國 立 交 通 大 學

資訊科學與工程研究所

碩 士 論 文

利用架設在視訊監控車上成對的

全方位攝影裝置作週遭環境監控之研究

A Study on Surrounding Environment Monitoring by a Video

Surveillance Car with Two 2-camera Omni-imaging Devices

研 究 生：陳俊甫

指導教授：蔡文祥　教授

中 華 民 國　一 ＯＯ年 六 月

利用架設在視訊監控車上成對的
全方位攝影裝置作週遭環境監控之研究

A Study on Surrounding Environment Monitoring by a Video Surveillance Car with Two 2-camera Omni-imaging Devices

研　究　生：陳俊甫　　　　Student：Chun-Fu Chen

指導教授：蔡文祥　　　　Advisor：Wen-Hsiang Tsai

國 立 交 通 大 學
多 媒 體 工 程 研 究 所
碩 士 論 文

A Thesis
Submitted to Institute of Multimedia Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer Science

June 2011

Hsinchu, Taiwan, Republic of China

中華民國一 OO 年六月

利用架設在視訊監控車上成對的全方位攝影裝置作週遭環境監控之研究

研究生: 陳俊甫　　指導教授:蔡文祥 博士

國立交通大學資訊科學與工程研究所

摘要

本研究利用架設在一視訊監控車頂上的兩組全方位攝影裝置來達到視訊監控的功能，主要應用於監控行車視角的盲點和周遭的車輛。

在本研究中，此二全方位攝影機裝置可用以監看車輛周遭任何角度的影像畫面。此外，本研究利用光流分析法直接套用在連續擷取的影像上，並利用影像的移動向量分析目前車輛的行走方向，產生對應方位的透視影像(perspective-view image)，方便駕駛觀看。另一方面，本研究亦提出一種「透視對應表」(perspective mapping table)，可以快速地將全方位影像轉成透視影像，提供駕駛觀看行車紀錄。

同時，本研究利用全方位攝影系統所拍攝的影像，可靜態監控周遭靜止的車輛並求得立體資訊。另利用影像處理技術取出影像中的車體區域，並偵測車窗底緣的對應點，計算車輛位置，進而產生監控車週遭環境的俯視圖。

除了偵測靜態的車輛，本研究也提出了行駛當中的視訊監控車偵測到停止或移動的周遭車輛的方法。另亦使用光流分析法，配合全方位攝影機所擷取到的連續影像，利用有高度的物體會產生較大移動向量的性質，將車體給大致分割出來，進而使用「k 均值分群法」(k-means clustering)去偵測出車體，接著透過區域增長法去找出較完整的車體，最後再利用預先造好的車輛模型去做比對，藉以取得周遭車輛的位置資訊，劃出車輛周遭的俯視圖，供車輛駕駛觀看。

上述方法的實驗結果皆甚良好，顯示所提視訊監控系統確實可行。

A Study on Surrounding Environment Monitoring by a Video Surveillance Car with Two 2-camera Omni-imaging Devices

Student: Chun-Fu Chen Advisor: Prof. Wen-Hsiang Tsai

Institute of Multimedia Engineering, College of Computer Science
National Chiao Tung University

ABSTRACT

In this study, methods are proposed for video surveillance by a video surveillance vehicle equipped with a pair of two-camera omni-imaging devices on its roof, with emphasis on monitoring of blind spots and nearby cars around the vehicle.

First, for generating perspective-view images to facilitate inspection of the vehicle's surrounding environment, a space-mapping table and an *r*-ρ mapping table are created to accelerate the related coordinate transformation process. Also, a method for generating the perspective-view image of the surrounding area of the vehicle by estimating the vehicle's moving direction using optical flow analysis is proposed. For off-line inspection of the driving history, a method of using a perspective-mapping table proposed in this study to generate a series of perspective-view images of any view direction decided by mouse clicking is proposed as well.

Furthermore, a method for monitoring a nearby static car around the surveillance vehicle is proposed, which employs image processing and pattern recognition techniques like ground region elimination, moment-preserving thresholding, region growing, etc. to segment a car shape out of the omni-image. Also proposed is a method for extracting the bottom-edge points of the car window and eliminating the outlier points by simple linear regression, in order to compute the 3D data of the detected car and generate a surround map.

In addition, a method for monitoring a nearby static or moving car from a moving video surveillance vehicle is proposed, which may be used to segment the nearby car region in the omni-image by the use of motion vector lengths. To further grow a complete car shape from the segmented regions, a method for finding the pixels of the car body by a k-means algorithm and using the pixels as seed points to grow the entire car region by the use of color information is also proposed. With the aid of a space-mapping table, car masks derived from a simple car model are used for locating the position of the detected car. Finally, a top-view surround map showing the relative position of the detected car with respect to the video surveillance vehicle is generated.

Good experimental results are also shown, which prove the feasibility of the proposed methods for real video surveillance applications.

ACKNOWLEDGEMENTS

The author is in hearty appreciation of the continuous guidance, discussions, and support from his advisor, Dr. Wen-Hsiang Tsai, not only in the development of this thesis, but also in every aspect of his personal growth.

Appreciation is also given to the colleagues of the Computer Vision Laboratory in the Institute of Computer Science and Engineering at National Chiao Tung University for their suggestions and help during his thesis study.

Finally, the author also extends his profound thanks to his dear mom and dad for their lasting love, care, and encouragement.

CONTENTS

ABSTRACT (in Chinese)……………………………………………………i
ABSTRACT (in English) ………………………………………..…….ii
ACKNOWLEDGEMENTS……………………………………………...iv
CONTENTS………………………………….…………………...…….v
LIST OF FIGURES…………………….………………..…………….viii
LIST OF TABLES………………………….……………………...…….xii

Chapter 1 Introduction...1

1.1 Motivation..1
1.2 Survey of Related Studies..3
1.3 Overview of Proposed Methods...5
1.4 Contributions..7
1.5 Thesis Organization 8

Chapter 2 Idea of Proposed Methods and System Design...............................10

2.1 Idea of Analyzing Surrounding Environment and Vehicles................10
2.2 System Configuration 14
2.2.1 Hardware configuration 14
2.2.2 Software configuration..16
2.2.3 Network Configuration 16
2.3 Review of Adopted Camera System and 3D Data Acquisition Process 17
2.4 System Processes 22

Chapter 3 Generation of Perspective-view Images Using Pano-mapping Tables 26

3.1 Review of Adopted Pano-mapping Method for Omni-image Unwarping..26
3.2 Construction of Pano-mapping Table 27
3.2.1 Landmark Learning...27
3.2.2 Estimation of Coefficients of Radial Stretching Function.......28
3.2.3 Filling of Pano-mapping Table Entries 30
3.2.4 Creation of r-ρ Mapping Table 33
3.3 Image Unwarping and Generation of Perspective-view Images..........34
3.4 Construction of Perspective Mapping Table for Computation Speedup 39

Chapter 4 Car-driving Assistance by Analyzing Omni-images of Surrounding Environment 41

4.1 Idea of Proposed Method 41
4.2 Analysis of Car Direction by Motion Vectors in Omni-images 42
4.2.1 Idea of car direction analysis by motion vectors 42
4.2.2 Car Direction Detection and Display of Corresponding Perspective-view Images 43
4.2.3 Algorithm 53
4.3 Sequential Driving Recording for Off-line Inspection of Driving History 56
4.3.1 Idea 56
4.3.2 Inspection of Sequential Driving Record via Perspective-view Image 58
4.3.3 Algorithm 59

Chapter 5 Monitoring of a Nearby Static Car around a Static Video Surveillance Vehicle 62

5.1 Idea of Static Car Detection in Omni-images 62
5.2 Nearby Vehicle Detection 63
5.2.1 Ground Region Learning 63
5.2.2 Object Segmentation by Moment-preserving Thresholding 65
5.2.3 Noise Elimination 66
5.3 Distance Estimation of a Static Car 69
5.3.1 Car Side Extraction and Analysis 69
5.3.2 Elimination of Noise by Simple Linear Regression 74
5.3.3 Calculation of Car Distance and Creation of Surround Map 77

Chapter 6 Monitoring of a Nearby Static or Moving Car with a Moving Video Surveillance Vehicle 79

6.1 Idea of Detection of Static or Moving Car in Omni-images 79
6.2 Moving Car Detection by Motion Vectors Generated by Optical Flow Analysis 81
6.2.1 Detection of Car Region by Motion Vector Lengths 81
6.2.2 Detection of Car Body by k-means Algorithm 83
6.2.3 Detection of Car Region by Color Information 86
6.3 Updating of Car State 89
6.3.1 Estimation of Car Location by Rectangular-shaped Models 89
6.3.2 Update of Car State and Generation of Surround Map 94

Chapter 7 Experimental Results and Discussions 98

7.1 Experimental Results 98
7.2 Discussions 106

Chapter 8 Conclusions and Suggestions for Future Works 108

8.1 Conclusions 108
8.2 Suggestions for Future Works 109

References 111

LIST OF FIGURES

Figure 2.1 The video surveillance vehicle used in this study with a pair of two-camera omni-directional devices affixed on the car roof. (a) A front view of the video surveillance vehicle. (b) A side view of the video surveillance vehicle. 10
Figure 2.2 Positions of cameras affixed to the video surveillance vehicle roof and the corresponding FOV. (a) The omni-camera is affixed at the rear-middle of the car roof. (b) The omni-camera is affixed at the right-rear of the car roof. 11
Figure 2.3 An example of static nearby car detection. (a) An omni-image of a static car parked at the nearby roadside. (b) A generated surround map showing the relative position from the top view. Note that the direction of an object is 180^{o} reversed in the omni-image when compared with the real situation as illustrated in (b). 13
Figure 2.4 Structure of the proposed surveillance system. 15
Figure 2.5 The network architecture of transmission between two laptops. 16
Figure 2.6 (a) Relation between the world coordinates and the image coordinates (b) Geometry between the mirror and the CMOS sensor in camera. 19
Figure 2.7 Computation of depth using the two-camera omni-directional imaging device. (a) The ray tracing of a scene point P in the imaging device with a hyperboloidal-shaped mirror. (b) A triangle in detail (part of (a)). 20
Figure 2.8 System configuration of upper omni-camera with a hyperboloidal-shaped mirror. 22
Figure 2.9 Flowchart of proposed learning process. 23
Figure 2.10 Flowchart of the moving direction analysis. 24
Figure 2.11 Flowchart of vehicle detections 25
Figure 3.1 An interface to for user to select the landmark points. 28
Figure 3.2 Mapping between a radius distance r and elevation angle ρ. 29
Figure 3.3 Illustration of mapping between the azimuth-elevation angle pair of the omni-image and the horizontal and vertical axes of the pano-mapping table, respectively. 31
Figure 3.4 An example of generating the perspective-view image. 35
Figure 3.5 A top view configuration of generating a perspective-view image. 37
Figure 3.6 A lateral-view configuration of generating a perspective-view image. 38
Figure 3.7 A top view of segmenting an omni-image. 40
Figure 4.1 Illustration of selecting the detection region where the red points represent the spots on which optical flows need be found. (a) Detection region used

in the case of turning to the right. (b) Detection region used in the case of moving forward. (c) Detection region used in the case of turning to the left. 43

Figure 4.2 The optical flow pattern and the corresponding detection region. (a) The case of turning to the right. (b) The case of moving forward. (c) The case of turning to the left. 44

Figure 4.3 An example of results of implementing the optical flow analysis method. (a) An image frame taken at time *t*. (b) An image frame taken at time *t* + *dt*. (c) The result of the motion vectors produced by the optical flow analysis method with (a) and (b) as inputs. 47

Figure 4.4 Transformation of a motion vector from the ICS to the WCS. (a) An illustration of the camera system and the motion vector. (b) The ray tracing of a scene point *P* on the ground projected on the hyperboloidal-shaped mirror...49

Figure 4.5 A distribution chart of the direction angle of motion vectors. 51

Figure 4.6 A graph of finite state machine proposed to determine the moving direction. 52

Figure 4.7 Structure of the communication between two laptops used in this study...54

Figure 4.8 An example of results of optical flow analysis on omni-images and corresponding perspective-view images, where the red arrowheads represent motion vectors. (a) Optical flows of "turning to the left." (b) Optical flow of "moving forward." (c) Optical flow of "turning to the right." (d) ~ (e) Corresponding perspective-view images of (a) ~ (c), respectively. 57

Figure 4.9 The car-driving assistance by analyzing omni-images of the surrounding environment. 57

Figure 4.10 An interface for inspecting the sequential driving record. 59

Figure 4.11 The result of inspecting the driving history. (a) The omni-image and the perspective-view image obtained from transforming the omni-image acquired with the right-front camera. (b) The omni-image and the perspective-view image obtained from transforming the omni-image acquired with the left-rear camera. 60

Figure 4.12 A flowchart of sequential driving recording for off-line inspection.........61

Figure 5.1 A flow chart of static car detection with a static video surveillance vehicle. 63

Figure 5.2 The interface for ground learning. (a) An example of initializing the region of the ground. (b) An example of selecting the ground region by a user...64

Figure 5.3 Related images of noise elimination. (a) The original omni-image. (b) The bi-level image of eliminating the ground and thresholding in the image (a).

....67
Figure 5.4 The bi-level images of the nearby static car detection. (a) The image before noise elimination. (b) The image after noise elimination. 67
Figure 5.5 An illustration of the region growing process — the blue region represents the car region and the white region represents the non-car region. Once the scan point finds the car region, the region growing process starts. 69
Figure 5.6 An illustration of detecting the edge points in bi-level image...................70
Figure 5.7 An example of edge-point extraction. (a) The bi-level image *b* for searching the bottom-edge points of the vehicle window (a) An image to show the result of finding the edge points, and the red points represent the edge points corresponding to (a)...72
Figure 5.8 The result of edge point extraction. (a) The original omni-image acquired with the omni-camera. (b) The image with the bottom-edge points of the vehicle window represented by red points...74
Figure 5.9 An example of simple linear regression, where the blue points represent the edge points transformed into the WCS and the black line is the result. 76
Figure 5.10 A surround map from the top view...78
Figure 6.1 Flowchart of nearby car detection with a moving video surveillance vehicle. 80
Figure 6.2 An example of block-based omni-image — the block region is the video surveillance vehicle roof that we ignore and the red points are the selected points..81
Figure 6.3 A result of separating the car region from the non-car region, where the red points are used to represent the car region and the green points to represent the non-car region. 83
Figure 6.4 An illustration of k-means algorithm. (a) The image of initialize the cluster centers. (b) The image of associating every data with the nearest mean. (c) The image of reassigning the cluster centers. (d) The result image of k-means algorithm. 84
Figure 6.5 A result of region growing by the color information. (a) An image to show the result of the region growing, and the purple points represent the growing region. (b) The corresponding bi-level image of the image (a). ...89
Figure 6.6 A rectangular-shaped car model and the corresponding mask. (a) A blue region of rectangular-shaped model and its representive points in the WCS. (b) The mask image. 90
Figure 6.7 The result of mask in the omni-image. (a) The near mask with respect to the video surveillance vehicle. (b) The far mask with respect to the video surveillance vehicle..92
Figure 6.8 A result of matching a detected car by a mask — the yellow region

represents the detected car and the blue mask represents the result of the rectangular-shaped model transformation...94
Figure 6.9 Flowchart of updating the car state. 96
Figure 6.10 The result of detecting the static nearby car with a moving video surveillance vehicle. (a) The original omni-image. (b) The surround map from the top view...97
Figure 7.1 An experimental result of generating the perspective-view image. (a) An original omni-image. (b) The perspective-view image of the right-rear direction. (c) The perspective-view image of the rear direction. (d) The perspective-view image of the left-rear direction. 99
Figure 7.2 A real example of car direction detection and display of corresponding perspective-view images. (a) The case of turning to the left. (c) The case of turning to the right. (e) The case of moving forward. (b), (d), and (f) The perspective-view images corresponding to (a), (c), (e), respectively. 100
Figure 7.3 The experimental result of monitoring a nearby car around a static video surveillance vehicle. (a) The omni-image acquired with an upper camera. (b) The omni-image acquired with a lower camera. (c) The surround map from the top view. Note that the direction of an object is 180^o reversed in the omni-image when compared with the real situation as illustrated in (c). 102
Figure 7.4 An illustration of the detecting a nearby static car with a moving video surveillance vehicle...103
Figure 7.5 The result of a nearby static car detection with a moving video surveillance vehicle. (a)~ (f) The results of detecting a nearby car parked at the road side and the generated top-view surround maps. Note that the direction of an object is 180^o reversed in the omni-image when compared with the real situation as illustrated in (b) (d) (f). 104
Figure 7.6 An illustration of the detecting a nearby moving car with a moving video surveillance vehicle..105
Figure 7.7 The result of a nearby moving car detection with a moving video surveillance vehicle. (a)~ (f) The result of detecting a moving car. Note that the direction of an object is 180^o reversed in the omni-image when compared with the real situation as illustrated in (b) (d) (f). 105

LIST OF TABLES

Table 3.1 An example of the pano-mapping table. …………………………………….32
Table 3.2 An example of the r-ρ mapping table. ……………………………………33
Table 4.1 The range of the angles of the three vehicle moving directions…………..51

Chapter 1 Introduction

1.1 Motivation

Nowadays, because the computer technology progresses quickly, video cameras are getting more popular and used more widely. In people's daily life, video cameras can be used to improve human beings' welfare. For example, people often equip cars with driving-assistance systems like digital driving recorders or car-backing monitoring systems. With the assistance of cameras in these systems, a driver is able to observe surrounding environments easily. Once a traffic accident occurs, he/she can clarify the responsibility for the event by inspecting the video record.

Moreover, video cameras are also useful for developing vision-based techniques for many applications. Through image processing and other techniques, much information can be obtained from images captured with video cameras. For example, a license plate recognition system or a face recognition system requires the use of video cameras to capture images for analysis of specific objects. In this study, it is desired to design a video surveillance system using video cameras on a vehicle, called *video surveillance vehicle*, for car-driving assistance and car surrounding monitoring applications.

Most researches of vision-based techniques for the mentioned applications are based on the use of traditional *projective* cameras; however, the *limited* field of view (FOV) of the traditional camera is a problem. For instance, only the scene in front of a car can be seen when a projective camera is affixed to the car to "see" forward. If we want to monitor the entire car surrounding, four or more cameras are required. The requirement of extra cameras to cover the entire surrounding will raise the cost and complexity to develop a video surveillance system on the car. Therefore, we choose omni-cameras (or simply *omni-cameras*) to be the imaging devices in this study. Each device consists of two axis-aligned omni-cameras. A wider view of the environment around the video surveillance vehicle can be covered by such a camera system.

Besides, most vision-based systems are affixed to some pre-determined places, such as ceilings or utility poles. It is a difficult task to move a system of such a nature entirely to another place to do surveillance works. In this study, we set up two pairs of omni-cameras on the roof of a video surveillance vehicle. With the camera system being carried, the vehicle can move to any place to conduct surveillance works. Furthermore, the cameras equipped on the video surveillance vehicle can also be used to develop functions for various applications.

To sum up, the goal of this study is to develop a video surveillance system on a video surveillance vehicle for the applications of car-driving assistance and car surrounding monitoring. The system is composed of two pairs of two-camera omni-imaging devices, each consisting of two vertically-aligned omni-cameras, which are affixed to the surveillance's roof. With the advantages of mobility of the video surveillance vehicle and the wide FOV's of the omni-cameras, we can develop a mobile surveillance system to monitor car surroundings completely for the two applications. Listed below are the more detailed desired capabilities of the proposed system.

(1) The surrounding environment images are captured by omni-cameras, and the captured image sequence is analyzed to decide the driving direction of the vehicle and generate the perspective-view image of the car surrounding environment with respect to a selected view direction.

(2) The proposed video surveillance system is able to detect any static surrounding car parked at the nearby roadside, and displays a *surround map* to show its relative position with respect to the surveillance vehicle from the top view.

(3) When the video surveillance vehicle is moving, the proposed video surveillance system can detect a static car or a passing car in the nearby surrounding as well, and displays a top-view surround map as mentioned above to show the relative position of the car.

1.2 Survey of Related Studies

In this section, we conduct a survey of related studies about video surveillance, including designs of omni-cameras for uses on vehicles, techniques for surrounding vehicle detection, and the optical flow method which we use in the proposed video surveillance system for various purposes.

In recent years, video surveillance for various applications has been widely investigated. Christian et al. [1] proposed a method to track concerned objects based on the use of a video surveillance system on a vehicle. The vision-based surveillance system can be used to monitor a parking lot or conduct traffic surveillance works [2, 3].

An omni-camera in a video surveillance system is useful for localizing objects. It includes just a projective camera and a mirror. Onoe et al. [4] and Mituyosi et al. [5] proposed methods to track people on video surveillance systems with omni-cameras. Moreover, applications using different combinations of projective cameras and mirrors to construct new types of omni-imaging systems have also been investigated. For example, a method to obtain stereo information for mobile robot navigation with an omni-imaging system which consists of two mirrors and one camera was proposed by He et al. [6]. Also, Koyasu et al. [7] proposed a stereo system which consists of two omni-cameras aligned vertically for obstacle detection and tracking.

In this study, to obtain the stereo information from omni-images, we adopt the space-mapping method proposed by Jeng and Tsai [8] to calibrate omni-cameras without knowing the intrinsic and extrinsic parameters of the cameras. Moreover, because omni-images captured with omni-cameras may be processed to produce panoramic images and estimate the relevant stereo information of surrounding objects, many studies replace conventional projective cameras with omni-cameras. For example, to assist a driver to observe the entire car surrounding environment, researchers proposed techniques to generate surrounding bird's-eye views from omni-images, such as Ehlgen and Pajdla [9]. Gandhi and Trivedi [10] also proposed a method to use omni-cameras to conduct detection of moving persons and vehicles on a mobile platform. By the way, the large FOV's of an omni-camera is a great benefit to monitor the entire surrounding environment. Some researchers combine this advantage and the mobility of vehicles to develop applications [11-13].

In addition, the optical flow method is useful for analyzing the motions within two consecutive image frames. Lucas and Kanade [14] proposed a method to compute the displacement of the image contents between two image frames within the neighborhood of a point. In many studies, the optical flow method is used to detect ego-motions for analyzing the car moving direction. Kim and Suga [15] proposed a method to detect a moving obstacle using an optical flow method for a mobile robot with an omni-camera.

For long, the topic of vehicle detection has been widely studied. Various techniques were proposed to detect vehicles. For instance, background subtraction is a common technique used to extract vehicles [16, 17]. Tsai et al. [18] proposed a method to conduct vehicle detection from static images using color and edges information. In this study, we propose methods to detect a static surrounding car by ground-region subtraction and to detect a moving car by using the motion vectors and color information of the car.

1.3 Overview of Proposed Methods

1.3.1 Terminologies

The definitions of some related terms used in this study are described as follows.

1. *Omni-camera*: a camera system with a traditional projective camera and a reflective mirror which can be used to capture images of 360-degree FOV's.
2. *Omni-image*: an image captured with an omni-camera.
3. *Video surveillance vehicle*: a car with a pair of two-camera omni-imaging devices equipped on the car roof as well as two laptops for use as control units inside the car to develop a video surveillance system.
4. *Optical flow*: a method to estimate the motions of shapes, surfaces, and edges of concerned objects between two sequential images.
5. *Motion Vector*: motion vectors produced by the optical flow method to represent the velocity and direction of a concerned object.
6. *Perspective-view image*: an image obtained by projecting a scene onto a flat surface as it is seen by the human eye.
7. *Surround map*: an image showing the relative position of a surrounding car from the top view.

1.3.2 Brief Descriptions of Proposed System

There are four goals in developing the proposed system as described in the following.

1. The system is able to analyze the car driving direction using the consecutively acquired omni-images and display corresponding perspective-view images to the driver.
2. The system is capable of recording the surrounding images during driving and let the user see the perspective-view image in any selected view direction constructed from these sequential images as well as inspect the sequential images off-line.
3. The system is able to monitor a static surrounding car and obtain related stereo information of it to generate and display a top-view surround map.
4. The proposed system is able to monitor a passing car or a static car in the surrounding environment when the surveillance vehicle is moving, and display a top-view surround map to show its relative position.

In order to achieve the above goals, the following are the major steps of the system process of the proposed video surveillance system.

1. Set up the previously-mentioned pair of two-camera omni-image devices on the roof of the video surveillance vehicle with one on the front-right corner and the other on the rear-left corner of the car roof.
2. Calibrate the omni-cameras for six outward view directions and use the space-mapping method to generate six corresponding pano-mapping tables.
3. While the surveillance vehicle is moving, analyze its moving direction by the optical flow method, transform the extracted motion vectors into the world

coordinate system (WCS), and estimate the moving directions by some pattern recognition methods.

4. Generate a perspective-view image according to the driver's selection of the view direction by projecting an omni-image onto a flat surface by looking up the pano-mapping tables.
5. Detect any static surrounding car by elimination of the ground regions in an omni-image, extract the edge points of a detected car, use the points to compute the stereo information of the detected car, and display the corresponding top-view surround map including the car.
6. Detect a passing-by car or a static surrounding car by the optical flow method, use the lengths of these motion vectors to roughly separate ground regions and car regions, apply region growing based on the color information to find the complete car shape, estimate the stereo information of the car, and generate a top-view surround map to show the relative position of the car.

1.4 Contributions

The following is a list of the major contributions made in this study.

1. A method for obtaining the stereo information of a concerned object by the pano-mapping method using a pair of two-camera omni-imaging devices is proposed.
2. A method for constructing the pano-mapping tables for six outward view directions is proposed.
3. A method for constructing tables of perspective mapping described in detail in Chapter 3 between the omni-images and the perspective-view image to shorten the processing time of image transformation is proposed.

4. A method for analyzing the driving directions of the video surveillance vehicle by the optical flow method and using the direction to generate corresponding perspective-view images is proposed.
5. A local network is constructed, which integrates a pair of two-camera omni-directional imaging devices and two laptop computers for video surveillance use.
6. A method for detecting a static surrounding car and computing its accurate 3D information is proposed.
7. A method to detect a moving surrounding car or a static surrounding car by thresholding the lengths of motion vectors and extracting the region of the car is proposed.
8. A method for generating rectangular-shaped models and transforming them into the camera coordinate system to mask a detected car for computing the location of the car is proposed.
9. A method for generating a surround map to display the relative location of a detected car in an acquired omni-image with respect to the surveillance vehicle is proposed.

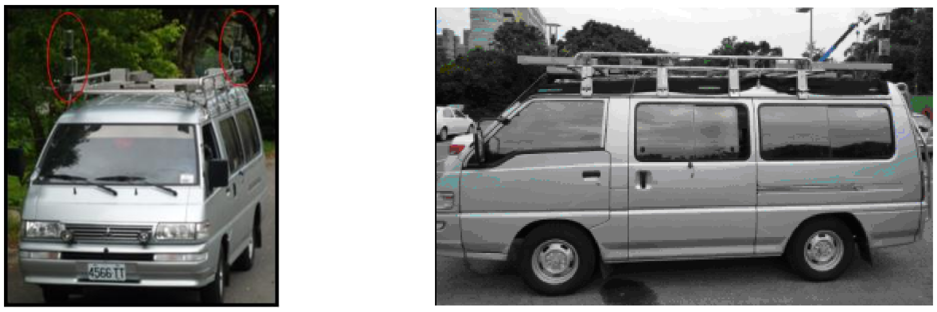
1.5 Thesis Organization

In the remainder of this thesis, we introduce the system configuration and the idea of the proposed method in Chapter 2. The designs of the camera system and the method to obtain stereo information are also described. In Chapter 3, the construction of the pano-mapping tables by the space-mapping technique and the technique of using the pano-mapping tables for unwarping an omni-image into multiple perspective-view images are described. In Chapter 4, the proposed methods for computing the car moving direction and for generating the corresponding perspective-view image are described. In Chapter 5, the proposed method for detecting a static surrounding car is described. In Chapter 6, the proposed method for detecting a moving surrounding car is described. In Chapter 7, experimental results and discussions are included. Finally, conclusions and some suggestions for future works are given in Chapter 8.

Chapter 2
Idea of Proposed Methods and System Design

2.1 Idea of Analyzing Surrounding Environment and Vehicles

In order to monitor the surrounding environment of the video surveillance vehicle, we choose omni-cameras instead of traditional projective cameras to acquire environment images. The acquired omni-images can be used to generate corresponding panoramic images and so provide necessary information for security monitoring or driving assistance. In this study we affix a pair of two-camera omni-imaging devices to the surveillance vehicle roof for this purpose as shown in Figure 2.1. Each device includes two omni-cameras aligned coaxially and back to back, as mentioned previously.



(a) (b)

Figure 2.1 The video surveillance vehicle used in this study with a pair of two-camera omni-directional devices affixed on the car roof. (a) A front view of the video surveillance vehicle. (b) A side view of the video surveillance vehicle.

An advantage of the mobility of the video surveillance vehicle is that we can move the entire system to everywhere to conduct surveillance works. Besides, to get useful views as far as possible, we decided to affix one omni-image device at the right-front position of the surveillance vehicle roof, and the other at the left-rear. As illustrated in Figure 2.2, if instead we affixed a device at the front (or back) middle of the vehicle roof, a half of the acquired omni-image is useless, covering just the roof of the vehicle.

