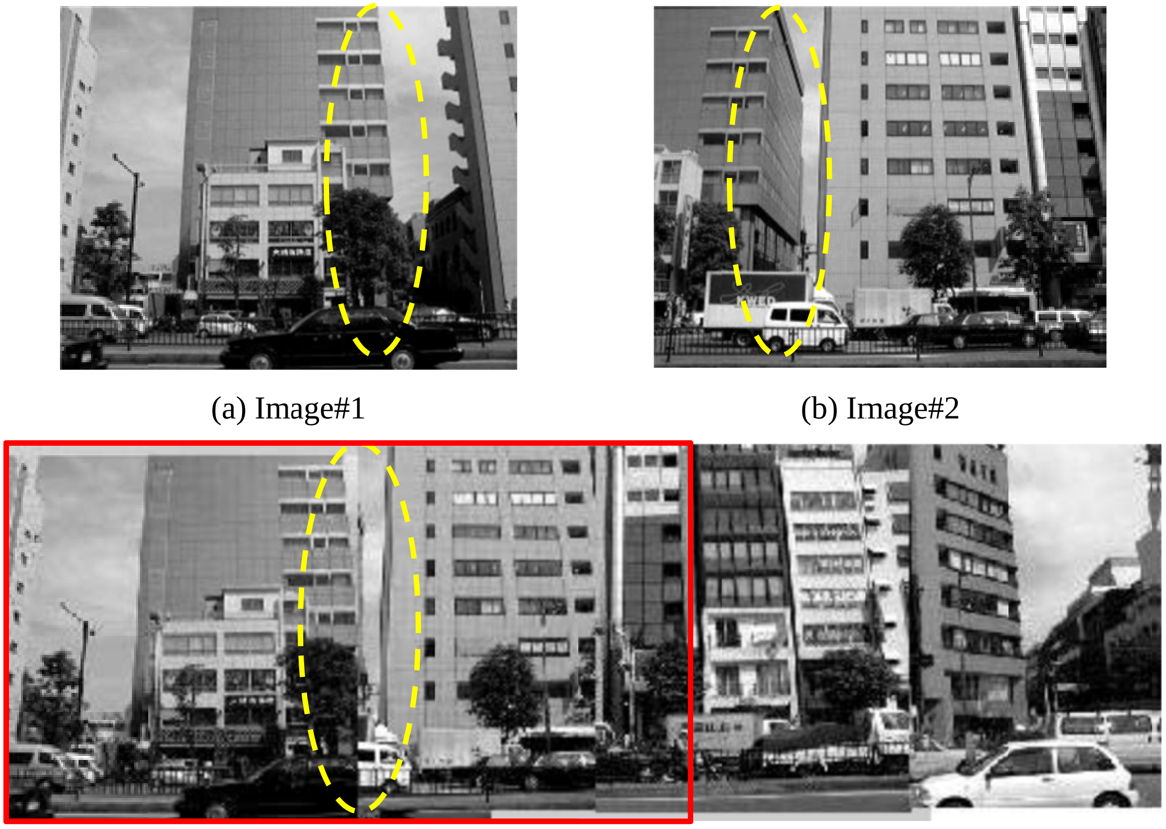


Figure 2.8: The ghosting effect of the foreground.



(a) Image#1

(b) Image#2

(c) The stitched result by optimal seam selection. The region in yellow circle was distorted due to parallax effect.

Figure 2.9: The example of the panoramic image obtained by optimal seam selection [2].

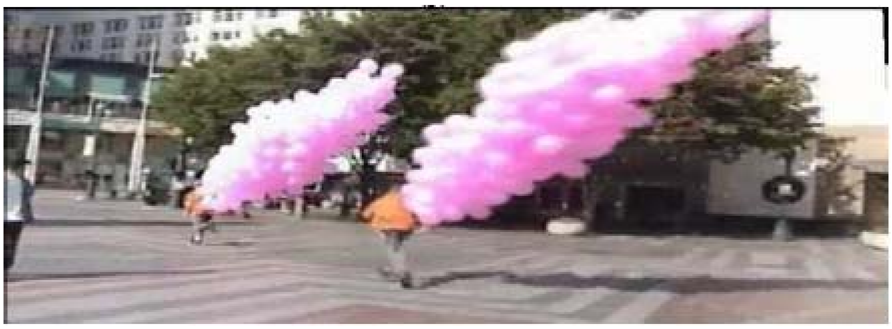


Figure 2.10: Result from Hybrid method [5].

## 2.2 Foreground rejection

Foreground rejection has the similar objective to foreground extraction, background rejection and background extraction. In this thesis, foreground rejection is used as the superset of all these four terms. Foreground rejection algorithms can be divided into two major categories: the pixel-based and the block-based rejection. The efficiency is evaluated based on rejection speed, memory requirement and rejection accuracy.

Three conventional pixel-based rejection are presented in Section 2.2.1 and two block-based rejection are presented in Section 2.2.2.

### 2.2.1 Pixel-based rejection

One of the simplest pixel-based foreground rejections is the running Gaussian average technique [21]. The background intensity is estimated as the weighted average between the intensity at the current frame and the mean intensity.

$$\mu_t = \alpha f_t + (1-\alpha)\mu_{t-1}, \qquad (2.29)$$

where $\mu_t$ is the average intensity at time $t$;

$\alpha$ is a constant with the value is [0,1].

$s_t$ is the pixel's intensity.

The pixel is the foreground if

$$\left| f_t - \mu_{t-1} \right| > k\sigma_t, \tag{2.30}$$

where $\sigma_t$ is the standard deviation of the intensity distribution;

*k* is the pre-defined constant.

The running Gaussian average has the benefit of low computation (fast) and low memory requirement. However, from (2.29) it has the drawback in that it always update the mean (background) intensity with the intensity of the current frame even when the current intensity belongs to the foreground. Koller et al. [22] added the additional variable *M* to allow the update only when the pixel is defined as the background. The update becomes

$$\mu_t = M\mu_{t-1} + (1-M)(\alpha f_t + (1-\alpha)\mu_{t-1}), \tag{2.31}$$

where *M* is the binary variable with the value of 0 and 1 for the background and the foreground pixel, respectively.

Even though a mean is a good representative, it is still not robust to noise. To increase the noise robustness, the median [23][24] can be used in place of the mean at the expense of higher computation cost. Furthermore, the memory requirement is greater, because the buffer to holds *n* previous frames is necessary. It should also be noted that the deviation from the median is more difficult to interpret than the deviation from the mean.

In natural images, background may consist of textures changing with time, such as leaves in the tree, ocean wave, etc. The mean or the median as used in the already mentioned method is not appropriate, because it can represent only one type of the background texture. In order to represent multiple textures, the mixture of Gaussian is proposed as the background model in [25]. The algorithm requires the training phase where the mean and the variance for each background texture is initially estimated. Afterwards, only the model that the pixel belongs to is updated. Though the mixture of Gaussian provides the solution for the multiple texture background, it has high computation.

Instead of finding the Gaussian pdf for each background model as in [25], Elgammal et al. [24] uses the non-parametric method, where the background is represents by $n$ previous background intensity. $n$ is in the order of 100. The distribution of the deviation is assumed to be Gaussian. The probability that the current pixel with the intensity $f_t$ belongs to the background can then be estimated as

$$P(f_t) = \frac{1}{n}\sum_{i=1}^{n}\eta\left(f_t - f_i, \sigma_t\right), \tag{2.32}$$

where $\eta$ is the Gaussian pdf;

$f_i$ is the $i$-th previous background intensity;

$\sigma_t$ is the SD which can be estimated from the change of intensity between frames.

The pixel is the foreground if $P(f_t)$ is lower than the predefined threshold.

The method in [24] has only one Gaussian distribution under the assumption that if the background contains multiple textures, the change in intensity among frame will be high leading to the higher deviation in Gaussian distribution. This method can be improved by applying mean shift [27-29] to initialize the background intensity [11]. Mean shift is not applied on-line because it requires high computation.

### 2.2.2 Block-based rejection

In block-based rejection, the intensity variation (to the temporal mean) can be used to detect and reject foreground pixels. Furthermore, by using the block of pixels, it is possible to model the variation in the neighborhood by Principal Component Analysis (PCA) [30-32]. It has high rejection accuracy at the cost of the resolution.

In [31], it is assumed that if the blocks belong to the same background, their variation will be similar in the time domain. When the intensity variation is described as the weighted sum of the principal components, two blocks of the same background should have the same weight. Since the intensity variation is included into the model, PCA based rejection provides a good result. However, the assumption of the same variation does not hold at the boundary of the object, so a number of rejection errors exist at the boundary. Another difficulty in implementation is that it is not well specified when the principal components should be updated.

In [32], PCA is applied over an image to avoid the problem of object's boundary within a block. The principal components are extracted from *n* training images. Since the foreground changes the position, it will not be well represented by this set of principal component. Therefore, the foreground is extracted as the pixel whose intensity cannot be well described by the principal components. The accuracy depends on the quality of the training images. Similar to [31], it is not well specified when the principal component should be updated.

PCA based algorithm is not suitable in a video sequence with the presence of global motion, since the background is different among frame. Background can be applied only at the overlapped region. The implementation becomes more complex. Since PCA, itself, requires high computation, the additional complexity is to be avoided.

# CHAPTER III

# THE PROPOSED METHOD

In this chapter, the proposed foreground rejection method is presented. The overview of the proposed method is described in Section 3.1. The method is divided into 2 stages: (1) edge extraction and (2) rendering. There are 2 substages in each stage. In the edge extraction stage, the edge of the foreground (parallax object) is obtained by applying logical OR operators and voting to the displaced frame difference. The edge is then refined by tracking. The edge extraction and the edge refining are presented in Sections 3.2 and 3.3, respectively. In the rendering stage, the foreground area is detected by two-stage rendering. The foreground area is then refined by tracking. Two-stage rendering and the foreground refining are described in Sections 3.4 and 3.5, respectively. The adaptation of intensity-based registration for the frames with its foreground removed is presented in Section 3.6.

## 3.1 Overview

Figure 3.1 depicts the block diagram of the video stitching algorithm implemented in this thesis. Because the camera is moving, the frame coordinates are different for each frame. Thus, the frames are first aligned to the reference frame by registration based on SIFT feature. Alignment is further refined by intensity based registration. However, a foreground (moving objects, objects with parallax effect) is an outlier and degrades the registration's accuracy. Thus, in this thesis, a foreground rejection method is applied before the intensity based registration.

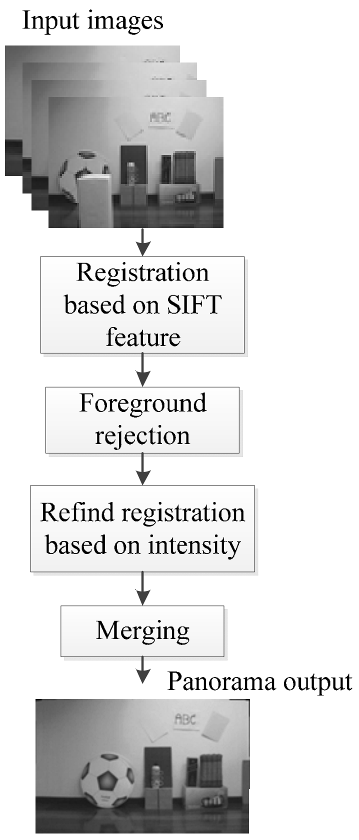

Input images
Registration based on SIFT feature
Foreground rejection
Refind registration based on intensity
Merging
Panorama output

Figure 3.1: Block diagram of the video stitching algorithm implemented in this thesis.

The demand for rejection accuracy in this thesis is not as severe as the general background subtraction previously proposed in Chapter II. There are two objectives in the proposed foreground rejection algorithm. The first objective is to remove sufficient foreground for accurate intensity based registration. The second objective is to provide the merging algorithm images with less outliers (foreground) such that the ghosting of the foreground is not apparent.

A parallax object appears at a different location in a different frame; hence, its motion is different than the global motion. Its edge can be extracted as the pixel with high Displaced Frame Difference (DFD) among aligned frames. Figure 3.2 shows the block diagram of the proposed method. As the accuracy is not severely required, the edge is extracted with the approximated location in the edge extraction stage. First, the foreground edge is extracted by applying OR operator and voting to the pixels with high DFD. The edge obtained from this substage is noisy. The tracking is then applied to refine the edge under the assumption that an object contains only one motion. Edge pixels from other frame can be used to reject the false edge. After this stage, most edge are obtained and sufficient for the subsequent rendering stage.

In the rendering stage, two-stage rendering is first applied to fill the area between edges as the foreground area. Because some edge of an object may be missing, tracking is used to fill the missing foreground. Tracking is also used to reject the area of false foreground. The detected foreground is then removed.

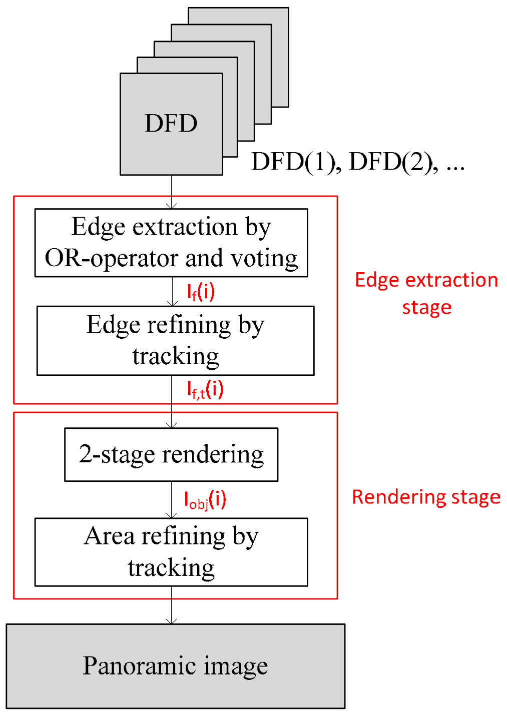


Figure 3.2: Block diagram of the proposed method.

## 3.2 Edge extraction by logical OR operator and voting

Assume that the set of images to be stitched is from a video sequence. The edge extraction algorithm for the $i-th$ frame is as follows.

Input: $I(i+nk)$: the aligned $(i+nk)-th$ frame, where $n \in \{0,1,2,3,4,5,6\}$ and $k$ is the pre-defined step size. $k$ is chosen such that the motion of the foreground is apparent.

Output: $I_{edge}(i)$: the edge of the foreground in the $i-th$ frame

The number of frames is fixed to 7 by experiment to be presented in Chapter IV. There are three steps in this algorithm: (1) extraction of intensity change regions in Section 3.2.1, (2) extraction of the near-frame n Section 3.2.2 and the far-frame foreground and (3) extraction of the foreground n Section 3.2.3.

### 3.2.1 Extraction of intensity change regions

The displaced frame difference (*DFD*) is the absolute difference between two aligned frames and is defined as

$$DFD(m) = \left| I(i) - I(i+mk) \right|, \tag{3.1}$$

where $m = \{1, 2, 3, 4, 5, 6\}$.

Pixels are classified into the intensity changing region if *DFD* is larger than the threshold, i.e.

$$CR(m) = \begin{cases} 0; & DFD(m) < T_1 \\ 1; & DFD(m) \geq T_1 \end{cases}, \tag{3.2}$$

where $CR(m)$ is the matrix indicating the changing region between the $i-th$ and the $(i+mk)-th$ frames and $T_1$ is the predefined threshold. This threshold can be automatically chosen by the method such as the one given in [7].

### 3.2.2 Extraction of the near-frame and the far-frame foreground

Even though foreground can be defined as the highly changing region, the $'1'$- elements in $CR$ do not always describe the complete foreground edge. More complete foreground is acquired by combining $CR$ of different $m$ together. The combination can be effectively done by the logical OR operator. Since $CR(m)$ consists of the foreground edge in the $i-th$ frame and the one in the $(i+mk)-th$ frame, the removal of the combination of every $CR$ leads to the excessive removal of the background regions. The foreground is classified into 2 groups:

(1) The near-frame foreground is the foreground of the $i-th$ frame and the frames near the $i-th$ frame. In this case, the $(i+k)-th$, $(i+2k)-th$ and $(i+3k)-th$ frames are considered as the near frame.

(2) The far-frame foreground is the foreground of the $i-th$ frame and the frames further away from the $i-th$ frame. In this case, the $(i+4k)-th$, $(i+5k)-th$ and $(i+6k)-th$ frames are considered as the far frame.

The near-frame foreground ($E_{near}$) and the far-frame foreground ($E_{far}$) are defined as follows.

$$\begin{aligned} E_{near} &= CR(1) \cup CR(2) \cup CR(3) \\ E_{far} &= CR(4) \cup CR(5) \cup CR(6) \end{aligned}, \tag{3.3}$$

where $\cup$ is the OR operation.

It should be noted that the number of frames for the near frame foreground and far frame foreground may be changed. Furthermore, it is also possible to categorized foreground into more than 2 groups. From the experiment to be presented in Chapter IV, it was found that this configuration (2 groups, 3 frames in each group) was the most optimal.

Morphological opening operator is applied to $E_{near}$ and $E_{far}$ to remove the small (incorrect) foreground.

