國 立 交 通 大 學

多媒體工程研究所

碩 士 論 文

以雙鏡面環場攝影機及自動車作人行道上導盲犬之研究

A Study on Construction of a Machine Guide Dog Using a Two-mirror Omni-camera and an Autonomous Vehicle on Sidewalks

研 究 生：黃致瑋

指導教授：蔡文祥 教授

中 華 民 國 一 百零一 年 六 月

以雙鏡面環場攝影機及自動車作人行道上導盲犬之研究

A Study on Construction of a Machine Guide Dog Using a Two-mirror Omni-camera and an Autonomous Vehicle on Sidewalks

研 究 生：黃致瑋　　Student：Chih-Wei Huang

指導教授：蔡文祥　　Advisor：Wen-Hsiang Tsai

國 立 交 通 大 學
多 媒 體 工 程 研 究 所
碩 士 論 文

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

Computer Science

June 2012

Hsinchu, Taiwan, Republic of China

中華民國一百零一年六月

A Study on Construction of a Machine Guide Dog Using a Two-mirror Omni-camera and an Autonomous Vehicle on Sidewalks

Student: Chih-Wei Huang Advisor: Wen-Hsiang Tsai

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

ABSTRACT

Various techniques for construction of a machine guide dog using a two-mirror omni-camera and an autonomous vehicle for navigations on sidewalks are proposed. The autonomous vehicle can compute 3D information from acquired omni-images to localize itself using pre-selected landmarks, and guide a blind person to follow a planned path to a destination on a sidewalk. Firstly, a method for learning the sidewalk environment is proposed to construct a navigation map, including a navigation path, along-path landmark locations, and relevant vehicle guidance parameters. Next, a navigation system with self-localization and automatic guidance capabilities using landmarks including curb lines, tree trunks, stop lines on roads, lawn corners, traffic cones, and signboards is proposed. By the use of a space-mapping technique, three space line detection techniques for use directly on the omni-image are proposed, which can be used to compute the 3D position of a specific space line in the shape of a sidewalk landmark.

Moreover, based on the techniques for detecting three space lines, techniques for detections and localizations of the above-mentioned natural and artificial landmarks are proposed. Using these vehicle self-localization techniques, imprecise vehicle positions due to incremental mechanic errors can be adjusted. In addition, for the purpose of continuous navigation, a curb line following technique is proposed as well to guide the vehicle along a sidewalk when landmarks are not available during the navigation process. To detect landmarks in the outdoor environment, techniques for dynamic threshold adjustments are also proposed for adapting the system’s capability to varying lighting conditions in navigation environments.

Good experimental results showing the flexibility and feasibility of the proposed methods for real applications are also included.

以雙鏡面環場攝影機及自動車作人行道上導盲犬之研究

研究生：黃致瑋　　　　指導教授：蔡文祥 博士

國立交通大學多媒體工程研究所

摘要

本研究提出了一個應用於機器導盲犬在人行道上行走的自動車系統，利用一部搭載雙鏡面環場攝影機的自動車當實驗平台，在環場影像中直接求出實際物體的立體資訊，並引領盲人沿著規劃的路徑行走。首先，我們利用建立的環境學習系統來記錄導航地圖，此地圖包含自動車在戶外環境中行走的路徑、路徑中附近路標的位置，以及相關的導航參數。接著，利用人行道上常出現的路標(人行道路緣、樹幹、草地角點、交通錐、道路白線和招牌)作自動車定位，以輔助導航。

此外，我們提出空間映射的方法來發展新的直線偵測技術，在環場影像上直接偵測出直線特徵，並計算出人行道上與自動車相互平行或垂直的直線位置，藉以偵測前述自然特徵物及人工特徵物之位置，作機械誤差之校正，並計算出正確的自動車位置。

最後，本研究也提出動態調整門檻值的技術，讓系統適應戶外環境的各種光影變化所造成的影響。實驗結果顯示本研究所提方法完整可行。

ACKNOWLEDGEMENTS

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

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

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

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

1.1 Motivation

There are millions of blind people in the world. Some of them use blind canes to travel on the road. However, blind canes can only be used to detect obstacles at short distances by blind people. Besides using blind canes, a better choice is to use guide dogs. Guide dogs can assist blind people to avoid obstacles and walk in indoor or outdoor environments smoothly even if there is no barrier-free facility along the way. For the welfare of blind people, guide dogs not only improve the safety, but also enhance the quality, of their lives.

However, there are few guide dogs in Taiwan. According to the information provided by Taiwan Guide Dog Association [20] and Taiwan Foundation for the blind [21] in 2012, there are almost sixty thousand blind people but just twenty-eight guide dogs in Taiwan. This rate is too low to provide sufficient supports for the blind people. The reasons why guide dogs are so few are listed below:

1. only certain breeds of dogs can be trained as guide dogs;
2. a trainer has to spend very much time to teach a young guide dog;
3. the individual difference between the personalities of the master and the guide dog is a problem which should be solved;
4. it costs at least one million NT dollars to train a guide dog.

It is wished in this study to utilize technology and knowledge to solve this problem of guide dog shortage, so we follow the idea of providing a *machine guide dog* for each blind person. Each machine guide dog can be a replacement for a real one because it can be manufactured very quickly once designed to be effective. A machine guide dog basically is constructed by the use of an autonomous vehicle and a camera. To implement a machine guide dog, it is desired that a vision-based autonomous vehicle can automatically navigate in outdoor environments and keep watch over the camera's field of view (FOV) automatically. It is also hoped that when the vehicle detects a risk area like a hole or an obstacle along the way, it can safely guide the blind person to avoid the danger; and when the autonomous vehicle reaches the destination, it will inform the blind person so.

For this purpose, the critical problem is how to navigate a vision-based autonomous vehicle successfully in outdoor environments. Normally, an autonomous vehicle is equipped with an odometer, and we can use the odometer readings to compute the current position of the vehicle with respect to its initial position of a navigation session. However, the position which the odometer provides is often not sufficiently precise because the autonomous vehicle usually suffers from incremental mechanic errors due to manufacturing imprecision. One good solution is to continually estimate the vehicle position by monitoring natural or artificial objects in the surroundings along the navigation path, which is a sidewalk in this study.

In normal cases, a blind person has to pass regular scenes with objects along sidewalks, so we may train the autonomous vehicle beforehand to recognize the path and the surrounding objects, just like training a guide dog in the place. That is to say, we may let the autonomous vehicle to "remember" along-path *landmarks* in advance, and construct the on-board navigation system to capture the current landmark information during each navigation session.

In summary, the goal of this study is to develop a vision-based autonomous vehicle for use as a machine guide dog on sidewalks. The system is expected to possess the following capabilities:

1. learning paths on sidewalks semi-automatically;

2. learning common landmarks along sidewalks;
3. navigating automatically along sidewalks using learned landmarks for localization;
4. navigating to goals successfully on learned paths.

1.2 Survey of Related Works

In this section, we conduct a survey of related works about assistances to the blind people, including new application systems for the blind, localization techniques in indoor or outdoor environments, and landmark detection via the use of stereo omni-cameras.

More and more research results about developing walking aids for the blind people have appeared. As an improvement on the blind cane, an easy way is to install a sensor device on a blind cane so as to detect obstacles at a certain distance. Borenstein and Ulrich [1] developed the “GuideCane” which can detect obstacles by ultrasonic sensors to help blind people to pass dangerous areas. Some helpful navigation systems for the blind were also proposed. Ivanchenko et al. [2] proposed a system for impaired wheelchair users to detect the presence of obstacles or other terrain features by computer vision techniques and warn the user. A robotic travel aid (RTA) called HITOMI was proposed by Mori et al. [3] which can detect vehicles and pedestrians using multiple sensors. Besides, Hsieh [4] utilized two cameras installed on a cap to find accessible obstacles and regions in unknown environments and alert the impaired person by voice. Likewise, Shang [5] used two cameras assembled on the shoulder to walk independently in known or unknown environments.

In addition, localization is an important issue in implementing a navigation system. Willis and Helal [6] provided a navigation system for the blind which uses the radio frequency identification (RFID) technology to identify locations in buildings and rooms. Lisa et al. [7] utilized a DGPS (differential GPS) device to localize a blind person in indoor and outdoor environments. Also, Chen and Tsai [8] proposed an indoor autonomous vehicle navigation system using ultrasonic sensors. In outdoors, the GPS can be used as a localization system for the vehicle [9].

Furthermore, vision-based sensors have been widely utilized for vehicle navigation. Atiya and D. Hager [10] proposed the vision-based system which computes the location in real-time. Chen and Tsai [11] proposed a vehicle localization technique which modifies the position of a vehicle by keeping watch over learned objects. Another method of vehicle localization in indoor environments by watching house corners was proposed by Chiang and Tsai [12]. Moreover, a system which uses stereo cameras and a low-cost GPS sensor was proposed by Agrawal and Konolige [13]. Tsai and Tsai [14] used a PTZ camera and an ultrasonic sensor to direct vehicle patrolling and people following successfully. Another application using a combination of cameras and other devices was proposed by Lopez et al. [15], who connected a laser and a robot’s camera to compute the robot location.

An omni-directional camera has the advantage of having a large FOV in contrast with a traditional CCD camera. Because of this advantage, we choose to use the omni-camera to design the machine guide dog system in this study. So, we conduct a survey of related works about vehicle navigation systems using omni-cameras as well here. To enhance the accuracy of localization, Lui and Jarvis [16] implemented an algorithm which implements omni-directional vision on a GPU. To detect landmarks in environments, Fu et al. [17] proposed a navigation system with embedded omni-vision for landmark recognition. A method which was conducted to achieve vehicle self-localization by matching omni-directional images was proposed by Ishizuka et al [18]. Wu and Tsai [19] detected circular landmarks on ceilings to conduct vehicle indoor navigation.

1.3 Overview of Proposed System

The goal of this study is to lead a vision-based machine guide dog to navigate in outdoor environments automatically. The most important task to achieve this goal is *vehicle localization*. The method for vehicle localization we propose is to detect landmarks along the path. Besides, some strategies for navigation on the learned path are proposed for use in this system. In this section, we will introduce the vision-based autonomous vehicle system which we use in this study. The operation process of this system may be divided into two stages: the learning stage and the navigation stage. The learning stage includes primarily the task of training the autonomous vehicle to acquire the along-path information useful for vehicle guidance before navigation. Then, we conduct vehicle navigation along the path which we pre-select in the navigation stage. The details of the two stages are illustrated in Figs. 1.1 and 1.2, respectively, and discussed in the following.

A. The learning stage

In the process of learning which is necessary before the vehicle can navigate, at first we combine the camera system into the autonomous vehicle. Unfortunately, the camera system does not perfectly match the vehicle's structure. It should so be calibrated to find the relation between the image coordinates and the real-world locations. In this system, we utilize a two-mirror omni-camera system as the visual sensor. Because of the special structure of the two-mirror omni-camera system, its parameters cannot be acquired and calculated easily. Therefore, we adopt a space-mapping method proposed by Jeng and Tsai [20] to solve the problem. We get accordingly a space-mapping table, called *pano-tabl*e, to calibrate the camera system instead. After we finish the calibration work, we can obtain the range data of concerned feature points in an omni-image directly using the pano-table and continue the navigation. Secondly, we learn relevant path information for vehicle guidance, including the environment parameters, the vehicle position in the path, the location of each used landmark, and some landmark segmentation parameters.

After we guide the vehicle to a particular spot, a path learning work is started. In this phase of the learning stage, we propose the use of two navigation modes: one being *navigation by following the sidewalk*; the other *manual control by the trainer.* If we choose the first mode, the autonomous vehicle starts to navigate to the goal and the information of the vehicle pose at each visited spot is recorded. If we want the system to "memorize" a specific landmark along the path, we can use the second mode to achieve the landmark information acquisition and position estimation. Besides, each path we choose is along a sidewalk in an outdoor environment, and some information about the outdoor environment along the path is also recorded during this phase of learning. Finally, all of the data so learned are integrated into a set of path information.

B. The navigation stage

As we mentioned above, the path information which is acquired in the learning stage is used in the navigation stage. In the navigation stage, three major works are conducted by the vehicle. One is moving forward, another is obstacle detection, and the third is vehicle location estimation and modification.

Generally, the vehicle can move forward continually toward the goal we pre-select. Between every two nodes in a path, the vehicle can choose one of two navigation modes— *navigation by the vehicle odometer* (called the *blind navigation mode*) and *navigation by the side walk*. When the *navigation by the side walk* mode is selected, the vehicle detects the sidewalk curb with a prominent color continually and adopts a line following technique to guide the vehicle. When no curb can be used along the current path segment, the system is set in the *navigation by the vehicle* *odometer* mode and moves forward "blindly" according to the odometer reading on the pre-selected path. Also, it is desired to find fixed obstacles and have a strategy to pass dangerous areas.

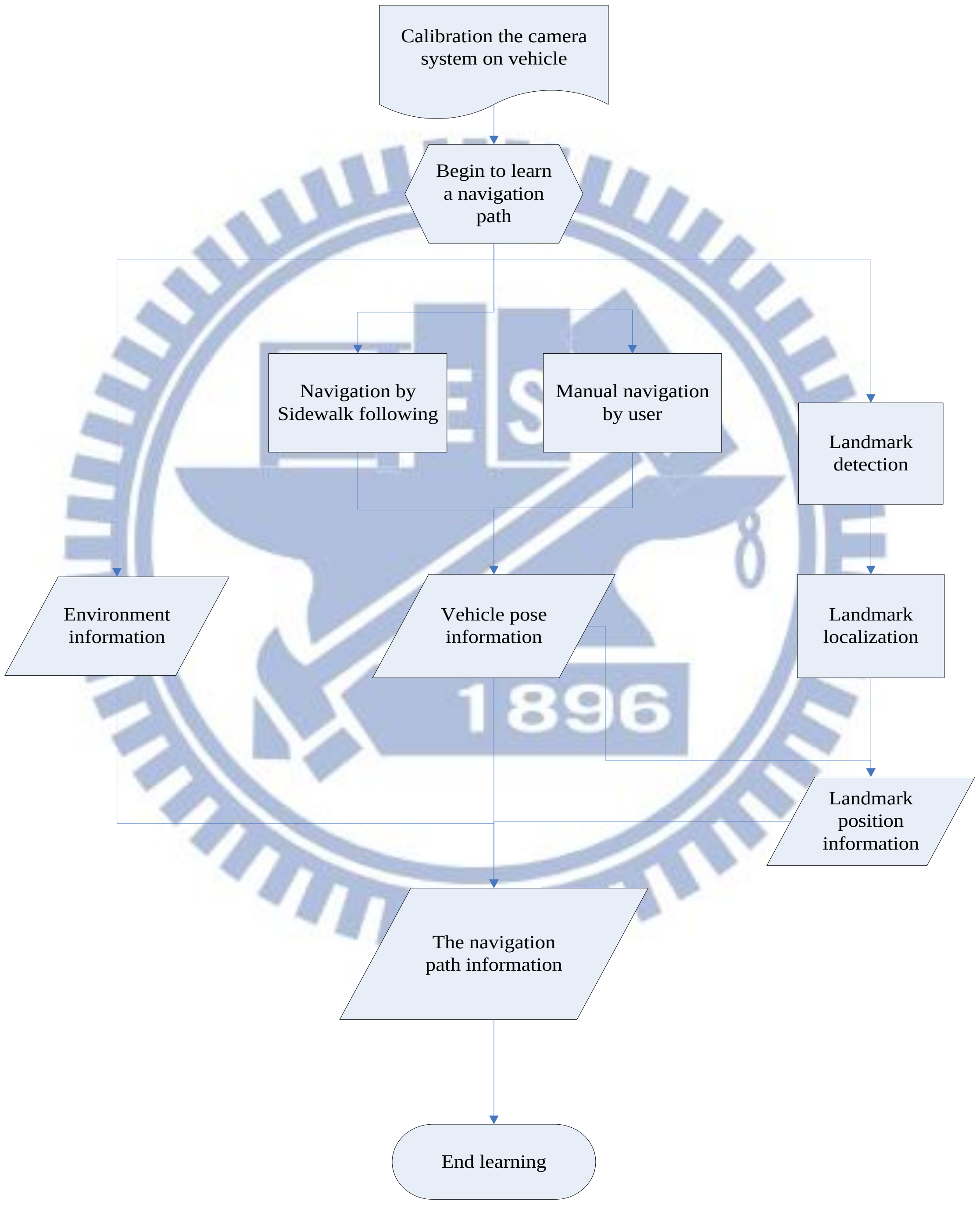

Calibration the camera system on vehicle
Begin to learn a navigation path
Navigation by Sidewalk following
Manual navigation by user
Landmark detection
Environment information
Vehicle pose information
Landmark localization
Landmark position information
The navigation path information
End learning

Figure 1.1 Flowchart of proposed learning stage.

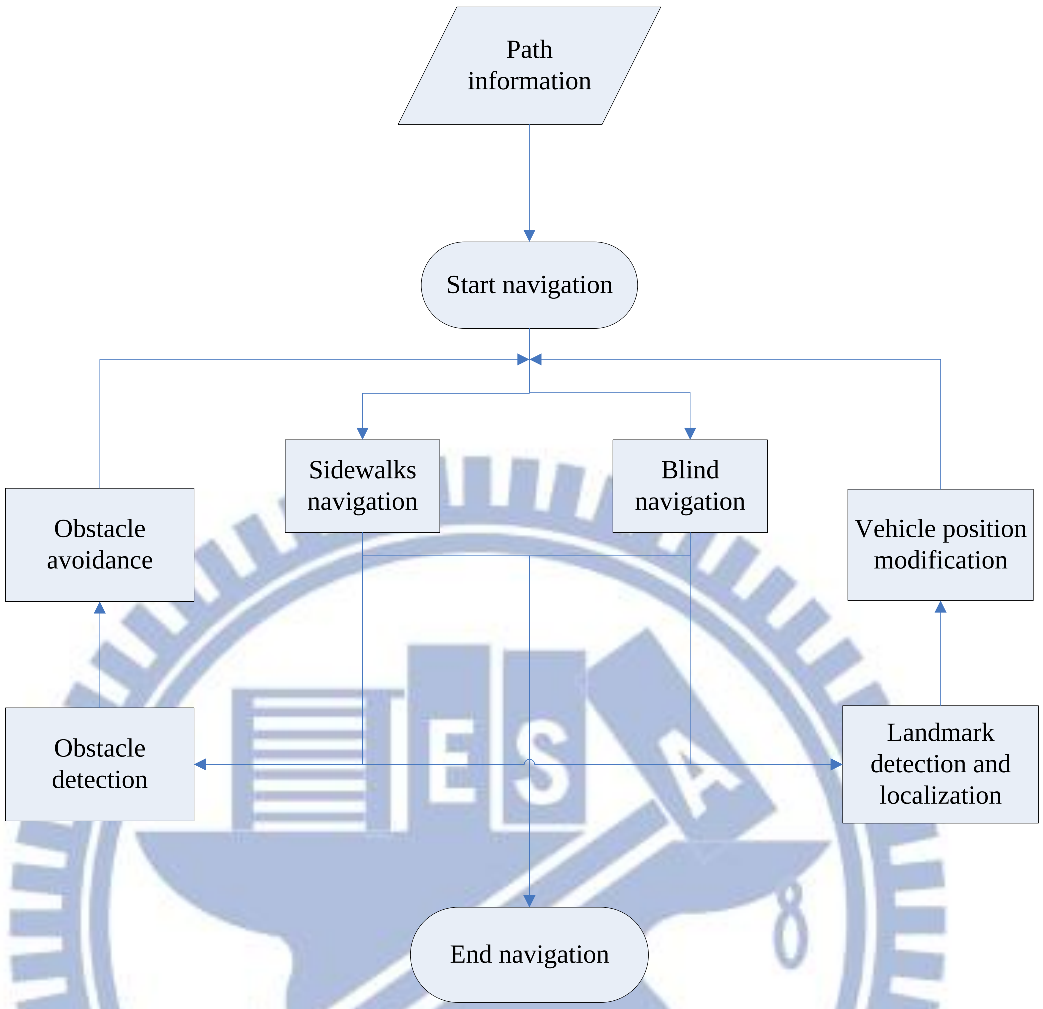


Figure 1.2 Flowchart of navigation stage.

Moreover, we use some objects which are commonly seen along sidewalks, such as trees, road stop lines, traffic cones, signboards, and corner points of the lawn, to localize the vehicle in this study. That is, we modify the vehicle position with respect to each located landmark to eliminate accumulated mechanical or vision-processing errors during the navigation process. Finally, we propose three new vertical space line detection methods to calculate the locations of detected landmarks. By all of these techniques, the autonomous vehicle can navigate safely to the end of the navigation stage hopefully.

1.4 Contribution of This Study

Some contributions of this study are described as follows.

1. A semi-automatic system for training an autonomous vehicle for outdoor navigation along sidewalks using ordinarily-seen objects seen along paths is proposed.
2. Two new space line detection techniques and two localization techniques using the pano-mapping table are proposed.
3. Schemes for detecting natural landmarks like corner points of the lawn and trees for vehicle localization are proposed.
4. Techniques for detection and localization of artificial landmarks (signboards, stop line on road, traffic cones) are proposed.
5. A method for correcting the mechanical errors of the vehicle position resulting from long-time autonomous vehicle navigation is proposed.
6. A method for dynamically adjusting the guidance parameters for outdoor navigation is proposed.

1.5 Thesis Organization

The remainder of this thesis is organized as follows. In Chapter 2, we introduce the configurations of the proposed system and the system processes. In Chapter 3, the proposed method for learning guidance parameters and navigation paths are described. In Chapter 4, we describe the proposed navigation strategies, including ideas, guidance techniques, and detailed navigation algorithms. In Chapter 5, the two proposed new space line detection techniques are introduced and their applications for natural landmark detection are described; and in Chapter 6, their applications for artificial landmark detection are described. In Chapter 7, we show some experimental results of sidewalk navigation to show the feasibility of the proposed system. At last, some conclusions and suggestions for future works are given in Chapter 8.

Chapter 2
System Design and Processes

2.1 Ideas of System Design

As mentioned in Chapter 1, many good facilities for the blind people have been proposed in the past. Using a vision-based autonomous vehicle as a machine guide dog is a good idea because the trainer does not have to spend much time to train it and it can work all day without taking a rest. If the vehicle is equipped with a camera, it will be able to "see" the environment around and avoid obstacles along the way. However, the task of combining the camera and the vehicle system is not easy to accomplish. We need a control unit which connects the camera and the vehicle system, analyzes the acquired image data, integrates all the information, and makes decisions. In this chapter, we will describe in Section 2.2 the software and hardware systems of the proposed machine guidance dog which accomplishes the above-mentioned tasks, and the detail of the proposed method for 3D data acquisition using the camera will be described in Section 2.3.

Because originally the vehicle does not have "knowledge" to navigate on the sidewalk, it will cause accidents, like collisions with obstacles or falling outside the sidewalk. Therefore, before the machine guide dog can navigate by itself, we should "teach" it to know the outdoor information and deal with different conditions. In addition, strict strategies for navigation should be designed to protect the blind people and the vehicle from accidents. Finally, the vehicle should be designed to be capable of navigating along a learned path again and again. To reach the above goals, we have to organize the system processes for the autonomous vehicle system well. The system processes will be described in Section 2.4, including the learning process in Section 2.4.1 and the navigation process in Section 2.4.2.

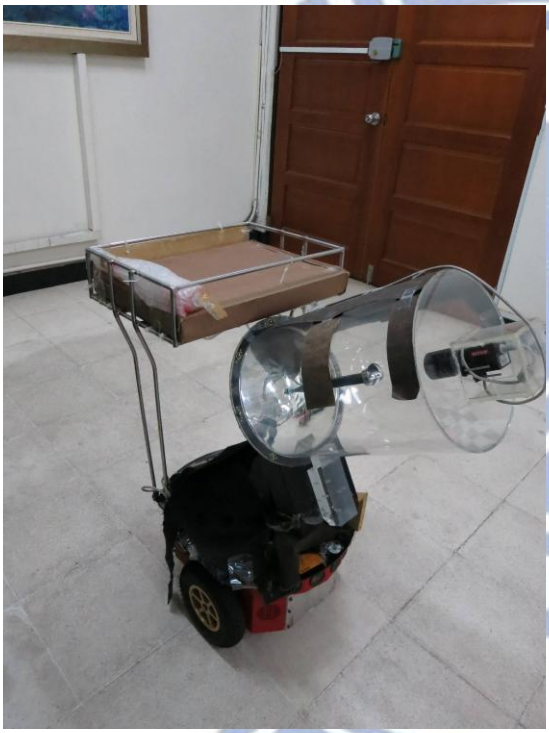
2.2 System Configuration

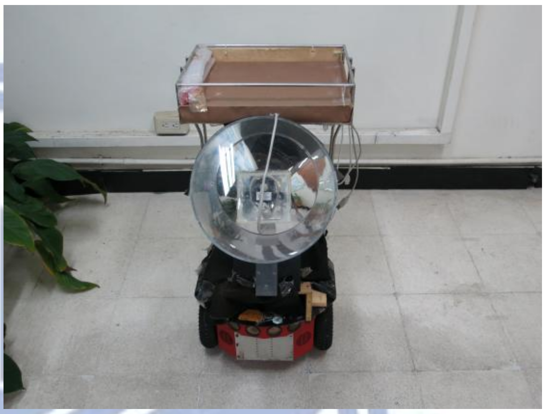
To construct the proposed system, we adopt a Pioneer 3-DX vehicle which is made by MobileRobots Inc. The vehicle is equipped with an imaging system composed of a stereo omni-camera. The imaging system is not only part of the vehicle system but also plays an important role of accumulating the information data and locating the vehicle. The autonomous vehicle and other associated hardware equipments will be introduced in Section 2.2.1, and the camera system will be described in Section 2.2.2. Besides the hardware, software is needed to provide a friendly interface to users in order to control the vehicle conveniently. The software system we develop for use in the study will be described in Section 2.2.3.

2.2.1 Hardware configuration

The hardware architecture of the proposed machine guide dog is shown in Figure 2.1. It can be partitioned into three principal systems: the vehicle system, the camera system, and the control system. We will describe these systems, respectively, in the following.

In the vehicle system, the Pioneer 3-DX as mentioned is shown is Figure 2.2, which has a 44cm×38cm×22cm aluminum body with two 19cm wheels and a 7cm caster. It can reach a speed of 5.76 kilometer per hour on flat floors; has the maximum rotation speed of 300 degrees per second; and can climb up an incline with the largest slope of 30 degrees. Moreover, the vehicle has sixteen ultrasonic sensors. They are installed around the vehicle body. The vehicle can carry payloads up to 23kg. It has three 12V rechargeable lead-acid batteries and can run 18-24 hours if the batteries are fully charged initially. Furthermore, the vehicle is equipped with an odometer which records the pose of the vehicle, including the position and the orientation with respect to its initial pose, for each navigation cycle. The odometer provides also the readings of the vehicle speed, the battery voltage, etc.