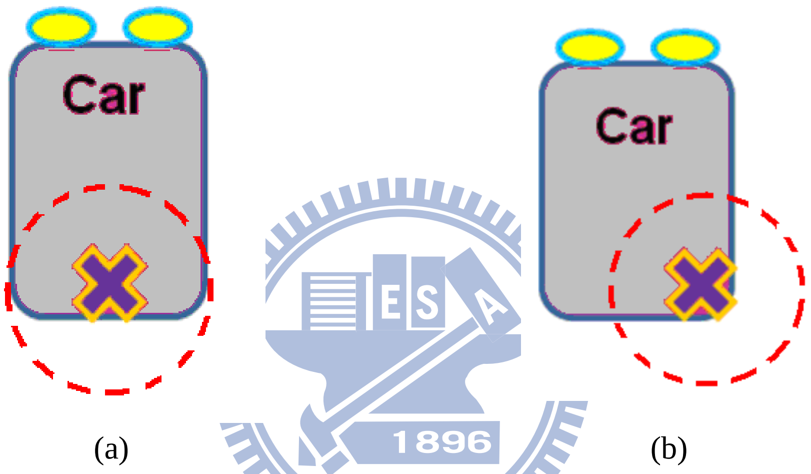


Figure 2.1 Positions of cameras affixed to the video surveillance vehicle roof and the corresponding FOV. (a) The omni-camera is affixed at the rear-middle of the car roof. (b) The omni-camera is affixed at the right-rear of the car roof.

Many car accidents occur because the driver ignores “blind spots” which cannot be seen in the mirrors equipped inside and outside the car. To show the views of these blind spots, we can use the mentioned pair of omni-imaging devices on the surveillance vehicle roof to generate perspective-view images around the car on every driver-specified view direction. The construction of the perspective-view image from an omni-image conducted in this study is based on the space-mapping method proposed by Jeng and Tsai [8].

In addition, when a driver wants to turn to the left or to the right, the blind spots behind the surveillance vehicle are apt to be neglected. Therefore, we analyze the motion vectors produced by the optical flow method in consecutively acquired images. With these motions, we can estimate the vehicle moving direction and show the corresponding perspective-view image. During driving, we can also store images captured with omni-cameras. As a driver recorder, the system can then display these images in sequence, or let the user to choose a view direction and display the corresponding perspective-view image for closer observation.

Furthermore, in order to detect a static car parked at the nearby roadside and compute the stereo information of it, we use the color feature to separate the car region from the ground in the acquired omni-image. In doing this, we assume that the background is uncomplicated with almost the same color as that of an asphalt road, and that the color of the detected car is different from the ground presumably. Therefore, the car shape can be extracted as the foreground by elimination of the ground color.

Moreover, we also want to obtain the stereo information of the detected car. For this, the corresponding points of the bottom-window edge of the car in a pair of images captured with the upper omni-camera and the lower omni-camera are chosen. Then, the image data of these points are used to compute the desired stereo information. However, some points like the outlier ones might incur errors in the computed stereo information, so they are eliminated by a linear regression method in this study. As a result, we can compute the location of the car by using the image data of the remaining points, and generate accordingly a surround map. An example of the result of this process is shown in Fig. 2.3.

Finally, we use motion vectors to analyze the acquired omni-images for several purposes. Such motion vectors are produced from consecutive omni-images directly when concerned objects are moving in the omni-images. Specifically, the angles and the lengths of these motion vectors are almost all equal when the vehicle is driven on a flat field with roughly identical texture everywhere. Accordingly, if another car is driven aside to overtake the surveillance vehicle gradually, the angles of the motion vectors of the car will differ from those of the motion vectors produced from the entire environment. This characteristic so can be utilized to detect a nearby car in an acquired omni-image.

Another feature used in this study is motion vector length. If a concerned object is higher than the ground, the lengths of the motion vectors yielded by it will be longer than those yielded by the ground. After roughly locating the car using this feature, we use a third feature, the color of the monitored car, to grow the car region in the omni-image, and compute accordingly the location of the car by the use of a mask model of the car. Finally, a surround map is generated to show the relative position of the nearby car with respect to the surveillance vehicle from the top view for driving assistance.

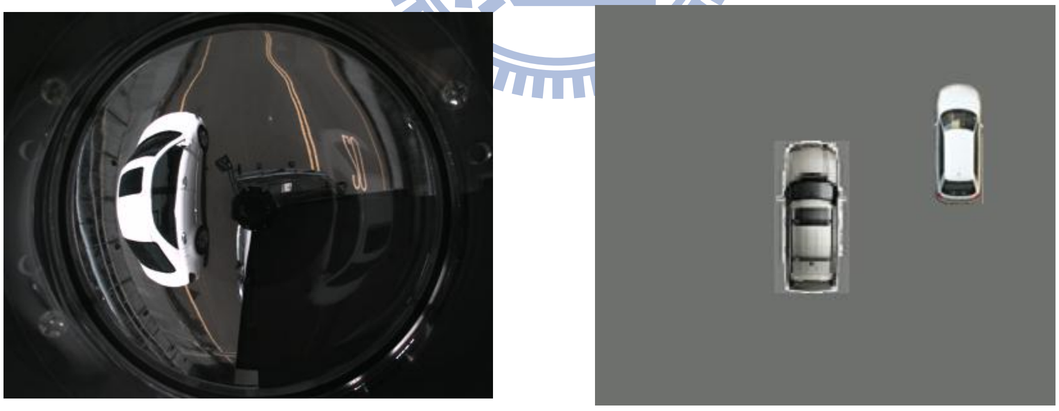

(a) (b)

Figure 2.3 An example of static nearby car detection. (a) An omni-image of a static car parked at the nearby roadside. (b) A generated surround map showing the relative position from the top view. Note that the direction of an object is 180^{o} reversed in the omni-image when compared with the real situation as illustrated in (b).

2.2 System Configuration

In this section, we will describe the video surveillance system elaborately. The proposed system is mainly divided into three parts. The first part is the hardware which includes a video surveillance vehicle, a pair of two-camera omni-directional imaging devices, and two laptop computers. The second part is the software. In this part, we will introduce the software development environment and the accompanying SDK and driver programs for the CCD cameras. The third part is the network. Because we use two laptops to handle the pair of two-camera omni-directional imaging devices, respectively, a local network is used for communication between the two laptops.

2.2.1 Hardware configuration

The surveillance vehicle, named Delica, is made by Mitsubishi Co. It is a 469cm ×169cm×196cm vehicle with a working table and a power supply. System operators may sit inside the surveillance vehicle to operate the laptop computers and monitor the entire surrounding environment. Moreover, a steel frame is affixed to the car roof, on which the omni-image devices can be affixed. And four extension USB cords crossing the video surveillance vehicle were added to receive images which are captured with the two omni-imaging devices. Detailed descriptions of the imaging devices will be given in Section 2.3. The entire video surveillance system is shown in Fig. 2.4.

In order to control the entire video surveillance system, in this study we use two laptops as control units, each handling an omni-imaging device. Both laptops are produced by TOSHIBA Computer Inc., and their detailed specifications are listed in

Table 2.1. To exchange commands and images between the two laptops, we use a cross-over cable to connect then and set up a local network for between-computer communication.

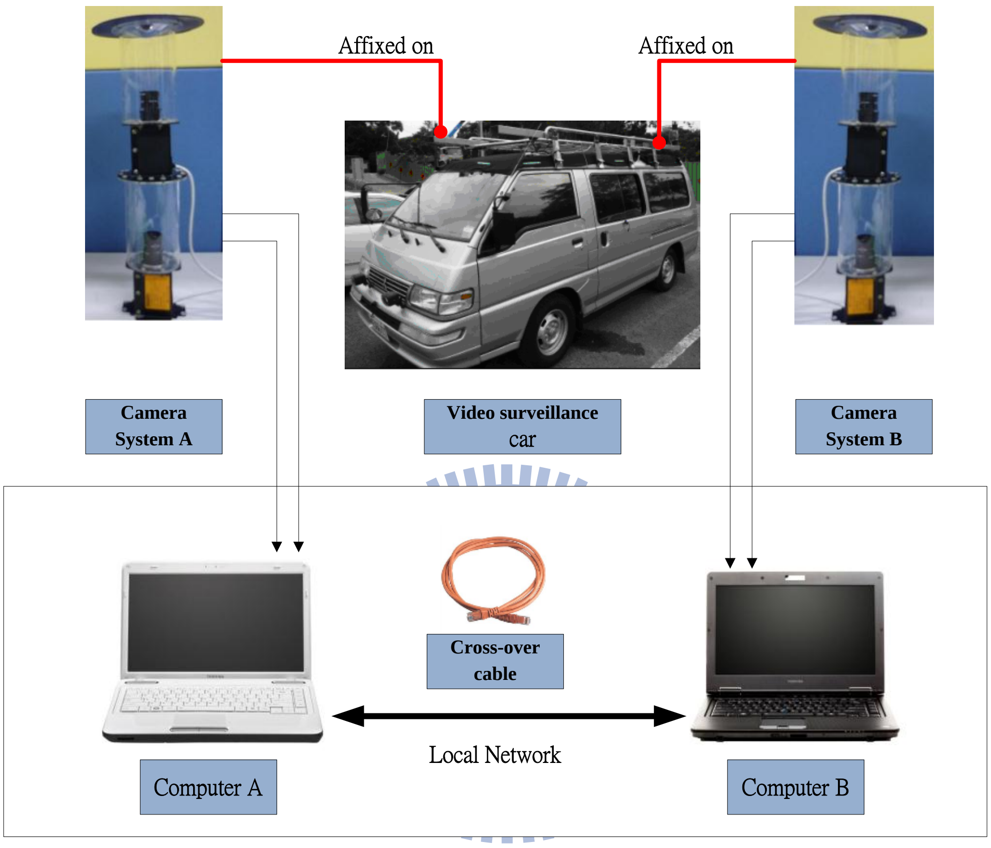


Figure2.4 Structure of the proposed surveillance system.

Table 2.1 Specifications of the laptop computers used in this study.

	Tecra M11	Satellite A660
CPU	Intel Core i7-620M 2.66/3.33GHz	Intel Core i5-480M 2.66/2.93GHz
RAM	4G DDR3 1066MHz	2G DDR3 1066MHz
GPU	nVidia NVS 2100M	ATI HD5650
Network	Gigabit LAN	Fast Ethernet LAN

2.2.2 Software configuration

We use Borland C++ Builder (BCB) V6 as a developed platform to build our video surveillance system. The BCB is a program development tool for the operating system of Windows; therefore, we can create a graphic user interface (GUI) conveniently and quickly. The programming language we use is C++. It is a widely used language. One of the laptops, the Tecra M11 computer, uses the operating system of Windows 7, and the other, Satellite A660, uses Windows XP.

Before developing a video surveillance system, we have to install the drivers of the ARTCAM-200SO cameras and those of the ARTCAM-200SS cameras in the laptop computers. The camera company also provides corresponding software development kits (SDKs) and some simple source codes. Accordingly, we can adjust the parameters of each camera, such as the value of exposure or the global color gain, through the SDK. The SDK is an object-oriented toolkit, and the camera company not only provides the BCB version but also the C, VB.NET, C#.NET or Delphi version to the programmers.

2.2.3 Network Configuration

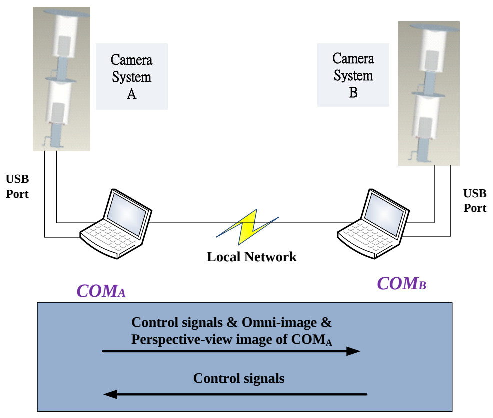


Figure 2.5 The network architecture of transmission between two laptops.

A network configuration is needed for communication between two laptop computers because four omni-images are acquired from the pair of two-camera omni-directional imaging devices and each imaging device is processed by a respective laptop. As a result, to communicate between the two laptops, we set up a local area network to send images and control signals.

As shown in Fig. 2.5, laptop computer COM_B is used to display the perspective-view image and the acquired omni-image, therefore, laptop computer COM_B needs to receive these images from COM_A through the local network. Moreover, the control signals of the selected view direction produced by COM_A are sent to COM_B for generating the corresponding perspective-view image.

2.3 Review of Adopted Camera System and 3D Data Acquisition Process

In this section, we review the adopted camera system and the corresponding 3D data acquisition process proposed in Yuan el at. [19]. First of all, we introduce the detail of building the camera system. The entire system includes four lenses of model LV0612H, two CMOS cameras of model ARTCAM-200SO, and two CMOS cameras of model ARTCAM-200MI. Table 2.2 lists the specifications of the COMS cameras.

To build an omni-camera, the most important task is to combine a projective CCD camera and a hyperboloidal-shaped mirror into an omni-camera. In the design process of the omni-camera, an optics manufacturer was requested to produce hyperboloidal-shaped mirrors. The parameters of each of the mirrors are described here. The radius r of the hyperboloidal-shaped mirror is 4cm. The projective camera has a focal length f of 6 mm and a sensor width S_w of 2.4mm. And the axis of the camera is aligned with the axis of the hyperboloidal-shaped mirror. Therefore, by the principle of similar triangles, the distance d between the optical center of the lens and the mirror center can be computed from the following equation:

$$\frac{d}{r} = \frac{f}{S_w}. \tag{2.1}$$

Also, as shown in Fig. 2.6(a), the hyperboloidal shape of the mirror in the camera coordinate system may be described as:

$$\frac{R^2}{a^2} - \frac{Z^2}{b^2} = -1, \qquad R = \sqrt{X^2 + Y^2}. \tag{2.2}$$

To get the parameters a and b of the hyperboloidal shape of the mirror, first the elevation angle α in Figure 2.6 (a) can be obtained from the relation between the CCS and the ICS of an omni-camera system derived by Wu and Tsai [20] as follows.

$$\tan\alpha = \frac{(b^2 + c^2)\sin\beta - 2bc}{(b^2 - c^2)\cos\beta}. \tag{2.3}$$

Furthermore, by the simple formula d = 2c where the value c is the distance from the center O shown in Fig. 2.6 to the mirror center O_m, the angles θ and β can be computed as follows:

$$\theta = \tan^{-1}\frac{r}{2c}, \qquad \beta = \frac{\pi}{2} - \theta. \tag{2.4}$$

In Eq. (2.3), let the omni-camera have the largest FOV, the incidence angle α be set 0, and by using Eq. (2.4), the parameter b can be obtained by solving Eq. (2.3). Finally, the parameter a is derived from the following equation:

$$c = \sqrt{a^2 + b^2}. \tag{2.5}$$

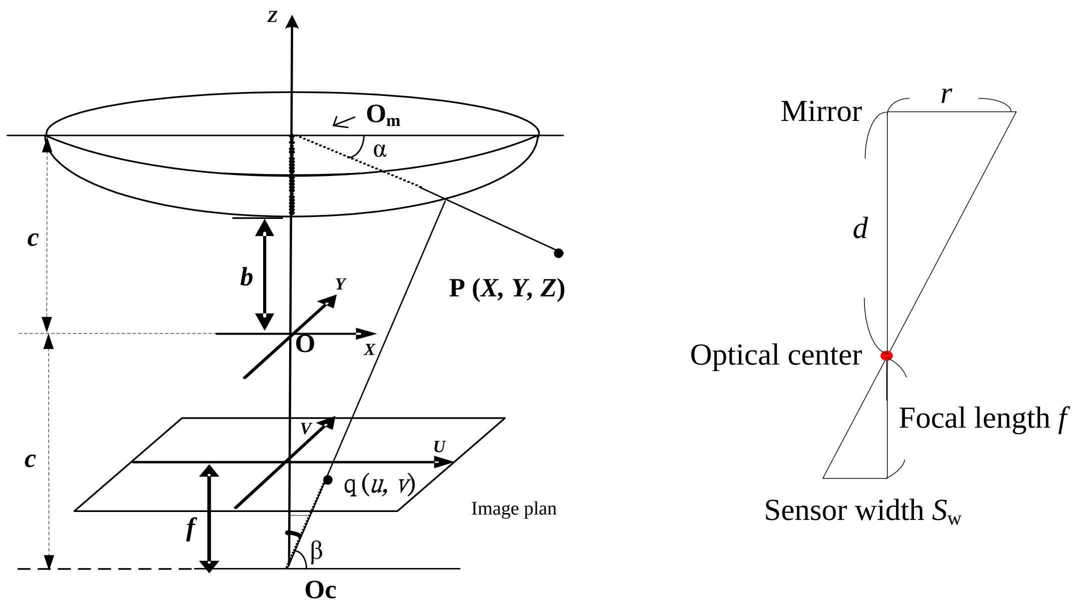


Figure 2.6 (a) Relation between the world coordinates and the image coordinates (b) Geometry between the mirror and the CMOS sensor in camera.

Each omni-camera was built with these parameter values, and a two-camera omni-directional device can be constructed with two omni-cameras aligned vertically.

Table 2.2 Specifications of used COMS cameras.

	ARTCAM-200SO	ARTCAM-200MI
Resolution	2.0 M pixels(1600*1200)	2.0 M pixels(1600*1200)
Dimension	33mm × 33mm × 50mm	33mm × 33mm × 50mm
CMOS sensor size	1/2” (6.4×4.8mm)	1/2” (6.4×4.8mm)
Mount	C-mount	C-mount
Frame per second	8 fps	5 fps
Direct show camera	Yes	No

After describing the way of building the cameras, we now describe the adopted method to compute stereo information from a two-camera omni-directional imaging device. In the omni-imaging device, relevant 3D data can be computed by two elevation angles and an azimuth angle of a scene point *P*. As shown in Figure 2.7(a), the point *P* projects on each hyperboloidal-shaped mirror and forms a pair of corresponding points in the upper image and the lower image captured with a two-camera omni-imaging device. The elevation angles of point *P* on the hyperboloidal-shaped mirrors are defined as α_1 and α_2, respectively. Also, the center of the upper hyperboloidal-shaped mirror is assumed to be the origin of the world coordinates (0, 0, 0). It is desired now to compute the stereo depth data of point *P* in terms of the two elevation angles α_1 and α_2.

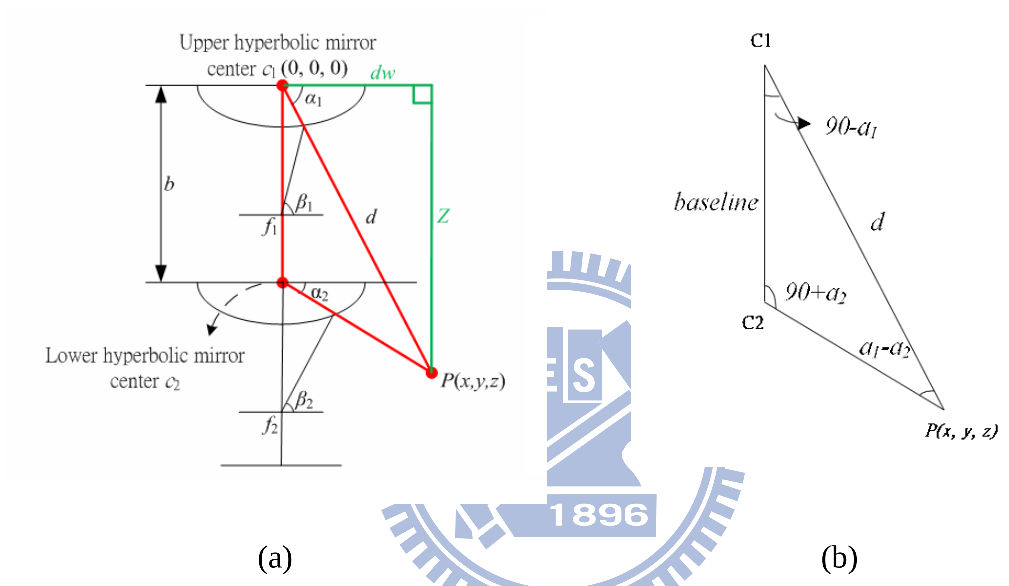


Figure2.7 Computation of depth using the two-camera omni-directional imaging device. (a) The ray tracing of a scene point *P* in the imaging device with a hyperboloidal-shaped mirror. (b) A triangle in detail (part of (a)).

To obtain stereo depth of a scene point *P*(*x*, *y*, *z*), finding two elevation angles α_1 and α_2 by looking up a pano-mapping table is required, and the construction of pano-mapping table will be described in Chapter 3. As shown in Figure 2.7(b), the distance *d* between the point *P* and the upper mirror center c_1 is computed by the triangulation principle shown in Figure 2.7(a) using the equation below:

$$\frac{d}{\sin(90^\circ+\alpha_2)}=\frac{b}{\sin(\alpha_1-\alpha_2)}, \tag{2.6}$$

where the parameter *b* is the baseline of the stereo imaging device.. The equation of

(2.6) may be reduced to be the following equation by trigonometry:

$$d = \cos\alpha_2 \times \frac{b}{\sin\alpha_1 \times \cos\alpha_2 - \cos\alpha_1 \times \sin\alpha_2},$$
$$d = \frac{1}{\cos\alpha_1} \times \frac{b}{\tan\alpha_1 - \tan\alpha_2}. \quad (2.7)$$

As a result, the horizontal distance *dw* and the vertical distance *Z* may be computed as follows:

$$z = d\sin\alpha_1 = \frac{\tan\alpha_1}{\tan\alpha_1 - \tan\alpha_2} \times b,$$
$$dw = d\cos\alpha_1 = \frac{1}{\tan\alpha_1 - \tan\alpha_2} \times b. \quad (2.8)$$

Assume that point *P* at world coordinates (*x*, *y*, *z*) is projected on a point *I* at image coordinates (*u*, *v*) in the image coordinate system (ICS). Then, we can use point *I* to calculate the azimuth angle θ. A triangulation which is illustrated in Figure 2.8 includes an azimuth angle θ between the *X*-axis and point *I*. As a result, the azimuth angle θ can be computed by the following equation:

$$\theta = \cos^{-1}\frac{\mu}{\sqrt{u^2+v^2}} = \sin^{-1}\frac{v}{\sqrt{u^2+v^2}}. \quad (2.9)$$

According to the characteristic that the axis of the camera is aligned vertically with the axis of the hyperboloidal-shaped mirror as well as the rotation-invariance property of omni-imaging, the azimuth angle of a point in the ICS is the same as that of the corresponding point in the WCS. We can calculate the parameters *x* and *y* by the distance *dw* and the azimuth angle θ in the WCS as follows:

$$x = dw \times \cos\theta = \frac{1}{\tan\alpha_1 - \tan\alpha_2} \times b \times \cos\theta,$$
$$y = dw \times \sin\theta = \frac{1}{\tan\alpha_1 - \tan\alpha_2} \times b \times \sin\theta. \quad (2.10)$$

As a result, by the use of the pano-mapping table, each pixel in an omni-image can be transformed to an elevation angle and an azimuth angle. Once the azimuth angle θ and a pair of elevation angles α_1 and α_2 are obtained, we are able to compute the location of point P in the WCS. Therefore, a pair of matching points (one is in an omni-image taken by the upper omni-camera, and the other is in an omni-image taken by the lower omni-camera) is known, the stereo information of the unique point in the WCS may be obtained.

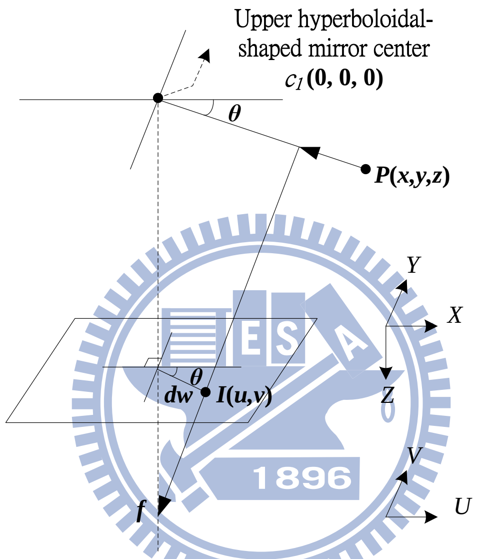


Figure2.8 System configuration of upper omni-camera with a hyperboloidal-shaped mirror.

2.4 System Processes

To get stereo information from a pair of two-camera omni-imaging devices, the omni-cameras need to be calibrated. For this purpose, the space-mapping technique is applied, and the technique is based on the use of a pano-mapping table. The process of constructing the table is shown by Fig. 2.9. We will introduce the process in Chapter 3 elaborately. Moreover, the method to unwarp an omni-image into a perspective-view image using the pano-mapping table is also described in Chapter 3.

The above-mentioned process is an advance preparation before developing the system. As shown in Figure 2.10, to develop the car-driving assistance application, we need to read the related tables at the beginning. Computer COM_A used in this study is responsible for analyzing omni-images of the surrounding environment by the optical flow method and estimating the moving direction of the video surveillance vehicle by the produced motion vectors. Then, the analyzed result and acquired images are sent to another computer COM_B for generating the corresponding perspective-view image. The process will be introduced elaborately in Chapter 4. Computer COM_B generates the corresponding perspective-view images with respect to these signals. The method of quickly generating a perspective-view image is described in Chapter 3. Moreover, the images of the driving history in Computer COM_A are sent to Computer COM_B for off-line inspection. The images of the driving history can be sequentially displayed; in the meantime, the user may select the view direction to inspect the corresponding perspective-view image sequence.

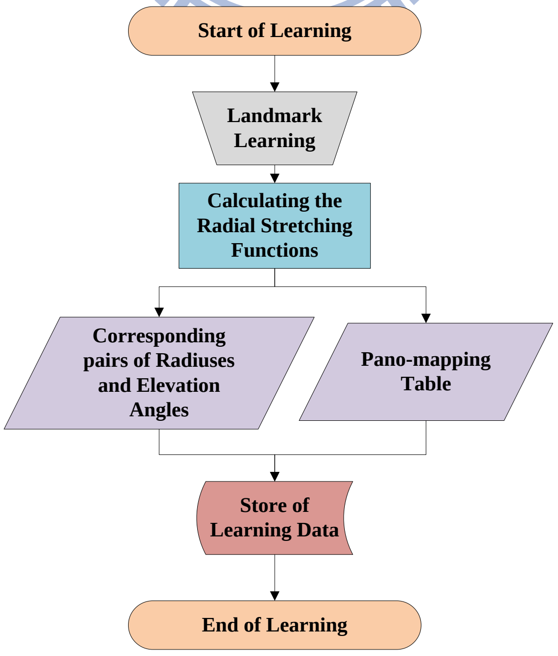

Start of Learning
Landmark Learning
Calculating the Radial Stretching Functions
Corresponding pairs of Radiuses and Elevation Angles
Pano-mapping Table
Store of Learning Data
End of Learning

Figure 2.9 Flowchart of proposed learning process.

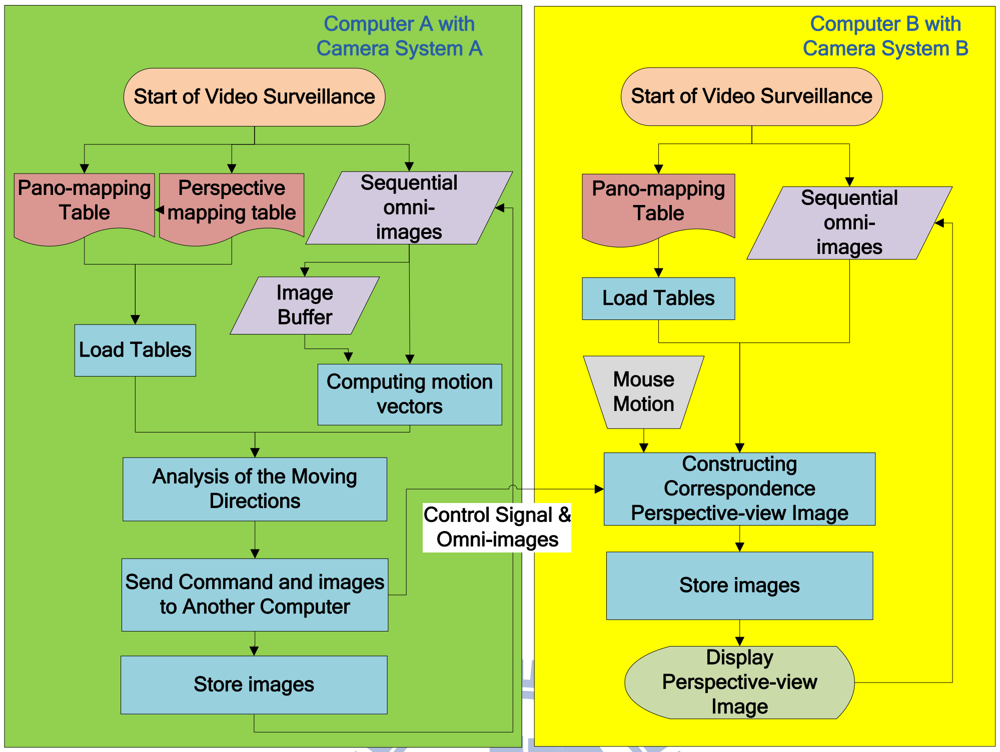

Computer A with Camera System A
Start of Video Surveillance
Pano-mapping Table
Perspective mapping table
Sequential omni-images
Image Buffer
Load Tables
Computing motion vectors
Analysis of the Moving Directions
Send Command and images to Another Computer
Store images
Control Signal & Omni-images
Computer B with Camera System B
Start of Video Surveillance
Pano-mapping Table
Sequential omni-images
Load Tables
Mouse Motion
Constructing Correspondence Perspective-view Image
Store images
Display Perspective-view Image

Figure 2.10 Flowchart of the moving direction analysis.