### 3.2.3 Extraction of the foreground edge

The foreground of the $i-th$ frame exists in both $E_{near}$ and $E_{far}$. However, due to the simple extraction algorithm, the position in $E_{near}$ and $E_{far}$ does not always agree with each other. Hence, voting is applied to check if the foreground in $E_{near}$ is in the foreground region of $E_{far}$. The voting is described as follows.

i. For each foreground pixel at $(x_0, y_0)$ in $E_{near}$, count the number of the foreground pixels located within the $n \times n$ windows in $E_{far}$. The square window has the top-left and the bottom-right corners at $(x_0 - \left\lfloor \frac{n}{2} \right\rfloor, y_0 - \left\lfloor \frac{n}{2} \right\rfloor)$ and $(x_0 + \left\lfloor \frac{n}{2} \right\rfloor, y_0 + \left\lfloor \frac{n}{2} \right\rfloor)$, respectively.

ii. Define the pixel at $(x_0, y_0)$ as the foreground if the number of foreground pixels within the $n \times n$ window in $E_{far}$ is larger than $T_2$. From the experiment given in Chapter IV, $T_2$ and $n$ are set to 2 and 3, respectively.

## 3.3 Foreground edge refining by tracking

The result of the extraction in Section 3.2 contains the edge of a foreground and spurious background pixels (false edge) (See Figure 3.3 for one example). Since a parallax object as the foreground does not change much of its shape in a video sequence, its edges appear almost identical in every frame. It can be safely assumed that one parallax object contains only one motion; thus, it is possible to track the motion of the foreground edge. By mapping foreground edges of different frames to the current frame, it is possible to distinguish the still background from the moving foreground. In this substage, the foreground edge is tracked under 2D-translation model and the process is described in Section 3.3.1. Then the edge of other frames is mapped back to the current frame according to the motion in Section 3.3.1. The spurious background is then removed. The removal of spurious background by tracking is described in Section 3.3.2.

### 3.3.1 Tracking of foreground edges

The motion is approximated as 2D- translation and is estimated by the variance of the 3-step search. The input are the extracted foreground in the $i+m-th$ frame, where $m=\{0,1,2,...,n\}$. The number of $n$ is determined according to the perceived motion of the foreground. It is fixed at 4 in this thesis. The motion of the foreground is estimated according to the following optimization problem:

$$\left(T_{x,best},T_{y,best}\right)=\underset{(T_x,T_y)}{\arg\max}\sum_{k=1}^{n}\sum_{(x,y)}I_f(x+T_x,y+T_y,i+k-1)\bullet I_f(x,y,i+k), \qquad (3.4)$$

where $I_f(x,y,i)$ is the foreground edge at $\left(x,y\right)$ coordinate in the $i-th$ frame; it is 1 and 0 for the foreground and the background pixel, respectively;

$(T_{x,best},T_{y,best})$ is the translation motion of the foreground;

$n$ is the predetermined constant and is fixed at 4 in this thesis;

$\bullet$ represents the dot operator.

The estimation procedure is as follows.

1. Initialization: define $iteration = 0$, $T_{x,best} = T_{y,best} = 0$, $\Delta x = 10$ and $\Delta y = 1$. Note that the step size in x axis ($\Delta x$) is much larger than the one in y axis ($\Delta y$) because the major motion in most video sequence is the horizontal motion. $\Delta x, \Delta y$ can be changed to accommodate the motion characteristic.
2. Estimation for $(T_{x,best}, T_{y,best})$:

   (1) Let $T_x = \{T_{x,best}, T_{x,best} + \Delta x, T_{x,best} - \Delta x\}$ and $T_y = \{T_{y,best}, T_{y,best} + \Delta y, T_{y,best} - \Delta y\}$

   (2) Find the match between the edge extracted in frame# $i+k-1$ and frame# $i+k$, where $k = \{1, 2, ..., n\}$, as follows.

$$match(T_x, T_y) = \sum_{k=1}^{n} \sum_{(x,y)} I_f(x+T_x, y+T_y, i+k-1) \bullet I_f(x, y, i+k) \tag{3.5}$$

   (3) Update $(T_{x,best}, T_{y,best})$ according to

$$(T_{x,best}, T_{y,best}) = \arg\max_{(T_x, T_y)} match(T_x, T_y) \tag{3.6}$$

3. Update system parameters: $\Delta x = \frac{\Delta x}{2}$, $\Delta y = \frac{\Delta y}{2}$ and $iteration = iteration + 1$.
4. Termination: terminate if $iteration = 5$; otherwise, go to 2.

This above process is designed under the assumption that there is only one foreground object in an image. If there are more than one foreground, iteration is required and the optimal motion is not guaranteed.

### 3.3.2 Spurious background removal

In this step, the foreground edges of different frames are mapped back to the current frame according to $(T_{x,best}, T_{y,best})$ in Section 3.3.1. Background pixels do not move; hence, they are mapped to different locations. On the other hand, foreground pixels move with approximately the same motion; thus, they are mapped to the same location. Since the location of edge pixels are not accurate, the foreground is dilated before pixel counting. The pixel at $(x, y)$ in the $i-th$ frame is the foreground, if

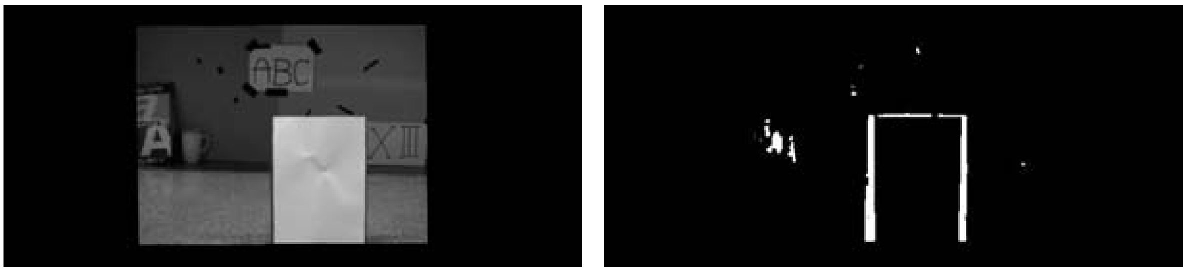
$$\sum_{k=1}^{n} I_f(x, y, i) \bullet I_{f,D}(x - kT_{x,best}, y - kT_{y,best}, i + k) \geq T_3, \tag{3.7}$$

where $I_{f,D}$ is the $I_f$ dilated with the structure element of a $3 \times 3$ square;

$T_3$ is the pre-defined threshold.

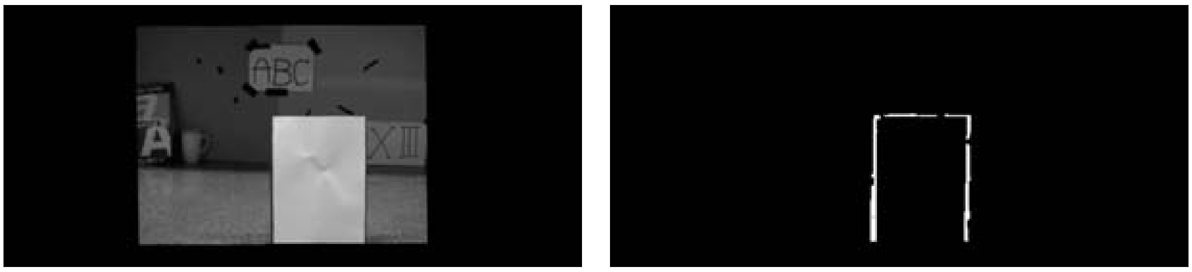
The value $n$ and $T_3$ are determined in accordance to the perceived motion of a foreground. It is assumed that if more than half of $n$ frames consider the pixel at $(x, y)$ as a foreground, it is a foreground. Since $n$ is fixed at 4, $T_3$ is set to 2.

The remaining spurious background is removed by morphological opening with a structure element of a $2 \times 2$ square.



(a) Original image (b) Detected foreground

Figure 3.3: The example of extracted edges with spurious edges.



(a) Original image (b) Detected foreground

Figure 3.4: The example of the extracted foreground after Section 3.3.

## 3.4 Two-stage rendering

Figure 3.4 shows the example of the result of the refined foreground from Section 3.3. The figure clearly shows that the extracted foreground was mostly the edge. The object is extracted by the two-stage rendering which consist of (1) simple rendering (described in Section 3.4.1) and (2) hole filling (described in Section 3.4.2).

### 3.4.1 Simple rendering

In the simple rendering, it is hypothesized that the edge extraction provides the pair of the vertical edges (the left and the right edges) and/or the pair of the horizontal edges (the top and the bottom edges). The rendering is to fill the area between the horizontal and the vertical edges as the object.

Input: $I_{f,t}(i)$: the foreground edge extracted by the algorithm described in Sections 3.2 and 3.3.

i. **Initialization:** set every element in $I_{obj}(i)$ to 0.

ii. **Row rendering:** for every row, do as follows.

(a) Initialization: set *startPix*, $p$ and *contFore* to -1, 1 and 0, respectively.

```
(b) If the pixel at the $p-th$ column is the foreground,
    then
        If $contFore$ is 0,
        then
            If $startPix$ is -1,
            then
                Set $startPix$ to $p$.
            else
                If the rendering length is not greater than the threshold, $T_4$.
                render the pixel between the $startPix$ and $p$ to foreground.
                Set $startPix$ to -1.
        end
        Set $contFore$ to 1.
    else
        Set $contFore$ to 0.
    end
```

(c) Increment $p$ by 1.

(d) If $p$ is not larger than the image's width, go to b. Otherwise, terminate.

iii. **Column rendering**

The procedure is similar to the row rendering. The algorithm in the row rendering is changed to the column rendering as follows.

1) Replace the word 'column' with 'row' and 'row' with 'column' in Step (b)
2) The stopping criterion in Step (d) is changed to "If $p$ is not larger than the image's height, go to b. Otherwise, terminate."

The threshold in this stage is set in accordance with the size of a parallax object. Figure 3.5 depicts one example of simple rendering. The white pixels are the foreground pixels. $T_4$ is assumed to be infinite. The black (background) pixels between the 3rd to the 7th column is considered to be inside the object whose left and right edges are at the 2nd and the 8th column, respectively. The black pixels in the 10th and 11th column is considered to be between the edge of the different objects and is considered background. With the similar reason to the ones between the 3rd to the 7th column, the pixel at the 14th column is considered foreground.

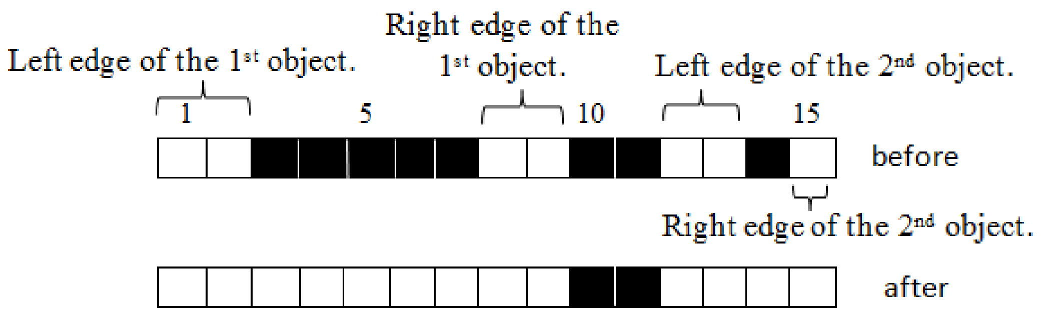
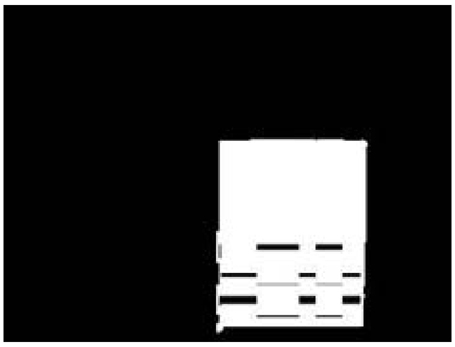

Left edge of the 1st object.
Right edge of the 1st object.
Left edge of the 2nd object.
1
5
10
15
before
Right edge of the 2nd object.
after

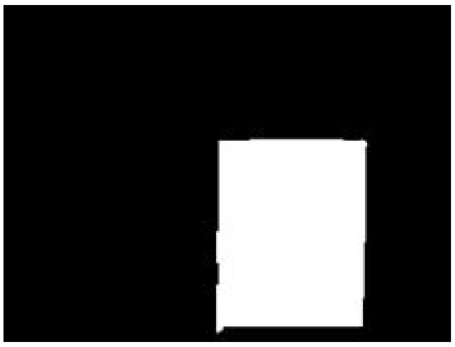
Figure 3.5: The example of simple rendering.

### 3.4.2 Hole filling

The foreground rendering in the previous stage may contain small holes (See Figure 3.6 (a) for one such example). In this stage, the small background engulfed by the foreground is rendered as background. The algorithm is easily adapted from the simple rendering algorithm in Section 3.4.1. “set *startPix* to -1” in Step (b) of simple rendering is removed. The thresholds in this stage are manually set and are always smaller than the one in the first stage. Figure 3.7 depicts the example of hole filling in a row. The white and the black pixels are the foreground and the background, respectively. The maximum filling size (threshold) is set to 4. Since the small background region at the 4th and the 5th column is engulfed by the first and the second objects, it is filled up and considered as foreground. On the other hand, the large background region between the 9th and 15th column is left as it is. The rendered foreground after hole-filling is shown in Figure 3.6 (b).



(a) after simple rendering



(b) after hole filling

Figure 3.6: The example of the rendered foreground.

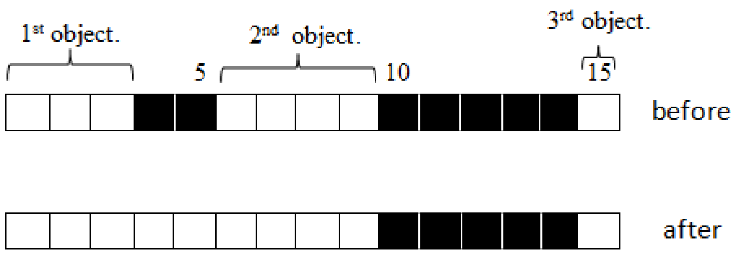


Figure 3.7: The example of hole-filling.

## 3.5 Foreground area refining by tracking

When a parallax object starts appearing from the edge of an image or goes out of the field of view, its horizontal and its vertical edges may not appear in pair. Two-stage rendering has no mechanism to decide which side of the edge the object belongs to. The object area is not extracted. Furthermore, the remaining spurious background leads to the mismatching edge. Consequently, the background is extracted as part of the foreground. Since the parallax object appears in many frames with very little change in shape, it is possible to use the parallax object in other frames to obtain the missing foreground and to remove the background. The algorithm is similar to the one in Section 3.3. The foreground motion can be taken from the result of Section 3.3.1. The pixel at $(x, y)$ is considered as foreground if

$$\sum_{k=0}^{n} I_f(x - kT_{x,best}, y - kT_{y,best}, i + k) \geq T_5, \tag{3.8}$$

where $T_5$ is the pre-defined threshold;

$n$ is the number of frame, similar to the one in Section 3.3.1.

The pixel is foreground if more than half of $I_f$ consider it as foreground. $n$ is set to 4 as in Section 3.3.1 so there are 5 frames in (8); hence, $T_5$ is set to 3.

## 3.6 Registration with foreground rejection

Since foreground is an outlier and its exact location is not known, its information should not be included in the registration. The MSE matching criterion is changed to

$$MSE\left(T\right)=\frac{1}{\left|M\right|}\sum_{(x,y)}\left[f\left(x,y,k\right)-f\left(T(x,y),k'\right)\right]^{2}M(x,y,T(x,y),k,k'), \qquad (3.9)$$

where $M(x,y,T(x,y),k,k')=(1-I_{obj}(x,y,k))(1-I_{obj}(T(x,y),k'));$

$$\left|M\right|=\sum_{(x,y)}M(x,y,T(x,y),k,k').$$

The remaining algorithm is the same as the registration without foreground rejection.

# CHAPTER IV

# EXPERIMENT AND DISCUSSION

## 4.1 Experimental set-up

### 4.1.1 Hardware and software specification

The proposed algorithm was implemented using MATLAB R20011a on Dell Studio 1458 notebook with the following environment:

- Operation system: Microsoft Windows 7 Ultimate 64 bits
- CPU: Intel Core I5-520m
- Memory: 4 GB

### 4.1.2 Setting

Thirteen video sequences were used to evaluate the proposed algorithm. In each video sequence, there is only one parallax object between the camera and the background. Twenty-one frames from the test video sequence were extracted and stitched to create a panoramic image. The thirteenth video sequences had the vertical motion as the major motion, so $\Delta x$ and $\Delta y$ in tracking algorithm (Sections 3.3 and 3.5) were swapped. The selected frames are summarized in Table 4.1. Every frame of twelve video sequences was converted to a gray-scale image and rescaled to the size of 320×240 pixels. In addition, the thirteenth video sequence was rescaled to the size of 480×270. The characteristics of 13 video sequences are described in Table 4.2. Sample frames of all thirteen video sequences are shown in Figures 4.1 - 4.13.

The experiment was divided into 4 parts. First, the configuration and the fixed parameters as described in Chapter III were evaluated in Section 4.2. The parameters related to the characteristics of a video sequence were not considered. Then the effect of refining by tracking to the foreground rejection was evaluated in Section 4.3. After that the effects of the foreground rejection to the registration accuracy and stitching were investigated in Sections 4.4 and 4.5, respectively.

