國立交通大學

資訊科學系

碩 士 論 文

以環場影像作基於視覺的電腦輔助駕駛之研究

A Study on Vision-Based Computer Assisted Driving by Omni-directional Images

研 究 生：吳師毅

指導教授：蔡文祥　教授

中 華 民 國　九 十 四　年 六 月

以環場影像作基於視覺的電腦輔助駕駛之研究
A Study on Computer-Vision Assisted Driving by Omni-directional Images

研 究 生：吳師毅　　Student：Shi-Yi Wu

指導教授：蔡文祥　　Advisor：Wen-Hsiang Tsai

國 立 交 通 大 學
資 訊 科 學 系
碩 士 論 文

A Thesis
Submitted to Department of Computer and Information Science
College of Electrical Engineering and Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer and Information Science

June 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

利用電腦視覺與環場影像作車輛輔助駕駛之研究

研究生：吳師毅　　指導教授：蔡文祥博士

國立交通大學資訊科學研究所

摘 要

本論文提出了一套基於電腦視覺及環場影像來作車輛輔助駕駛的系統，此系統具備在車輛行徑時偵測車輛移動狀態以及追蹤鄰近車輛的能力。我們設計了一個自動化的學習程序取代手動的操作來擷取影像中的特徵資訊。這些資訊包含了偵測車輛移動狀態以及追蹤鄰近車輛所需之資訊。我們提出一套偵測車輛移動狀態的方法透過在學習過程中取得之特徵資訊來分析輸入的影像並且識別出目前車輛的移動狀態。另外，我們設計了一套追蹤鄰近車輛的方法來偵測並且追蹤鄰近的車輛。接下來我們提出一套使用有限狀態自動機的狀態變化來偵測危險情況的方法來偵測一些可能發生的危險情況。我們最後以成功的實驗結果證明本系統的完整性與可行性。

A Study on Vision-Based Computer Assisted Driving by Omni-directional Images

Student : Shi-Yi Wu Advisor: Dr. Wen-Hsiang Tsai

Department of Computer and Information Science
National Chiao Tung University

ABSTRACT

A vision-based computer assisted driving system with the capabilities of analyzing vehicle movements and tracking neighboring vehicles using an omni-directional camera is proposed. A vehicle with computation and image grabbing capabilities is used as a test bed. An automatic learning process is designed for extraction of the features in grabbed images without manual instructions. Through this learning process, useful features for analyzing the vehicle movement and tracking neighboring vehicles are obtained. A vehicle movement analysis method is used to detect and analyze the current vehicle movement. Also, a vehicle tracking method is proposed to detect and track neighboring vehicles surround our vehicle. Furthermore, a risk condition detection method using a finite state-transition model is proposed to detect risk conditions in car driving. Finally, experimental results showing the flexibility of the proposed methods for computer-assisted driving in outdoor environment are also included.

ACKNOWLEDGEMENTS

I am in hearty of the continuous encouragement, support, and technical guidance received from my advisor, Dr. Wen-Hsiang Tsai, not only in the development of this thesis, but also in every aspect of my personal growth.

Thanks are due to Mr. Shi-Chei Hung, Mr. Ming-Che Chen, Mr. Lien-Yi Weng, and Mr. Yuei-Cheng Chuang for their numerous discussions and suggestions. Appreciation is also given to the colleagues of the Computer Vision Laboratory in the Department of Computer and Information Science at National Chiao Tung University for their suggestions and help.

I also extend my profound thanks to my family during my school life for their lasting love, support, and encouragement. I dedicate this dissertation to my parents.

CONTENTS

LIST OF FIGURES

Chapter 1
Introduction

1.1 Motivation

More and more traffic accidents occur on roads and highways in recently years. These accidents mostly occurred due to two reasons: the limited front views of cars and the bad physical conditions of drivers. For the first reason, although three rear-view mirrors are usually equipped on a car, the driver can not monitor the entire surrounding of the car simultaneously. The driver's attention mainly falls on the front view of the car during driving. Our motivation of this study is to design a vision-based system to give assistance to drivers. We want to use computer vision techniques to monitor the surrounding of the car for the driver. It is desired the system can keep monitoring the surrounding and give warnings to the driver when dangerous conditions are detected. For the second reason, a driver with bad physical conditions, such as being drunk or feeling sleepy, will have narrower front views of the car than ordinary drivers. In this condition, the driver's ability of detecting dangerous conditions and taking appropriate actions is also down. Accidents may occur under such conditions. When an accident is going to occur, there will usually be some abnormal car movements. Thus not only the surrounding of the car should be monitored, but also the movement of the car should be tracked. If abnormal car movements are detected, the system will also give warnings to the drivers. Detection of the two conditions described above is the main goals that we want to achieve in this

study.

Nowadays, omni-directional cameras which take 360-degree panoramic views of surroundings are used in many applications, such as robotics, vehicle navigation and environment surveillance. An omni-directional camera has a wider range of field of view than a traditional camera. It can provide a full view of a vehicle's surrounding. Thus we will use an omni-directional camera to grab panoramic images of the surroundings of vehicles in this study. And then we can analyze these images to achieve the goal of computer-assisted driving.

1.2 Survey of Related Studies

Recently, omni-directional cameras have been used in more and more applications because of the advantages of panoramic images grabbed by the camera. Images grabbed by an omni-directional camera have wider ranges of field of view, as mentioned previously. But panoramic images are hard to analyze because there are more deformations in the images. In Gaspar, Winters, and Santos-Victor [1], a method was proposed to transform panoramic images grabbed from omni-directional cameras to bird's eye views by radial correction to eliminate the distortion in omni-directional images. Then they used the transformed images to conduct the application of indoor navigation. In Gandhi and Trivedi [2], a method called flat plane transformation was proposed to recover the deformations in omni-directional images. Flat plane images are easier to process. Then they used the processed flat plane images for the application of vehicle surround analysis.

In the researches of vehicle driving analysis, detection of lanes and neighboring

vehicles are major topics. There are many researches on them. In Smith et al. [4], a method for detection of traffic objects of interests was proposed. The method finds the difference between the background image and the input one. This technique is called static frame difference and is suitable for the application where merely the background changes. A vehicle tracking method was also proposed. Yim and Oh [5] proposed a method for lane detection which is based upon a measure of match similarity in a three-dimensional (3-D) space spanned by the three features of a lane boundary, namely, starting position, direction (or orientation), and gray-level intensity. Takagi, Nishi, and Yasuda [6] proposed a computer-assisted driving support method. They used an intention reasoning technique to learn dangerous conditions and the actions which should be taken to avoid the conditions. When dangerous conditions are recognized, corresponding actions are taken immediately to avoid dangers.

1.3 Overview of Proposed System

In this study, we try to design a vision-based computer-assisted driving system for use in outdoor environment. To achieve this goal, firstly an autonomous learning process that extracts parameters automatically by image processing techniques is proposed. Secondly, a vehicle movement analysis method which finds lane lines and analyzes them to know the current vehicle movement condition is also proposed. Finally, a method of detection and tracking of neighboring objects is proposed to conduct risk condition recognition.

More specifically, in the system designed in this study using these proposed techniques, the following tasks are conducted.

(1). Designing a car roof rack with adjustable width that can fit various sizes of vehicles to fix an omni-directional camera on the top, as shown in Figure 1.1.

(2). Analyzing the images grabbed by the omni-directional camera to obtain parameters that we need.

(3). Locating and finding lane lines in input images.

(4). Analyzing the lane lines to find the vehicle's moving direction.

(5). Detecting and classifying the objects of the vehicle's surrounding.

(6). Conducting the task of risk condition recognition.

The vision-based computer-assisted driving system proposed in this study consists roughly of four stages in its performance. The first stage is *outdoor environment learning*. In order to perform the task of computer-assisted driving, the system analyzes the images of outdoor environment and extracts pertinent features for the subsequent stages. The second stage is *lane line processing*. First, the system locates lane lines in each image grabbed by the camera. Then it finds the vehicle movement by analyzing the lane lines. In order to locate lane lines, pixels within the lane line regions found in the learning process are classified into two groups, namely, lane line pixels, and non-lane ones. Then a line fitting method is employed to find lines that can fit the lane line pixels. Movement of the vehicle then can be found from analyzing the features of the lane lines.

The third stage is *neighboring object detection*. In this stage, the tasks are to detect neighboring objects and classify them. The task of detecting neighboring objects can be accomplished by computing the image difference between two consecutive images. Then a merging method is employed to merge the pixels in the results of frame differencing into objects. So far what the obtained objects are is unknown yet. We classify these objects into three groups, namely, neighboring vehicles, obstacles on the

ground, and non-objects, by some properties of objects, such as width, height, and centroid. Then we track these objects in the consecutively grabbed images for the detection of risk condition which is main task of the fourth stage. Risk conditions can be categorized into two main types: conditions caused by dangerous driving behaviors and those by neighboring objects. The conditions of the first type are detected by a finite-state transition automaton in this study. The conditions of the second type are detected by the result of tracking the neighboring objects. When a risk condition is detected, a corresponding warning sound will be played to inform the driver what kind of risk condition has been detected. A flowchart of the proposed system is shown in Figure 1.2.

Figure 1.1 Car roof rack used in this study.

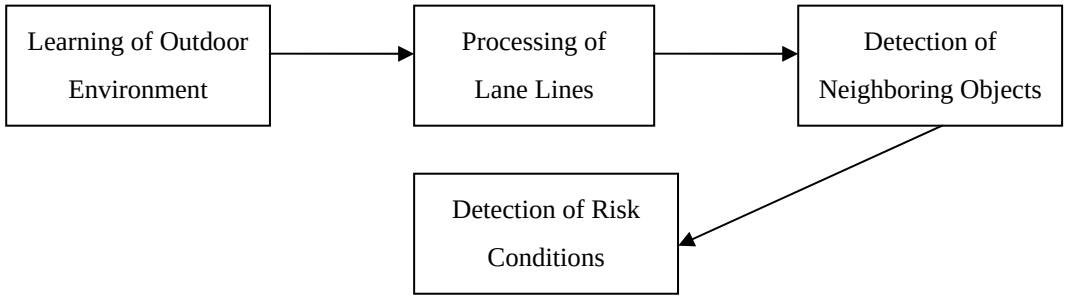


Figure 1.2 Flowchart of four stages of proposed system..

1.4 Contributions

The main contributions of this study are summarized in the following:

(1) A computer-assisted driving system using an omni-directional camera in outdoor environment is proposed.

(2) An automatic learning process to extract features of outdoor environment is proposed.

(3) A method of finding and analyzing lane lines by computer vision and image processing techniques is proposed.

(4) A method of detecting and classifying neighboring objects by image sequence processing is proposed.

(5) A method for risk condition checking for driving by a state-transition automaton model is proposed.

(6) Error tolerance techniques to reduce the influence of changing lighting conditions of outdoor environment are proposed.

1.5 Thesis Organization

The remainder of this thesis is organized as follows. The configuration of the system used in this study and introductions to outdoor environment properties are described in Chapter 2. The proposed autonomous learning process is described in Chapter 3. In Chapter 4, the proposed methods for vehicle movement analysis are described. In Chapter 5, the proposed methods for detection and tracking of neighboring objects are described. Risk condition definitions and the proposed

detection method are described in Chapter 6. Satisfactory experimental results are shown in Chapter 7. Finally, some conclusions and suggestions for future works are given in Chapter 8.

Chapter 2
System Configuration and Outdoor Driving Environment

2.1 Introduction

In this study, an omni-directional camera is used to grab the images of the surroundings of a vehicle. The main advantage of using an omni-directional camera is described in Section 2.2. And the configuration of the proposed system is introduced in Section 2.3.

The images of outdoor environment have some properties that cause them difficult to analyze by computer vision and image processing techniques. Section 2.4.1 discusses the properties of outdoor environment. Some of the properties may cause effects to degrade the accuracy of the result of image processing. The main effects are lighting and shadow, which are described in Section 2.4.2. Some techniques are developed in this study to eliminate these effects and enhance the correctness of image processing. They are described in Section 2.4.3.

2.2 Advantages of Using Omni-Directional Cameras

An omni-directional camera has a wide range of field of view, as shown in Figure 2.1. It has a 360-degree field of view horizontally and 115-degree field of view vertically. Because a traditional camera has a narrower angle of view, we need to mount multiple traditional cameras to obtain a wider angle of views. The main advantage of using an omni-directional camera is that it has a 360-degree horizontal field of view without dead space. In order to analyze the surroundings of a vehicle, the images of the front side, back side, left side, and right side of the vehicle should be grabbed. Thus we have to mount multiple traditional cameras to grab images of the four sides of the vehicle. These four traditional cameras can be replaced by a single omni-directional camera. Thus in this study, we choose to use the omni-directional camera. Figure 2.2 shows a panoramic image of a driving car in outdoor environment grabbed by an omni-directional camera on the top of the car.

Figure 2.1 Field of view of an omni-directional camera.

2.3 System Configuration

As shown in Figure 2.3, an omni-directional camera is fixed on the top of a vehicle by a car roof rack, as described in Chapter 1. The car roof rack used in this study is removable with its width adjustable from 70 cm to 95 cm, as shown in Figure 2.4, and thus the omni-camera can be used for vehicles of various sizes. We only need

to unlock the controllers at the two ends of the car roof rack; then the width can be extended or shortened manually.

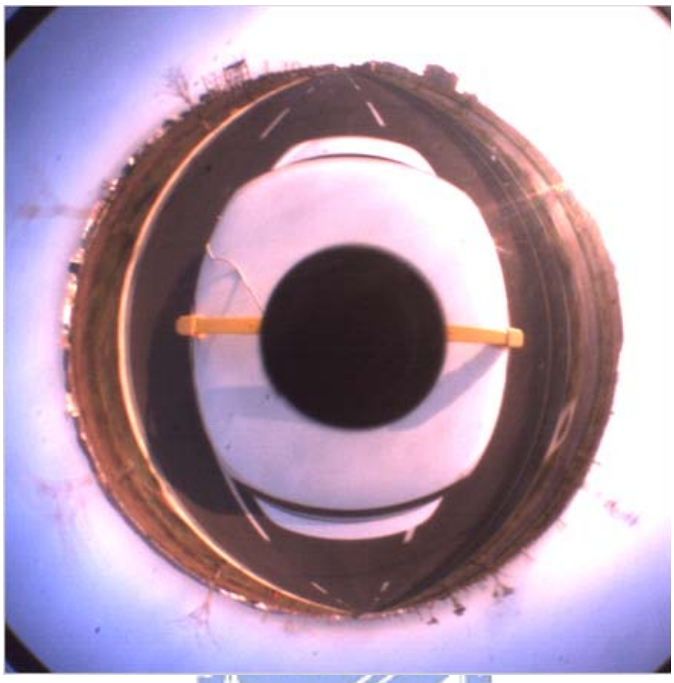


Figure 2.2 A panoramic image of surrounding of a driving car in outdoor environment.

Figure 2.3 The omni-camera used in this study.

(a)

(b)

Figure 2.4 Adjustable width of the car roof rack used in this study. (a) The minimum width of it. (b) The maximum width of it.

2.3.1 Hardware Configuration

The hardware configuration of the system used in this study contains two main parts: a vision system and a computation system. The first is a vision system which consists of an omni-directional camera as mentioned previously. The input images grabbed by the camera are of the resolution of 880×880 pixels. However, in this study we reduce the resolution of the input images to 440×440 for the reason of raising image processing speed. The frame rate of the camera is 20 frames per second. The second part is a computation system which is simply a notebook PC with a Dothan 2.0G CPU, a 512MB DDR RAM, and a 5400 rpm 80G HDD. The kernel program can be executed in this system to analyze input images and give warnings to drivers.

Images grabbed from the camera are transmitted to the computation system through a USB 2.0 port. The transmission rate of the port can be up to 50Mbps, and so the input images can be transmitted with no delay. The power supply for the camera is directly provided by the notebook PC through the USB port.

2.3.2 Software Configuration

In order to send commands to the camera and retrieve panoramic images, we use ARTCAM 130MI SDK which is developed by ARTRAY Company. This API provides a complete command interface. We can grab input images and modify image qualities such as their resolutions, RGB Gain, white balance, and other image properties by sending corresponding commands to the camera through the API. Developers can use this API as an interface to grab specific kinds of images of his/her need. We use Visual C++ as the development tool in this study.

2.4 Outdoor Driving Environment

In this section, we will discuss relevant issues about outdoor environment for car driving, including properties of outdoor environment and problems caused by them in image processing. We will discuss these problems and their corresponding solutions.

2.4.1 Properties of Outdoor Environment

Unlike indoor environment, outdoor environment contains some features that are changing constantly. For example, the lighting condition of outdoor environment

changes as time goes on. Similarly, the scenes also change while the vehicle is moving. These features are normal to outdoor environment, but they cause serious problems to the results of image processing. In order to analyze the images of outdoor environment taken during driving, some features such as lane line region, lighting condition, and the center of the vehicle in images need to be found first. Figure 2.5 shows an illustration of the above-mentioned important features of an image of outdoor environment taken during driving. The main problems caused by outdoor conditions are lighting and shadow effects, as described in the following section.

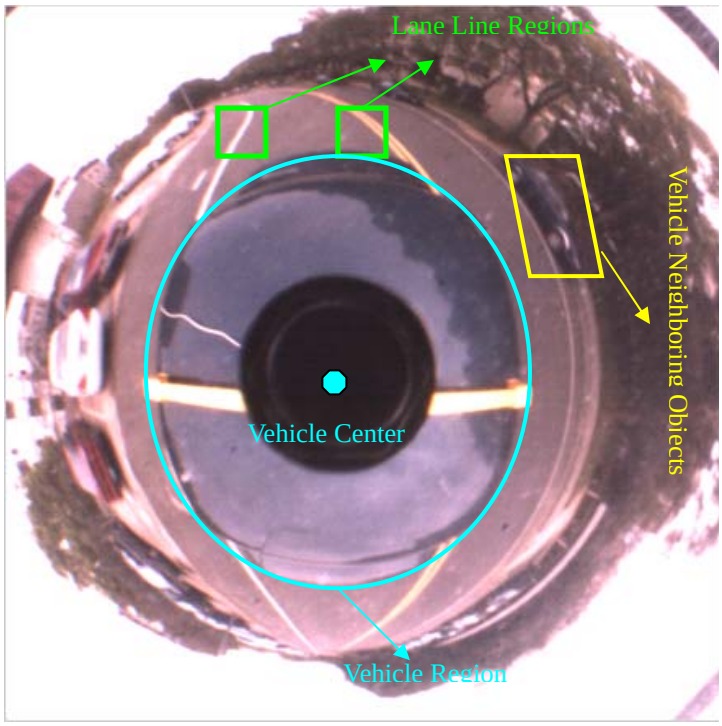


Figure 2.5 An illustration of important features of an image of outdoor environment.

2.4.2 Lighting and Shadow Effects

The lighting and shadow effects of outdoor environment cause a great influence on the results of image processing. Lighting causes the magnitudes of the RGB values

of the pixels in an image to increase abruptly, and shadow causes them to decrease. These two kinds of changes make it difficult to use a global threshold for image thresholding. Besides, lighting also causes the colors of pixels to change unpredictably. The color of a pixel means the proportions of its RGB components. If the proportions of the RGB values change, the color of a pixel will also changes. Figure 2.6(a) shows the original color of a lane line and Figure 2.6(b) shows the color of the lane line affected by lighting. In this example, the R, G, and B values of the lane line shown in Figure 2.6(a) are nearly the same. But in Figure 2.6(b), the color of the lane line is affected by lighting and the R value is higher than the G and B values. This phenomenon will cause serious problems in the lane line processing stage. Thus some techniques are developed to reduce the effects of lighting and shadow in this study, as described next.

RGB= (253,248,247)

(a)

RGB= (254,206,202)

(b)

Figure 2.6 An example of lighting effect. (a) The original color of a lane line. (b) The color of the lane line affected by lighting.

2.4.3 Elimination of Lighting and Shadow Effects

In order to eliminate lighting and shadow effects, two problems described in the previous section need to be solved. The two problems and their corresponding solutions are discussed as follows:

Problem (1). The intensity of RGB values of each pixel in the input image varies

because of lighting and shadow effects. This will cause erroneous results in lane line processing.

We can not use a global threshold for image thresholding because each region in the input image has a different lighting condition. For example, the left and right lane line regions may have totally different lighting conditions. We need to compute local thresholds for left and right lane line regions in each input image to eliminate this problem. For each input image, we thus compute respectively the thresholds of the left and right lane line regions. These local thresholds can be used in lane line processing to avoid erroneous image thresholding results.

Problem (2). The proportions of RGB values change as mentioned before.

The idea of white-balancing is used here as a solution. Before we start analyzing input images, we need to find what changes have been applied to the proportions of the RGB values of the pixels because of the lighting effect. Then we use the predefined white color with identical R, G, and B values as a reference to restore the changed white color to be the original one. In this way, errors in lane line processing can be reduced.

Chapter 3
Proposed Learning Principle and Process

3.1 Introduction

In order to perform the task of computer assisted driving in outdoor environment, some features of input images must be extracted first for the following two processes: vehicle movement analysis and detection and tracking of neighboring vehicles. These features can be categorized into three types. The first type includes the features for vehicle location. More specifically, we want to find the center of the vehicle and the region of the vehicle in each input image. These methods are described in Section 3.3.1. The second type includes the features for vehicle movement analysis. We define the lane line regions for lane line detection and find the color of lane lines. These works are described in Section 3.3.2. The third type of features includes the features for detection and tracking of neighboring objects. We define the warning region around a car and check the left and right regions for detection and tracking of neighboring objects. These works are described in Section 3.3.3.

The results of the two processes: vehicle movement analysis and detection and tracking of neighboring vehicles are easily affected by the changes of lighting conditions. We need to extract the features for the processes, as well as the features for elimination of lighting effect. In Section 3.3.4, the problem caused by lighting

conditions in outdoor environment is discussed. The detailed learning process and the feature extraction algorithm are described in the following sections.