Both the application of nearby static car detection with a static video surveillance vehicle and the application of nearby static or moving car detection with a moving video surveillance vehicle require reading table files as a preparation task as shown in Fig. 2.11. In the application of nearby static car detection with a static video surveillance vehicle, two omni-images are captured with a two-camera omni-directional imaging device and the detection process is conducted. Finally, the stereo information of the car is computed. The detailed process will be introduced in Chapter 5. Additionally, nearby static or moving car detection with a moving video surveillance vehicle requires two consecutively acquired images to conduct the detection process. We need to create an image buffer to keep the previous image and acquire a current image with the omni-camera for analyzing the omni-images of the surrounding environment. The detection and position estimation of the static or moving car will be described in Chapter 6. Finally, the two applications will both display the surround map. Both tasks are required complex processes, so we will introduce above-mentioned processes in the remaining chapters elaborately

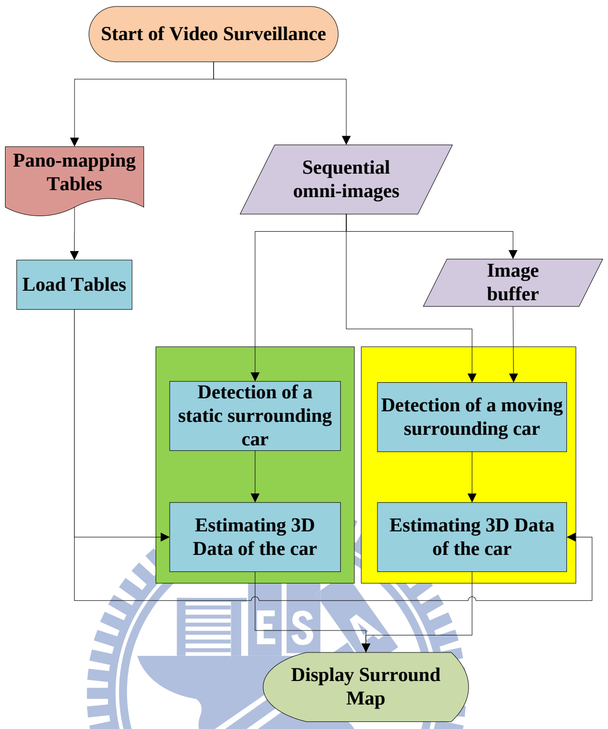

Start of Video Surveillance
Pano-mapping Tables
Sequential omni-images
Load Tables
Image buffer
Detection of a static surrounding car
Detection of a moving surrounding car
Estimating 3D Data of the car
Estimating 3D Data of the car
Display Surround Map

Figure 2.11 Flowchart of vehicle detections

Chapter 3
Generation of Perspective-view Images Using Pano-mapping Tables

In this chapter, we describe the details of the scheme we use to generate perspective-view images from omni-images acquired with the omni-image devices attached to the roof of the video surveillance vehicle used in this study. Before describing the detail in Sections 3.2 through 3.4, we review first in Section 3.1 a space-mapping technique [8] we adopt for use in coordinate mapping from the omni-image to the perspective-view image.

3.1 Review of Adopted Pano-mapping Method for Omni-image Unwarping

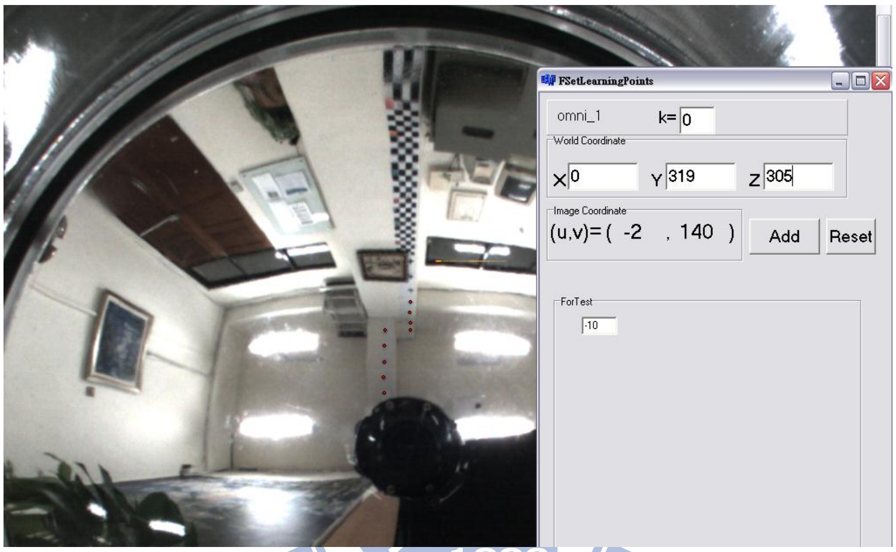
The scene appearing in an omni-image is distorted due to the light reflection on the hyperboloidal-shaped mirror. In order to facilitate observation of the distorted image, we want to unwarp the omni-image into a perspective-view image. The conventional method for unwarping the omni-image requires the parameters of the hyperboloidal-shaped mirror and the camera. However, sometimes we cannot obtain such parameter information completely. The space-mapping technique proposed by Jeng and Tsai [8] can solve this problem and is adopted in this study. The technique is based on the use of a so-called pano-mapping table which records the relationship between the pixel in the image and the elevation and azimuth angles of the corresponding world-space point with respect to the focal center of the mirror. The creation of the pano-mapping table includes three major steps: (1) landmark learning; (2) estimation of coefficients of radial stretching function; and (3) filling of the pano-mapping table entries. We will introduce these steps in Section 3.2.

With the use of the pano-mapping table, an omni-image can be transformed into perspective-view images. The transformation process will be introduced in Section 3.3. Besides, the process of generating perspective-view images is generally complicated. To shorten the computation time, we divide the omni-image into portions seen from six outward viewing directions, and create a table to record the relationship between each of the six omni-image portions and its perspective-view image.

3.2 Construction of Pano-mapping Table

3.2.1 Landmark Learning

The first step of creating the pano-mapping table is to establish several pairs of world-space point and the corresponding image point in the taken omni-image. The coordinates of the world-space points in these pairs, called *landmark points*, are measured *manually* with respect to a selected origin in the world space. To facilitate selecting the landmark point pairs, a user interface was provided, as shown in Fig. 3.1. We define the focal center O_c in the image coordinates (u_0, v_0) as the origin in the image coordinates system and O_m as the focal point of the hyperboloidal-shaped mirror at the world coordinates (X_0, Y_0, Z_0). Assume that n sets of corresponding points are selected, and each set include a landmark point p_i at image coordinate (u_i, v_i) with respect to O_c as well as the corresponding world-space point P_i at the coordinates (X_i, Y_i, Z_i) with respect to the origin O_m, where $i = 0, 1, \ldots, n - 1$. In this step, the construction of a mapping table requires learning at least six landmark points. And the more landmark points are selected, the more accurate the table is.



omni_1 k= 0

World Coordinate

X 0 Y 319 Z 305

Image Coordinate

(u,v)=(-2 , 140)

ForTest

-10

Figure 3.1 An interface to for user to select the landmark points.

3.2.2 Estimation of Coefficients of Radial Stretching Function

Due to the nonlinear shape of the hyperboloidal-shaped mirror, the radial-directional mapping must be represented as a non-linear function. As shown in Figure 3.2, each elevation angle corresponds to a radial distance. More specifically, each elevation angle ρ of a scene spot P at world coordinates (X, Y, Z) corresponds to the radius r of the corresponding point p in the omni-image. Therefore, we want to find out the relationship between the radius r and the elevation angle ρ by the use of a non-linear function f_r, called *radial stretching function*. In this study, f_r is approximated by the following 5th-degree polynomial equation:

$$r = f_r(\rho) = a_0 + a_1 \times \rho^1 + a_2 \times \rho^2 + a_3 \times \rho^3 + a_4 \times \rho^4 + a_5 \times \rho^5 . \qquad (3.1)$$

To compute the desired coefficients a_0 through a_5, the following algorithm is performed.

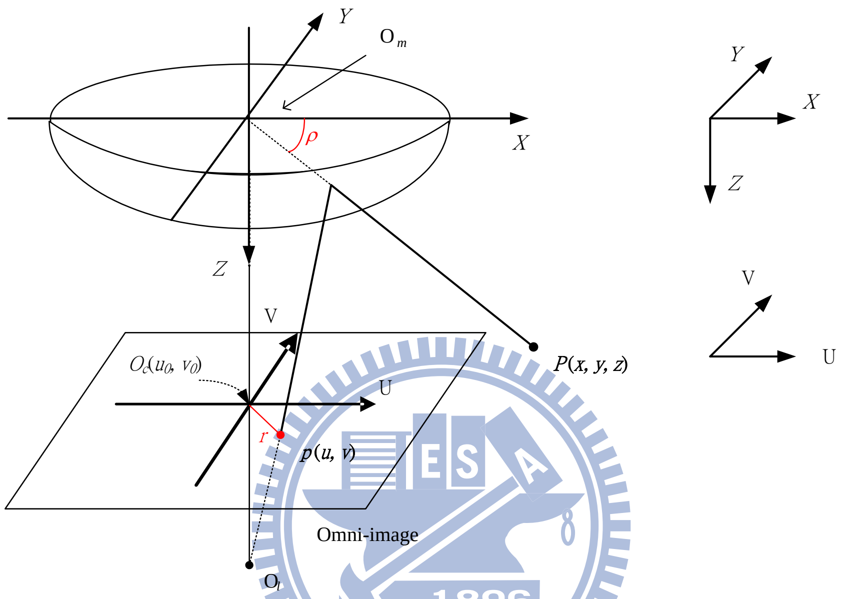


Figure 3.2 Mapping between a radius distance r and elevation angle ρ.

Algorithm 3.1 Computing the coefficients of the radial stretching function for the mirror.

Input: a set of n landmark point pairs (p_k, P_k) selected in advance for the radial stretching functions f_r, where p_k is a point in an omni-image I and P_k is the corresponding point in the world space.

Output: the six coefficients a_0 through a_5 of f_r.

Steps.

Step 1. For each selected landmark point pairs, (P_k, p_k), including world-space point P_k at coordinates (X_k, Y_k, Z_k) in the WCS and image point p_k at coordinates (u_k, v_k) in I, compute the radius r_k of p_k in I and the elevation angle ρ_k of P_k

in the WCS by the following equations:

$$r_k = \sqrt{u_k^{\ 2} + v_k^{\ 2}}; \quad \rho_k = \tan^{-1}\frac{Z_k}{\sqrt{X_k^{\ 2} + Y_k^{\ 2}}}. \tag{3.2}$$

Step 2. Substitute r_k and ρ_k into the equation for estimating f_r described by Eq. (3.1) to obtain n simultaneous equations as follows:

$$\begin{aligned}
r_0 &= f_{ri}(\rho_0) = a_0 + a_1\times\rho_0^{\ 1} + a_2\times\rho_0^{\ 2} + a_3\times\rho_0^{\ 3} + a_4\times\rho_0^{\ 4} + a_5\times\rho_0^{\ 5}\\
r_1 &= f_{ri}(\rho_1) = a_0 + a_1\times\rho_1^{\ 1} + a_2\times\rho_1^{\ 2} + a_3\times\rho_1^{\ 3} + a_4\times\rho_1^{\ 4} + a_5\times\rho_1^{\ 5}\\
&\vdots\\
r_{n-1} &= f_{ri}(\rho_{n-1}) = a_0 + a_1\times\rho_{n-1}^{\ 1} + a_2\times\rho_{n-1}^{\ 2} + a_3\times\rho_{n-1}^{\ 3} + a_4\times\rho_{n-1}^{\ 4} + a_5\times\rho_{n-1}^{\ 5}.
\end{aligned} \tag{3.3}$$

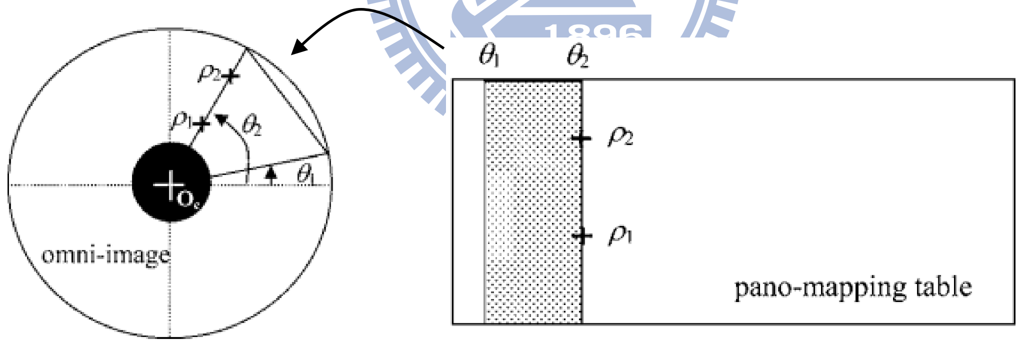
Step 3. Solve the equations in (3.3) to obtain the desired coefficients (a_0, a_1, a_2, a_3, a_4, a_5) for f_r by a numerical analysis method.

In the above process of computing the radial stretching function, it is assumed that the mirror is perfect with rotational symmetry in the entire angle range of 0^o through 360^o. However, this is not the case in real applications; the surface of the hyperboloidal-shaped mirror cannot be so perfect. To increase the accuracy of the estimated f_r and so the precision of the constructed pano-mapping table, we divided the 360^o range of azimuth angles of the mirror equally into six parts, each with 60^o, and then applied the above process to obtain six radial stretching functions f_{r1} through f_{r6} for the six parts with each f_{ri} described by the coefficients a_{0i} through a_{5i} with i = 1, 2, …, 6.

3.2.3 Filling of Pano-mapping Table Entries

The procedure of constructing the pano-mapping table with the radial stretching function for each of the six azimuth angle ranges is described here. The so-called pano-mapping table is a 2-dimensional table with its horizontal and vertical axes specifying the azimuth angle θ and the elevation angle ρ, respectively. An illustration of the mapping between the azimuth and elevation angle pair of the omni-image and the horizontal and vertical axes of the pano-mapping table, respectively, is shown in Fig. 3.3, and an example of the pano-mapping table is shown in Table 3.1.

Each entry E_{ij} with indices (i, j) in the pano-mapping table corresponds to an azimuth-elevation angle pair (θ_i, ρ_j). The azimuth-elevation pair represents an infinite set of points on a light ray with the azimuth angle θ_i and the elevation angle ρ_j with respect to the focal center O_m in the WCS. We divide the range 2π of the azimuth angles into M intervals and the range of the elevation angles between two pre-selected limits, ρ_s and ρ_e, into N intervals. Due to the property of rotational invariance of omni-imaging, the azimuth angle ϕ of the scene point P in the WCS with respect to the X-axis is identical to the azimuth angle θ of the corresponding point p in the image with respect to the u-axis. Thai is, we have $\theta = \phi$.



(a) (b)

Figure 3.3 Illustration of mapping between the azimuth-elevation angle pair of the omni-image and the horizontal and vertical axes of the pano-mapping table, respectively.

With the above estimated six sets of coefficients for the six radial stretching functions, the corresponding pano-mapping table T_{pm} can be filled with the corresponding image coordinates by the following algorithm.

Table 3.1 An example of the pano-mapping table.

	θ_1	θ_2	θ_3	θ_4	…	θ_M
ρ_1	(u_{11}, v_{11})	(u_{21}, v_{21})	(u_{31}, v_{31})	(u_{41}, v_{41})	…	(u_{M1}, v_{M1})
ρ_2	(u_{12}, v_{12})	(u_{22}, v_{22})	(u_{32}, v_{32})	(u_{42}, v_{42})	…	(u_{M2}, v_{M2})
ρ_3	(u_{13}, v_{13})	(u_{23}, v_{23})	(u_{33}, v_{33})	(u_{43}, v_{43})	…	(u_{M3}, v_{M3})
ρ_4	(u_{14}, v_{14})	(u_{24}, v_{24})	(u_{34}, v_{34})	(u_{44}, v_{44})	…	(u_{M4}, v_{M4})
…	…	…	…	…	…	…
ρ_N	(u_{1N}, v_{1N})	(u_{2N}, v_{2N})	(u_{3N}, v_{3N})	(u_{4N}, v_{4N})	…	(u_{MN}, v_{MN})

Algorithm 3.2 Construction of the pano-mapping table.

Input: six coefficient sets of six radial stretching functions f_{r1} through f_{r6} for the six parts of the azimuth angle range mentioned previously.

Output: a pano-mapping table T_{pm} of the dimension $M\times N$.

Steps.

Step 1. Divide the range 2π of the azimuth angles into M intervals, and compute the ith azimuth angle θ_i by

$$\theta_i = (2\pi/M)\times i \qquad \text{for } i = 0, 1, \ldots, M-1. \tag{3.4}$$

Step 2. Divide a pre-selected range $[\rho_s, \rho_e]$ of the elevation angle into N intervals, and compute the j-th elevation angle ρ_j by

$$\rho_j = [(\rho_e - \rho_s)/N]\times j + \rho_s \qquad \text{for } j = 0, 1, \ldots, N-1. \tag{3.5}$$

Step 3. Fill the entry of (θ_i, ρ_j) with the corresponding image coordinates (u_{ij}, v_{ij}) computed as follows:

$$u_{ij} = r_j \times \cos\theta_i, \quad v_{ij} = r_j \times \sin\theta_i, \tag{3.6}$$

where the radius r_j is computed by the radial stretching function as follows:

$$r_j = f_{ri}(\rho_j) = a_{0i}+a_{1i}\times\rho_j^{1}+a_{2i}\times\rho_j^{2}+a_{3i}\times\rho_j^{3}+a_{4i}\times\rho_j^{4}+a_{5i}\times\rho_j^{5}, \tag{3.7}$$

with the *index i* in (3.7) computed by:

$$i = \lfloor \theta_i / (\pi/3) \rfloor \tag{3.8}$$

and a_{0i} through a_{5i} being the coefficients for function f_{ri} obtained by Algorithm 3.1 for the i-th sub-range of azimuth angles mentioned previously.

3.2.4 Creation of *r*-*ρ* Mapping Table

Moreover, in this study we also construct an additional table for mapping the value of ρ to the value of r, which is called the *r*-*ρ mapping table*. It is also generated from the functions f_{r1} through f_{r6} obtained previously. The table, as shown in Table 3.2, is a simpler form of the pano-mapping table, which records only the relations between the elevation angle ρ and the corresponding radius r in the six parts of the azimuth angle range. This table may be used to accelerate the computation of point coordinate transformation for some applications, which will be described in the subsequent chapters. The detail of the generation process is described as an algorithm in the following.

Table 3.2 An example of the *r*-*ρ* mapping table.

P_1	(r_{11}, ρ_1)	(r_{21}, ρ_2)	(r_{31}, ρ_3)	(r_{41}, ρ_4)	…	(r_{N1}, ρ_N)
P_2	(r_{12}, ρ_1)	(r_{22}, ρ_2)	(r_{32}, ρ_3)	(r_{42}, ρ_4)	…	(r_{N2}, ρ_N)
…	…	…	…	…	…	…
P_6	(r_{16}, ρ_1)	(r_{26}, ρ_2)	(r_{36}, ρ_3)	(r_{46}, ρ_4)	…	(r_{N6}, ρ_N)

Algorithm 3.3 Construction of an r-ρ mapping table.

Input: six coefficient sets of six radial stretching functions f_{r1} through f_{r6} for the six parts of the azimuth angle range mentioned previously.

Output: an *r*-*ρ* mapping table $T_{r\rho}$ of dimension $6 \times N$.

Steps.

Step 1. Divide the pre-selected range $[\rho_s, \rho_e]$ of elevation angles into N intervals and compute the jth elevation angle ρ_j by

$$\rho_j = [(\rho_e - \rho_s)/N]\times j + \rho_s \qquad \text{for } j = 0, 1, ..., N-1. \qquad (3.9)$$

Step 2. Divide the azimuth angle range from 0^o through 360^o into six equal parts, P_1, P_2, …, P_6.

Step 3. Fill the entry of (P_i, ρ_j) of Table $T_{r\rho}$ with the corresponding pair (r_{ij}, ρ_j) computed as follows:

$$r_{ij} = f_{ri}(\rho_j) = a_{0i}+a_{1i}\times\rho_j^1+a_{2i}\times\rho_j^2+a_{3i}\times\rho_j^3+a_{4i}\times\rho_j^4+a_{5i}\times\rho_j^5, \qquad (3.10)$$

where i = 1, 2, …, 6 and a_{0i} through a_{5i} are the coefficients for function f_{ri} computed by Algorithm 3.1 described previously.

To utilize the table $T_{r\rho}$, once the radius distance r of a point p in the omni-image is obtained, we can determine which part P_i point p belongs to according to the azimuth angle θ of p, and search the table for all values of r and find out the one, say r_{ij}, whose corresponding ρ_j in the table is closest to r in value. Then, ρ_j is the mapping result we want. Moreover, the table may also be used for mapping the value of ρ to that of r in a reverse way, which will be described in later chapters.

3.3 Image Unwarping and Generation of Perspective-view Images

To generate a perspective-view image, a view plane which is perpendicular to the ground and in front of the camera is imagined as shown in Fig. 3.4. Every pixel on the view plane can be assigned a corresponding point in the omni-image by the use of the pano-mapping table. By finding this relationship, we are able to unwarp the omni-image into perspective-view images by getting the color value of the omni-image point to assign the value on the view plane. The generation of the perspective-view image from the omni-image with the aid of the pano-mapping table is described as follows.

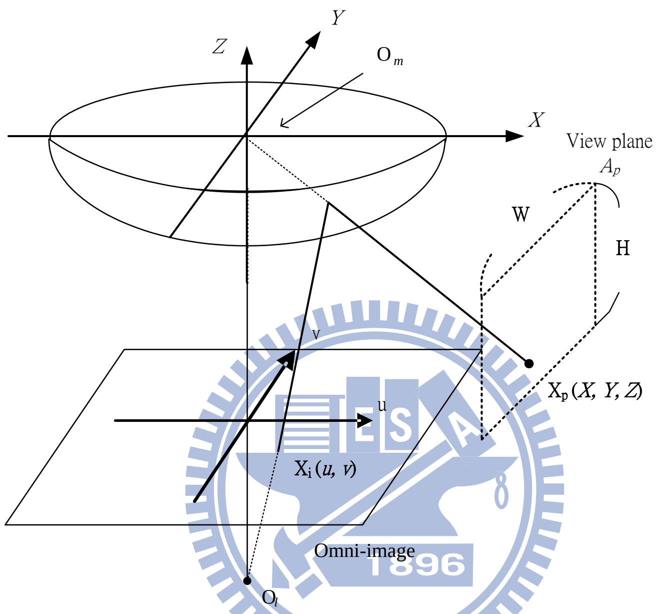


Figure 3.4 An example of generating the perspective-view image.

Algorithm 3.4 Construction of a perspective-view image.

Input: an omni-image I, the pano-mapping table T_{pm} with $M \times N$ entries, and a planar rectangular region A_{p} of size $W \times H$ at a distance D with respect to the mirror center O$_{\text{m}}$.

Output: a perspective-view image Q of any size $M_Q \times N_Q$.

Steps:

Step 1. Calculate the angle ϕ of each pixel q_{kl} at coordinates (k, l) in Q according to the following equation which is obtained from trigonometry shown in Fig. 3.5(a):

$$W^2 = D^2 + D^2 - 2 \times D \times D \times \cos\phi, \tag{3.11}$$

or equivalently,

$$\phi = \cos^{-1}(1 - \frac{W^2}{2 \times D^2}). \tag{3.12}$$

Step 2. Compute the angle β of q shown in Fig. 3.5(a) by trigonometry again:

$$\beta = \frac{(2\pi - \phi)}{2}. \tag{3.13}$$

Step 3. Compute the index i of the entry E_{ij} in table T_{pm} to find the corresponding image coordinates (u_{ij}, v_{ij}) in I for q_{kl} in the following way.

(1) Let P_{ij} denote the intersection point of the light ray R_q projected onto q_{kl} and the planar projection region A_{p} (note that each entry E_{ij} has a corresponding P_{ij}).

(2) Compute the distance d between point P_{ij} and the border point P_{r} shown in Figure 3.6(b) by linear proportionality as

$$d = W \times \frac{k}{M_Q}, \tag{3.14}$$

because the projection region A_{p} has a width of W, the image Q has a width of M_Q pixels, and pixel q_{kl} has an index of k in the horizontal direction.

(3) Compute the distance between the focal center O_{m} and the projected point P_{ij} by

$$L = \sqrt{D^2 + d^2 - 2 \times d \times D \times \cos\beta}. \tag{3.15}$$

(4) With the distance h from point P_{ij} to the line segment $\overline{\text{O}_{\text{m}} P_{\text{r}}}$ connecting O_{m} and P_{r} as shown in Fig. 3.5(b) being

$$h = d \times \sin \beta, \tag{3.16}$$

and the azimuth angle θ_q satisfying

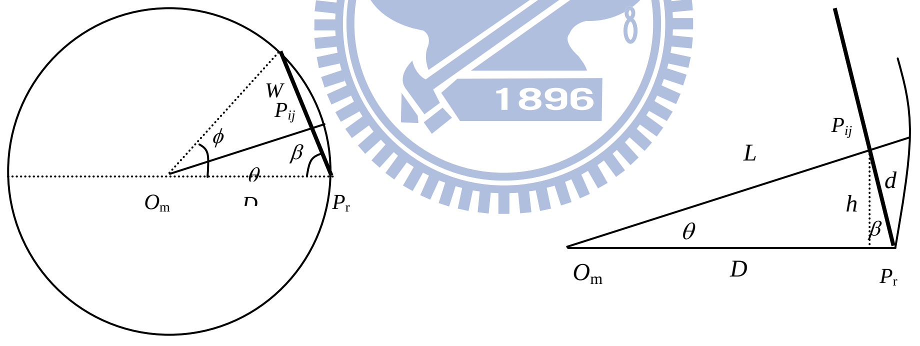
$$\sin \theta_q = \frac{h}{L} = \frac{d \times \sin \beta}{\sqrt{D^2 + d^2 - 2 \times d \times D \times \cos \beta}}, \tag{3.17}$$

compute the azimuth θ_q of point P_{ij} with respect to $\overline{\mathrm{O_m} P_\mathrm{r}}$ as

$$\theta_q = \sin^{-1} \frac{h}{L} = \sin^{-1}\left[\frac{d \times \sin \beta}{\sqrt{D^2 + d^2 - 2 \times d \times D \times \cos \beta}}\right]. \tag{3.18}$$

(5) Compute the index i of entry E_{ij} by linear proportionality as

$$i = \frac{\theta_q}{2\pi} \times M. \tag{3.19}$$



(a) (b)

Figure 3.5 A top view configuration of generating a perspective-view image.

Step 4. With a lateral view illustrating the imaging configuration shown in Fig. 3.6, compute the index j of the entry E_{ij} in table T_{pm} in the following way:

(1) With the height of A_p being H and the height of image Q being H divided into N_Q intervals, compute the height of P_{ij} by linear proportionality as:

$$H_q = l \times \frac{H}{N_Q}. \tag{3.20}$$

(2) By trigonometry, derive the elevation angle ρ_q as:

$$\rho_q = \tan^{-1}\left(\frac{H_q}{L}\right). \tag{3.21}$$

(3) Compute the index j of the E_{ij} by proportionality again as:

$$j = \frac{(\rho_q - \rho_s)\times N}{(\rho_e - \rho_s)}. \tag{3.22}$$

Step 5. With the indices (i, j) of the entry E_{ij} as computed in the last two steps, obtain image coordinates (u_{ij}, v_{ij}) by looking up Table T_{pm}.

Step 6. Assign the color value of the image pixel at coordinate (u_{ij}, v_{ij}) in image I to the pixel q_{kl} of Q at coordinates (k, l).

Step 7. After all pixels of image Q are processed, take the final image Q as the desired perspective-view image.

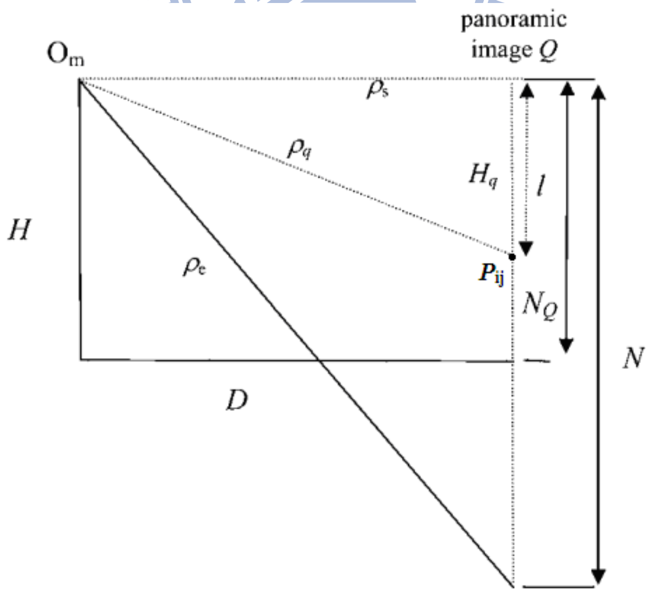


Figure 3.6 A lateral-view configuration of generating a perspective-view image.

3.4 Construction of Perspective Mapping Table for Computation Speedup

In the last section, we reviewed the method proposed by Jeng and Tsai [8] to transform an omni-image into a flat perspective-view image. However, the transformation process takes much time to compute the involved formulas. To shorten the computation time, a new table of six outward view directions is created. Specifically, the 2π range of azimuth angles are divided into portions seen from the six outward viewing directions and the interval between every two directions includes 60 degrees. See Figure 3.7 for an illustration. We denote the index of the six intervals by k which is related to the viewing direction of the perspective-view image. More specifically, we add a shift angle θ' to change the transformation range of the omni-image, and the shift angle is computed as follows:

$$\theta' = k \times \frac{2\pi}{6}. \tag{3.23}$$

Therefore, Eq. (3.17) can be rewritten as follows:

$$\theta_q = \sin^{-1}\frac{h}{L} + \theta'. \tag{3.24}$$

We transform the omni-image into the perspective view image Q_k in the six different viewing directions with index k, where k = 0, 1, …, 5. Once all pixels in Q_k are processed, we record the entries of all pixels in the table and denote the entries as T_k. As a result, the new table T_{pp}, called *perspective mapping table*, is created, which contains six sets T_0 through T_5 of perspective mapping entries to map pixels from the omni-image to the perspective-view image. When we want to generate a perspective-view image, it can be generated immediately by looking up the table T_{pp}.