Table 4.1: The selected frames in the test video sequence.

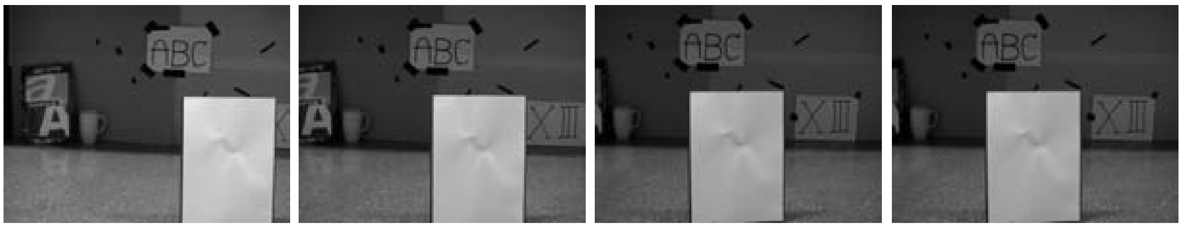
| No. | Total number of frames | Start frame | Reference frame | Last frame | Selected frames $i = \{0,1,2,\ldots,20\}$ |
|---|---|---|---|---|---|
| 1 | 340 | 10 | 160 | 310 | $(10+15i)-th$ |
| 2 | 350 | 50 | 150 | 250 | $(50+10i)-th$ |
| 3 | 550 | 120 | 220 | 320 | $(120+10i)-th$ |
| 4 | 400 | 120 | 220 | 320 | $(120+10i)-th$ |
| 5 | 600 | 30 | 180 | 330 | $(30+15i)-th$ |
| 6 | 290 | 90 | 140 | 190 | $(90+5i)-th$ |
| 7 | 330 | 110 | 170 | 230 | $(110+6i)-th$ |
| 8 | 420 | 155 | 215 | 275 | $(155+6i)-th$ |
| 9 | 305 | 10 | 110 | 210 | $(10+10i)-th$ |
| 10 | 350 | 160 | 210 | 260 | $(160+5i)-th$ |
| 11 | 320 | 160 | 240 | 320 | $(160+8i)-th$ |
| 12 | 550 | 120 | 220 | 320 | $(120+10i)-th$ |
| 13 | 580 | 120 | 220 | 320 | $(120+10i)-th$ |

Table 4.2: The characteristics of the test video sequences.

| No. | Major motion (Translation Direction) | Type of background | Shape of foreground object |
|---|---|---|---|
| 1 | horizontal | indoor | square |
| 2 | horizontal | indoor | square |
| 3 | horizontal | outdoor | cylinder |
| 4 | horizontal | indoor | square |
| 5 | horizontal | indoor | square |
| 6 | horizontal | indoor | complex |
| 7 | horizontal | indoor | ellipse |
| 8 | horizontal | indoor | complex |
| 9 | horizontal | indoor | square |
| 10 | horizontal | outdoor | cylinder |
| 11 | horizontal | outdoor | cylinder |
| 12 | horizontal | outdoor | cylinder |
| 13 | vertical | indoor | square |

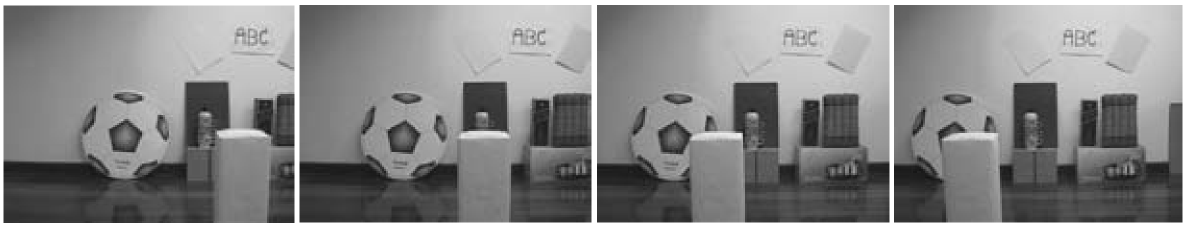
### 4.1.3 Evaluation method

The evaluation was done by visual inspection because there was no ground truth. Median filter was used to stitch image as the error would not be easily concealed and would be visible, when more than half of the available data were incorrect.



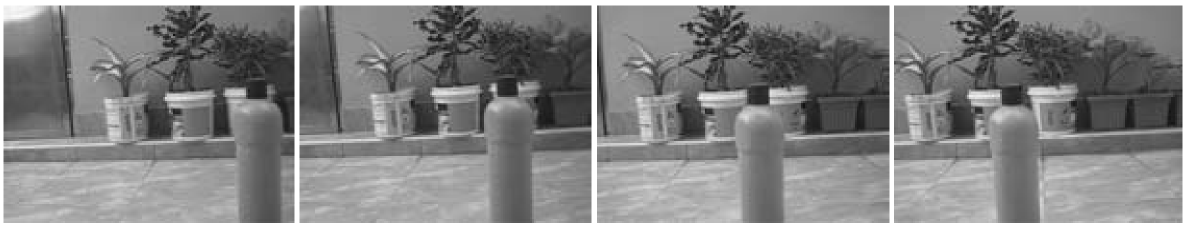
(a) frame#10 (b) frame#130 (c) frame#235 (d) frame#310

Figure 4.1: Sample frames of the first video sequence.



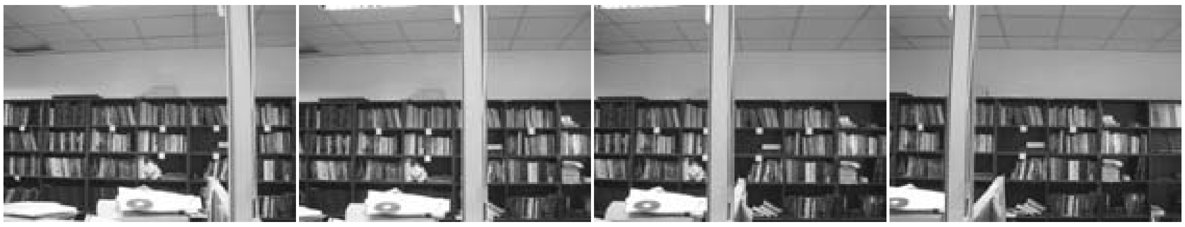
(a) frame#50 (b) frame#130 (c) frame#200 (d) frame#250

Figure 4.2: Sample frames of the second video sequence.



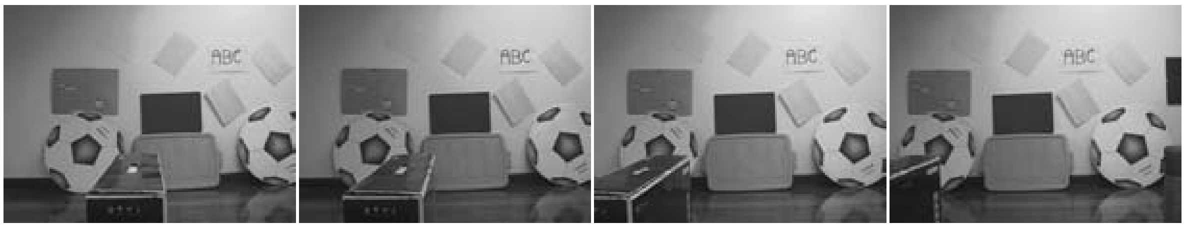
(a) frame#120 (b) frame#200 (c) frame#270 (d) frame#320

Figure 4.3: Sample frames of the third video sequence.



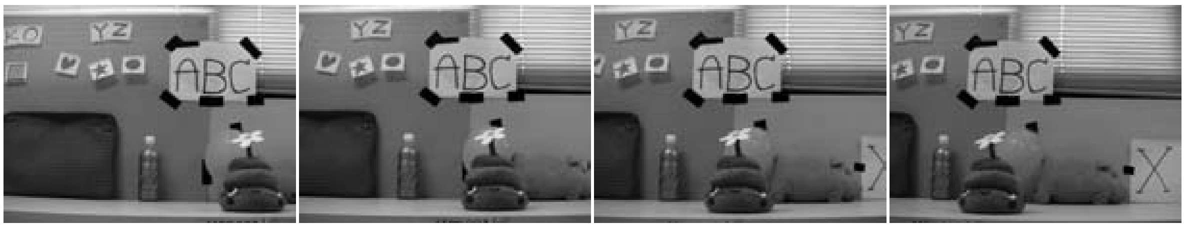
(a) frame#120 (b) frame#200 (c) frame#270 (d) frame#320

Figure 4.4: Sample frames of the forth video sequence.



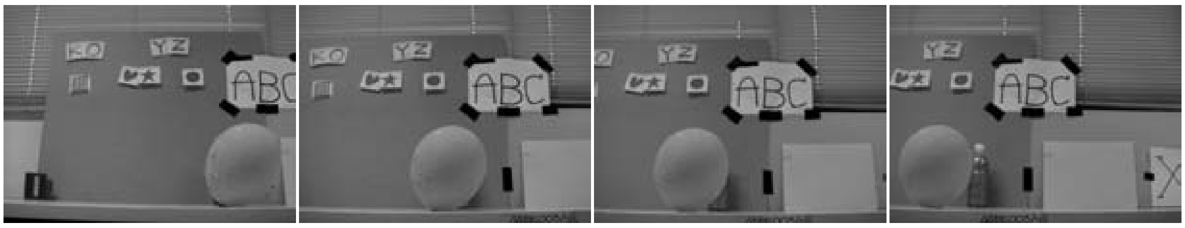
(a) frame#30 (b) frame#150 (c) frame#255 (d) frame#330

Figure 4.5: Sample frames of the fifth video sequence.



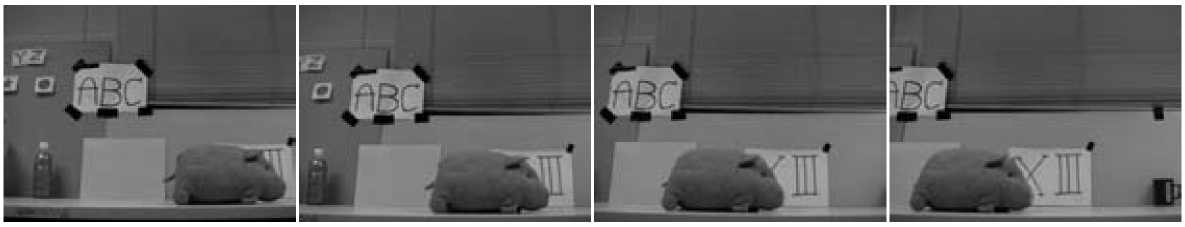
(a) frame#90 (b) frame#130 (c) frame#165 (d) frame#190

Figure 4.6: Sample frames of the sixth video sequence.



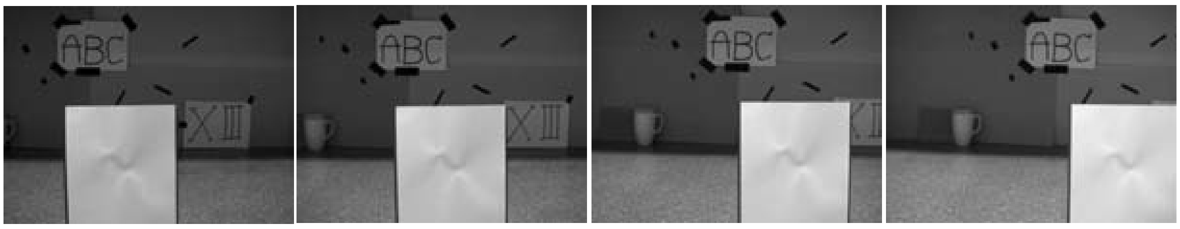
(a) frame#110 (b) frame#158 (c) frame#200 (d) frame#230

Figure 4.7: Sample frames of the seventh video sequence.



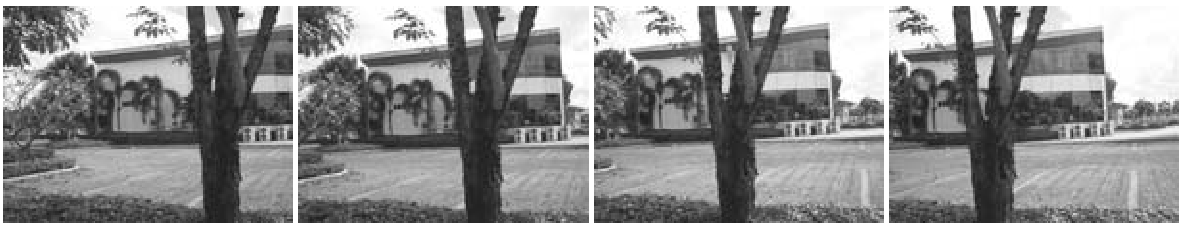
(a) frame#155 (b) frame#203 (c) frame#245 (d) frame#275

Figure 4.8: Sample frames of the eighth video sequence.



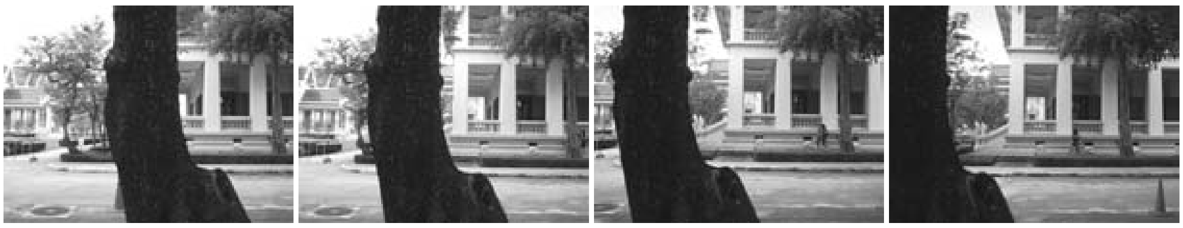
(a) frame#10 (b) frame#90 (c) frame#160 (d) frame#210

Figure 4.9: Sample frames of the ninth video sequence.



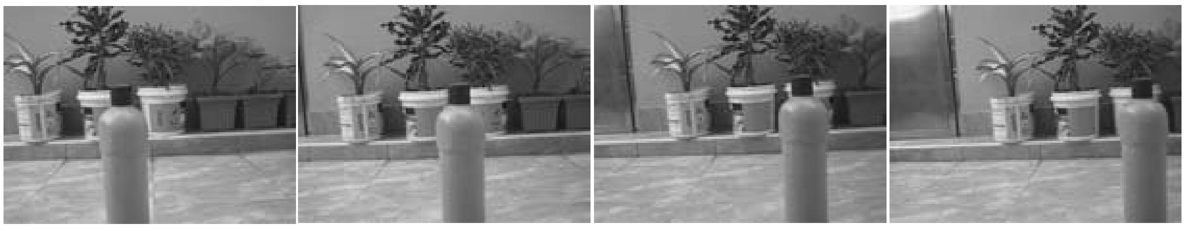
(a) frame#160 (b) frame#200 (c) frame#235 (d) frame#260

Figure 4.10: Sample frames of the tenth video sequence.



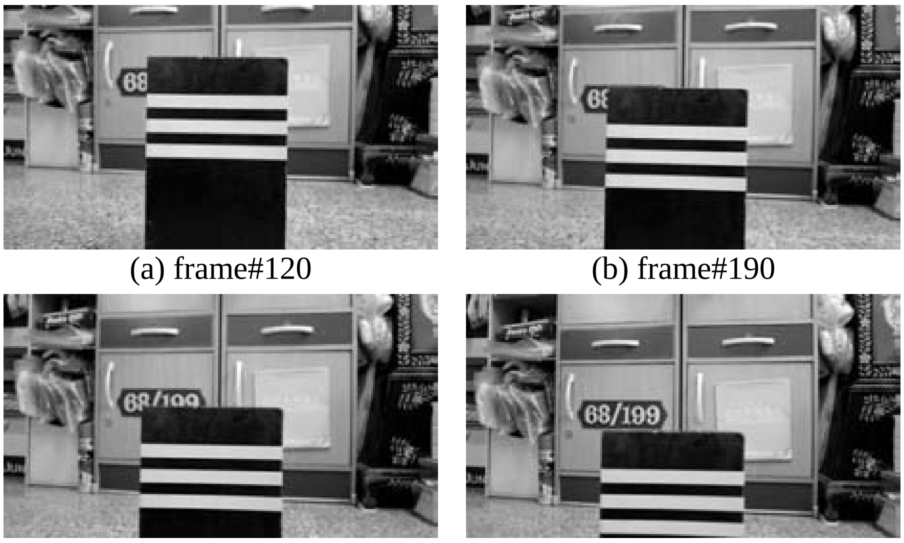
(a) frame#160 (b) frame#224 (c) frame#280 (d) frame#320

Figure 4.11: Sample frames of the eleventh video sequence.



(a) frame#120 (b) frame#200 (c) frame#270 (d) frame#320

Figure 4.12: Sample frames of the twelfth video sequence.



(a) frame#120 (b) frame#190

(c) frame#260 (d) frame#320

Figure 4.13: Sample frames of the thirteenth video sequence.