3.2 Proposed Learning Principle and process

In order to extract features as parameters for image processing, a learning strategy is proposed, whose principle is described here. The input image for learning should contain two properties. First, the quality of the input image should be clear enough. Second, the scene of the image should not be too complicated. Input images with complicated scenes may cause the learning process to yield erroneous results. In this study, we assume that the first input image is suitable for learning which we denote as I_{learn}. Three main processes are included in the learning stage, as described above. The first process includes two main steps: the first is to find the vehicle center $C_{vehicle}$ in I_{learn}, the second is to define the vehicle region $R_{vehicle}$ in I_{learn}. The second process includes two main steps: the first is to define lane line regions R_{llane} and R_{rlane}, and the second is to find the lane line colors of the two regions. The third process includes two major steps: the first is to find the warning region R_{warn} in I_{learn} for detection and tracking of the neighboring objects and the second is to check whether the side of the vehicle is a lane or not. Figure 3.1 shows a flowchart of the learning process. The detailed processes and algorithms of each step are described in the following sections.

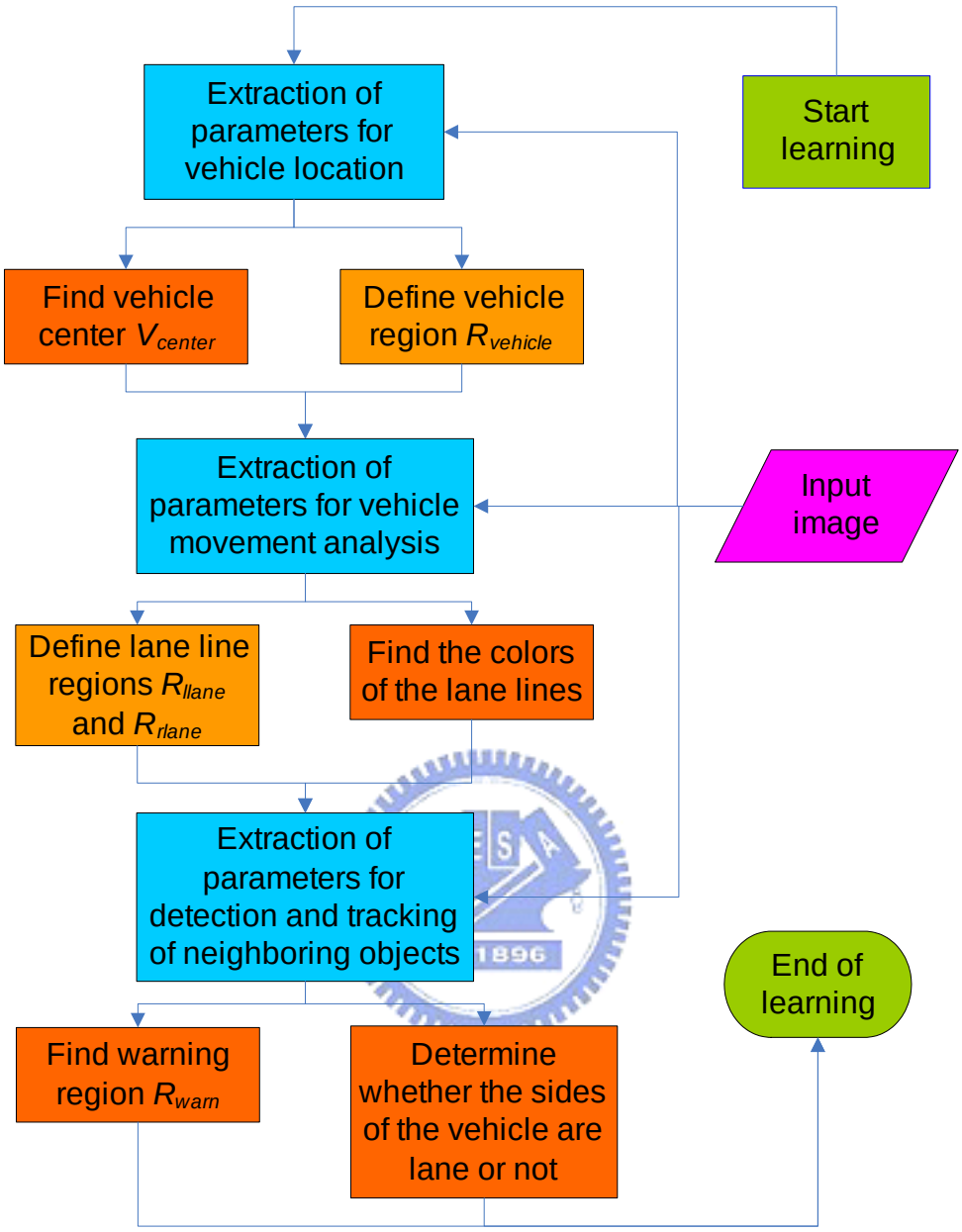


Figure 3.1 Flowchart of learning process.

3.2.1 Learning of Features for Vehicle Location

In this section, we use image processing techniques to find the features of the vehicle in I_{learn}. To find the region of the vehicle $R_{vehicle}$, the center of the vehicle $C_{vehicle}$ need to be found first. The algorithm of finding the center of the vehicle is

described as follows.

The main idea of this algorithm is to locate the vehicle center by the black circle as shown in Figure 3.2(a). Because the omni-directional camera is installed at the center of the vehicle, we can find the center of the vehicle by finding the center of the black circle. The center of the black circle can be found by locating the pixels within the black circle and computing the mean positions of the pixels. Then we can find the center of the vehicle $C_{vehicle}$. The algorithm is described as follows.

Algorithm 3.1. *Computation of the vehicle center in the input image by image processing techniques.*

Input: (1) an input image I_{learn}; (2) a square region R_a with 80 pixels in width, 80 pixels in height, and centered at the position (220,220) which is the center of the input image as shown in Figure 3.2(b).

Output: the center of the vehicle $C_{vehicle}$.

Steps:

Step 1 For all the pixels in I_{learn}, compute the intensity which is the mean of the R, G, and B values of each pixel.

Step 2 Compute the mean of the intensity of all the pixels in I_{learn} to get the mean intensity T_a.

Step 3 Use T_a as a threshold to perform thresholding in region R_a. If the intensity of the current pixel is smaller than T_a, then classify this pixel to be a circle pixel, else ignore this pixel.

Step 4 Compute the mean positions of the circle pixels found in Step 3 to get the center of the vehicle $C_{vehicle}$ as shown in Figure 3.2(c)

In Step 3, we use white color to mark the circle pixels and use black color to

mark the ignored pixels, as shown in Figure 3.2(c).

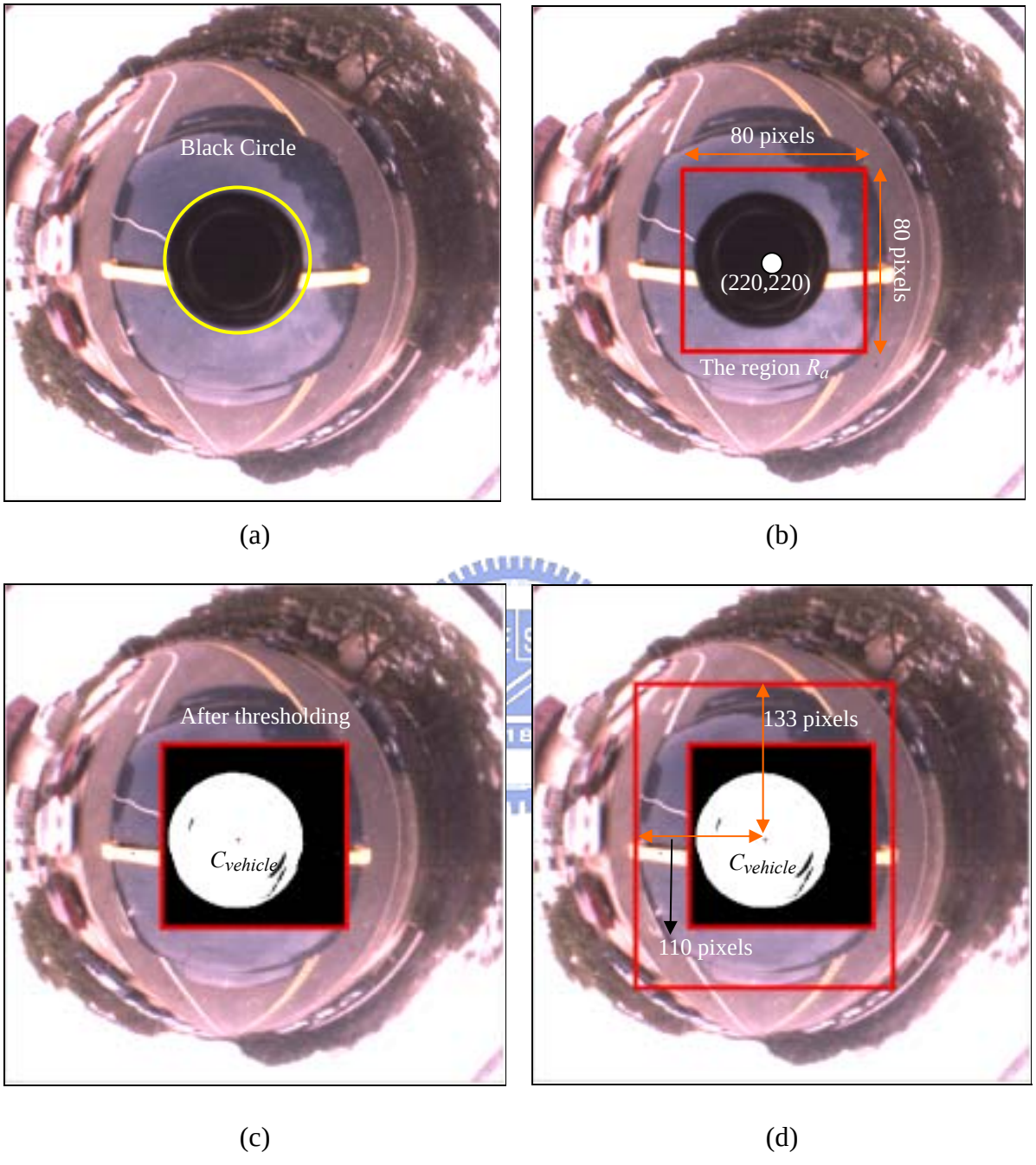


Figure 3.2 An example of learning features for vehicle location. (a) Original image I_{learn}. (b) Predefined region R_a. (c) Result of finding the center of the vehicle $C_{vehicle}$. (d) Result of finding the region of the vehicle $R_{vehicle}$.

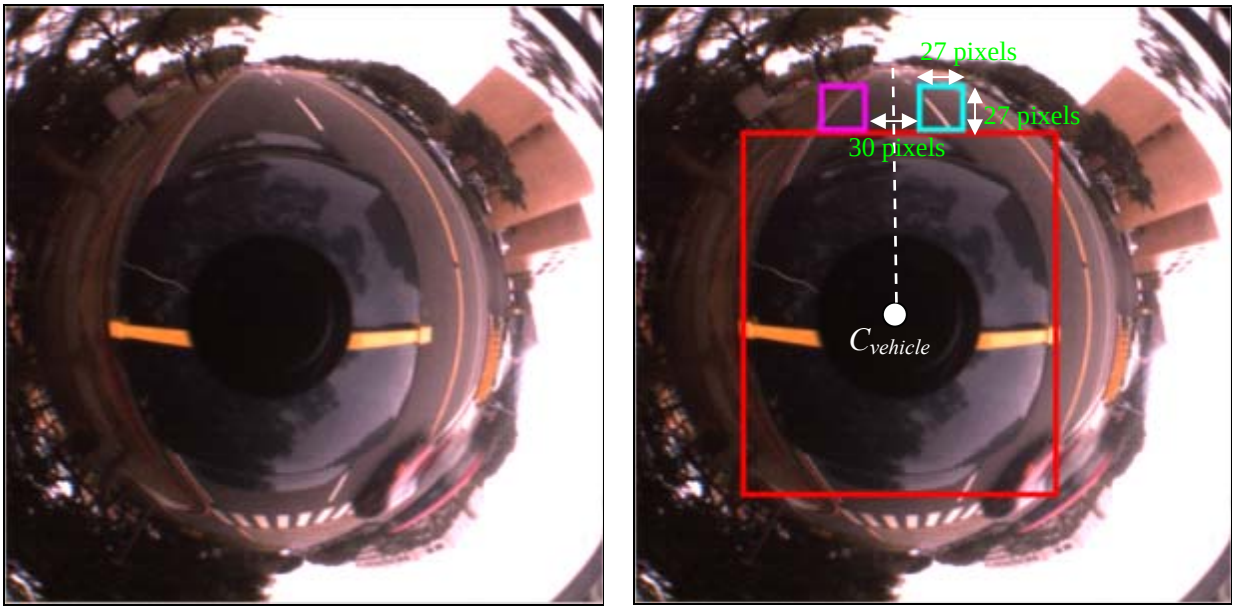
With $C_{vehicle}$ found in the above algorithm, the subsequent task is to find the

region of the vehicle $R_{vehicle}$ in the input image. We use a rectangle region $R_{vehicle}$ centered at $C_{vehicle}$ to fit the vehicle, and the size of the rectangle region varies with the size of the vehicle. In this study, the rectangle is taken to be 220 pixels in width and 266 pixels in height. Figure 3.2(d) shows an example of finding the region of the vehicle in the input image.

3.2.2 Learning of Features for Vehicle Movement Analysis

In this section, we describe how to find the features for vehicle movement analysis, including the left lane line region R_{llane} obtained by left lane line detection and the right lane line region R_{rlane} obtained by right lane line detection. These regions can be defined from $C_{vehicle}$ and $R_{vehicle}$. The lane line regions are 27 pixels in width and 27 pixels in height. They are symmetric with $C_{vehicle}$. The left lane line region is 15 pixels to the left from $C_{vehicle}$ and the right lane region is 15 pixels to the right from $C_{vehicle}$. Figure 3.3 shows an example of finding R_{llane} and R_{rlane} in the input image.

After finding the regions R_{llane} and R_{elane}, the subsequent task is to detect the lane line colors of the two lane lines. Normally a lane line may be of one of three colors: white, yellow, and red, as shown in Figure 3.3. However, the red lane line is seldom used; we only discuss the white and yellow lane line colors in this study. If we can find out the colors of the two types of lane lines, we may detect these lane lines by their corresponding colors. A lane line color detection algorithm is proposed as follows.



(a) (b)

Figure 3.3 An example of finding R_{llane} and R_{rlane}. (a) Original Image I_{learn}. (b) Result of finding R_{llane} and R_{rlane}.

The main idea of this algorithm is to find the numbers of pixels of white and yellow lane line colors for each lane line region by using the color of each pixel. Then we can compare the numbers of pixels of the two colors in each lane line region to determine the color of the lane line region.

Algorithm 3.2. *Detection of left and right lane line colors.*

Input: (1) an input image I_{learn}; (2) lane line regions R_{llane} and R_{rlane}.

Output: the color of the two lane line regions.

Steps:

Step 1. For each pixel in R_{llane} and R_{rlane}, transform the R, G, and B values of the pixel into the HSI (hue, saturation, and intensity) color model.

Step 2. For each pixel in R_{llane} and R_{rlane}, classify the pixel by its HSI components into three groups, namely, white lane line pixel group, yellow lane line pixel

group and non-lane line pixel group.

Step 3. For R_{llane} and R_{rlane}, compare the numbers of pixels of the three groups. If the number of pixels of the white lane line pixel group is larger than the yellow group, determine the color of this lane region to be white, else determine it to be yellow.

In Step 1, we transform the RGB color model into the HSI color model by the following formula:

$$\begin{cases} H = \begin{cases} \theta & \text{if B} \le \text{G} \\ 360 - \theta & \text{if B} > \text{G} \end{cases} \\ \theta = \cos^{-1}\left\{ \dfrac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\left[(R-G)^2+(R-B)(G-B)\right]^{\frac{1}{2}}} \right\}. \\ S = \left\{ 1 - \dfrac{3}{(R+G+B)}\left[\min(R,G,B)\right] \right\} \times 180. \\ I = \dfrac{1}{3}(R+G+B). \end{cases}$$

Hue is a color attribute that describes a pure color (pure yellow, orange, or red), whereas saturation gives a measure of a degree to which a pure color is diluted by white light. Intensity is a subjective descriptor [12]. The hue component contains the color information of the R, G, and B components. Thus in Step 2 we only need to compare the hue component of a pixel with that of the predefined white and yellow lane line colors. We can do this by the use of a measure the similarity between them. If a pixel has its color similar to the white lane line color, this pixel will be classified to white lane line pixels group. Otherwise, if its color is similar to the yellow lane line color, it will be classified to be in the yellow lane line pixels group. Finally we can determine the colors of the lane lines by comparing the numbers of pixels in the three groups.

3.2.3 Learning of Features for Detection and Tracking of Neighboring Objects

In this section, we describe how to extract the features for detection and tracking of neighboring objects. These features include: warning region R_{warn}, the region for detection and tracking of the neighboring vehicles and the information of whether the side of the vehicle is a lane or not. The extraction procedure is described as follows.

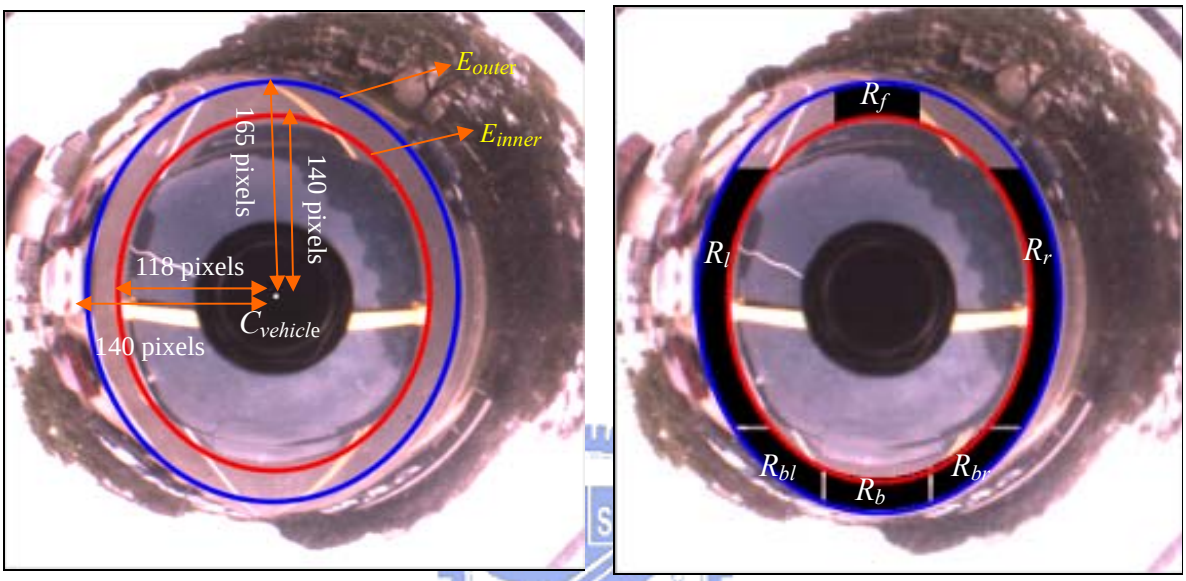
The feature of the warning region R_{warn} is set to be the strip between two ellipses, as shown in Figure 3.4(a). The red ellipse E_{inner} is defined to be that fits the vehicle in the input image, which is 118 pixels in width and 140 pixels in height. The blue ellipse E_{outer} is defined to be the maxima region for detection and tracking of neighboring objects, which is 140 pixels in width and 165 pixels in height. The equations of the two ellipses are as follows:

$$E_{outer} : b_o^2 \times (x-m)^2 + a_o^2 \times (y-n)^2 \le a_o^2 \times b_o^2 ;$$
$$E_{inner} : b_i^2 \times (x-m)^2 + a_i^2 \times (y-n)^2 \ge a_i^2 \times b_i^2 .$$

In these two equations, m and n are the x coordinate and the y coordinate of the vehicle center $C_{vehicle}$. The value of a_o is 140 pixels and b_o is 165 pixels. The value of a_i is 118 pixels and b_i is 140 pixels.

We divide R_{warn} into six regions, namely, the front region R_f, back region R_b, left region R_l, right region R_r, blind region of left R_{bl}, and blind region of right R_{br}, as shown in Figure 3.4(b). Normally, people only watch the four regions: R_f, R_b, R_l and R_r during driving. R_{bl} and R_{br} are dead spaces of a driver's eyesight. Many traffic accidents resulted from paying no attention to these two regions. Figure 3.5 shows an explanation of dead spaces of a driver's eyesight. The regions of gray color are dead space of driver's eyesight. In the four regions of R_1 to R_4, R_3 and R_4 are the regions

that the driver's eyesight can not reach. R_1 and R_2 are just two small regions in front of the vehicle, so we do not take them into consideration. We add these two regions into our warning regions. These six regions will be monitored in detection and tracking of neighboring objects.



(a) (b)

Figure 3.4 An example of defining warning region R_{warn}. (a) Result of finding R_{warn} by using two ellipses. (b) Result of dividing R_{warn} into six regions.

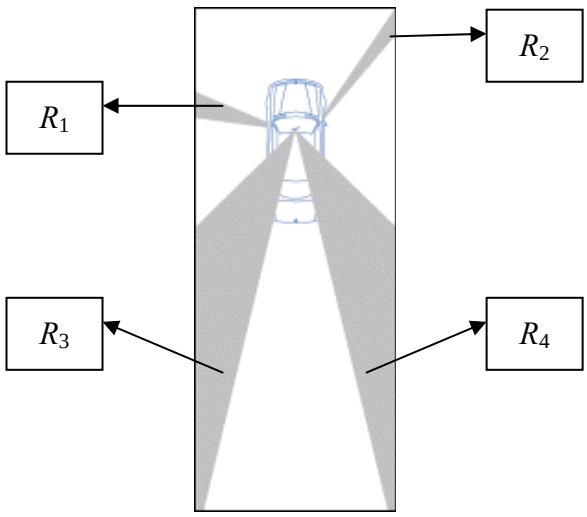


Figure 3.5 An explanation of dead space of driver's eyesight.

Another feature to find is the information of whether the side of the vehicle is lane or not. An algorithm for obtaining such information of decision is proposed here. Normally, most pixels in the front region R_f are ground pixels as shown in Figure 3.4(b). We can find the representative color of the front region R_f as a reference color of the ground. The main idea of this algorithm is to compare the representative color of the regions R_r and R_l with the representative color of R_f. If R_f and R_l are similar enough, the left side of the vehicle is decided to be a lane. Else, it is regarded not to be so. The process of checking the right lane is the same as that of the left lane. The algorithm is described as follows.