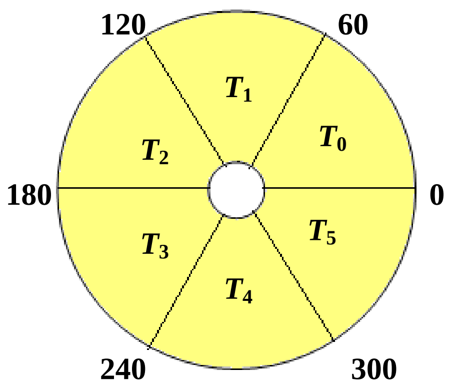


Figure 3.7 A top view of segmenting an omni-image.

Chapter 4
Car-driving Assistance by Analyzing Omni-images of Surrounding Environment

4.1 Idea of Proposed Method

While driving the video surveillance vehicle, we want to monitor the surrounding environment for driving assistance. Owing to the wide FOV of the omni-camera and the affixed positions of the pair of two-camera omni-directional imaging devices on the vehicle roof, the monitored range of the camera system covers the entire car surround. Besides, the omni-images acquired with the omni-camera system may be used for producing panoramic images and estimating the relevant stereo information of surrounding objects. In this study, we develop two applications using the camera system for environment monitoring.

One application is to use the proposed system to provide the driver a perspective-view image corresponding to the moving direction of the video surveillance vehicle, which is useful for inspection of the possible bind spots around the surveillance vehicle in order to avoid car accidents. Another application is using the proposed system as a *driving recorder* which may be used to record the surrounding environment images in the *driving history*, and to allow the user to see the perspective-view image sequence in any selected view direction which is constructed in an off-line fashion from the acquired sequential omni-images.

4.2 Analysis of Car Direction by Motion Vectors in Omni-images

4.2.1 Idea of car direction analysis by motion vectors

When driving the video surveillance vehicle, it is desired to analyze the motions in the consecutively acquired omni-images to determine the vehicle moving direction, and generate and display the corresponding perspective-view image assist the driver to observe blind spots around the vehicle. We apply the optical flow analysis method to implement this idea in this study. We produce motion vectors on the consecutively acquired images and analyze these motion vectors to estimate the vehicle moving direction. We divide the vehicle direction estimation work into six steps: (1) select the detection region (2) compute the motion vectors; (3) transform these vectors from the ICS to the WCS; (4) eliminate the outliers of these vectors; (5) estimate the moving direction; and (6) display of the corresponding perspective-view image. These steps will be introduced in the following section.

In this study, we use two laptops as the control units to handle the above tasks. More specifically, one of the laptops is responsible for analyzing the moving direction of the video surveillance vehicle, and the analyzed result is sent to another computer to generate the corresponding perspective-view image. To generate perspective-view images quickly, we use the perspective mapping table introduced in Chapter 3 to speed up the computation.

4.2.2 Car Direction Detection and Display of Corresponding Perspective-view Images

In this section, we describe the proposed technique to estimate the moving direction of the video surveillance vehicle. The involved six major steps mentioned previously are described in order subsequently.

A. Selection of the detection region for optical flow analysis

To analyze the omni-image acquired with the omni-camera system, it is required to choose a *detection region* in the image and apply the optical flow analysis method in the region to estimate the motion vectors. The selection of the detection region is divided into three cases, as illustrated in Fig. 4.1:

(1) when the video surveillance vehicle is turning to the right, we select the right-front region of the omni-image as the detect region;

(2) on the contrary, when turning to the left, the left-front region is selected; and

(3) if the video surveillance vehicle is moving forwarding, the front region will be selected.

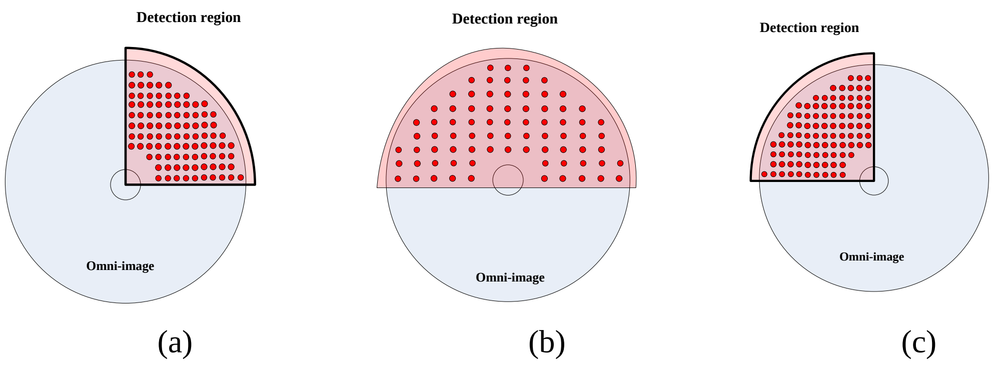


Figure 4.1 Illustration of selecting the detection region where the red points represent the spots on which optical flows need be found. (a) Detection region used in the case of turning to the right. (b) Detection region used in the case of moving forward. (c) Detection region used in the case of turning to the left.

The optical flow pattern is different in each different motion case, and each pattern corresponds to a type of motion of the video surveillance vehicle (turning to the left, turning to the right, or moving forward), as illustrated in Fig. 4.2. Therefore, we cannot analyze the motion vectors in an unchanged detection region all the time to estimate the vehicle moving direction; otherwise, the optical flow pattern in the case of the left turn and the right turn might result in wrong analysis results about the vehicle moving direction. To solve the problem, we let the detection region be changed dynamically in accordance with the previous moving direction detection result, as illustrated in Fig. 4.2.

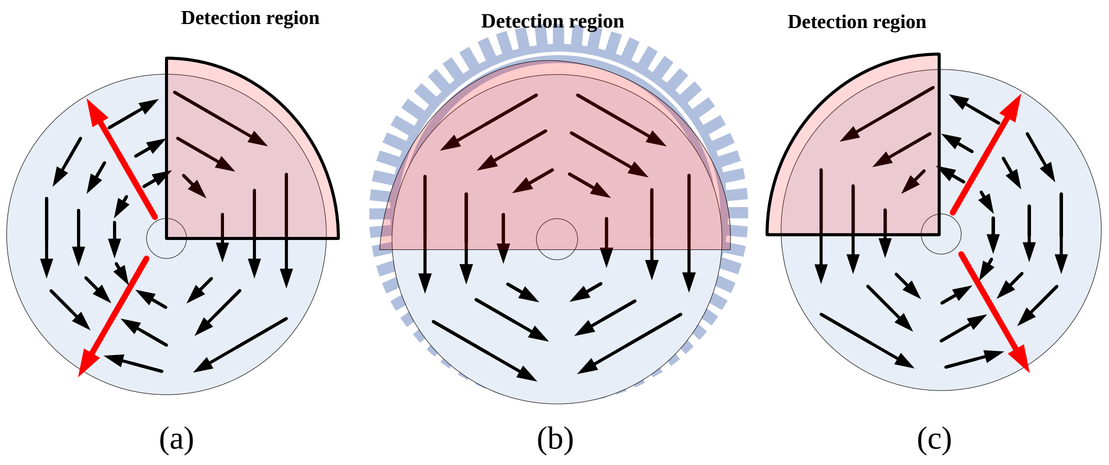


Figure 4.2 The optical flow pattern and the corresponding detection region. (a) The case of turning to the right. (b) The case of moving forward. (c) The case of turning to the left.

B. *Estimation of motion vectors by optical flows*

The optical flow analysis method may be used to estimate the motion vectors of objects, surfaces, and edges caused by the relative motions between two consecutive images. Assume that the light is stable and the displacements of concerned objects in the image are small. Under such conditions, the motion vectors between two consecutive image frames which are taken at times t and $t + dt$ can be estimated by the optical flow analysis method in the following way. If the image intensity is *continuous* and can be differentiated, the image intensity at time instant t is constrained by

$$I(x, y, t) = I(x + dx, y + dy, t + dt), \tag{4.1}$$

where the function I is the image intensity, x and y specify the location of the point in the image, and t is the sampling time. The image constraint at $I(x + dx, y + dy, t + dt)$ in Equation (4.1) can be expressed as a truncated Taylor series in the following way:

$$I(x+dx,\ y+dy,\ t+dt) = I(x,\ y,\ t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt\ . \tag{4.2}$$

From Eqs. (4.1) and (4.2), it follows that:

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t}\frac{dt}{dt} = 0,$$

or equivalently, that

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0, \tag{4.3}$$

where $V_x = dx/dt$ and $V_y = dy/dt$ represent the *velocity or optical flow* of $I(x, y, t)$ and $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, and $\frac{\partial I}{\partial t}$ are the derivatives of point p at coordinates (x, y). Therefore, Eq. (4.3) may be rewritten as:

$$I_x(p)V_x + I_y(p)V_y = -I_t(p) \tag{4.4}$$

where $I_x(p)$, $I_y(p)$, and $I_t(p)$ are equal to $\partial I/\partial x$, $\partial I/\partial y$, and $\partial I/\partial t$, respectively, all of point p at coordinates (x, y).

However, Eq. (4.4) derived from a single point p has two unknowns V_x and V_y, and cannot be solved uniquely using the data of the single point p. This is a traditional optical flow problem called “the aperture problem.” In order to solve this problem, the

Lucas-Kanade method [14] is adopted. The Lucas-Kanade method divides an image into small regions and assumes that the displacements of the image content within a small neighborhood of the concerned point p are small and approximately constant. Accordingly, we may set a window around point p with n pixels, p_1, p_2, ..., p_n inside the window. Then, the local image motion vector (V_x, V_y) at p with image coordinates (x, y) must satisfy the following equations according to Eq. (4.4):

$$\begin{aligned} I_x(p_1)V_x + I_y(p_1)V_y &= -I_t(p_1), \\ I_x(p_2)V_x + I_y(p_2)V_y &= -I_t(p_2), \\ &\vdots \\ I_x(p_n)V_x + I_y(p_n)V_y &= -I_t(p_n). \end{aligned} \tag{4.5}$$

Eqs. (4.5) can be expressed in a matrix form:

$$Av = b,$$

where

$$A = \begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \\ I_x(p_n) & I_y(p_n) \end{bmatrix}, \quad v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, \quad and \quad b = \begin{bmatrix} -I_t(p_1) \\ -I_t(p_2) \\ \vdots \\ -I_t(p_n) \end{bmatrix}. \tag{4.6}$$

A solution derived by the least square principle to solve the above matrix equation is:

$$A^T Av = A^T b,$$

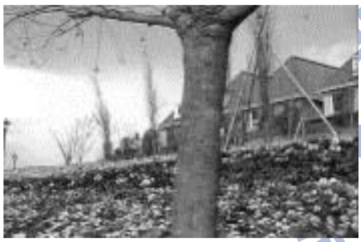
or equivalently,

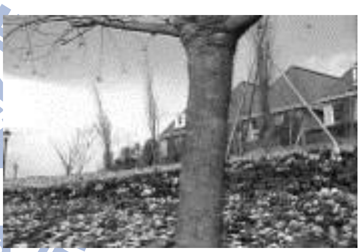
$$v = (A^T A)^{-1} A^T b. \tag{4.7}$$

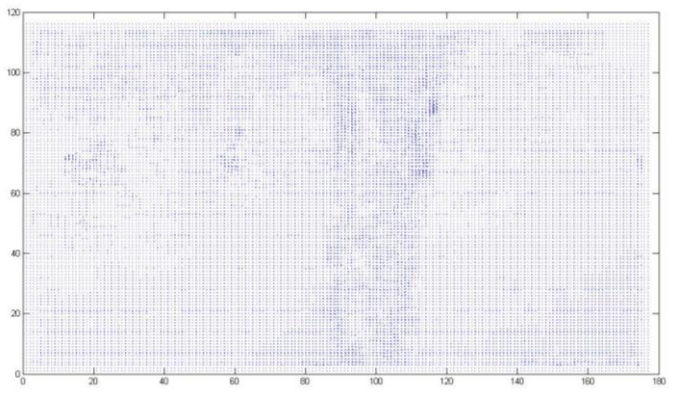
Accordingly, we can estimate the motion vectors of consecutively acquired images using Eq. (4.7) above. An example of the results of applying the optical flow analysis method is shown in Fig. 4.3.

C. *Transformation of motion vectors*

To estimate the moving direction of the video surveillance vehicle, we have to analyze these motion vectors produced by the optical flow analysis method. However, the images captured with the omni-cameras are distorted due to the light reflection on the hyperboloidal-shaped mirrors in the omni-camera system. As a result, before computing the direction angle of these motion vectors, the transformation of the motion vectors from the omni-image plane to the world coordinate system as shown in Fig. 4.4(b) is necessary. The configuration of such a transformation of the motion vector of a real-world point on the ground is shown in Fig. 4.4(a). We divide the transformation process into three steps as described in the following algorithm.


(a)


(b)



(c)

Figure 4.3 An example of results of implementing the optical flow analysis method. (a) An image frame taken at time *t*. (b) An image frame taken at time *t* + *dt*. (c) The result of the motion vectors produced by the optical flow analysis method with (a) and (b) as inputs.

Algorithm 4.1 Transformation of a motion vector from the ICS to the WCS.

Input: the beginning point P_s at image coordinates (u, v) in an image frame I_t and the ending point P_e at image coordinates (u', v') in the next image frame I_{t+1}, both of the motion vector V_i of a real-world point P on the ground, and the r-ρ mapping table $T_{r\rho}$.

Output: the directional angle of the motion vector with respect to the X-axis in the WCS.

Steps.

Step 1. Compute the elevation angle ρ_1 and the azimuth angle θ_1 of the beginning point P_s and the ending point P_e in the ICS by the following way.

1.1 Compute the azimuth angle θ_1 of point P_s at image coordinates (u, v) by

$$r = \sqrt{u^2 + v^2}; \quad \theta_1 = \sin^{-1}\frac{v}{r} = \cos^{-1}\frac{u}{r}. \tag{4.8}$$

1.2 Look up the r-ρ mapping table $T_{r\rho}$ to obtain the elevation angle ρ_1 with the radius distance r.

1.3 Compute θ_2 and ρ_2 of the ending point P_e in a similar way.

Step 2. Transform the image coordinate (u, v) of point P_s in image I_t to world coordinates (X, Y, Z) of point P by the following way.

2.1 Compute the horizontal distance d_w between P and the focal center of the mirror O_m by

$$d_w = H_m \times \cot(\rho_1), \tag{4.9}$$

where the distance between the mirror center and ground is known to be H_m.

2.2 Compute as follows the world coordinates (X, Y, Z) of point P_s according to the property of rotational invariance of omni-imaging which says that the azimuth angle ϕ of point P in the WCS with respect to the X-axis is identical to the azimuth angle θ of P_s in the ICS :

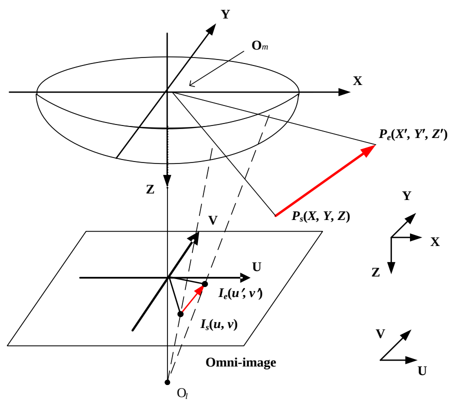
$$\begin{aligned} X &= d_w \times \cos\theta_1; \\ Y &= d_w \times \sin\theta_1; \\ Z &= H_m. \end{aligned} \tag{4.10}$$

2.3 Transform the image coordinates (u', v') of the ending point P_e in image I_{t+1} to the world coordinates (X', Y', Z') of point P in a similar way.

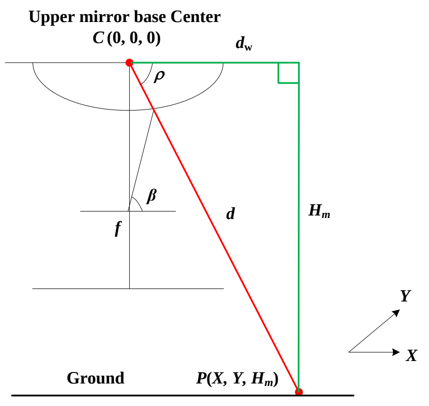
Step 3. Compute the directional angle A_i of the motion vector V_i with respect to the X-axis by

$$\begin{aligned} A_i &= \sin^{-1}\left(\frac{Y'-Y}{\sqrt{(Y'-Y)^2+(X'-X)^2}}\right); \\ &= \cos^{-1}\left(\frac{X'-X}{\sqrt{(Y'-Y)^2+(X'-X)^2}}\right). \end{aligned} \tag{4.11}$$

By Algorithm 4.1, we can transform all the motion vectors produced by the optical flow analysis method into the WCS and get all directional angles of the motion vectors for analyzing the moving direction of the video surveillance vehicle.



(a)



(b)

Figure 4.4 Transformation of a motion vector from the ICS to the WCS. (a) An illustration of the camera system and the motion vector. (b) The ray tracing of a scene point P on the ground projected on the hyperboloidal-shaped mirror.

D. *Elimination of Outlier*

For analysis of the vehicle moving direction, only motion vectors with lengths be larger than a threshold value *TH* need be considered. The reason is that the shake of the omni-cameras due to rough road conditions or the shake of the car engine might create short-length motion vector which should be considered as noise and eliminated to increase the accuracy of the vehicle moving direction estimation result.

To eliminate the outlier of the motion vectors, each directional angle of the remaining motion vectors is regarded as a feature and the standard deviation value is computed accordingly, as shown in Fig. 4.5. More specifically, let the angles of these motion vectors be denoted as A_i, and let the total number of motion vectors be denoted as n. Then, the mean value $\overline{A}$ of these motion vectors may be computed as follows:

$$\overline{A} = \frac{1}{n}\sum_{i=1}^{n} A_i. \tag{4.11}$$

Once the mean value is computed, we can calculate the standard deviation value S_{n} of the motion vector data as follows:

$$S_n = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(A_i - \overline{A})^2}. \tag{4.12}$$

If the value A_i of a certain motion vector lies outside the range $[-S_n + \overline{A}, \overline{A} + S_n]$ set by the standard deviation S_n, we will regard it as an outlier and discard it. After all the outliers are eliminated, we compute the mean value of the remaining data as the desired directional angle of the moving direction of the video surveillance vehicle.

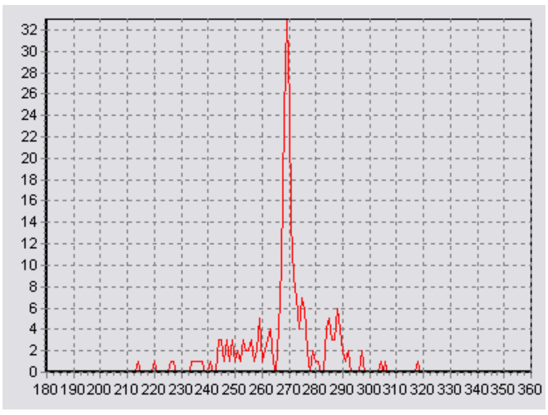


Figure 4.5 A distribution chart of the direction angle of motion vectors.

E. Estimation of Moving Direction

The moving directions of the video surveillance vehicle may be categorized into three classes — turning to the right, turning to the left, and moving forward. And the ranges of the directional angles of the three classes are determined by our experimental experiences. They are listed in Table 4.1, which may be used to classify the results of the directional angles derived by Eq. (4.11) into the three vehicle moving directions.

Table 4.1 The range of the angles of the three vehicle moving directions.

State	Degree
Moving forward	261° ~ 279°
Turn to the left	180° ~260°
Turn to the right	280° ~ 360°

Besides, we use the concept of *finite state machine* (FSM) to determine the moving direction of the video surveillance vehicle. The finite state machine designed for use in this study and illustrated in Fig. 4.6 is composed of six states, which can be categorized into three classes: (1) turning to the right; (2) turn to the left; and (3) moving forward. Specifically, the video surveillance vehicle, in one of the states, goes through a transition to another state depending on the input which is the analysis result of the moving direction. If the analysis result for the next cycle is identical to that of the current cycle in the FSM, the input to the FSM is taken to be “1”; else, to be “0.” Also, we propose in this study a concept of *giving a second chance for the state-changing check, i.e.*, if the moving direction of the next cycle is analyzed to be different from that of the current cycle, then one more check is allowed, as can be seen in the FSM illustrated in Fig. 4.6. With this *second-chance check scheme* included in the FSM, the probability of erroneous estimations of the vehicle moving direction may be decreased. Note that we take the current state in the FSM as the moving direction of the video surveillance vehicle in the current cycle.

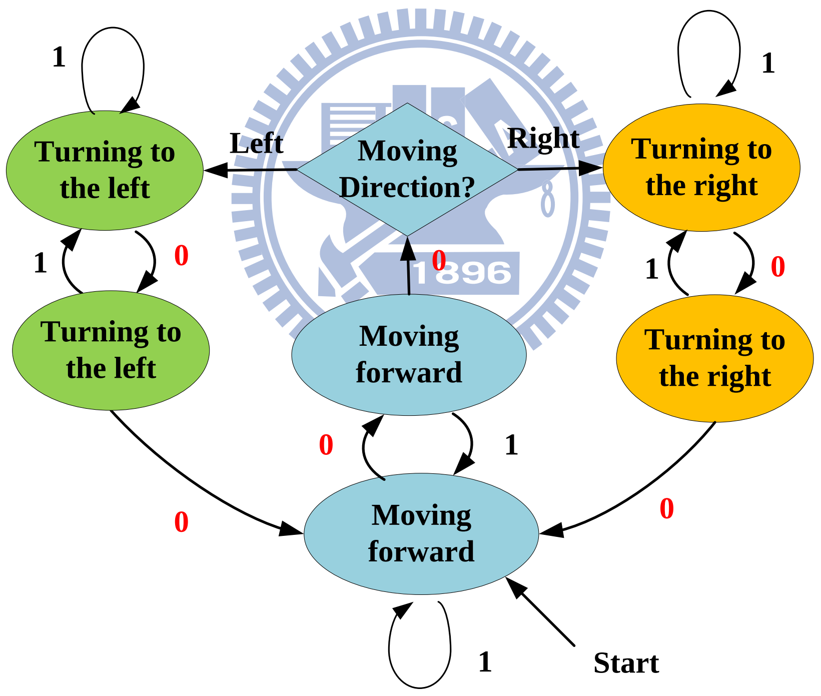


Figure 4.6 A graph of finite state machine proposed to determine the moving direction.

F. *Display of Corresponding Perspective-view Image*

To provide the driver with the views of the blind spots around the vehicle which are often neglected, the system automatically generates and displays the corresponding perspective-view image after analyzing the moving direction of the video surveillance vehicle. To implement these tasks, a local network is set up as mentioned previously, which integrates two laptops for between-computer communication and the structure is illustrated in Fig. 4.7. Computer COM_A analyzes the omni-images acquired with the right-front camera on the vehicle roof to estimate the moving direction and sends the results to computer COM_B. Computer COM_B receives commands from the driver and starts the process of constructing perspective-view images. To speed up generation of perspective-view images, the program is designed to look up the perspective-mapping tables T_{pp} introduced in Chapter 3 for shortening the computation time.

Also, some rules have been designed in this study for constructing and displaying the corresponding perspective-view image, as described in the following:

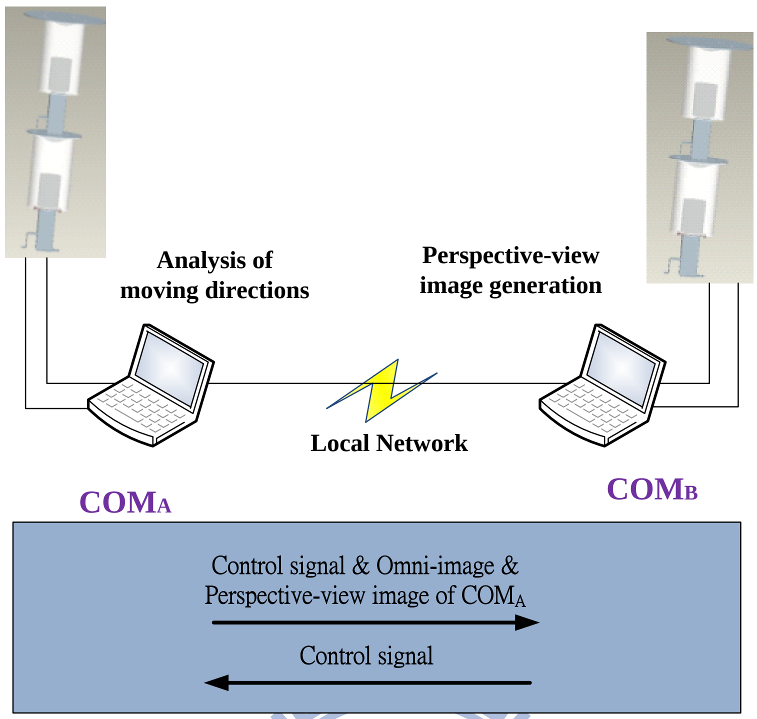
(1) when the analysis result of the moving direction is "turning to the right," construct and display the perspective-view image of the right-rear view of the video surveillance vehicle; or

(2) if the analysis result is "turning to the left," then construct and display the perspective-view image of the left-rear view of the vehicle;

(3) otherwise, decide the vehicle to be "moving forward," and construct and display the perspective-view image of the rear area of the vehicle.

4.2.3 Algorithm

We propose a method of analyzing the omni-images of the surrounding environment for the purpose of providing the driver corresponding perspective-view images to inspect the views of the blind spots. The detail is described as an algorithm below.


Analysis of moving directions
Perspective-view image generation
Local Network
COMA
COMB
Control signal & Omni-image & Perspective-view image of COMA
Control signal
Figure 4.7 Structure of the communication between two laptops used in this study.

Algorithm 4.2: car direction detection and display of corresponding perspective-view images.

Input: the consecutive omni-images acquired with the upper omni-cameras.

Output: the moving direction of the video surveillance vehicle and the corresponding perspective-view image.

Steps.

Step 1. Initialize the detection region to be the front region of the omni-image, as illustrated in Fig.4.9.

Step 2. Create an image buffer to keep the previous omni-image for optical flow analysis.

Step 3. Select a detection region by the previous detection result of the vehicle moving direction by the technique described previously in Section 4.2.2.*A*.

Step 4. Apply the optical flow analysis technique reviewed in Section 4.2.2.*B* previously on the computer COM_A to produce the motion vectors in the consecutive images.

Step 5. Transform all the motion vectors into the WCS by the technique described in Section 4.2.2.*C* and record all the directional angles in a buffer *B* for further analysis.

Step 6. Adopt the statistical method introduced in Section 4.2.2.*D* to estimate the directional angle using the data stored in buffer *B* to increase the accuracy of the estimated direction angle.

Step 7. Get the result of the directional angle and determine the moving direction of the video surveillance vehicle by the technique described in Section 4.2.2.*E*.

Step 8. Send the result of the moving direction to another computer COM_B.

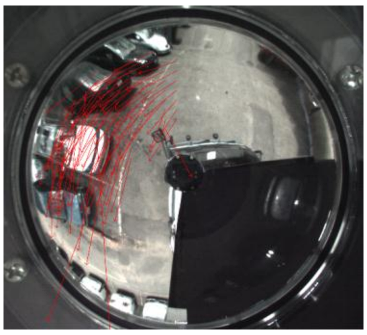
Step 9. Generate and display the perspective-view image corresponding to the determined vehicle moving direction on the computer COM_B based on the rules described in Section 4.2.2.*F*, and go to Step 2 to repeat the process again.

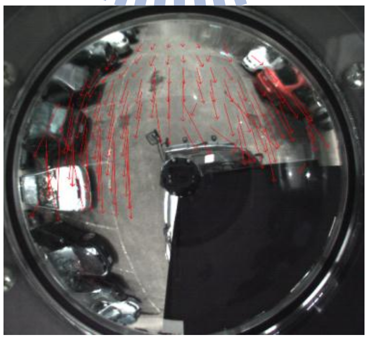
An example of the experimental result of detecting the direction of the video surveillance vehicle and displaying the corresponding perspective-view images is shown in Fig 4.8. The corresponding perspective-view image can aid the driver to inspect the views of the blind spots around the vehicle dynamically. This greatly enhances the driving safety.

4.3 Sequential Driving Recording for Off-line Inspection of Driving History

4.3.1 Idea

During driving the video surveillance vehicle, the program continuously records the sequential omni-images of the driving history. The wide FOV of images acquired with a pair of two-camera omni-directional devices equipped on the vehicle roof covers the whole surround of the video surveillance vehicle. As a result, the user can watch and understand the environment on every surrounding position of the video surveillance vehicle. Moreover, through the technique of real-time transformation from the omni-image into the perspective-view image, the user can see perspective-view image sequence in any view direction.