## 4.2 System parameter evaluation

The effect of voting window size and the threshold $(T_2)$ are investigated. In Section 3.2.3, the window size is 3×3 and $T_2$ is fixed to be 3. In this section, the window size was varied from 3×3, 5×5 and 7×7. The values of $T_2$ were varied according to the percentage of the window size from 30% to 80% at the step of 10%. The values of $T_2$ are summarized in Table 4.3.

Table 4.3: $T_2$ used in the experiment.

| Voting window size | The value of $T_2$ | | | | | |
|---|---|---|---|---|---|---|
| | 30% | 40% | 50% | 60% | 70% | 80% |
| $3\times3$ (9 pixels) | $2.7\approx3$ | $3.6\approx4$ | $4.5\approx5$ | $5.4\approx5$ | $6.3\approx6$ | $7.2\approx7$ |
| $5\times5$ (25 pixels) | $7.5\approx8$ | 10 | $12.5\approx13$ | 15 | $17.5\approx18$ | 20 |
| $7\times7$ (49 pixels) | $14.7\approx15$ | $19.6\approx20$ | $24.5\approx25$ | $29.4\approx29$ | $34.3\approx34$ | $39.2\approx39$ |

It is possible to categorize foreground edge differently than the one described in Section 3.2.2. Edge pixels can be categorized into more than 2 groups (near-frame foreground and far-frame foreground). Furthermore, the number of frame in each group can be different. In this section, the effect of the number of groups (R) and the number of frame (C) is investigated. Seven configurations were investigated. They were 2R2C, 2R3C, 2R4C, 3R2C, 3R3C, 3R4C and 4R4C, where *n*R*m*C stands for *n* groups and each group consists of *m* frames. The changing region (CR), as defined in Equation (3.2), of frames within each group were combined together to form an edge group. The $i-th$ group in *n*R*m*C configuration had its edge group ($E_i$) defined as follows.

$$E_i = CR(im+1)\cup CR(im+2)\cup ...\cup CR(im+m), \tag{4.1}$$

where $CR(k)$ is *CR* of the $k-th$ frame;

$\cup$ is an OR operator.

Table 4.4 shows the summarization of the above process.

Table 4.4: The calculation of edge groups in *n*R*m*C configuration

| Type | | 2R2C | 2R3C | 2R4C | 3R2C | 3R3C | 3R4C | 4R4C |
|---|---|---|---|---|---|---|---|---|
| No. of groups | | 2 | 2 | 2 | 3 | 3 | 3 | 4 |
| No. of frames in a group | | 2 | 3 | 4 | 2 | 3 | 4 | 4 |
| Edge pixel in a group | First group ($E_1$) | $CR(1)\cup CR(2)$ | $CR(1)\cup CR(2)\cup CR(3)$ | $CR(1)\cup CR(2)\cup CR(3)\cup CR(4)$ | $CR(1)\cup CR(2)$ | $CR(1)\cup CR(2)\cup CR(3)$ | $CR(1)\cup CR(2)\cup CR(3)\cup CR(4)$ | $CR(1)\cup CR(2)\cup CR(3)\cup CR(4)$ |
| | Second group ($E_2$) | $CR(3)\cup CR(4)$ | $CR(4)\cup CR(5)\cup CR(6)$ | $CR(5)\cup CR(6)\cup CR(7)\cup CR(8)$ | $CR(3)\cup CR(4)$ | $CR(4)\cup CR(5)\cup CR(6)$ | $CR(5)\cup CR(6)\cup CR(7)\cup CR(8)$ | $CR(5)\cup CR(6)\cup CR(7)\cup CR(8)$ |
| | Third group ($E_3$) | N/A | N/A | N/A | $CR(5)\cup CR6)$ | $CR(7)\cup CR(8)\cup CR(9)$ | $CR(9)\cup CR(10)\cup CR(11)\cup CR(12)$ | $CR(9)\cup CR(10)\cup CR(11)\cup CR(12)$ |
| | Fourth group ($E_4$) | N/A | N/A | N/A | N/A | N/A | N/A | $CR(13)\cup CR(14)\cup CR(15)\cup CR(16)$ |

Note: $CR(m)$ stands for the changing region as calculated between the $i-th$ frame and the $i+m-th$ frame (Section 3.2.1).

$\cup$ stands for the logical OR operator.

The edge groups were merged as follows.

1) Apply voting (Section 3.2.3) to (1) $\mathrm{E}_1$ and $\mathrm{E}_2$, (2) $\mathrm{E}_1$ and $\mathrm{E}_3$ and (3) $\mathrm{E}_1$ and $\mathrm{E}_4$. (Omit the pair that contains $\mathrm{E}_i$ with N/A)
2) Merge all the result by AND operator.

The evaluation matrices in this experiment are adapted from the accuracy [34] and the sensitivity [35] measurement in binary classification which are defined as follows.

$$\text{accuracy} = \frac{\text{number of true positives+number of true negatives}}{\text{number of true positives} + \text{false positives+false negatives+true negatives}} \quad [34] \qquad (4.2)$$

$$\text{sensitivity} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}} \quad [35] \qquad (4.3)$$

Edge pixels were considered as positive pixels. The ground truth edge was manually created. The true positive was the ground truth edge that was detected by the algorithm in Section 3.2. The false positive pixels were the pixels incorrectly detected as edge pixels (according to the ground truth). The true negative pixels were the pixels correctly detected as non-edge pixels. The false negative pixels were the ground truth edge pixels that the algorithm failed to detect. The number of the background (negative) pixels was much larger than the number of the edge (positive) pixels; therefore, the numbers of the true and false negative pixels were not included in the accuracy calculation. In this experiment, the accuracy measurement defined in (4.2) was calculated as follows.

$$\text{accuracy} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false positives}} \qquad (4.4)$$

The requirement of the good foreground detection is the high accuracy and the high sensitivity. In the same algorithm, the accuracy can be increased at the expense of the sensitivity, and vice versa.

The first to the forth video sequences were used in this investigation of the accuracy and the sensitivity at different voting window size and $T_2$ for different configuration. The average accuracy and the average sensitivity in percent were presented in Tables 4.4 - 4.10. Each table shows the result at different R and C. The size of the voting window had only a little effect on the accuracy and the sensitivity. Except in the case of 4R4C, most of the best results were acquired with the voting window of $3\times3$.

Table 4.12 shows the accuracy and the sensitivity when the voting size was $3\times3$. In most cases, the highest value was obtained when $T_2$ = 3. There were 5 cases (2R3C, 2R4C, 3R2C, 3R3C and 3R4C), where the sensitivity was higher than 30%. 3R3C and 3R4C required more frames than the other three cases; however, they did not provide much higher accuracy and sensitivity. Therefore, 3R3C and 3R4C were ignored. 2R4C provided the lowest accuracy and used the highest number of frames among the remaining three cases, so it was not optimal. Regarding 2R3C and 3R2C, further investigation was required. Figures 4.14 and 4.15 show the extraction results by 2R3C and 3R2C. In both cases, there were only a few edges extracted by 3R2C, while almost complete edge was extracted by 2R3C. So 2R3C was considered as the most optimal configuration.

Table 4.5: The accuracy and the sensitivity at different voting window size and $T_2$ for 2R2C configuration.

| $T_2$ in term of the percent of window size. | Evaluation metrics (%) | voting window size | | |
|---|---|---|---|---|
| | | 3×3 | 5×5 | 7×7 |
| 30% | accuracy | **29** | **29** | **29** |
| | sensitivity | **29** | **29** | **29** |
| 40% | accuracy | 28 | **29** | **29** |
| | sensitivity | 28 | **29** | **29** |
| 50% | accuracy | **27** | **27** | **27** |
| | sensitivity | **28** | 27 | 27 |
| 60% | accuracy | **27** | 23 | 24 |
| | sensitivity | **27** | 21 | 21 |
| 70% | accuracy | **23** | **23** | 22 |
| | sensitivity | **20** | **20** | 19 |
| 80% | accuracy | **22** | **22** | 19 |
| | sensitivity | **19** | **19** | 13 |

Table 4.6: The accuracy and the sensitivity at different voting window size and $T_2$ for 2R3C configuration.

| $T_2$ in term of the percent of window size. | Evaluation metrics (%) | voting window size | | |
|---|---|---|---|---|
| | | 3×3 | 5×5 | 7×7 |
| 30% | accuracy | **20** | **20** | **20** |
| | sensitivity | 44 | 44 | **45** |
| 40% | accuracy | 19 | **20** | **20** |
| | sensitivity | 42 | **44** | 43 |
| 50% | accuracy | **19** | **19** | 18 |
| | sensitivity | **41** | **41** | 40 |
| 60% | accuracy | **19** | 16 | 15 |
| | sensitivity | **41** | 33 | 32 |
| 70% | accuracy | **16** | 15 | 14 |
| | sensitivity | **32** | 31 | 29 |
| 80% | accuracy | **16** | 14 | 11 |
| | sensitivity | **32** | 30 | 22 |

Table 4.7: The accuracy and the sensitivity at different voting window size and $T_2$ for 2R4C configuration.

| $T_2$ in term of the percent of window size. | Evaluation metrics (%) | voting window size | | |
|---|---|---|---|---|
| | | 3×3 | 5×5 | 7×7 |
| 30% | accuracy | 16 | 16 | **17** |
| | sensitivity | 52 | 52 | **53** |
| 40% | accuracy | **16** | **16** | **16** |
| | sensitivity | 48 | **51** | **51** |
| 50% | accuracy | **15** | **15** | **15** |
| | sensitivity | **47** | **47** | 46 |
| 60% | accuracy | **15** | 12 | 12 |
| | sensitivity | **46** | 36 | 36 |
| 70% | accuracy | **12** | **12** | 11 |
| | sensitivity | **35** | 34 | 30 |
| 80% | accuracy | **12** | 11 | 8 |
| | sensitivity | **34** | 32 | 21 |

Table 4.8: The accuracy and the sensitivity at different voting window size and $T_2$ for 3R2C configuration.

| $T_2$ in term of the percent of window size. | Evaluation metrics (%) | voting window size | | |
|---|---|---|---|---|
| | | 3×3 | 5×5 | 7×7 |
| 30% | accuracy | **34** | **34** | **34** |
| | sensitivity | 34 | 34 | **35** |
| 40% | accuracy | 32 | **34** | **34** |
| | sensitivity | 31 | **34** | **34** |
| 50% | accuracy | **32** | **32** | 31 |
| | sensitivity | **31** | 30 | 30 |
| 60% | accuracy | **32** | 26 | 25 |
| | sensitivity | **30** | 22 | 22 |
| 70% | accuracy | **26** | 25 | 23 |
| | sensitivity | **22** | 21 | 19 |
| 80% | accuracy | **26** | 23 | 16 |
| | sensitivity | **21** | 19 | 12 |

Table 4.9: The accuracy and the sensitivity at different voting window size and $T_2$ for 3R3C configuration.

| $T_2$ in term of the percent of window size. | Evaluation metrics (%) | voting window size | | |
|---|---|---|---|---|
| | | 3×3 | 5×5 | 7×7 |
| 30% | accuracy | **24** | **24** | **24** |
| | sensitivity | 39 | **40** | **40** |
| 40% | accuracy | 23 | **24** | **24** |
| | sensitivity | 36 | **39** | **39** |
| 50% | accuracy | **23** | **23** | 22 |
| | sensitivity | **35** | **35** | 34 |
| 60% | accuracy | **23** | 19 | 19 |
| | sensitivity | **35** | 27 | 27 |
| 70% | accuracy | **19** | **19** | 18 |
| | sensitivity | **26** | **26** | 23 |
| 80% | accuracy | **19** | 18 | 12 |
| | sensitivity | **26** | 24 | 17 |

Table 4.10: The accuracy and the sensitivity at different voting window size and $T_2$ for 3R4C configuration.

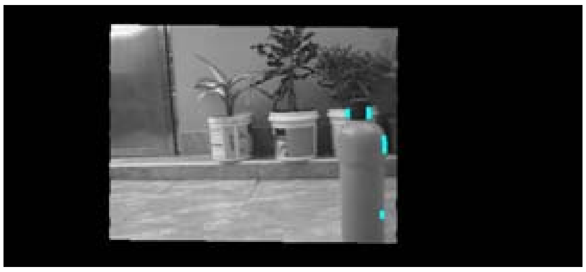
| $T_2$ in term of the percent of window size. | Evaluation metrics (%) | voting window size | | |
|---|---|---|---|---|
| | | 3×3 | 5×5 | 7×7 |
| 30% | accuracy | **16** | **16** | **16** |
| | sensitivity | 46 | **47** | **47** |
| 40% | accuracy | 15 | **16** | **16** |
| | sensitivity | 42 | **46** | **46** |
| 50% | accuracy | **15** | **15** | **15** |
| | sensitivity | **41** | 40 | 40 |
| 60% | accuracy | **15** | 12 | 11 |
| | sensitivity | **40** | 30 | 29 |
| 70% | accuracy | **11** | **11** | 10 |
| | sensitivity | **28** | **28** | 25 |
| 80% | accuracy | **11** | 10 | 7 |
| | sensitivity | **27** | 25 | 16 |

Table 4.11: The accuracy and the sensitivity at different voting window size and $T_2$ for 4R4C configuration.

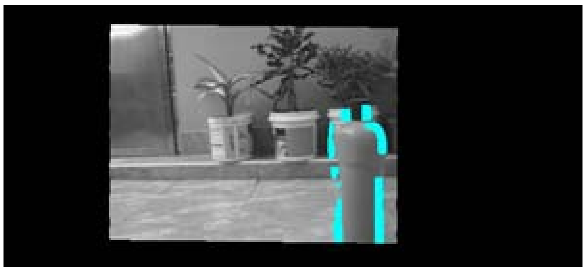
| $T_2$ in term of the percent of window size. | Evaluation metrics (%) | voting window size | | |
|---|---|---|---|---|
| | | 3×3 | 5×5 | 7×7 |
| 30% | accuracy | **1** | **1** | **1** |
| | sensitivity | **5** | **5** | 4 |
| 40% | accuracy | **2** | 1 | 1 |
| | sensitivity | **8** | 6 | 6 |
| 50% | accuracy | 2 | 2 | **3** |
| | sensitivity | **8** | **8** | **8** |
| 60% | accuracy | 2 | **4** | **4** |
| | sensitivity | 8 | **11** | **11** |
| 70% | accuracy | **4** | **4** | **4** |
| | sensitivity | 10 | **11** | **11** |
| 80% | accuracy | 4 | 4 | **5** |
| | sensitivity | 10 | 11 | **12** |

Table 4.12: The accuracy and the sensitivity at different configuration and $T_2$ when the size of the voting window was $3\times3$. The cases with the highest accuracy and the highest sensitivity are shown in gray.

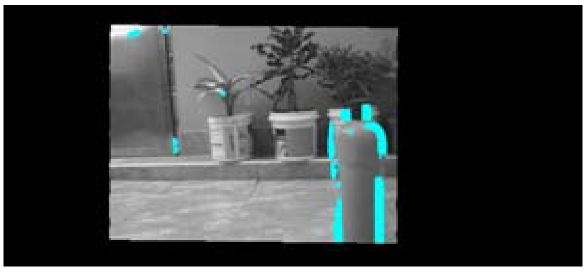
| $T_2$ | Evaluation metrics (%) | Configuration | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 2R2C | 2R3C | 2R4C | 3R2C | 3R3C | 3R4C | 4R4C |
| 3 | accuracy | 29 | 20 | 16 | 34 | 24 | 16 | 1 |
| | sensitivity | 29 | 44 | 52 | 34 | 39 | 46 | 5 |
| 4 | accuracy | 28 | 19 | 16 | 32 | 23 | 15 | 2 |
| | sensitivity | 28 | 42 | 48 | 31 | 36 | 42 | 8 |
| 5 | accuracy | 27 | 19 | 15 | 32 | 23 | 15 | 2 |
| | sensitivity | 28 | 41 | 47 | 31 | 35 | 41 | 8 |
| 6 | accuracy | 27 | 19 | 15 | 32 | 23 | 15 | 2 |
| | sensitivity | 27 | 41 | 46 | 30 | 35 | 40 | 8 |
| 7 | accuracy | 23 | 16 | 12 | 26 | 19 | 11 | 4 |
| | sensitivity | 20 | 32 | 35 | 22 | 26 | 28 | 10 |