Algorithm 3.3. *Determination of whether the two sides of the vehicle are lanes or not.*

Input: (1) a input image I_{learn}; (2) the warning regions R_f, R_l and R_r.

Output: determination results of the two sides of the vehicle.

Steps:

Step 1. For all the pixels in R_f, count the mean of R, G, and B values of each pixel. Denote the mean of the R values as R_{fmean}, that of the G values as G_{fmean}, and that of the B values as B_{fmean}.

Step 2. For the all pixels in R_l and R_r, repeat Step 1 and compute the parameters for each region. In R_l, compute the parameters R_{lmean}, G_{lmean}, and B_{lmea}. In R_r, compute the parameters of R_{rmean}, G_{rmean}, and B_{rmean}.

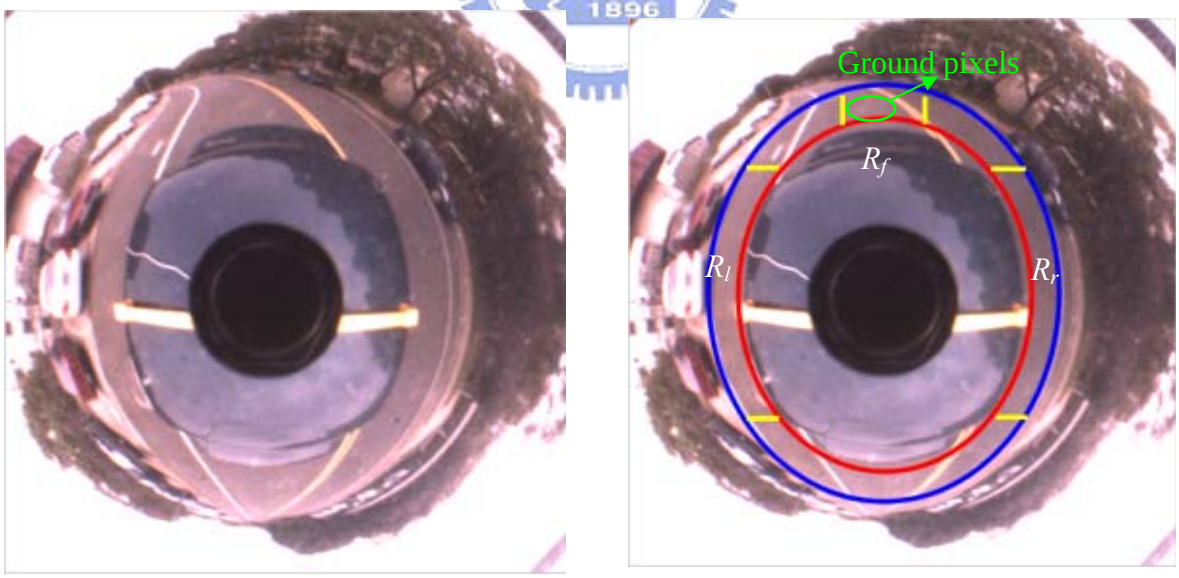
Step 3. Transform the means of the R, G, and B values of the three regions into the HSI color model. In R_f, denote the HSI components as H_g, S_g, and I_g. In R_l, denote the HSI components as H_l, S_l, and I_l. In R_r, denote the HSI components as H_r, S_r, and I_r.

Step 4. Compute the similarity between the parameters of R_f and the parameters of

R_l and R_r to determine whether the side of the vehicle is a lane or not. The formula for decision is as follows:

$$if: \left(\left|H_g - H_l\right|\right) + \left(\left|S_g - S_l\right|\right) + \left(\left|I_g - I_l\right|\right) < T_s \Rightarrow \text{ left side of the vehivle is a lane.}$$
$$if: \left(\left|H_g - H_r\right|\right) + \left(\left|S_r - S_r\right|\right) + \left(\left|I_g - I_r\right|\right) < T_s \Rightarrow \text{ right side of the vehivle is a lane.}$$

In the above algorithm, we determine whether the left side of the vehicle is a lane or not by computing the sum of the differences of the hue component, the saturation component, and the intensity component between R_f and R_l. The sum represents the color dissimilarity between the two regions. If the sum is smaller than a threshold T_s, it means that the color of the left region of the vehicle is similar enough to the one of the front region. Then we determine the left side of the vehicle as a lane. Else we will not determine it to be so. The method for doing so for the right side of the vehicle is similar. Figure 3.6 shows an example of determining whether a side of the vehicle is a lane or not.



(a) (b)

Figure 3.6 An example of determining whether a side of the vehicle is a lane or not. (a) Original Image. (b) Find parameters of R_f, R_l and R_r. Parameters of R_f: H_g=350, S_g=20, I_g=138; parameters of R_l: H_l=347, S_l=22, I_l=151; parameters of R_r: H_r=339, S_r=20, I_r=122.

3.2.4 Problem Caused by Lighting in Learning Process

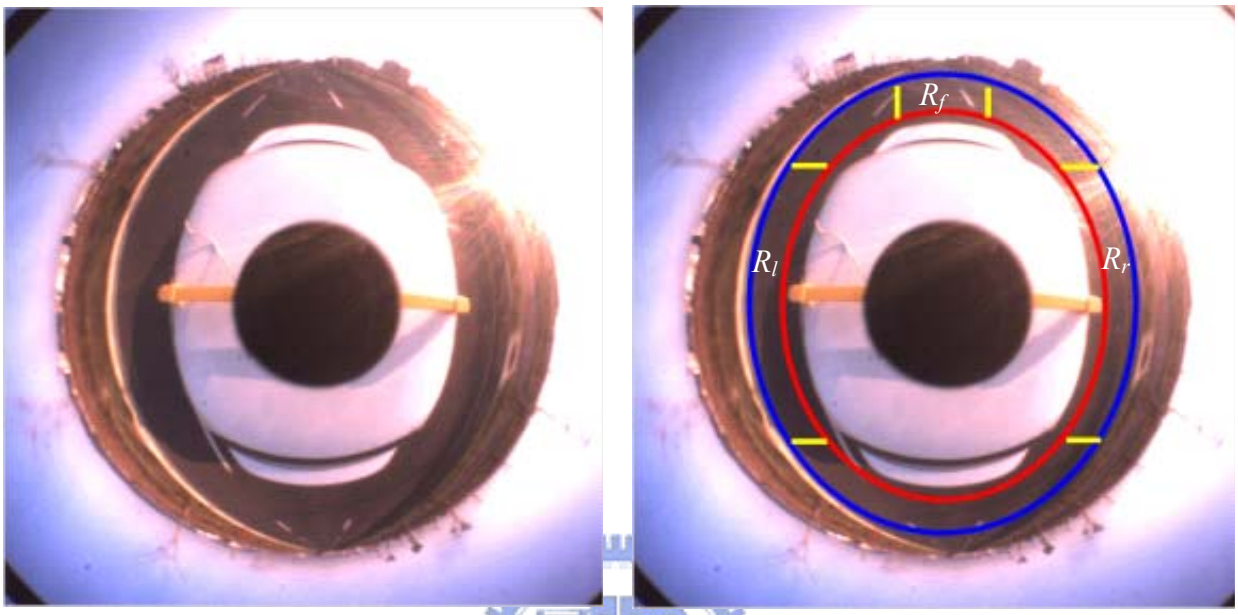
All the processes of learning have been described in the previous sections. But some of the processes will yield erroneous results because of the lighting conditions, especially the process of determining whether a side of the vehicle is a lane or not. The pixels within R_f, R_l, and R_r will be affected by lighting condition. The R, G, and B values of a pixel change and so do the hue, saturation, and intensity components of the pixel. Then the process will produce erroneous results. To overcome this problem, we need to find the variations of the HSI components of a pixel affected by lighting. From our observance, we find that lighting makes the hue and saturation components of a pixel to increase slightly and the intensity component to increase greatly. Thus the previously-described determination formula is modified as follows:

$$\begin{cases} 1.if: \left(\left|H_g - H_l\right|\right) + \left(\left|S_g - S_l\right|\right) + \left(\left|I_g - I_l\right|\right) < T_s \\ 2.if: \left| \frac{\left(\left|H_g - H_l\right|\right)}{2} - \left(\left|S_g - S_l\right|\right) \right| < T_p \end{cases} \Rightarrow \text{ left side of the vehivle is a lane.}$$

$$\begin{cases} 1.if: \left(\left|H_g - H_r\right|\right) + \left(\left|S_g - S_r\right|\right) + \left(\left|I_g - I_r\right|\right) < T_s \\ 2.if: \left| \frac{\left(\left|H_g - H_r\right|\right)}{2} - \left(\left|S_g - S_r\right|\right) \right| < T_p \end{cases} \Rightarrow \text{ right side of the vehivle is a lane.}$$

Each formula includes two conditions. Condition 1 is used for normal input images. Condition 2 is used for input images that are affected by lighting. In condition 2, we ignore the intensity component because this component changes abruptly if the pixel is affected by lighting. We use only the hue and saturation components in the formula. In this way we can reduce the erroneous results caused by lighting. Figure 3.7 shows an example of determining whether the two sides of the vehicle are lanes or

not with the input image affected by lighting. The result of this example is that the two sides of the vehicle are both lanes.



(a) (b)

Figure 3.7 Determining whether sides of the vehicle are lanes or not with the input image affected by lighting. (a) Image affected by lighting. (b) Find parameters of R_f, R_l and R_r. Parameters of R_f: H_g=346, S_g=21, I_g=107; parameters of R_l: H_l=341, S_l=20, I_l=62; parameters of R_r: H_r=358, S_r=27, I_r=99.

Chapter 4
Vehicle Movement Analysis

4.1 Introduction

In order to detect the vehicle movement for the current input image, we have to find the lane lines first. Then we can detect the vehicle movement by analyzing the obtained lane lines. In this chapter, a lane detection process described in Section 4.2 is proposed to find the left and right lane lines within the lane line regions defined in the learning process. In Section 4.3, a vehicle movement detection process is proposed to find the directions of the two lane lines and combine them into a vehicle movement direction. Then we can find the vehicle movement for the current input image by analyzing the vehicle movement direction. A vehicle movement analysis process is proposed, as described in Section 4.4.

The results of the lane detection process and the vehicle movement detection process are easily affected by the noise in input images. Such noise may come from the vibration of the camera while the vehicle is moving, or may be produced by the objects on the ground with colors similar to those of the lane lines. Such noise in input images will cause erroneous results of the two proposed processes. Some error tolerance techniques are developed in this study, as described in Section 4.3.3. These techniques can not only detect and ignore the erroneous results, but also reduce the error rate of the two processes.

4.2 Image Processing for lane line detection

In this section, a lane detection process is proposed. After the learning process described in the previous chapter, the two lane line regions R_{llane} and R_{rlane} were found. In this section, we use a thresholding method to find the pixels of the lane lines within regions R_{llane} and R_{rlane}. But we can not use a global threshold to find the left and right lane lines because the two lane line regions may have different lighting conditions and the lighting conditions of each input image is also different. Thus we use two thresholds respectively for the two lane line regions and the thresholds are designed to vary with the lighting conditions of each input image. Figure 4.1 shows a flowchart of the lane detection process.

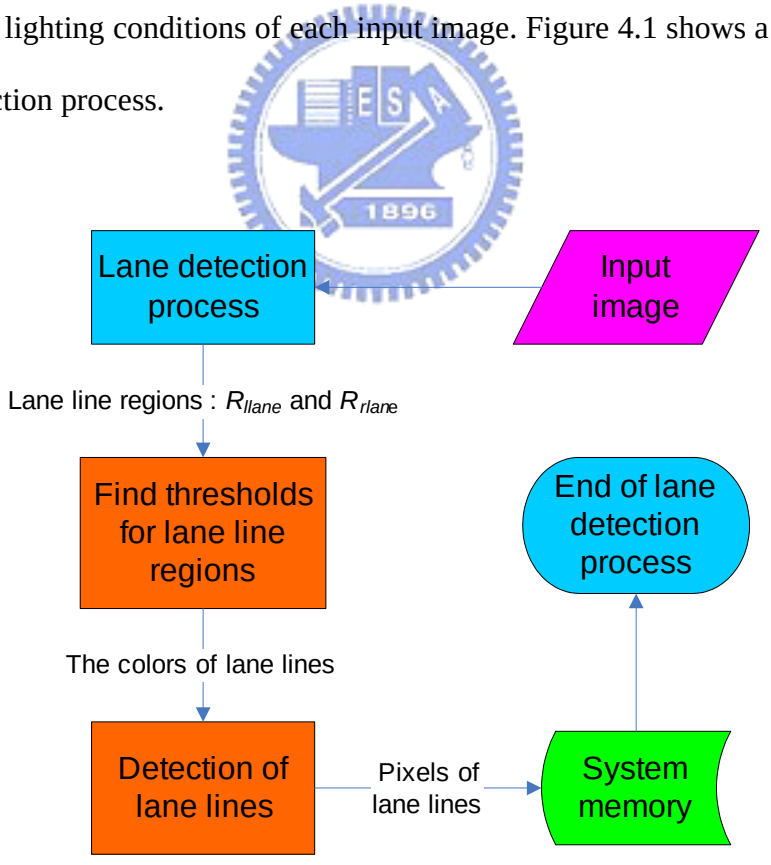


Figure 4.1 Flowchart of lane detection process.

4.2.1 Preprocessing

Before we use the thresholding method to find the lane lines in the lane line regions R_{llane} and R_{rlane}, the threshold for each region should be found first. For each lane line region, two thresholds need to be found which are the threshold T_w for finding the white lane line and the threshold T_y for finding the yellow lane line. In more detail, T_{lw} and T_{ly} are the thresholds for the left lane line region, T_{rw} and T_{ry} are the thresholds for the right lane line region. These thresholds are found by the following steps.

Step 1. Compute the intensity of each pixel within the two lane line regions R_{llane} and R_{rlane}, denoted as I_{xy}.

Step 2. For each lane line region, compute the mean intensity of the pixels within the region, denoted as M_{llane} and M_{rlane}.

Step 3. Use M_{llane} and M_{rlane} to compute the thresholds for the two lane line regions by the following equation:

$$\left.\begin{aligned} T_{lw} &= M_{llane} + T_{white} \\ T_{ly} &= M_{llane} + T_{yellow} \end{aligned}\right\} \text{ the thresholds for the left lane line region.}$$

$$\left.\begin{aligned} T_{rw} &= M_{rlane} + T_{white} \\ T_{ry} &= M_{rlane} + T_{yellow} \end{aligned}\right\} \text{ the thresholds for the right lane line region.}$$

We use the mean intensities of each lane line region as a threshold. Because most of the pixels within the lane line regions are ground pixels, the M_{llane} and M_{rlane} found In Step 2 represent the mean intensities of the ground pixels within the lane line regions. We can take them as thresholds to classify ground pixels and lane line pixels. The intensities of the colors of white and yellow lane lines are higher than the intensity of the ground color. And the intensities of the white and yellow colors of

lane lines are also different. For the threshold of the white lane line color, we add an offset T_{white} to M_{llane} and M_{rlane}. And for the threshold of the yellow lane line color, we add another offset T_{yellow} to M_{llane} and M_{rlane}. Then all the four thresholds can be computed from the above equation.

4.2.2 Dynamic Image Thresholding for Lane Line Detection

In this section, we describe how we find the lane lines in each input image. After find the two thresholds for the two lane line regions, we use a thresholding method to find the lane lines in the lane line regions. A method for this purpose is proposed in this section. The main idea of this method is to measure the similarity between the color of each pixel within the lane line regions and the color of the lane lines. If the similarity of a pixel is large enough, the pixel is classified to be a pixel of lane lines. Otherwise, this pixel is classified to be a pixel of non-lane lines. A detailed algorithm of the proposed method is described as follows.

Algorithm 4.1. *Detection of Lane lines.*

Input: (1) an input image, I_{input}; (2) lane line regions, R_{llane} and R_{rlane}; (3) the thresholds for the left and right lane line regions, $T_{lw,}$, T_{ly}, T_{rw}, and T_{ry}.

Output: (1) two sets of lane line pixels for left and right lane line regions, respectively; (2) the positions of the two lane lines, P_{llane} and P_{rlane}.

Steps:

Step 1. Choose a proper threshold for each lane line region by the colors of the lane lines found in the learning process. For example, if the color of the left lane line is white, T_{lw} is chosen to be the threshold of left lane line region T_{llane}.

Otherwise, T_{ly} is chosen.

Step 2. For each pixel within R_{llane} and R_{rlane}, compute the hue component, H_{xy}, saturation component, S_{xy}, and intensity component, I_{xy}, from the R, G, and B values using the formula described in Section 3.2.2.

Step 3. Classify the pixels within R_{llane} and R_{rlane} into two groups, namely, the lane line pixel group, and the non-lane line pixel group by the similarity between the color of each pixel and the color of the lane line.

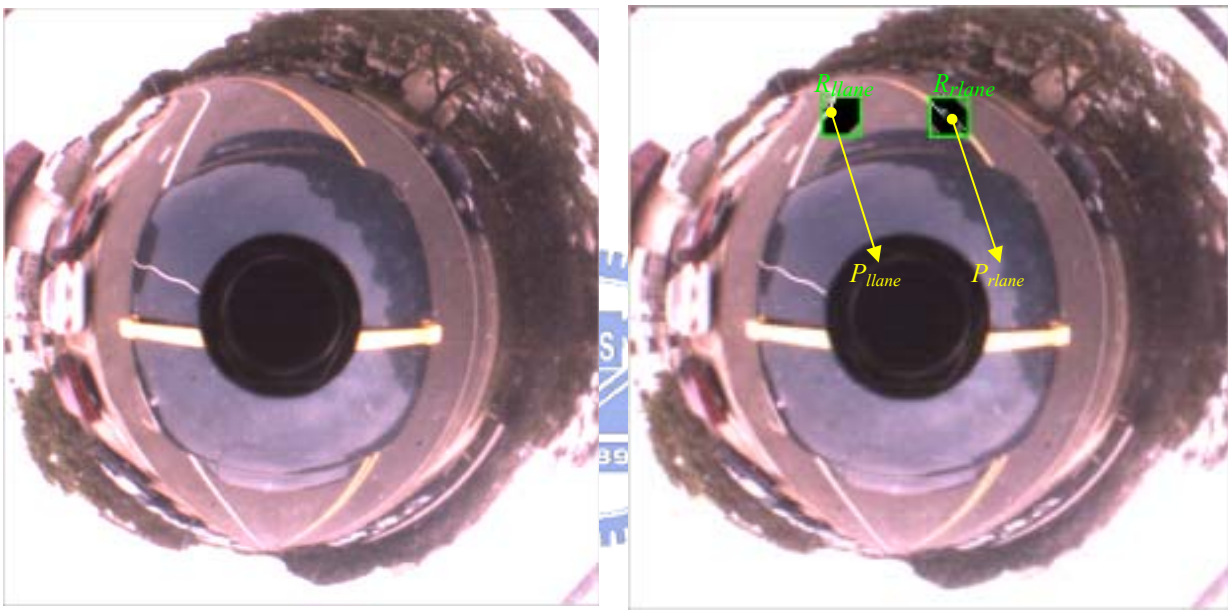
Step 4. For each lane line pixel group in R_{llane} and R_{rlane}, compute the mean positions of the lane line pixels to get the mean positions of the left and right lane lines, denoted as $P_{llane}(X_{llane}, Y_{llane})$ and $P_{rlane}(X_{rlane}, Y_{rlane})$.

In the above algorithm, we classify each pixel in R_{llane} and R_{rlane} by two conditions. One condition is whether the intensity component I_{xy} is larger than the threshold of the lane line region or not. The other condition is whether the hue and saturation components are similar to the ones of the lane line. If the above two conditions are satisfied, the pixel is classified to be a lane line pixel. For example, if the H_{xy} of a pixel in R_{llane} falls between H_{lmin} and H_{lmax} and the S_{xy} falls between S_{lmin} and S_{lmax}, the pixel is classified to be a lane line pixel. The values of H_{lmin}, H_{lmax}, S_{lmin}, and S_{lmax} depend on the color of the left lane line. If the color of the left lane line is white, the values of H_{lmin}, H_{lmax}, S_{lmin}, and S_{lmax} are the minimum and maximum values of the hue and saturation components of the predefined white color. The classification formulas are as follows.

$$\text{For each pixel in } R_{llane}: \begin{cases} 1.\ I_{xy} \geq T_{llane} \\ 2.\ \begin{cases} H_{l\min} < H_{xy} < H_{l\max} \\ S_{l\min} < S_{xy} < S_{l\max} \end{cases} \Rightarrow \text{The pixel is a left lane line pixel.} \end{cases}$$

$$\text{For each pixel in } R_{rlane}: \begin{cases} 1.\ I_{xy} \geq T_{rlane} \\ 2.\ \begin{cases} H_{r\min} < H_{xy} < H_{r\max} \\ S_{r\min} < S_{xy} < S_{r\max} \end{cases} \Rightarrow \text{The pixel is a right lane line pixel.} \end{cases}$$

Figure 4.2 shows an example of finding the pixels of the lane lines. In Figure 4.2(b), the pixels of white color in R_{llane} and R_{rlane} are the lane line pixels and the pixels of black color are non-lane line pixels. We can use these lane line pixels to perform the vehicle movement detection in the following process. P_{llane} and P_{rlane} are the mean positions of the lane line pixels and they will be used by the proposed error tolerance techniques.



(a) (b)

Figure 4.2 An example of finding the pixels of the lane lines. (a) Original image. (b) Results of finding the pixels of the lane lines.

4.3 Computation of Vehicle Movement Parameters

In this section, a vehicle movement detection process is proposed. After

classifying the pixels in R_{llane} and R_{rlane} into the lane line pixels group and the non-lane line pixels group, we can use the left and right lane line pixels to find the lane line directions. After finding the left and right lane line directions, we can combine them into the vehicle movement direction. In addition, some error tolerance techniques are included to prevent noise from causing erroneous results. Figure 4.3 shows a flowchart of the vehicle movement detection process.