(a)


(b)

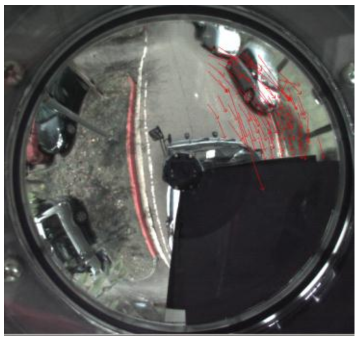

(c)

Figure 4.8 An example of results of optical flow analysis on omni-images and corresponding perspective-view images, where the red arrowheads represent motion vectors. (a) Optical flows of "turning to the left." (b) Optical flow of "moving forward." (c) Optical flow of "turning to the right." (d) ~ (e) Corresponding perspective-view images of (a) ~ (c), respectively.

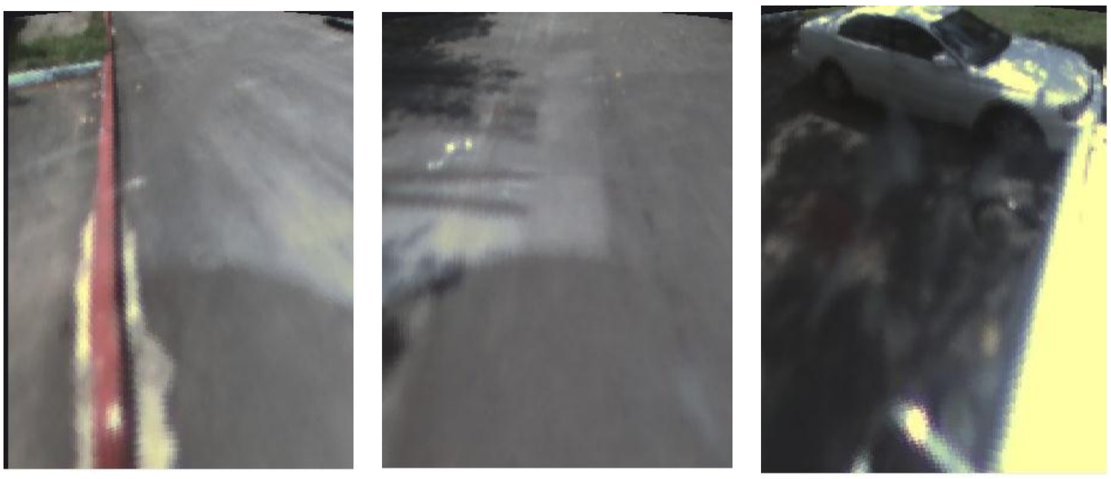

(d) (e) (f)

Figure 4.8 An example of results of optical flow analysis on omni-images and corresponding perspective-view images, where the red arrowheads represent motion vectors (continue). (a) Optical flows of "turning to the left." (b) Optical flow of "moving forward." (c) Optical flow of "turning to the right." (d) ~ (e) Corresponding perspective-view images of (a) ~ (c), respectively.

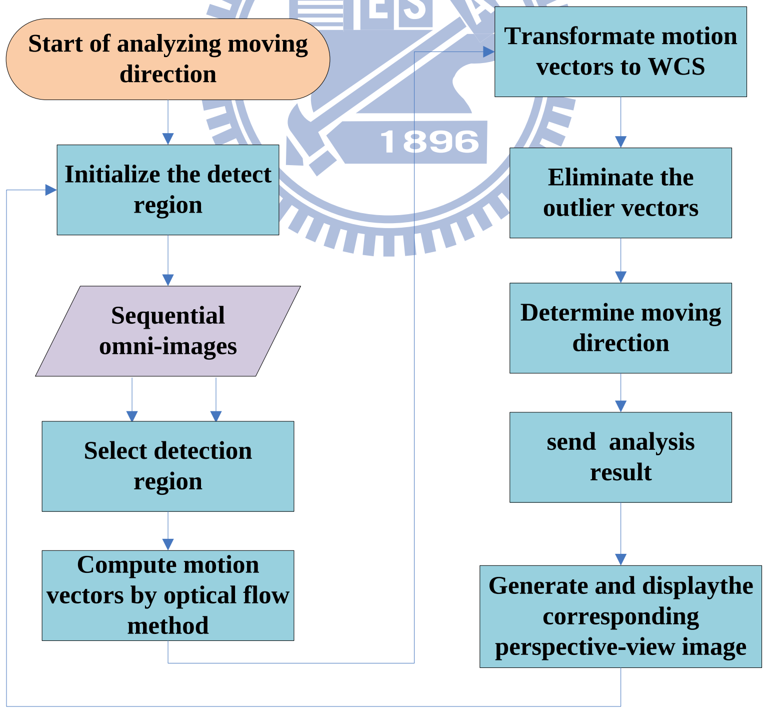

Start of analyzing moving direction
Initialize the detect region
Sequential omni-images
Select detection region
Compute motion vectors by optical flow method
Transformate motion vectors to WCS
Eliminate the outlier vectors
Determine moving direction
send analysis result
Generate and displaythe corresponding perspective-view image

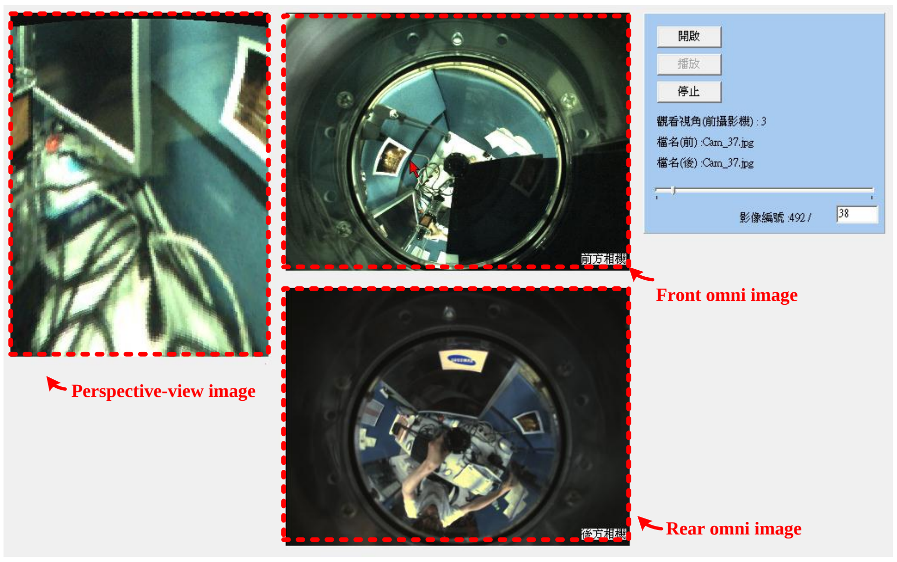
Figure 4.9 The car-driving assistance by analyzing omni-images of the surrounding environment.

4.3.2 Inspection of Sequential Driving Record via Perspective-view Image

In this section, we describe the proposed techniques to record and inspect the image sequence of the driving history. The detail of each technique is described as follows.

A. To achieve the synchronization of recording the driving history, we let laptop computer COM_B save the omni-image captured with camera system B and simultaneously send a signal to trigger computer COM_A for saving the omni-image from camera system A. Besides, in order to save the storage space and accelerate the image transmission between the two computers, the recorded images are stored as JPEG files and named in serial numbers for sequential transmission. The image files of the driving history are stored in each computer and, all image files in computer COM_A will be sent to computer COM_B through the local network when the user wants to inspect the sequential images off-line.

B. For inspecting the driving history in an off-line fashion, computer COM_B needs to load the image files and displays the down-sampled omni-images in sequence. To generate the perspective-view image in real time, computer COM_B needs to load two perspective-mapping tables in advance, one for generating perspective-view images for camera system A and another for camera system B. Moreover, we develop an interface as shown in Fig. 4.10 to let the user change the view direction of the perspective-view image by moving a mouse. Hence, the user can use a mouse to choose any view direction conveniently to observe the scene which he/she is concerned with.



觀看視角(前攝影機) : 3
檔名(前) :Cam_37.jpg
檔名(後) :Cam_37.jpg
影像編號 :492 / 38

Figure 4.10 An interface for inspecting the sequential driving record.

4.3.3 Algorithm

The following algorithm introduces the proposed of inspecting the sequential driving recording for inspection in an off-line fashion as shown in Fig. 4.12.

Algorithm 4.3 A method of inspecting the sequential driving recording of driving history.

Input: omni-images and two pano-mapping tables T_{pm} of the camera system *A* and the camera system *B*.

Output: the perspective-view image and the down-sampling omni-image sequences.

Steps.

Step 1. Record the image and transfer the image files by the technique described in Section 4.3.2.*A*.

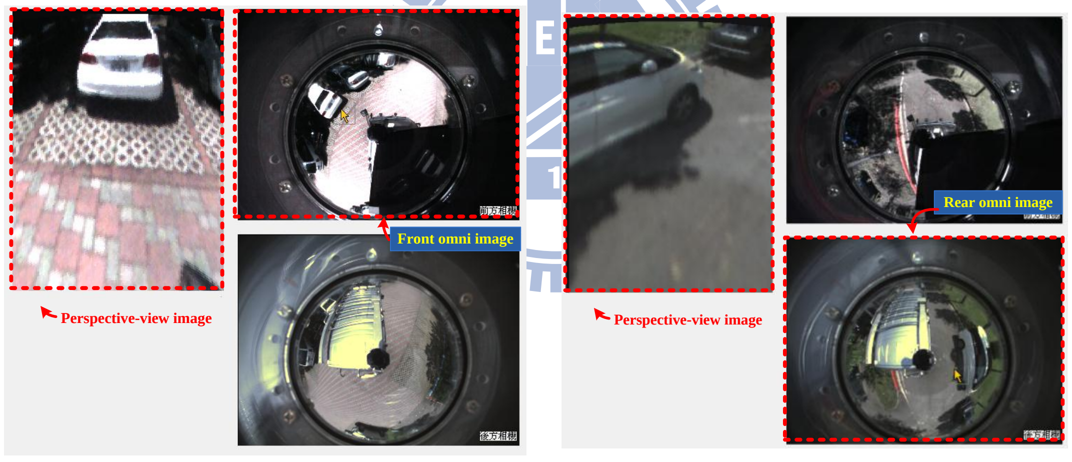
Step 2. Load two perspective mapping tables for the perspective-view image generation process.

Step 3. Read the omni-images from a directory and generate the perspective-view images of the selected view direction by the corresponding perspective mapping table by the technique described in Section 4.3.2.*B*.

Step 4. Display the perspective-view image.

Step 5. Go to Step 3 and repeat until all the image files have been read.

In Figure 4.11, we show an example of the driving history which is sequentially displayed as perspective images for inspection, after the user clicks on the down-sampled omni-images. With a driving recorder functioning like this, the user is able to observe every surrounding position of the video surveillance vehicle.



(a) (b)

Figure 4.11 The result of inspecting the driving history. (a) The omni-image and the perspective-view image obtained from transforming the omni-image acquired with the right-front camera. (b) The omni-image and the perspective-view image obtained from transforming the omni-image acquired with the left-rear camera.

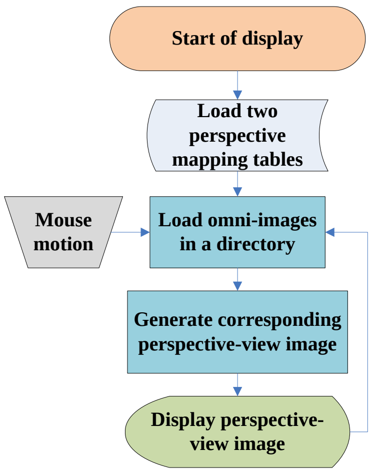

Start of display
Load two perspective mapping tables
Mouse motion
Load omni-images in a directory
Generate corresponding perspective-view image
Display perspective-view image

Figure 4.12 A flowchart of sequential driving recording for off-line inspection.

Chapter 5
Monitoring of a Nearby Static Car around a Static Video Surveillance Vehicle

5.1 Idea of Static Car Detection in Omni-images

In this chapter, we describe the proposed method for detecting a static nearby car in the omni-image around a video surveillance vehicle, and that for constructing the top-view surround map including the nearby car. Specifically, the nearby static car is detected from the omni-image by ground elimination. Then, the 3D data of the vehicle edge points are estimated. Finally, the relative position of the detected car with respect to the video surveillance vehicle is computed, and the surround map from the top view generated. The proposed method is divided into two major stages and a flow chart of the method is shown in Fig. 5.1.

In the first stage, the process of car shape extraction consists of three major steps: (1) ground learning; (2) determination of the threshold value by moment-preserving thresholding proposed by Tsai [21]; and (3) noise elimination by region growing. The second stage is the process of estimating the 3D data of the detected car and generating the surround map. This stage includes three major steps as well: (1) extraction of the corresponding edge point pairs; (2) estimation of the 3D data of the point pairs; and (3) generation of the surround map. These steps of the two stages will be introduced in detail in the following sections.

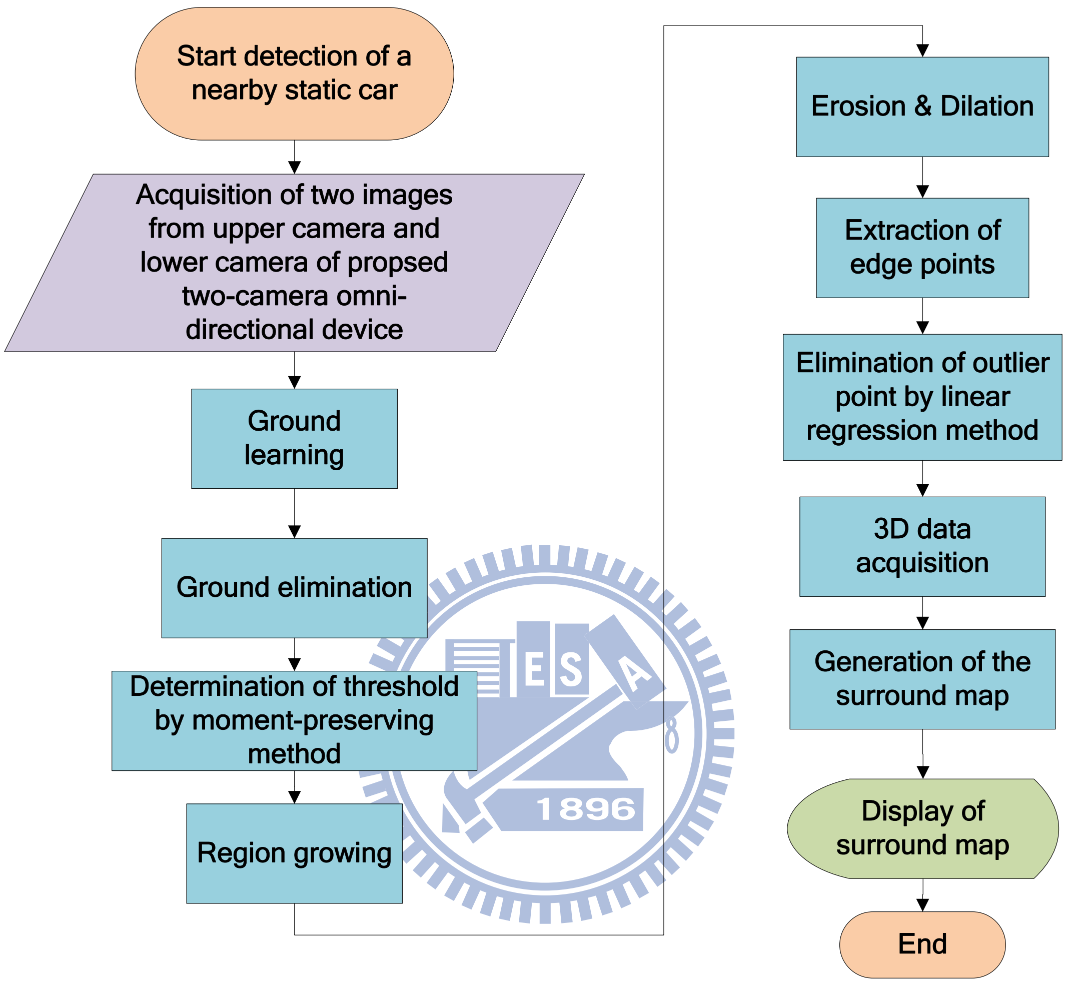

Start detection of a nearby static car
Acquisition of two images from upper camera and lower camera of propsed two-camera omni-directional device
Ground learning
Ground elimination
Determination of threshold by moment-preserving method
Region growing
Erosion & Dilation
Extraction of edge points
Elimination of outlier point by linear regression method
3D data acquisition
Generation of the surround map
Display of surround map
End

Figure 5.1 A flow chart of static car detection with a static video surveillance vehicle.

5.2 Nearby Vehicle Detection

5.2.1 Ground Region Learning

At the beginning of group region learning, we acquire two omni-images with the lower and upper cameras of each of the two 2-camera omni-directional devices. Then, we perform the following process with the images as input. The first step is transformation of each original omni-image I_o into a grayscale omni-image I_g. By this step, we learn a mean gray value of the ground region in I_g for ground elimination. This is accomplished by selecting automatically an initial region near the tire position of the video surveillance vehicle, because the region is probably part of the ground. Alternatively, we also allow the user to select a region of the ground manually as shown in Fig 5.2.

After selecting the ground region, let the total pixel number of the region be denoted by n and the gray value of each pixel in this region by $I_g(u, v)$. The mean gray value g_m of the learned region is computed according to the following equation:

$$g_m = \frac{1}{n}\sum_u \sum_v I_g(u, v). \tag{5.1}$$

A difference image f then is generated by subtracting the mean gray value g_m from the gray value of each pixel in I_o.

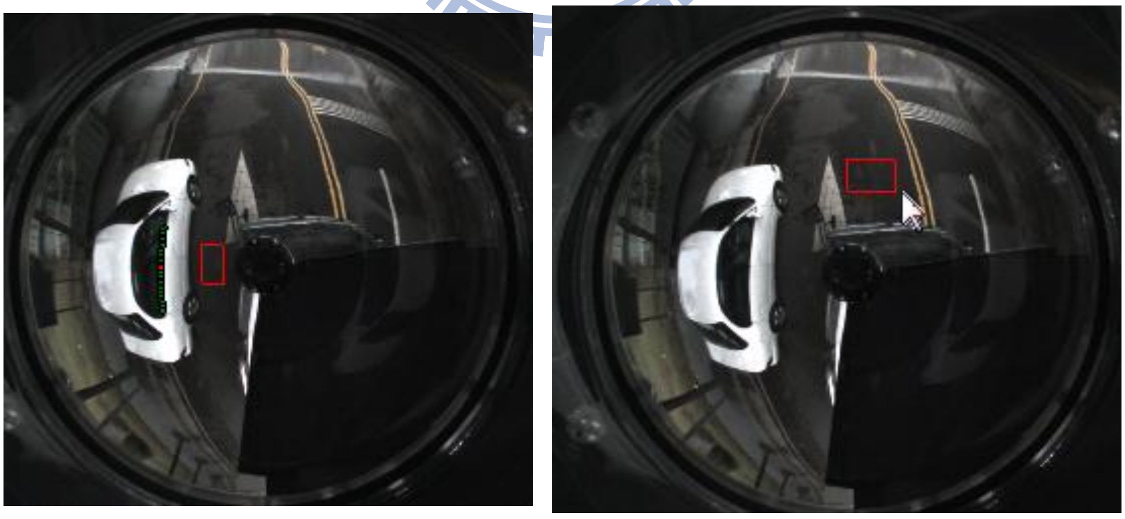

(a) (b)

Figure 5.2 The interface for ground learning. (a) An example of initializing the region of the ground. (b) An example of selecting the ground region by a user.

5.2.2 Object Segmentation by Moment-preserving Thresholding

If we can separate the background and the foreground in an image, the subsequent image analysis process will become simpler and easier. For this, we segment the car region out from the omni-image by thresholding the difference image *f* into a bi-level image by the use of a threshold value *TH*, with the car region labeled by "1" and the other region by "0." The moment-preserving thresholding method proposed by Tsai [21] is used here to decide the threshold value *TH* automatically. It is reviewed subsequently.

Given an image *f* with *n* pixels whose gray value at a pixel with coordinates (*x*, *y*) is denoted by *f*(*x*, *y*), the *i*-th moment m_i of *f* is defined as

$$m_i = \frac{1}{n}\sum_x \sum_y f^i(x, y), \quad i = 0, 1, 2, 3. \tag{5.2}$$

The moments also can be computed by the use of the gray-level histogram in the following way, where n_j is the total number of pixels in *f* with gray value z_j and $p_j = n_j/n$:

$$m_i' = \frac{1}{n}\sum_{j=0}^{1} n_j (z_j)^i = \sum_{j=0}^{1} p_j (z_j)^i, \quad i = 0, 1, 2, 3. \tag{5.3}$$

Assume that the image resulting from thresholding only contains two gray values z_0 and z_1, with z_1 is larger than z_0. A pixel value in the image greater than the threshold value to be found is replaced by z_1; on the contrary, a value smaller than the threshold is replaced as z_0. To find the desired probability values of p_0 and p_1, Eq. (5.3) can be solved to get the following equations:

$$c_d = \begin{vmatrix} m_0 & m_1 \\ m_1 & m_2 \end{vmatrix}; \quad c_0 = \frac{1}{c_d}\begin{vmatrix} -m_2 & m_1 \\ -m_3 & m_2 \end{vmatrix}; \quad c_1 = \frac{1}{c_d}\begin{vmatrix} m_0 & -m_2 \\ m_1 & -m_3 \end{vmatrix};$$

$$z_0 = \frac{1}{2}\left[-c_1 - (c_1^2 - 4c_0)^{\frac{1}{2}}\right]; \; z_1 = \frac{1}{2}\left[-c_1 + (c_1^2 - 4c_0)^{\frac{1}{2}}\right]; \tag{5.4}$$

$$p_d = \begin{vmatrix} 1 & 1 \\ z_0 & z_1 \end{vmatrix}; \quad p_0 = \frac{1}{p_d}\begin{vmatrix} 1 & 1 \\ m_1 & z_1 \end{vmatrix}; \quad p_1 = 1 - p_0.$$

To obtain the threshold value *TH*, we need to accumulate the probability values from the smallest gray value until the accumulated value reaches p_0, as described by the following equation:

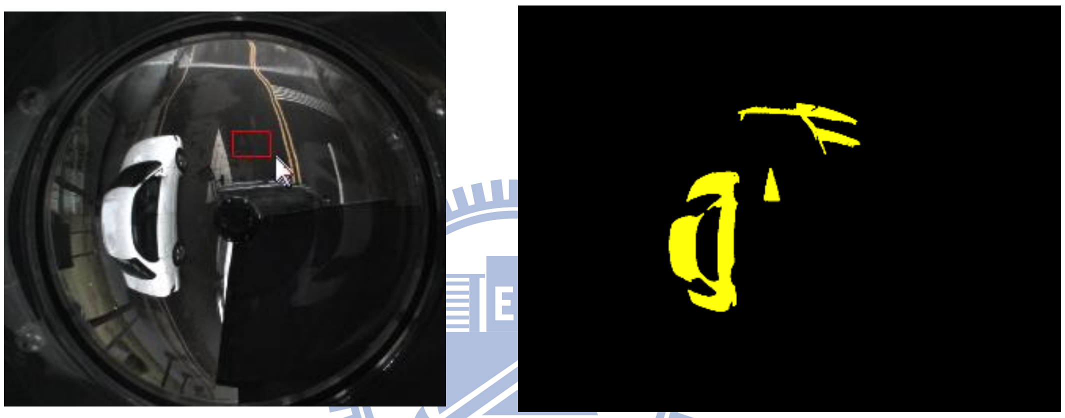
$$p_0 = \frac{1}{n}\sum_{z_j \le t} n_j. \tag{5.5}$$

Now, to conduct the thresholding work, all pixels in the image *f* are scanned and their values are compared with the threshold value *TH*. If a pixel value in *f* is checked to be larger than *TH*, the corresponding pixel in the bi-level image *b* is labeled by “1”; else, it is labeled by “0.” We regard the region labeled by “1” as a car region. Moreover, a morphological process including erosion and dilation operations is used to remove small noise regions and smooth the car shape. An example of the result of moment-preserving thresholding for object segmentation is shown in Fig. 5.3.

5.2.3 Noise Elimination

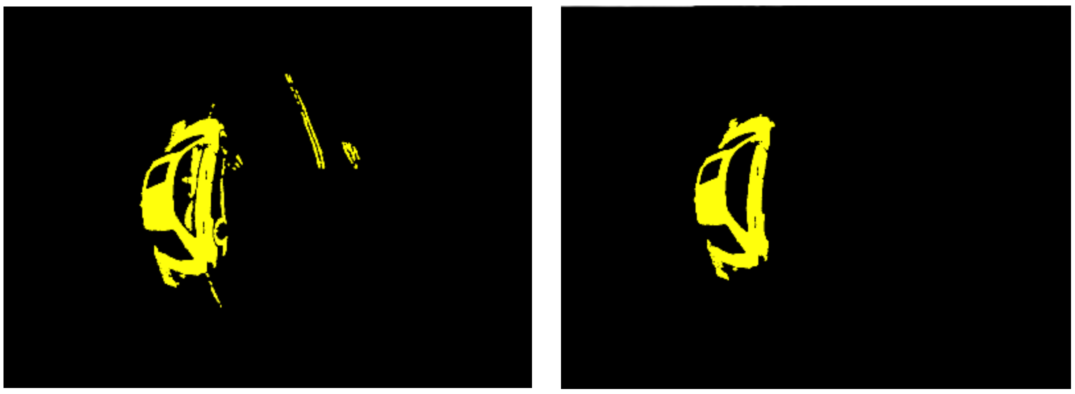
After the thresholding process, the resulting regions in the bi-level image, which are labeled by “1,” may not be the car region because the method of ground elimination cannot eliminate the non-car region clearly all the time. For example, the color of the drive line might be different from the ground, and so the resulting region labeled by “1” will sometimes also include the drive line. An example is shown in Fig.

5.4(a). However, these noise components are usually smaller than the car region. To find the region of the detected car, we have to remove such noise components in the bi-level image. For this, we use the region growing method to find the largest connected component and regard it as a nearby static car. The image resulting from removing noise from Fig. 5.4(a) is shown in Fig. 5.4(b), in which only the car region is left. The proposed method of region growing is described in the following algorithm.



(a) (b)

Figure 5.3 Related images of noise elimination. (a) The original omni-image. (b) The bi-level image of eliminating the ground and thresholding in the image (a).



(a) (b)

Figure 5.4 The bi-level images of the nearby static car detection. (a) The image before noise elimination. (b) The image after noise elimination.

Algorithm 5.1 Region growing for noise elimination in the bi-level image.

Input: a bi-level image B and a threshold value TH for eliminating small regions.

Output: an image I_r including regions whose sizes are larger than TH.

Steps.

Step 4. Initialize an empty stack S as well as a new image I_r whose size is to the same as that of image B for use in recording the searched regions.

Step 5. Divide the range 2π of the azimuth angles into N intervals and define the i-th azimuth angle θ_i as

$$\theta_i = \frac{2\pi}{N} \times i,\ i = 1,\ 2,\ \ldots,\ N-1. \tag{5.6}$$

Step 6. Scan each radial line l_i through the image center in the entire range of 2π according to a pre-selected azimuth angle interval to find objects in image B in the following way:

if an *unsearched* point p labeled by “1” is found to exist on l_i in image I_r, then mark p as *searched* and push it into stack S; else, continue scanning the line l_i until all points on l_i are processed.

Step 7. Grow the region from the scanned points in stack S in the following way as illustrated in Fig 5.5.

4.1 Pop a point p from stack S.

4.2 Search the pixels around p.

4.3 Push the neighboring pixels around p into stack S if the pixels are labeled by “1” in image B and marked as unsearched in image I_r.

4.4 Go to Step 4.1 to repeat the growing process until stack S is empty.

4.5 If the region size is computed to be larger than the threshold value TH, keep the region; else, ignore the region.

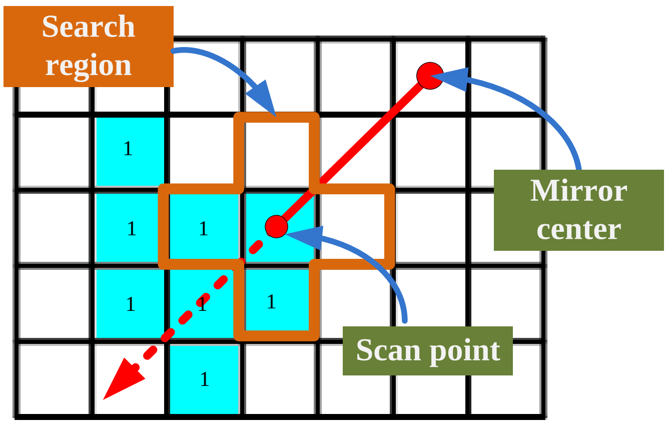


Figure 5.5 An illustration of the region growing process — the blue region represents the car region and the white region represents the non-car region. Once the scan point finds the car region, the region growing process starts.

Through the region growing process described above to find the larger connected component regions, the noise in the bi-level image *B* can be removed and the resulting region is just the detected car shape.

5.3 Distance Estimation of a Static Car

5.3.1 Car Side Extraction and Analysis

In this section, we will introduce how to extract the corresponding point pairs from a pair of images acquired with one of the pair of two-camera omni-directional devices. As observed from the bi-level image *B* resulting from thresholding a given surveillance image, the window region of a detected car is always marked as “0” and the body of the detected car is labeled by “1.” The bottom edge of the vehicle window is a boundary between these two regions. Therefore, it is feasible to detect the bottom-edge points of the vehicle window as feature points to compute the stereo information of the detected car. Accordingly, the proposed technique of car side extraction is divided into two stages — the first stage is to detect the edge points, and the second stage is to find the bottom-edge point pair of the vehicle window from the edge points detected in Stage 1.