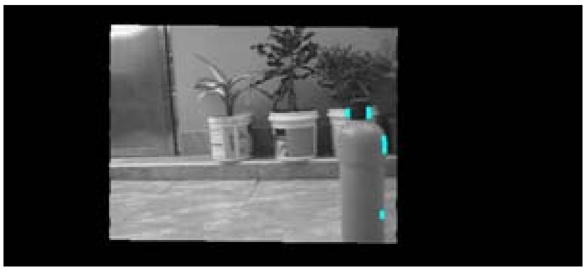
(a) 2R2C



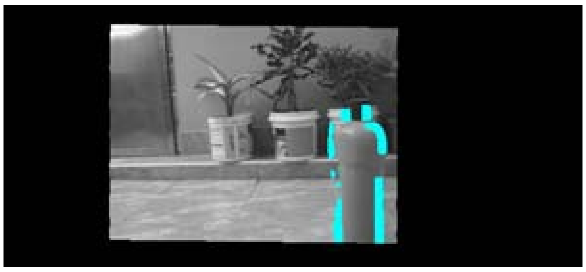
(b) 2R3C



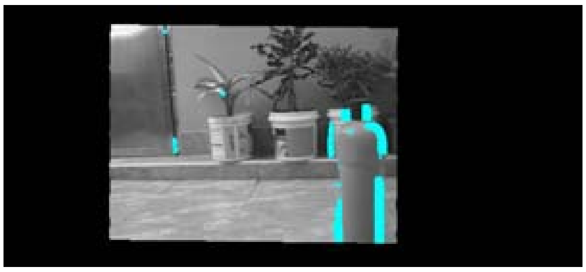
(c) 2R4C



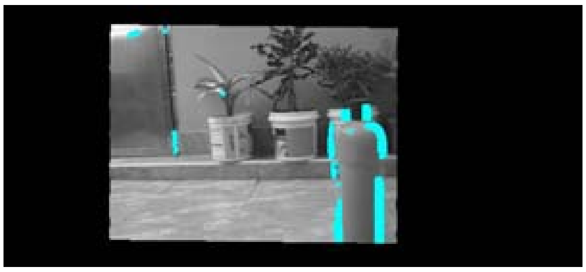
(d) 3R2C



(e) 3R3C

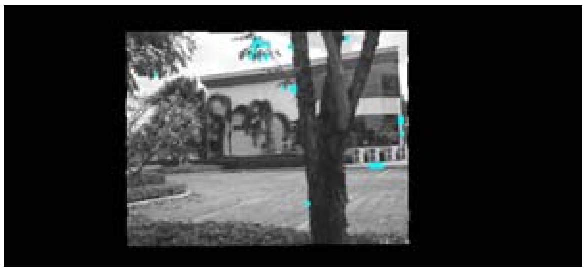


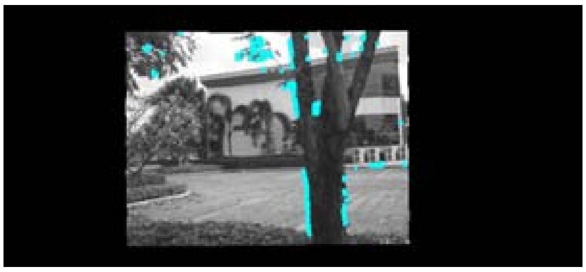
(f) 3R4C

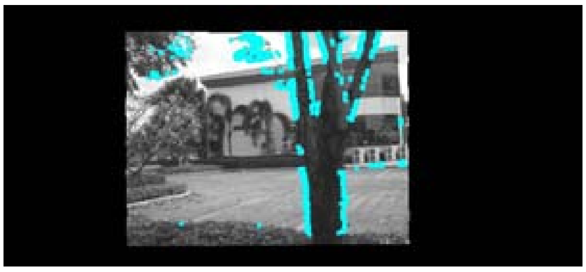


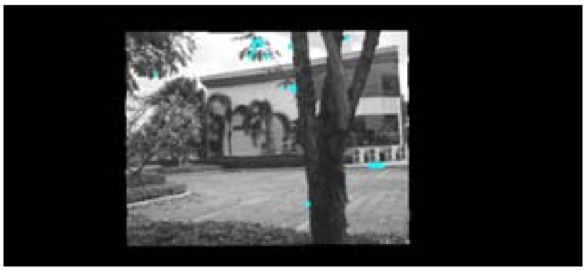
(g) 4R4C

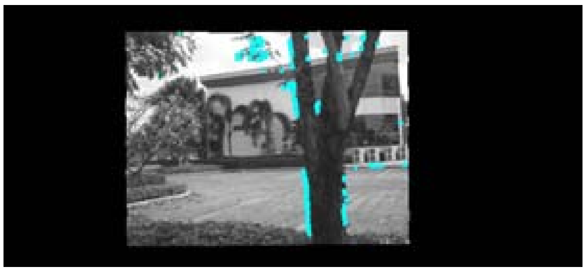
Figure 4.14: The extracted edge of the 120th frame of the third video sequence when the number of the edge group and the number of frame per group are different. The voting window size and $T_2$ are $3 \times 3$ and 3, respectively. The foreground edge is shown in cyan.

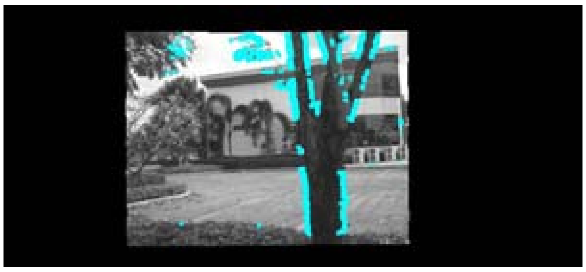

(a) 2R2C


(b) 2R3C


(c) 2R4C


(d) 3R2C


(e) 3R3C


(f) 3R4C

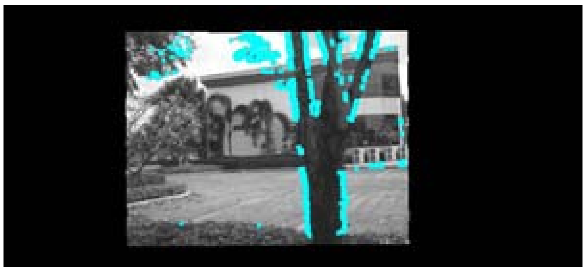

(g) 4R4C

Figure 4.15: The extracted edge of the 175th frame of the tenth video sequence when the number of the edge group and the number of frame per group are different. The voting window size and $T_2$ are $3\times3$ and 3, respectively. The foreground edge is shown in cyan.

## 4.3 Effect of tracking

Figures 4.16 - 4.18 show the edge extracted before (left column) and after (right column) edge refining by tracking. Most of the spurious background was removed. Figures 4.19 – 4.31 show the foreground rejection before (left column) and after (right column) area refining. The remaining spurious edges led to the incorrect rendering where the large area of background was considered as foreground. Holes in the foreground object could be ignored if it was small. By comparing the right column to the left column, it was found that less background was deleted and less foreground area was misdetected as background.

The better foreground rejection would lead to better registration and more information for merging. This experiment showed that the 2D translation model was sufficient and the refining by tracking should be integrated to the foreground rejection.

## 4.4 Effect of foreground rejection to the registration accuracy

Figures 4.22 - 4.24 show the registration results with and without foreground rejection. The first, the reference and the last frame are displayed in different color channel. When an area contains the information from all three frames, the misregistration is shown in color.

Figures 4.22 and 4.23 showed that the effect of the foreground rejection was almost negligible when the foreground object was small. Figure 4.24 indicated that when the parallax object was large, the foreground rejection led to better registration accuracy. The 'A' letter was incorrectly registered and shown as the cyan shadow in Figure 4.24 (f). Less misregistered area was achieved. Furthermore, the inclusion of refining by tracking led to better registration and more crisp edges were obtained.

## 4.5 Effect of foreground rejection to stitching

Figures 4.25 – 4.37 show the stitched images from 13 video sequences. In all figures, (a) and (b) show the stitched images from the video stitching algorithm without and with foreground rejection; (c) and (d) shows the enlarged area inside the dashed window in (a) and (b), respectively.

The black areas in (b) and (d) of Figures 4.25 – 4.37 are the areas without the background information. The differences between the stitched image with and without foreground rejection are as follows.

1) The stitched image with foreground rejection had sharper and more correct edge (Figures 4.25, 30, 31, 33, 34, 35 and 37).
2) The stitched image with foreground rejection had less ghosting effect (Figures 4.26 - 4.29, and 4.36).

There was the loss of large background area in the stitched result with foreground rejection of the 8th sequence (Figure 4.32). The rendering stage removed too much background due to the spurious background pixels. Nevertheless, it could be seen that the tape in the top right corner in Figure 4.32 (d) had the correct intensity; whereas, it was wrong in the stitched result without foreground rejection (Figure 4.32 (c)).

The results indicated that the proposed foreground rejection method improved the visual quality of video stitching as less visible ghosting and crisper stitched image. The intensity of the background was more similar to the actual one.

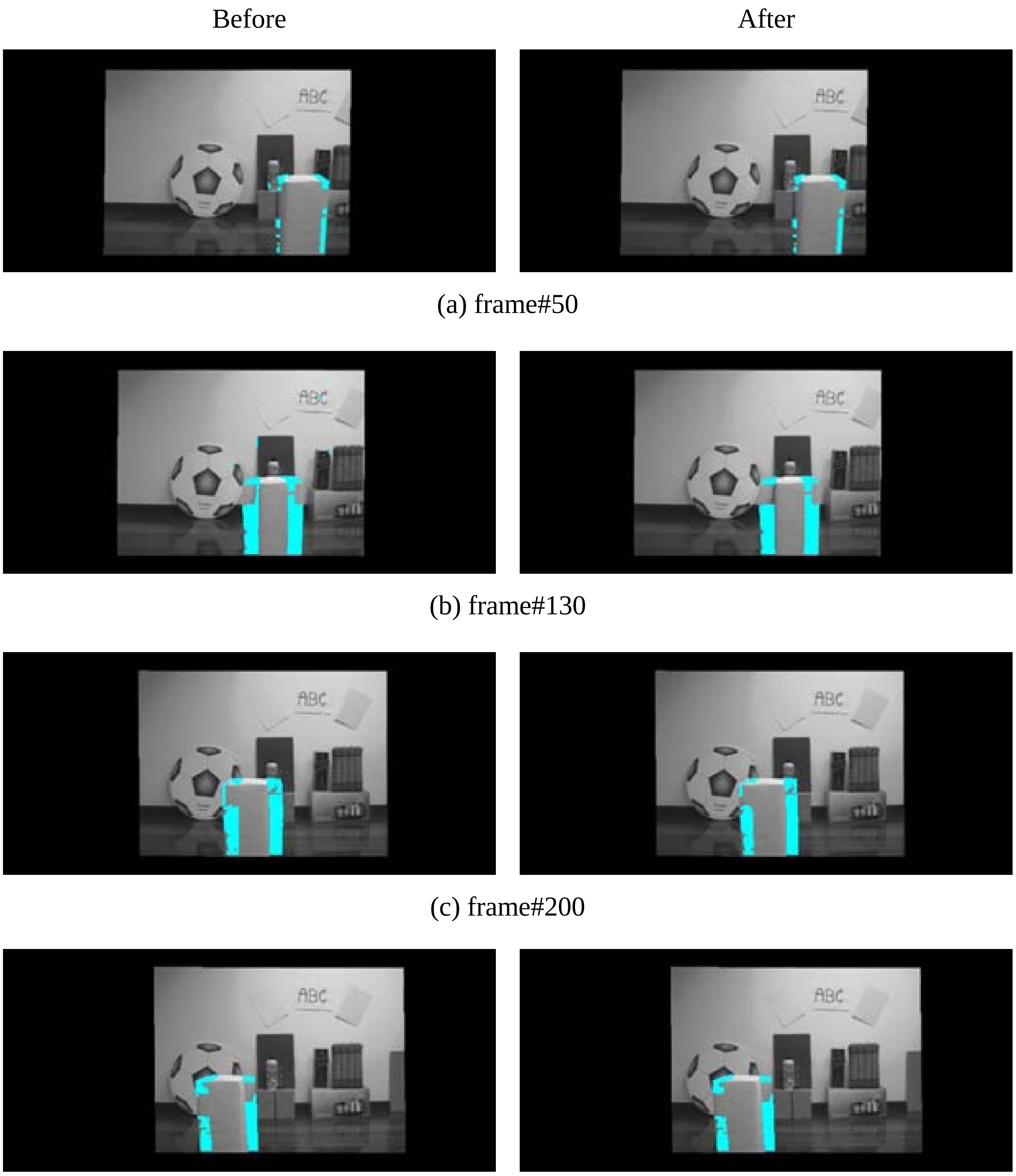
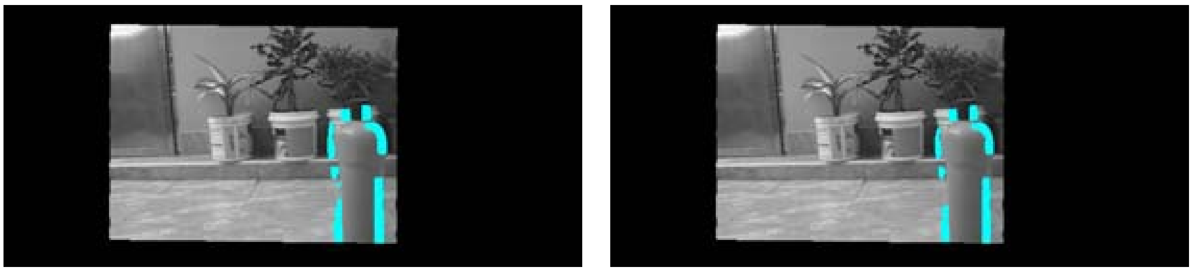
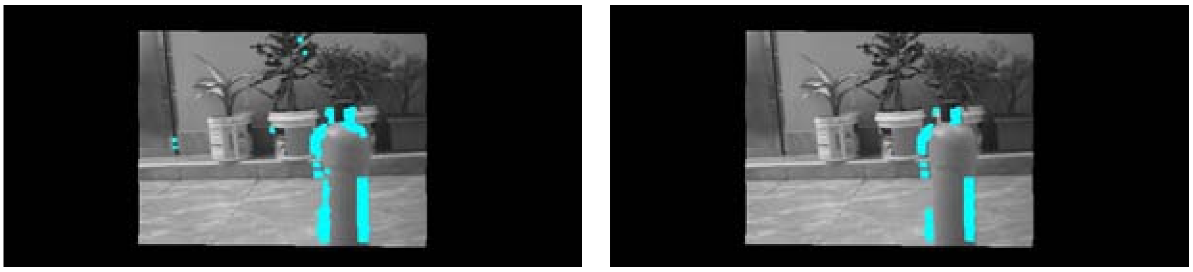


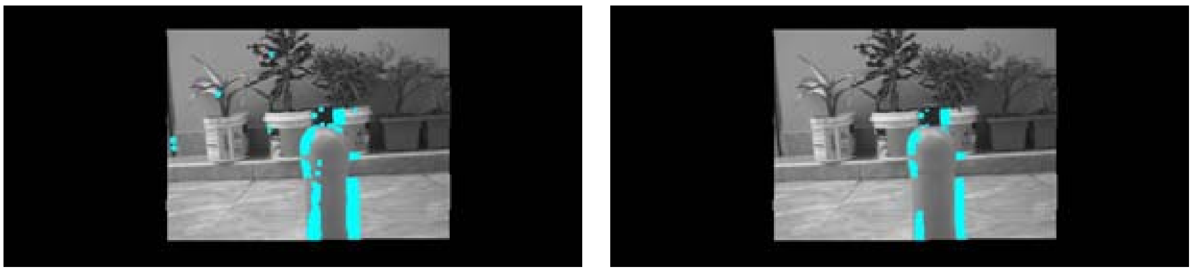
Figure 4.16: The extracted edge before (left column) and after (right column) the edge refining for the 2nd sequence. The foreground edge is shown in cyan.



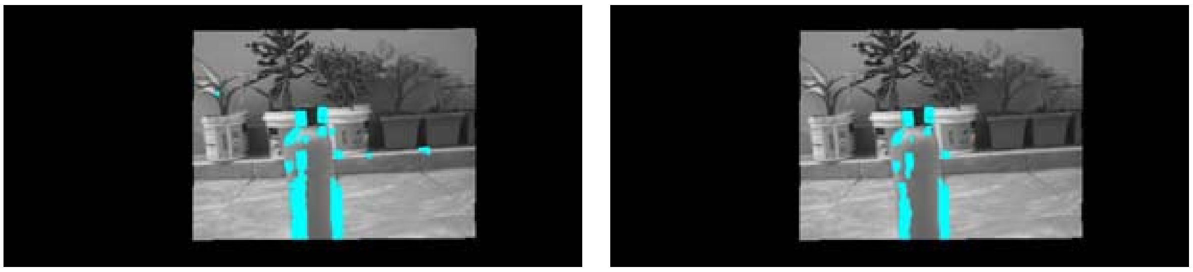
(a) frame#120



(b) frame#200



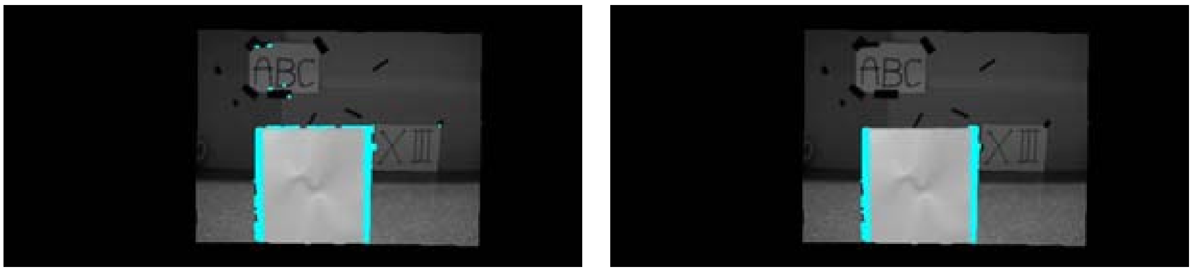
(c) frame#270

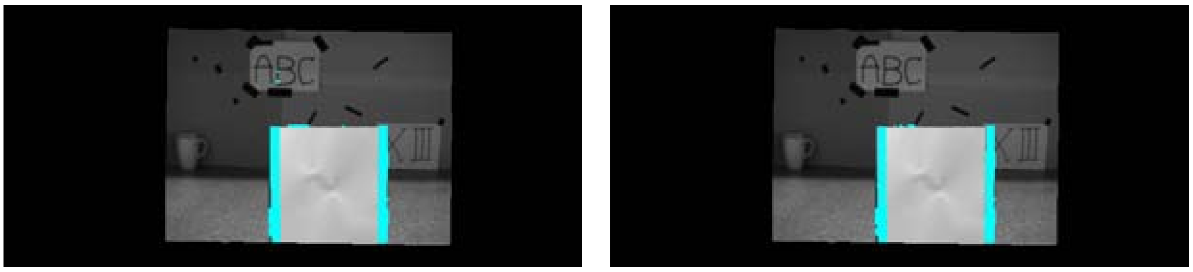


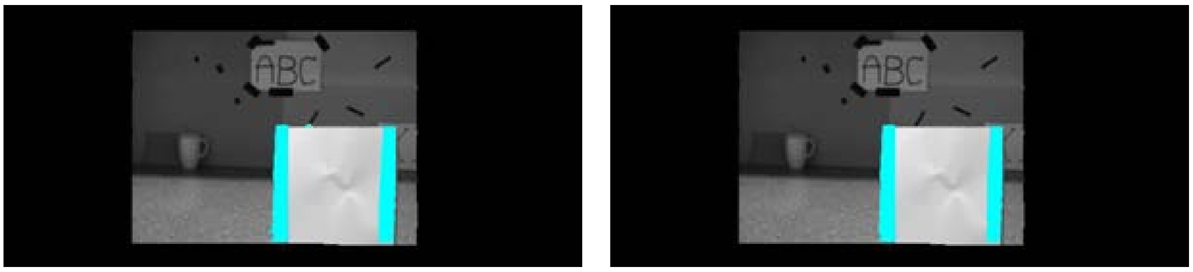
(d) frame#320

Figure 4.17: The extracted edge before (left column) and after (right column) the edge refining for the 3rd sequence. The foreground edge is shown in cyan.