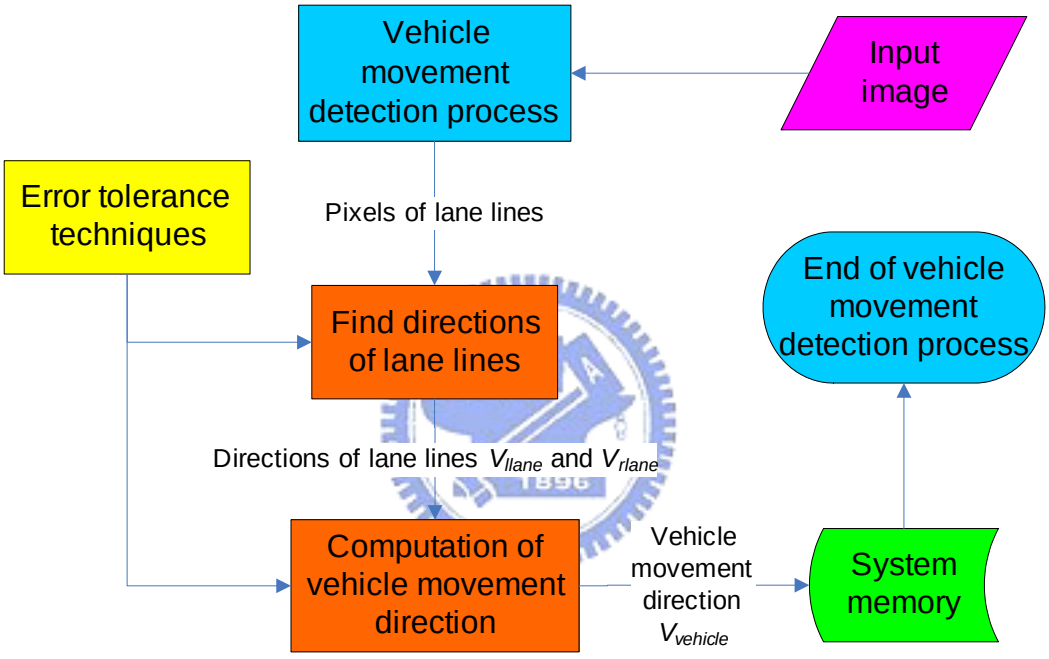


Figure 4.3 Flowchart of vehicle movement detection process.

4.3.1 Computation of Lane Line Direction by Line Fitting

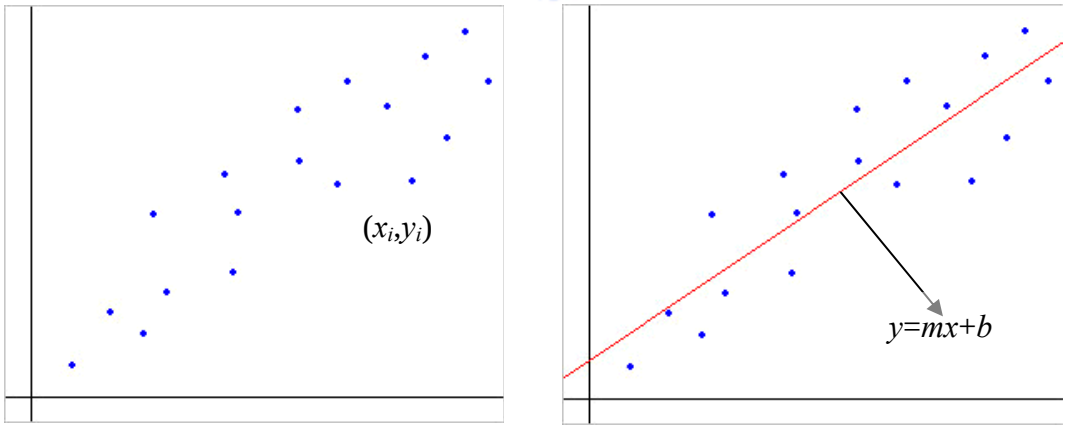
In this section, we want to compute the lane line directions for the left and right lane line pixels. For each lane line region, we want to find a line to fit the pixels found in the lane detection process. A line fitting method is used in this section. Let the

equation of the line to fit the lane line pixels be: $y = mx + b$. Then the slope of the line m and the constant b can be computed by the following equations:

$$\Rightarrow m = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^{\,2} - \left(\sum_{i=1}^{n} x_i\right)^2}$$

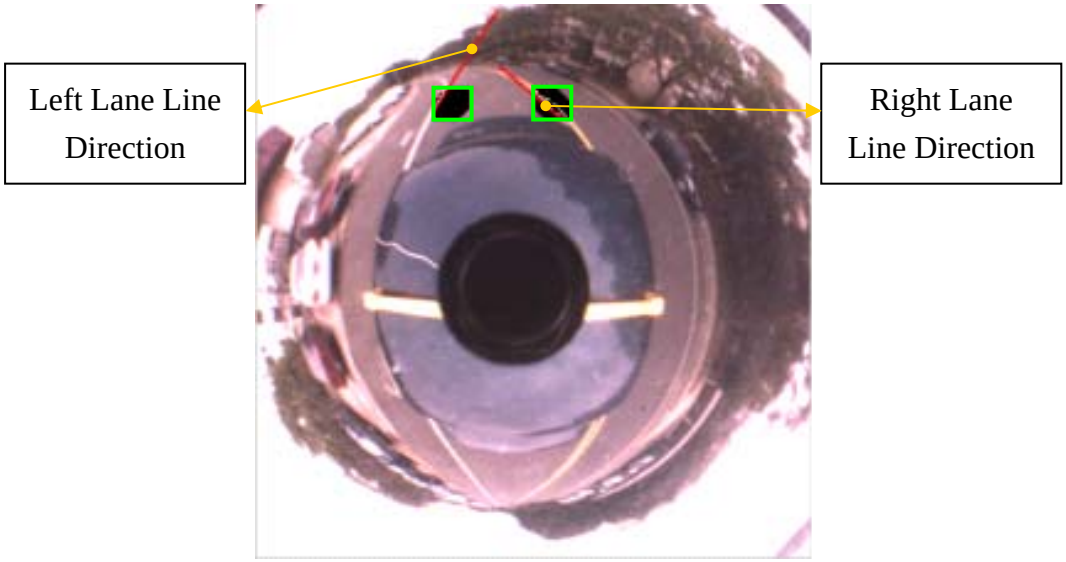
$$\Rightarrow b = \frac{\sum_{i=1}^{n} x_i^{\,2} \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i y_i \sum_{i=1}^{n} x_i}{n\sum_{i=1}^{n} x_i^{\,2} - \left(\sum_{i=1}^{n} x_i\right)^2}$$

In the above equations, n denotes the number of the lane line pixels; x_i and y_i denote the x coordinate and the y coordinate of the current pixel. Then we can find the lines to fit the left and right lane line pixels. Figure (a) and (b) show an illustration of the line fitting method. The slope of the line represents the direction of the lane line. Figure (c) shows an example of finding the left and right lane line directions. Then we can combine the two lane line directions into the vehicle movement direction, as described in the following section.



(a) (b)

Figure 4.4 Illustration of line fitting and results of finding lane line directions. (a) Input pixels for line fitting. (b) Results of line fitting. (c) An example of finding left and right lane line directions.



(c)

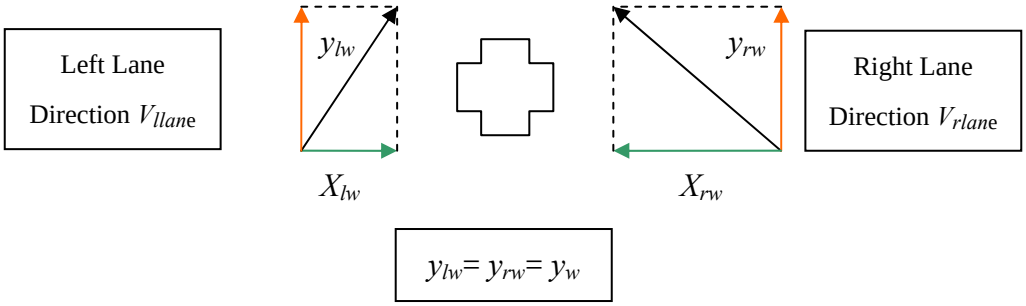
Figure 4.4 Illustration of line fitting and results of finding lane line directions. (a) Input pixels for line fitting. (b) Results of line fitting. (c) An example of finding left and right lane line directions (continued).

4.3.2 Computation of Vehicle Movement Direction

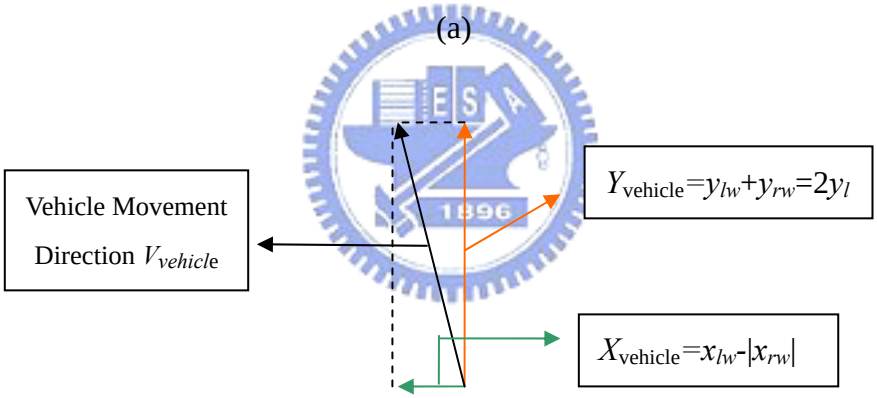
In this section, we describe how we combine the left and right lane line directions into the vehicle movement direction. The combination process is described by the following steps.

Step 1. Denote the slopes of the left and right lane line directions as V_{llane} and V_{rlane}. Also denote the x and y coordinate weights of V_{llane} and V_{rlane} as (x_{lw}, y_{lw}) and (x_{rw}, y_{rw}). Let y_{lw} and y_{rw} to be the same value as y_w. Compute the x coordinate weight of V_{llane} by the formula: $x_{lw} = \frac{y_w}{V_{llane}}$. And computer the one of V_{rlane} by the formula: $x_{rw} = \frac{y_w}{V_{rlane}}$. Figure 4.5(a) shows an example.

Step 2. Compute the combination of the left and right lane line directions and get the vehicle movement direction as shown in Figure 4.5(b). The x coordinate weight is: x_{lw}-$|x_{rw}|$ and the y coordinate weight is $2y_{lw}$. Then the vehicle movement direction is computed as: $V_{vehicle} = \frac{x_{lw} - |x_{rw}|}{2y_{lw}}$.



(a)



(b)

Figure 4.5 An example of combining the two lane line directions. (a) Step 1. (b) Step 2.

4.3.3 Error Tolerance Technique for Vehicle Movement Detection

During the process of the lane detection process and the vehicle movement

detection process, there are three main conditions that cause erroneous results. Three error tolerance techniques are developed in this section to eliminate the effects of the three error conditions. These error conditions and their corresponding solutions are discussed in the following.

Condition 1、 In the vehicle movement detection process, only one of the two lane lines is found in the input image, and the other lane line is missed because the lane line is not continuous.

Figure (a) shows an example of this condition. The difference between two consecutive images grabbed by the camera is small because the time interval between two images is small. The lane line directions will not have great changes within such a short time interval. Thus for each input image, our solution to handle this condition is to keep the left and right lane line directions computed from the previous image in advance. Let the left and right lane line directions computed from the previous image are V_{pllane} and V_{prlane}. If one of the two lane line directions is unavailable, we use the lane line direction of the previous frame to replace the lost lane line direction.

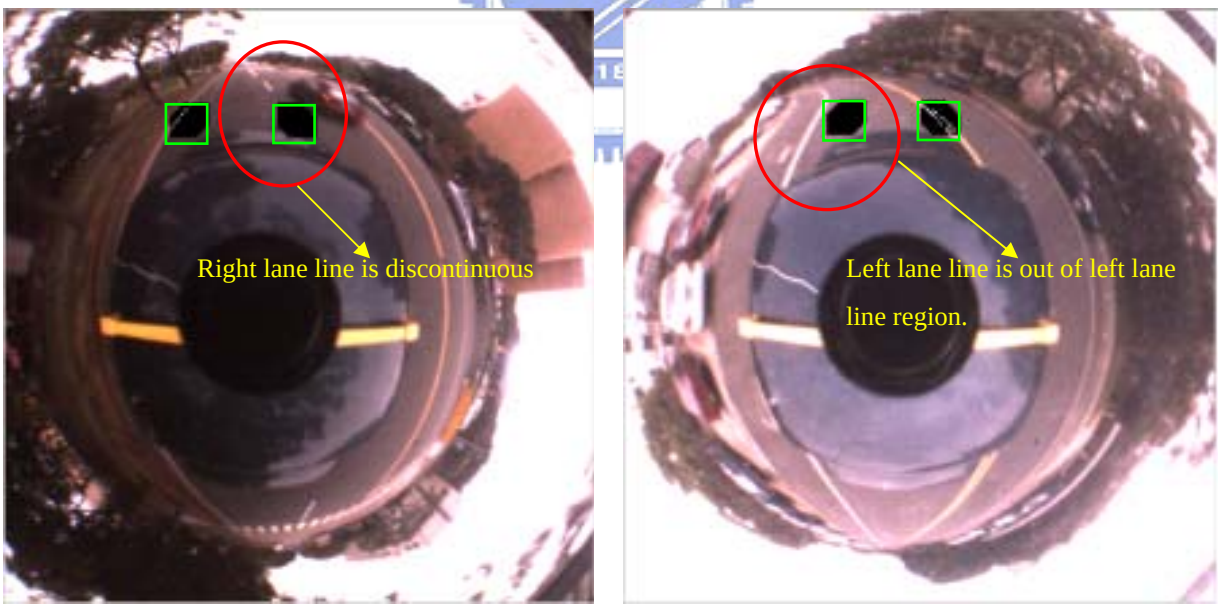
Condition 2、 In lane detection process, the lane line in the input image is out of the predefined lane line region because of the movement of the vehicle.

Figure (b) shows an example of this condition. In order to keep tracking of the lane lines, we need to use two offsets, σ_{llane} and σ_{rlane}, to modify the positions of the left and right lane line regions as the lane lines move. The two offsets can be computed from the changes of the positions of the lane lines between two consecutive images. P_{llane} and P_{rlane}, which are the mean positions of the left and right lane lines computed in the lane detection process, are used for computation of σ_{llane} and σ_{rlane}.

Besides, the mean positions of the left and right lane lines computed in the previous input image are also kept as $P_{pllane}(X_{pllane}, Y_{pllane})$ and $P_{prlane}(X_{prlane}, Y_{prlane})$. Then the two offsets, σ_{llane} and σ_{rlane}, can be computed from the following equation.

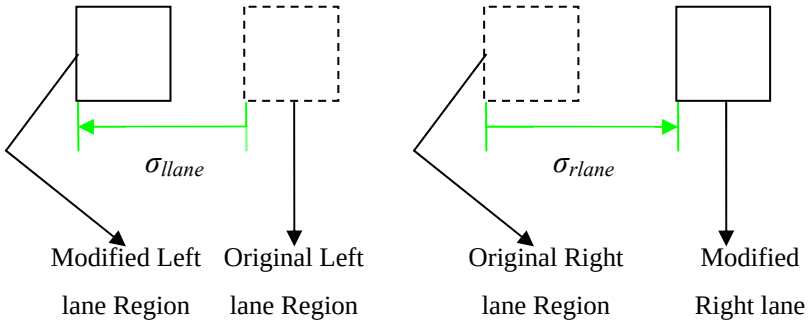
$$\sigma_{llane} = x_{llane} - x_{pllane};$$
$$\sigma_{rlane} = x_{rlane} - x_{prlane}.$$

Thus the positions of the lane line regions can be modified automatically by the changes of the positions of the lane lines. Normally, the offset σ_{llane} and σ_{rlane} will not have great changes because the changes of the positions of the lane lines in the input image are slight. If one of the offsets σ_{llane} and σ_{rlane} is lager than a threshold, this offset will be determined to be noise. The lane line region will not be modified by the offset. Figure (c) shows an example of modifying the lane line regions by the use of the lane line offsets.



(a) (b)

Figure 4.6 Examples of error tolerance techniques. (a) An example of Condition 1. (b) An example of Condition 2. (c) An example of modifying lane line regions by lane line offsets.



(c)

Figure 4.6 Examples of error tolerance techniques. (a) An example of Condition 1. (b) An example of Condition 2. (c) An example of modifying lane line regions by lane line offsets (continued).

Condition 3、 The vehicle movement detection process gets an erroneous lane line direction of the left or right lane line region because of the noise in the input image?

There are many kinds of noises that cause erroneous results to the lane detection process. For example, a pixel with color similar to the lane line color has a chance to be classified to be a lane line pixel. This pixel will cause erroneous results to the process of line fitting. Figure 4.7 shows an example of this condition. The same as the concept involved in Condition 1, the difference between two consecutive images is slight. Thus the changes of the lane line directions between the lane line directions of the current input image and the ones of the previous input images are also small. Thus if the change between V_{llane} and V_{pllane} is lager than a threshold, then V_{llane} is determined to be a noise result and will be dismissed. For V_{rlane} and V_{prlane}, the same operation is applied. Furthermore, if one of the lane line directions is unavailable, the

vehicle movement direction of the current input image will also be unavailable.

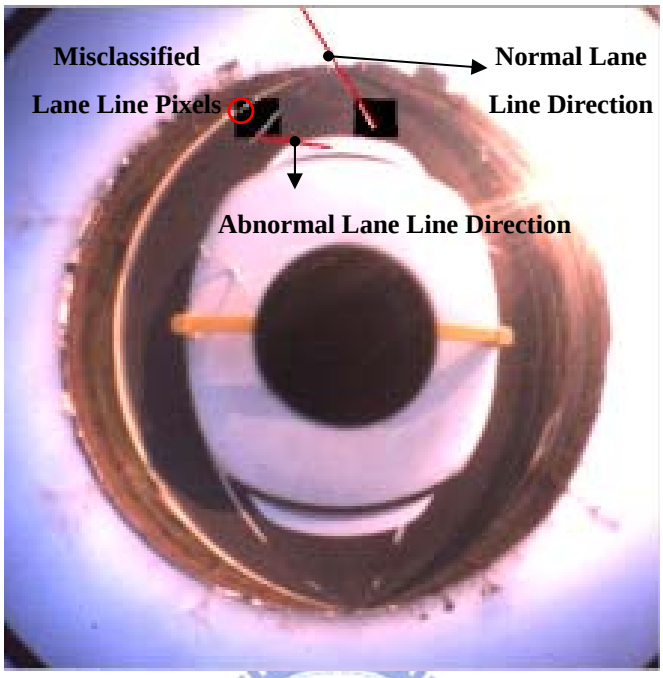


Figure 4.7 An example of an abnormal lane line direction.

Although we can use the error tolerance technique described above to detect the error lane line directions caused by the noise in the lane line regions, this technique can only ignore the erroneous results. It can not reduce the rate of getting erroneous results. If the unnecessary regions of the lane line regions can be eliminated, the error rate caused by the noise can be decreased. The original lane line regions are rectangles which are 28 pixels in width and 28 pixels in height. To avoid erroneous results caused by noise, the sizes and shapes of the lane line regions need to be modified by referring to the positions and directions of the lane lines. A lane line region modification algorithm is developed here. The main idea of this algorithm is to eliminate the regions that are far away from the lane lines in the input image by using the positions and directions of the lane lines. The detailed algorithm is described as follows.

Algorithm 4.2. *Lane line regions modification.*

Input: (1) the left and right lane line positions, P_{llane} and P_{rlane}; (2) the slopes of the left and right lane line directions, V_{llane} and V_{rlane}.

Output: the modified left and right lane line regions.

Steps:

Step 1. For the left lane line region, find the pixel P_{rllane} which is 6 pixels higher than P_{llane} horizontally and 8 pixels lower than P_{llane} vertically. And find the pixel P_{lllane} which is 6 pixels lower than P_{llane} and 8 pixels higher than P_{llane}.

Step 2. For the right lane line region, find the pixel P_{rrlane} which is 6 pixels higher than P_{llane} horizontally and 8 pixels higher than P_{llane} vertically. And find the pixel P_{lllane} which is 6 pixels lower than P_{llane} and 8 pixels lower than P_{llane}. Figure 4.8(a) shows an example.

Step 3. For the left lane line region, create a line L_{rllane} with the slope V_{llane} which passes through P_{rllane} and create a line L_{lllane} with the same slope which passes through P_{lllane}. For the right lane line region, create the lines L_{lrlane} and L_{rrlane} by the same process. Figure 4.8(b) shows an example.

Step 4. Take the desired modified left lane line region as the region between L_{lllane} and L_{rllane}. And take the desired modified right lane line region as the region between L_{lrlane} and L_{rrlane}. Figure 4.8(c) shows an example.

Figure 4.8(d) shows an example of the modification result. Thus unnecessary regions will be eliminated and the noise in these regions will also be discarded. The rate of getting erroneous results will thus be reduced. This algorithm is applied to each input image. And the lane line regions will vary with each input image.

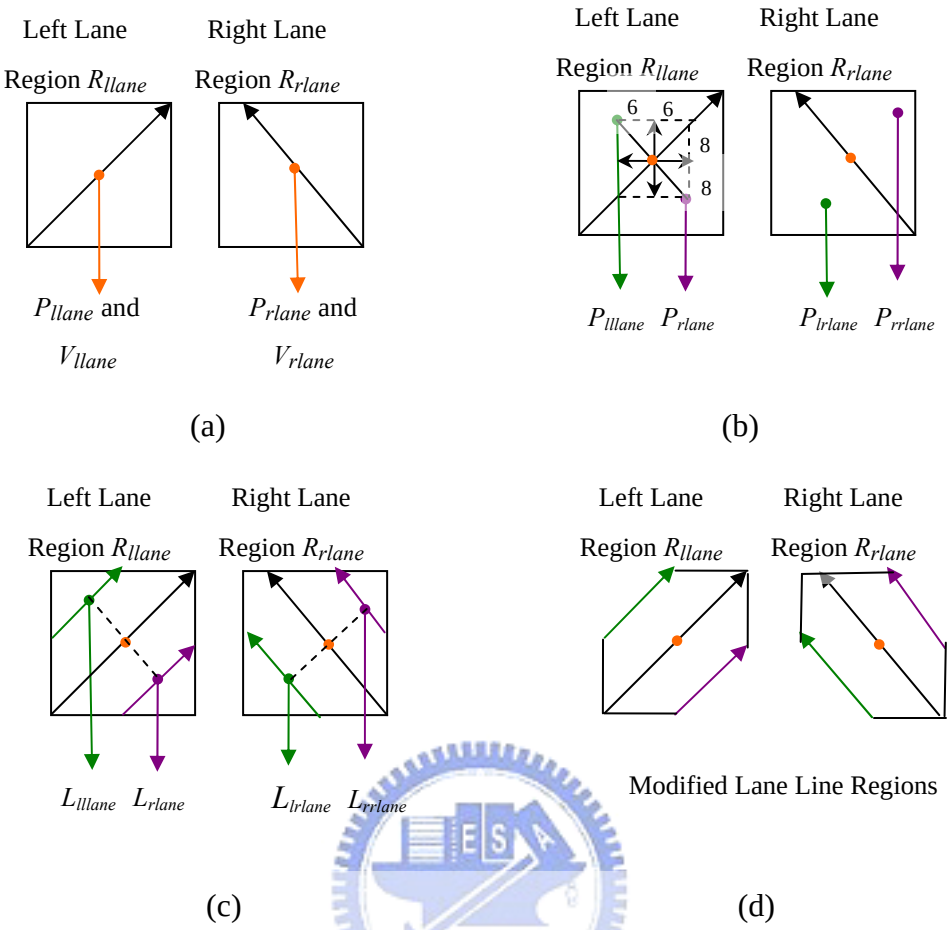


Figure 4.8 Illustration of modifying lane line regions.