(a)


(b)

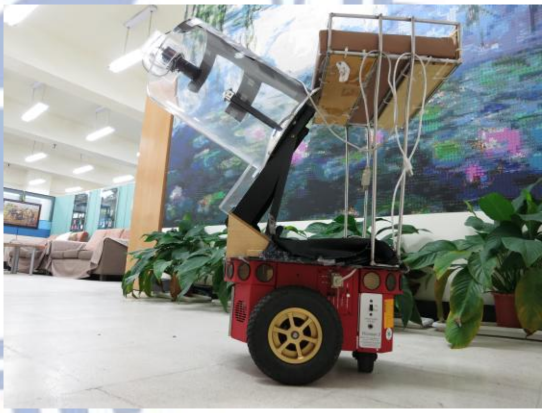
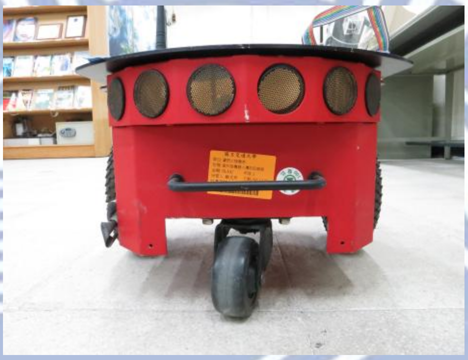

(c)

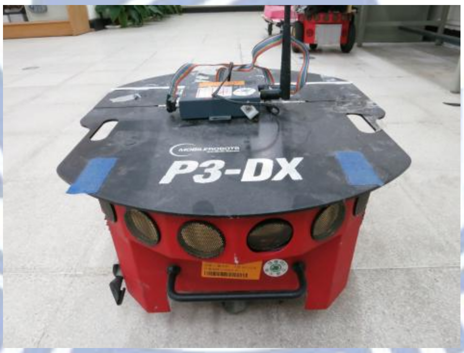
Figure 2.1 Three different views of the used hardware architecture, which includes a vehicle and a stereo camera. (a) A 45^{o} view. (b) A front view. (c) A side view.

The second part of the system hardware is the camera system. It is a two-mirror omni-camera which consists of one perspective camera, one lens, and two reflective mirrors of different sizes, all integrated into a single structure. A picture of the camera system is shown Figure 2.3. The perspective camera and the lens are shown in Figure

2.4. The camera is of the model ARCAM-200SO, which is produced by ARTRAY Company with the size of 33mm×33mm×50mm and the resolution of 2.0M pixels. The detailed specifications of the camera are listed in Table 2.1. The lens is produced by Sakai Co. and has a variable focal length of 6-15mm. The two reflective mirrors are produced by Micro-Star International Co. The structure of the camera system will be described in more detail in the next section.



(a)



(b)

Figure 2.2 The Autonomous vehicle, Pioneer 3-DX, produced by MobileRobots Inc., used in this study. (a) A back view. (b) A front view.

In the control system, we utilize a laptop computer as the main unit. It is of model R840 produced by TOSHIBA Computer Inc. as shown in Figure 2.5. We use an RS-232 to connect the laptop computer and the autonomous vehicle and use a USB to connect the computer and the camera system. The specification of the laptop is listed in Table 2.2.

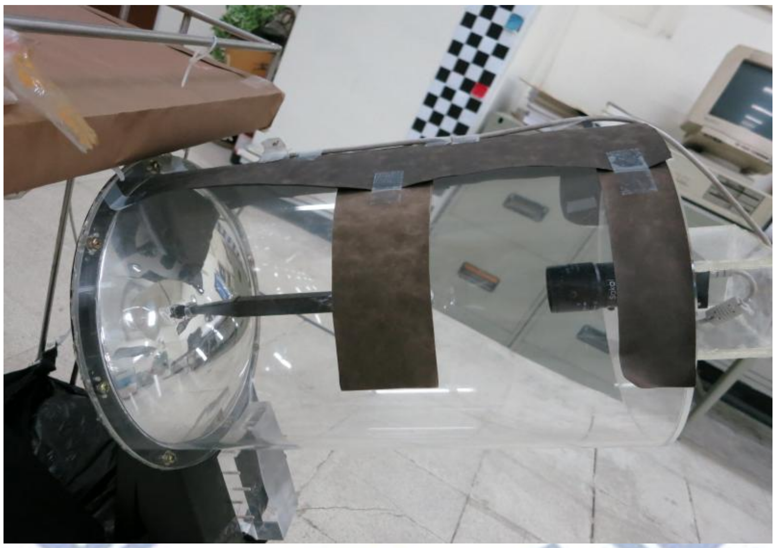
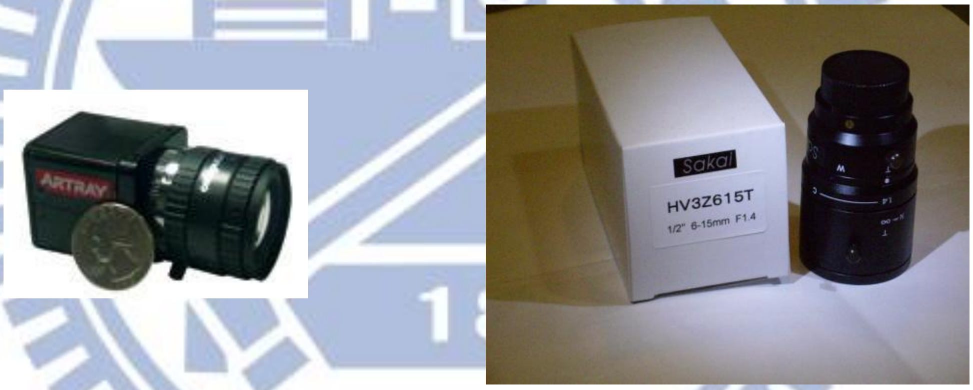


Figure 2.3 The camera system used in this study.



(a) (b)

Figure 2.4 The used camera and lens. (a) The camera of model Arcam-200so produced by ARTRAY Co. (b) The lens produced by Sakai Co.

Table 2.1 The specification of Arcam-200so.

Size	33mm×33mm×50mm
CMOS Size	1/2" (6.4 × 4.8mm)
Mount	C-mount
Max resolution	2.0 M pixels
Frame per second with max resolution	8 fps

Figure 2.5 The laptop computer of model TOSHIBA R840 used in this study.

2.2.2 Structure of used two-mirror omni-camera

In this section we will introduce the two-mirror omni-camera we use in this study. As shown in Figure 2.6, a space point *G* is projected by the two mirrors onto the image plane of the camera system. The light ray coming from point *G* is reflected by the two mirrors to go through the lens center. The two mirrors are both made to be of the hyperboloidal shape. We will call the big mirror *Mirror B*, and the small one *Mirror S*, respectively, in the sequel of this thesis. As is well known, the hyperboloidal shape has two focal points: one being the focal point of *Mirror S* which is denoted by f_s and the other the focal point of *Mirror B* which is denoted by f_b subsequently. The configuration of the two mirrors is designed in such a way that the focal points of the two mirrors are located at an identical point which is just the lens center f_c of the camera. Besides, the distance from the mirror center of Mirror *B* to the mirror center of Mirror *S* has a length of 20 cm according to our manual measurement in this study. We call it the *baseline*. The detailed information of the two hyperboloidal-shaped mirrors is listed in Table 2.3.

Table 2.2 Specification of the laptop computer.

CPU	Intel Sandy Bridge Core i5-2410M 2.3GHz
RAM	4G DDR 1333MHz
GPU	AMD Radeon HD 6450 /1024MB
HDD Size	640 GB

Table 2.3 Specifications of the used two hyperboloidal-shaped mirrors.

	Radius	Parameter a	Parameter B
Mirror S	2 cm	2.41 cm	4.38 cm
Mirror B	12 cm	11.46 cm	9.68 cm

In spite of having two focal points, the hyperboloidal shape has another property as shown Figure 2.7: if a light way goes through one of the focal point, it will be reflected to go through the other focal point by the mirror. This property has been utilized to construct the omni-camera in a previous study [22] . According to this property, a space point *G* will first go into the centers of the two mirrors, then reflected by the mirrors to go through the lens center f_c, and finally projected onto the CMOS sensor of the camera. Therefore, we have two distinct image points corresponding to the single space point *G*. Based on such a phenomenon, we can compute the range data of *G*. The detail will be described later in this chapter.

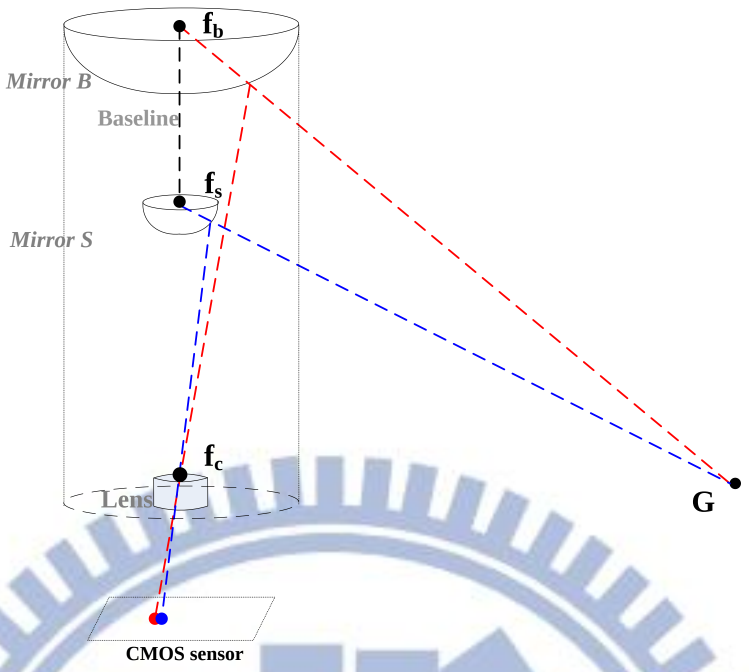


Figure 2.6 An illustration of the two-mirror omni-camera and a space point projected on the CMOS sensor of the camera.

In addition, initially we place the two-mirror omni-camera in such a way that the axis going through Mirror *S* and Mirror *B* is perpendicular to the ground, as illustrated in Figure 2.7. However, it was found out in [22] that in the resulting fields of view (FOV's) of mirrors *B* and *S*, the overlapping area on the ground is too small to be useful for computing precise range data. In this study, it is desired that the FOV is as large as possible. To solve this problem, the camera system is slanted for an angle of γ as shown in Figure 2.8. It can be seen that the overlapping region is now bigger than before.

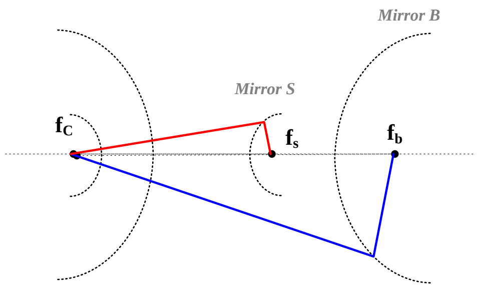
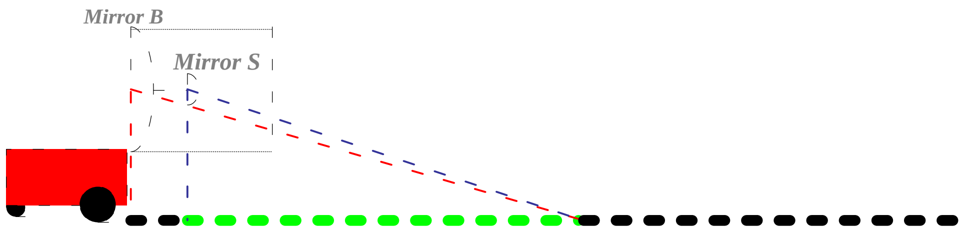
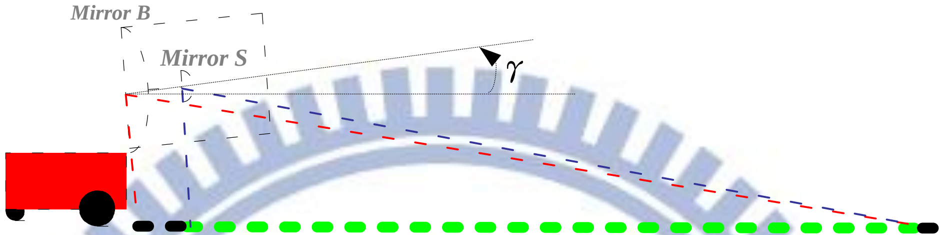


Figure 2.7 The reflection property of the two hyperboloidal-shaped mirrors in the camera system.



(a)



(b)

Figure 2.8 Two different placements of the two-mirror omni-camera on the vehicle and the region of overlapping. (a) The optical axis going through the two mirrors is parallel to the ground. (b) The optical axis through the two mirrors is slanted up for an angle of γ.

2.2.3 Software configuration

MobileRobots Inc., which provides the autonomous vehicle for use in this study, provides an application interface, called ARIA (Advanced Robotics Interface Application), for the user to control the vehicle. The ARIA is an object-oriented interface which can be used under the Linux or Win32 operating system using the C^{++} language. Therefore, we can use the ARIA to communicate with the embedded sensor system in the vehicle and obtain the information which the vehicle offers to control the position of the vehicle.

For the camera system, the ARTRAY provides a tool which is called *Capture Module Software Developer Kit (SDK)*. It is an object-oriented interface and its application interface is written in several computer languages like C, C^{++}, VB.net, C$^{\#}$.net and Delphi. We use the SDK to capture image frames with the camera and change many parameters of the camera, such as the exposure. To develop our control system, we use Borland C^{++} Builder 6 with updated pack 4 on the Windows XP operating system. The Borland C^{++} Builder 6 is a GUI-based interface development environment (IDE) software. It is convenient for us to provide a friendly interface for the user.

2.3 3D Data Acquisition by the Two-mirror Omni-camera

2.3.1 Review of imaging principle of two-mirror omni-camera

In this section, we review the two-mirror omni-camera proposed in Huang and Tsai [22] and used in this study, as well as the formulas for range data computation using images captured by such a camera system. First, we review the image projection principle of an omni-camera. As shown in Figure 2.9, we use the two coordinate systems, the *image coordinate system* (ICS) and the *camera coordinate system* (CCS), to illustrate the principle of imaging process. The image coordinate system is a two-dimensional *U*-*V* coordinate system and the other is a three-dimensional *X*-*Y*-*Z* coordinate system. The origin of the first one is the center of the omni-image, and the second is the focal point of the hyperboloidal-shaped mirror. As mentioned previously, a light ray G at (x, y, z) in the CCS go through the focal point of the hyperboloidal-shaped mirror $O_{m,}$ and reflected by the mirror. Then, it goes through the other focal point at center of lens O_c. Finally it is projected onto an image point I on the omni-image plane at (u, v). As a result, each image point in an omni-image can be specified by an elevation angle α and an azimuth angle θ. After the azimuth angle θ and the elevation angle α are obtained, we are able to calculate the location of the space point *G*.

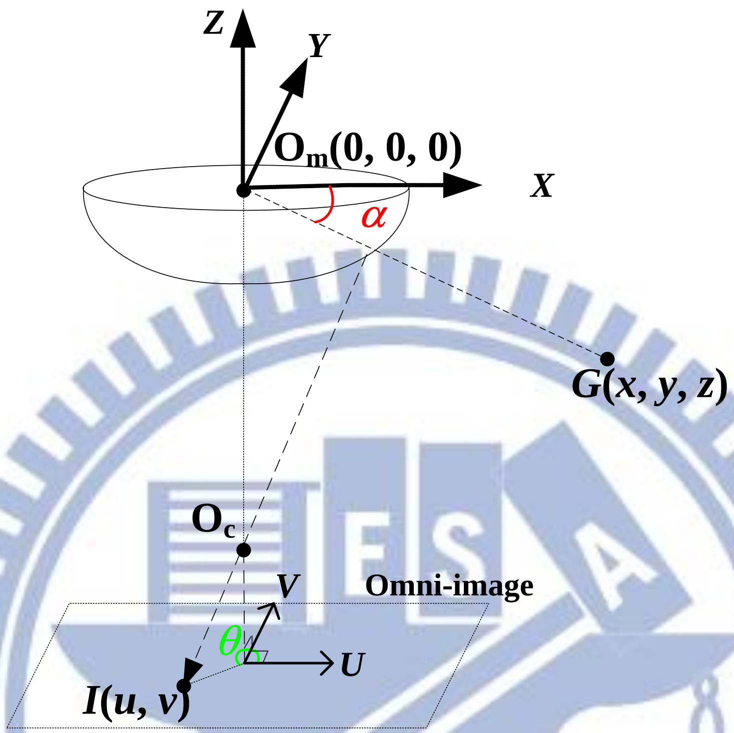


Figure 2.9 Imaging principle of a space point *G* using an omni-camera.

2.3.2 Derivation of formulas for 3D data acquisition

In this section, we will introduce the principle of the computation of the 3D range data. We will now define the direction of the camera coordinate system (CCS) $\mathrm{CCS}_{\mathrm{local}}$ in Figure 2.10. As seen in the figure, two light rays from *G* in $\mathrm{CCS}_{\mathrm{local}}$ go through the center of Mirror *S* and that of Mirror *B*, and α_1 and α_2 are the elevation angles, respectively. The points O_s, O_b, *G* form a triangle ΔO_sO_bG which is illustrated in Figure 2.11. We know the distance from O_s to O_b by manual measurement which is called the baseline in the last section. We can derive the following equations by the *law of sines* based on the geometry:

$$\frac{d}{\sin(90^o+\alpha_b)}=\frac{b}{\sin(\alpha_\alpha-\alpha_b)}. \qquad (2.1)$$

Then, we can calculate accordingly the baseline as follows:

$$d=\frac{b\times\sin(90^o+\alpha_b)}{\sin(\alpha_\alpha-\alpha_b)}=\frac{b\times\left(-\cos\left(\alpha_b\right)\right)}{\sin\left(\alpha_a-\alpha_b\right)}. \qquad (2.2)$$

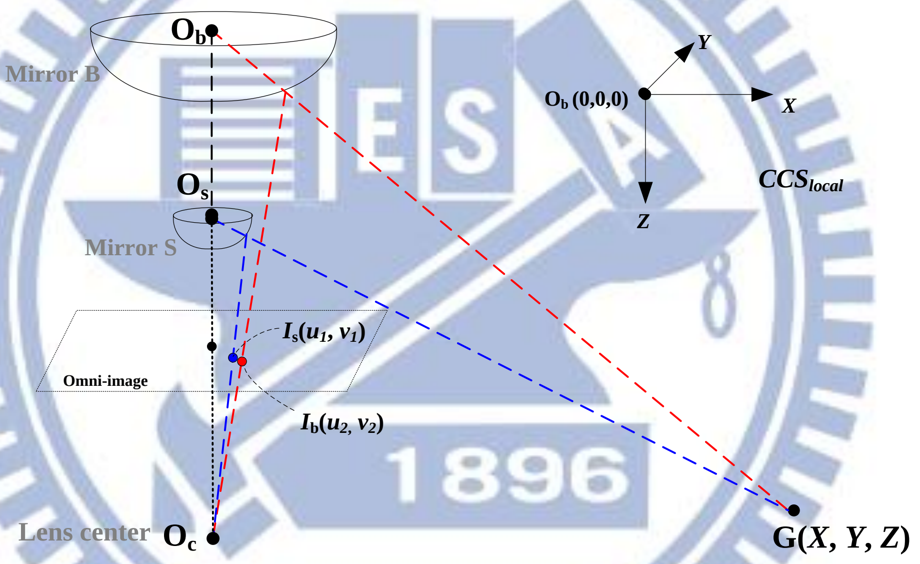


Figure 2.10 The cameras coordinate system CCS_{local}, and a space point G projected on the omni-image acquired by the two-mirror omni-camera.

Then, we want to know the azimuth angles of the two light rays. According to the property of rotational invariance of the omni-image, these two azimuth angles actually are equal, which we denote by θ. From Figure 2.12, the azimuth θ in the ICS can be computed by using the image coordinates (u_1, v_1) of G according to the following equations:

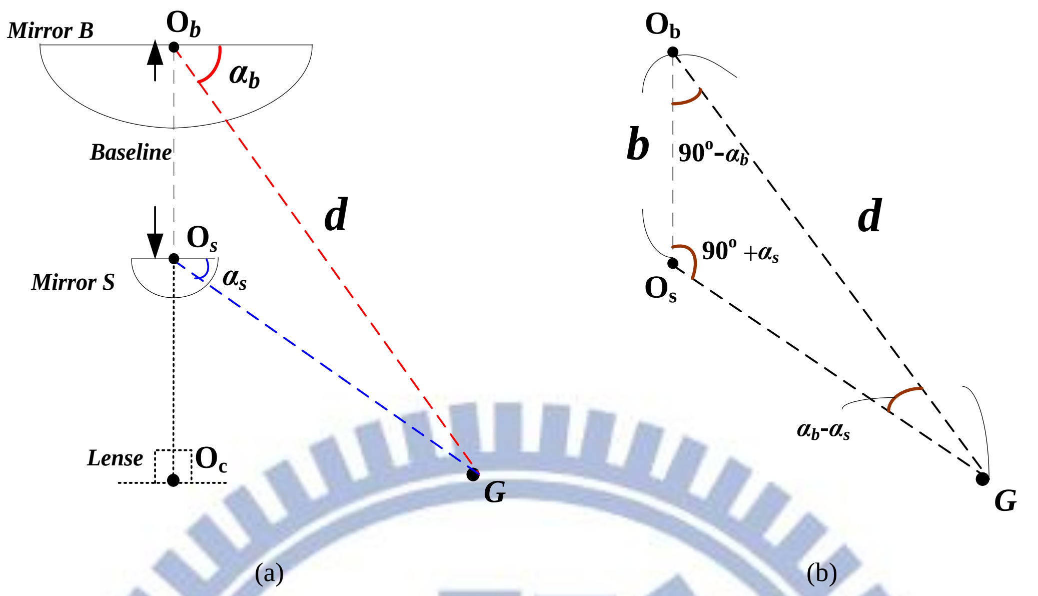


Figure 2.11 An illustration of the relation between a space point *G* and the two mirrors in the used camera. (a) A side view of *G* projected onto the two mirrors. (b) A triangle ΔO_bO_sG used in deriving 3D data.

$$\sin\theta = \frac{u_1}{\sqrt{u_1^{\,2}+v_1^{\,2}}}; \qquad \cos\theta = \frac{v_1}{\sqrt{u_1^{\,2}+v_1^{\,2}}};$$

$$\theta = \sin^{-1}\left(\frac{u_1}{\sqrt{u_1^{\,2}+v_1^{\,2}}}\right) = \cos^{-1}\left(\frac{v_1}{\sqrt{u_1^{\,2}+v_1^{\,2}}}\right). \tag{2.3}$$

After obtaining the distance *d* by Equation (2.2) and the azimuth angle θ by Equation (2.3), we can compute the position of *G*, namely, the global coordinates (*X*, *Y*, *Z*), in CCS_{local} by the following equations:

$$X = d \times \cos\alpha_a \times \sin\theta,$$

$$Y = d \times \cos\alpha_a \times \sin\theta,$$

$$Z = d \times \sin\alpha_a. \tag{2.4}$$

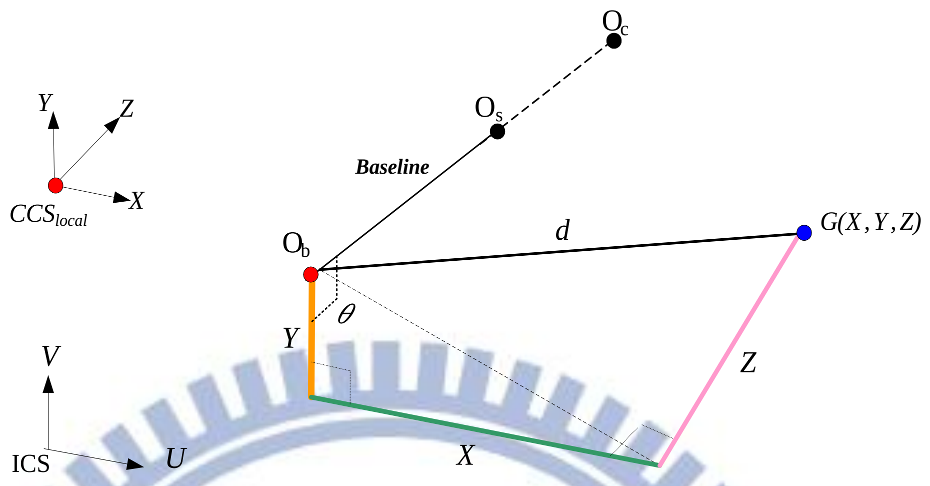


Figure 2.12 An illustration of a space point G at coordinates (X, Y, Z) in CCS_{local} .

As mentioned previously, the camera system we use in this study is not set parallel to the ground so as to enlarge the overlapping area of both mirrors; instead, it is slanted up for an angle of γ as shown 2.13. It is desired that the Z-axis of CCS_{local} could be parallel to the ground. We define another camera coordinate system CCS which coincides with CCS_{local} except the Z-axis is be elevated by an angle of γ. Finally, we can use a rotational matrix R to represent the G in the CCS by the following equations:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-\gamma) & -\sin(-\gamma) \\ 0 & \sin(-\gamma) & \cos(-\gamma) \end{bmatrix}; \tag{2.5}$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-\gamma) & -\sin(-\gamma) \\ 0 & \sin(-\gamma) & \cos(-\gamma) \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \tag{2.6}$$

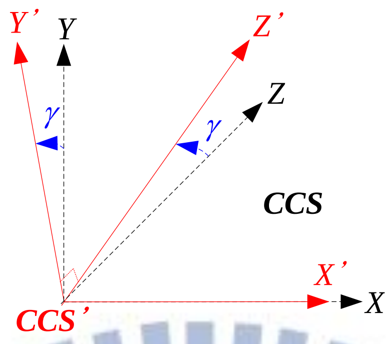


Figure 2.13 The relation between the two camera coordinate systems *CCS* and CCS_{local} .

2.4 System Operation Processes

2.4.1 Learning process

In the learning process, it is important for the autonomous vehicle to "learn" the selected path and conduct navigation automatically. In this section, we will describe the information which the vehicle should "memorize" in detail. Initially, the vehicle has to record where the selected path in the outdoor environment is. For this study, the experimental place is a sidewalk in the campus of National Chiao Tung University. Because the vehicle navigates on the sidewalk, we can take the advantage of the *sidewalk curb* to implement a "curb following" function for vehicle guidance. It also helps the vehicle to calibrate the odometer precisely. In addition, the lighting condition is a concern in the outdoor environment. So, the different location information must be recorded in the navigation path.

Moreover, another odometer calibration method adopted in this study is via landmark detection. By this method, the trainer can choose the landmarks which should be learned in the selected path, and decide where to localize the vehicle by the learning landmarks. Next, landmark detection is accomplished by a space line detection technique in this study, which is described in Chapter 5. After collecting enough information of the learned landmark, some parameters for landmark detection and the position of each landmark can be recorded.

To facilitate the user to learn navigation paths, a user learning interface is designed for the trainer. They may use it to control the autonomous vehicle to construct the navigation path. Furthermore, after the vehicle detects a landmark, we provide a *semi-automatic* learning process to adjust the parameters which we sat initially for the trainer to deal with some varying conditions of the environment. Also, the trainer should establish the navigation rules in advance for the vehicle to follow in the navigation process.