Before After


(a) frame#10


(b) frame#90


(c) frame#160

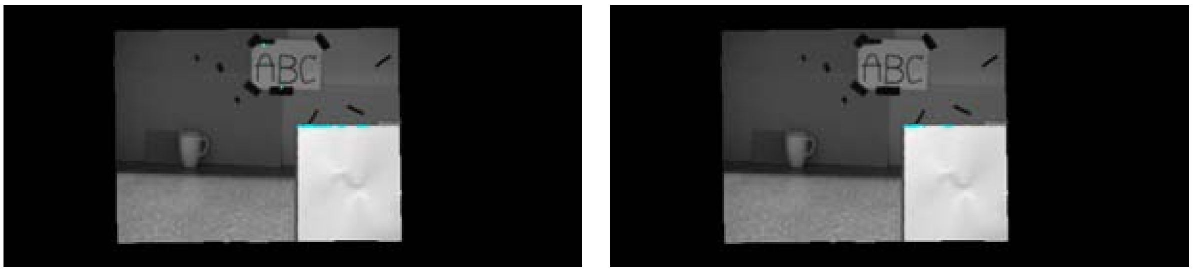
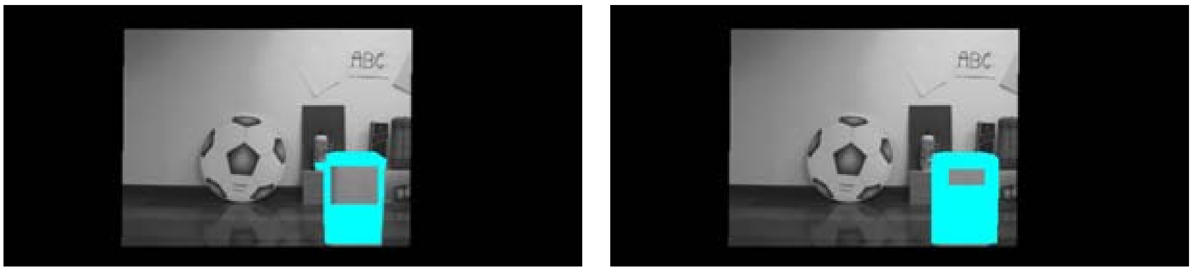
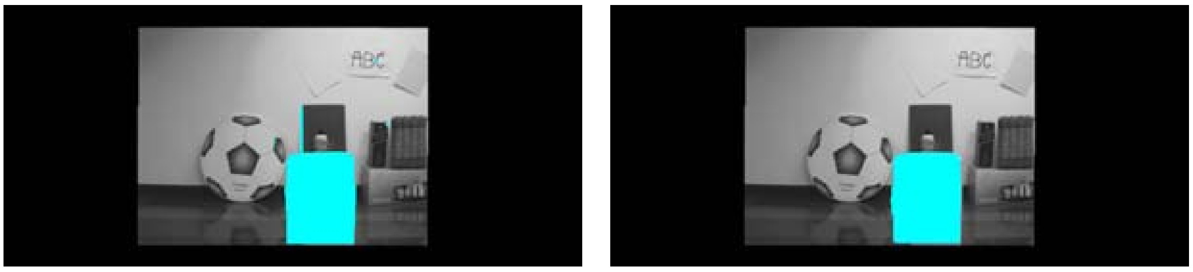
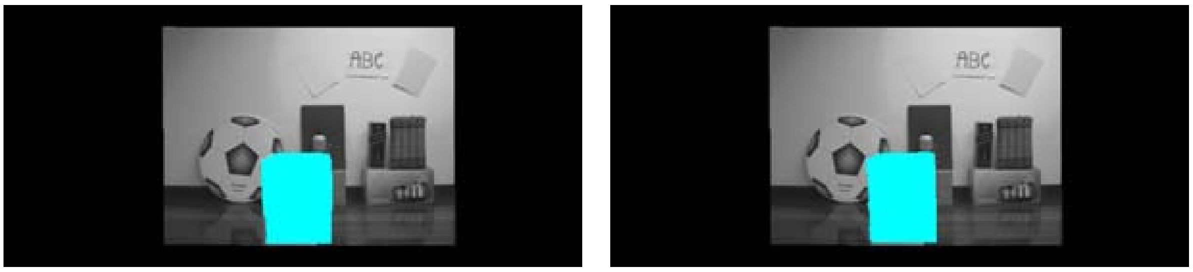

(d) frame#210

Figure 4.18: The extracted edge before (left column) and after (right column) the edge refining for the 9th sequence. The foreground edge is shown in cyan.

Before After


(a) frame#50


(b) frame#130


(c) frame#200

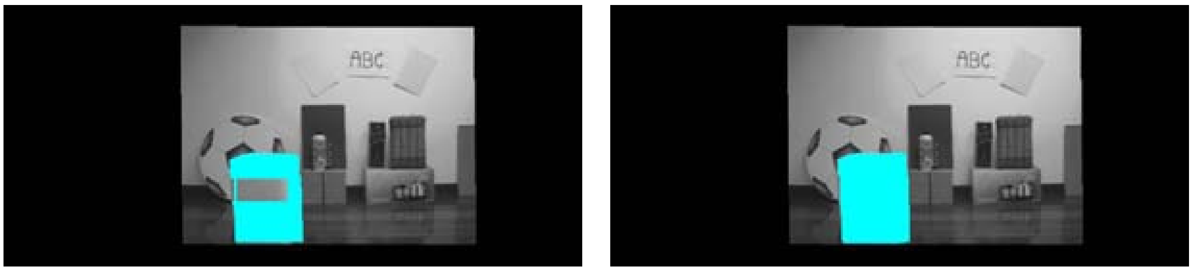
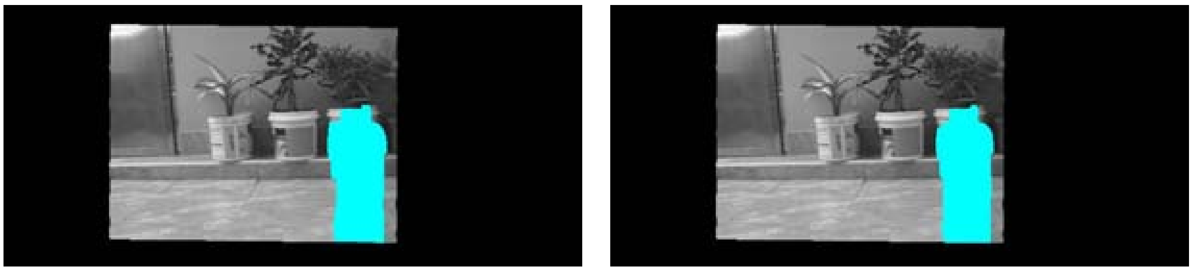
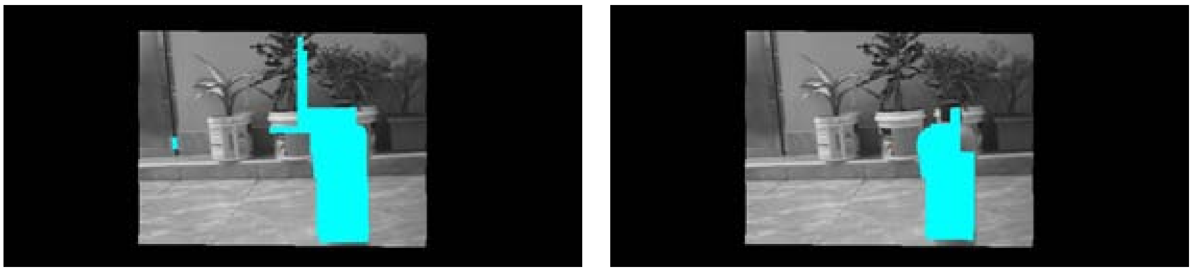

(d) frame#250

Figure 4.19: The result of foreground removal before (left column) and after (right column) the area refining for the 2nd sequence. The foreground area is shown in cyan.

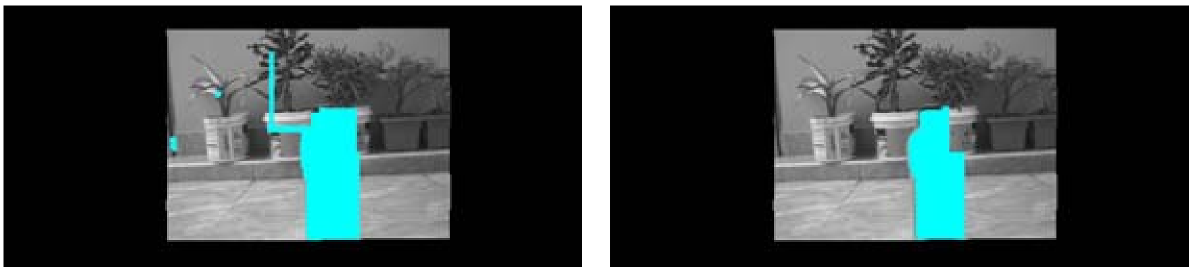
Before After



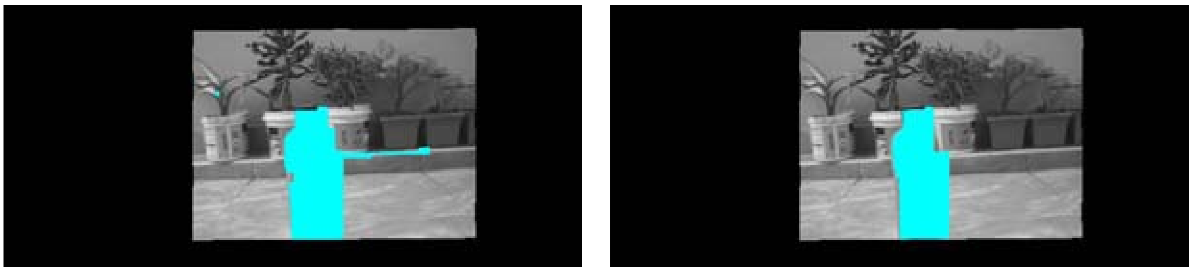
(a) frame#120



(b) frame#200



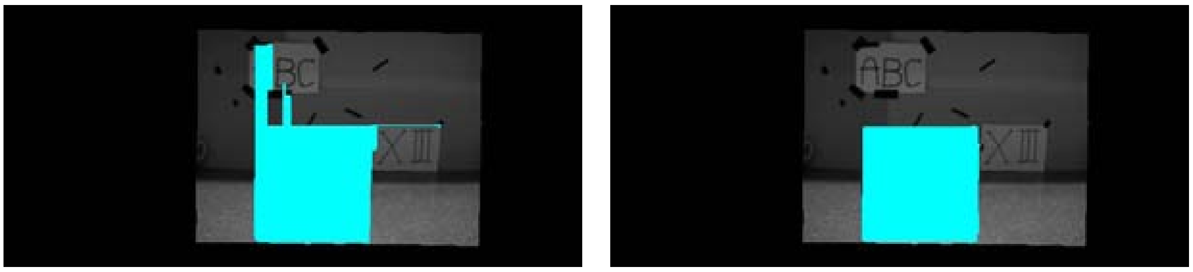
(c) frame#270



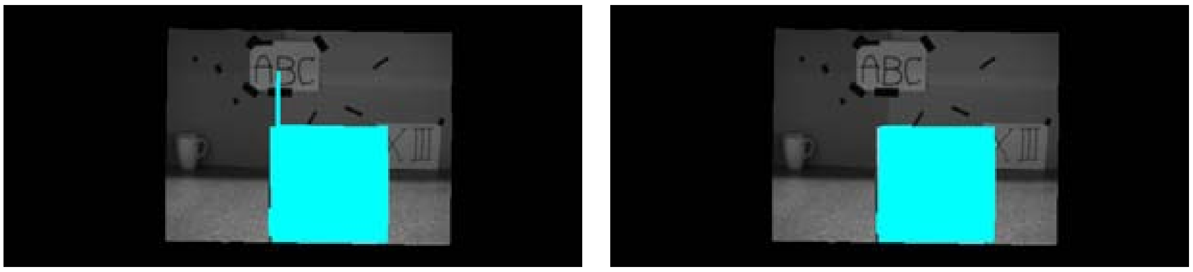
(d) frame#320

Figure 4.20: The result of foreground removal before (left column) and after (right column) the area refining for the 3rd sequence. The foreground area is shown in cyan.

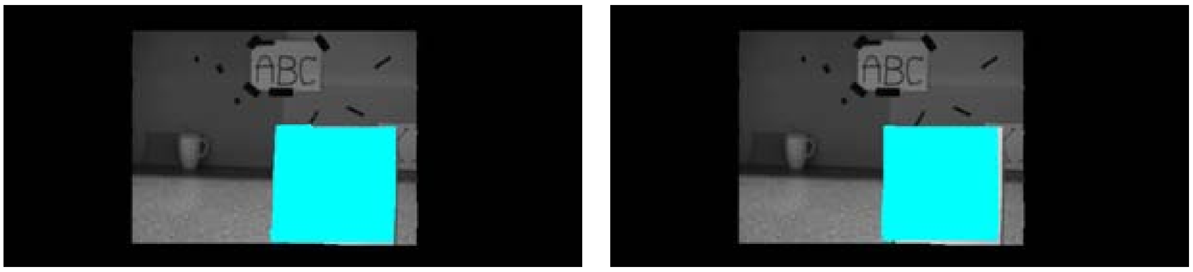
Before After



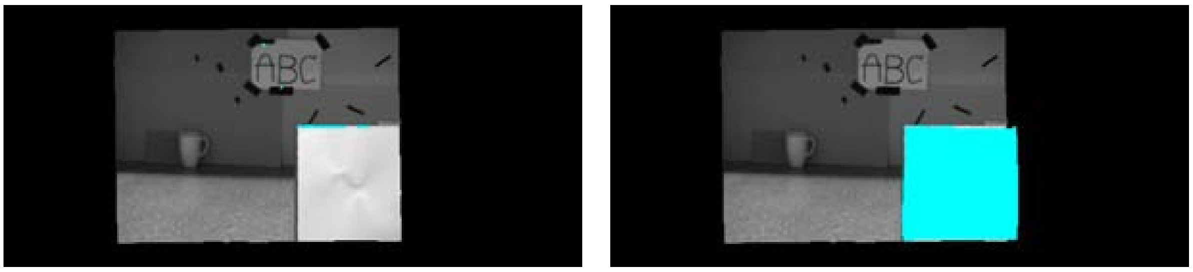
(a) frame#10



(b) frame#90

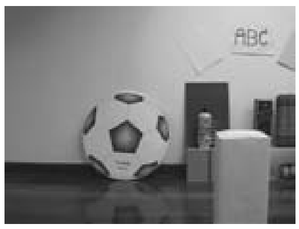


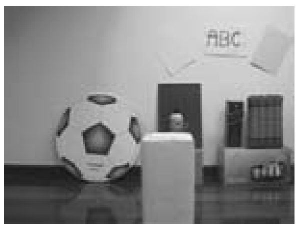
(c) frame#160

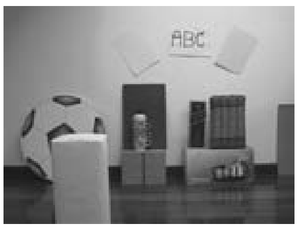


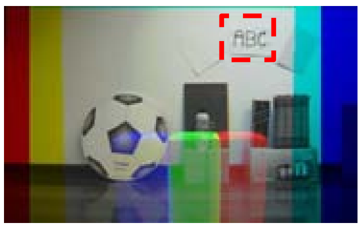
(d) frame#210

Figure 4.21: The result of foreground removal before (left column) and after (right column) the area refining for the 9th sequence. The foreground area is shown in cyan.