4.4 Vehicle Movement Analysis

In this section, a vehicle movement analysis process is proposed. After combining the lane line directions into the vehicle movement direction, we want to analyze the vehicle movement direction and determine what the vehicle movement condition is now. First, we have to define the vehicle movement conditions. Then a method is developed to recognize these vehicle movement conditions. Figure 4.9 shows a flowchart of the vehicle movement analysis process.

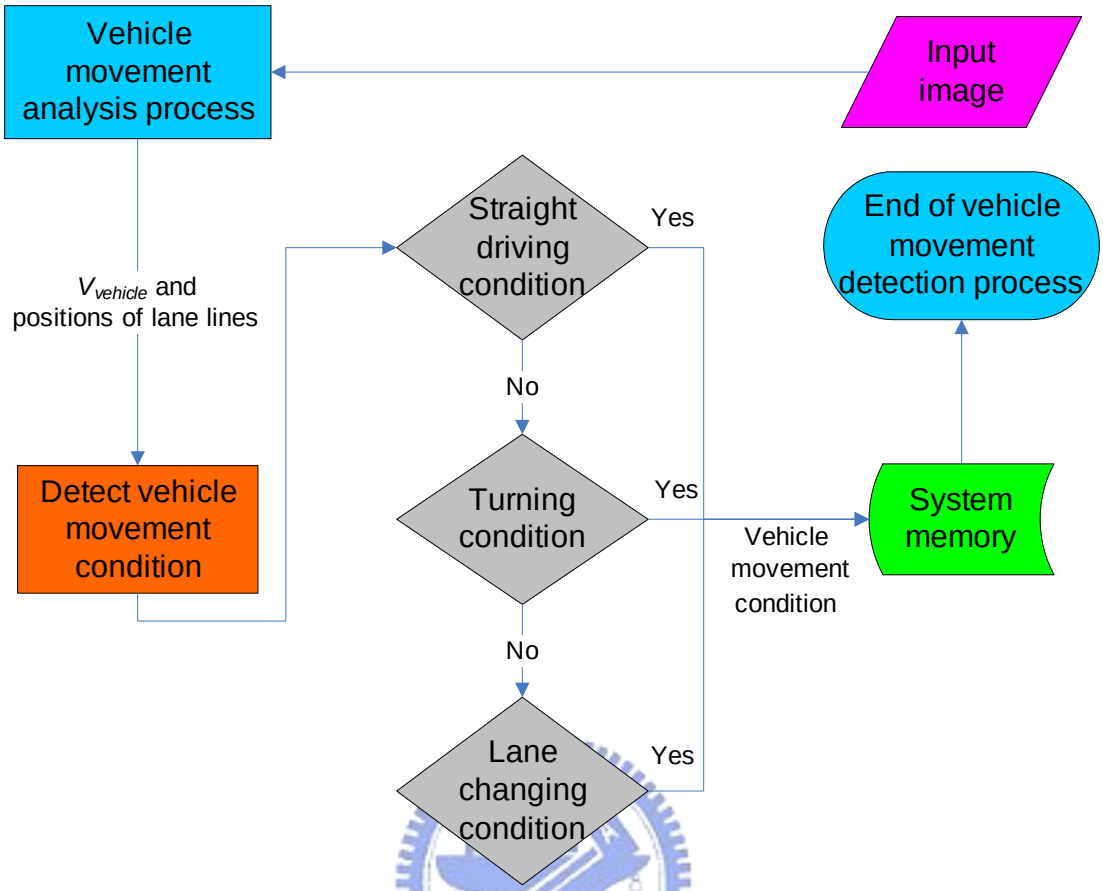
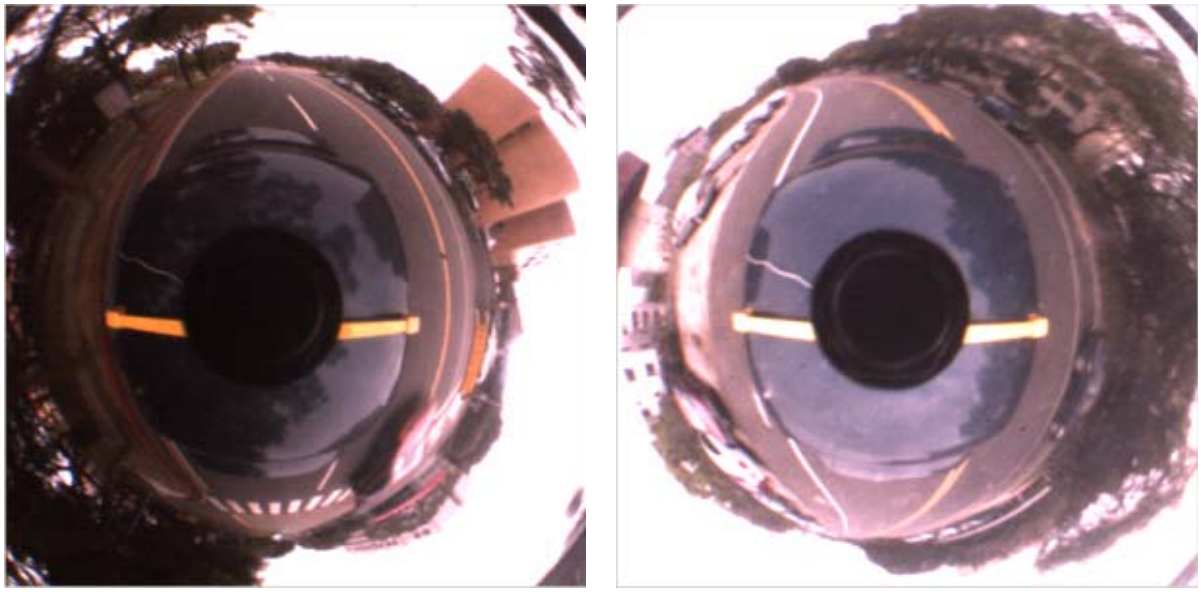


Figure 4.9 Flowchart of the vehicle movement analysis process.

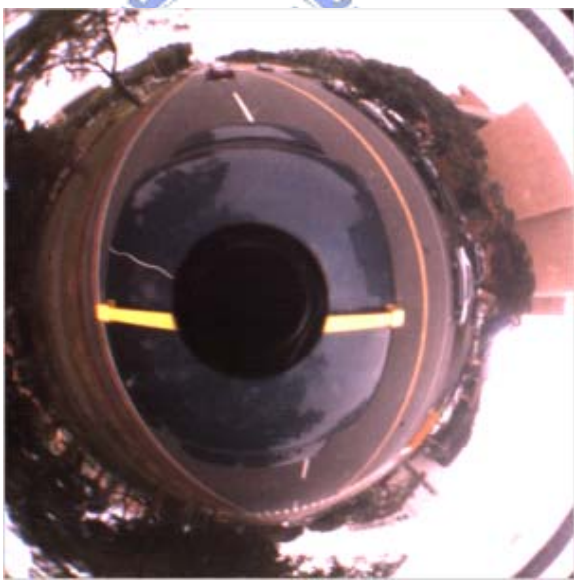
4.4.1 Vehicle Movement Conditions

During driving in outdoor environment, there are many kinds of vehicle movement conditions. The main conditions are: the vehicle moves straightly, the vehicle turns left or right, and the vehicle moves to the left or right lane. The three conditions are discussed in this section. Although these conditions are normal behaviors during driving, it is not easy to classify these conditions using image processing techniques. In the following section, a method is developed to classify these conditions. Figure 4.10 shows some examples of the vehicle movement

conditions.



(a) (b)



(c)

Figure 4.10 Examples of vehicle movement conditions. (a) An example of straight driving. (b) An example of left turning. (c) An example of right lane changing.

4.4.2 Analysis and Recognition of Vehicle Movement

Conditions

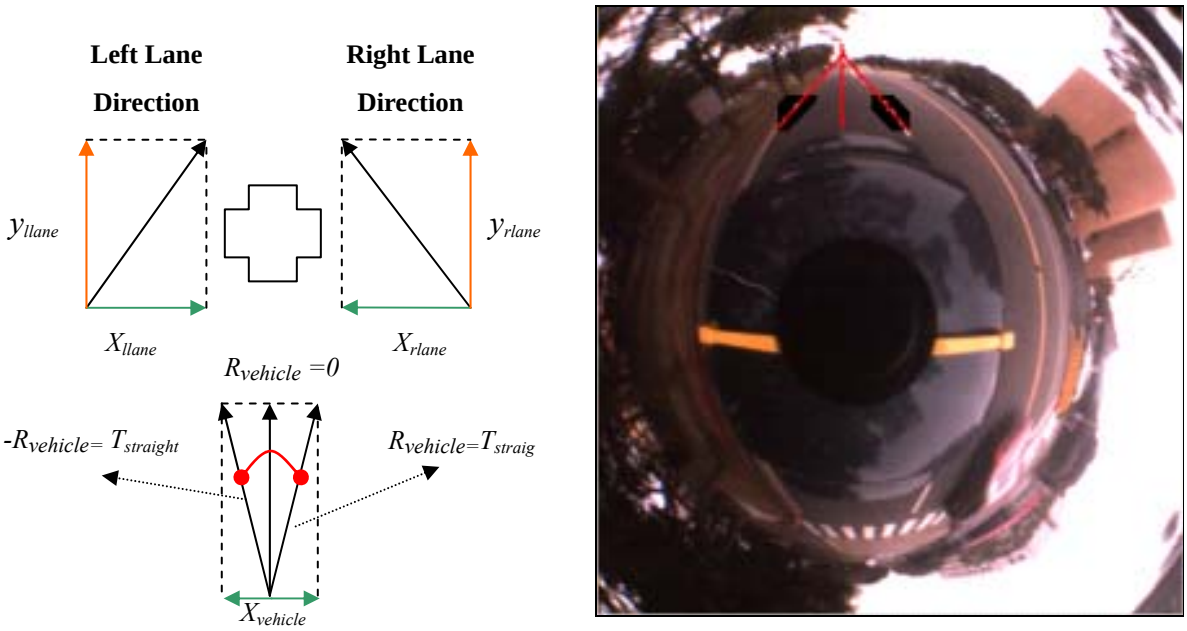
For the conditions described in the previous section, we want to classify them by using image processing techniques. The main method used is to check the lane line directions and the vehicle movement direction. But it is not enough for classifying the lane changing condition. Thus the lane line positions found in the lane detection process is included into the process of classifying the three conditions. The classification methods are described in the following section.

4.4.2.1 Straight Driving Condition

In the straight driving condition, the left and right lane line directions have x coordinate weights X_{llane} and X_{rlane} which are opposite in direction and equal in length. Thus the combination of X_{llane} and X_{rlane} will nearly cancel out; only the y coordinate weights Y_{llane} and Y_{rlane} are left. The x coordinate weight of the vehicle movement direction $X_{vehicle}$ is nearly zero. And so is the vehicle movement direction, $R_{vehicle}$, which is computed from the equation: $R_{vehicle} = \frac{X_{vehicle}}{Y_{vehicle}}$. We use a threshold $T_{straight}$ to determine whether the vehicle movement condition is straight driving or not. The classification formula is defined as follows:

$$\begin{cases} \text{if: } |R_{vehicle}| \geq T_{straight} \Rightarrow \text{non-straight driving condition;} \\ \text{if: } |R_{vehicle}| < T_{straight} \Rightarrow \text{straight driving condition.} \end{cases}$$

Figure 4.11(a) shows the range of $R_{vehicle}$ to be classified as the straight driving condition. Figure 4.11(b) shows an illustration of the straight driving condition.



(a) (b)

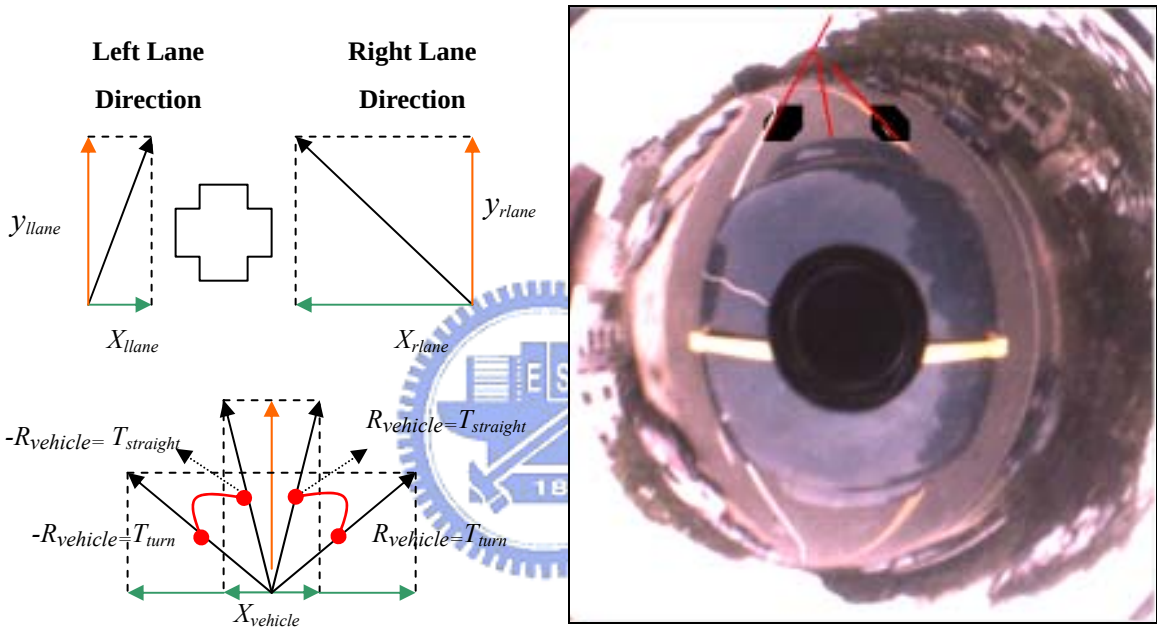
Figure 4.11 Illustration of straight driving condition. (a) The range of $R_{vehicle}$ to be classified as the straight driving condition. (b) An example of the straight driving condition.

4.4.2.2 Turning Condition

In the left or right turning condition, the combination of X_{llane} and X_{rlane} will not cancel out. For the condition of turning left, the length of X_{rlane} will be larger than X_{llane}. And for the condition of turning right, the length of X_{llane} will be larger than X_{rlane}. We use another threshold, T_{turn}, for the determination of the turning condition. We modify the classification formula used for the straight driving condition to be:

$$\begin{cases} \text{if: } |R_{vehicle}| < T_{straight} & \Rightarrow \text{staright driving condition;} \\ \text{else if: } T_{turn} \geq R_{vehicle} \geq T_{straight} & \Rightarrow \text{right turning condition;} \\ \text{else if: } T_{turn} \geq -(R_{vehicle}) \geq T_{straight} & \Rightarrow \text{left turning condition.} \end{cases}$$

Classification of the left and right turning conditions depends on the direction of $R_{vehicle}$. If $R_{vehicle}$ is positive and its value is between $T_{straight}$ and T_{turn}, the direction of $R_{vehicle}$ is from left to right. Then this is the right turning condition. If $R_{vehicle}$ is negative and its absolute value is between $T_{straight}$ and T_{turn}, then this is the right turning condition. Figure 4.12(a) shows the range of $R_{vehicle}$ to be classified as the turning condition. Figure 4.12(b) shows an example of the left turning condition.



(a) (b)

Figure 4.12 Illustration of the turning condition. (a) The range of $R_{vehicle}$ to be classified as the turning condition. (b) An example of the left turning condition.

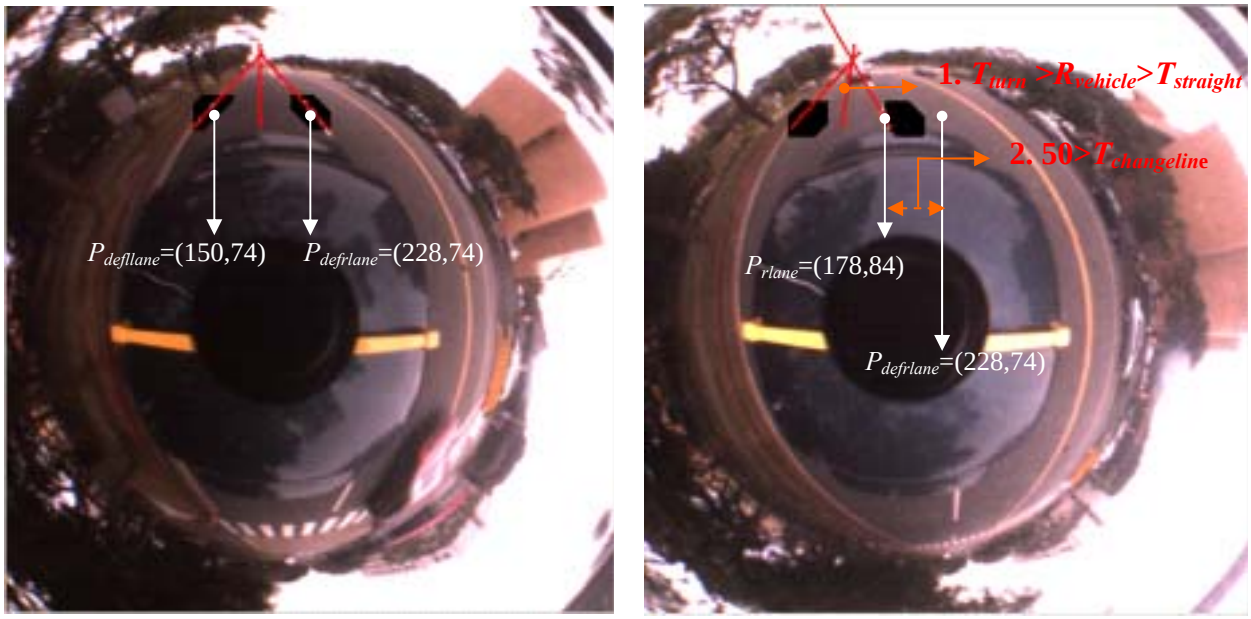
4.4.2.3 Lane Changing Condition

The process of the classification of the lane changing condition is just like the process of the turning condition. But it is difficult to classify such conditions by the

vehicle movement direction only. The main difference between the turning condition and the lane changing condition is that the lane line positions will change rapidly within a short time in the lane changing condition. However, this phenomenon does not exist in the turning condition. Thus we can classify these two conditions by tracking of the lane line positions. To measure the movement of the lane line positions in the current image, the default positions of the lane lines need to be found first. The lane line positions of the first input image are defined to be the default lane line positions, $P_{defllane}$ and $P_{defrlane}$. If the distance between the current lane line position and the default lane line position is larger than a predefined threshold, $T_{changeline}$, this condition is classified as the lane changing condition. We expand the classification formulas as follows.

$$\begin{cases} \text{if: } |R_{vehicle}| < T_{straight} & \Rightarrow \text{straight driving condition;} \\ \text{else if: } (T_{turn} \geq R_{vehicle} \geq T_{straight}) \,\&\, (|P_{rlane} - P_{defrlane}| > T_{changline}) & \Rightarrow \text{right lane changing condition;} \\ \text{else if: } (T_{turn} \geq R_{vehicle} \geq T_{straight}) \,\&\, (|P_{rlane} - P_{defrlane}| < T_{changline}) & \Rightarrow \text{right turning condition;} \\ \text{else if: } (T_{turn} \geq -(R_{vehicle}) \geq T_{straight}) \,\&\, (|P_{llane} - P_{defllane}| > T_{changline}) & \Rightarrow \text{left lane changing condition;} \\ \text{else if: } (T_{turn} \geq -(R_{vehicle}) \geq T_{straight}) \,\&\, (|P_{llane} - P_{defllane}| < T_{changline}) & \Rightarrow \text{left turning condition.} \end{cases}$$

Thus three main vehicle movement conditions can be classified by the previously-mentioned formulas. Figure 4.12 shows an example of the lane changing condition. Figure 4.12(a) shows the default lane line positions and Figure 4.12(b) shows an example of the right lane changing condition. In this example, we can find that the distance between P_{rlane} and $P_{defrlane}$ is 50 pixels and the vehicle movement direction is 0.2537. This condition is classified as the right lane changing condition.



(a) (b)

Figure 4.13 An example of lane changing condition. (a) The default lane line positions $P_{defllane}$ and $P_{defrlane}$. (b) Classification of the lane changing condition.

Chapter 5
A Method for Detection and Tracking of Neighboring Objects

5.1 Introduction

In order to keep the safety of vehicles, vehicle neighboring objects should be tracked and monitored continuously. When risk conditions are detected, we have to inform drivers immediately. In this chapter, a method for detection and tracking of neighboring objects is proposed. In Section 5.2, a neighboring object detection process is proposed. Secondly, we have to determine what the obtained objects are by classifying these objects into three groups, namely, neighboring vehicles, obstacles on the ground and non-objects. An object classification process is proposed in Section 5.3. Finally, we have to keep tracking and monitoring the neighboring vehicles after classification of the objects. This method is used for detection of risk conditions such as the case that a neighboring vehicle is approaching and the distance is short or the case that an obstacle is approaching. A neighboring vehicle tracking algorithm is described in Section 5.4.