In the first stage, we try to detect the edge points of the car window from a pair of images, as illustrated in Fig. 5.6. For this, we scan the points on a radial line starting from the image center to the image boundary. In this scanning process, we have to make sure that the point we scan is not just a noise point. Also, to accomplish the detection of the car window, we have to collect a number of sequential points, called a *consecutive segment*, in the radial line. The following algorithm describes these major steps ;in more detail.

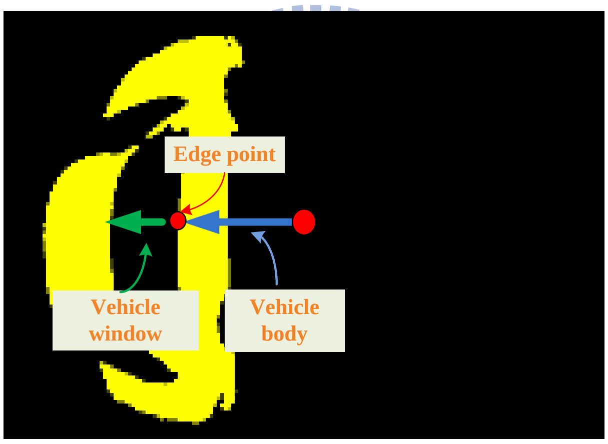


Figure 5.6 An illustration of detecting the edge points in bi-level image.

Algorithm 5.2 Detection of the edge points of the car window.

Input: a bi-level image B with the region of the detected car body labeled by “1” and the other region labeled by “0.”

Output: a buffer B_{p_up} collecting the car window edge points.

Steps.

Step 1. Initialize two flags *Flag_1* and *Flag_2* to be “false” and create an empty buffer B_{temp} to collect searched points.

Step 2. Scan the points on an *unscanned* radial line starting from the image center to find the car body in the following way.

2.1 Sequential check each pixel on the scan line: if the pixel is labeled as "1" in the bi-level image B, push the point into the buffer B_{temp}.

2.2 If the difference between the radial distances of the currently-detected point p_{cur} and that of the last point is smaller than a threshold, then push p_{cur} into the buffer B_{temp}; otherwise, reset the buffer B_{temp} to be empty.

2.3 If the number of the collected point in buffer B_{temp} is larger than a threshold TH_1, set the flag *Flag_1* true.

Step 3. Scan the points on each radial line in a similar way to find the car window in the following way.

3.1 If the flag *Flag_1* is set to be true, then start the process of collecting points of the car window which is similar to the operation conducted in Step 2.

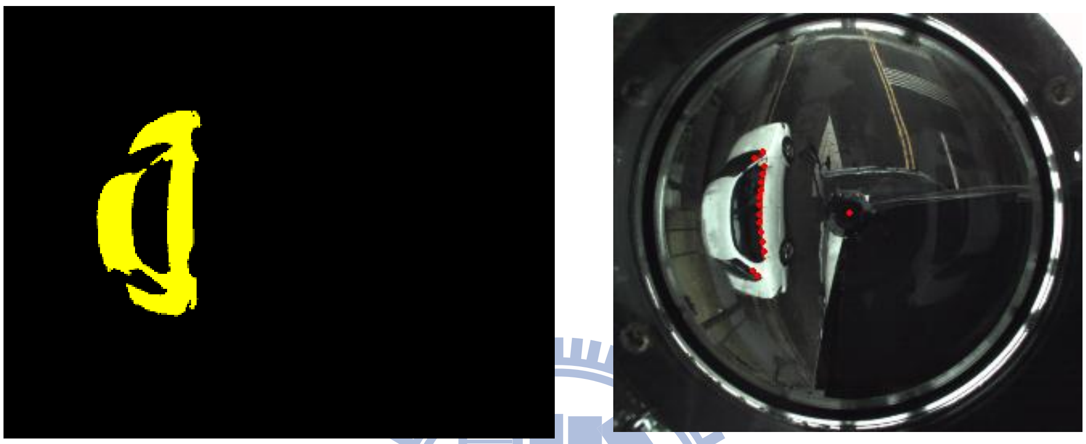
3.2 If the number of points in the consecutive segment reaches another threshold value TH_2, set the flag *Flag_2* true.

Step 4. If the *Flag_2* flag is set, take the beginning point of buffer B_{temp} as the car window edge point and push it into B_{p_up}.

Step 5. Reset the two flags to false and clear the buffer B_{temp}, and go to Step 2 to search the next radial line until all the radial lines are scanned.

Both the images acquired with the upper and the lower cameras are checked by the same process described above to find the bottom-edge points of the vehicle's window and put them into another buffer B_{p_down}. Consequently, we obtain two buffers, B_{p_up} and B_{p_down}, of the bottom-edge points, and these points are analyzed further to complete the work of car window edge detection, as described subsequently.

Although we have detected the edge points as shown in Fig. 5.7 in the upper-camera and lower-camera images, we cannot confirm that all these points are useful. We have to find out the *real* point pairs of the window edge from two buffers B_{p_up} and B_{p_down}. For this, three rules are established to filter noise points.



(a) (b)

Figure 5.7 An example of edge-point extraction. (a) The bi-level image *b* for searching the bottom-edge points of the vehicle window (a) An image to show the result of finding the edge points, and the red points represent the edge points corresponding to (a).

First of all, due to the property of rotational invariance of omni-imaging, there exist two points in the upper-camera image and in the lower-camera image which have the same azimuth angle. And such points are called a corresponding point pair. For each azimuth angle, there exists a corresponding point pair.

Second, we assume that the difference between the radius distances of two consecutive points of the bottom window edge in the buffer B_{p_up} or B_{p_down} is shorter than a threshold value TH_1.

Finally, the number of consecutive edge points of the vehicle window is larger than a threshold TH_2 on the radial line of each azimuth angle.

The details of the above-discussed scheme of collecting the bottom-edge points of the vehicle window are described in the following algorithm.

Algorithm 5.3 Collection of useful corresponding point pairs of the car window edge.

Input: two buffers B_{p_up} and B_{p_down} containing the candidate bottom-edge points of a car window.

Output: a buffer B_{car} for recording the real consecutive point pairs of the bottom-edge points of the car window.

Steps.

Step 1. Initialize an empty buffer B_{temp} for temporarily recording the corresponding point pairs.

Step 2. Scan each azimuth angle θ and check if there exist two points in the two buffers B_{p_up} and B_{p_down}, respectively, which have the same azimuth angle.

Step 3. Check every consecutive point pair of buffer B_{p_up} or B_{p_down} about the following two aspects until all point pairs are exhausted:

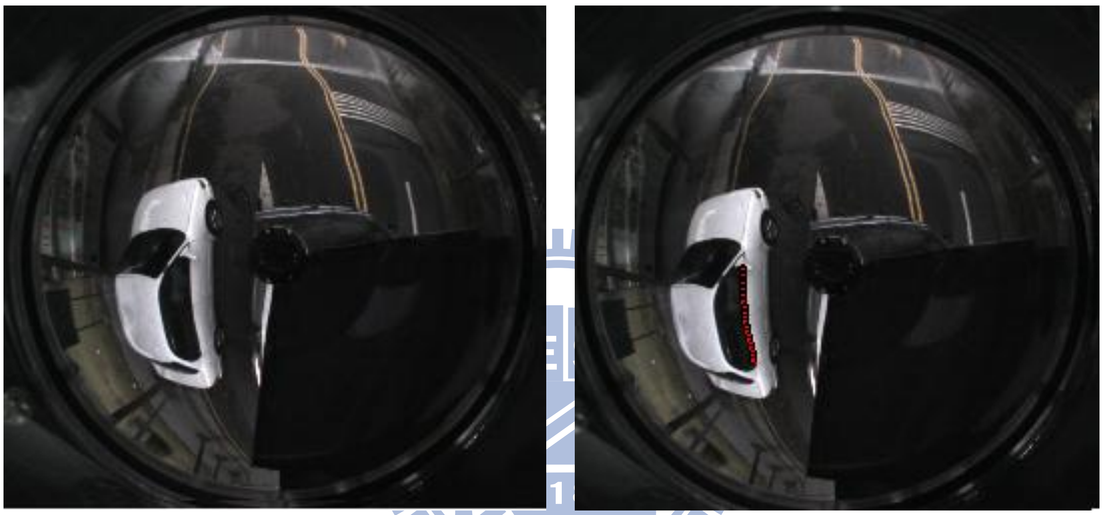
(1) whether the difference between the radius distances of the two consecutive points is smaller than a threshold value TH_1 or not;

(2) whether the difference between the azimuth angles of the two consecutive points is smaller than another threshold value TH_2 or not;

and if both aspects are checked to be true, then push the point pair into the buffer B_{temp} and go to Step 3 to check another pair; else, continue.

Step 4. Check if the number of point pairs in buffer B_{temp} is larger than a third threshold value TH_3, and if so, regard the involved points as coming from the car window edge and push all the point pairs into B_{car} for computing the 3D data of the car window; else, clear buffer B_{temp} and go to Step 2 to continue the search.

As a result of executing the above algorithm, with a pair of bi-level images, where the window region of the detected car is always marked by “0” and the body of the detected car is labeled by “1,” we are able to detect consecutive point pairs of the bottom-edge of the vehicle window as shown in Fig 5.8. The 3D information of the detected car can be computed from these corresponding point pairs, as described in subsequent sections.



(a) (b)

Figure 5.8 The result of edge point extraction. (a) The original omni-image acquired with the omni-camera. (b) The image with the bottom-edge points of the vehicle window represented by red points.

5.3.2 Elimination of Noise by Simple Linear Regression

After extracting the corresponding edge point pairs between the two omni-images acquired with the upper camera and lower cameras, we want to compute the 3D information of these points to infer the position of the nearby car. However, to accomplish this task more precisely, it is desired to transform the points in buffer B_{car} detected by Algorithm 5.3 into the WCS and eliminate possibly the outliers of the point pairs. We do this by the simple linear regression method in this study. In more detail, we divide this process of eliminating noise point pairs into two stages.

The first stage is to transform the points in buffer B_{car} into the WCS. Under the premise that the bottom-edge points of the vehicle's window are highly similar in their characteristics, we can assume that the height of the points are H, and then transform these points to the WCS accordingly. The transformation process is introduced previously in Chapter 4, where the image coordinates (u_i, v_i) of a point was transformed into the world coordinates (X_i, Y_i, Z_i) of a point in the WCS. We use the values of X_i and Y_i in the subsequent linear regression process without considering the value Z_i.

The second stage is to eliminate the outlier point pairs by the use of the simple linear regression method, which is a least-squares estimator with a single predictor variable. The goal is to find the equation of the straight line by Eq. (5.7) below, which fits the given points in a minimum least-square-error (MLSE) sense.

$$Y = aX + b. \tag{5.7}$$

In other words, the *MLSE line* is taken to be the one which minimizes the sum of squared residuals described by

$$Q = \sum_{i=1}^{n} (Y_i - aX_i - b)^2 , \tag{5.8}$$

where it is assumed that n points with coordinates (X_i, Y_i, Z_i) are available for the fitting process. To minimize the value of Q above, first set the partial derivatives of it to be zero's as follows:

$$\frac{\delta Q}{\delta a} = 0, \quad \frac{\delta Q}{\delta b} = 0 . \tag{5.9}$$

Then, the desired values a and b can be solved from the two equations to be as follows:

$$a=\frac{n\sum_{i=1}^{n}X_iY_i-\sum_{i=1}^{n}X_i\sum_{i=1}^{n}Y_i}{n\sum_{i=1}^{n}X_i^2-\left(\sum_{i=1}^{n}X_i\right)^2};\quad b=\frac{1}{n}\sum_{i=1}^{n}Y_i-a\times\frac{1}{n}\sum_{i=1}^{n}X_i. \tag{5.10}$$

With the parameters a and b computed, we finally can draw a *linear regression line* L_{reg} on the X-Y plane in the WCS, as shown in Fig. 5.9. And the distance between each point with coordinates (X_i, Y_i, Z_i) and the line L_{reg} may be calculated by the following equation according to trigonometry:

$$d_i=\frac{|Y_i-aX_i-b|}{\sqrt{a^2+b^2}}. \tag{5.11}$$

Accordingly, the distance d_i of each edge point in buffer B_{car} is computed by Eq. (5.11) and if the distance of a certain point so computed is larger than a threshold value TH, it will be regarded as an outlier and removed from buffer B_{car}. Finally, we compute the 3D data of the remaining corresponding point pairs in buffer B_{car}. The method for doing this has been discussed previously in Section 2.3.

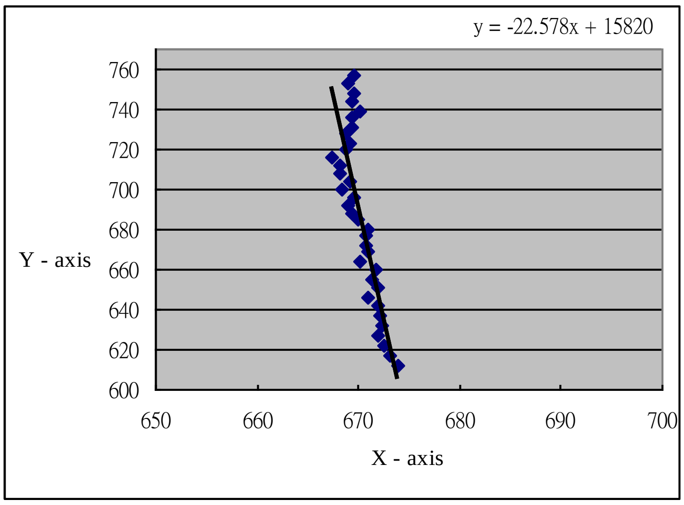


Figure 5.9 An example of simple linear regression, where the blue points represent the edge points transformed into the WCS and the black line is the result.

5.3.3 Calculation of Car Distance and Creation of Surround Map

To show the relative position of the video surveillance vehicle and the detected car on a surround map, it is required to get the 3D position information of the detected car. For this purpose, the 3D data of the corresponding point pairs is computed by the 3D data acquisition method described in Section 2.3. Let the height and the distance of each point pair be denoted as H_i and D_i, respectively, and the total number of the corresponding point pairs be as n. The height H_{car} and distance D_{car} of the detected car is taken to be the mean value of all the values of H_i and D_i of these points, respectively, as follows:

$$H_{\text{car}} = \frac{1}{n}\sum_{i=1}^{n} H_i, \quad D_{\text{car}} = \frac{1}{n}\sum_{i=1}^{n} D_i\ . \tag{5.12}$$

In addition, from each point pair we can obtain the azimuth angle with respect to the X-axis. And the azimuth angle θ_c of the *middle* point pair among these corresponding points is selected to represent the azimuth angle θ_{car} of the detected car. With the horizontal distance D_{car} and azimuth angle θ_{car}, the relative position of the detected car can be described as the coordinates ($u_{\text{car},}$ v_{car}) in a top-view 2D coordinate system created for displaying the surround map, which may be computed in the following way:

$$u_c = (D_{\text{car}}\times\cos\theta_{\text{car}})/ratio; \qquad v_c = (D_{\text{car}}\times\sin\theta_{\text{car}})/ratio, \tag{5.13}$$

where the value *ratio* is a scaling factor to scale the real WCS distance down into the top-view 2D coordinate system.

Once the 3D information of the detected car is obtained, we can generate the surround map from the top view as shown the example of Fig 5.10. A detailed algorithm for doing this is given in the following.

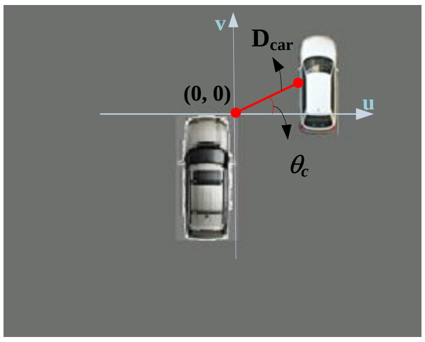


Figure 5.10 A surround map from the top view.

Algorithm 5.4 A method of generating the surround map.

Input: the position of a detected nearby car at top-view coordinates (u_c, v_c) computed by Eqs. (5.13).

Output: a top-view surround map of the vehicle environment including the video surveillance vehicle and the detected nearby car.

Steps.

Step 1. Initialize a background image *I* with all pixels colored in a gray color like that of the asphalt road.

Step 2. Paste a graphic model of the video surveillance vehicle at the center of image *I*.

Step 3. Select the front-right corner of the video surveillance vehicle model as the origin with coordinates (0, 0) of a top-view 2D coordinate system for displaying the desired surround map.

Step 4. Paste a graphic model of the nearby car on *I* at coordinates (u_c, v_c).

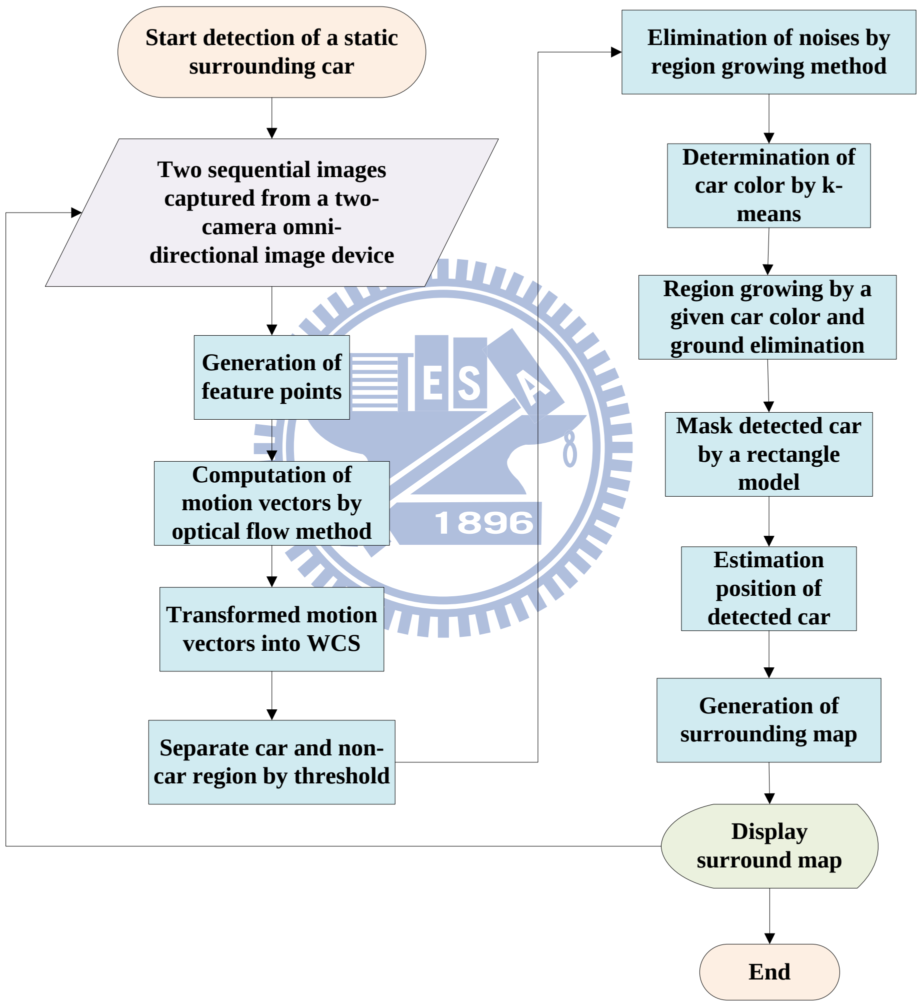
Step 5. Take the final *I* as the desired surround map.

Chapter 6
Monitoring of a Nearby Static or Moving Car with a Moving Video Surveillance Vehicle

6.1 Idea of Detection of Static or Moving Car in Omni-images

In Chapter 5, we have described the proposed method for detection and display of a nearby car. Both the detected car and the video surveillance vehicle are assumed to be static there. However, in this study we propose further a method to detect a nearby moving or static car while the video surveillance vehicle is being driven. Optical flow analysis may be used again here to estimate the motion of an object in consecutively acquired omni-images, and if the concerned object is higher than the ground, its motion in the image will produce motion vectors with larger lengths. This property may be used to segment the car from the background. Moreover, we also analyze the color of the detected car by the k-means algorithm and use the color information to segment out the car region in the omni-image. Finally, the position of the detected car is estimated by a template matching method proposed in this study, and a surround map is generated accordingly.

As a summary, we may divide the proposed method into five major steps: (1) computing the motion vectors by optical flow analysis; (2) separating the car region from the non-car one roughly according to the motion vector lengths; (3) segmenting out the car region by the color information; (4) estimating the position of the detected car; and (5) generating a surround map. A flowchart illustrating these major steps of the proposed method is shown in Figure 6.1. All of the above steps will be elaborately introduced in the following sections.



Start detection of a static surrounding car
Two sequential images captured from a two-camera omni-directional image device
Generation of feature points
Computation of motion vectors by optical flow method
Transformed motion vectors into WCS
Separate car and non-car region by threshold
Elimination of noises by region growing method
Determination of car color by k-means
Region growing by a given car color and ground elimination
Mask detected car by a rectangle model
Estimation position of detected car
Generation of surrounding map
Display surround map
End

Figure 6.1 Flowchart of nearby car detection with a moving video surveillance vehicle.

6.2 Moving Car Detection by Motion Vectors Generated by Optical Flow Analysis

6.2.1 Detection of Car Region by Motion Vector Lengths

To separate the car region from the non-car one in the consecutively acquired omni-images, we use optical flow analysis to produce the motion vectors and detect the car region by the motion vector lengths. The details of this process are described in order subsequently.

A. Block Based Processing

To monitor the surrounding environment of the video surveillance vehicle, we evenly select points in the omni-image to compute the motion vectors by optical flow analysis discussed previously. For the selected points to be evenly distributed in the omni-image, we divide the omni-image into equal-sized blocks and select the points to be the centers of the blocks. An example of the block-based omni-image is shown in Figure 6.2. The following process will take image block as the unit of processing.

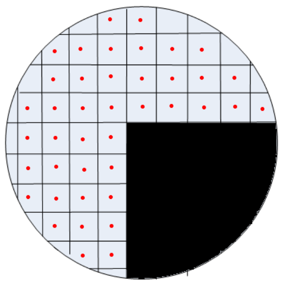


Figure 6.2 An example of block-based omni-image — the block region is the video surveillance vehicle roof that we ignore and the red points are the selected points.

B. *Estimation and Transformation of Motion Vectors*

We want to estimate the motions of objects in the surrounding environment from two consecutive omni-images. The process is divided into two stages: (1) estimation of the motion vectors by optical flow analysis; and (2) transformation of the motion vectors from the ICS into the WCS. The optical flow analysis method used in Stage 1 has been reviewed in Section 4.2.2.B. And the transformation process used in Stage 2 is identical to that described in Section 4.2.2.C.

C. *Detection of Ground and Car Regions by Motion Vector Lengths*

Because the driving speed of the video surveillance vehicle is not constant and the lengths of the motion vectors are roughly proportional to the car speed, we can use dynamic thresholding to separate the car region from the background, as shown by the example seen in Fig. 6.3. We use the standard deviation value of all the motion vector lengths to set the threshold value. Assume that the length of each motion vector is denoted as L_i and the total number of motion vectors is n. Then, the mean value $\bar{L}$ and the standard deviation value S_n of the motion vector lengths are computed as follows:

$$\bar{L} = \frac{1}{n}\sum_{i=1}^{n} L_i, \quad Sn = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(L_i - \bar{L})^2}. \tag{6.1}$$

The threshold values for detecting the car and the ground are respectively set to be as follows:

$$L_{car} = \bar{L} + Sn; \tag{6.3}$$

$$L_{ground} = \bar{L} - Sn. \tag{6.4}$$

To record the car region, we initialize a *record image* I_r which is of the same size as that of the original omni-image. The car region to be put into the mage I_r is labeled by "1"; and the other regions, by "0." Each motion vector produced by optical flow analysis is checked and compared with the threshold values L_{car} and L_{ground}. If the length of the motion vector is larger than L_{car}, we regard the block yielding the vector as a region of the detected car and label it by "1" in image I_r; else, we label the block by "0." Besides, we also define the threshold value L_{ground} and use it in the following way: if the length of a motion vector is smaller than L_{ground}, every pixel in the corresponding region is regarded as a ground point, and pushed into a buffer B for use in conducting a *ground learning* process described in the subsequent section.

In summary, we list the rules for car detection as follows:

$$\begin{cases} \textit{if length of motion vector} \geq \overline{L} + S_n, \textit{ then label the region as "car"}; \\ \textit{if length of motion vector} \leq \overline{L} - S_n, \textit{ then label the region as "ground."} \end{cases} \tag{6.5}$$

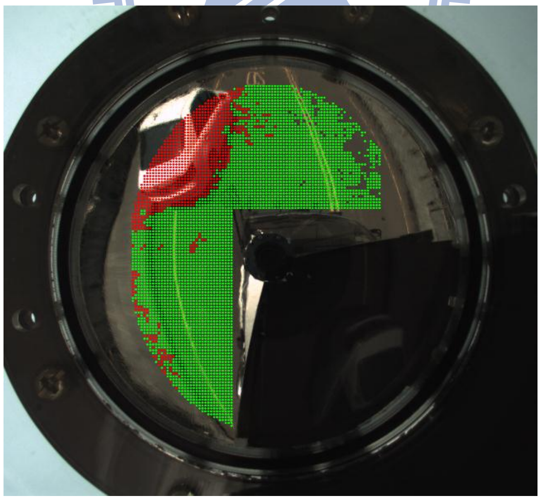


Figure 6.3 A result of separating the car region from the non-car region, where the red points are used to represent the car region and the green points to represent the non-car region.

6.2.2 Detection of Car Body by k-means Algorithm

In order to detect pixels of the car body in image I_r for region growing method, we divide the process into two steps: (1) use the k-means algorithm to partition a set of feature points into three clusters (2) determine which cluster is the car body. The steps of the k-means algorithm are illustrated in Fig. 6.4, and a detailed algorithm implementing it is introduced as follows.

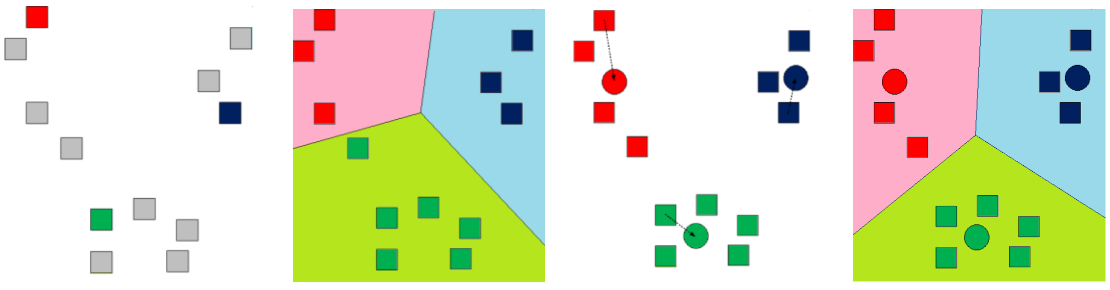


Figure 6.4 An illustration of k-means algorithm. (a) The image of initialize the cluster centers. (b) The image of associating every data with the nearest mean. (c) The image of reassigning the cluster centers. (d) The result image of k-means algorithm.

Algorithm 6.1 Partitioning feature data into clusters.

Input: a set of feature points D_i and a total number k of clusters.

Output: the center point C_j of each cluster and a set of input feature points labeled with cluster index.

Steps.

Step 1. Initialize k cluster centers randomly among the input data.

Step 2. Perform the following steps to the feature points until either the number of iterations reaches a pre-selected limit or the centers of clusters become stable (with no change in the positions of the cluster centers).

2.1 Calculate the distance between each feature point D_i and each cluster center, and label D_i with the index of the closest cluster center.

2.2 Update each cluster center C_j by calculating the mean value of the feature points which are labeled as j.

As a result, the *k*-means algorithm may be used to partition a set of feature points into *k* clusters, with each point belonging to the nearest cluster. To use the algorithm in this study, the RGB values of the center pixel of each block in image I_r are taken as the input feature points into the k-means algorithm, and *k* is taken to be 3, i.e., all input feature data are partitioned into three clusters, in which one is the car body of a certain color. Note that in this study, we assume that each car is of a single color.

Furthermore, the car region in image I_r consists of three possible types of objects — the body of the detected car, the windows of the car, and noise components. Therefore, we have to find the cluster of the car body by analyzing the center of each cluster for the region growing method, and the region growing method will be introduced in the following section.

Moreover, the color of the car window is sometimes similar to that of the ground. To avoid growing the ground, each cluster with its center's gray value close to the ground value g_{m} should be ignored. That is, if the gray value of a cluster center is close to the ground value g_{m}, we will not conduct region growing with the cluster as the starting point, as described in Section 6.2.3.*B*. The computation of the value g_{m} will be introduced in Section 6.2.3.*A*. The algorithm of determining the cluster of the car body from those yielded by the above algorithm (Algorithm 6.1) is described in detail as follows.

Algorithm 6.2 Determination of the cluster of the car body.

Input: the center points C_i of the clusters S_i found by Algorithm 6.1 with RGB values (R_i, G_i, B_i), i = 1, 2, 3; the number N_i of feature points in each cluster S_i, and the gray value g_{m} of the ground.

Output: a cluster S_j of feature points of the car body or none.

Step.

Step 1. Sort the numbers N_i of feature points of all the clusters S_i, $i = 1, 2, 3$, and pick up the largest one with index j, namely, N_j, which is the number of feature points of cluster S_j.

Step 2. Compute the difference D between the gray value of the center C_j of cluster S_j and that of the ground, g_{m}, as follows:

$$D = |(R_j+G_j+B_j)/3 - g_{\mathrm{m}}|.$$

Step 3. If the difference D is larger than a threshold TH, output the cluster S_j with index j as the desired car body; else, ignore the cluster S_j and go to Step 1 to process the remaining cluster(s).