At last, after leading the autonomous vehicle to the destination, the learning process is finished. The learned information is organized into a *learned path* which is composed of several path nodes with guidance parameters. We finally acquire the navigation path map which combined the landmark information and the environment information, and store in the disk. The entire learning process proposed in this study is shown in Figure 2.14.

2.4.2 Navigation process

In the navigation process, the autonomous vehicle can analyze the current location using various stored information obtained in the learned process and navigate to the next node on the learned path. The entire navigation process proposed in this study is shown in Figure 2.15.

In general, the autonomous vehicle analyzes the current environment *node by node* to navigate to the goal according to the learned information data retrieved from the storage. Before the vehicle starts to navigate to the next node, the environment information should be checked by the autonomous vehicle. If the image is too dark or too light, the vehicle can be "confused" by the image we got from the camera system. According to the learning environment information, the system was designed to be able to adjust the exposure of the camera dynamically if necessary.

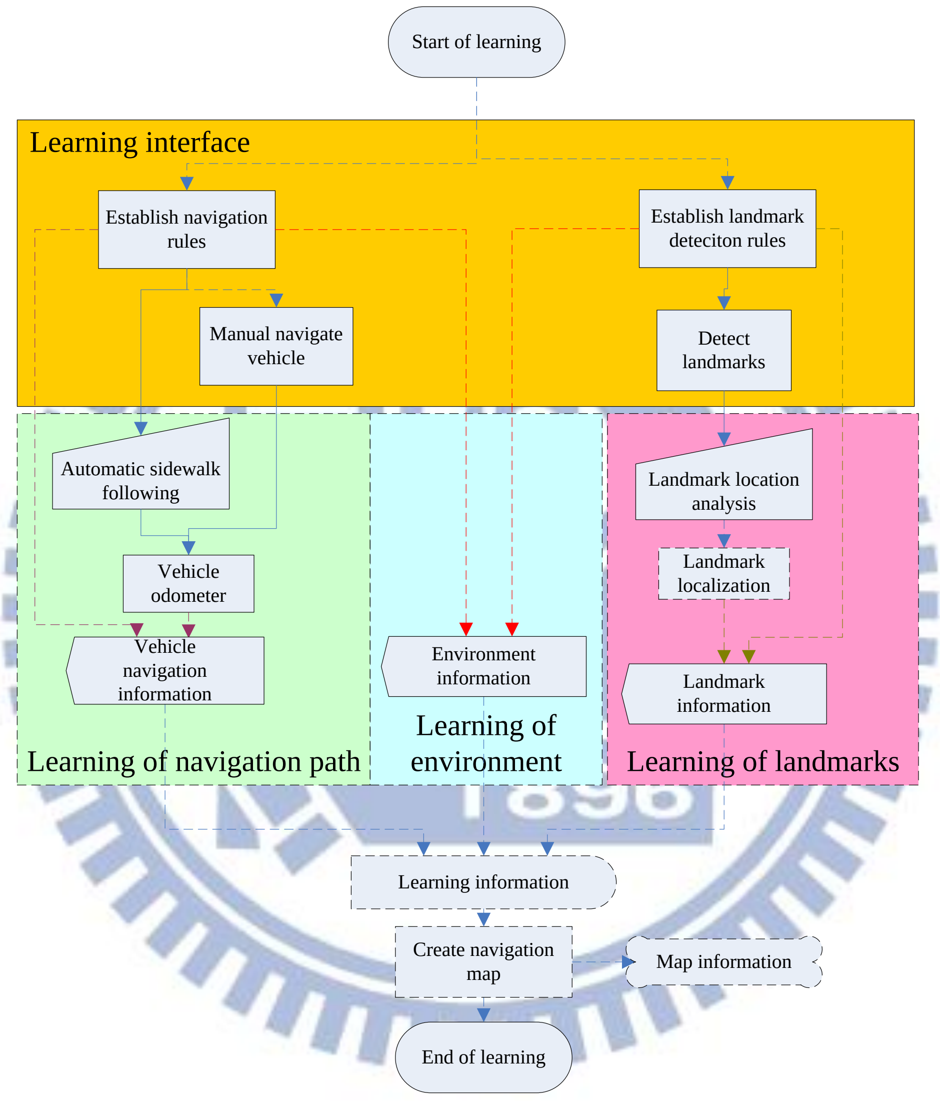


Figure 2.14 Learning process.

Besides, the autonomous vehicle always checks if any obstacle exists in front of the vehicle. As soon as an obstacle is found and checked to be too close to the vehicle, a procedure of collision avoidance is started automatically to perform collision avoidance. In addition, if the vehicle gets a node of "landmark detection," the autonomous vehicle will adjust the detection pose and load the parameters for landmark detection. If a landmark is found successfully, the landmark's position is used to modify the odometer of the vehicle; if not, some strategy of recovering the landmark are started, such as changing the parameters for landmark detection or changing the pose of the vehicle to detect landmark successfully.

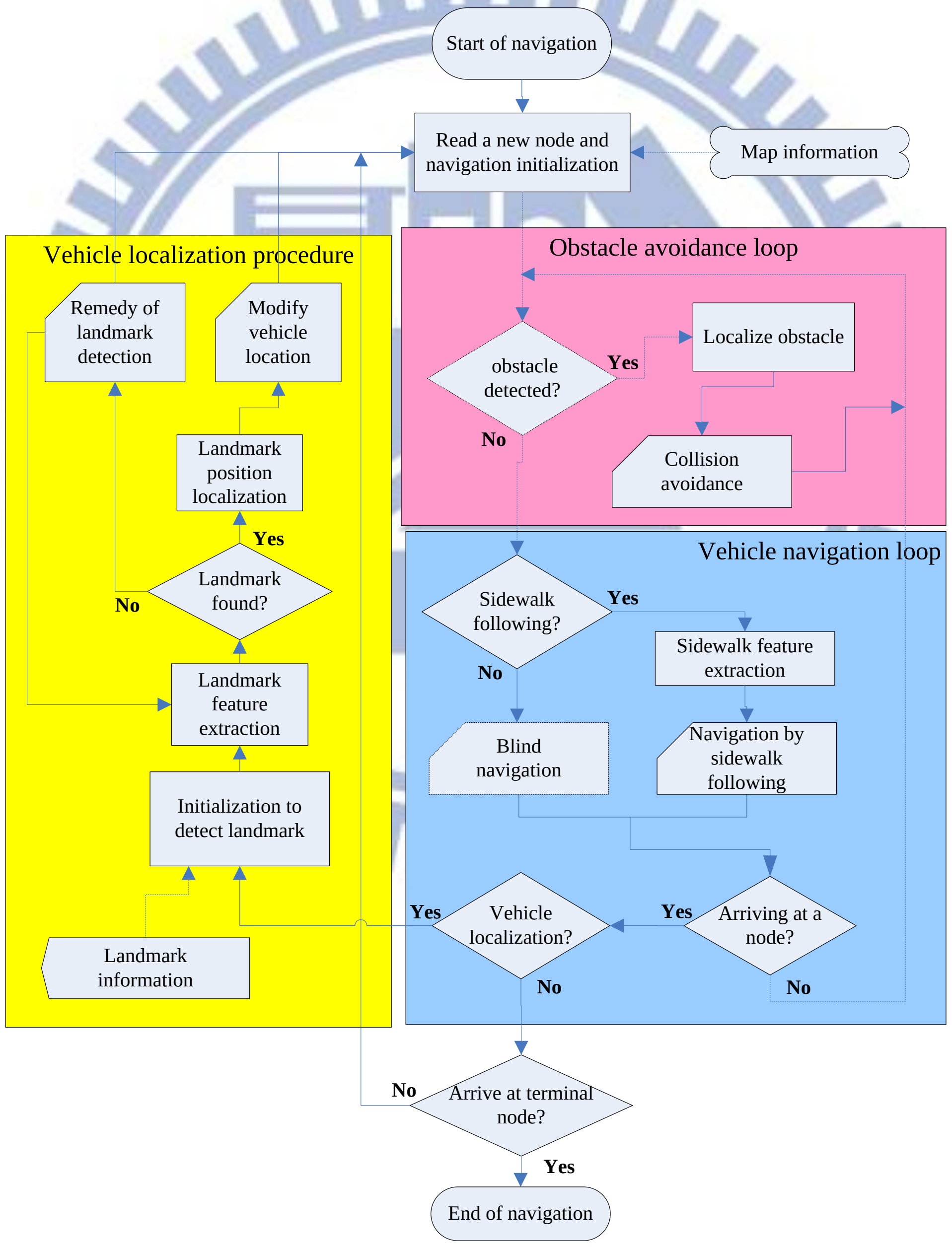


Figure 2.15 Navigation process.

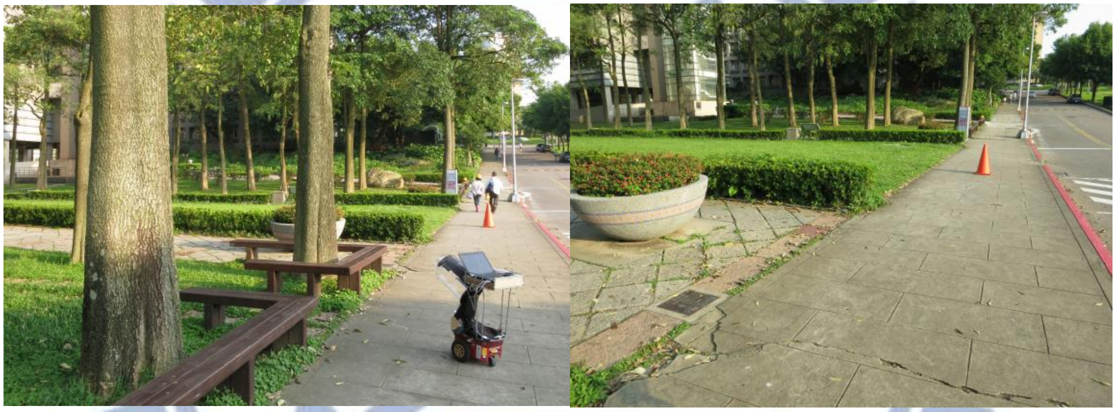
Chapter 3
Learning Strategy for Automatic Navigation

3.1 Introduction

The purpose of the learning process for the proposed machine guide dog system is to create a path on a sidewalk to guide a blind person to a selected destination. Before starting to learn a path, we have to do some works. First, at first we have to choose some landmarks for vehicle localization. Then, we have to calibrate the camera system. The third task is to infer some guidance parameters. At last, we should adopt a learning strategy to learn certain information about each selected landmark.

3.1.1 Selected landmarks in outdoor environments for this study

When the vehicle is in the navigation process, mechanic errors usually will accumulate up to cause imprecise odometer readings of the vehicle location and orientation. To solve such problems, in this study we adopt the approach of *vehicle localization using landmarks*. For this purpose, some objects should be selected as landmarks at first to conduct the localization work. Chou and Tsai [23] detected the light pole and the hydrant to localize the vehicle. In this study, we select instead some other natural and artificial objects as landmarks, which are commonly seen on sidewalks. Specifically, we select two types of natural landmarks, *tree trunk* and *lawn corner*, for vehicle localization in this study, as shown in Figure 3.1. Also selected for the same purpose are three types of artificial landmarks, namely, *signboard*, *traffic cone*, and *stop line on roads*, as shown in Figure 3.2. With more types of landmarks so selected, we can have more information along the way for localization, and we can then guide the autonomous vehicle to the destination more reliably. The detailed proposed methods for vehicle localization using landmarks will be introduced later in Chapters 5 and 6. In this chapter, we discuss the learning process for these landmarks and other information.



(a) (b)

Figure 3.1 Two types of natural landmarks selected for use in this study. (a) Tree landmark. (b) Lawn corner point.

3.1.2 Camera calibration

As mentioned in Chapter 1, it is a complicated task to calibrate a camera's intrinsic and extrinsic parameters. A space-mapping technique [24], called pano-mapping, is adopted instead in this study to "calibrate" the two-mirror omni-camera system used in this study. We will introduce the adopted technique in Section 3.2.

3.1.3 Learning of guidance parameters

To navigate in outdoor environments, a trainer of the proposed vehicle system should guide the system to learn and record some parameters of the sidewalk environment for use in the navigation process. The parameters to be learned in this study include *image segmentation thresholds* and some other *environment parameters*. The proposed technique for learning the adopted environment parameters will be described in Section 3.4.1. Also, the proposed method for learning the used image segmentation thresholds will be described in Section 3.4.2. Finally, a process which we propose to create the navigation path is described in Section 3.5.3.

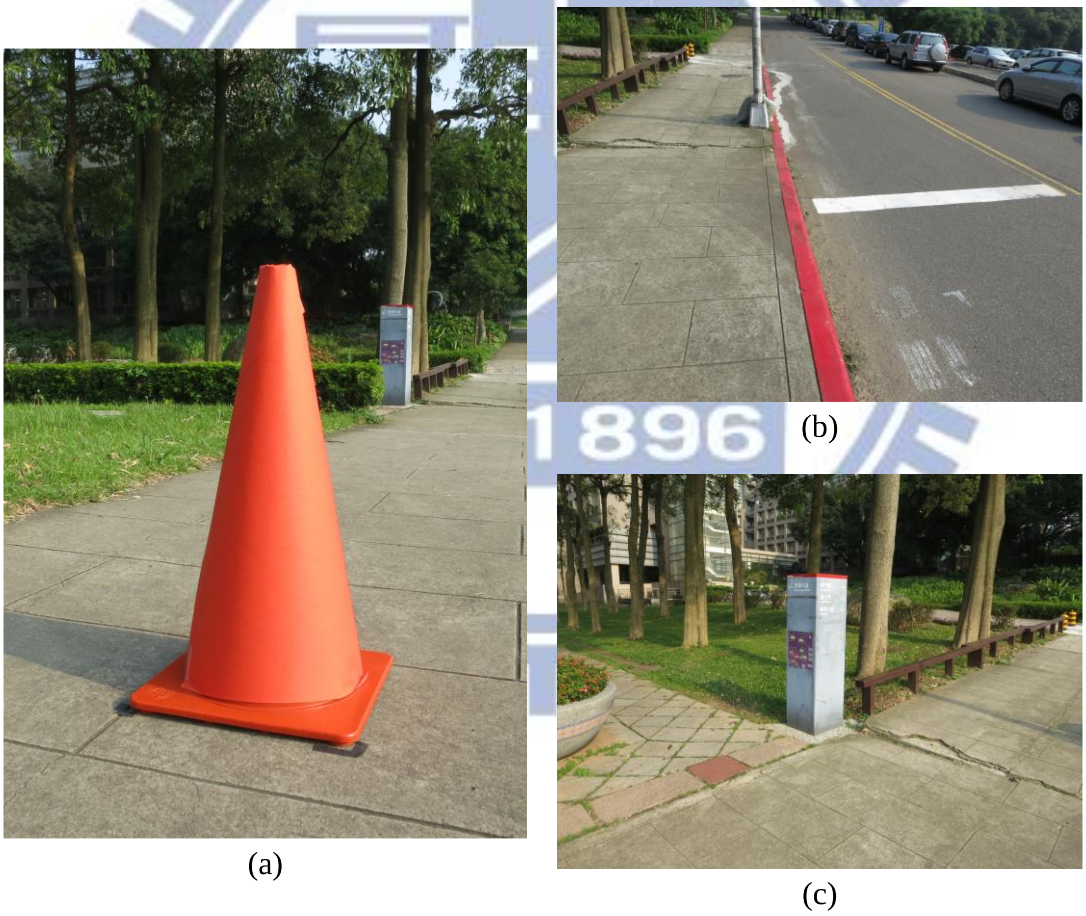

(a)
(b)
(c)

Figure 3.2 Three artificial landmarks used in this study. (a) Tree landmark. (b) Road stop line landmark. (c) Signboard landmark.

3.2 Camera Calibration by Space-mapping Approach

To calibrate the camera system used in this study, we use the pano-mapping technique proposed by Jeng and Tsai [24]. Specifically, it is desired to establish a *pano-mapping table* to record the relations between the locations of image points and those of the corresponding real-world points. For this, as mentioned in Chapter 2, we assume first that a light ray going through a world-space point P with the elevation angle α and the azimuth angle θ is projected onto a specific point p at coordinates (u, v) in the omni-image. The pano-mapping table specifies the relation between the coordinates (u, v) of the pixel p in the image and the azimuth-elevation angle pair (θ, α) of the corresponding world-space point P. We construct the pano-mapping table once, and the table can be looked up to retrieve 3D information forever. Specifically, we establish two pano-mapping tables for *Mirrors S* and *B*, respectively, in the camera system used in this study. The details are described in the following algorithm.

Algorithm 3.1 Construction of pano-mapping tables.

Input: two sets of six landmark point pairs (p_i, P_i) and (q_j, Q_j) selected in advance manually where p_i and q_i are points in an omni-image I and P_i and Q_j are the corresponding points in the world space.

Output: two pano-mapping tables of dimension $M\times N$ for *Mirrors B* and *S*.

Steps.

Step. 1. Let the six known image pixels p_i be located at coordinates $(u_i, v_i,)$ in the *Mirror B* region in omni-image I and the six corresponding known world-space points P_i be at coordinates (x_i, y_i, z_i) in the camera coordinate system, where $i = 1, 2, \ldots, 6$, as shown in Figure 3.3.

Step. 2. Similarly let the six known image pixels q_j be located at coordinates (U_j, V_j,) in the *Mirror S* region in omni-image *I* and the six corresponding known world-space points Q_j be at coordinates (X_j, Y_j, Z_j) in the camera coordinate system, where $j = 1, 2, \ldots, 6$.

Step. 3. Calculate the radial distances r_i and R_j in the image plane from the image pixels p_i and q_j to the image center, respectively, by the following equations:

$$r_i = \sqrt{u_i^{\,2} + u_i^{\,2}};\quad R_i = \sqrt{U_i^{\,2} + V_i^{\,2}}. \tag{3.1}$$

resulting in six pairs of radial distances r_i and R_i for *Mirrors S* and *B*, respectively.

Step. 4. Calculate the elevation angles α_i and β_i for the world-space points P_j and Q_j, respectively, by the following equations:

$$\alpha_i = \tan^{-1}(z_i / \sqrt{x_i^{\,2} + y_i^{\,2}});\quad \beta_i = \tan^{-1}(Z_i / \sqrt{X_i^{\,2} + Y_i^{\,2}}), \tag{3.2}$$

resulting in six pairs of elevation angles for *Mirrors S* and *B*, respectively.

Step. 5. Under the assumption that the surface geometries of *Mirrors S* and *B* are radially symmetric in the range of 360 degrees, use two *radial stretching functions*, denoted as f_S and f_B, to describe the relationship between the radial distances r_i and the elevation angles α_i as well as that between R_j and β_j, respectively, by the following equations where $i = 1, 2, \ldots, 6$:

$$r_i = f_S(\alpha_i) = s_0 + s_1 \times \alpha_i^{\,1} + s_2 \times \alpha_i^{\,2} + s_3 \times \alpha_i^{\,3} + s_4 \times \alpha_i^{\,4} + s_5 \times \alpha_i^{\,5};$$

$$R_i = f_B(\beta_i) = b_0 + b_1 \times \beta_i^{\,1} + b_2 \times \beta_i^{\,2} + b_3 \times \beta_i^{\,3} + b_4 \times \beta_i^{\,4} + b_5 \times \beta_i^{\,5}. \tag{3.3}$$

Step. 6. Solve the above 6-th degree polynomial equations for f_A and f_B by the uses of the six radial-distance pairs for *Mirrors S* and *B*, respectively, obtained in

Step 4 as well as a numerical method to obtain the coefficients a_0 through a_5 and b_0 through b_5.

Step. 7. By the use of the function f_B with the known coefficients b_0 through b_5, construct a pano-mapping table T_B for *Mirror B* in a form as that shown in Table 3.1(a) according to the following rule:

for each world-space point P_{ij} with the azimuth-elevation pair (θ_i, β_j), compute the corresponding image coordinates (u_{ij}, v_{ij}) by the following equations:

$$u_{ij} = r_j \times \cos\theta_i; \quad v_{ij} = r_j \times \sin\theta_i . \tag{3.4}$$

Step. 8. In a similar way, construct a pano-mapping table T_S for *Mirror S* by the use of the function f_S with the known coefficients s_0 through s_5 in a form as that shown in Table 3.1(b).

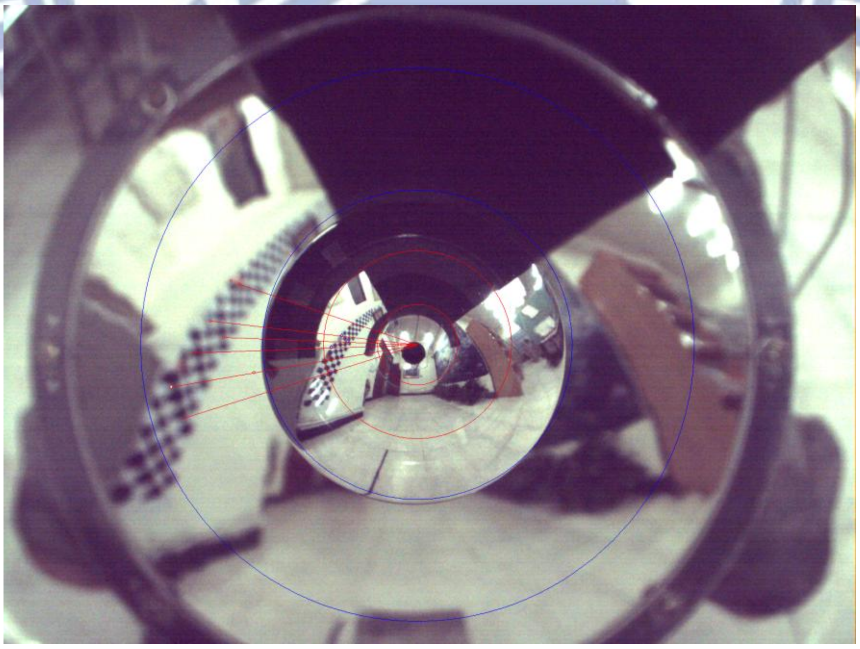


Figure 3.3 Illustration of constructing pano-mapping tables in this study.

Table 3.1 Two pano-mapping tables used for the two-mirror omni-camera used in this study. (a) Pano-mapping table used for *Mirror B*. (b) Pano-mapping table used for *Mirror S*.

	θ'_1	θ'_2	θ'_3	θ'_4	…	θ'_S
β_1	(u'_{11}, v'_{11})	(u'_{21}, v'_{21})	(u'_{31}, v'_{31})	(u'_{41}, v'_{41})	…	(u'_{S1}, v'_{S1})
β_2	(u'_{12}, v'_{12})	(u'_{22}, v'_{22})	(u'_{32}, v'_{32})	(u'_{42}, v'_{42})	…	(u'_{S2}, v'_{S2})
β_3	(u'_{13}, v'_{13})	(u'_{23}, v'_{23})	(u'_{33}, v'_{33})	(u'_{43}, v'_{43})	…	(u'_{S3}, v'_{S3})
β_4	(u'_{14}, v'_{14})	(u'_{24}, v'_{24})	(u'_{34}, v'_{34})	(u'_{44}, v'_{44})	…	(u'_{S4}, v'_{S4})
...	…	…	…	…	…	…
β_T	(u'_{1T}, v'_{1T})	(u'_{2T}, v'_{2T})	(u'_{3T}, v'_{3T})	(u'_{4T}, v'_{4T})	…	(u'_{ST}, v'_{ST})

(a)

	θ_1	θ_2	θ_3	θ_4	…	θ_M
α_1	(u_{11}, v_{11})	(u_{21}, v_{21})	(u_{31}, v_{31})	(u_{41}, v_{41})	…	(u_{M1}, v_{M1})
α_2	(u_{12}, v_{12})	(u_{22}, v_{22})	(u_{32}, v_{32})	(u_{42}, v_{42})	…	(u_{M2}, v_{M2})
α_3	(u_{13}, v_{13})	(u_{23}, v_{23})	(u_{33}, v_{33})	(u_{43}, v_{43})	…	(u_{M3}, v_{M3})
α_4	(u_{14}, v_{14})	(u_{24}, v_{24})	(u_{34}, v_{34})	(u_{44}, v_{44})	…	(u_{M4}, v_{M4})
...	…	…	…	…	…	…
α_N	(u_{1N}, v_{1N})	(u_{2N}, v_{2N})	(u_{3N}, v_{3N})	(u_{4N}, v_{4N})	…	(u_{MN}, v_{MN})

(b)

3.3 Coordinate Systems

In this study, the following four coordinate systems are used to describe the vehicle location and the navigation environment. The coordinate systems are illustrated in Figure 3.4 and defined in the following.

(1). Image coordinate system (ICS, denoted as *u*-*v*): The origin O_I of the image coordinate system is located at the center of the image plane, and the *u*-*v* plane coincides with the image plane.

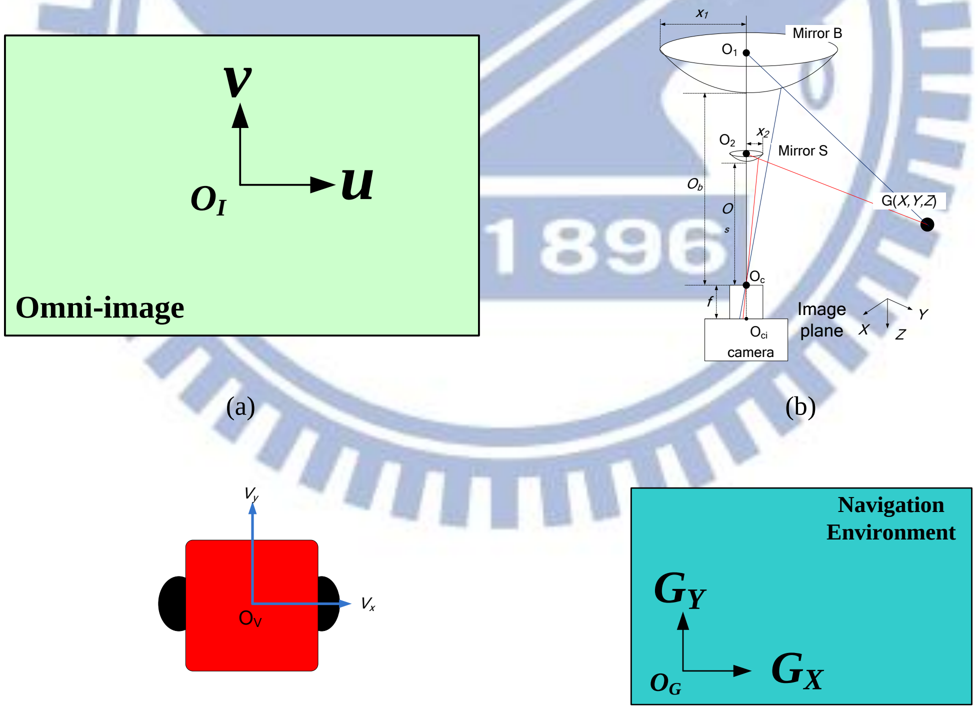
(2). Camera coordinate system (CCS, denoted as *X*-*Y*-*Z*): The origin O_C of the CCS is located at the focal point of *Mirror B*. The *X*-*Z* plane is parallel to the ground and the *Y*-axis is perpendicular to the ground.