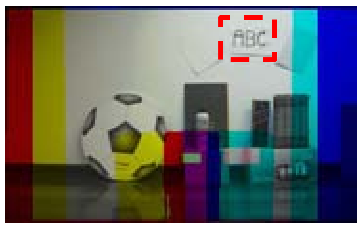

(a) original first frame


(b) original reference frame

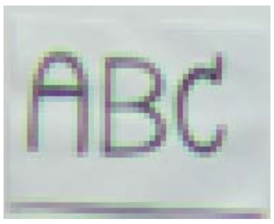

(c) original last frame



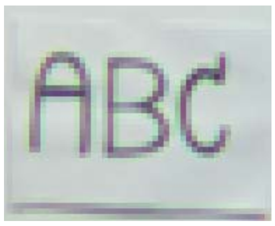
(d) without foreground rejection



(e) with foreground rejection



(f) enlarged image inside the dashed window in (d).



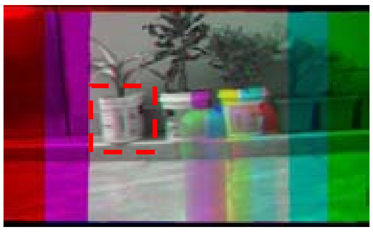
(g) enlarged image inside the dashed window in (e).

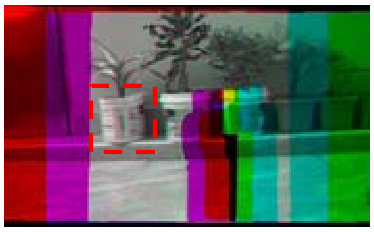
Figure 4.22: Composite color image showing the registration of the first (R-channel) and the last frame (B-channel) to the reference frame (G-channel) in the 2nd sequence.

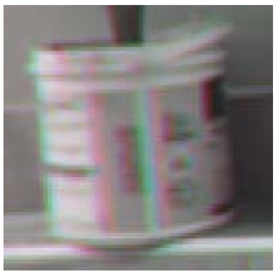
(a) original first frame

(b) original reference frame

(c) original last frame


(d) without foreground rejection


(e) with foreground rejection


(f) enlarged image inside the dashed window in (d).

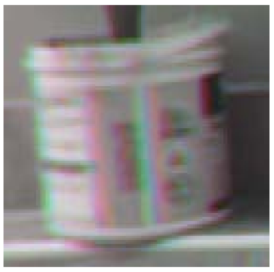
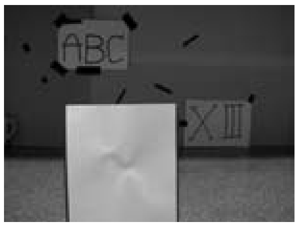
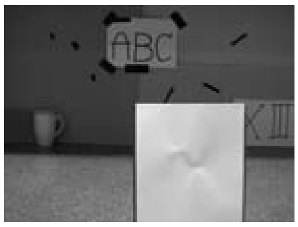
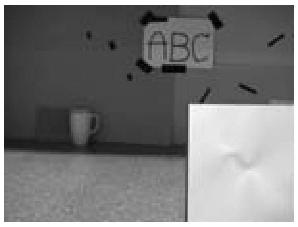
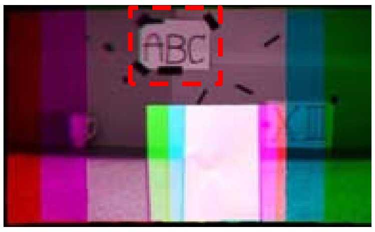

(g) enlarged image inside the dashed window in (e).

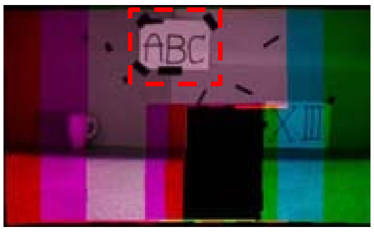
Figure 4.23: Composite color image showing the registration of the first (R-channel) and the last frame (G-channel) to the reference frame (B-channel) in the 3rd sequence.

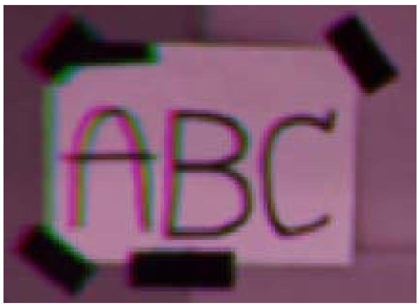

(a) original first frame


(b) original reference frame


(c) original last frame


(d) without foreground rejection


(e) with foreground rejection


(f) enlarged image inside the dashed window in (d).

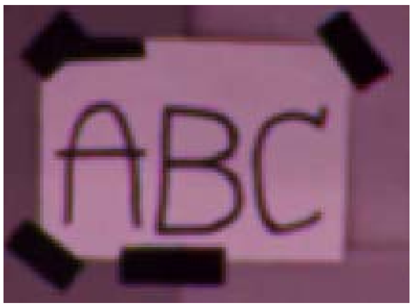
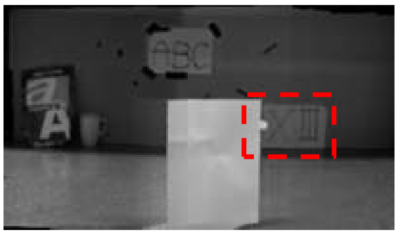
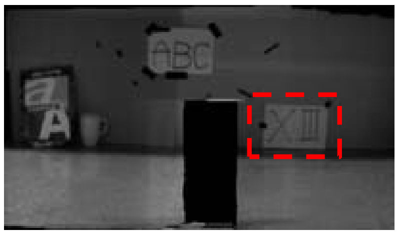

(g) enlarged image inside the dashed window in (e).

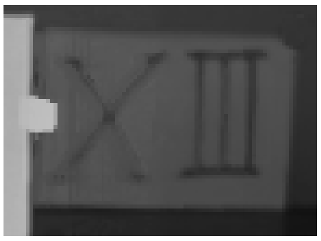
Figure 4.24: Composite color image showing the registration of the last (R-channel) and the first frame (G-channel) to the reference frame (B-channel) in the 9th sequence.



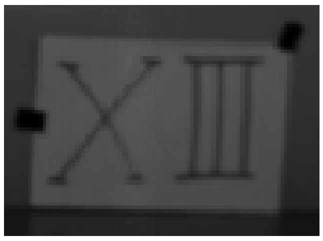
(a) without foreground rejection



(b) with foreground rejection

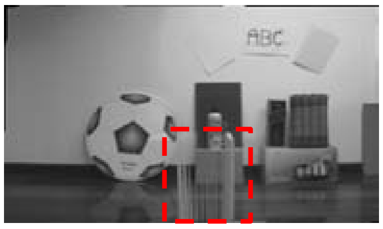


(c) enlarged image inside the dashed window in (a).

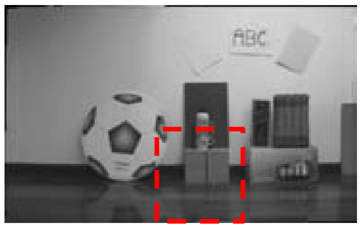


(d) enlarged image inside the dashed window in (b).

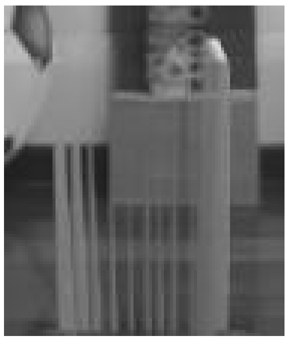
Figure 4.25: Stitched image from the first video sequence.



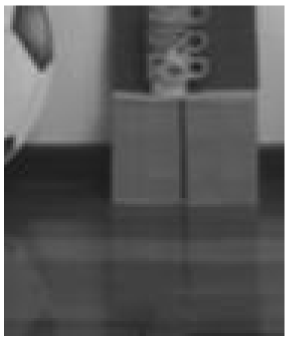
(a) without foreground rejection



(b) with foreground rejection

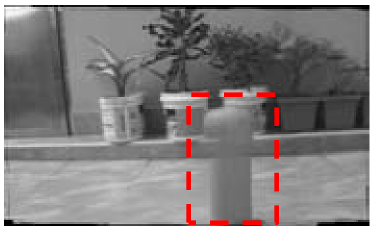


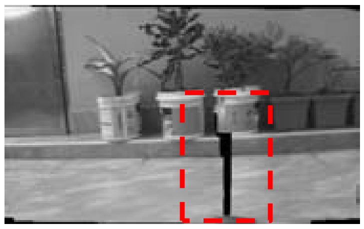
(c) enlarged image inside the dashed window in (a).

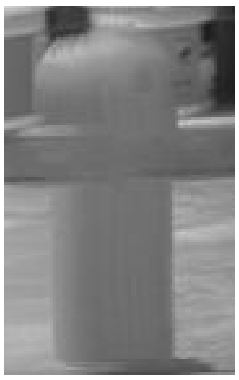


(d) enlarged image inside the dashed window in (b).

Figure 4.26: Stitched image from the second video sequence.


(a) without foreground rejection


(b) with foreground rejection


(c) enlarged image inside the dashed window in (a).

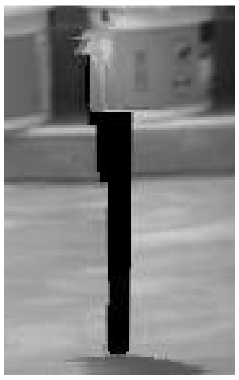
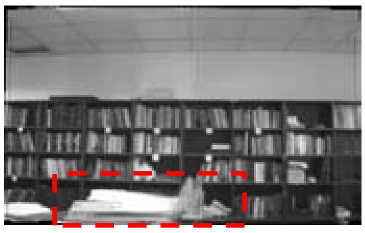
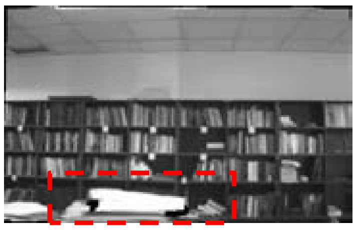
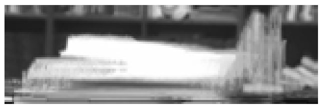

(d) enlarged image inside the dashed window in (b).

Figure 4.27: Stitched image from the third video sequence.


(a) without foreground rejection


(b) with foreground rejection


(c) enlarged image inside the dashed window in (a).

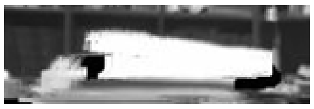
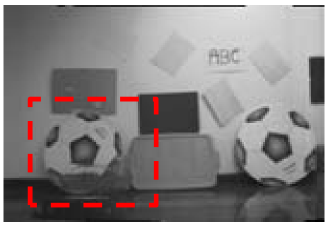
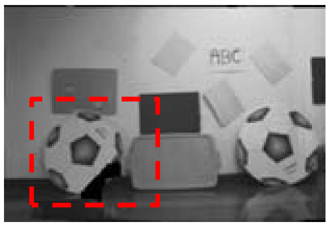
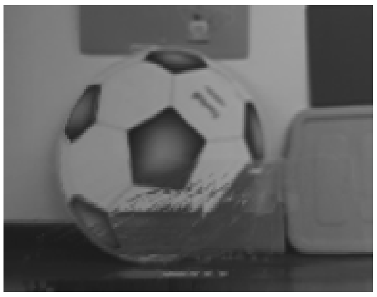

(d) enlarged image inside the dashed window in (b).

Figure 4.28: Stitched image from the fourth video sequence.


(a) without foreground rejection


(b) with foreground rejection


(c) enlarged image inside the dashed window in (a).

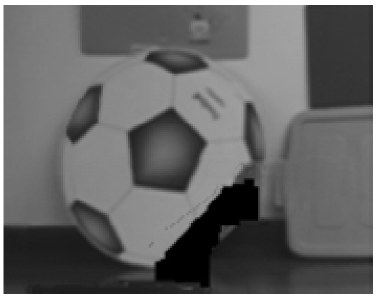
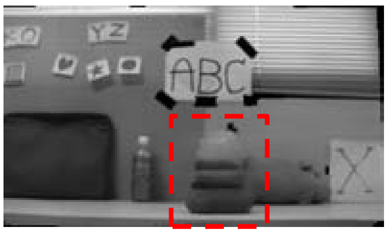
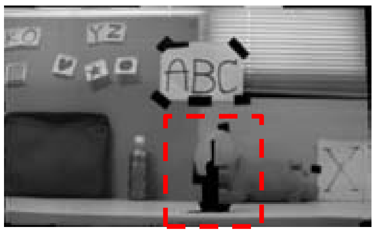
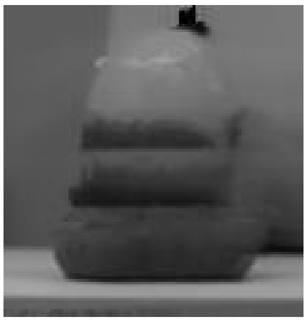

(d) enlarged image inside the dashed window in (b).

Figure 4.29: Stitched image from the fifth video sequence.


(a) without foreground rejection


(b) with foreground rejection


(c) enlarged image inside the dashed window in (a).

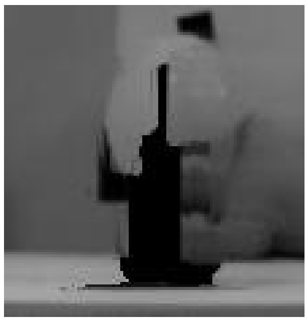
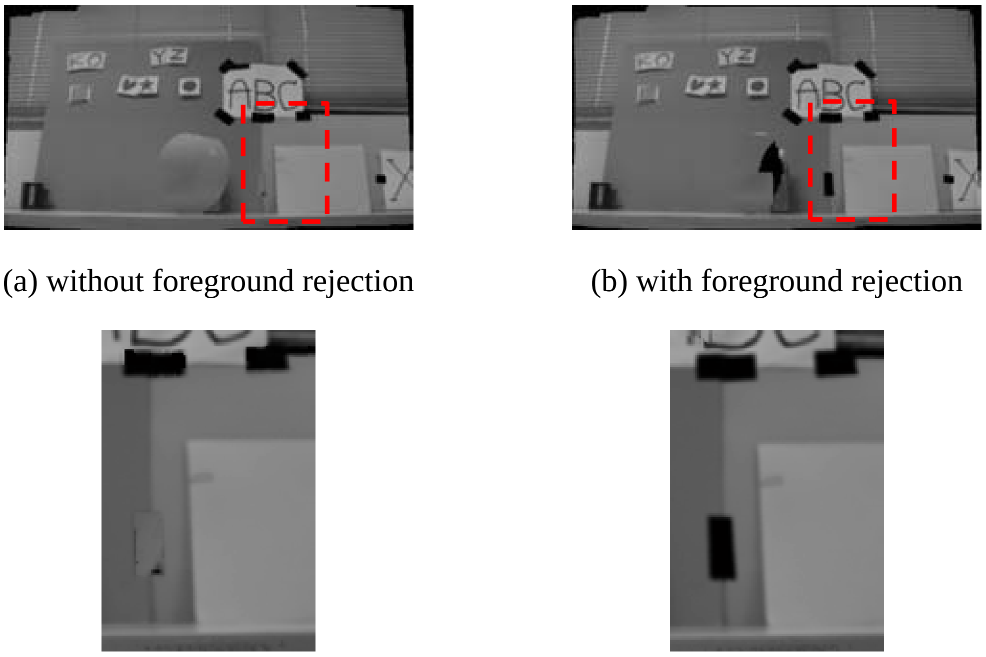

(d) enlarged image inside the dashed window in (b).

Figure 4.30: Stitched image from the sixth video sequence.



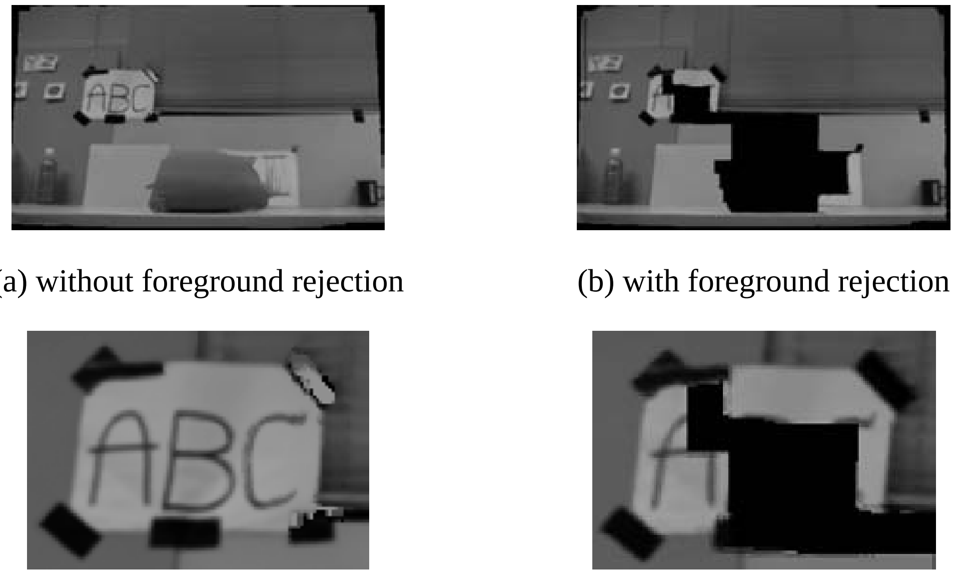
(a) without foreground rejection

(b) with foreground rejection

(c) enlarged image inside the dashed window in (a).