6.2.3 Detection of Car Region by Color Information

After finding out the feature-point cluster of the detected car body by the above two algorithms, we are able to detect the entire car region more completely in the image I_r produced in Section 6.2.1.*C* by eliminating the ground area and growing the region of the detected car body, as described in the following.

A. *Elimination of the Ground Region*

To decrease the probability of false alarms and erroneous detections, we have to learn the ground information by the use of a buffer B which collects the ground pixels found in Section 6.2.1.*C* and eliminate the ground regions from the image I_{r}. The mean gray value g_{m} of the ground is computed as follows:

$$g_{\mathrm{m}} = \frac{1}{n}\sum_{i=1}^{n}(R_i + G_i + B_i)/3, \tag{6.6}$$

where n is the total number of pixels in the buffer B, and R_i, G_i and B_i represent the RGB values of the i-th pixel in B.

To eliminate the ground region, we scan all the blocks in image I_r which are labeled by “1” and apply the following classification rule:

"if the difference value satisfies the following condition:

$$g_m - diff \leq I(x, y) \leq g_m + diff, \tag{6.7}$$

then mark the block as the ground region in image I_r;
else, continue,"

where $I(x, y)$ is the grayscale value of the pixel in the block center at image coordinates (x, y) and *diff* is a gray value threshold value. After this process of eliminating the ground region, the detection of the car region in image I_r will be more accurate.

B. Detection of Car Region by Region Growing within a Color Tolerance

To make the detected car shape more complete, we want to select the points in the region of the detected car body and regard these points as seed points to grow the neighboring regions within a color tolerance. More specifically, after we use Algorithm 6.2 to find the feature points of a detected car body, we want use the points further to grow the entire car region. Each feature point in the cluster of the car body corresponds to a pixel in the image I_r, and we want to take the pixel as a seed point to grow the neighboring points under two conditions: (1) the difference of the color value between the seed point and the neighboring point is within a color tolerance; and (2) the neighboring point is inside the growing range centered at the seed point. If the color value of the neighboring point is similar to the seed point, we will regard the neighboring point as belonging to the car region and label the block by “1” in the image I_r. The method of detecting the car region with a given color tolerance is described in Algorithm 6.3 below.

Algorithm 6.3 Detecting the car region by filling regions within a given color

tolerance.

Input: the record image I_r, the original image I_o acquired with the omni-camera, and the output data (the cluster centers) of the k-means algorithm described in Algorithm 6.1 in Section 6.2.2.

Output: the bi-level image I_r with the car region labeled by “1.”

Step.

Step 1. Scan each input pixel p in I_r and check whether the label of p is the same as that of the cluster center of the car body or not. If the same, continue; else, scan the next input pixel in I_r.

Step 2. Regard the pixel p as a seed point and check the following classification rule for region growing by the color information:

$$\begin{array}{l}\text{"if a neighboring point of } p \text{ satisfies the following conditions:} \\ \quad I_o(u,v)r - diff \le I_o(u',v')r \le I_o(u,v)r + diff, \\ \quad I_o(u,v)g - diff \le I_o(u',v')g \le I_o(u,v)g + diff, \\ \quad I_o(u,v)b - diff \le I_o(u',v')b \le I_o(u,v)b + diff, \\ \text{then label by "1" the corresponding block as belonging to the car} \\ \text{region in the image } I_r\text{; else, continue,"}\end{array} \tag{6.8}$$

where $I_o(u, v)c$ with $c = r, g$, and b denotes the currently-observed pixel’s c-color value of the seed point located at image coordinates (u, v), $I_o(u', v')c$ with $c = r, g$, and b denotes the c-color value of a neighboring pixel at image coordinate (u', v'), and the value $diff$ is the color tolerance between the currently-observed pixel and one of its neighbor.

As shown in Fig. 6.5, after the region growing process by the color information using the above algorithm, we obtain a more accurately detected car shape from the omni-image.

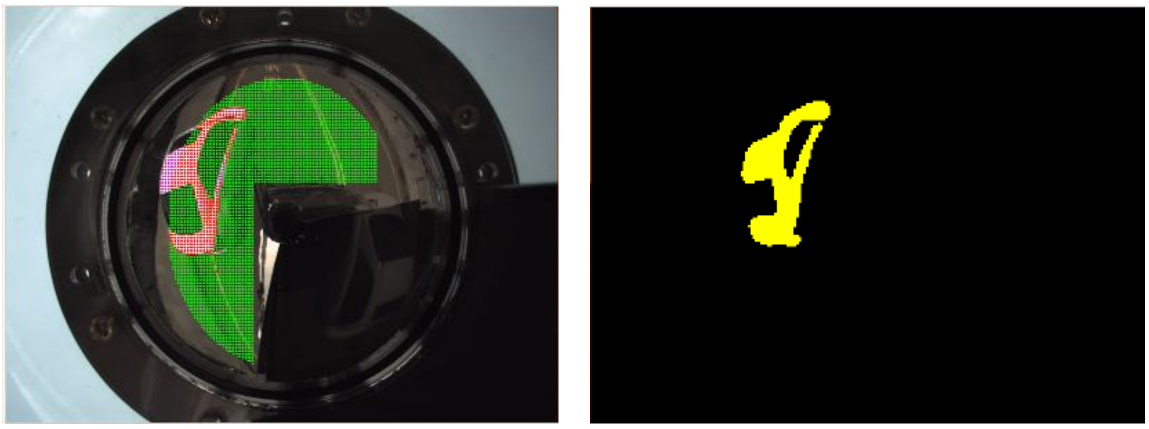

(a) (b)

Figure 6.5 A result of region growing by the color information. (a) An image to show the result of the region growing, and the purple points represent the growing region. (b) The corresponding bi-level image of the image (a).

6.3 Updating of Car State

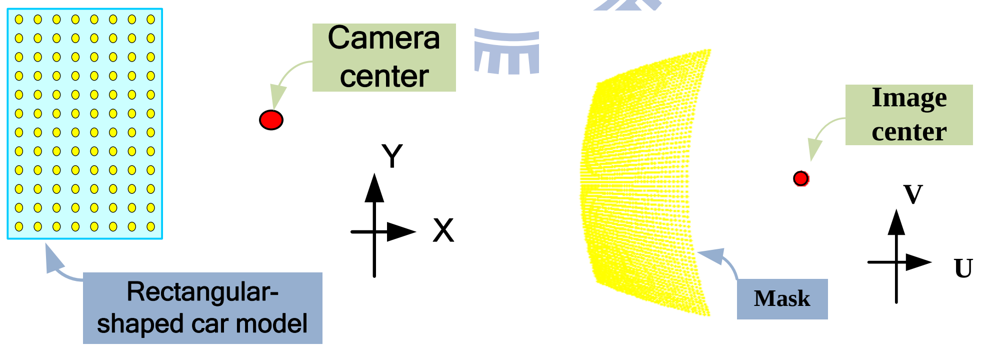
6.3.1 Estimation of Car Location by Rectangular-shaped Models

To estimate the location of a nearby car, we match the detected car region in the image I_r by a mask for estimation of the approximate car position. The method of the estimating the car location includes two stages: (1) generation of the car mask by transforming a rectangular-shaped car model from the WCS into the ICS; and (2) detection of the car location by a template matching scheme.

Stage 1. Generation of the car mask on the image plane

To generate the car mask on the omni-image plane for matching the detected car, we have to transform the car model in the WCS to the ICS as shown in Fig. 6.6. Because the car shape is similar to a rectangle, we imagine a rectangular-shaped car model on the *X*-*Y* plane and transform the model into the image plane for matching the detected car in order to locate the car. For this, in this study we create a series of rectangular-shaped car models besides the video surveillance vehicle under the premise that the detected car is driving around the video surveillance vehicle. Moreover, in order to shorten the computation time of the matching process, we just select sparse points on each rectangular-shaped model to represent the model instead of transforming the whole rectangular-shaped model region for matching the detected car.

More specifically, we divide the rectangular-shaped model into small blocks and transform the center point of each block from the WCS into the ICS to create a car mask. We use buffers to record the points transformed from each model in the ICS for the further matching process. The algorithm for such a transformation from the WCS into the ICS is described as follows.



(a) (b)

Figure 6.6 A rectangular-shaped car model and the corresponding mask. (a) A blue region of rectangular-shaped model and its representive points in the WCS. (b) The mask image.

Algorithm 6.3 Coordinate transformation of a car model point from WCS to ICS.

Input: a point *P* of the rectangular-shaped model with world coordinates (X_i, Y_i, Z_i), the origin of the mirror center O_m at coordinates (X_0, Y_0, Z_0), and the pano-mapping table T_{pm}.

Output: a point P' at image coordinates (u_i, v_i) corresponding to the point P in the WCS.

Steps.

Step 1. Compute the azimuth and elevation angles of the point P in the following way.

1.4 Compute the horizontal distance d between the point P and the mirror center O_m by the following formula:

$$d = \sqrt{(X_i - X_0)^2 + (Y_i - Y_0)^2}\,. \tag{6.9}$$

1.5 Calculate the azimuth angle θ and the elevation angle ρ of point P as follows:

$$\begin{aligned}\theta &= \cos^{-1}\frac{X_i - X_0}{d} = \sin^{-1}\frac{Y_i - Y_0}{d},\\ \rho &= \frac{\pi}{2} - \tan^{-1}\frac{Z_i - Z_0}{d}.\end{aligned} \tag{6.10}$$

Step 2. Find the corresponding point P' at image coordinates (u_i, v_i) by looking up the pano-mapping table with the azimuth angle θ and the elevation angle ρ.

After conducting the above process, we transform all the points in the rectangular-shaped car model in the real world space to the image plane and keep all the coordinates (u_i, v_i) in a buffer for matching the detected car. Furthermore, we use the center of the mask to represent the car location, and the center of the mask is computed by:

$$u_c = \frac{1}{n_i}\sum_{i=1}^{n_i} u_i,\; v_c = \frac{1}{n_i}\sum_{i=1}^{n_i} v_i\,, \tag{6.11}$$

where (u_i, v_i) are the image coordinates of the i-th point in the mask and n_i is the total number of points in the mask.

To conduct the matching process, we generate in advance a series of masks for use at different positions as shown in Fig. 6.7 for accelerating the matching process. However, the point density of each mask is not exactly the same. The points of a far rectangular-shaped model, after being transformed from the WCS into the ICS, will result in a high-density mask and the points in the mask may overlap each other. Accordingly, the spacing d_s between two points of the s-th rectangular-shaped model is related to the distance between the video surveillance vehicle and the model, and we compute the value d_s as follows:

$$d_\mathrm{s} = 10 + |s - (n_{model}/2)| \times 5, \tag{6.11}$$

where the n_{model} is the total number of the points in the model.

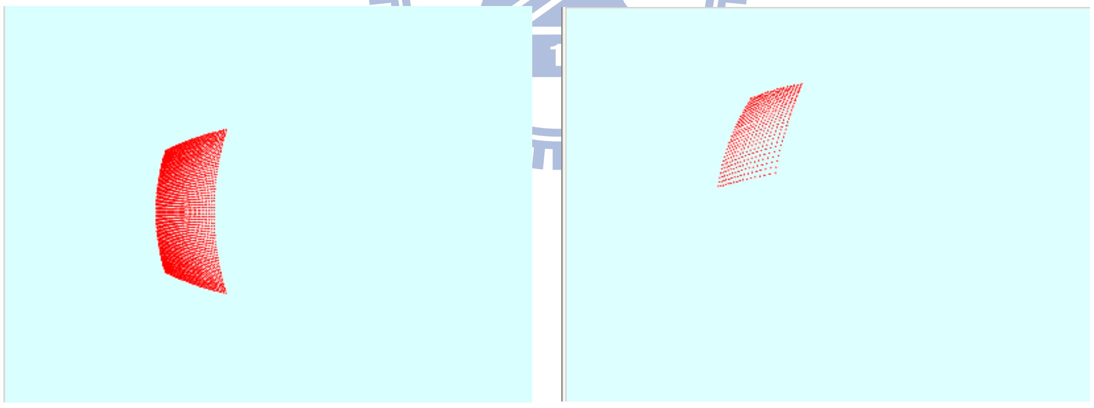

(a) (b)

Figure 6.7 The result of mask in the omni-image. (a) The near mask with respect to the video surveillance vehicle. (b) The far mask with respect to the video surveillance vehicle.

Stage 2. Detection of the detected car location by template matching method

To locate the detected car in the omni-image, we use a template matching method to overlap the detected car with the mask. See Fig. 6.8 for an illustration. The masks are generated in the previous stage, Stage 1, and we have to check each overlapping ratio between the mask points of the *i*-th mask and the detected car shape in the image I_r. The mask which results in the highest overlapping ratio value will be regarded the as the most suitable one for the detected car, and the center point of the mask at coordinates (u_i, v_i) is finally taken as the position of the detected car. The algorithm of detecting the car location by template matching is described as follows:

Algorithm 6.4 Detecting the car location by template matching.

Input: the record image I_r.

Output: a center point of the mask which matches the detected car the best at coordinates (u_i, v_i).

Steps.

Step 1. Initialize a counter *c* to record the number of overlapping pixels.

Step 2. Count the overlapping pixels between the *i*-th mask and the car region in the image I_r by an AND operation.

Step 3. Calculate the corresponding overlapping ratio by dividing the counter value *c* into the total number of the points in the *i*-th mask.

Step 4. Repeat Steps 2 and 3 until the overlapping ratios of all masks have been computed.

Step 5. Find the *i*-the mask which results in the highest ratio and output the center point of the mask at coordinate coordinates (u_i, v_i).

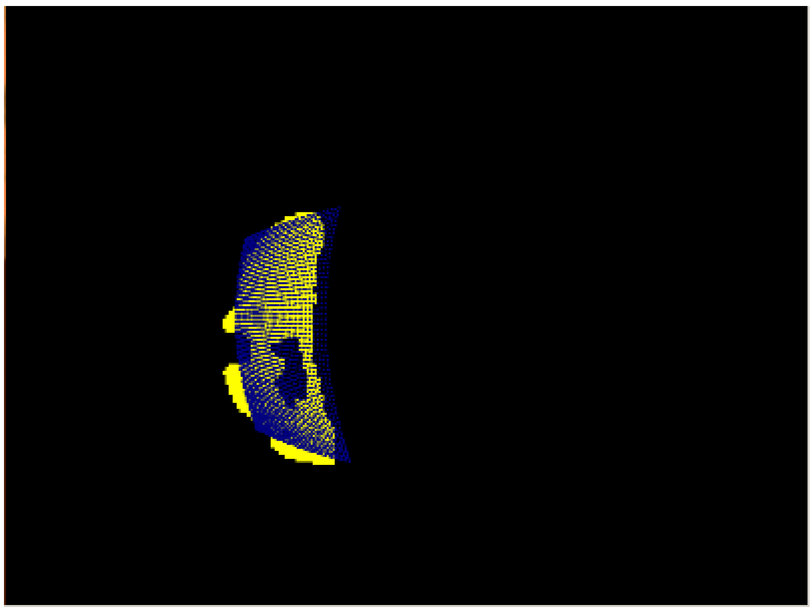


Figure 6.8 A result of matching a detected car by a mask — the yellow region represents the detected car and the blue mask represents the result of the rectangular-shaped model transformation.

6.3.2 Update of Car State and Generation of Surround Map

The state of the detected car recorded in a buffer *B* will be updated in each cycle of the car detection process for generating the surround map. We use the state of the currently-detected car to update the car state in buffer *B* and the surround map from the top view as illustrated in Fig. 6.9. The state of the detected car is defined to include four values: (1) the distance between the omni-camera and the detected car; (2) the azimuth angle of the car with respect to the *u*-axis; (3) a time-to-live value; and (4) a flag to confirm that the currently-detected data is not a false alarm. For smooth and steady displaying of the detected car in the surround map, the time-to-live value is used to avoid the case of the erroneous detection, and the confirmation flag is used to avoid the case of false alarming. The proposed method for updating the detected car state and generating the surround map is divided into five steps as described in the following algorithm.

Algorithm 6.4 Updating the car state for generating the surround map

Input: a state of the currently-detected car and the record buffer *B*.

Output: an updated surround map.

Steps.

Step 1. Decrease the time-to-live value of the cars in the buffer *B* by 1.

Step 2. Discard the car state in the buffer *B*, if the time-to-live value of the car is equal to zero.

Step 3. Push the state of the currently-detected car into the buffer and finish the updating process, if the buffer *B* is empty; else, continue.

Step 4. Update the car state in the buffer *B*.

4.1 Calculate the azimuth angle difference θ_d between the azimuth angle of the currently-detected car and the azimuth angle of the car state in the buffer.

4.2 If the θ_d is smaller than thirty degrees, continue; else, push the state of the currently-detected car into the buffer.

4.3 Update the length of the car in buffer *B* as follows:

$$L_{his} = p \times L_{cur} + (1 - p) \times L_{his}, \tag{6.12}$$

where L_{his} and L_{cur} are the length of the car in buffer *B* and the length of the currently-detected car, respectively, and *p* is a pre-determined weight of history data on the update data.

4.4 Update the azimuth angle of the car in buffer *B* as follows:

$$A_{his} = p \times A_{cur} + (1 - p) \times A_{his}. \tag{6.13}$$

where L_{his} and L_{cur} are the length of the car in buffer *B* and the length of the currently-detected car, respectively, and *p* is the weight of history data with respect to the updated data.

4.5 Set the confirmation flag "true" for displaying the detected car.

4.6 Reset the time-to-live value of the car state in the buffer *B* to three.

Step 5. Update the surround map.

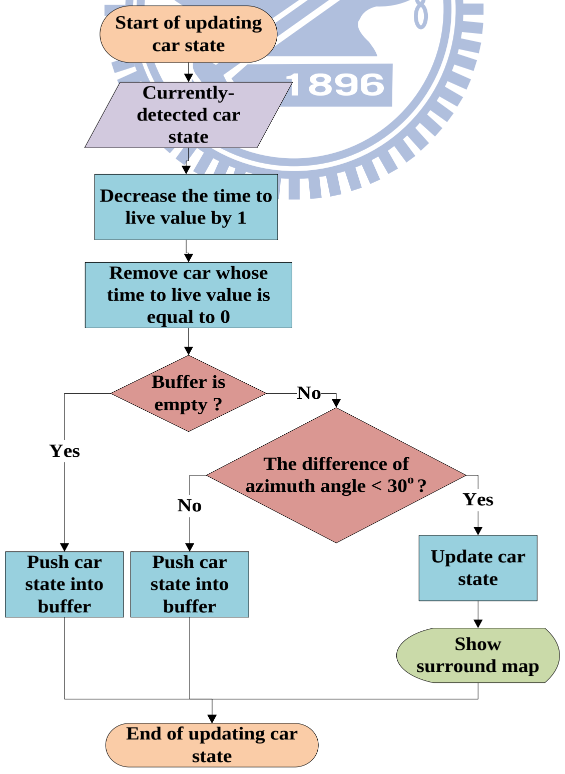
5.1 Check the confirmation flag of the car state in buffer *B*: if the flag is set, update the position of the detected nearby car at top-view coordinates (u_c, v_c).

5.2 Compute the coordinates (u_c, v_c) of the detected car as follows:

$$u_c = (L_{his} \times \cos A_{his})/ratio; \qquad v_c = (L_{his} \times \sin A_{his})/ratio, \tag{5.14}$$

where the *ratio* value is a scaling factor to scale down the real distance into the surround map for image display.

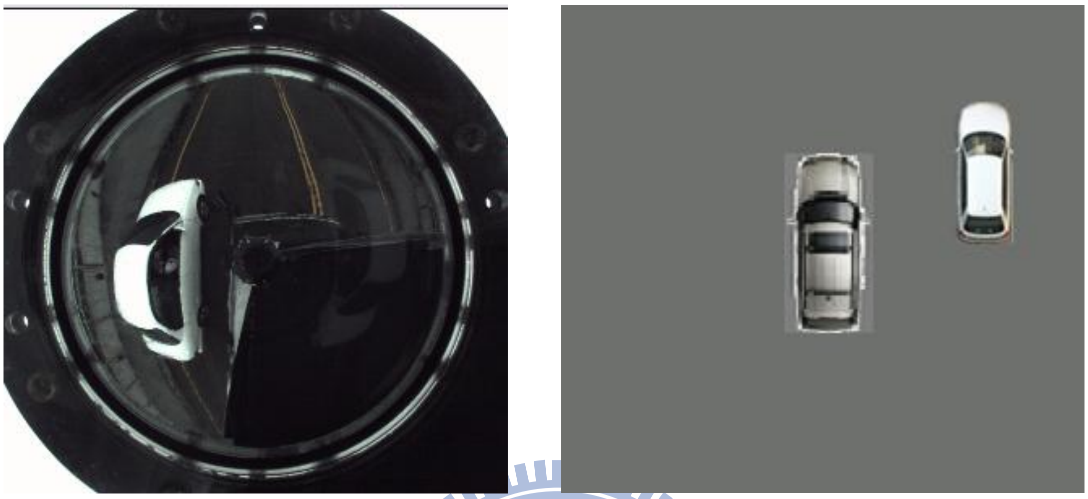
5.3 Show the surround map by the Algorithm 5.4.



Start of updating car state
Currently-detected car state
Decrease the time to live value by 1
Remove car whose time to live value is equal to 0
Buffer is empty ?
No
Yes
The difference of azimuth angle < 30° ?
No
Yes
Push car state into buffer
Push car state into buffer
Update car state
Show surround map
End of updating car state

Figure 6.9 Flowchart of updating the car state.

The experimental result shown in Fig. 6.10 is an example of the resulting surround map after detecting a nearby static car with a moving video surveillance vehicle. With the surround map, we can observe the surrounding environment easily.



(a) (b)

Figure 6.10 The result of detecting the static nearby car with a moving video surveillance vehicle. (a) The original omni-image. (b) The surround map from the top view.

Chapter 7
Experimental Results and Discussions

7.1 Experimental Results

In this chapter, we will show some experimental results of the proposed methods for use on a video surveillance vehicle with two 2-camera omni-imaging devices. The experiments were mainly conducted on an open space area, including a parking lot and a spacious around-campus road with an asphalt surface in National Chiao Tung University.

The first experiment was to generate the perspective-views image from an omni-image acquired with one of the upper cameras affixed on the video surveillance vehicle. With the perspective mapping table T_{pp}, the perspective-view image was generated from the omni-image in six view directions. The second experiment was to analyze the omni-image of the surrounding environment for estimation of the moving direction of the video surveillance vehicle in the lanes of the parking lot and to display the corresponding perspective-view image. The third experiment was to detect the static nearby car on the around-campus road while the video surveillance vehicle was in a static state, and then to show the surround map from the top view by computing the 3D information of the detected car. The final experiment was to monitor a nearby moving or static car with the video surveillance vehicle in a moving state, and show the relative position of the detected car in the surround map.

A. Experimental Results of Perspective-view Image Generation

In this experiment, we acquired an omni-image from the upper omni-camera of one of the two two-camera omni-directional imaging devices to generate the perspective-view images for three directions by the use of the perspective mapping table T$_{pp}$. The method of image transformation is introduced in Chapter 3, and an experimental result is shown in Fig. 7.1.

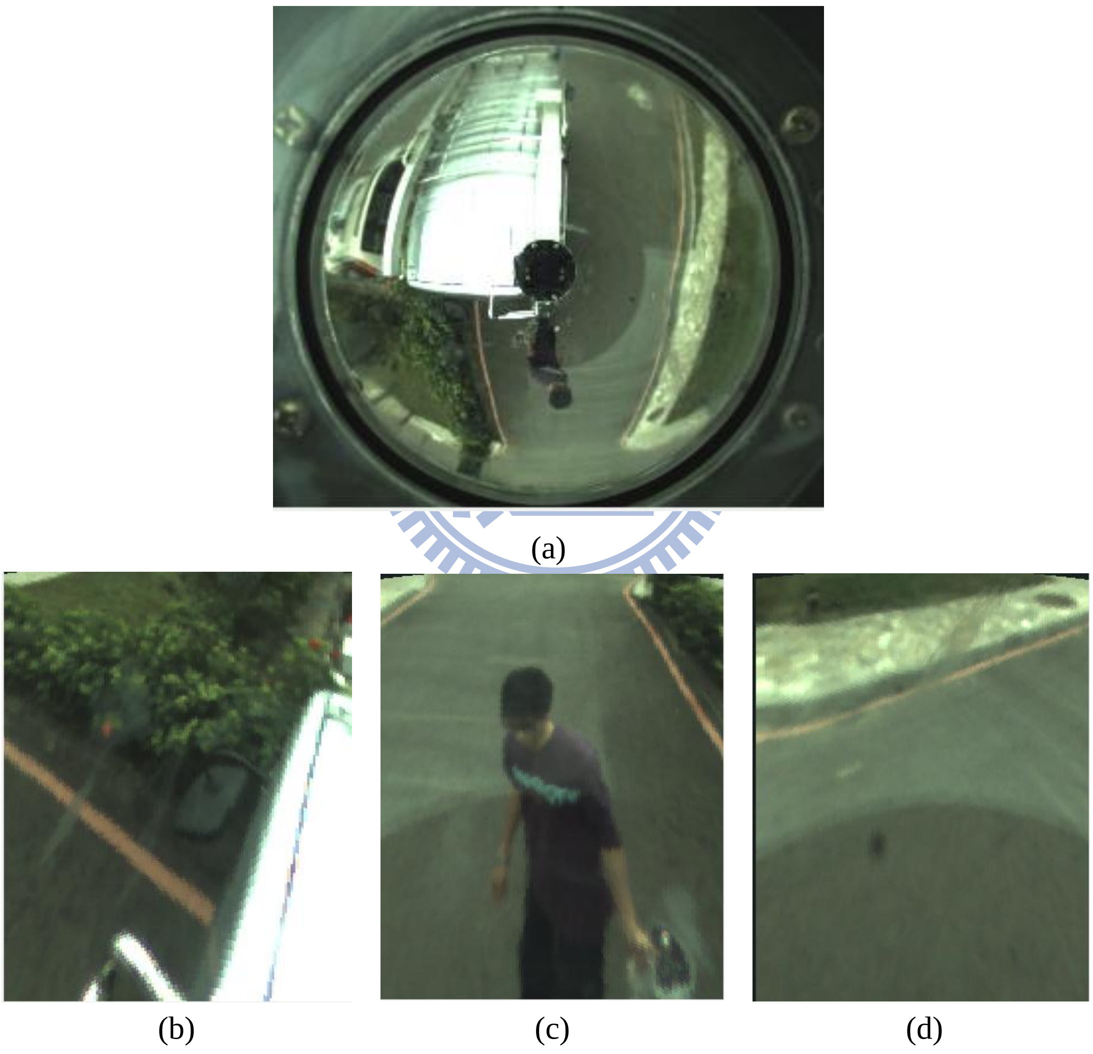
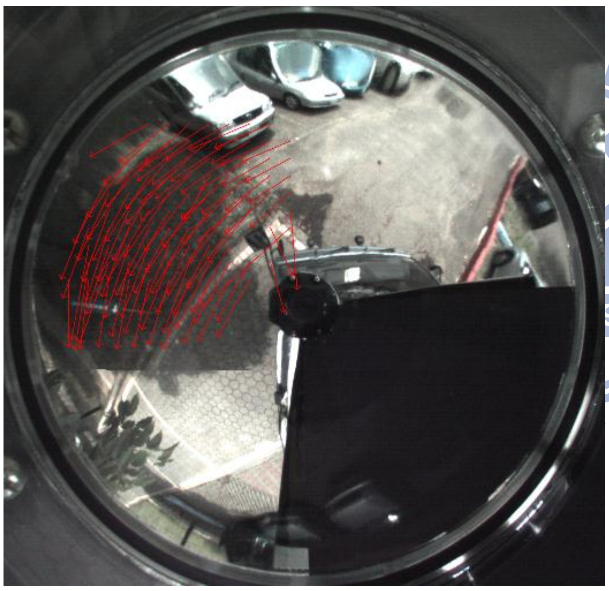


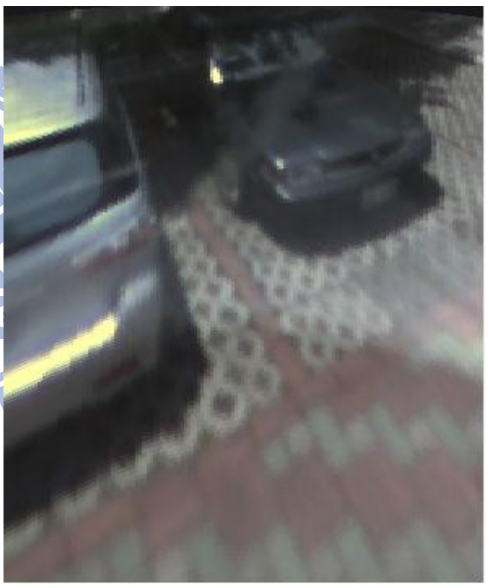
Figure 7.1 An experimental result of generating the perspective-view image. (a) An original omni-image. (b) The perspective-view image of the right-rear direction. (c) The perspective-view image of the rear direction. (d) The perspective-view image of the left-rear direction.

B. Experimental Results of Car Direction Detection and Display of Corresponding Perspective-view Images

The environment for this experiment is an open space area in a parking lot in National Chiao Tung University. We conducted the experiment of estimating the moving direction when the video surveillance vehicle was being driven on the lanes of the parking lot as well as generating the corresponding perspective-view images. We conducted the experiment for three cases: (1) turning to the right; (2) turning to the left; and (3) moving forward. Some experimental results are shown in Fig. 7.2.