(3). Vehicle coordinate system (VCS, denoted as V_X-V_Y): The origin O_V of the vehicle coordinate system is located at the center of the autonomous vehicle, and the V_X-V_Y plane coincides with the image plane.

(4). Global coordinate system (GCS, denoted as G_X-G_Y): The origin O_G of this system is always set at the start position of the vehicle in the navigation path, and the G_X-G_Y plane coincides with the ground.

When the vehicle is moving in the navigation session, we have to know the relationships among the coordinate systems. It is advantageous to utilize an odometer to localize the vehicle position in the GCS, though the odometer readings are not very accurate all the time. At the beginning of each navigation process, the VCS and CCS follow the vehicle, and the VCS coincides with the GCS. After the vehicle moves for a short distance, as illustrated in Figure 3.5, and stops at a position V at world coordinates (C_x, C_y) with a rotation angle θ, we can derive the coordinate transformation between the coordinates (V_X, V_Y) of the VCS and the coordinates (G_X, G_Y) of the GCS by the following equations:

$$\begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} + \begin{bmatrix} C_x \\ C_y \end{bmatrix}. \tag{3.5}$$



(a) (b)

(c) (d)

Figure 3.4 Four coordinate systems used in this study. (a) The ICS. (b) The CCS. (c) The VCS. (d) The GCS.

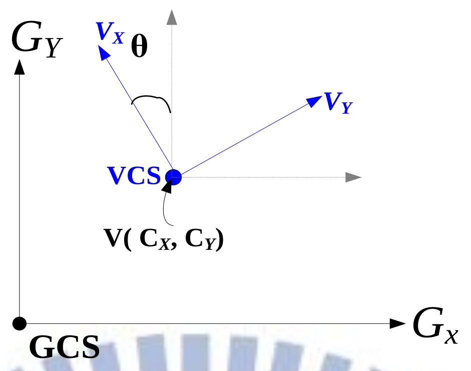


Figure 3.5 A vehicle at coordinates (C_x, C_y) with a rotation angle θ with respect to the GCS.

Besides, the relationship between the CCS and the VCS is illustrated in Figure 3.6. Because the origin of the CCS which projects onto the ground does not coincide with the origin of the VCS, we have to provide the transformation function. As illustrated in Figure 3.7, there is a distance between the two origins, which we denote as S_y, on the V_y-axis. Thus, the transformation function between the CCS and the VCS can be derived by the following equations:

$$V_X = X\ ;\quad V_Y = Z + S_y\,. \tag{3.6}$$

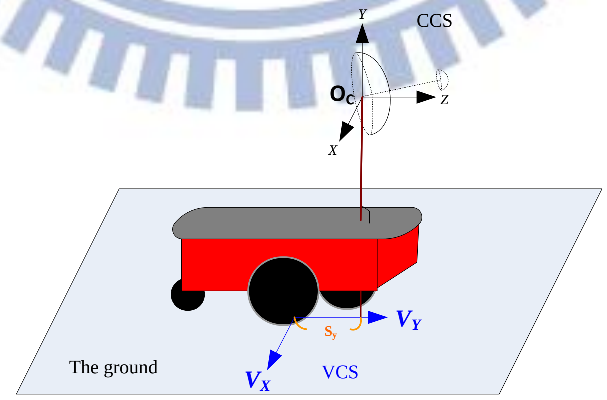


Figure 3.6 An illustration of the relation between the CCS and the VCS.

3.4 Learning of environment and landmark parameters

3.4.1 Learning of environment intensity in windows

In the navigation process, we navigate the vehicle along the learned path. Because the aim is to navigate along the path again and again in this study, each landmark is usually projected onto a fixed region in the image. By this property, we can define regions of interest (ROIs) in the image as shown in Figure 3.7, which are also called *environment windows*. Some advantages can be obtained from this approach as follows:

1. we can reduce the computation time for detecting the desired landmark;
2. if a feature similar to the detected landmark appears in the environment, it is easy to distinguish the object from the noise.

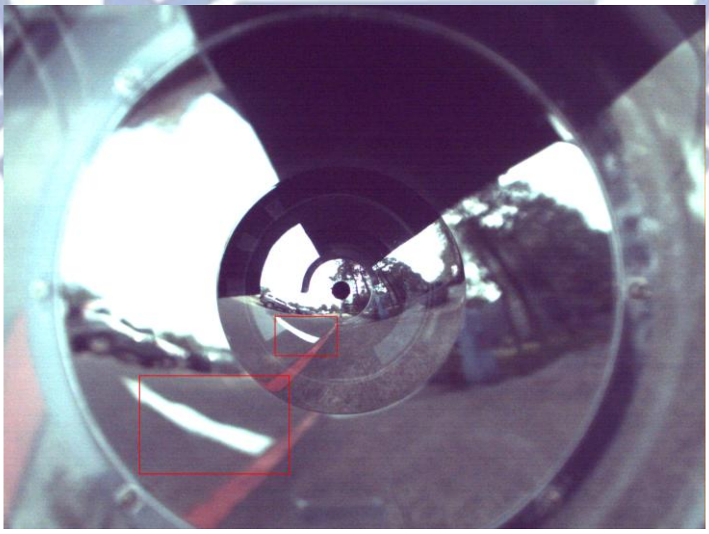


Figure 3.7 A pair of environment windows for road stop line detection.

However, this property alone does not solve the problem totally in outdoor environments. According to our experimental experience, varying lighting conditions influence the results of the environment analysis work as well. For instance, as shown in Figure 3.9(a), because of the overexposure duo to the lighting condition, the feature of the curb line is not obvious enough to be recognized. For this reason, we provide a system which is based on Chou and Tsai [23] for the trainer to adjust the exposure of the camera for the purpose of detecting the landmark successfully. When the system adjusts the exposure of the camera to a suitable value, it means that the landmark we want to detect can be extracted well in this condition. Then, we may record the image illumination parameter into the path information as part of the learned parameters. To be more specific, we learn a suitable image intensity, called *environment intensity* hereafter, on the image in the environment windows during the learning process. A detailed algorithm for the above process is described in the following.

Algorithm 3.2 ***Learning of the environment intensity parameter at a path node.***

Input: a relevant set of environment windows Win_{set} for a certain path node with a pre-selected landmark under the assumption that the vehicle arrives at the node currently.

Output: an environment intensity parameter I_{en}.

Steps.

Step 1. Adjust the camera exposure and acquire a suitable image I_{all}.

Step 2. Check if the desired landmark feature is well imaged in the current illumination: if not, go to Step 1; otherwise, continue.

Step 3. For each pixel in image I_{all} with color (R, G, B) in win_B of Win_{set}, calculate its intensity value Y_i by the following equation and record Y_i into a set S_Y:

$$Y_i = 0.299 \times R + 0.587 \times G + 0.114 \times B. \tag{3.7}$$

Step 4. Calculate an average value I_{en} of all Y_i in the following way as output by the use of the data in S_Y where N is the size of win_B of Win_{en}:

$$I_{en} = \frac{1}{N}\sum_{i=1}^{N} Y_i \,. \tag{3.8}$$

Some examples of suitable illuminations for navigation tasks are shown in Figures 3.8(b), and 3.9(b). The environment intensity parameters learned in the above way for them will be recorded as part of the learning result of landmark detection described later.

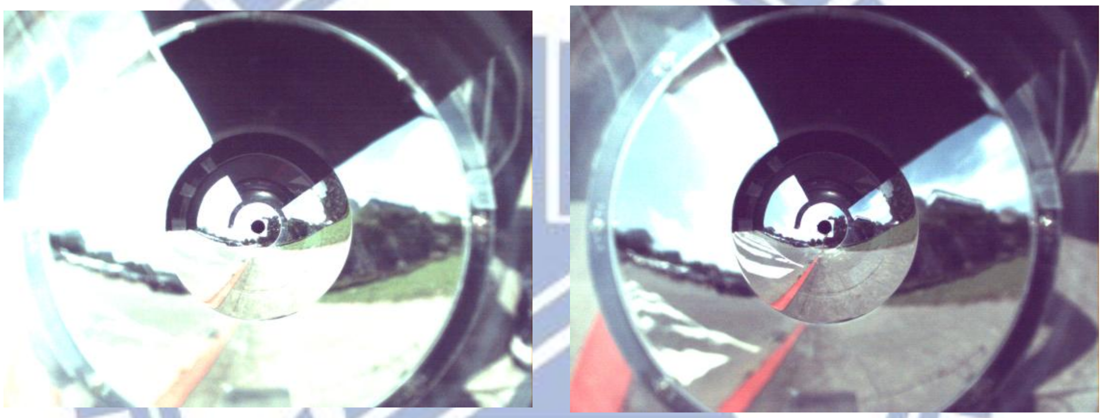


Figure 3.8 Two different illuminations for curb line detection. (a) An instance of overexposure. (b) A suitable case.

3.4.2 Learning of artificial landmark segmentation parameters

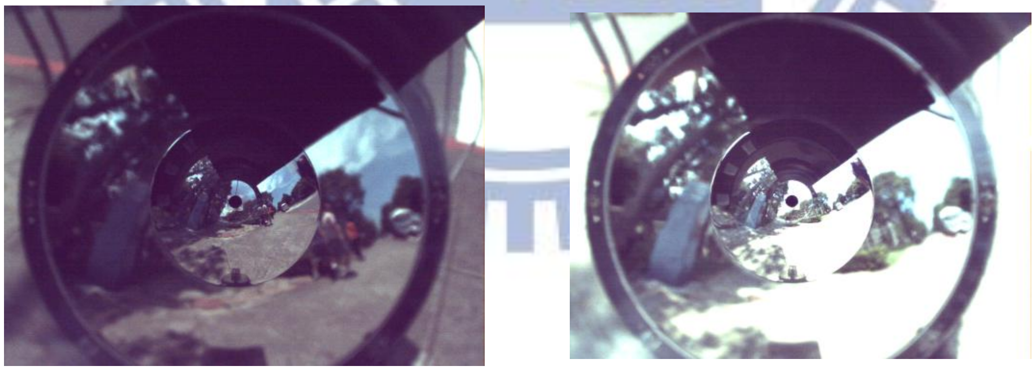
It is very important for us to localize landmarks in this study. Before landmark localization, we have to utilize some segmentation methods for image analysis. In this section, we introduce the segmentation parameters proposed for use in this study for artificial landmark segmentation. Regarding natural landmark detection, we utilize the moment-preserving thresholding proposed by Tsai [26] to conduct landmark segmentation. The moment-preserving thresholding will be introduced in Chapter 5.

Also, the used landmark detection methods will be described in Chapters 5 and 6.

(1) *For sidewalk curb segmentation* — we use the color information (hue and saturation) and the image thresholding technique to find the curb feature in the image utilizing the HSI color model first. Then we adopt the Canny edge detection technique to extract the desired landmark shape. The thresholds for hue and saturation values are collected as a set of *curb segmentation parameters*. Also, for the road stop line and the traffic cone, we conduct similar works.

(2) *For signboard segmentation* — we use the HSI color model to extract the signboard shape. The threshold values for hue and saturation and also the contour of the signboard described by the principal components obtained from principal component analysis are collected as a set of *signboard segmentation parameters*.

(3) *For tree segmentation* — we use the moment-preserving thresholding as mentioned previously to extract the tree shape. The contour of the tree also described by principal components obtained from principal component analysis is collected as a set of *tree segmentation parameters.*



(a) (b)

Figure 3.9 Two different illuminations for signboard detection. (a) An instance of underexposure. (b) A suitable case.

When conducting landmark learning, the trainer can detect a desired landmark by a user interface of the system, and adjust the values of the related set of segmentation parameters. After obtaining an appropriate result from the landmark detection process, the segmentation parameters and the learned landmark information which the trainer will use are recorded together as part of the learned path. The process for learning landmark segmentation parameters is shown in Figure 3.10.

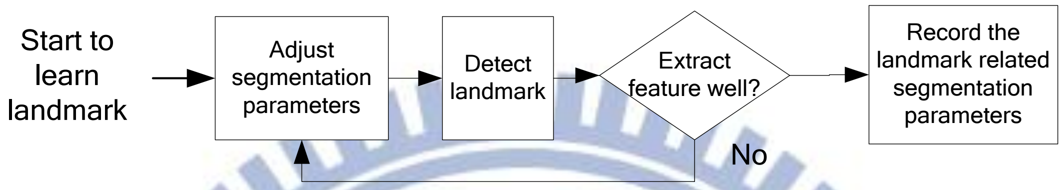

Start to learn landmark
Adjust segmentation parameters
Detect landmark
Extract feature well?
No
Record the landmark related segmentation parameters

Figure 3.10 The process for learning landmark segmentation parameters.

3.5 Learning Processes for Creating a Navigation Path

In this section, we introduce the proposed method for learning a navigation path in the learning process. The method proposed is based on Chou and Tsai [23]. In the learning process, we use the odometer to localize the vehicle position and approximate the detected landmark position in general. The proposed strategy for learning landmarks for vehicle localization is described in Section 3.5.1. Additionally, there are some obstacles on the sidewalk along the way. The obstacles may block the vehicle. As shown in Figure 3.11, there is a hole on the sidewalk. It may cause the vehicle to fall outside the sidewalk. Thus, we propose a method to learn the positions of such obstacles along the way, called *fixed obstacles* hereafter. The method is described in Section 3.5.2. Finally, we introduce the entire proposed procedure to learn a navigation path in Section 3.5.3.

3.5.1 Strategy for learning landmark positions and related vehicle poses

We introduce the proposed strategy for learning a landmark and its position in this section. Simply speaking, for a landmark to be learned well, we have to guide the vehicle to appropriate positions to detect it. To increase the accuracy of the position of the learned landmark, we take images of the landmark a number of times from a number of different positions or different directions. The reason why we take multiple images is that the outdoor condition might cause the taken images to be all different, especially when there are clouds floating across the sun in the sky during the noon time. After we collect multiple images and analyze the feature data, a more precise landmark position with the corresponding vehicle pose can be obtained. Then, it is recorded as part of the learned navigation path.

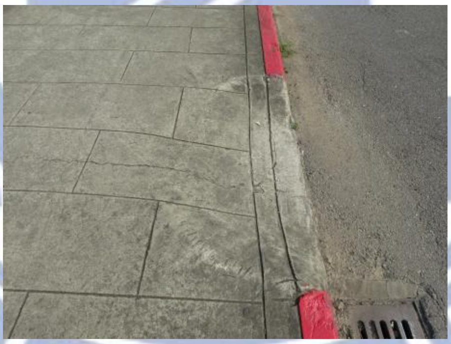


Figure 3.11 A fixed obstacle in a navigation path which may cause the vehicle to fall outside the sidewalk.

To be more specific, after we detect the landmark in omni-images a multiple times with the vehicle in different poses, we can calculate the mean of all the detected landmark positions as an *estimated* landmark position, denoted as $P_{landmark}$. Furthermore, we choose the vehicle pose among the multiple ones, which is closest to the one to yield the estimated $P_{landmark}$, for use as the *learned pose*, denoted as $P_{vehicle}$, corresponding to the estimated $P_{landmark}$. The detailed algorithm for the above process is described in the following.

Algorithm 3.2 ***Learning of the landmark position and related vehicle pose.***

Input: a landmark type of the appointed landmark to be learned.

Output: an estimated landmark position $P_{landmark}$ and a corresponding vehicle pose $P_{vehicle}$.

Steps.

Step 1. Initialize three parameters i, j and k to be zeros, where i, j and k represent the k-th landmark detection, the j-th vehicle orientation, and the i-th vehicle position, respectively.

Step 2. Guide the vehicle to a position $V_i = (Px_i, Py_i)$ and record this vehicle position V_i into a set S_V.

Step 3. Turn the vehicle into an orientation Th_{ij} and record this orientation into a set S_{Th}.

Step 4. According to the type of landmark, localize the landmark by the use of the corresponding localization technique (described in Chapter 5) to obtain the landmark position $p_{ijk} = (x_{ijk}, y_{ijk})$, and record this landmark position p_{ijk} into a set S_L.

Step 5. Go to Step 4 for K times as needed, and record the number of recoded landmark positions in the j-th vehicle orientation and the i-th vehicle position, denoted as $N_{ij} = K$.

Step 6. Go to Step 3 for J times as needed, and record the number of different vehicle orientations in the i-th vehicle position, denoted as $N_i = J$.

Step 7. Go to Step 2 for I times as needed, and record the number of the different vehicle position, denoted as $N = I$.

Step 8. Compute the desired landmark position $P_{landmark}$ using the set S_L by the following equation:

$$p_{landmark} = \frac{1}{\sum_{i=1}^{N}\sum_{j=1}^{N_i} N_{ij}} \sum_{i=1}^{N}\sum_{j=1}^{N_i}\sum_{k=1}^{N_{ij}} p_{ijk} = \frac{1}{\sum_{i=1}^{N}\sum_{j=1}^{N_i} N_{ij}} \sum_{i=1}^{N}\sum_{j=1}^{N_i}\sum_{k=1}^{N_{ij}} (x_{ijk}, y_{ijk}). \qquad (3.9)$$

Step 9. In S_V, select the c-th vehicle pose $v_c = (Px_c, Py_c)$, where v_c has the minimum distance to P_{landmark} computed in terms of the Euclidean distance.

Step 10. Choose a median orientation Th_c from all Th_{ca} in S_{Th}, where a is 1 through N_c, and set the desired vehicle position $P_{vehicle}$ as $P_{vehicle}= (Px_c, Py_c, Th_c)$.

3.5.2 Learning of fixed obstacles in a navigation path

In this section, we propose a function for use on the learning interface which can be used to learn fixed obstacles. It is based on Chou and Tsai [23]. When we guide the vehicle to a location where a fixed obstacle is projected onto the image region of both *Mirrors S* and *B*, we utilize this function to learn the fixed obstacle. We know that the fixed obstacle is located on the sidewalk, so we can use this property to learn the fixed obstacles more easily. As shown in Figure 3.12, first we use the mouse to click the position of the fixed obstacle on the region of *Mirr*or *B* in the image. Then, the system will record the pixel in the image for use later to calculate the learned position of the fixed obstacle. After selecting sufficient obstacle points in the omni image, the position of the fixed obstacles W_{obs} and some parameters for avoiding the obstacles are recorded together as part of the learned path information. Finally, the trainer may set the distance parameters to let the vehicle cross the fixed obstacle safely. The detailed algorithm to implement the above-mentioned ideas is described in the following.

Algorithm 3.3 ***Computation of fixed obstacle positions.***

Input: an image I_{input}, the distance from the center of *Mirror B* to the ground, and a set S_{obs} of the image points in the region of *Mirror B*, denoted as $b_i = (u_{1i}, v_{1i})$, where $i = 1$ through N.

Output: a fixed obstacle position W_{obs}.

Steps.

Step 1. Manually choose the fixed obstacle point b_i at coordinates (u_{1i}, v_{1i}) in the region of *Mirror B* in I_{input} and record b_i.

Step 2. Repeat Steps 1 for N times.

Step 3. For a set of points b_i, calculate the 3D position (c_{xi}, c_{yi}, c_{zi}) of point C_i in the CCS by the derivations mentioned in Section 2.3.2 using the two pano-mapping tables.

Step 4. Use the camera coordinates (c_{xi}, c_{yi}, c_{zi}) of point C_i and the coordinate transformation from the CCS to the WCS described by Equations (3.4) and (3.5) to calculate the position (x_i, y_i) of the corresponding point W_i on the ground in the WCS, and record W_i into a set W_{obs}.

Step 5. Repeat Steps 3 and 4 for N times.

Step 6. Derive the position (obs_x, obs_y) of point W_{obs} in the WCS as the location of the obstacle by the following equations:

$$obs_x = \frac{1}{N}\sum_{i=1}^{N} x_i\ ; \quad obs_y = \frac{1}{N}\sum_{i=1}^{N} y_i\ . \tag{3.10}$$

3.5.3 Algorithm for path learning

In this section, we introduce the method we propose to establish a navigation path in the learning process. We define eleven types of navigation nodes in this study, as listed in Table 3.2. These navigation nodes include a set of different works which have to be conducted by the vehicle. We guide the vehicle to learn a navigation path which we select, and some pre-selected landmarks by the utilization of the navigation node are recorded. Also, other parameters, like the environment intensity and landmark segmentation parameter, are recorded in the learned path. When the learning process is terminated, construction of the navigation path is finished. A navigation path consists of a number of the navigation nodes and some relevant parameters. When the navigation process is started, the navigation path can be used for vehicle navigation successfully. The detailed algorithm to implement the learning system is described in the following, and a flowchart of the process for navigation path creation is illustrated in Figure 3.13

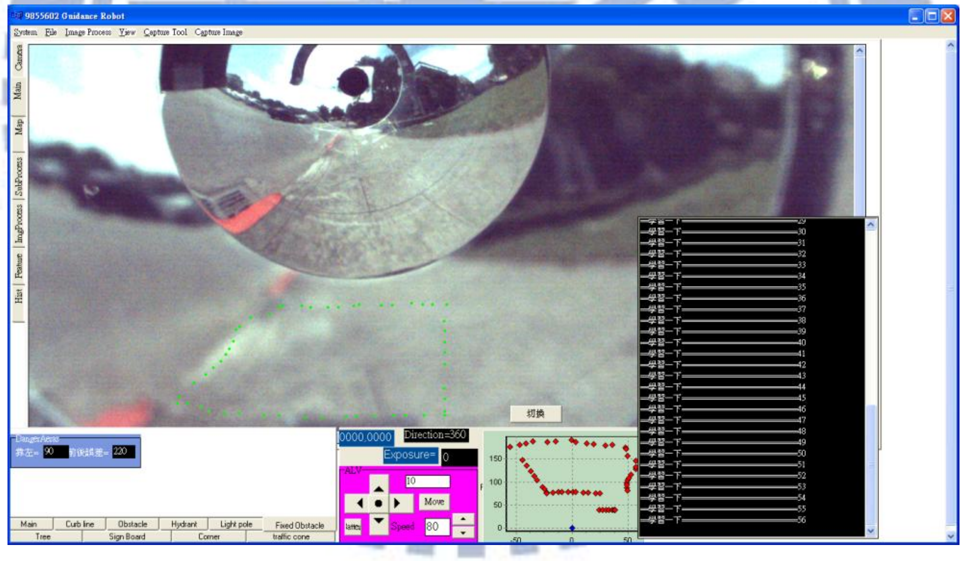

9855602 Guidance Robot
System File Image Process View Capture Tool Capture Image
Camera Main Map SubProcess ImgProcess Feature Hist
切換
Direction=360
Exposure=
Move
Speed
Main Curb line Obstacle Hydrant Light pole Fixed Obstacle
Tree Sign Board Corner traffic cone

Figure 3.12 A learning interface for the trainer to learn the position of the fixed obstacle by clicking mouse on corresponding points in the image region of *Mirror B*.

Algorithm 3.4 ***Creation of a navigation path.***

Input: Odometer readings of vehicle poses, denoted as (P_x, P_y, P_{th}), where P_x and P_y represent the vehicle location and P_{th} represents the direction of the vehicle, in the WCS.

Output: A set of navigation nodes denoted as N_{path}.

Steps.

Step 1. Record into N_{path} the start node N_{begin} of *Type* 0 with the odometer readings $(P_x, P_y, P_{th}) = (0, 0, 0)$.

Step 2. Set the navigation mode, and let the vehicle to navigate forward until arriving at a destination node and stop the vehicle.

Step 3. According to the appointed navigation mode, record into N_{path} the current vehicle pose, denoted as $N_{cur} = (P_x, P_y, P_{th})$ obtained from the odometer readings in *Type* 1 or *Type* 2; and select one of the following seven *additional* learning tasks.

(1) Learn a tree landmark by the method described in Section 5.4.2 to obtain a tree position N_{tree} and the related vehicle pose N_{car}, and record N_{car} in *Type* 4 and N_{tree} in *Type* 5 into N_{path}.

(2) Learn a corner of the lawn landmark by the method (with the detail in Section 5.4.2), obtain a tree position N_{cor} and the related vehicle pose N_{car}, and record N_{car} in *Type* 4 and N_{cor} in *Type* 6 into N_{path}.

(3) Learn a traffic cone landmark by the method mentioned in Section 3.4.2, obtain a traffic cone position N_{tc} and the related vehicle pose N_{car}, and record N_{car} in *Type* 4 and N_{tc} in *Type* 7 into N_{path}.

(4) Learn a road stop line landmark by the method mentioned in Section 3.4.2, obtain a road stop line position N_{WL} and the related vehicle pose N_{car}, and record N_{car} in *Type* 4 and N_{WL} in *Type* 8 into N_{path}.

(5) Learn a signboard landmark by the method mentioned in Section 3.4.2, obtain a signboard position N_{SB} and the related vehicle pose N_{car}, and record N_{car} in *Type* 4 and N_{SB} in *Type* 9 into N_{path}.

(6) Learn a fixed obstacle N_{obs} using the proposed function discussed in

Section 3.5, and record N_{obs} in *Type* 10 into N_{path}.

(7) Learn a curb line calibration node N_{cali}, where the vehicle can "see" a complete curb line segment without occlusion and will calibrate its pose by the "seen" curb line information in the navigation process, and record N_{cali} in *Type* 3 into N_{path}.

Step 4. Go to Step 2 if the destination is not reached yet, where the destination position is selected by the trainer.

Step 5. Record the terminal node N_{end}, denoted as (P_x, P_y, P_{th}), according to the current odometer readings, in *Type* 0 into N_{path}.

Table 3.2 Eleven different types of navigation path nodes.