(d) enlarged image inside the dashed window in (b).

Figure 4.31: Stitched image from the seventh video sequence.



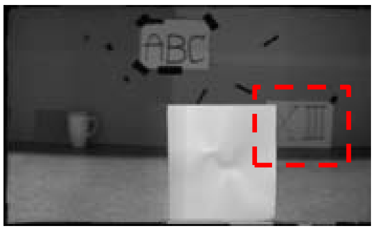
(a) without foreground rejection

(b) with foreground rejection

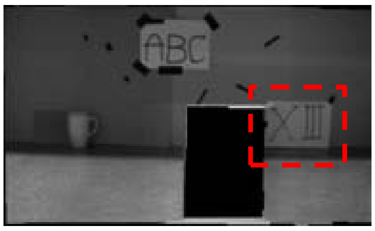
(c) enlarged image inside the dashed window in (a).

(d) enlarged image inside the dashed window in (b).

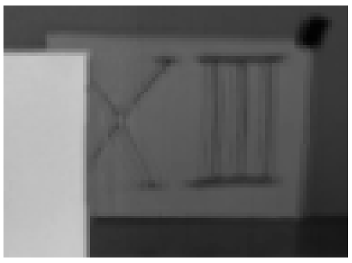
Figure 4.32: Stitched image from the eighth video sequence.



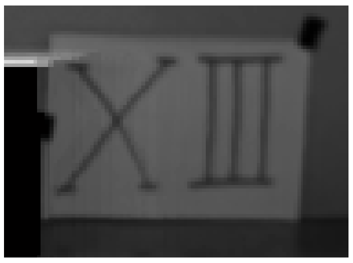
(a) without foreground rejection



(b) with foreground rejection

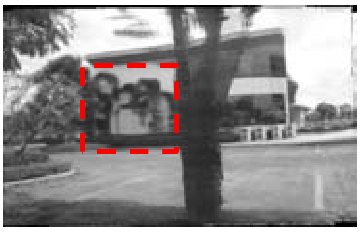


(c) enlarged image inside the dashed window in (a).

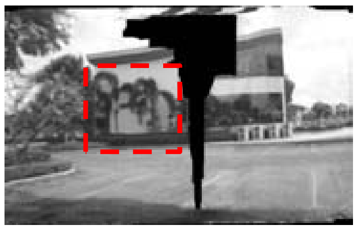


(d) enlarged image inside the dashed window in (b).

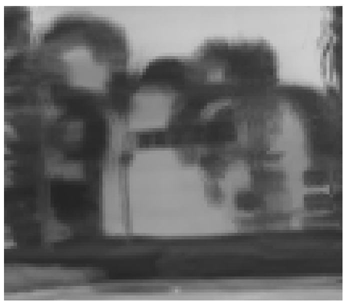
Figure 4.33: Stitched image from the ninth video sequence.



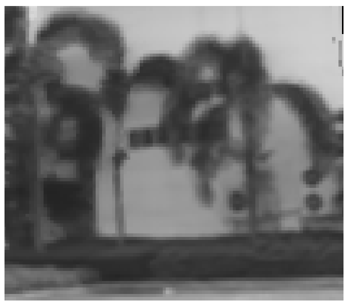
(a) without foreground rejection



(b) with foreground rejection

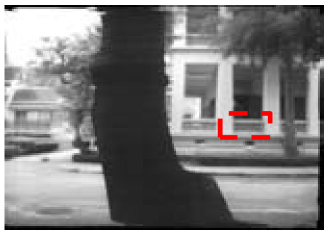


(c) enlarged image inside the dashed window in (a).

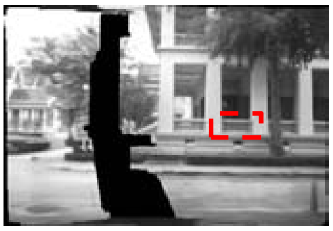


(d) enlarged image inside the dashed window in (b).

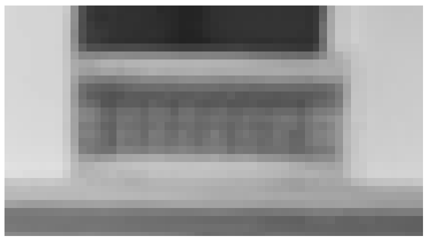
Figure 4.34: Stitched image from the tenth video sequence.



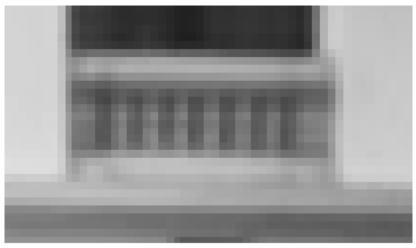
(a) without foreground rejection



(b) with foreground rejection

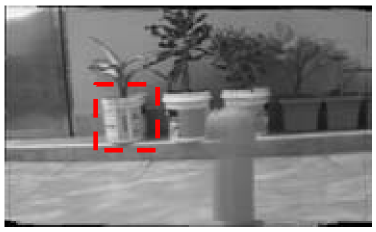


(c) enlarged image inside the dashed window in (a).

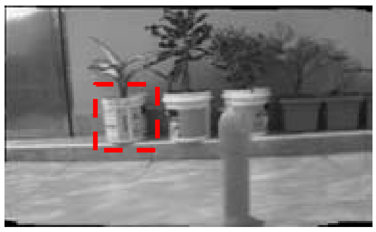


(d) enlarged image inside the dashed window in (b).

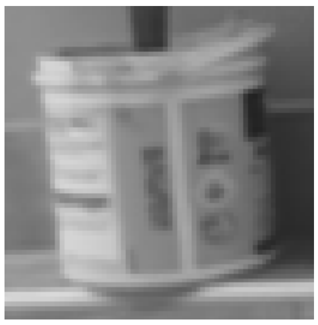
Figure 4.35: Stitched image from the eleventh video sequence.



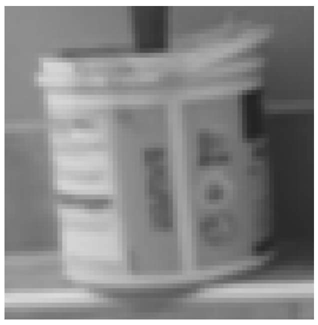
(a) without foreground rejection



(b) with foreground rejection

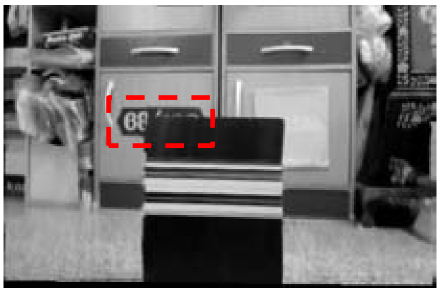


(c) enlarged image inside the dashed window in (a).

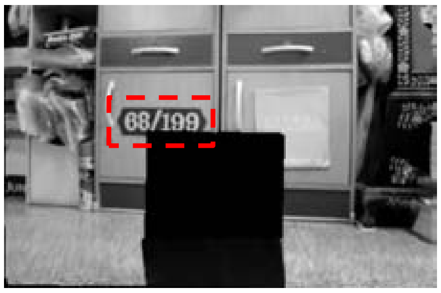


(d) enlarged image inside the dashed window in (b).

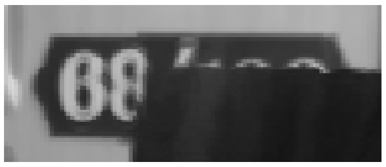
Figure 4.36: Stitched image from the twelfth video sequence.



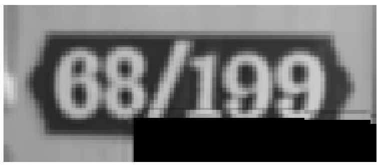
(a) without foreground rejection



(b) with foreground rejection



(c) enlarged image inside the dashed window in (a).



(d) enlarged image inside the dashed window in (b).

Figure 4.37: Stitched image from the thirteenth video sequence.

# CHAPTER V

# CONCLUSIONS

## 5.1 Conclusions

Parallax objects lead to inaccurate registration and ghosting effect of the foreground objects in the stitched image, and should be removed. This thesis proposed the foreground rejection for parallax removal in video sequence stitching. The proposed method consists of (1) edge extraction and (2) rendering. The result in each stage is refined using the motion found from tracking algorithm.

In the edge extraction stage, the logical OR-operator and voting are used to classify foreground edge pixels. Tracking is applied to find the motion of the foreground. The motion is then used to remove the spurious background pixels. After that two-stage rendering is applied to extract the entire foreground. Finally, the motion from the tracking algorithm is used to refine the foreground area.

From the experiments in Chapter IV, the proposed foreground rejection method improves the video stitching in the following aspects:

1) more accurate registration,
2) the reduction of the ghosting effect,
3) sharper stitched result,
4) more background information in the result.

## 5.2 Future works

The proposed method is designed within the scope of one parallax object in the fixed background. However, in practice, it is very difficult to have only one object with the different depth in the fixed background. Thus, the following two topics should be included in the future development.

1) The detection of changing region (CR) in the time-varying background (e.g. leaves in a tree, wave in an ocean, etc.). The simple thresholding is used in the CR detection in Section 3.2.1. To allow some intensity changes in the background, adaptive thresholding should be considered.
2) The removal of multiple parallax objects. The current tracking assumption is that there is only one foreground motion in a video sequence. The algorithm to track multiple foregrounds should be used.

The rendering algorithm in the proposed method uses a fixed threshold to reject the background that is incorrectly detected as the foreground. The spurious background leads to the loss of the large background area. The mechanism to reject the spurious background should also be included.

In this thesis, the proposed method was the offline method, because it was implemented on MATLAB 2011a. The major processes inside the proposed method are the logical operator (OR, AND), the voting (condition and summation) and 2D-translation tracking and mapping. The first two processes have very low computing load, and the third process can be implemented in real time by using hardware programming. Consequently, it is possible to implement the proposed method for the real time application.

# REFERENCES

[1] Yamamoto, T. 3D movie camera design using commodity components for digital archiving of cultural heritages. In Proc. IEEE of Electronic Imaging & the Visual Arts (EVA) (April 2010).

[2] Linhong, Y., and Hirakawa, M. A stitching algorithm of still pictures with camera translation. In Proc. IEEE of the First International Symposium on Cyber Worlds (CW) 101, 701 (2002) : 1-6.

[3] Kwatra, V., Schödl, A., Essa, I., Turk, G., and Bobick, A. Graphcut textures: image and video synthesis using graph cuts. ACM Transactions on Graphics 22, 3 (2003) : 277-286.

[4] Zhao, X., Wang, H., and Wang, Y. Medical image seamlessly stitching by SIFT and GIST. In Proc. IEEE E-Product E-Service and E-Entertainment (ICEEE) (November 2010) : 1-4.

[5] Arora, C., Singla, S., Singh, S., and Gupta, A. Seam reconstruct: dynamic scene stitching with large exposure difference. In Proc. IEEE Int. Conf. on Applications of Digital Information and Web Technologies (ICADIWT) (August 2009) : 574-578.

[6] Zhang, Z. M-estimators [Online]. 1996. Available from : http://research.microsoft.com/en-us/um/people/zhang/INRIA/Publis/Tutorial-Estim/node24.html [2011, August 26]

[7] Nixon, M.S., and Aguado, A.S. Feature extraction & image processing. In M.A. Fischler and O. Firschein (eds.), Computer Vision: Issues, Problem, Principle, and Paradigms, pp. 726-740. CA, USA : Morgan Kaufmann, 1987.

[8] Lowe, D. Distinctive image features from scale-invariant keypoints. Int. J. of Computer Vision 60, 2 (2004) : 91-110.

[9] Donporntun, V. Keypoints detection with Dop-RPPRBF for SIFT feature descriptor on Low Complexity Processor. In Nat. Conf. on Computer Information Technologies (January 2010) : 130-135.

[10] Pratchett, T. RANSAC [Online]. 2011. Available from : http://en.wikipedia.org/wiki/RANSAC [2011, September 1]

[11] Szeliski, R. Image alignment and stitching. In N. Paragios, and others (eds.), Mathematical Models in Computer Vision, pp. 273-307. NY, USA : Springer, 2006.

[12] Piccardi, M. Background subtraction techniques: A review. In Proc. IEEE Int. Conf. on Systems, Man and Cybernetics (October 2004) : 3099–3104.

[13] Weis, M. Search methods in motion estimation [Online]. 2010. Available from : http://blog.weisu.org/2008/12/search-methods-in-motion-estimation.html [2012, March 20]

[14] Agarwala, A., and others. Interactive digital photomontage. In Proc. ACM Transactions on Graphics 23, 3 (August 2004) : 292-300.

[15] Boykov, Y., Veksler, O., and Zabih, R. Fast approximate energy minimization via graph cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (November 2001) : 1222-1239.

[16] Burt, P., and Adelson, E. A multiresolution spline with applications to image mosaics. ACM Transactions on Graphics 2, 4 (October 1983) : 217-236.

[17] Levin, A., Zomet, A., Peleg, S., and Weiss, Y. Seamless image stitching in the gradient domain. In Proc. European Conference on Computer Vision (ECCV) (May 2004) : 377–389.

[18] Milgram, D. Computer methods for creating photomosaics. IEEE Transactions on Computers C-24 (November 1975) : 1113-1119.

[19] Davis, J. Mosaics of scenes with moving objects. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (June 1998) : 354-360.

[20] Irani, M., and Anandan, P. Video indexing based on mosaic representations. In Proc. IEEE 86, 5 (May 1998) : 905-921.

[21] Wren, C., Azarhayejani, A., Darrell, T., and Pentland, A.P. Pfinder: real-time tracking of the human body. IEEE Trans. on Patfern Anal. and Machine Infell. 19, 7 (1997) : 780-785.

[22] Koller, D., and others. Towards Robust Automatic Traffic Scene Analysis in Real-time. In Proc. ICPR 94 (November 1994) : 126-131.

[23] Lo, B.P.L., and Velastin, S.A. Automatic congestion detection system for underground platforms. In Proc. ISIMP 2001 (May 2001) : 158-161.

[24] Cucchiara, R., Grana, C., Piccardi, M., and Prati, A. Detecting moving objects, ghosts, and shadows in video streams. IEEE Trans on Paftem Anal. and Machine Infell. 25, 10 (2003) : 1337-1442.

[25] Stauffer, C. and Grimson, W.E.L. Adaptive background mixture models for real-time tracking. In Proc. IEEE CVPR 1999 (June 1999) : 246-252.

[26] Elgammal, A., Hanvood, D., and Davis, L.S. Nonparametric model for background subtraction. In Proc. ECCV 2000 (June 2000) : 751-767.

[27] Comaniciu, D. and Meer, P. Mean shift a robust approach toward feature space analysis. IEEE Trans. on Paftem Anal. and Machine Infell. 24, 5 (2002) : 603-619.

[28] Comaniciu, D. An algorithm for data-driven bandwidth selection. IEEE Trans. on Paftem Anal. and Machine Infell. 25, 2 (2003) : 281-288.

[29] Piccardi, M., and Jan, T. Efficient mean-shift background subtraction. In Proc. of IEEE ICIP 2004 (October 2004).

[30] Han, B., Comaniciu, D., and Davis, L.S. Sequential kernel density approximation through mode propagation: applications to background modeling. In Proc. Asian Conf. on Computer Vision (January 2004).

[31] Seki, M., Wada, T., Fujiwara, H., and Sumi, K. Background subtraction based on cooccurrence of image variations. In Proc. CVPR 2003 2 (2003) : 65-72.

[32] Oliver, N.M., Rosario, B., and Pentland, A.P. A Bayesian computer vision system for modeling human interactions. IEEE Trans. on Paftem Anal. and Machine Infell. 22, 8 (2000) : 831-843.

[33] Patanaviji, V. and Jitapunkul, S. Fast affine block-based image registration using a novel three-step search algorithm for super-resolution image reconstruction. In Proc. of Int. Workshop on Advanced Image Technology (January 2006).

[34] Smith, V. Accuracy and precision [Online]. 2012. Available from : http://en.wikipedia.org/wiki/Accuracy_and_precision [2012, April 26]

[35] Victors, J. Sensitivity and specificity [Online]. 2012. Available from : http://en.wikipedia.org/wiki/Sensitivity_and_specificity [2012, April 26]

# BIOGRAPHY

Thanissorn Panarungsun was born in Bangkok, Thailand, in 1989. She received her Bachelor’s Degree in Electrical Engineering from Chulalongkorn University, Thailand, in 2010. She was granted “Sitkonkuti” Scholarship from Electrical Engineering Department to pursue her Master’s degree in electrical engineering at Chulalongkorn University, Thailand, since 2011. Her research interest is Multimedia and Signal Processing.