5.2 Detection of Neighboring Objects

In this section, we want to find neighboring objects in each input image by image processing techniques. The main idea is to use image differencing techniques to find the difference image between two consecutive images. Then we can analyze the difference image and find the objects from it. We can then compute the parameters of each object. Each parameter of an object represents a property of the object. Thus we can classify these objects by their properties. Figure 5.1 shows a flowchart of the neighboring object detection process.

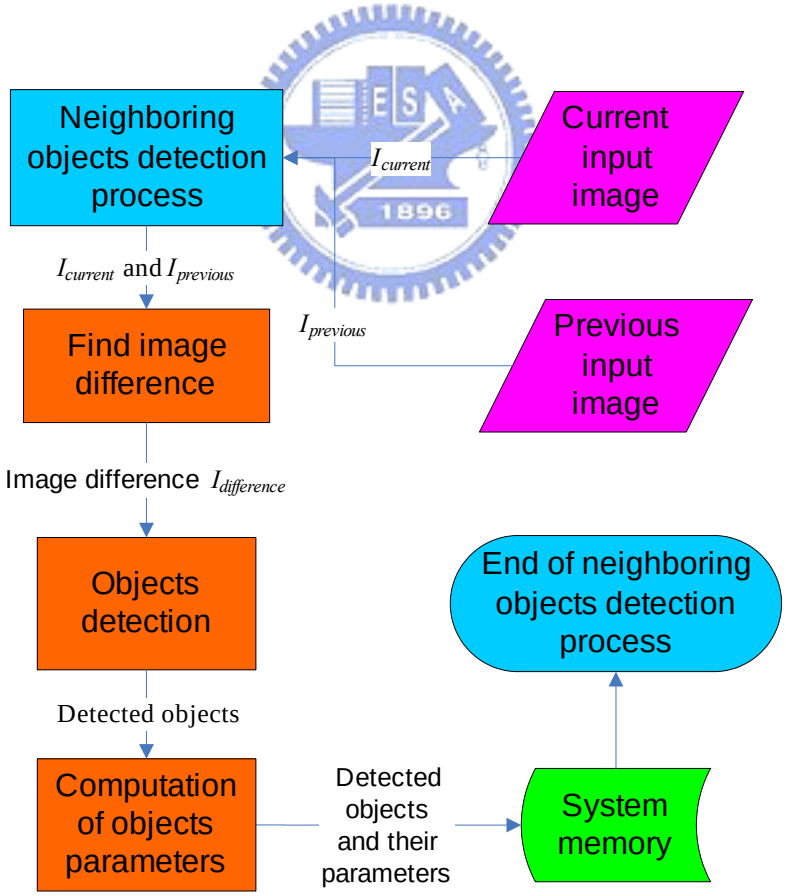
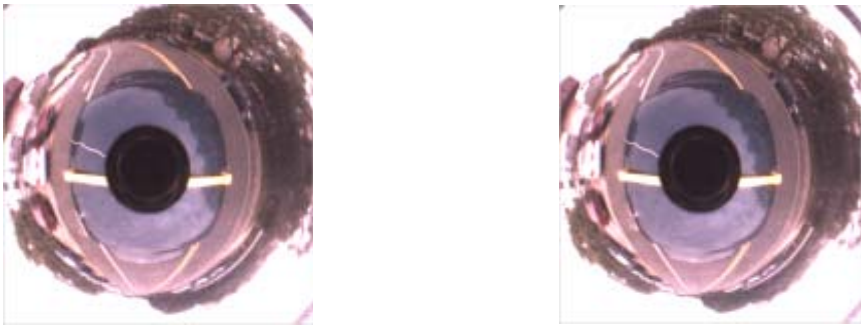


Figure 5.1 Flowchart of neighboring object detection process.

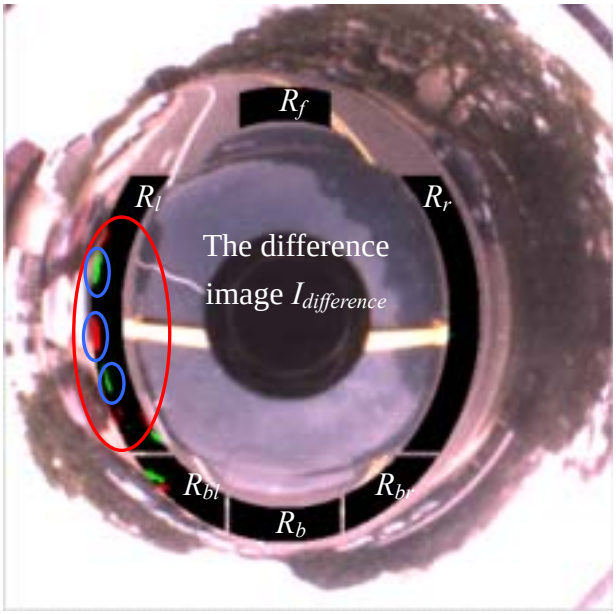
5.2.1 Estimation of Changes between Two Consecutive Images

In this section, we want to find the difference image between two consecutive images. We assume that the two neighboring images are the current input image, $I_{current}$, and the previous input image, $I_{previous}$. We simply subtract $I_{previous}$ from $I_{current}$ and the difference image, $I_{difference}$, can be acquired. However, it costs much time to apply the subtraction operation to the entire image. We want to apply the subtraction operation only on regions of interest. The warning region R_{warn} defined in the learning process is the region around the vehicle that we want to monitor. The warning region is divided into six regions: front region, R_f, back region, R_b, left region, R_l, right region, R_r, blind left region, R_{bl}, and blind right region, R_{br}. These regions are used for detection and tracking of neighboring objects. The image difference process is applied only to these regions. After obtaining the image difference, the changes of these regions between $I_{previous}$ and $I_{current}$ within a short time interval can be found. Figure 5.2 shows an example of finding the image difference of the six warning regions.



(a) (b)

Figure 5.2 An example of finding the image difference of the six warning regions. (a) Current input image $I_{current}$. (b) Previous input image $I_{previous}$. (c) Results of finding image difference of the six regions.



(c)

Figure 5.2 An example of finding the image difference of the six warning regions. (a) Current input image $I_{current}$. (b) Previous input image $I_{previous}$. (c) Results of finding image difference of the six regions (continued).

5.2.2 Detection of Neighboring Objects

After finding the difference image, in this section we describe how to analyze the obtained difference image and find objects from it. The idea is to collect neighboring pixels in $I_{difference}$. Figure 5.2(c) shows an example of objects; the neighboring pixels of green or red color within each blue ellipse are classified as an object. A region growing algorithm is used to collect neighboring pixels in the difference image. A region growing algorithm is applied to the six warning regions in the difference image. Then we can find all objects within the regions.

So far we have found all the objects in $I_{difference}$, but some of the objects actually belong to the same object in input image. In other words, they are parts of the object.

We have to merge them into an object. They were classified as individual ones due to the discontinuity of pixels or noise in $I_{difference}$. Thus we have to detect and merge the neighboring objects. To merge the neighboring objects, an object merging algorithm is developed in this study. The main idea of this algorithm is to merge the objects by comparing the distance between them. Two objects with a small distance between them are merged into an object. The detailed algorithm is described as follows.

Algorithm 5.1. *Object merging.*

Input: (1) all the objects found in $I_{difference}$; (2) a threshold T_{dmin}.

Output: objects after applying object merging.

Steps:

Step 1. For each object, compute the mean positions of the pixels within it. Assume that the mean position of the *i*th object is M_i.

Step 2. For each object, find all the distances between its mean position and all the other objects' mean position.

Step 3. For each object, if one of the distances is smaller than the threshold T_{dmin}, merge the two objects and re-compute the mean of the merged object. And then return to Step 2.

Step 4. If no objects were merged in Step 3, the process has come to its end and the objects after applying object merging are obtained.

In the above algorithm, we simply use the distance between the mean positions of objects to determine whether two objects should be merged. If two objects are merged, the pixels of one object will be taken as part of the other. And the mean position of the new object will be re-computed by the equation: $M_i' = \frac{M_a + M_b}{2}$, where M_a and

M_b are the mean positions of the two objects for merging. Figure 5.3 shows an example of applying this algorithm.

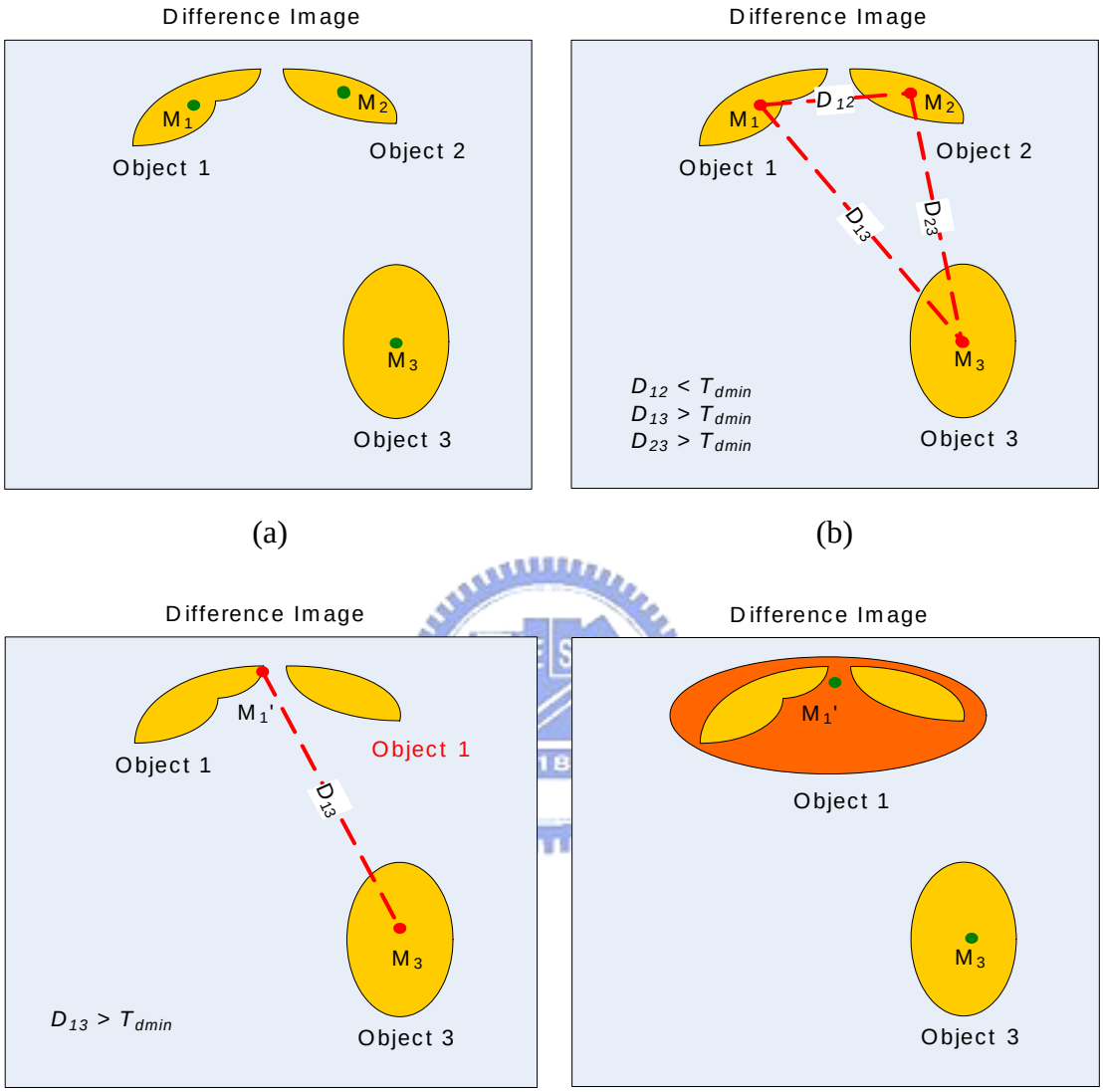


Figure 5.3 An example of applying object merging. (a) Objects for merging. (b) Finding all the distances between them. (c) Merge of Object 1 and Object 2. (d) The result of object merging.

5.2.3 Computation of Neighboring Object

Parameters

After merging the objects, we can get the output objects. Before classifying these objects, we have to find the parameters of each object. The parameters of an object represent the properties of it. These parameters include object region R_{obj}, object center M_{obj}, and object density ρ_{obj}. The methods for computing these three parameters are described as follows.

(1). Object region R_{obj}

This parameter is a rectangle region that completely encloses the object. To find the parameter, we have to find the four positions of the rectangle region. Firstly, we compare the positions of the pixels within this object and find the maxima and the minima positions in the *x*- and *y*-coordinates. The maxima and minima positions in the *x*-coordinates are denoted by X_{min} and X_{max} and those in the *y*-coordinates by Y_{min} and Y_{max}. The object region can be defined from the four positions, X_{min}, X_{max}, Y_{min}, and Y_{max}. Figure 5.4(a) shows an illustration of defining the region of an object. The width and height of the object region are W_{obj} and H_{obj} which can be computed from the equation: $W_{obj} = X_{\max} - X_{\min}; H_{obj} = Y_{\max} - Y_{\min}$.

(2). Object center M_{obj}(*Mx*, *My*)

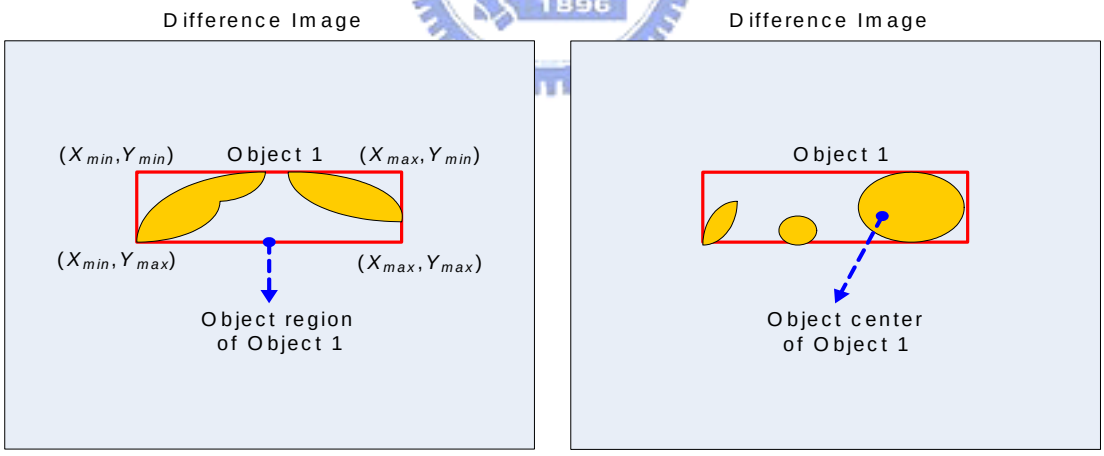
This parameter is a position which is found by computing the mean position of the pixels within the object. The object center of each object has been computed and used in the object merging algorithm described above. It represents the mass center of this object. In other words, if most of the pixels within the object fall on the right side of

the object region, the object center of this object also falls on the right side of the object region. Figure 5.4(b) shows an example.

(3). Object density ρ_{obj}

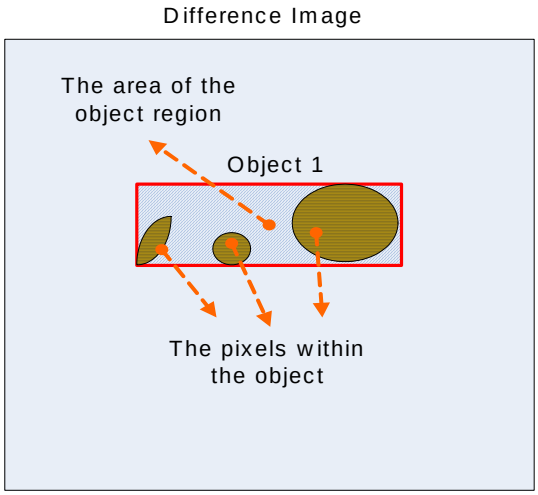
This parameter is the ratio of the area of the object region, A_{obj}, to the number of the pixels within the object, N_{obj}. The area can be computed from the width and height of the object region. Then the object density of this object can be computed from the equation: $\rho_{obj} = \frac{N_{obj}}{A_{obj}}$. Figure 5.4(c) shows an example.

The three parameters of an object represent the properties of the object. They are used in the object classification process as described in Section 5.3. We classify these objects by analyzing the parameters of them.



(a) (b)

Figure 5.4 Illustration of the object's parameters. (a) Object region of an object. (b) Object center of an object. (c) Object density of an object.



(c)

Figure 5.4 Illustration of the object's parameters. (a) Object region of an object. (b) Object center of an object. (c) Object density of an object (continued).

5.3 Classification of Neighboring Objects

So far what the obtained objects are is unknown. In this section, we want to classify these objects into three groups, namely, neighboring vehicles, obstacles on the ground, and non-objects. Objects of each group have their own properties; we can classify the objects by analyzing the three parameters of each object found in the previous section. The detailed classification methods are described as follows. Figure 5.5 shows a flowchart of the objects classification process.

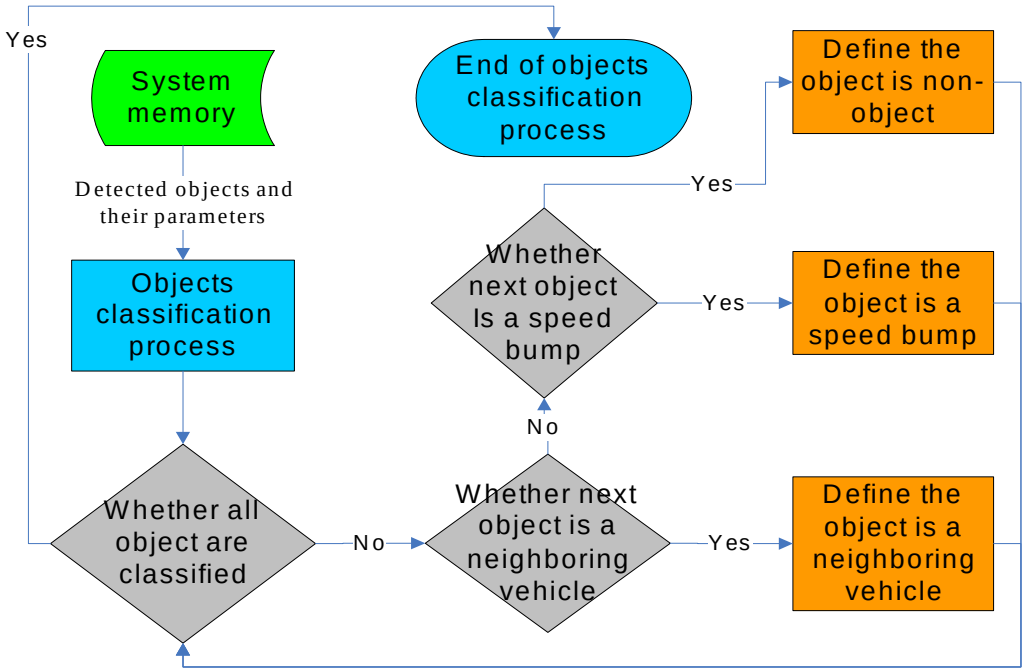
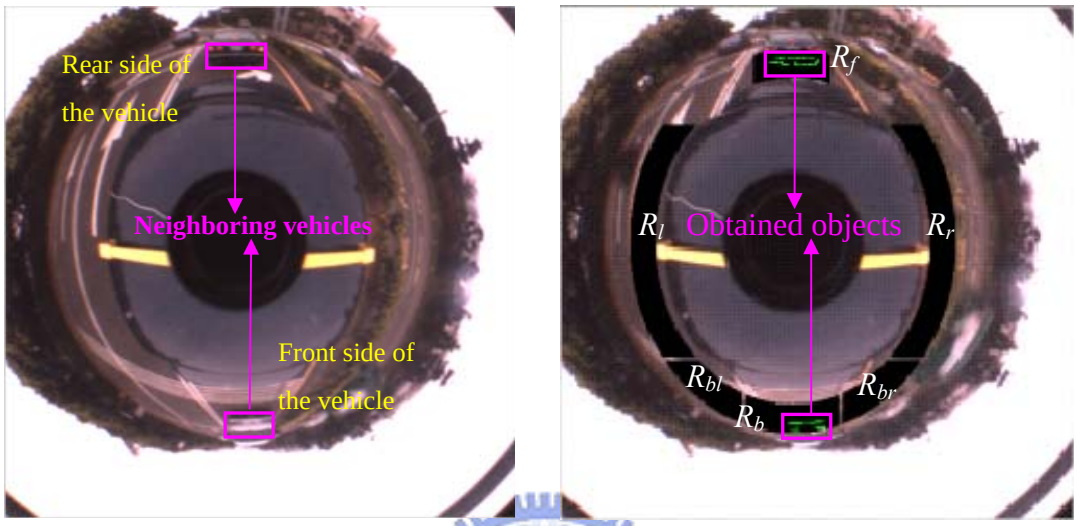


Figure 5.5 Flowchart of objects classification process.

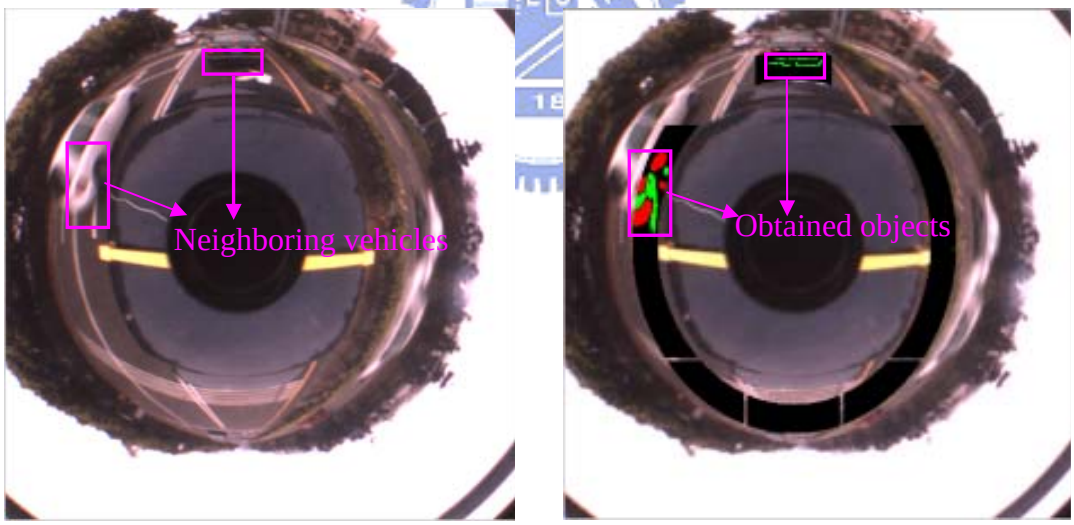
5.3.1 Neighboring Vehicles

The neighboring vehicle group includes the obtained objects which are vehicles surrounding the vehicle. Neighboring vehicles are the major objects that need to be tracked for the safety of the vehicle. Neighboring vehicles in the six warning regions need to be classified. However, the properties of the neighboring vehicles of the front and back regions are similar. And the properties of the neighboring vehicles of the left, right, blind left, and blind right regions are also similar. Thus the classification method only has to deal with two cases: neighboring vehicles of the front and back regions, and neighboring vehicles of the left, right, blind left, and blind right regions. To determine whether the obtained object is a neighboring vehicle or not, we have to analyze its parameters. Figure 5.6 shows some examples of neighboring vehicles in input images. Figure 5.6(a) shows an example of detected neighboring vehicles of the

front and back regions. Figure 5.6(b) shows examples of neighboring vehicles of the left region. The classification method of the two cases is described as follows.