Type of number	Type of node
Type 0	Start / Terminal node
Type 1	Curb-following navigation node
Type 2	Blind navigation node
Type 3	Curb-line calibration node
Type 4	Localization node
Type 5	Tree landmark node
Type 6	Corner of the lawn landmark node
Type 7	Traffic cone landmark node
Type 8	Road stop line landmark node
Type 9	Signboard landmark node
Type 10	Fixed obstacle node

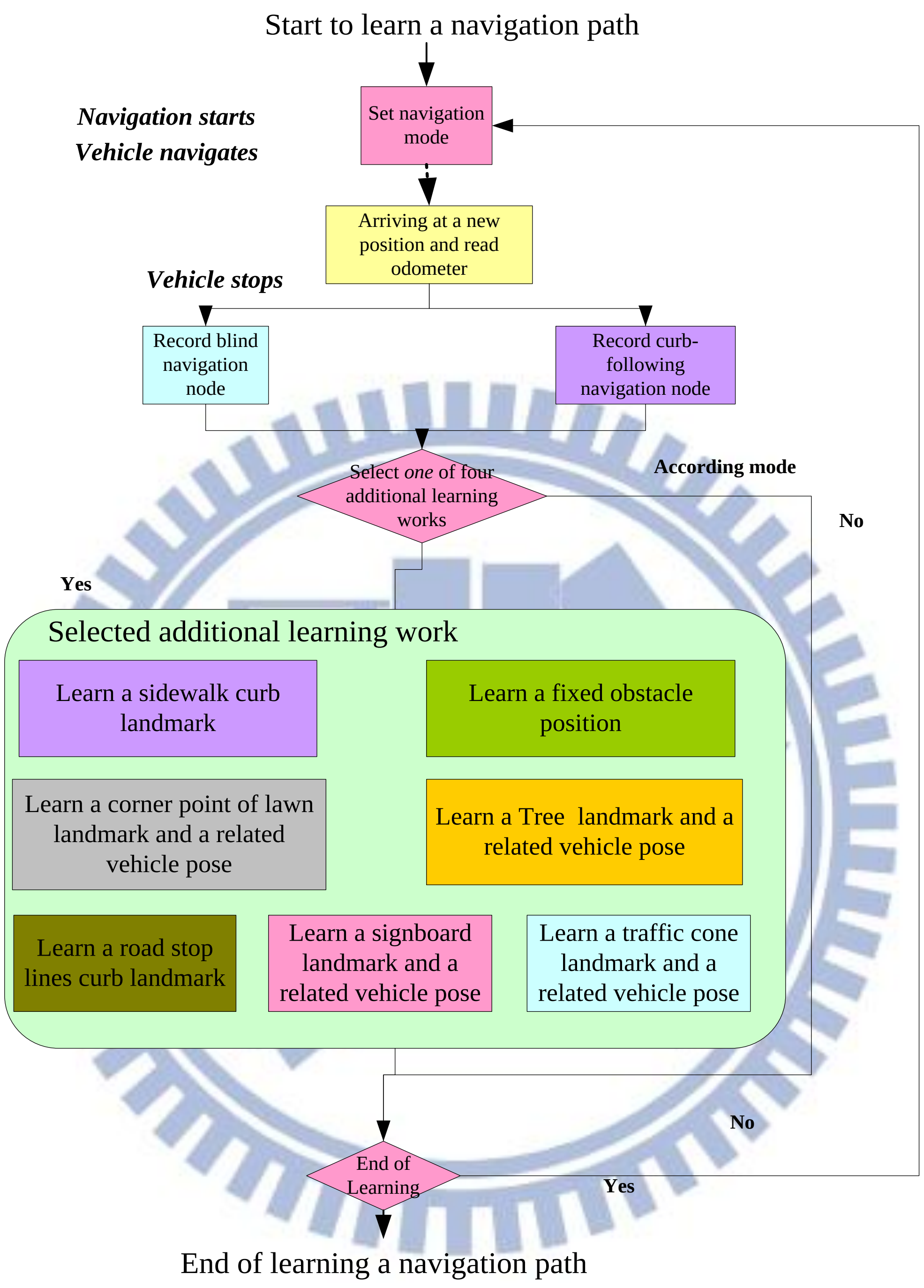

Start to learn a navigation path
Navigation starts
Vehicle navigates
Set navigation mode
Arriving at a new position and read odometer
Vehicle stops
Record blind navigation node
Record curb-following navigation node
Select one of four additional learning works
According mode
No
Yes
Selected additional learning work
Learn a sidewalk curb landmark
Learn a fixed obstacle position
Learn a corner point of lawn landmark and a related vehicle pose
Learn a Tree landmark and a related vehicle pose
Learn a road stop lines curb landmark
Learn a signboard landmark and a related vehicle pose
Learn a traffic cone landmark and a related vehicle pose
No
End of Learning
Yes
End of learning a navigation path

Figure 3.13 The process for navigation path creation.

Chapter 4
Navigation Strategy in Outdoor Environments

4.1 Strategy for Automatic Navigation

After learning the navigation environment, we get the learned environment information including guidance parameters and a navigation path. Then, we can use them to navigate the vehicle automatically in the learned path. But, it is usually difficulty to let the autonomous vehicle “walk” safely in the complicated conditions on sidewalks. In this chapter, we introduce some strategy for automatic safe vehicle navigation. The principles of conducting the navigation work are described in Section 4.2.1. The detailed algorithm for the navigation process is introduced in Section 4.3.

4.1.1 Vehicle localization by alone-path objects

As mentioned in Chapter 2, Because of manufacturing imprecision, the autonomous vehicle usually suffers from incremental mechanic errors during navigation, causing unstable navigation trajectories. To solve the problem, the strategy adopted in this study is to guide the vehicle to constantly localize its position by the learned fixed landmark position. In Chapter 5, the techniques for localizing a landmark along a navigation path will be introduced. Then we can adjust the vehicle posture by changing its orientation and position to modify the odometer.

4.1.2 Dynamic Adjustment learning parameters in

navigation process

Because of the varying lighting condition on the sidewalk, it is not good to always adopt fixed guidance parameters recorded in the learning process to conduct image analysis works. We adopt a “flexible” strategy to accomplish such works in this study, i.e., techniques for dynamic guidance parameter adjustment are designed for use in vehicle navigation. The techniques are based on Chou and Tsai [23]. Also, we use the contour of the signboard to help the vehicle to adjust the learned parameters for signboard detection by principal component analysis (PCA). The method will be introduced in Chapter 6.

4.2 Guidance Technique in Navigation Process

4.2.1 Principle of proposed navigation process

When the vehicle starts navigation, how to arrive at the node recorded in the learning process is an important issue. In this section, we describe the principles [23] behind the proposed technique for vehicle navigation on the learned path. When the vehicle prepares to start navigation, it retrieves a navigation path and some guidance parameters which were recorded in the learning process. The navigation path consists of plenty of sequential nodes, so the vehicle can be guided to the destination through the nodes sequentially. Because the vehicle has mechanic errors, it usually reaches the next node at an imprecise location. So some principles are proposed for use to guide the vehicle to correct such errors and navigate safely to the desired destination. They are described as follows.

(1). The vehicle always follows the sidewalk curb except when it is necessary to detect farther landmarks or to avoid obstacles. After detecting and localizing the curb line, the vehicle modifies its orientation to maintain a safe distance with respect to the curb on the sidewalk.

(2). The vehicle localizes its position and corrects the odometer along the navigation path after every constant-time interval. According to the landmark information which we learned, the vehicle detects and localizes the landmark by the use of the proposed landmark localization techniques. Finally, we conduct adjustment of the vehicle pose.

(3). The vehicle always keeps navigating safely by avoiding collisions along the path. After learning the positions of fixed obstacles, the vehicle conducts a specific procedure to dodge the static obstacle. The procedure will be introduced in Section 4.2.3.

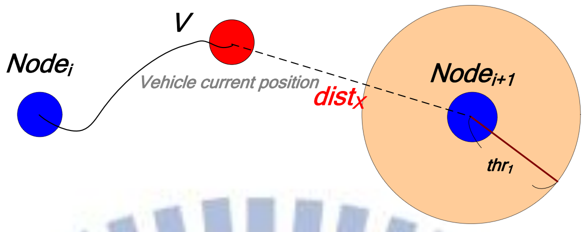
(4). When detecting a landmark using techniques such as dynamic thresholding, the vehicle can adjust guidance parameters if necessary.

General speaking, the vehicle usually localizes itself by the odometer readings to conduct *node-based navigation*. Considering the mechanic errors, we establish two conditions to decide whether the vehicle has arrived at the next node in node-based navigation. The two conditions are introduced in the following.

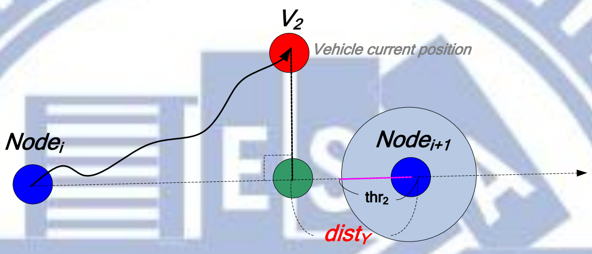
(1). Condition 1 — as shown in Figure 4.1(a), the distance $dist_X$ between the current vehicle position V and the position of the next node $Node_{i+1}$ should be smaller than a threshold thr_1.

(2). Condition 2 — as shown in Figure 4.1(b), the distance $dist_Y$ between the next node $Node_{i+1}$ and the position of the projection of the vehicle on the vector formed by $Node_i$ and $Node_{i+1}$ should be smaller than a threshold thr_2.

By checking the above two conditions, the vehicle can be guided to navigate to the next node and move to the destination. A flowchart illustrating the proposed node-based navigation is shown in Figure 4.2.



(a)

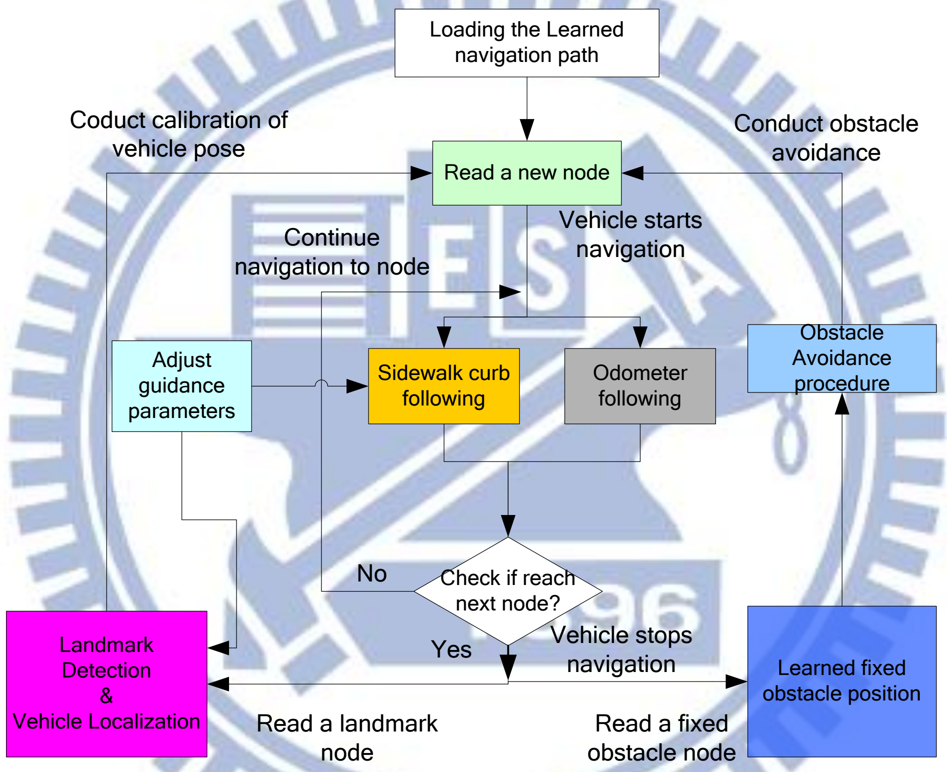


(b)

Figure 4.1 Two conditions to decide if the vehicle arrives at the next node in the navigation process. (a) According to the distance between the vehicle position and the next node position. (b) According to the distance between the next node position and the position of the projection of the vehicle on the vector connecting the current node and the next node.

4.2.2 Automatic vehicle localization by selected landmarks on sidewalks

Although the odometer provides three values P_x, P_y, and P_{th} for the vehicle to identify its position (P_x, P_y) and moving direction P_{th}, they are in general too imprecise to guide the vehicle to the next position correctly in the node-based navigation. To reduce the influence of the incremental mechanic errors on the vehicle's navigation, we may employ as many landmarks along the path as possible. Then, we can use the landmark position to calibrate the odometer reading. In this section, we introduce the proposed technique to calibrate the vehicle position. As shown in Figure 4.3, at first we use the curb line landmark to modify the orientation reading P_{th} of the odometer. When the vehicle arrives at the position of a recorded landmark, we utilize the proposed method for detecting landmarks to calculate its position and then adjust the current position (P_x, P_y) of the vehicle provided by the odometer. Combining the above two strategies, we can make the vehicle reach the next node more precisely.



Loading the Learned navigation path
Coduct calibration of vehicle pose
Conduct obstacle avoidance
Read a new node
Vehicle starts navigation
Continue navigation to node
Adjust guidance parameters
Sidewalk curb following
Odometer following
Obstacle Avoidance procedure
No
Check if reach next node?
Yes
Vehicle stops navigation
Landmark Detection & Vehicle Localization
Learned fixed obstacle position
Read a landmark node
Read a fixed obstacle node

Figure 4.2 Proposed node-based navigation process.

***(A) Adjustment of the odometer reading of the vehicle position conducted near the curb line*.**

Figure 4.4 illustrates the relation between two different positions of the vehicle, the curb, and the landmark. The process of adjustment of the vehicle pose using the sidewalk curb line is divided into two steps. When the vehicle arrives at a position which we recorded, we detect the straight curb line segment which is seen in the omni-image, and calculate the slope angle with respect to the vehicle. Compared with the learned navigation path, we can estimate the current direction of the vehicle. After adjusting the direction, then we start to detect the landmark and obtain its position. According to the landmark position we recorded in the learning process, we utilize the vehicle orientation which we just calibrate to compute the current vehicle position by the relation between the landmark position and the vehicle position in the GCS as shown in Figure 4.5. The adopted method to calibrate the odometer is described in the following algorithm.

Algorithm 4.3 ***Conducting adjustment of the odometer reading of the vehicle position near the curb line.***

Input: a recorded landmark position L_{record}, the odometer readings of the vehicle pose, a recorded slope angle θ of the curb line, and a recoded vehicle pose V_L (P_x, P_Y, P_{th}).

Output: none.

Steps.

Step 1. Turn the vehicle to the recorded direction P_{th}, compute the curb line detection process which is described in Chapter 6, and compute the slope angle θ' of the curb line relative to the vehicle direction.

Step 2. Compute an adjustment angle θ_{adj} by the following equation:

$$\theta_{adj} = \theta' - \theta, \tag{4.1}$$

and modify the orientation odometer reading to be θ_{adj} which is then taken as the correct vehicle orientation P_{th}'.

Step 3. Detect the landmark in the acquired image and compute its position at L_{ccs} in the CCS (using the method described in Chapter 5); and by the coordinate transformation between the CCS and the VCS described in Equation (3.6) with L_{ccs} in the CCS as input, compute the landmark position L_{VCS} and describe it with coordinates (l_x, l_y) in the VCS.

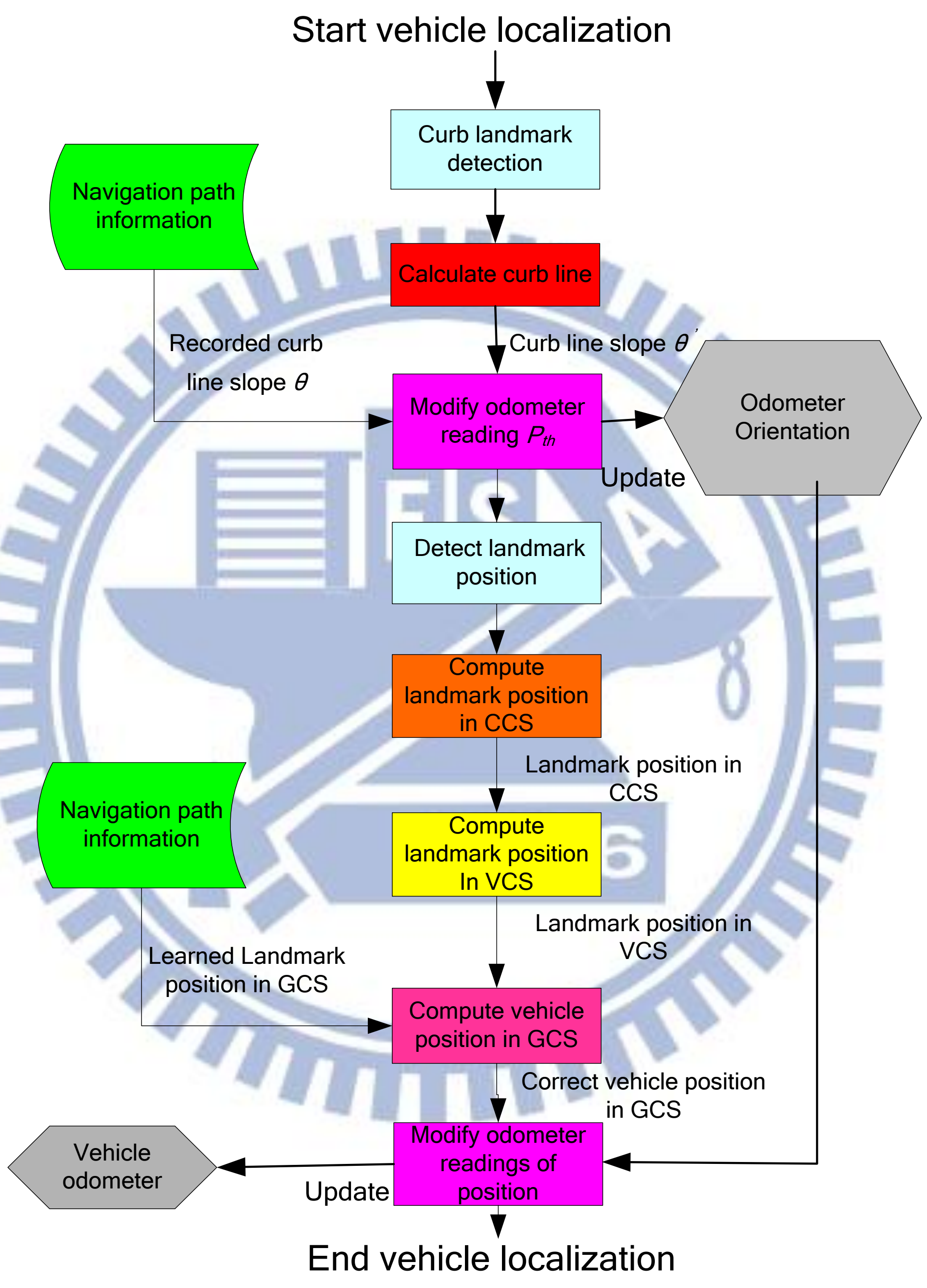
Step 4. From the learned navigation path, obtain the recorded landmark position L_{record} at coordinates (C_x, C_y) in the GCS, and use the calibrated orientation P_{th}' to compute the current vehicle position (X_{cali}, Y_{cali}) in the GCS by the following equations:

$$\begin{bmatrix} X_{cali} \\ Y_{cali} \end{bmatrix} = \begin{bmatrix} C_x \\ C_y \end{bmatrix} + \begin{bmatrix} \cos P_{th}' & \sin P_{th}' \\ -\sin P_{th}' & \cos P_{th}' \end{bmatrix} \begin{bmatrix} l_x \\ l_y \end{bmatrix}. \tag{4.2}$$

Step 5. Replace the imprecise position readings of the odometer, (P_X', P_Y'), by the calculated vehicle position (X_{cali}, Y_{cali}).

(B) Adjustment of the odometer reading of the vehicle position conducted far off the curb line.

The process for adjustment of the odometer reading of the vehicle position conducted far off the curb line is similar to the above-mentioned process for adjustment near the curb line. The detail of the process is shown in Figure 4.6. First, we detect and localize a nearby curb line segment for the purpose to adjust the orientation reading in a similar way as described previously at a node P_1 in the learned path. Next, we conduct a slight difference step, i.e., we guide the vehicle a step further to another node V_2, which is a location recorded in the navigation path far off the curb line, for the purpose of detecting the landmark at a close location. Comparing the two adjustment processes, the second one might cause some mechanical errors and provides erroneous odometer readings after the vehicle moves from V_1 to V_2. For this, after detecting the landmark and localizing it, we use the same technique to compute the current vehicle position by the relation between the landmark position and the vehicle position in the GCS as described previously.



Start vehicle localization
Curb landmark detection
Navigation path information
Calculate curb line
Recorded curb line slope θ
Curb line slope θ'
Modify odometer reading P_{th}
Odometer Orientation
Update
Detect landmark position
Compute landmark position in CCS
Landmark position in CCS
Navigation path information
Compute landmark position In VCS
Landmark position in VCS
Learned Landmark position in GCS
Compute vehicle position in GCS
Correct vehicle position in GCS
Vehicle odometer
Update
Modify odometer readings of position
End vehicle localization

Figure 4.3 Proposed odometer reading adjustment process.

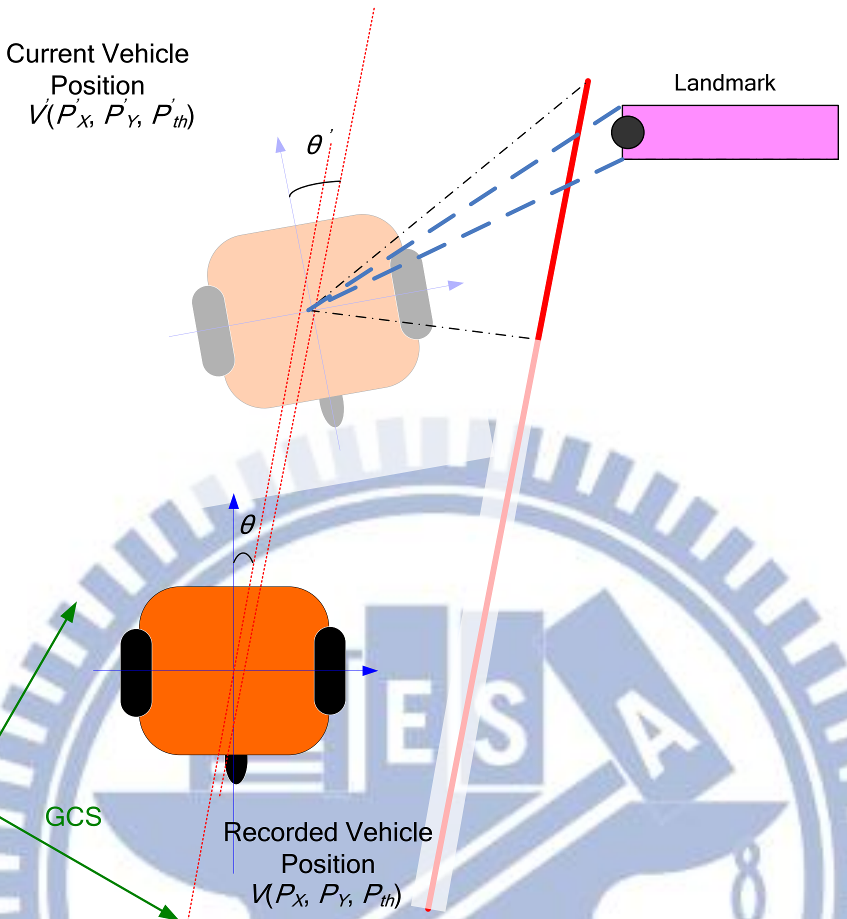
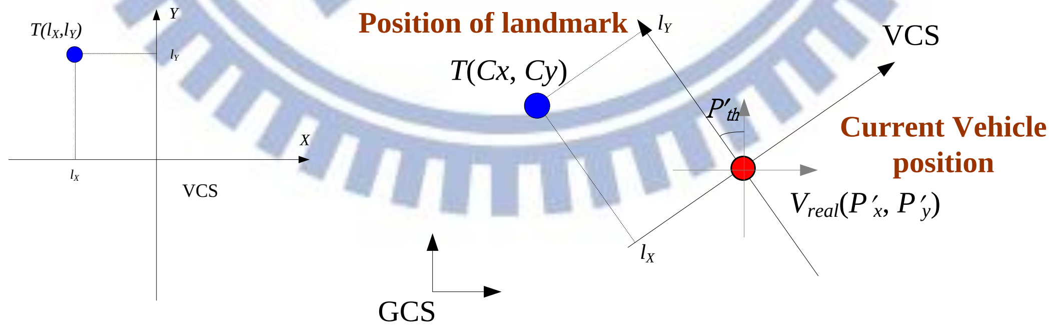


Figure 4.4 A recorded vehicle position V and the current vehicle position V' in the GCS.



(a) (b)

Figure 4.5 Landmark detection for vehicle localization at position T. (a) At coordinates (l_x, l_y) in VCS. (b) At coordinates (C_x, C_y) in GCS.

4.2.3 Fixed-obstacle avoidance process

As mentioned in Chapter 3, there might be fixed obstacles along the path. A fixed obstacle might block the vehicle or cause it to fall outside the sidewalk. So we propose a strategy to help the vehicle avoid the obstacle safely, as described in the following.

As shown in Figure 4.7, the vehicle is navigating to reach a node V_1 in the learned path. The next node V_2 is just occupied by an obstacle. According to the obstacle parameters O_x and O_y recorded in the learned navigation path which specify how far the vehicle should keep away from the obstacle, we utilize the two parameters to compute three locations by which we can insert three respective nodes in the navigation path as shown in Figure 4.7. The first is placed at the left rear side with respect to the obstacle position. The second is placed at the left front side. And the last is placed right in front of the obstacle. From the third newly-placed node, the vehicle can go back to the original path. By visiting the three new nodes in sequence, the vehicle can dodge the obstacle successfully and be guided to visit node V_3.

4.3 Detailed Algorithm of Navigation Process

In this section, we introduce the detailed algorithm proposed in this study for vehicle navigation in the navigation process. With the learned information, the vehicle navigates along the learned path by the way of visiting each recorded node consecutively and finishes works at specific positions until reaching the destination point. The flowchart of the whole navigation process is shown in Figure 4.8. The whole navigation process is described in the following algorithm.