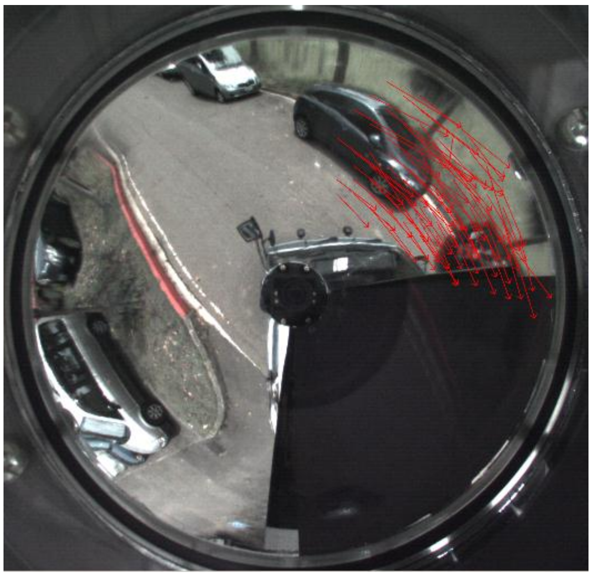


(a)

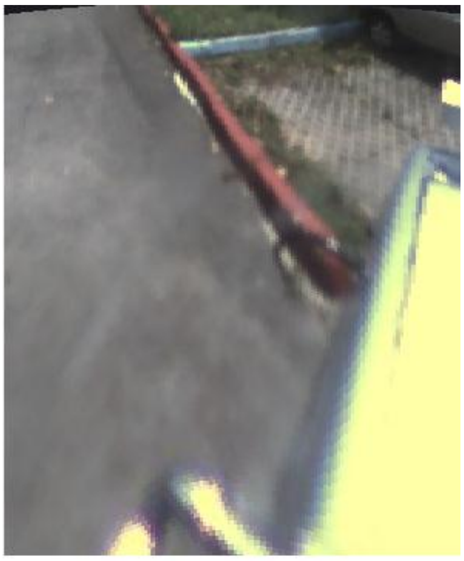


(b)

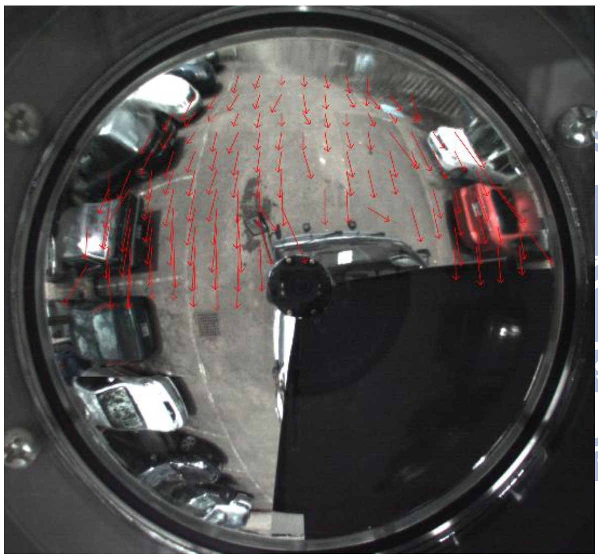
Figure 7.2 A real example of car direction detection and display of corresponding perspective-view images. (a) The case of turning to the left. (c) The case of turning to the right. (e) The case of moving forward. (b), (d), and (f) The perspective-view images corresponding to (a), (c), (e), respectively.



(c)



(d)



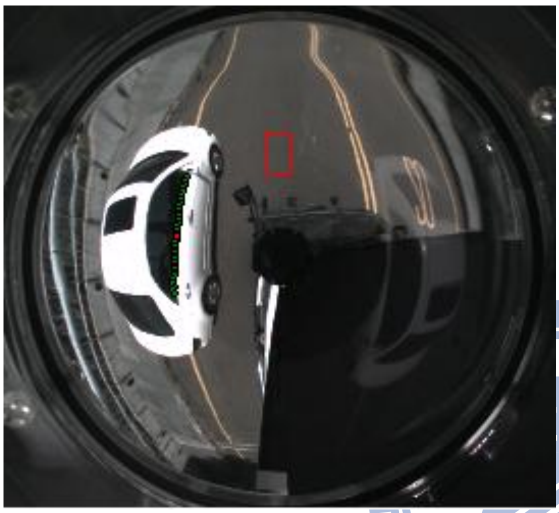
(e)

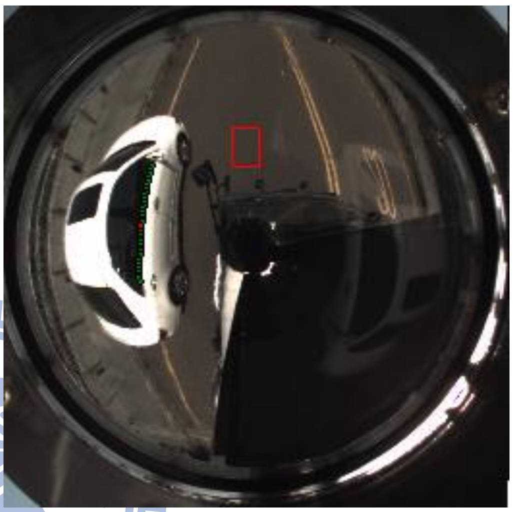
(f)

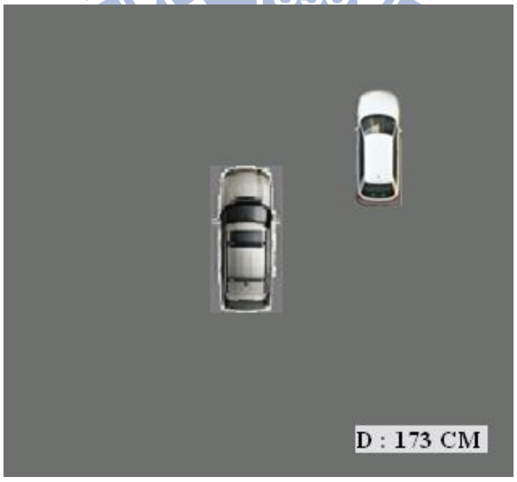
Figure 7.2 Figure 7.2 A real example of car direction detection and display of corresponding perspective-view images. (continued). (a) The case of turning to the left. (c) The case of turning to the right. (e) The case of moving forward. (b), (d), and (f) The perspective-view images corresponding to (a), (c), (e), respectively.

C. Experimental Results of Monitoring of a Nearby Car around a Static Video Surveillance Vehicle

In the experiment for monitoring a nearby car around a static video surveillance vehicle, the nearby car is a white vehicle produced by TOYOTA Company. The car was parked at the roadside and the video surveillance vehicle stopped at the car side to perform car detection. We used the omni-camera affixed on the right-front roof of the video surveillance vehicle to acquire the omni-images and conduct the detection work. An experimental result, which is the finally generated surround map with the detected nearby car seen from the top view included, is shown in Fig.7.3.


(a)


(b)



(c)

Figure 7.3 The experimental result of monitoring a nearby car around a static video surveillance vehicle. (a) The omni-image acquired with an upper camera. (b) The omni-image acquired with a lower camera. (c) The surround map from the top view. Note that the direction of an object is 180^{o} reversed in the omni-image when compared with the real situation as illustrated in (c).

D. *Experimental Results of Monitoring of a Nearby Static or Moving Car with a Moving Video Surveillance Vehicle*

The environment for this experiment is a straight lane segment in the previously-mentioned around-campus road in National Chiao Tung University as illustrated in Fig. 7.4. We divided the detection experiment into two cases. The first case was that we drove slowly the video surveillance vehicle to pass a static nearby car parked at the roadside. The proposed method described in Chapter 6 detected the nearby car and estimated the car position. Finally, a top-view surround map as shown in Fig. 7.5 was generated and displayed to the user.

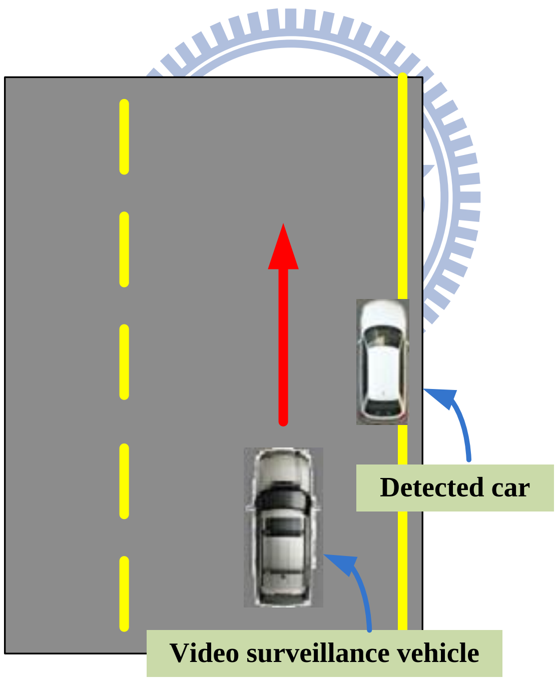
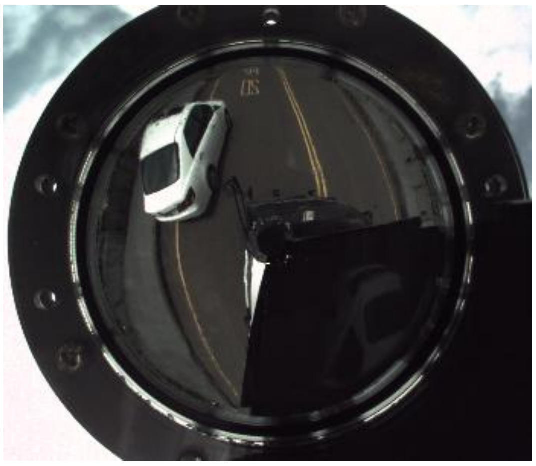
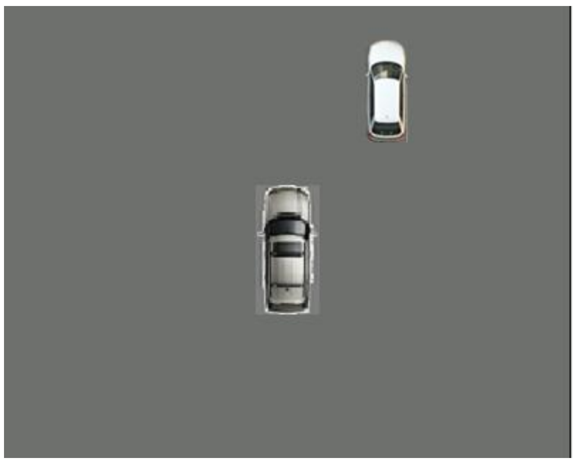
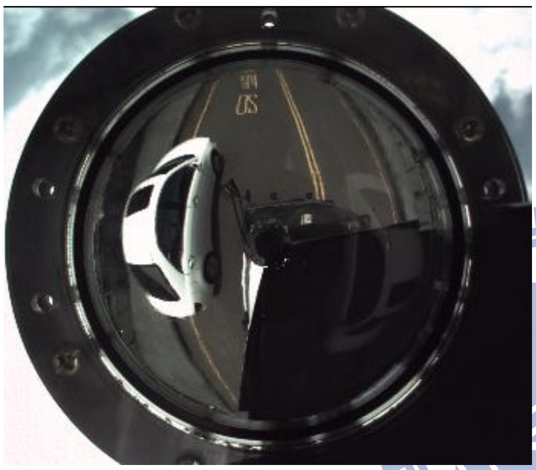
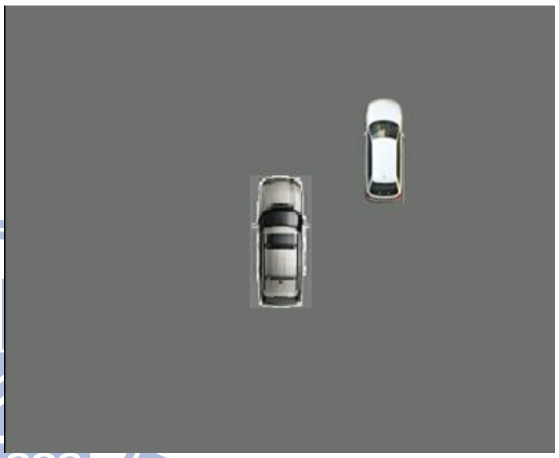


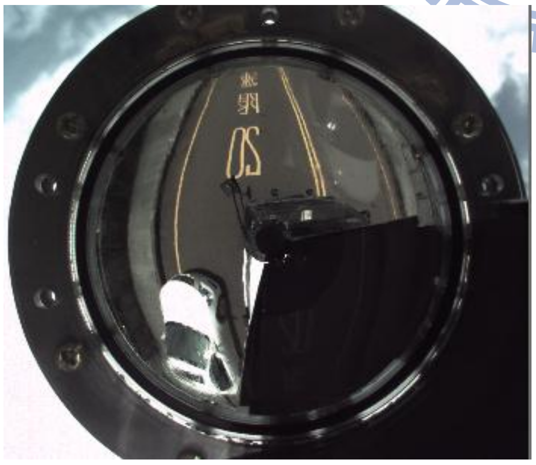
Figure 7.4 An illustration of the detecting a nearby static car with a moving video surveillance vehicle.


(a)


(b)


(c)


(d)


(e)

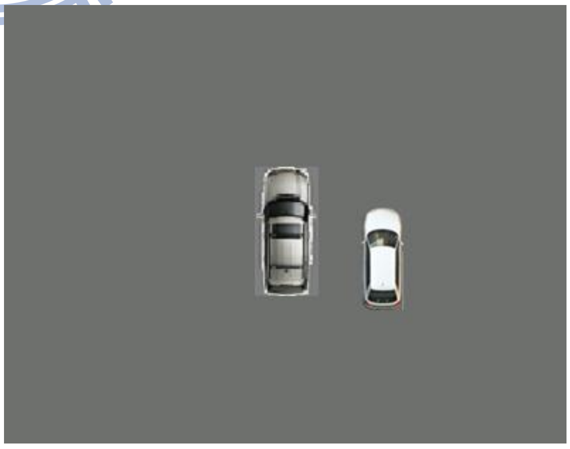

(f)

Figure 7.5 The result of a nearby static car detection with a moving video surveillance vehicle. (a)~ (f) The results of detecting a nearby car parked at the road side and the generated top-view surround maps. Note that the direction of an object is 180^o reversed in the omni-image when compared with the real situation as illustrated in (b) (d) (f).

Another detection case was that the detected car overtakes the video surveillance vehicle in a road lane. More specifically, the environment for this experiment is the same area as that for the experiment of detecting a nearby static car with a moving video surveillance vehicle mentioned previously. The passing-by car was driven on the right-hand side of the video surveillance vehicle and slowly overtook it as shown in Fig. 7.6. Fig. 7.7 shows an experimental result of detecting the nearby moving car.

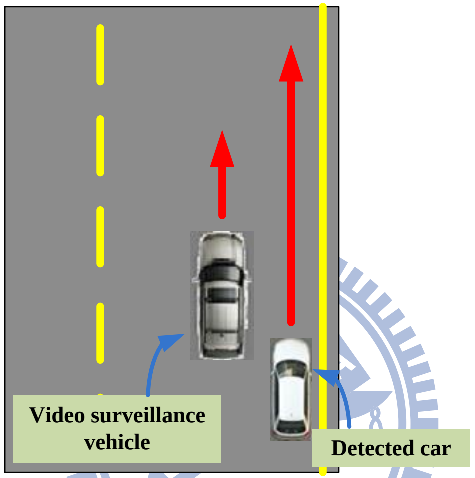


Figure 7.6 An illustration of the detecting a nearby moving car with a moving video surveillance vehicle.

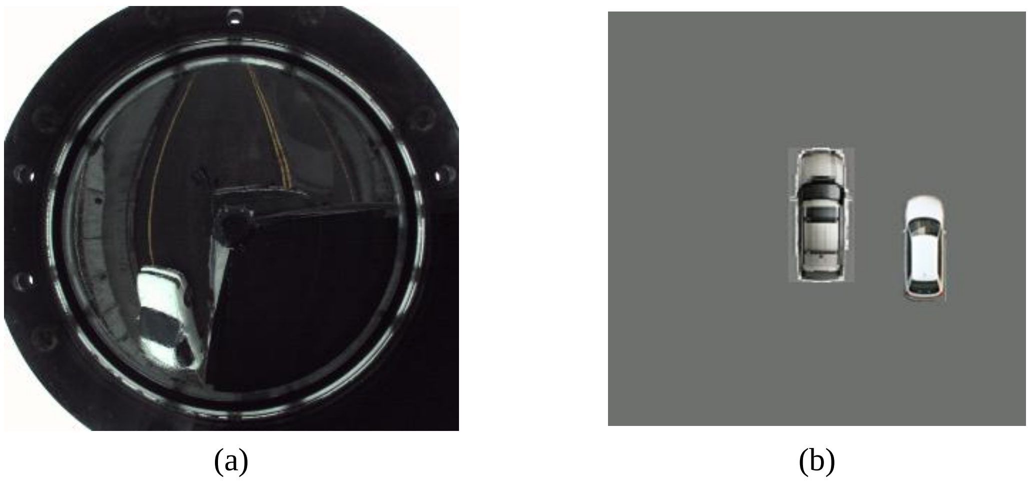
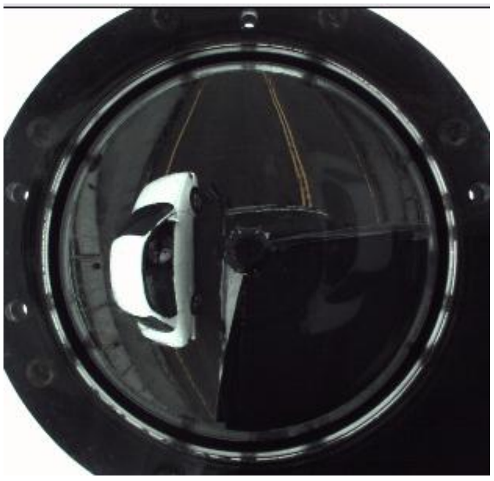
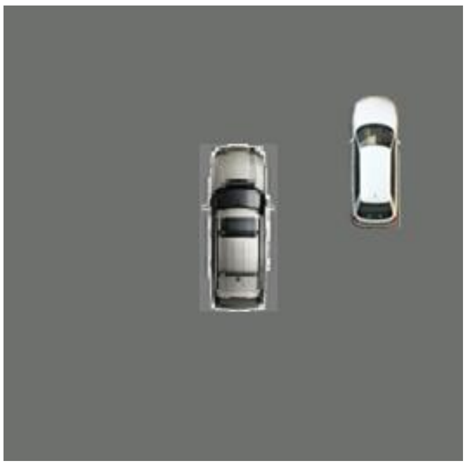


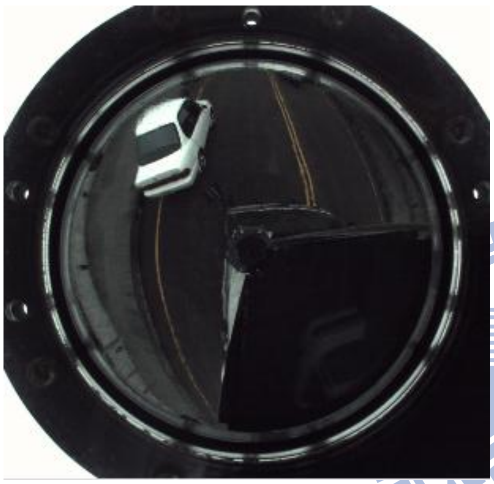
Figure 7.7 The result of a nearby moving car detection with a moving video surveillance vehicle. (a)~ (f) The result of detecting a moving car. Note that the direction of an object is 180^o reversed in the omni-image when compared with the real situation as illustrated in (b) (d) (f).



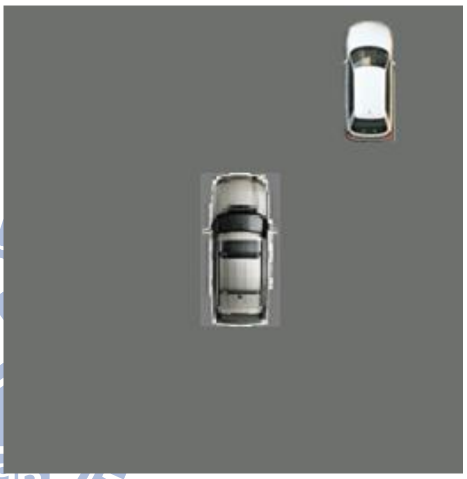
(c)



(d)



(e)



(f)

Figure 7.7 The result of a nearby moving car detection with a moving video surveillance vehicle (continue). (a)~ (f) The result of detecting a moving car. Note that the direction of an object is 180^o reversed in the omni-image when compared with the real situation as illustrated in (b) (d) (f).

7.2 Discussions

From our experiments and the results, we can see that the goal of utilizing a pair of two-camera omni-directional imaging devices equipped on the video surveillance vehicle roof to perform video surveillance of nearby cars has been achieved.

However, the proposed system still has some problems. In this study, we adopt the method of optical flow analysis to compute the motion vectors of the road surface appearing in consecutive omni-images and estimate the moving direction of the video surveillance vehicle using these motion vectors. As a result, moving objects, such as moving cars or a group of people, in the image will result in undesired motion vectors, leading to erroneous detection results of the moving direction of the vehicle itself. A possible solution is to add extra functions for detecting these unusual motion vectors and ignoring them.

Moreover, to conduct the video surveillance work at the outdoor space, the sun light is an important factor to consider. The unsuitable adjustment of the camera parameters and the shadow produced by the sun light will affect the result of the car detection experiments. A possible solution for this problem is to record typical climate conditions and the corresponding suitable camera parameters and threshold values for car shape segmentation and other image processing works. In this way, the system can be chosen appropriate data set for each climate condition at the time of nearby–car detection. Of course, it is always desired to have fully automatic method for car detection for all climate conditions.

In this study, the detected car in the experiments is a saloon car. As a result, to increase the accuracy of detecting car position, the assumption of car height is 80 cm. If the detected car is not a saloon car, we have to adjust the height parameter to make the car mask match the detected region for estimating car position.

Chapter 8
Conclusions and Suggestions for Future Works

8.1 Conclusions

In this study, a video surveillance system utilizing a pair of two-camera omni-directional imaging devices equipped on the video surveillance vehicle roof to monitor the surrounding environment has been proposed. With the advantage of mobility of the video surveillance vehicle and the wide FOV of the omni-camera system, several methods have been proposed for various purposes of driving condition monitoring and nearby car detection, as summarized in the following.

(1) *A method for speeding up generation of perspective-view images for vehicle surrounding environment monitoring* has been proposed, which, with the help of a perspective mapping table, can generate perspective-view images in realtime.

(2) *A method for analyzing the omni-images of surrounding environments for car-driving assistance* has been proposed, which, by optical flow analysis, provides the driver of the video surveillance vehicle a relevant perspective-view image of blind spots during car turning.

(3) *A method for off-line inspection of the driving history* has been proposed, by which all the sequential omni-images of the driving history can be recorded

online and displayed off-line in the form of a perspective-view image sequence with the viewing direction determined by mouse clicks.

(4) *A method for monitoring a nearby static car around a static video surveillance vehicle* has been proposed, which eliminates the ground region in acquired omni-images and detect the nearby car shape in the image by region growing and morphological techniques and displays a top-view surround map with the detected car included for inspection of its relative position.

(5) *A method of monitoring of a nearby static or moving car with a moving video surveillance vehicle* has been proposed, which uses motion vectors produced by optical flow analysis as well as color information of segmented objects to detect the nearby car shape, and uses a rectangular-shaped car model to match the detected car for estimating the car position and generating the surround map.

The experimental results shown in the previous chapters have revealed the feasibility of the proposed system

8.2 Suggestions for Future Works

According to the experience obtained this study, in the following we make suggestions of some interesting issues, which are worth further investigation in the future.

1. Increasing the speed of computation to achieve vehicle detection in realtime, e.g., by parallel computing.

2. Developing the capability of detecting and tracking multiple nearby vehicles in the surrounding environment.
3. Developing more applications of car-driving assistance using the omni-camera system, e.g., analysis of the driving behavior.
4. Adding the capability of detecting passing-by persons with a moving video surveillance vehicle.
5. Enhancing the image analysis capability to detect more information of the nearby car, e.g., the size of the vehicle.
6. Only using an omni-camera to compute the 3D data of the detected car.

References

[1] C. Micheloni, G.L. Foresti, C. Piciarelli and L. Cinque, “An autonomous vehicle for video surveillance of indoor environments,” *IEEE Transactions on Vehicular Technology*, vol. 56, no. 2, March 2007.

[2] G.L. Foresti, C. Micheloni and L. Snidaro, “Event classification for automatic visual-based surveillance of parking lots,” *Proceedings of 17th International Conference on Pattern Recognition*, vol. 3, pp. 314–317, 2004.

[3] M. Bramberger, R. P. Pflugfelder, A. Maier, B. Rinner, B. Strobl and H. Schwabach, “A smart camera for traffic surveillance,” *Proceedings of 1st Workshop on Intelligent Solutions in Embedded Systems*, pp. 153–164, Vienna, Austria, 2003.

[4] Y. Onoe, N. Yokoya, K Yamazawa, and H. Takemura, “Visual surveillance and monitoring system using an omnidirectional video camera,” *Proc. 1998 Int’l Conf. Pattern Recog*, vol. 1, pp. 588–592, Brisbane, Australia, Aug. 16-20, 1998.

[5] T. Mituyosi, Y. Yagi, and M. Yachida, “Real-time human feature acquisition and human tracking by omnidirectional image sensor,” *Proc. IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 258-263, Tokyo, Japan, July 30-Aug. 1, 2003.

[6] L. He, C. Luo, F. Zhu, Y. Hao, J. Ou and J. Zhou, “Depth map regeneration via improved graph cuts using a novel omnidirectional stereo sensor,” *Proceedings of 11th IEEE International Conference on Computer Vision (ICCV2007)*, pp. 1-8, Oct. 14-21, Rio de Janeiro, Brazil, 2007.

[7] H. Koyasu, J. Miura, and Y. Shirai, “Realtime omnidirectional stereo for obstacle detection and tracking in dynamic environments,” *Proceedings of 2001*

IEEE/RSJ Internatonal Conference on Intelligent Robots and Systems, pp. 31-36, Maui, Hawaii, USA, Oct./Nov, 2001.

[8] S. W. Jeng and W. H. Tsai, “Using pano-mapping tables for unwarping of omni-images into panoramic and perspective-view images,” *Journal of IET Image Processing*, Vol. 1, No. 2, pp. 149-155, June 2007.

[9] T. Ehlgen and T. Pajdla, “Maneuvering aid for large vehicle using omnidirectional cameras,” *IEEE Workshop on Applications of Computer Vision*, pp. 17－17, Austin, Texas, US, Feb. 2007.

[10] T. Gandhi and M.M. Trivedi, “Motion analysis for event detection and tracking with a mobile omni-directional camera,” *ACM Multimedia Systems Journal, Special Issue on Video Surveillance*, vol. 10, no. 2, pp. 131–143, 2004.

[11] N. Murakami, A. Ito, Jeffrey D. Will , Michael Steffen, K. Inoue, K. Kita, S. Miyaura, “Development of a teleoperation system for agricultural vehicles,” *Computers and Electronics in Agriculture*, vol. 63, pp. 81-88, Aug. 2008.

[12] R. Aufrȅre, J. Gowdy, C. Mertz, C. Thorpe, C.C. and T.Y. Wang, “Perception for collision avoidance and autonomous driving,” *Mechatronics*, Vol. 13, pp. 1149-1163 ,2003.

[13] Hughes, C., Glavin, M., Jones, E.and Denny, P. “Wide-angle camera technology for automotive applications: a review,” *IEEE Transactions on Intelligent Transportation Syste*ms, pp. 19－31, Mar. 2009.

[14] C. Hughes, M. Glavin1, E. Jones1 and P. Denny, “Wide-angle camera technology for automotive applications: a review,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 19－31, 2009.

[15] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” *Proc. 7th nternational Joint Conference onArtificial*

Intelligence, Vancouver, Canada, pp. 674 – 679, 1981.

[16] J. Kim and Y. Suga, “An omnidirectional vision-based moving obstacle in mobile robot,” *International Journal of Control, Automation, and Systems*, vol. 5, no. 6, pp. 663-673, Dec. 2007.

[17] S. Gupte, O. Masoud, R.F. K. Martin, and N.P. Papanikolopoulos, “Detection and classification of vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, Mar. 2002.

[18] R. Cucchiara, M. Piccardi, and P. Mello, “Image analysis and rule-based reasoning for a traffic monitoring system,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 2, June 2000.

[19] L. W. Tsai, J. W. Hsieh and K. C. Fan, “Vehicle detection using normalized color and edge map,” *IEEE Transactions on Image Processing*, vol. 16, no. 3, Mar. 2007.

[20] P. H. Yuan, K. F. Yang and W. H. Tsai, “Security monitoring around a video surveillance car with a pair of two-camera omni-directional imaging devices,” *Proceedings of 2010 Workshop on Image Processing, Computer Graphics, and Multimedia Technologies, International Computer Symposium*, pp. 325-330, Tainan, Taiwan, Dec. 2010.

[21] C. J. Wu, “New localization and image adjustment techniques using omni-cameras for autonomous vehicle applications,” *Ph. D. Dissertation*, Institute of Computer Science and Engineering, National Chiao Tung University, Hsinchu, Taiwan, Republic of China, July 2009.

[22] W. H. Tsai, “Moment-preserving thresholding: a new approach,” *Computer Vision, Graphics, and Image Processing*, vol. 29, no. 3, pp. 377-393, 1985.