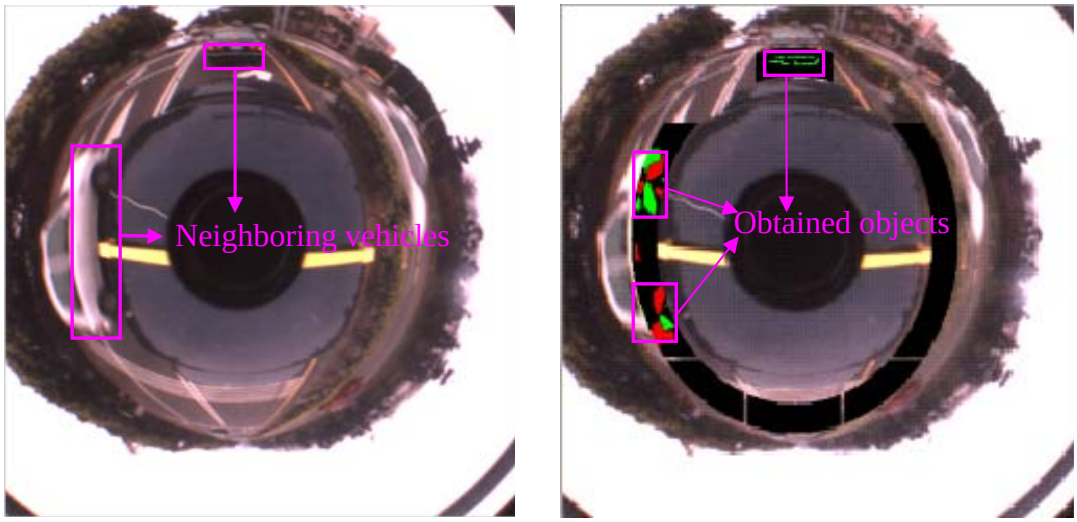


(a)



(b)-a

Figure 5.6 Examples of neighboring vehicles. (a) Neighboring vehicles of the front and back regions. (b)Neighboring vehicles of the front and left regions.



(b)-b

Figure 5.6 Examples of neighboring vehicles. (a) Neighboring vehicles of the front and back regions. (b) Neighboring vehicles of the front and left regions (continued).

Case 1 Neighboring vehicles of front and back regions.

Normally, only the rear side of a neighboring vehicle will be detected in the front region R_f and only the front side of a neighboring vehicle will be detected in the back region R_b as shown in Figure 5.6(a). Thus we can use the properties of the detected part of neighboring vehicles as a template to classify the obtained objects. For an object to be classified as a neighboring vehicle, three conditions have to be satisfied. The first is that the object center M_{obj} of the object is located near the center of the object region R_{obj}. Denote the center of the object region R_{obj} as $C_{obj}(Cx, Cy)$. Then the distance between M_{obj} and C_{obj} should be smaller than a threshold T. This is because the front or the rear side of a vehicle has symmetric property as shown in Figure 5.7. The second condition is that the width of the object region W_{obj} is between two thresholds, W_{min} and W_{max} which are the minima and maxima widths of the neighboring vehicles of the front and back regions. And the height of the object

region H_{obj} is lager than a threshold H_{fmin}. The third condition is that the object density ρ_{obj} of the object is larger than a threshold $\rho_{fvehicle}$. The classification method can be formalized as follows.

$$\left.\begin{array}{l} if \ \left|Mx-Cx\right|+\left|My-Cy\right| \le T \\ if \ W_{f\min} \le W_{obj} \le W_{f\max} \ \& \ H_{f\min} \le H_{obj} \\ if \ \rho_{obj} \ge \rho_{fvehicle} \end{array}\right\}$$

$\Rightarrow$ a neighboring vehicle of front/back region is detected.

Figure 5.7 Symmetric property of a vehicle.

Case 2 Neighboring vehicles of left, right, blind left, and blind right regions.

There are two conditions which need to be discussed in this case. One condition is that only the front or rear side of the neighboring vehicle is detected in the input image, as shown in Figure 5.6(b)-a. The other condition is that the entire neighboring vehicle is detected in the input image, as shown in Figure 5.6(b)-b. This condition only appears in the left and right regions.

For the first condition, we classify an object by the width and height of the R_{obj} of the object and its object density. The object center of the object is not used because the symmetric property dose not exists in the side view of a vehicle. For an object to be classified as a neighboring vehicle, two conditions have to be satisfied. One is that the width of the object W_{obj} is larger than a threshold W_{lmin} and the height of the object H_{obj} is between two thresholds, H_{lmin} and H_{lmax}. The other one is that the object density is larger than a threshold $\rho_{lvehicle}$. The classification formula is described as follows:

$$\left.\begin{array}{l} if\ \mathrm{H}_{\mathrm{lmin}} \le H_{obj} \le H_{l\max}\ \&\ W_{l\min} \le W_{obj} \\ if\ \rho_{obj} \ge \rho_{lvehicle} \end{array}\right\}$$

$\Rightarrow$ A neighboring vehicle of left or right regions is detected.

For the second condition, as shown in Figure 5.6(b)-b, although there is only one neighboring vehicle in the left region in the input image, two objects are detected. One is the front side of the vehicle and the other is the rear side of the vehicle. However, the two objects belong to the same vehicle actually. Thus the two objects should be determined to one neighboring vehicle of the left region. To achieve this, firstly we have to determine whether the two objects are neighboring objects of the left region by the classification formula described in the previous case. If the two objects are determined to be neighboring vehicles, we can use the distance between object centers of the two objects for analysis. If the distance is between two thresholds, D_{min} and D_{max}, the two objects will be determined to one neighboring vehicle. Figure 5.8 shows an example.

For the blind left and blind right regions, the classification method is just like the one of the first condition of the left and right regions described above.

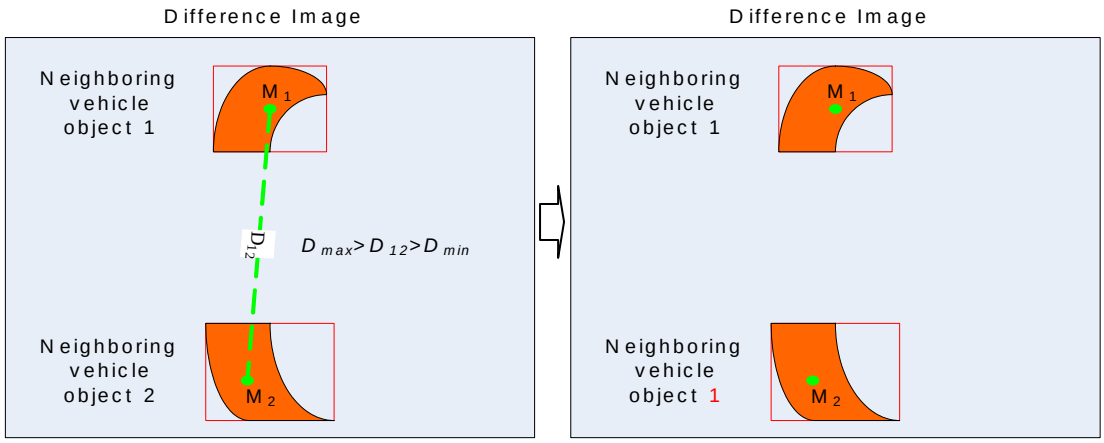


Figure 5.8 An example of determining two neighboring vehicle objects as the same vehicle.

5.3.2 Speed Bumps on the Ground

In order to inform drivers to decrease his speed in some dangerous sections of highways, speed bumps are installed on the ground. Thus we want to detect speed bumps on the ground in advance and inform drivers to decrease his speed. Only the speed bumps in the front region of the vehicle need to be detected. We determine whether the obtained objects are speed bumps or not by the properties of speed bumps in the input image. If an object is a speed bump, three conditions are satisfied. The first one is that the distance between M_{obj} and C_{obj} is smaller than a threshold T. The second condition is that the value of W_{obj} is larger than a threshold W_{spdbmp} and the value of H_{obj} is smaller than a threshold H_{spdbmp}. The third condition is that the value of ρ_{obj} is larger than a threshold $\rho_{spdbmp.}$ The classification formula is as follows.

$$\left.\begin{array}{l} if\ \ |Mx - Cx| + |My - Cy| \le T \\ if\ \ W_{spdbmp} \le W_{obj}\ \&\ H_{obj} \le H_{spdbmp} \\ if\ \ \rho_{obj} \ge \rho_{spdbmp} \end{array}\right\}$$

$\Rightarrow$ A speed bump in front regions is detected.

5.3.3 Non-Objects

There are also various kinds of objects that are detected in our system. These objects include two types. One type is meaningless objects such as trash on the ground, which are not objects of our interests. The other is noise object which is produced from the result of image difference. These two types of objects are non-objects. If an object is not a neighboring vehicle and not a speed bump on the ground, either, it will be determined to be a non-object. Then the object will be eliminated from the obtained objects.

5.4 Tracking of Neighboring Vehicles for Risk Detection

So far we have classified the obtained objects into the three groups as described in the previous sections. However, unlike speed bumps and non-objects, objects of neighboring vehicles may cause dangerous conditions. If a neighboring vehicle is moving too close, risk conditions will arise. We have to track the neighboring vehicles for risk condition detection. In this section, a neighboring vehicle tracking algorithm is developed.

5.4.1 Neighboring Vehicle Tracking

The main idea of the neighboring vehicle tracking algorithm is to keep a tracking list of neighboring vehicle objects. When the first input image is grabbed, we analyze the image and detect objects of neighboring vehicles from it by the classification method mentioned above. We put the detected neighboring vehicle objects into the tracking list. Then for each neighboring vehicle object in the tracking list, we try to find its corresponding neighboring vehicle object in the subsequntly grabbed images. If the corresponding neighboring vehicle object is found, the object in the tracking list will be replaced by its corresponding object. Otherwise, the object in the tracking list should be removed. A detailed algorithm is described as follows.

Algorithm 5.1. *Tracking of detected neighboring vehicle objects.*

Input: (1) detected neighboring vehicle objects in current input image, $O_{vehicle}$; (2) neighboring vehicle objects in the tracking list, O_{track}; (3) a threshold T_{dist}.

Output: updated neighboring vehicle objects in the tracking list.

Steps:

Step 1. For each object in the tracking list O_{track}, compute all the distances between the object center of the object and the object centers of all the objects in $O_{vehicle}$.

Step 2. For each object in the tracking list, compare all the distances found in Step 1 and mark the object by Steps 2.1 and 2.2.

2.1 If one of the distances is smaller than the threshold T_{dist}, mark the object in the tracking list with the serial number of the corresponding object in $O_{vehicle}$.

2.2 If none of the distances is smaller than the threshold T_{dist}, mark the object in the tracking list with a lost label.

Step 3. For each object in the tracking list, check the label marked on it by Steps 3.1 and 3.2.

3.1 If the label is the serial number of the object in $O_{vehicle}$, take the corresponding object to replace the object in O_{track}.

3.2 If the label is "lost", eliminate the object in O_{track}.

Step 4. The neighboring vehicle objects in the tracking list are then updated in the current input image.

In the above algorithm, we use the distances between the object center of the object and the object centers of all the objects in $O_{vehicle}$ for tracking the neighboring vehicle objects. Because the image difference between two neighboring images is small, the centers of the neighboring vehicle objects will not have great changes. If the distance is smaller than the threshold, the object corresponding to the one in the tracking list, O_{cur}, is found. If none of the distances are smaller than the threshold, it means that the object may have left the warning region. We will remove it from the

tracking list.

5.4.2 Erroneous Conditions

Although we can track objects by the algorithm described in the previous section, there are still one condition that will cause erroneous results. For example, assume that a neighboring vehicle keeps the same speed as our vehicle after it entered the warning region. At first, we can detect it using image differencing when it entered. Unfortunately, we will detect nothing and remove the neighboring vehicle object from the tracking list if it keeps the same speed as our vehicle. To solve this kind of problem, we have to detect the movement of the object. We can detect whether the neighboring vehicle object is approaching or leaving by the movement of the object centers. Take objects in the front region as an example, if the y-coordinate of the center of the object in the tracking list is smaller than the y-coordinate of the center of the corresponding one, the object is approaching our vehicle. Otherwise, the object is leaving. Assume that the object center of an object in the tracking list is $M_{track}(X_{track}, Y_{track})$ and the object center of the corresponding one is $M_{cur}(X_{cur}, Y_{cur})$. Figure 5.9 shows an illustration of detection of neighboring vehicles' movement.

After finding the movement of each neighboring vehicle object, if an object is detected to be approaching, it will not be removed from the tracking list although no corresponding object was found. Only the objects that are leaving will be removed from the tracking list when no corresponding objects were found. Thus this erroneous condition can be reduced.

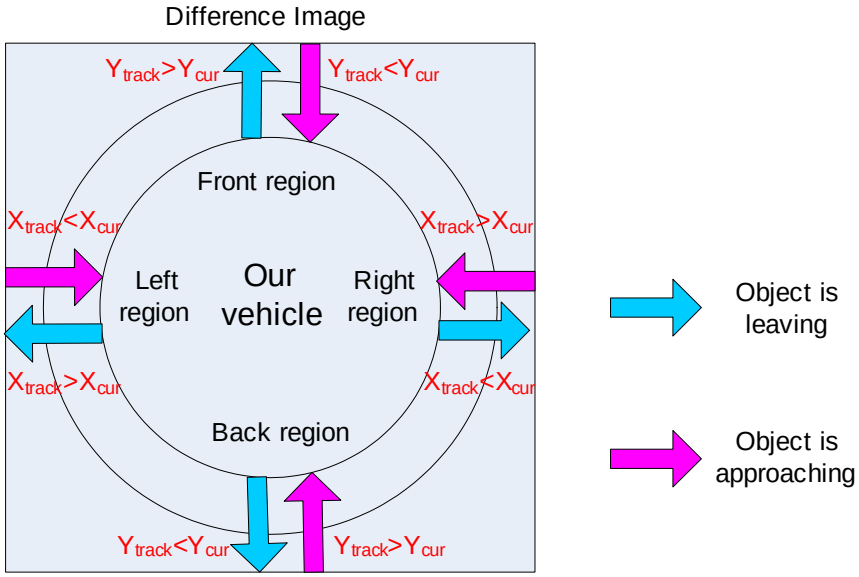


Figure 5.9 Illustration of detection of neighboring vehicles' movement.

Chapter 6
Risk Condition Detection for Driving Assistance

6.1 Introduction

In this chapter, we describe how we detect risk conditions and inform drivers to take proper actions. The causes of risk conditions can be categorized into two main types. One type is caused by dangerous driving behaviors made by drivers with bad physical conditions. We want to detect this type of condition and inform drivers to stop driving. For this purpose, a finite-state transition analysis method is developed in this study, which is described in Section 6.2. The other type of risk condition is caused by neighboring vehicles. For example, a neighboring vehicle might move too close and not keep a safety distance. An even worse case is that the driver can not monitor the entire surrounding of the car simultaneously. Then a traffic accident will occur easily. Thus a risk condition detection method by tracking neighboring vehicles is developed in this study, which is described in Section 6.3. When a risk condition is detected, the system will inform the driver to take care of the condition. Finally, an overall risk detection algorithm is described in Section 6.4.

6.2 Risk Detection by Vehicle

Movement Analysis and Finite-State Transition Analysis

In this section, we describe how we detect risk conditions by the results of the proposed vehicle movement analysis method described in Chapter 4. The result of the method is a description of the vehicle movement condition of the current input image. For each input image, we can find the vehicle movement condition of it. A finite-state transition analysis method is proposed here. Vehicle movement conditions are identified in this study to be of three different types, namely, straight driving, turning, and lane changing. Each condition represents a state in our finite-state transition model. Figure 6.1 shows a state transition diagram of the finite-state transition model used in this study. This model consists of 5 states, namely, the straight driving state, the turning state, the lane changing state, the risk condition state, and the abnormal terminating state. The states of straight driving, turning, and lane changing represent the current movement of the vehicle. The risk condition state means that the vehicle encounters a risk condition. The abnormal terminating state means that an error occurs.

When a driver starts to move his vehicle, the vehicle movement state transits into the straight driving state. Normally, the vehicle movement state stays at the straight driving state as the vehicle is moving forward. When the driver changes the driving path from a lane to the left one or the right one, the vehicle movement state transits from the straight driving state to the lane changing state. After finishing lane changing, the vehicle movement state will return to the straight driving state. The state transition of the turning condition during driving is just like the lane changing condition, as shown in Figure 6.1. These are normal driving behaviors during driving.

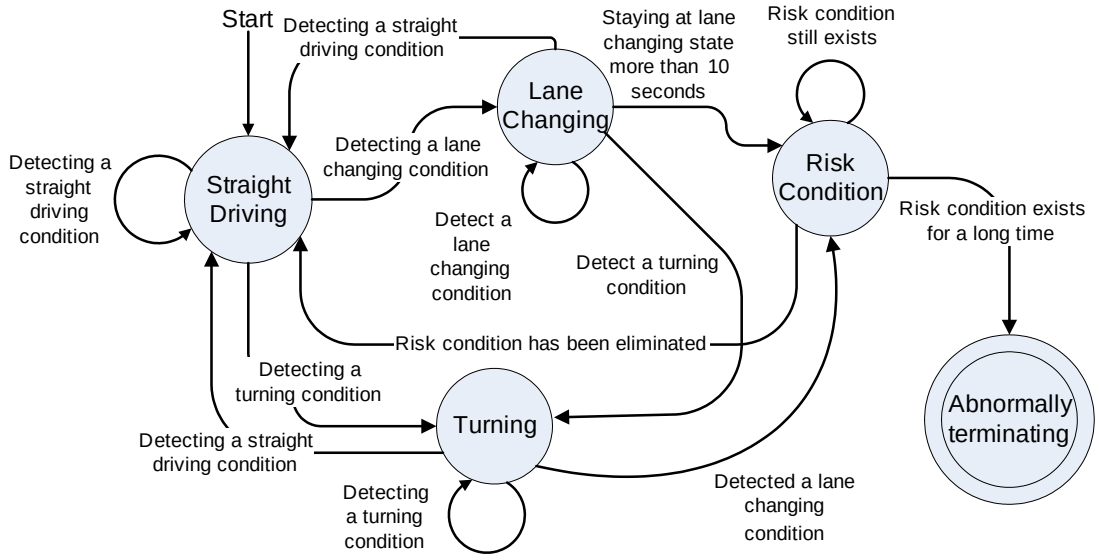


Figure 6.1 State transition diagram of proposed finite-state transition model.

Risk conditions can be detected from the specific kinds of state transitions. Normally, after a vehicle finishes turning, the vehicle movement should return to the straight driving state. If a turning condition is followed by a lane changing condition, this is an illegal transition and the state will transit into the risk condition state. It is dangerous to change lane during turning conditions. Besides, there are still two conditions that will be determined to be risk conditions. One condition is that the vehicle stays at the lane changing state for a long time. This condition means that the vehicle keeps driving across a lane line which is a kind of dangerous driving behavior. Another condition is that the vehicle movement state transits between the straight driving and lane changing states frequently within a short time. This condition means that the driver is conducting a snake driving which is also a kind of dangerous driving behavior. If one of the risk conditions described above is detected, the vehicle movement state will transit to the risk condition state and a sound is played to inform the driver to take actions. If the detected risk condition is removed later, the vehicle movement state will return to the straight driving state. If the detected risk condition

lasts for a long time, the vehicle movement state will transit to the abnormally terminating state. The system will play a warning sound contiguously.

6.3 Risk Detection by Detection and Tracking of Neighboring Vehicles

In this section, we describe how we detect the risk conditions caused by neighboring objects. By using the proposed method of detection and tracking of neighboring vehicles described in chapter 5, we can track the detected neighboring vehicles. When a vehicle moves too close to our vehicle, it means that a risk condition occurs. To determine whether a neighboring vehicle is moving too close, we have to define distance thresholds for certain warning regions. Our attention mainly falls on the front and back warning regions because most of accidents occurred in these two regions. Figure 6.2 shows the distance thresholds of the front and back warning regions we adopt in this study. When the region of a neighboring vehicle in the front region reaches the distance threshold of the front warning region, a risk condition warning should be issued. And when the region of a neighboring vehicle in the back region reaches the distance threshold of the back warning region, a risk condition warning should be issued, too. If a risk condition is detected, the proposed system will play a specific sound as a warning signal for the driver to take appropriate actions.