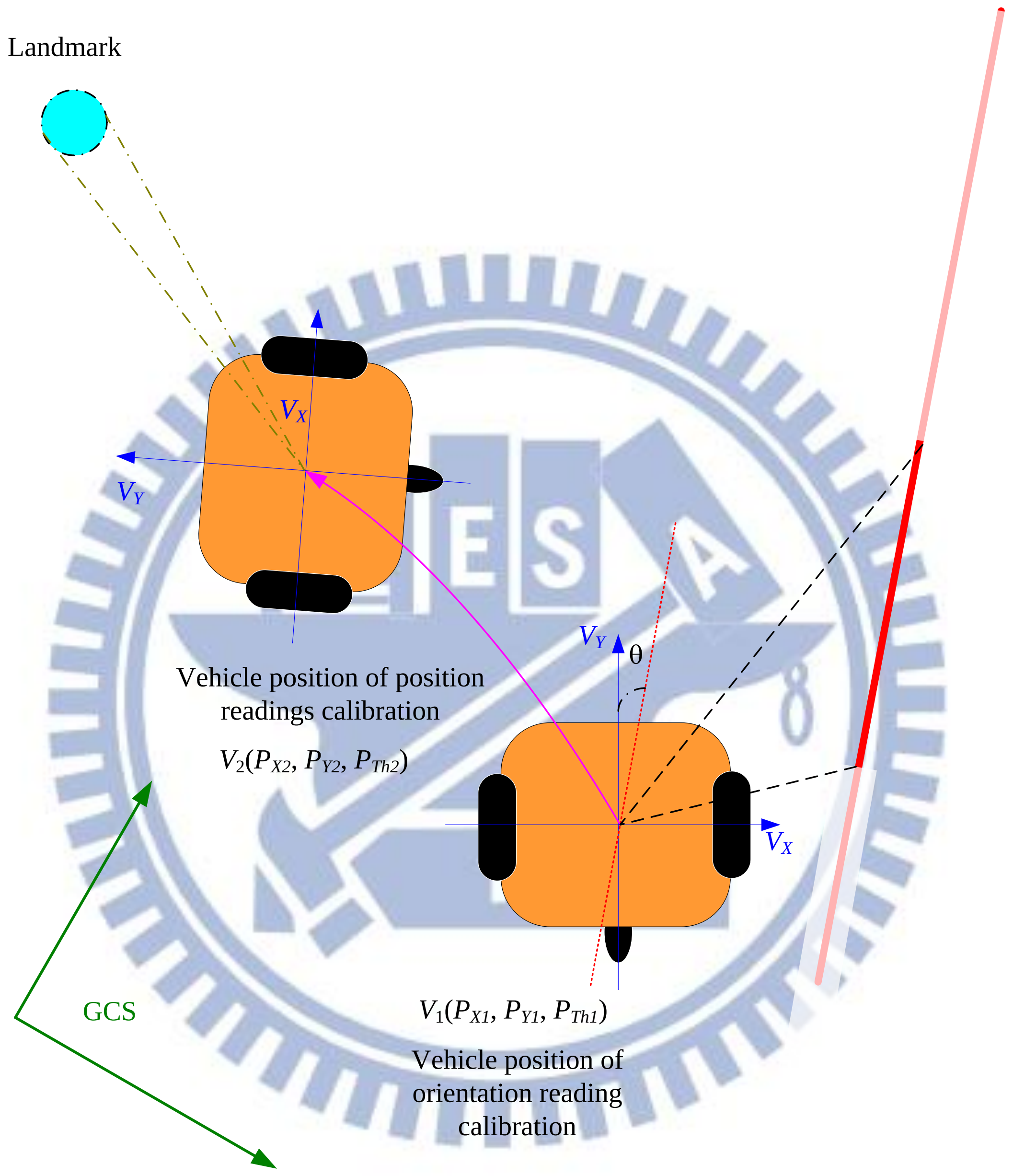

Landmark
V_X
V_Y
V_Y
θ
Vehicle position of position readings calibration
$V_2(P_{X2}, P_{Y2}, P_{Th2})$
V_X
GCS
$V_1(P_{X1}, P_{Y1}, P_{Th1})$
Vehicle position of orientation reading calibration

Figure 4.6 Process of odometer calibration position is far off the curb line. The vehicle detects the curb line at V_1 to calibrate the orientation and then navigates to V_2 to calibrate the position using landmark.

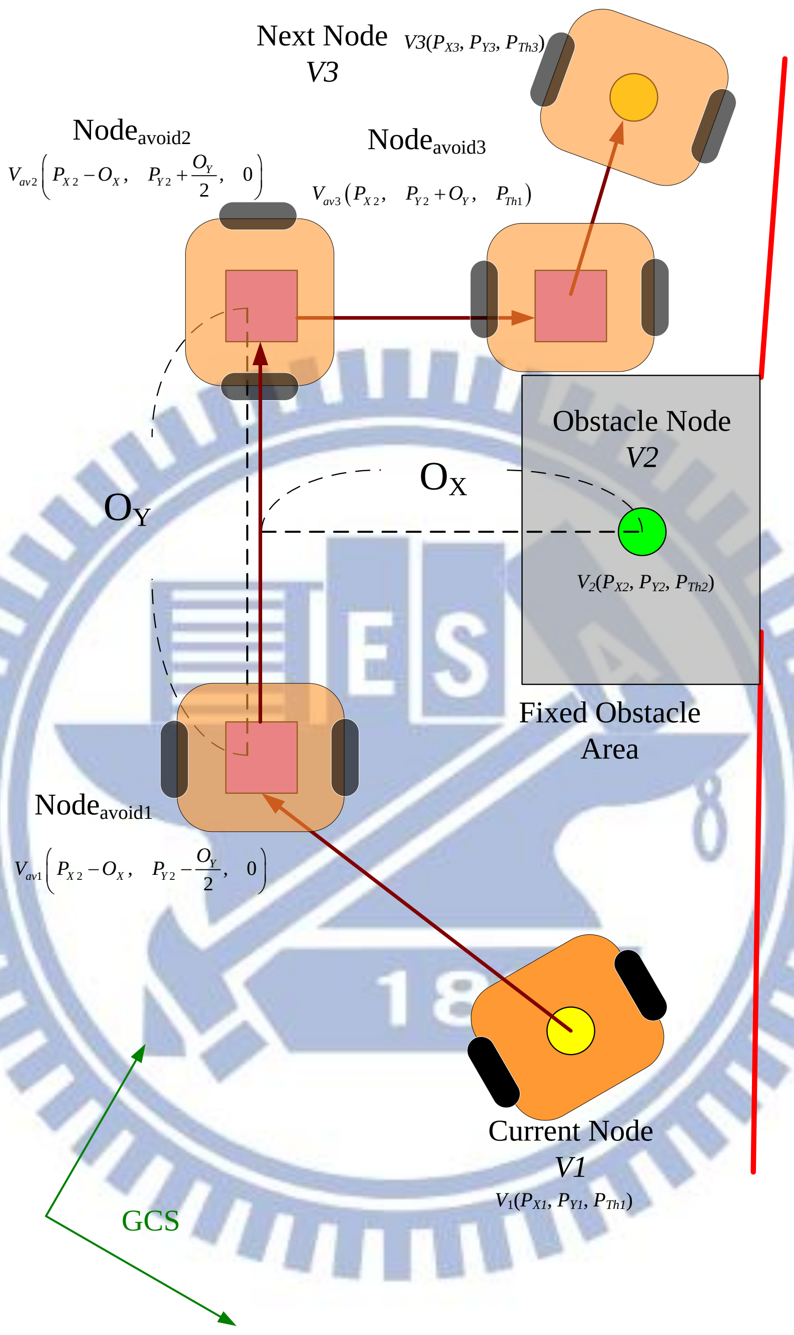

Next Node
$V3$
$V3(P_{X3}, P_{Y3}, P_{Th3})$
Node$_{\text{avoid2}}$
$V_{av2}\left(P_{X2}-O_X,\ P_{Y2}+\frac{O_Y}{2},\ 0\right)$
Node$_{\text{avoid3}}$
$V_{av3}(P_{X2},\ P_{Y2}+O_Y,\ P_{Th1})$
Obstacle Node
$V2$
O_X
O_Y
$V_2(P_{X2}, P_{Y2}, P_{Th2})$
Fixed Obstacle Area
Node$_{\text{avoid1}}$
$V_{av1}\left(P_{X2}-O_X,\ P_{Y2}-\frac{O_Y}{2},\ 0\right)$
Current Node
$V1$
$V_1(P_{X1}, P_{Y1}, P_{Th1})$
GCS

Figure 4.7 Process of fixed obstacle avoidance. We insert the Node$_{\text{avoid1}}$, Node$_{\text{avoid2}}$, and Node$_{\text{avoid3}}$.for obstacle avoidance in the original navigation path.

Algorithm 4.3 ***Navigation Process***.

Input: a learned navigation path N_{path} with relevant guidance parameters, and learned data of camera parameters.

Output: none.

Steps.

Step 1. Read from N_{path} a navigation node N_{next} and relevant guidance parameters.

Step 2. Rotate the vehicle toward the next node N_{next}..

Step 3. If a curb line following mode is adopted, modify the vehicle direction after localizing the curb landmark using the dynamic threshold adjustment technique as described in Section 6.1.1 [23].

Step 4. Check if the next node N_{next} is reached by the mentioned two principles in Section 4.2.1: if not, go to Step 3; otherwise, continue.

Step 5. If a fixed obstacle node is read from N_{path}, insert obstacle avoidance nodes into the navigation path and go to Step 8.

Step 6. If a landmark node is read from N_{path}, take the following steps and then go to Step 8.

(1) Adjust the exposure value to the desired illumination in the relevant environment windows in the image.

(2) Detect the landmark and calculate the landmark position as described in Chapters 5 and 6.

(3) Utilize the position of the landmark to localize the vehicle position and modify the odometer reading as described in Section 4.2.

Step 7. If a curb line calibration node is read from N_{path}, modify the orientation reading of the odometer after detecting and localizing a curb line segment, as described in Section 4.2.

Step 8. Repeat Steps 1 through 7 until there exists no remaining node in N_{path}.

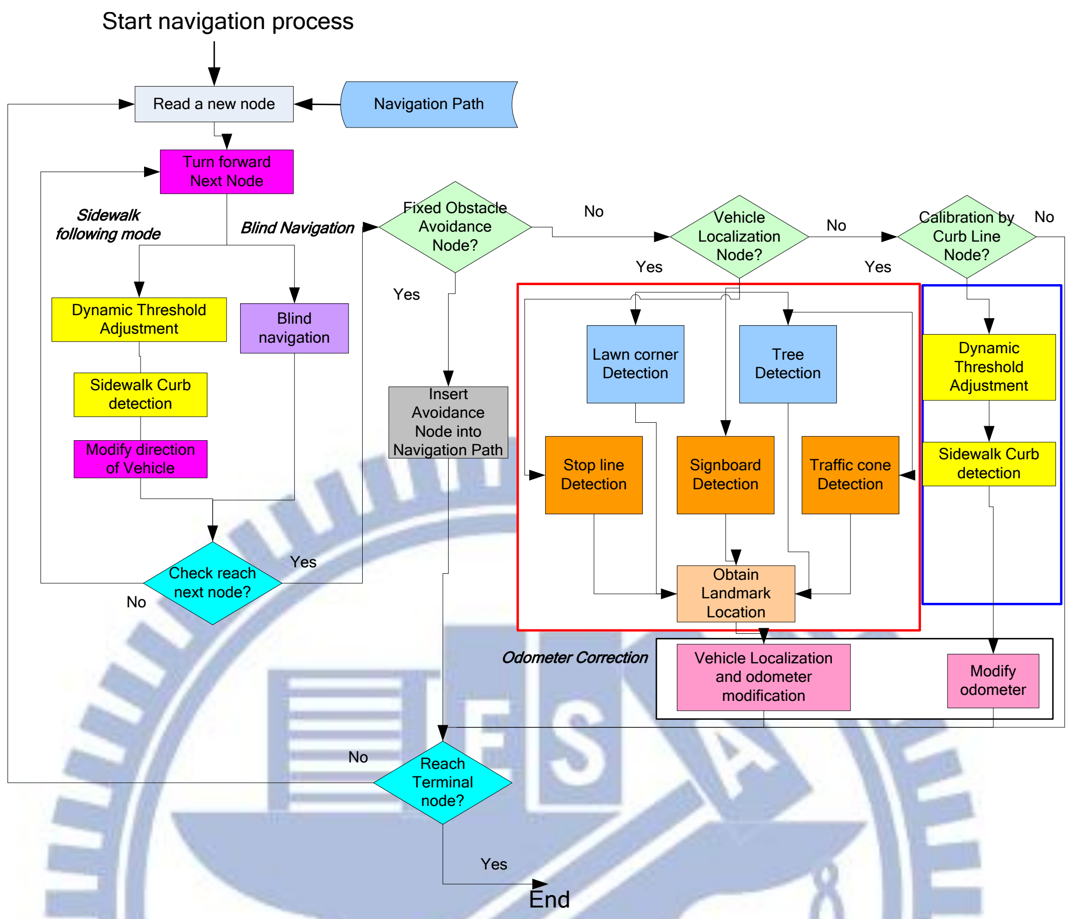


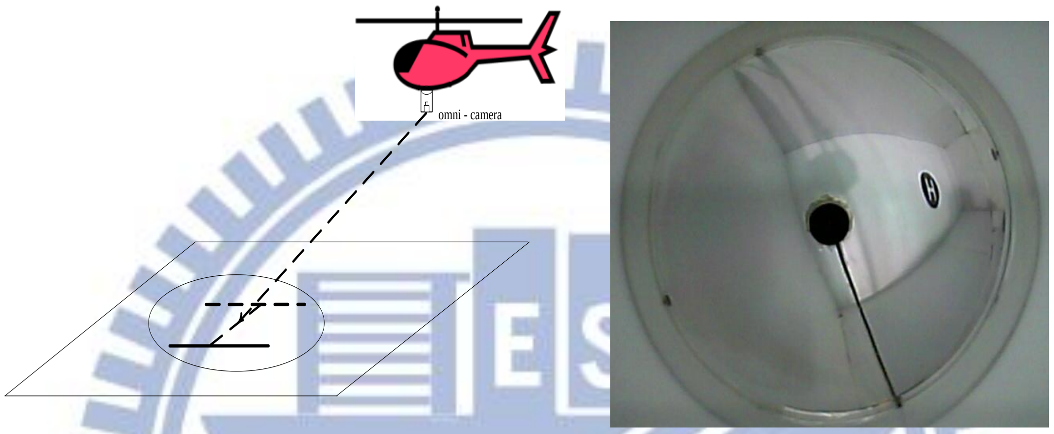
Figure 4.8 Flowchart of detailed proposed navigation process.

Chapter 5
Natural Landmark Detection in Images Using New Space Line Detection Technique

5.1 Idea of Proposed Space Line Detection Technique

In this study, it is desired to develop a space line detection method to localize landmarks on the navigation path for vehicle navigation. However, compared to the result of using the traditional projective camera, the projection of a space line on an omni-image using an omni-camera is not a line shape but a *conic-section curve* [25]. Wu and Tsai [25] proposed a line detection method which detects lines in an H-shaped landmark for use in automatic helicopter landing, as illustrated in Figure 5.1. Firstly, they proved that the projection of a space line in the omni-image is a conic-section curve. Then, by using a 2D Hough transform, they extracted the conic section curve in the omni-image and localized the boundary lines of the H shape for conducting the helicopter landing.

However, Wu and Tsai's method is based on the fact that the parameters of the hyperboloidal mirror are known. It is known that the parameters of a hyperboloidal mirror cannot be calibrated easily. As mentioned in Chapter 3, it is more convenient for us to utilize the *pano-mapping table* to calibrate the camera. Thus, we propose new space line detection techniques in this study by the use of the *pano-mapping table*. We utilize the *space plane* which goes through the space line and the center of the mirror, instead of trying to obtain directly the conic section curve in the omni-image. By the use of the two-mirror camera system, the proposed line detection technique is introduced in Section 5.2.1. Besides, we derive a method to obtain 3D information of three vertical lines based on the method for line detection proposed in this study. The derivation of 3D information is introduced in Section 5.2.2.



(a) (b)

Figure 5.1 Wu and Tsai [25] proposed a line detection method for the omni-image to conduct automatic helicopter landing. (a) Illustration of automatic helicopter landing on a helipad with a circled H shape. (b) An omni-image of a simulated helipad.

Finally, by the use of the proposed space line detection technique, many types of landmarks can be detected and utilized for vehicle navigation. We introduce the proposed tree trunk detection method in Section 5.4 and the proposed lawn corner detection technique in Section 5.5. Artificial landmark localization also utilizes these methods, which will be introduced in Chapter 6.

5.2 Proposed Technique for Space Line Detection

5.2.1 Line detection using pano-mapping table

If we want to detect a space line which is projected onto the omni-image, it costs much time to calibrate the camera parameters. In this section, a space line detection technique [23] for use on omni-images which are taken by the two-mirror omni-camera used in this study is proposed. Instead of directly obtaining the conic section curve, we detect the space plane which goes through a specified line and the mirror center. In the following, we describe the proposed detection process.

Assume that a *pano-mapping* table has been established in advance. Also, assume that the space line *L* to be detected is projected by *Mirror B* onto the omni-image, and that *G* is any space point on *L*. At first, we propose a way to represent a vector which goes through *G* and the center of mirror O_B used in this study. A light ray going through the space point *G* is projected by *Mirror B* onto an image point *I* as shown in Figure 5.2. The mirror center O_B and *G* together form a vector V_G', denoted as (G_x', G_y', G_z') in the CCS_{local}. This vector V_G' can be described using the azimuth and elevation angles θ and α by the following equations:

$$G_x' = \cos\alpha \times \cos\theta; \quad G_y' = \cos\alpha \times \sin\theta; \quad G_z' = \sin\alpha. \tag{5.1}$$

Also, as mentioned previously, to increase the front FOV of the camera, we make the camera system slant up for a specific angle, denoted as γ. By the use of the rotation matrix introduced in Equation (2.5), the transformation function between the coordinates (X', Y', Z') of the original CCS_{local} and the coordinates (X, Y, Z) of the rotated CCS can be described as follows:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-\gamma) & -\sin(-\gamma) \\ 0 & \sin(-\gamma) & \cos(-\gamma) \end{bmatrix} \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix}. \tag{5.2}$$

According to the coordinate transformation described by Equation (5.2), we can convert the vector V_G' into a new one V_G, which represents the vector with an azimuth angle θ and an elevation angle α going through the mirror center in the rotated CCS and may be described by the following equations:

$$V_G = \begin{bmatrix} G_x \\ G_y \\ G_z \end{bmatrix} = \begin{bmatrix} \cos\alpha \times \cos\theta \\ \cos\alpha \times \sin\theta \times \cos\gamma + \sin\alpha \times \sin\gamma \\ -\cos\alpha \times \sin\theta + \sin\alpha \times \cos\gamma \end{bmatrix}. \tag{5.3}$$

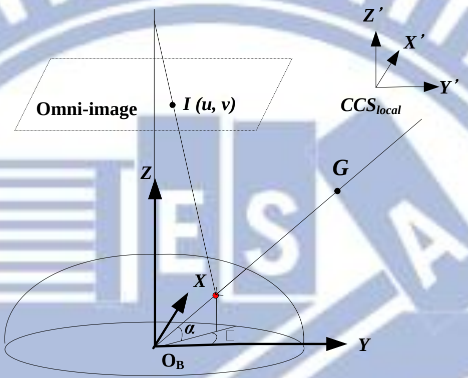


Figure 5.2 A space point with an elevation angle α and an azimuth angle θ.

Then, as shown in Figure 5.3, let I_L be the conic section curve resulting from projecting the space line L onto the omni-image. Also, let Q be the space plane which goes through L and the mirror center O_B. Assume that the coordinates (X, Y, Z) of a point is on the space plane Q, and let $N_Q = (l, m, n)$ be the normal vector of the space plane Q. Then, the space plane Q may be described by:

$$lX + mY + nZ = 0. \tag{5.4}$$

Besides, because the normal vector N_Q and the vector V_G are perpendicular to each other, the coefficients in (5.4) actually are related to the elements of the vector V_G of Q by the equality $N_Q \cdot V_G = (l, m, n) \cdot (G_x, G_y, G_z) = 0$ which we describe in the following:

$$N_Q \cdot V_G = (l, m, n) \cdot (G_x, G_y, G_z) = lG_x + mG_y + nG_z = 0. \tag{5.5}$$

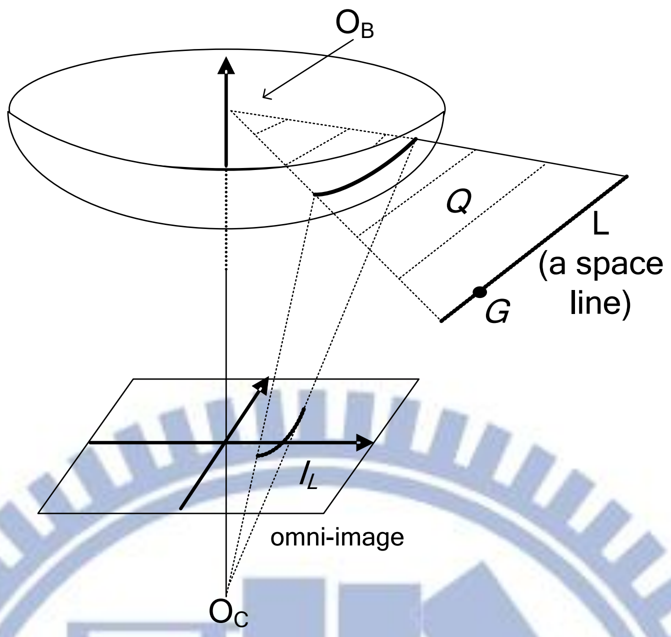


Figure 5.3 A space line L projected on I_L in an omni-image.

Then, substituting Equation (5.3) into Equation (5.5), we get

$$l(\cos\alpha\times\cos\theta)+m\times(\cos\alpha\times\sin\theta\times\cos\gamma+\sin\alpha\times\sin\gamma) \\ +n\times(-\cos\alpha\times\sin\theta+\sin\alpha\times\cos\gamma)=0. \quad (5.6)$$

Equation (5.6) may be divided by G_x to get

$$l+m\times\frac{(\cos\alpha\times\sin\theta\times\cos\gamma+\sin\alpha\times\sin\gamma)}{(\cos\alpha\times\cos\theta)}+n\times\frac{(-\cos\alpha\times\sin\theta+\sin\alpha\times\cos\gamma)}{(\cos\alpha\times\cos\theta)}=0. \quad (5.7)$$

However, the above equation consists of the three unknown parameters l, m, and n which represent the normal of the space plane Q. For the purpose of equation reduction, we assume that n is not equal to zero. Then, we may divide Equation (5.7) by n to get an alternative form as follows:

$$\frac{l}{n}+\frac{m}{n}\times\frac{(\cos\alpha\times\sin\theta\times\cos\gamma+\sin\alpha\times\sin\gamma)}{(\cos\alpha\times\cos\theta)}+\frac{(-\cos\alpha\times\sin\theta+\sin\alpha\times\cos\gamma)}{(\cos\alpha\times\cos\theta)}=0.$$

(5.8)

We may reduce Equation (5.8) further to get the following result:

$$B + A \times \frac{(\cos\alpha \times \sin\theta \times \cos\gamma + \sin\alpha \times \sin\gamma)}{(\cos\alpha \times \cos\theta)} + \frac{(-\cos\alpha \times \sin\theta + \sin\alpha \times \cos\gamma)}{(\cos\alpha \times \cos\theta)} = 0$$

(5.9)

where $A = m/n$, $B = l/n$, which may be rewritten as

$$B + A\, a_0 + a_1 = 0 \qquad (5.10)$$

where

$$A = \frac{m}{n},\quad B = \frac{l}{n},\quad a_0 = \frac{(\cos\alpha \times \sin\theta \times \cos\gamma + \sin\alpha \times \sin\gamma)}{(\cos\alpha \times \cos\theta)},$$

$$a_1 = \frac{(-\cos\alpha \times \sin\theta + \sin\alpha \times \cos\gamma)}{(\cos\alpha \times \cos\theta)}.$$

Finally, we utilize two variables *A* and *B*, as shown by (5.10), to represent the parameters *l*, *m*, and *n*. In summary, we can use a simple technique using the 2D Hough transform to compute the parameters *A* and *B*. A detailed algorithm is introduced in the following.

Algorithm 5.1 ***Space line detection.***

Input: an input edge-point image I_{edge} which includes the conic section projection *L*' in an image I_{edge} of a space line *L*, and the pano-mapping table for *Mirror B*.

Output: the values of the two parameters $A_{\max}$ and $B_{\max}$, representing a normal vector of the space plane described by Equation (5.10).

Steps.

Step 1. Set up a 2D Hough space *S* with the parameters *A* and *B* and set all cell values to be zero.

Step 2. For an edge point I at coordinates (u, v) in I_{edge}, and look up the pano-mapping table to obtain a corresponding azimuth-elevation angle pair (θ, α).

Step 3. Compute the parameter values A and B by Equation (5.9) using θ and α; and if A, B, θ, and α satisfy Equation (5.9), then increment the count in the cell (A, B) of the Hough space S by one.

Step 4. Repeat Steps 2 and 3 until all the edge points in I_{edge} are computed.

Step 5. Take the cell (A_{max}, B_{max}) with a maximum count in S as output.

We can obtain the normal vector (l, m, n) from the above-presented algorithm; however, it costs much time to calculate the 2D Hough transform. Chou and Tsai [23] proposed a method which detects a vertical line by using a 1D Hough transform and the normal vector of the space plane. The vertical line, called L_Y *line* hereafter, is parallel to the Y-axis line in the GCS. In this study, we propose a method for detecting two specific lines. One line of the two is parallel to the X-axis, called the L_X *line*. The other is parallel to the Z-axis, called the L_Z *line*. These three specific lines are illustrated in Figure 5.4. We will describe them, respectively

At first, we review the method for L_Y line detection. Note that the direction vector of L_Y is $D_Y = (d_{Yx}, d_{Yy}, d_{Yz}) = (0, 1, 0)$. Therefore, Equation (5.5) leads to $0\times l + 1 \times m + 0\times n = 0$. Then, it is easy to figure out that m is equal to zero. Thus, Equation (5.7) can be reduced to be the following equation:

$$l + n\times\frac{(-\cos\alpha\times\sin\theta+\sin\alpha\times\cos\gamma)}{(\cos\alpha\times\cos\theta)} = 0. \tag{5.11}$$

By Equation (5.10), we may reduce Equation (5.11) to be

$$B + a_1 = 0 \tag{5.12}$$

or equivalently, to be

$$B = -a_1. \tag{5.13}$$

In a similar way as described in Algorithm 5.1, we can use a 1D Hough transform to find the parameter B, which represents a normal vector of the specific space plane through the L_Y line and the mirror center.

Then, it can be that the direction vector of the L_X is $D_X = (d_{Xx}, d_{Xy}, d_{Xz}) = (1, 0, 0)$. Therefore, Equation (5.5) leads to $1\times l + 0\times m + 0\times n = 0$. Then, it is easy to figure out that l is equal to zero. Thus, Equation (5.7) can be reduced to the following equation:

$$m \times \frac{(\cos\alpha\times\sin\theta\times\cos\gamma+\sin\alpha\times\sin\gamma)}{(\cos\alpha\times\cos\theta)} + n\times\frac{(-\cos\alpha\times\sin\theta+\sin\alpha\times\cos\gamma)}{(\cos\alpha\times\cos\theta)} = 0\,. \tag{5.14}$$

By Equation (5.10), we can reduce Equation (5.14) to be

$$A\times a_0 + a_1 = 0, \tag{5.15}$$

or equivalently,

$$A = \frac{-a_1}{a_0} \tag{5.16}$$

So we can also use Algorithm 5.1 to find the parameter A, which represents a normal vector of the specific plane through the L_X line and the mirror center.

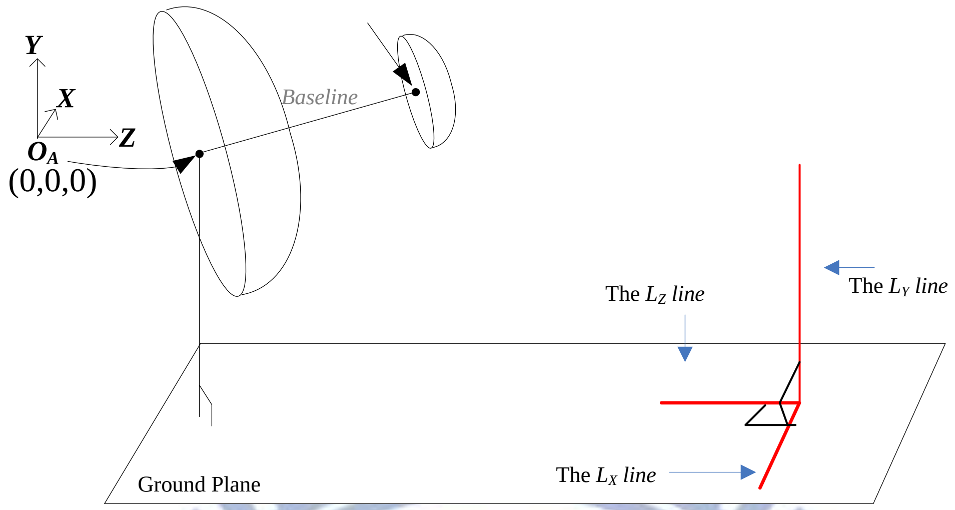


Figure 5.4 Three specific space lines.

The last one is about L_Z line detection in the space. The direction vector of the horizontal line L_Z is $D_Z = (d_{Zx}, d_{Zy}, d_{Zz}) = (0, 0, 1)$. Thus, Equation (5.5) leads to $0 \times l + 0 \times m + 1 \times n = 0$, or equivalently, $n = 0$. Equation (5.7) can thus be reduced to be the following equation:

$$l + m \times \frac{(\cos\alpha \times \sin\theta \times \cos\gamma + \sin\alpha \times \sin\gamma)}{(\cos\alpha \times \cos\theta)} = 0. \tag{5.17}$$

By Equation (5.10), we can reduce Equation (5.17) to be

$$l + m \times a_0 = 0, \tag{5.18}$$

or equivalently,

$$-a_0 = K \tag{5.19}$$

where

$$a_0 = \frac{(\cos\alpha \times \sin\theta \times \cos\gamma + \sin\alpha \times \sin\gamma)}{(\cos\alpha \times \cos\theta)}, \quad K = \frac{l}{m}.$$

Again, we can use a simple 1D Hough transform technique to compute the parameters *K*. The detailed algorithm is introduced in the following.

Algorithm 5.2 ***L_Z* space line detection.**

Input: an input edge-point image I_{edge} which includes the conic section projection *L*' in an image I_{edge} of a L_Z space line *L*, and the pano-mapping table for *Mirror B*.

Output: the values of the parameter $K_{\max}$ representing a normal vector of the space plane described by Equation (5.16).

Steps.

Step 1. Set up a 1D Hough space *S* with the parameter *K* and set all cell values to be zero.

Step 2. For an edge point *I* at coordinates (*u*, *v*) in I_{edge}, look up the pano-mapping table and obtain a corresponding azimuth-elevation angle pair (θ, α).

Step 3. Compute the parameter *K* by Equation (5.19) using θ and α, and if *K*, θ, and α satisfy Equation (5.19), then increment the count in the cell *K* of the Hough space *S* by one.

Step 4. Repeat Steps 2 and 3 until all the edge points in I_{edge} are processed.

Step 5. Take the cell $K_{\max}$ with a maximum count in *S* as output.

5.2.2 3D data computation using three space lines

In this section, based on the proposed space line detection technique described in the previous section, we can derive the 3D data from three vertical space lines from the omni-image, as described subsequently.

(A) 3D data computation using a L_Y line.

As shown in Figure 5.5, a L_X space line is projected onto I_{L1} and I_{L2} on the regions of *Mirrors B* and *S*, respectively. The center O_B of *Mirror B* is located at coordinates (0, 0, 0) in the CCS as we previously assumed. We can calculate the position of the center O_S of *Mirror S* by the slant angle γ and the length of *baseline* which is denoted as b to be (0, $b\sin\gamma$, $b\cos\gamma$) in the CCS coordinates system. According to Equation (5.4), two space planes Q_1 and Q_2 going through L_Y and the center of mirror, O_B and O_S, respectively, can be described by the following equations:

$$l_BX + m_BY + n_BZ = 0; \tag{5.20}$$

$$l_SX + m_S(Y - b\sin\gamma) + n_S(Z - b\cos\gamma) = 0, \tag{5.21}$$

where (l_B , m_B , n_B) represents the normal vector of Q_1 and (l_S , m_S , n_S) represents that of Q_2.

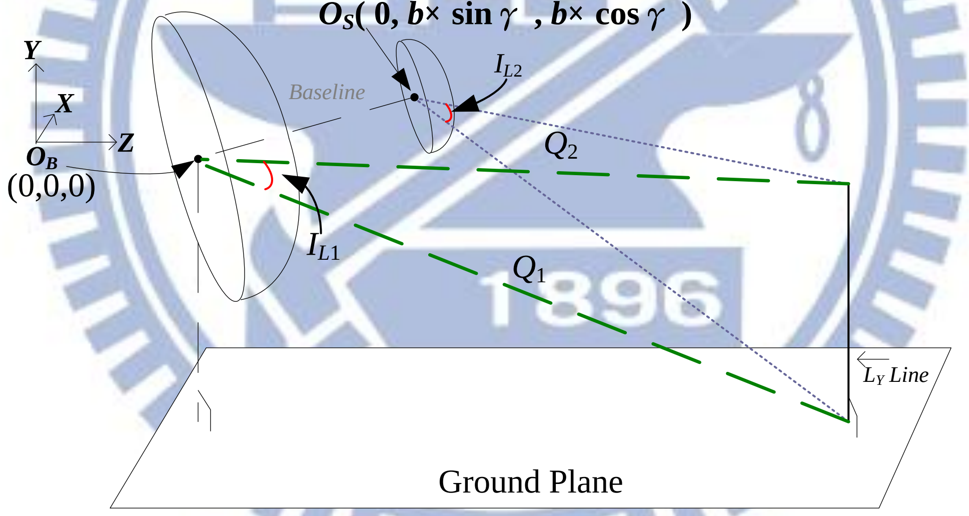


Figure 5.5 A space line projected onto I_{L1} and I_{L2} on two mirrors in the used two-mirror omni-camera.

As mentioned previously, the direction vector of L_Y is $D_Y = (d_{Yx}, d_{Yy}, d_{Yz}) = (0, 1, 0)$. Thus, we know that m_B and m_S are both zero, and the above two space plane equations can be reduced into the following forms:

$$l_BX + n_BZ = 0; \tag{5.22}$$

$$l_SX + n_S(Z - b\cos\gamma) = 0, \tag{5.23}$$

or equivalently,

$$B_1X + Z = 0; \tag{5.24}$$

$$B_2X + (Z - b\cos\gamma) = 0, \tag{5.25}$$

where $B_1 = l_B/n_B$ and $B_2 = l_S/n_S$.

Solving Equations (5.24) and (5.25), we can obtain the following desired solution for X and Z:

$$X = \frac{b\times\cos\gamma}{B_2 - B_1};$$

$$Z = -B_1\times X = -B_2\times X + b\times\cos\gamma\,. \tag{5.26}$$

It is noted that Equation (5.26) cannot be solved when B_1 is equal to B_2, resulting in a parallelism between the two space planes Q_1 and Q_2.

(B) 3D data computation using a L_X line.

The process for 3D computation using a L_X line is similar to that using a L_Y line. As shown in Figure 5.6, the equations of Q_3 and Q_4 may be described in the following:

$$l_{B1}X + m_{B1}Y + n_{B1}Z = 0; \tag{5.27}$$

$$l_{S1}X + m_{S1}(Y - b\sin\gamma) + n_{S1}(Z - b\cos\gamma) = 0, \tag{5.28}$$

where (l_{B1}, m_{B1}, n_{B1}) represents the normal vector of Q_3 and (l_{S1}, m_{S1}, n_{S1}) represents that of Q_4.

As mentioned previously, the direction vector of the L_X is $D_X = (d_{Xx}, d_{Xy}, d_{Xz}) = (1, 0, 0)$. Thus, we know that l_{B1} and l_{S1} are both zero, and the above two space plane equations can be reduced into the following forms:

$$m_{B1}Y + n_{B1}Z = 0; \tag{5.29}$$

$$m_{S1}(Y - b\sin\gamma) + n_{S1}(Z - b\cos\gamma) = 0, \tag{5.30}$$

or equivalently,

$$A_1 Y + Z = 0; \quad (5.31)$$

$$A_2 (Y - b\sin\gamma) + (Z - b\cos\gamma) = 0, \quad (5.32)$$

where $A_1 = m_{B1}/n_{B1}$ and $A_2 = m_{S1}/n_{S1}$.

Solving Equations (5.31) and (5.32), we can obtain the following desired solution for Y and Z:

$$Y = \frac{A_2 \times b \times \sin\gamma + b \times \cos\gamma}{A_2 - A_1};$$

$$Z = -A_1 \times Y = A_2 \times b \times \sin\gamma - A_2 \times Y + b \times \cos\gamma. \quad (5.33)$$

It is noted that Equation (5.33) cannot be solved when A_1 is equal to A_2, resulting in a parallelism between the two space planes Q_3 and Q_4.

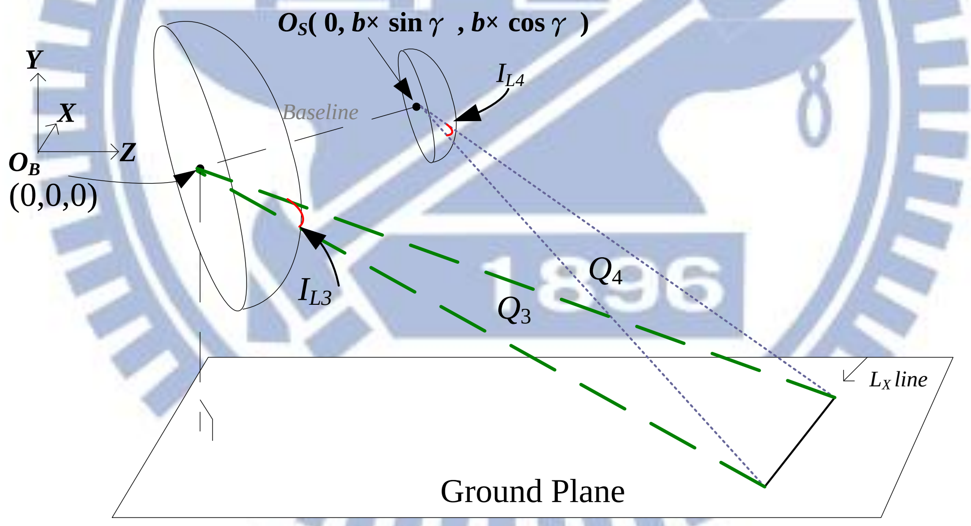


Figure 5.6 A space line projected onto I_{L3} and I_{L4} on two mirrors in the used two-mirror omni-camera.

(C) 3D data computation using a L_Z line.

As shown in Figure 5.7, the equations of Q_5 and Q_6 may be described as follows:

$$l_{B2}X + m_{B2}Y + n_{B2}Z = 0; \quad (5.34)$$

$$l_{S2}X + m_{S2}(Y - b\sin\gamma) + n_{S2}(Z - b\cos\gamma) = 0, \quad (5.35)$$

where (l_{B2}, m_{B2}, n_{B2}) represents the normal vector of Q_5 and (l_{S2}, m_{S2}, n_{S2}) represents that of Q_6.

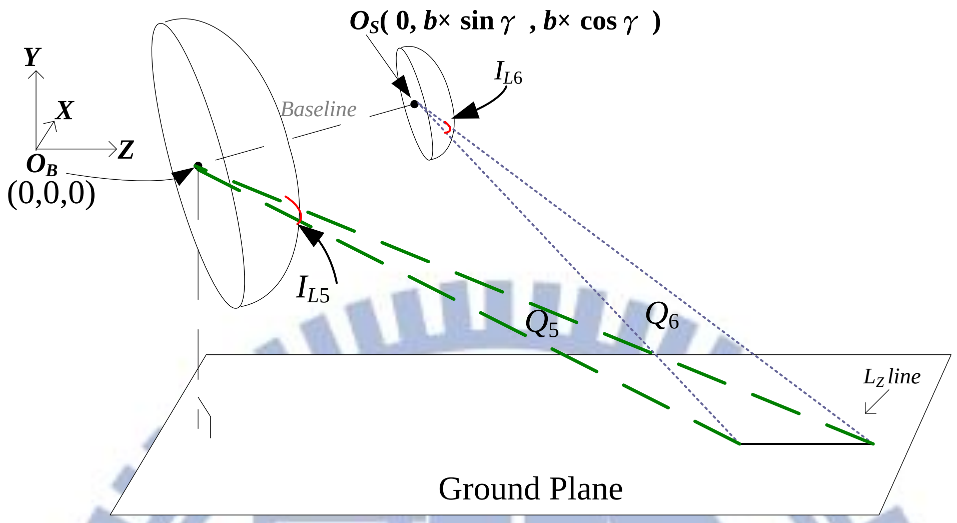


Figure 5.7 A space line projected onto I_{L5} and I_{L6} on two mirrors in the used two-mirror omni-camera.

The direction vector of the L_Z line is $D_Z = (d_{Zx}, d_{Zy}, d_{Zz}) = (0, 0, 1)$. Thus, we know that n_{B2} and n_{S2} are both zero. Thus, the above two space plane equations can be reduced into the following forms:

$$l_{B2}X + m_{B2}Y = 0; \tag{5.36}$$

$$l_{S2}X + m_{S2}(Y - b\sin\gamma) = 0 \tag{5.37}$$

which are equivalent to

$$K_1X + Y = 0; \tag{5.38}$$

$$K_2X + (Y - b\sin\gamma) = 0, \tag{5.39}$$

where $K_1 = l_{B2}/ m_{B2}$ and $K_2 = l_{S2}/ m_{S2}$. By solving Equations (5.38) and (5.39), we can obtain the following desired solution for X and Y:

$$X = \frac{b\times\sin\gamma}{K_2 - K_1} = \frac{b\times\sin\gamma - Y}{K_2};$$
$$Y = -K_1 \times X. \tag{5.40}$$

It is noted that Equation (5.40) cannot be solved when K_1 is equal to K_2, resulting in a parallelism between the two space planes Q_5 and Q_6.

5.3 Proposed Method for Tree Trunk Detection

In this section, we introduce the proposed method to localize a tree trunk. At first, we introduce the used method to describe a tree trunk contour and the learning of the tree trunk contour in Section 5.3.1. Next, we extract a feature of the tree trunk by moment-preserving thresholding [26] in Section 5.3.2. Then, estimating the position of tree trunk is described in Section 5.3.3. Finally, some experimental results for tree trunk detection by the proposed method are given in Section 5.3.4. The process of the tree trunk localization is illustrated in Figure 5.8.

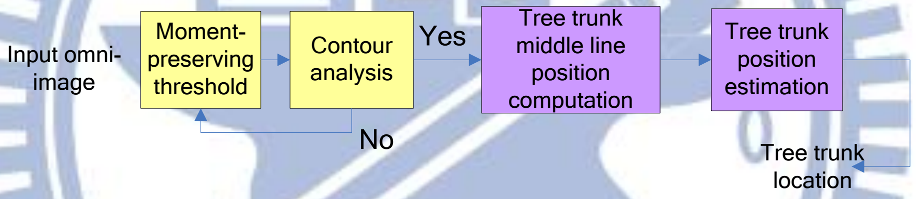

Input omni-image
Moment-preserving threshold
Contour analysis
Yes
No
Tree trunk middle line position computation
Tree trunk position estimation
Tree trunk location

Figure 5.8 Proposed method for tree trunk localization.

5.3.1 Tree trunk contour description

After conducting tree trunk segmentation in the omni-image, we want to ensure that the result of segmentation is correct. In this study, we use the center of a group of feature points and a simple description with two specific parameters obtained by *principal component analysis* (PCA) to ensure that the object which we want is existing. The method proposed is based on Chou and Tsai [23]. When it conducts the segmentation results, it utilizes the feature point to compute the covariance matrix C_x in the image. After obtaining the two eigenvalues and the two corresponding eigenvectors of the matrix C_x, we calculate the center of the feature points of the tree trunk, the length ratio η of the two eigenvalues of C_x, and the rotational angle ω

between the ICS and the principal component, respectively. Then, we utilize ω and η to describe the tree trunk contour as shown in Figure 5.9. The detailed algorithm is described as the follows.

Algorithm 5.3 ***Tree trunk contour parameter computation.***

Input: an input bi-level image I_{input} which includes the feature points of a tree trunk appearing in an omni-image.

Output: Three tree trunk contour parameters, the center m_x of all the feature points using their coordinates, a rotational angle ω, and a length ratio η.

Steps.

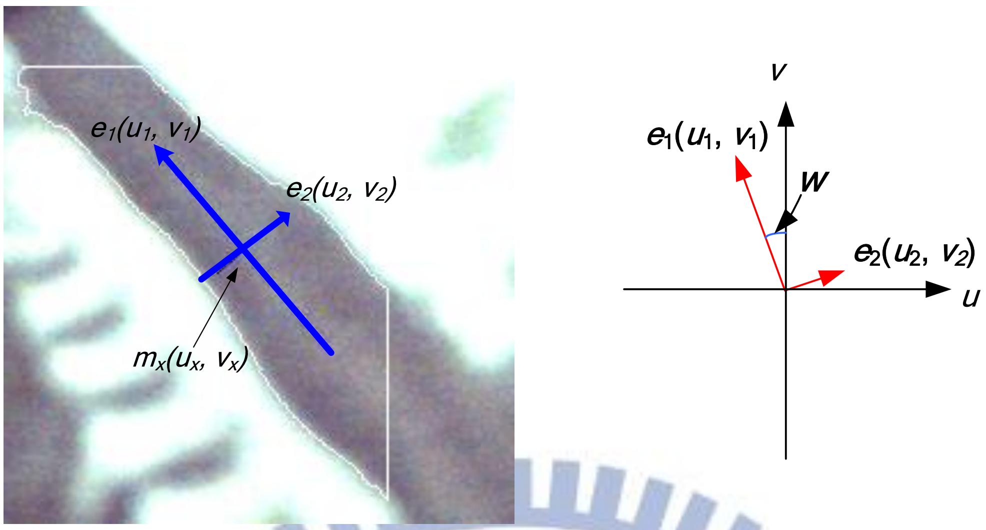
Step 1. Scan each feature point p with coordinates (u, v) in I_{input}, compute the center $m_x = (u_x, v_x)$ of all the feature points using their coordinates, and calculate the covariance matrix C_x of these feature points using their coordinates and m_x.

Step 2. Compute the eigenvectors $e_1 = (u_1, v_1)$ and $e_2 = (u_2, v_2)$ and the two corresponding eigenvalues λ_1 and λ_2 of matrix C_x, where e_1 represents the first principal component and e_2 the second.

Step 3. By the two eigenvectors e_1 and e_2, and the two eigenvalues λ_1 and λ_2, compute two parameters, the rotational angle ω of the first principal component e_1 with respect to the v-axis in the ICS and the ratio η of λ_1 to λ_2 by the following equations:

$$\omega = \tan^{-1}\left(\frac{v_1}{u_1}\right); \quad \eta = \frac{\lambda_1}{\lambda_2}. \tag{5.21}$$

Step 4. Take m_x, ω, and η as outputs.



(a) (b)

Figure 5.9 Principal component analysis for the tree trunk contour. (a) Illustrated principal components, e_1 and e_2, on the omni-image. (b) A rotation angle ω between the ICS and the computed principal components.

Additionally, because the vehicle cannot arrive at the same position to detect the tree trunk, the projections of the tree trunk on omni-images cause a little difference of their positions. To solve this problem, we guide the vehicle to take many of omni-images from different positions and directions in the learning process. After conducting the process of extracting the tree trunk feature points, we use the images to compute three parameters ω_i, η_i, and m_x, the center of the entire group of feature points. Collecting all of parameters, we set the tree trunk parameters in a range from a minimum angle $\omega_{\min}$ to a maximum angle $\omega_{\max}$, and set a range from a minimum ratio $\eta_{\min}$ to a maximum ratio $\eta_{\max}$. Also, we calculate the average of the center of tree trunk, and then set a threshold value V_{height} in the ICS to check that the height of the center of tree trunk is true. Finally, we record the five parameters V_{height}, $\omega_{\min}$, $\omega_{\max}$, $\eta_{\min}$, and $\eta_{\max}$ as the *tree contour thresholds*. In the navigation process, if the computed the height of the center of the tree trunk, the rotational angle ω, and the ratio η in tree truck detection are not in the learned ranges, we decide that the result of detection is not a pre-selected tree trunk.

5.3.2 Moment-preserving thresholding for tree trunk segmentation

In outdoor environments, it is difficult to detect the natural landmark like trees because the natural landmark does not have obvious color information. So we should find a strategy to detect it. In this section, we utilize the moment-preserving thresholding technique to extract feature points of the tree trunk.

In outdoor environments, varying lighting conditions will influence the image intensity. The moment-preserving thresholding approach to automatic threshold selection for segmenting desired object out of a given image is adopted in this study. We segment the tree trunk from an omni-image by thresholding the difference image *f* into a bi-level image by the use of a threshold value *Th*. The details are described in the following.

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

$$m_i = \frac{1}{n}\sum_x \sum_y f^i(x,y), \quad i = 0, 1, 2, 3. \tag{5.41}$$

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

$$m_i' = \frac{1}{n}\sum_{j=0}^{1} n_j (z_j)^i = \sum_{j=0}^{1} p_j (z_j)^i, \quad i = 0, 1, 2, 3. \tag{5.42}$$

Assume that the image resulting from thresholding only contains two gray values z_0 and z_1, with z_1 being larger than z_0. The method is to choose a threshold value to judge the pixel value in the gray-level in the environment window to be z_o or z_1. To find the desired *Th*, we can solve two equations described by Equations (5.41) and (5.42) to obtain p_0 and p_1, as described in the following:

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

$$z_0 = \frac{1}{2}\left[-c_1 - (c_1^2 - 4c_0)^{\frac{1}{2}}\right]; \; z_1 = \frac{1}{2}\left[-c_1 + (c_1^2 - 4c_0)^{\frac{1}{2}}\right]; \tag{5.43}$$

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

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

$$p_0 = \frac{1}{n}\sum_{z_j \le t} n_j. \tag{5.44}$$

Finally, conducting the thresholding work, all pixels in the environment window in the image *f* are scanned and their values are compared to the threshold value *Th*. If a pixel value in *f* is larger than *Th*, the corresponding pixel in the bi-level image *b* is labeled by “0”; else, it is labeled by “1.” We regard the region labeled by “0” as a tree trunk region.

5.3.3 Tree trunk localization

By using the tree trunk contour extraction process described above, we want to find the vertical axis line of the tree trunk to localize the tree trunk. We assume that the desired vertical line goes through both centers of the tree trunk appearing in the regions of *Mirrors S* and *B* in the omni-image. After extracting the two center positions of the tree trunk in the regions of *Mirrors S* and *B*, we can obtain further the two space planes which go through the axis line and the two mirror centers, respectively, by the use of the proposed vertical line detection method [23]. Finally, we can obtain the tree trunk position by the located axis line using the information of the two space planes. The detailed process is described as follows.

Algorithm 5.4 ***Tree trunk location computation.***

Input: an input bi-level image I_{bi} which includes tree trunk feature points, and an environment window Win_{tt}.

Output: a tree trunk position G_{tt} in the CCS.

Steps.

Step 1. Compute the center C_B with coordinates (u_B, u_B) of the tree trunk feature points in win_B of Win_{tt} and the center C_S with coordinates (u_S, u_S) of the tree trunk feature points in win_S of Win_{tt}.

Step 2. Look up the pano-mapping table to obtain the corresponding elevation angle α_B and azimuth angle θ_B of C_B and the corresponding elevation angle α_S and azimuth angle θ_S of C_S.

Step 3. By Equation (5.13), compute the parameter value B_B corresponding to C_B using θ_B and α_B as well as the parameter value B_S corresponding to C_S using θ_S and α_S.

Step 4. By the use of B_B and B_S, compute the position coordinates X and Z of the axis line L of the tree trunk by Equation (5.26).

Step 5. Compute the tree trunk position G_{tt} with coordinates (x_{tt}, y_{tt}, z_{tt}) in the CCS as follows:

$$x_{tt} = X;\ y_{tt} = -H,\ z_{tt} = Z \qquad (5.45)$$

where H is the height of the camera center.

Step 6. Take G_{tt} as output.

5.3.4 Experimental results for tree trunk detection

Some experimental results for tree trunk detection are shown in this section. The input image with a tree trunk on the regions of *Mirrors S* and *B*, respectively, is shown in Figure 5.10. The result of tree trunk segmentation using the moment-preserving threshold technique is shown in Figure 5.11. Finally, the result of detecting the vertical axis line of the tree trunk and the obtained tree trunk position are shown in Figure 5.12.

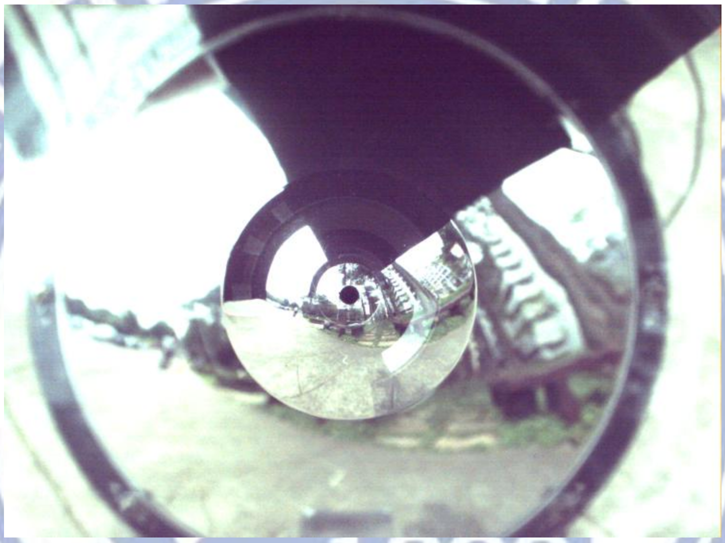


Figure 5.10 The input image with a tree trunk.