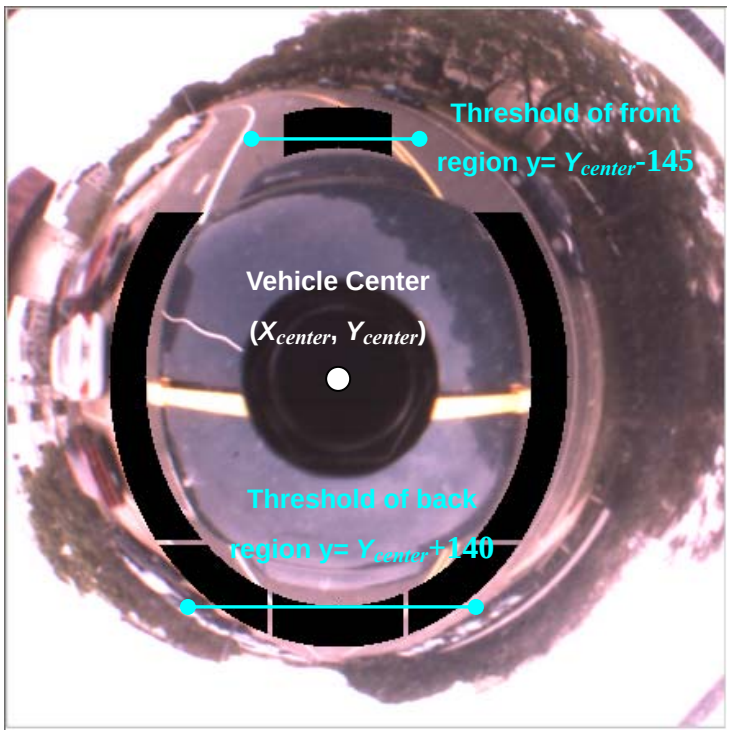


Figure 6.2 Distance thresholds of front and back regions.

6.4 Risk Detection Algorithm

The risk detection algorithm can be categorized into two types, namely, risk detection by vehicle movement analysis and finite-state transition analysis, and risk detection by detection and tracking of neighboring vehicles, as described in Section 6.2 and Section 6.3, respectively. The algorithm of each type is described in the following.

Algorithm 6.1. *Risk detection by vehicle movement analysis and finite-state transition analysis.*

Input: (1) the detected vehicle movement condition in each grabbed image; (2) a finite-state transition model.

Output: whether the current vehicle movement condition is a risk condition or not.

Steps:

Step 1. For each detected vehicle movement condition, transit the vehicle movement state by the state-transition model as shown in Figure 6.1.

Step 2. Check and apply the detected transitions. If an illegal transition or one of the two risk conditions described in Section 6.2 is detected, transit the state into the risk condition state and go to Step 3. Otherwise, stay at Step 2.

Step 3. Keep checking the vehicle movement state, and if the state stays at the risk condition state for a long time, then go to Step 4. Otherwise, return to Step 2.

Step 4. Transit the vehicle movement state to the abnormal terminating state and play a warning sound contiguously.

The main idea of this algorithm is described in Section 6.2.

Algorithm 6.2. *Risk detection by detection and tracking of neighboring vehicles.*

Input: (1) the detected neighboring vehicles in each input image; (2) the neighboring vehicles in the tracking list as described in Section 5.4.

Output: whether there is a neighboring vehicle moving too close or not.

Steps:

Step 1. For the neighboring vehicles in the tracking list, find their corresponding objects by the Algorithm 5.1 described in Section 5.4.1.

Step 2. For each corresponding object in the front and back warning regions obtained in Step 1, check the region of the object. If the region of an object reaches one of the thresholds described in Section 6.3, go to Step 3. Otherwise, stay at Step 2.

Step 3. Determine that a risk condition is detected and play a warning sound of the front or back region. Eliminate the risk condition and return to Step 2 if the detected neighboring vehicle keeps a safety distance with the vehicle

The main idea of this algorithm is described in Section 6.3.

Chapter 7
Experimental Results and Discussions

7.1 Experimental Results

In this section, we will show results of certain driving experiments in outdoor environment. All experiments of this study were conducted in the roads surrounding National Chiao Tung University. In Section 7.1.1, a vehicle movement analysis result is shown, and in Section 7.1.2, some neighboring vehicle detection results are shown.

7.1.1 Vehicle Movement Analysis

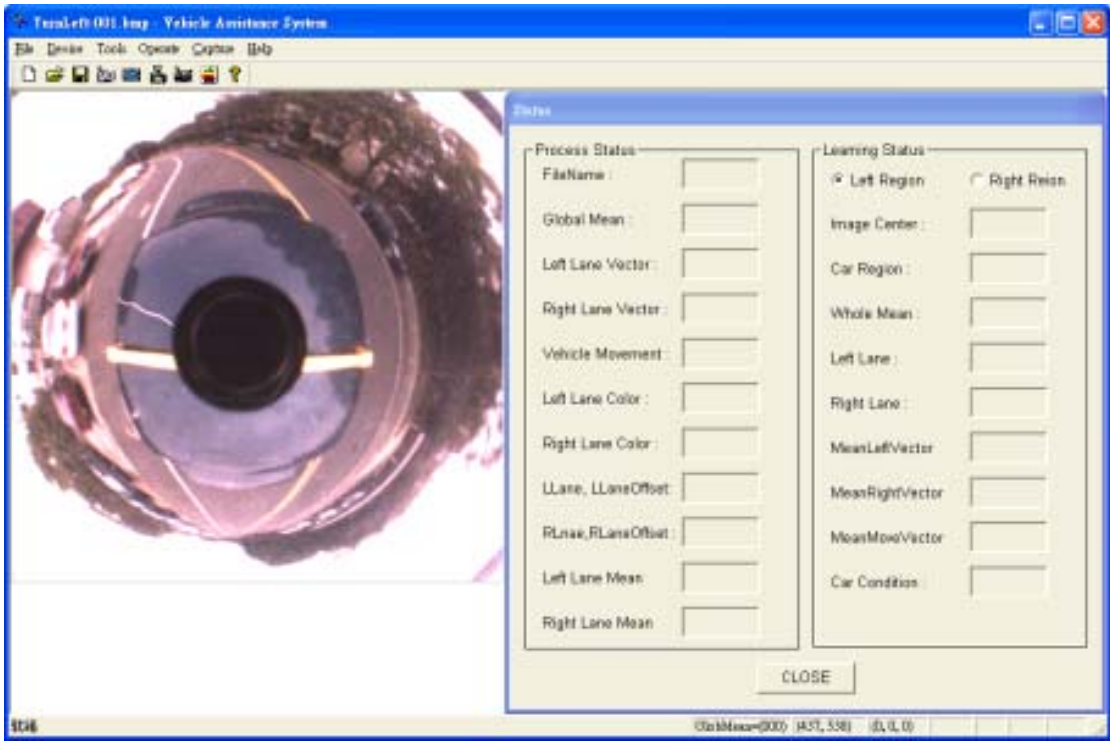
In this experiment, we want to show the detection result of the turning condition in this study. In a learning stage, we drove our vehicle to a starting location. Image (1) in Figure 7.1 shows the GUI interface of the system and a grabbed image for learning. Then the learning process started to analyze the first grabbed image and the learning result was shown in Image (2) in Figure 7.1. Then we started to drive our vehicle along a road. Firstly, a straight driving condition was detected in Image (4) in Figure 7.1. Then the vehicle movement changed into the straight driving condition. Later, the road started to curve, and a turning condition was detected, as shown in Images (5) through (6) in Figure 7.1. The vehicle movement condition changed to the left turning

condition later. Finally, the road became straight. Thus a straight driving condition was detected again, as shown in Images (7) through (8) in Figure 7.1, and the vehicle movement returned to straight driving.

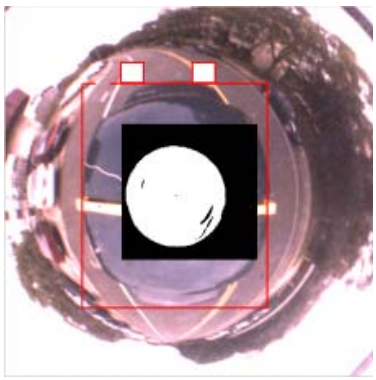
In this experiment, the average speed of the vehicle was about 35 km/hour.

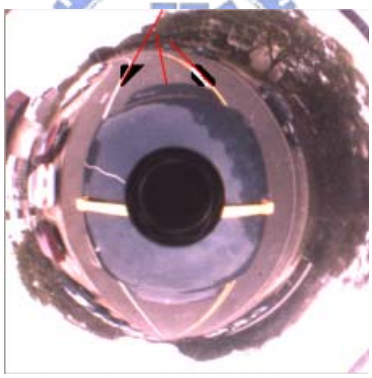
7.1.2 Neighboring Vehicle Detection

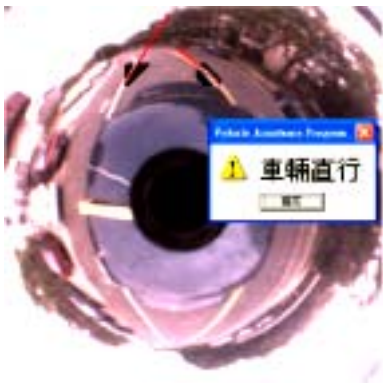
In this experiment, we want to show the result of neighboring vehicles in the front warning region. In the learning stage, the process was just like the one in the previous experiment. Image (1) in Figure 7.2 shows the learning result. Because the left side of the vehicle was determined to be not a lane, the warning region of the left side was disabled. Then we started to drive our vehicle to approach the front vehicle. Later, the neighboring vehicle in the front region entered the warning region and was detected by the system, as shown in Image (3) in Figure 7.2. And the detected neighboring vehicle was found to be approaching our vehicle. Then the system started to track the detected vehicle as shown in Image (4) in Figure 7.2. Later, we kept the same speed as the neighboring vehicle. Then the system detected nothing, as shown in Images (5) through (8) in Figure 7.2. But the neighboring vehicle was not removed from the tracking list because it was detected approaching our vehicle; instead, the system kept on tracking it till the end of the experiment. Although some noise appeared in the grabbed image during driving, they were not classified as objects of interests and were ignored.



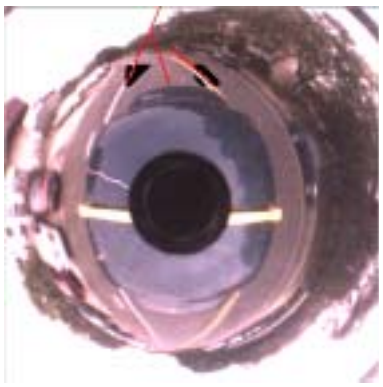
(1)

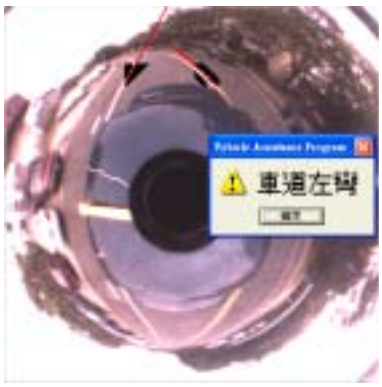


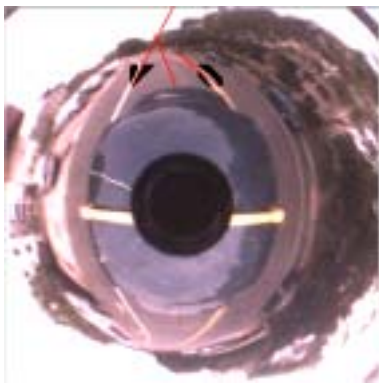




(2) (3) (4)

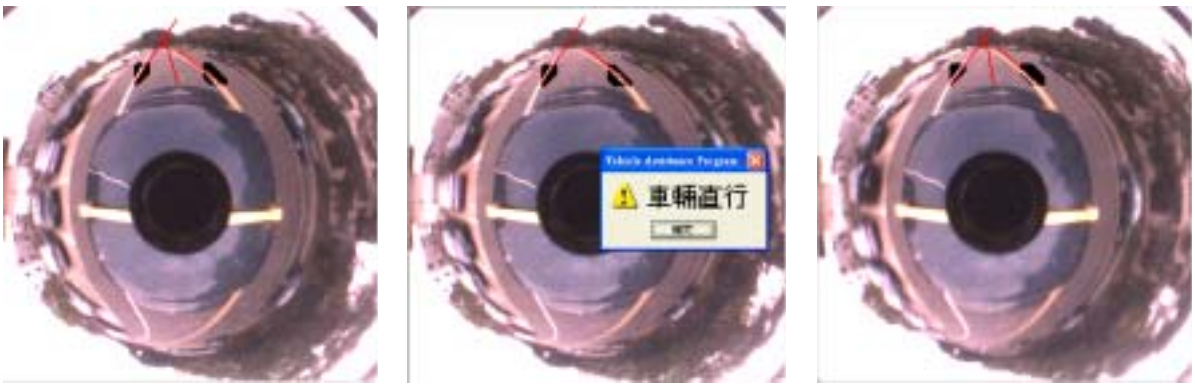






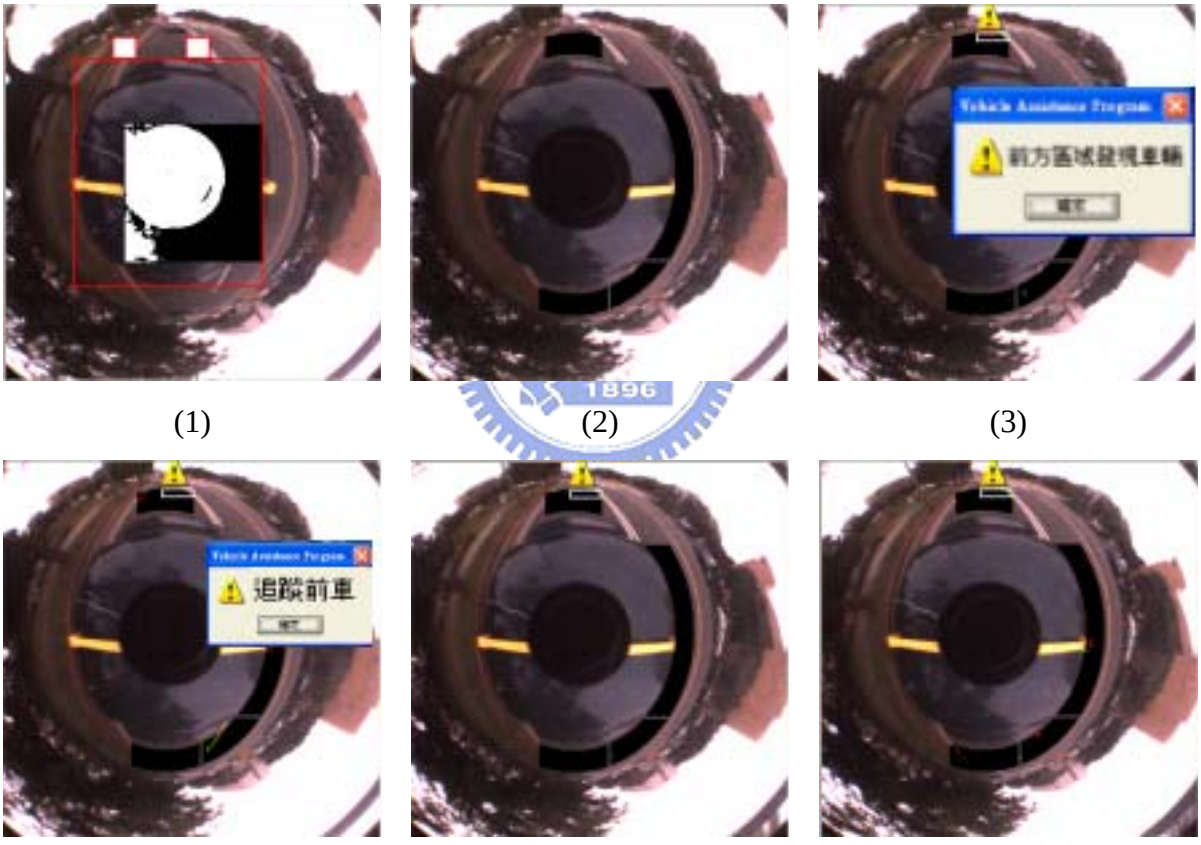
(5) (6) (7)

Figure 7.1 Experimental results of vehicle movement analysis.



(8) (9) (10)

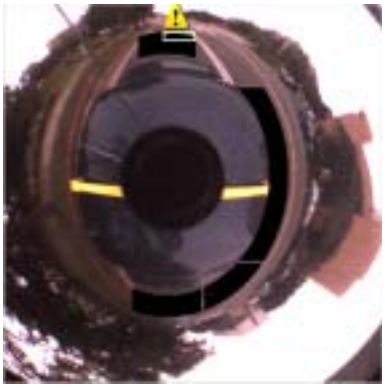
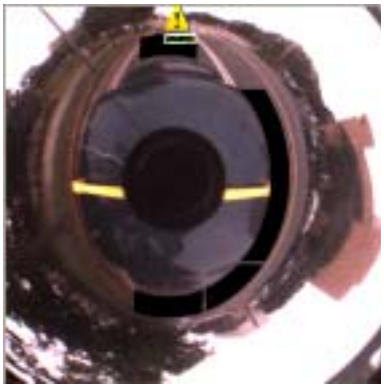
Figure 7.1 Experimental results of vehicle movement analysis (continued).



(1) (2) (3)

(4) (5) (6)

Figure 7.2 Experimental results of neighboring vehicle detection.

(7) (8)

Figure 7.2 Experimental results of neighboring vehicle detection (continued).

7.2 Discussions

We discover certain problems by analyzing the experimental results. And these problems are described as follows.

(1) The lane lines on the road in outdoor environment have to be as clear as possible. If the lane lines are blurred or discontinued, the proposed lane detection module may get erroneous results. But our system can only dispose the erroneous results; it can not avoid this kind of error.

(2) The time interval between two grabbed images has to be as small as possible. If the time interval is large, the image difference between two grabbed images is large. This may cause problem to the classification of neighboring objects. Thus we need to make the time interval as short as possible.

(3) The size of the vehicle used in this study has to have a fixed size. If we use a vehicle with its size larger or smaller than a normal car, the vehicle region defined in the learning process need to be re-defined.

Chapter 8
Conclusions and Suggestions for Future Works

8.1 Conclusions

In this study, several vision-based techniques and methods have been proposed and integrated into a system with autonomous learning capabilities to achieve the goal of computer assisted driving.

At first, an autonomous learning process has been proposed. Only an input image is acquired for the process and all the needed features or parameters are extracted automatically.

Next, a vehicle movement analysis method has been proposed to detect and analyze the lane lines in grabbed images. A dynamic thresholding technique is employed for use in the lane detection process to enhance the correctness of the analysis result. Three kinds of vehicle movement conditions, namely, straight driving, turning, and lane changing, are identified by the proposed analysis method. Some error tolerance techniques are also used to reduce failure rates.

Then a detection and tracking of neighboring objects method has also been proposed. Three kinds of neighboring objects, namely, neighboring vehicle, obstacle on the ground, and non-object, can be classified by the proposed object classification method. Furthermore, a neighboring vehicle tracking algorithm is proposed to track detected neighboring vehicles.

Finally, a risk condition detection method has been proposed. A state-transition analysis method has been proposed to detect the risk conditions caused by dangerous driving behaviors.

8.2 Suggestions for Future Works

The proposed techniques and methods, as mentioned previously, have been implemented on a system. Several related interesting issues are worth further investigation in the future. They are described as follows:

(1) designing an omni-directional camera with a capability of adjusting its height automatically for different scenes of outdoor environment;

(2) simplifying the methods used in this study to reach real-time image processing;

(3) adding the capability of measuring the vehicle movement speed simultaneously;

(4) adding the capability of recognizing pedestrians on the road;

(5) adding the capability of classifying moving and non-moving objects on the road;

(6) expanding the state-transition model to detect more risk conditions.

References

[1] Jos´e Gaspar, Niall Winters, and Jos´e Santos-Victor, “Vision-based Navigation and Environmental Representations with an Omni-directional Camera,” *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, Vol. 16, DEC. 2000, pp. 890-898.

[2] Tarak Gandhi and Mohan M. Trivedi, “Motion Based Vehicle Surround Analysis Using an Omni-Directional Camera,” *IEEE Intelligent Vehicles Symposium*, June 2004, pp. 560-565.

[3] Thomas Bücher, Cristobal Curio, Johann Edelbrunner, Christian Igel, David Kastrup, Iris Leefken,Gesa Lorenz, Axel Steinhage, and Werner von Seelen, “Image processing and behavior planning for intelligent vehicles,” *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, Vol.50, Issue 1, Feb. 2003, pp. 62-75.

[4] Christopher E. Smith, Charles A. Richards, Scott A. Brandt, and Nikolaos P. Papanikolopoulos, “Visual Tracking for Intelligent Vehicle-Highway Systems,” *IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY*, Vol. 45, NOV. 1996, pp. 744-759.

[5] Young UK Yim and Se-Young Oh, “Three Features Based Automatic Lane Detection Algorithm (TFALDA) for Autonomous Driving,” IEEE TRANSACTIONS ON INTELLGIENT TRANSPORTATION SYSTEMS, Vol. 4, DEC. 2003, pp. 219-225.

[6] Takagi, T. Nishi, T. and Yasuda, D., “Computer assisted driving support based on intention reasoning,” *Proceedings of 26th IEEE Annual Conference on Industrial Electronics Society*, Nagoya, Japan, Vol. 1, OCT. 2000, pp. 505-508.

[7] Y. C. Chen and W. H. Tsai, “Vision-based autonomous vehicle navigation in complicated room environments with collision avoidance capability by simple learning and fuzzy guidance techniques,” *Proceedings of 2004 Conference on Computer Vision, Graphics and Image Processing,* Hualien, Taiwan, Republic of China.

[8] C. C. Lai and W. H. Tsai, "Location Estimation and Trajectory Prediction of Moving Lateral Vehicle Using Two Wheel Shapes Information in 2-D Lateral Vehicle Images by 3-D Computer Vision Techniques," *Proceedings of 2003 IEEE Conference on Robotics and Automation,* Taipei, Taiwan, Republic of China.

[9] H. Y. Chiu and W. H. Tsai, “Automatic Vehicle Navigation and Parking in

Building Corridors Using Panoramic Sensing and 2D Image Analysis Techniques," *Proceedings of 2002 Conference on Computer Vision, Graphics and Image Processing,* Hsinchu, Taiwan, Republic of China.

[10] R. C. Liu and W. H. Tsai. "Security patrolling in building corridors by multiple-camera computer vision and automatic vehicle navigation techniques," *Proceedings of 2001 Conference on Computer Vision, Graphics and Image Processing,* Pingtung, Taiwan, Republic of China..

[11] Thomas H. Cormem, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithm*, MxGraw-Hill, Reading, London, England, 2001.

[12] Rafael C. Gonzalez and Richard E., *Digital Image Processing*, Addison-Wesley, Reading, MA, New York, U.S.A., 1993.