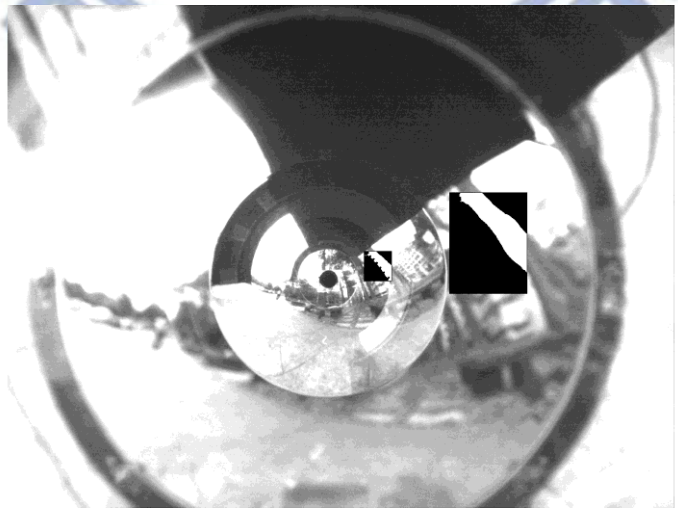
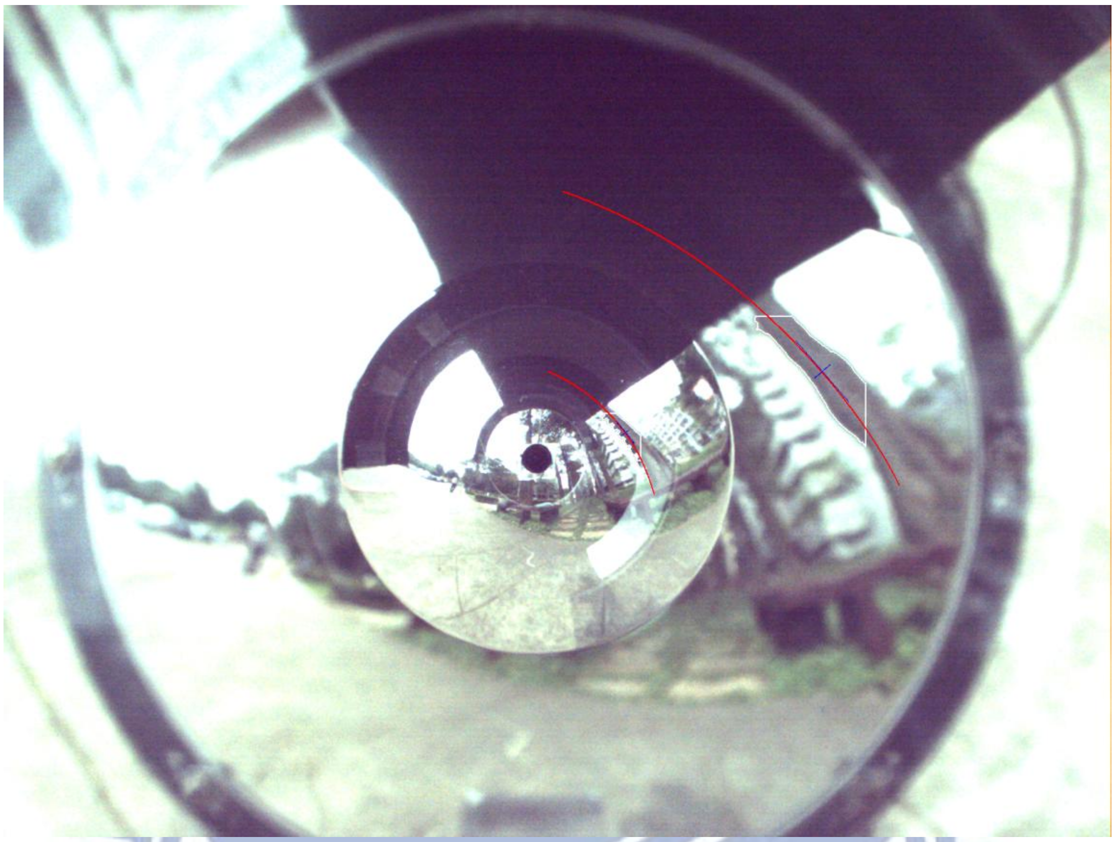
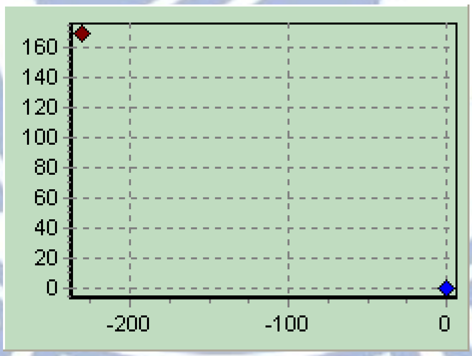


Figure 5.11 Tree trunk segmentation by moment-preserving thresholding.



(a)



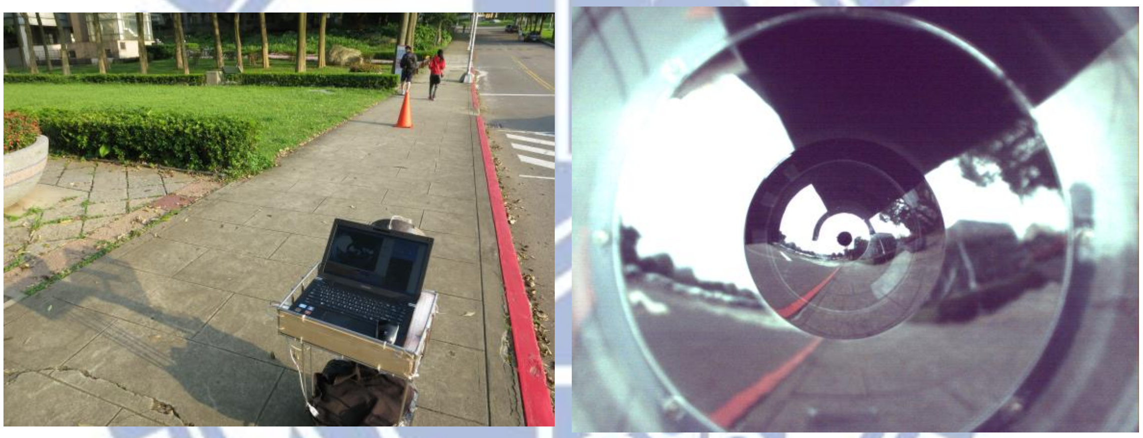
(b)

Figure 5.12 The result of tree truck detection and its position. (a) The result image of extracting the vertical axis line of the tree trunk. (b) The related tree truck position with respect to the vehicle position.

5.4 Proposed Method for Lawn Corner Detection

5.4.1 Lawn corner detection and localization

In this section we introduce the proposed method for lawn corner detection. As shown in Figure 5.13, the lawn corner is too obscure to recognize in the omni-image, so we can only see the part of the lawn corner By these observations, the detection is divided into two stages. When the vehicle arrives at the detection position, it detects a space line firstly. Then, it is guided to turn left, and detects another space line. The two space lines then are drawn to cross to form a corner. In this way, we can obtain the 3D data of the lawn corner successfully.



(a) (b)

Figure 5.13 Two different camera systems take the lawn corner. (a) By digital camera. (b) By omni-camera.

Generally speaking, the lawn has the special color of green. We utilize this color information to extract the lawn boundary. Unfortunately, the grove is in front of the lawn as shown in Figure 5.13. It is too dark to obtain the color information. By the use of image intensity difference between the lawn and the floor, we can detect an L_X space line on the ground instead of detecting the lawn object. First, we detect the lawn object using the moment-preserving thresholding technique as described previously in Section 5.3.2. Then, we obtain the boundary line between the lawn and the sidewalk using the Canny edge detection technique. By the use of the edge-point image, we use the above-mentioned space line detection technique to find an L_X space line on the ground. The detailed algorithm for implementing this idea of lawn corner detection is described as follows.

Algorithm 5.5 ***lawn boundary line detection.***

Input: an input image I_{input}, a pano-mapping table for *Mirrors B*, and a set of environment windows Win_{LC}.

Output: the parameters $A_{\max}$ representing the parameters of space planes through the boundary lines of the lawn and then through the *Mirror B* center

Steps.

Step 1. For I_{input} in the environment window Win_{LC}, use the moment-preserving threshold to find the lawn object, and obtain a bi-level image I_{bi}.

Step 2. For I_{bi} in the environment window Win_{LC}, use the Canny edge detector to conduct edge detection to extract the feature points of the boundary lines of the lawn, and obtain an edge-point image I_{edge}.

Step 3. Set a 1D space S with parameter A and initialize all cell counts to be zero.

Step 4. For each edge point I at coordinates (u, v) in win_B of Win_{LC}, look up the pano-mapping table to obtain an azimuth angle θ and an elevation angle α.

Step 5. Compute A by Equation (5.16) using θ and α, and increment by 1 the value of the cell with parameter A in S.

Step 6. Repeat Steps 4 and 5 until all edge points in win_B of Win_{LC} are computed.

Step 7. Find the cell, denoted as $A_{\max}$, with the maximum value in space S

After conducting the boundary of detection, we can obtain the space plane which goes through the boundary line on the ground. Then, the vehicle should turn left to detect another L_X space line. After we get two corresponding space plane parameters A_{first} and A_{second}, we utilize the results to compute the location of the lawn corner. The detailed algorithm is introduced as follows.

Algorithm 5.6 ***Lawn corner position computation.***

Input: two corresponding space plane parameters A_{first} and A_{second} obtained from Algorithm 5.4, of a lawn appearing in an omni-image and the height of ground H.

Output: a lawn position G_{LC} in the CCS.

Steps.

Step 1. By A_{first} and H, compute one boundary space line L_1 of the lawn by Equation (5.23) and obtain its equation as follows:

$$Y = -H;\quad Z = Z_1. \tag{5.46}$$

Step 2. By A_{second} and H, compute one boundary space line L_2 of the lawn by Equation (5.23) and obtain its equation as follows:

$$\mathrm{Y} = -H;\quad Z = X_1. \tag{5.47}$$

Step 3. Compute the coordinates (x_{LC}, y_{LC}, z_{LC}) of the lawn corner position G_{LC} in the CCS as follows:

$$X_{LC} = X_1;\quad y_{LC} = -H;\quad z_{LC} = Z_1 \tag{5.48}$$

where H is the height of the camera center.

Step 4. take G_{LC} as output.

5.4.2 Experimental results for lawn corner detection

An input image with the projection of a lawn corner on the regions of *Mirror B* is shown in Figure 5.14. The result of the lawn boundary detection is shown in Figure 5.15.

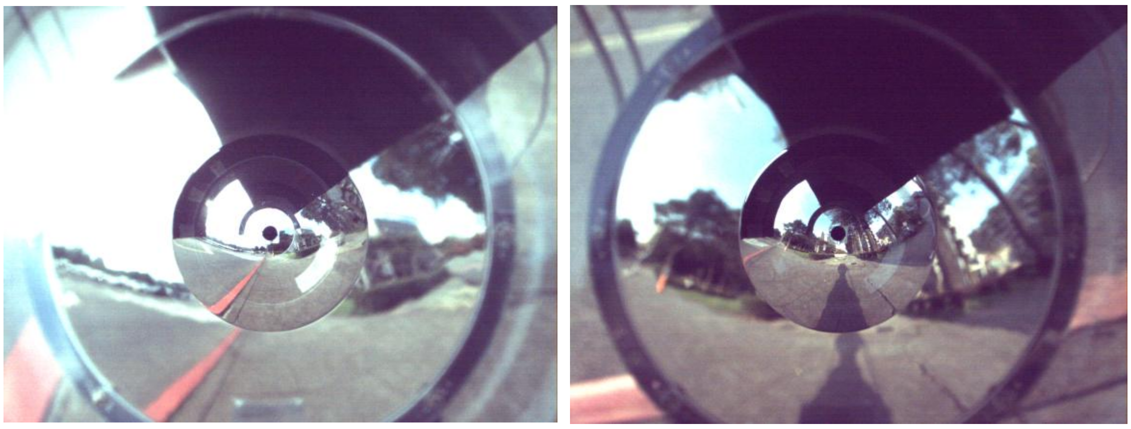


Figure 5.14 Two different directions in the image for lawn corner detection.

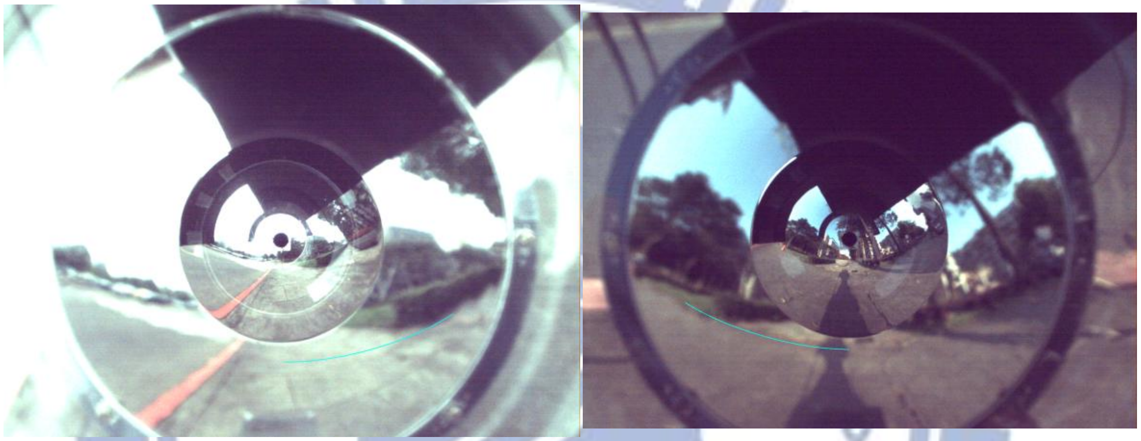


Figure 5.15 Two result of L_X line detection.

Chapter 6
Artificial Landmark Detection in Images Using Space Line Detection Technique

6.1 Proposed Technique for Curb Line Following

To conduct vehicle navigations on sidewalks, Chou and Tsai [23] proposed a technique to detect curb lines and compute their locations with respect to the vehicle. In this section, based on the method proposed by Chou and Tsai, we propose a new method to localize the curb line by the use of the projection of a curb line onto the region of *Mirror B* in the omni-image.

In this section, we review the method [23] in Section 6.1.1. In Section 6.1.2, the curb line localization method we propose is introduced. Finally, some experimental results for curb detection are shown in Section 6.1.3.

6.1.1 Review of adopted curb line following

Chou and Tsai [23] proposed a method for curb line detection. By the use of an environment window, a curb line segment can be detected in the window. First, the curb line color feature is extracted by the use of the HSI color model. Then, some morphological process including erosion and dilation operations are performed to eliminate small noise. Within the bi-level image I_b which includes the curb feature points, the *inner* boundary points of the curb line are found and their positions in the

CCS are computed. The property that the curb line is on the floor is utilized to compute the boundary point positions instead of using the space-mapping method. By the use of *Mirror B*, suppose that a space point G at coordinates (X, Y, Z) is projected onto the omni-image with an azimuth angle θ and an elevation angle α. As mentioned in Section 5.2.1, we can represent the vector from the mirror center O_B to space point G using the following equation:

$$V_G = \begin{bmatrix} G_x \\ G_y \\ G_z \end{bmatrix} = \begin{bmatrix} \cos\alpha \times \cos\theta \\ \cos\alpha \times \sin\theta \times \cos\gamma + \sin\alpha \times \sin\gamma \\ -\cos\alpha \times \sin\theta + \sin\alpha \times \cos\gamma \end{bmatrix}. \tag{5.3}$$

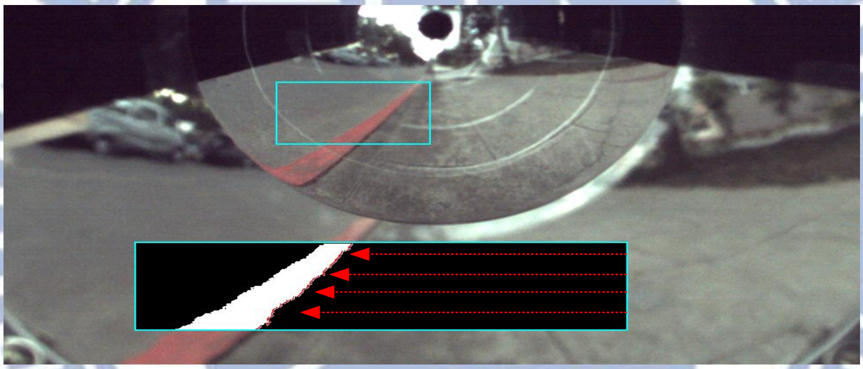


Figure 6.1 A detected curb line and the inner boundary points of the curb line on the omni-image.

Besides, according to the knowledge that the height H of the center of *Mirror B* is known in advance, one can get $Y = -H$. Accordingly, Equation (5.3) can be rewritten as follows, which describe the position of P:

$$X = \frac{-H \times (\cos\alpha \times \cos\theta)}{(\cos\alpha \times \sin\theta \times \cos\gamma + \sin\alpha \times \sin\gamma)};$$

$$Y = -H;$$

$$Z = \frac{-H \times (-\cos\alpha \times \sin\theta + \sin\alpha \times \cos\gamma)}{(\cos\alpha \times \sin\theta \times \cos\gamma + \sin\alpha \times \sin\gamma)}. \tag{6.1}$$

After calculating the position of the curb line boundary point in the CCS, it can be transformed from the CCS into the VCS. Then, the data are used to fit the line *L* to get its equation as follows:

$$Y = ax + b, \tag{6.2}$$

where the two parameters, *a* and *b*, are calculated by the following equations:

$$a = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2},$$
$$b = \frac{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i y_i \sum_{i=1}^{n} x_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \tag{6.3}$$

with (x_i, y_i) being the position coordinates of a boundary point.

Furthermore, a more precise position of the curb line can be estimated using the dynamic color thresholding technique by adjusting the saturation threshold in a pre-defined fixed range. After using all possible pre-selected threshold values in this range to extract curb boundary points, the best saturation threshold value can be estimated according to the minimum sum of errors in the results of fitting the curb boundary points with the computed line. A result of applying dynamic color thresholding technique is shown in Figure 6.2.

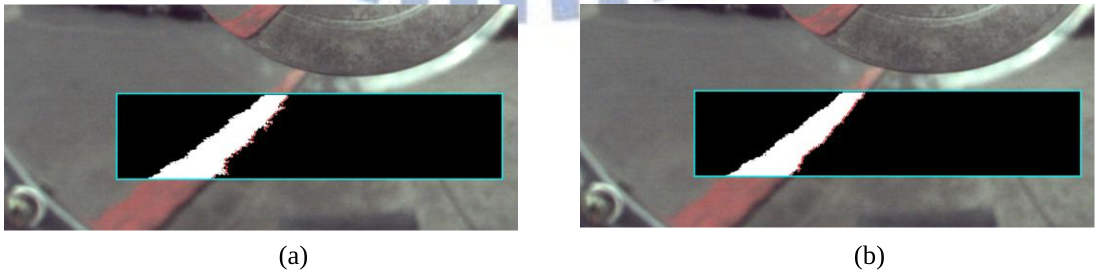

(a) (b)

Figure 6.2 Two images of curb segmentation resulting from using different threshold values (a) The segmentation result with original threshold value. (b) The segmentation result image by dynamic thresholding.

Finally, the slope angle of L and the distance d to the vehicle can be computed by the following equation:

$$\theta = \tan^{-1}\left(\frac{1}{a}\right); \quad d = \frac{|b|}{\sqrt{1+a^2}}\,. \tag{6.4}$$

6.1.2 Proposed curb line localization technique

In this section we describe the proposed method of curb line localization. As usual we get the feature points of the curb line by color information. Then, we use the Canny edge detector to obtain the two boundary lines of the curb line. In the resulting edge-point image, we use the proposed L_Z line detection method to find the two boundary lines based on a 1D Hough transform technique. Then, we choose the inner boundary line to compute its position. The proposed algorithm for curb line detection as discussed above is described as follows.

Algorithm 6.1 ***Curb line boundary detection.***

Input: a bi-level image I_{input} of the feature points of a curb line, a pano-mapping table for *Mirrors B*, and an environment window Win_{RL}.

Output: the parameters K_{inner} representing the space planes going through the inner boundary lines of the curb line and the *Mirror B* center.

Steps.

Step 1. For I_{input} in the environment window Win_{RL}, use the Canny edge detector to conduct edge detection to extract the feature points of the boundary lines of the lawn, and obtain an edge-point image I_{edge}.

Step 2. Set up a 1D space S with parameters K and initialize all cell counts to be zero.

Step 3. For each edge point I at coordinates (u, v) in win_B of Win_{RL}, look up the pano-mapping table to obtain an azimuth θ and an elevation angle α.

Step 4. Compute K by Equation (5.19) using θ and α, and increment by 1 the value of the cell with parameter K in S.

Step 5. Repeat Steps 3 and 4 until all edge points in win_B of Win_{RL} are computed.

Step 6. Find the cell, denoted as K_{inner}, with the maximum value in space S.

Step 7. Take K_{inner} as the output.

After successfully detecting the inner boundary line of a curb line, we can use it to compute the inner curb line location. By the parameter K_{inner} obtained by the above algorithm and the height H of the center of *Mirror B*, we can rewrite Equation (5.40) to obtain the position of the inner curb line L as follows:

$$X = \frac{H}{K_{inner}},$$
$$Y = -H. \qquad (6.5)$$

6.1.3 Experimental results of curb line detection

Some experimental results of curb detection using the proposed localization method are given in this section. An input omni-image with a curb line is shown in Figure 6.3. The extracted curb boundary points and computed best-fit line from Figure 6.3 are shown in Figure 6.4. The method for localizing the curb line using a 1D Hough transform is shown in Figure 6.5. Besides, we collected the statistics of our experimental results for curb detection as shown in Table 6.1. Comparing the method we propose with that by Chou and Tsai [23], we see that the error ratios of our method are more stable and smaller. By this result, we utilize the fitting line method to compute the slope angle of curb line and use a 1D Hough transform to localize the curb line in the navigation process.

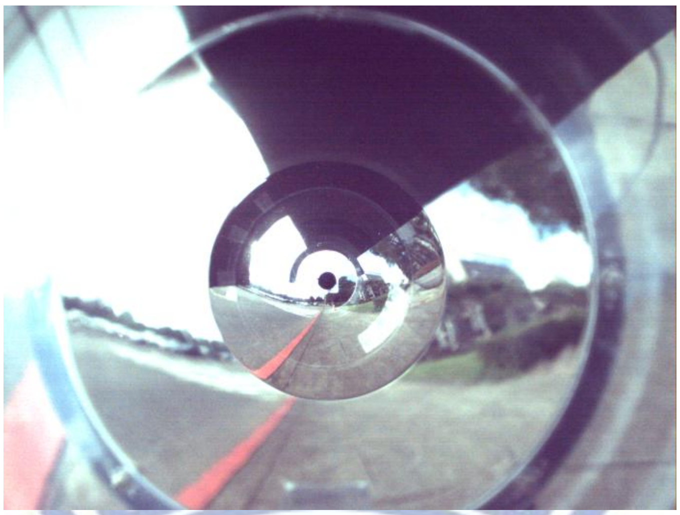
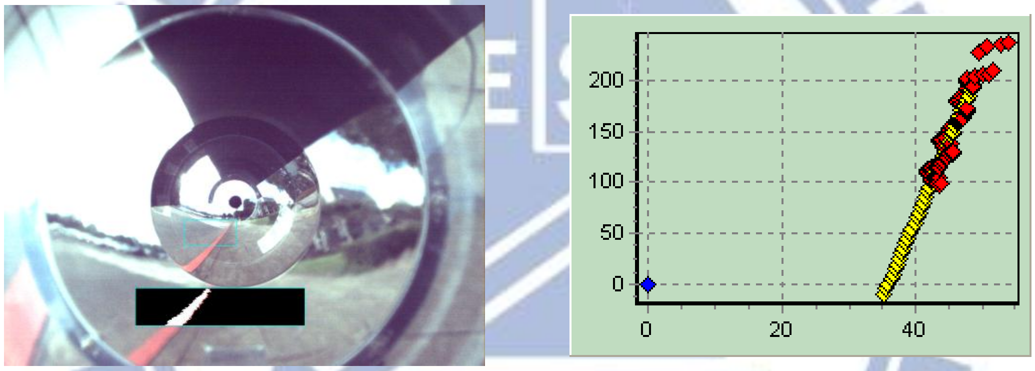
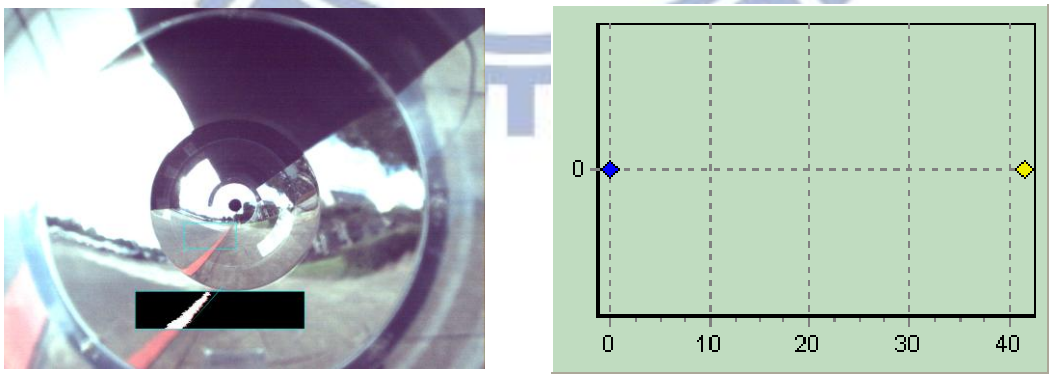


Figure 6.3 An omni-image with curb line landmark.



(a) (b)

Figure 6.4 The curb line detection method proposed by Chou and Tsai (a) the curb line segmentation result. (b) Illustration of extracted curb boundary points and a fitting line.



(a) (b)

Figure 6.5 The curb line detection method (a) the curb line segmentation result. (b) The related curb line position with respect to the vehicle position.

Table 6.1 Comparison of two methods for curb line detection.

Distance	Curb line fitting method	1-D Hough method	Error ratio in Curb line fitting	Error ratio in 1-D Hough
40	38	43	5.00%	7.50%
40	39	43	2.50%	7.50%
40	38	43	5.00%	7.50%
40	38	43	5.00%	7.50%
40	39	43	2.50%	7.50%
40	38	45	5.00%	12.50%
40	37	43	7.50%	7.50%
40	37	43	7.50%	7.50%
40	39	45	2.50%	12.50%
40	38	43	5.00%	7.50%
50	49	53	2.00%	6.00%
50	50	53	0.00%	6.00%
50	48	54	4.00%	8.00%
50	48	54	4.00%	8.00%
50	49	54	2.00%	8.00%
50	48	53	4.00%	6.00%
50	48	53	4.00%	6.00%
50	47	53	6.00%	6.00%
50	49	53	2.00%	6.00%
50	48	53	4.00%	6.00%
60	57	62	5.00%	3.33%
60	58	63	3.33%	5.00%
60	57	63	5.00%	5.00%
60	56	63	6.67%	5.00%
60	57	64	5.00%	6.67%
60	56	64	6.67%	6.67%
60	58	64	3.33%	6.67%
60	58	63	3.33%	5.00%
60	56	63	6.67%	5.00%
60	58	64	3.33%	6.67%
70	59	71	15.71%	1.43%
70	54	71	22.86%	1.43%
70	56	71	20.00%	1.43%
70	55	70	21.43%	0.00%
70	56	71	20.00%	1.43%
70	54	71	22.86%	1.43%
70	65	71	7.14%	1.43%
70	54	71	22.86%	1.43%
70	54	70	22.86%	0.00%
70	54	71	22.86%	1.43%
Average			8.16%	5.44%

6.2 Proposed Method for Signboard Detection

6.2.1 Signboard detection

The idea of the proposed method for signboard detection is to detect the contour of the signboard, like we do in detecting the tree trunk. Then, we use the boundary line of the signboard to localize it. Combining the two techniques, we can localize the signboard more precisely. The entire process to localize a signboard is shown in Figure 6.6. The computation of the signboard position is described in the next section.

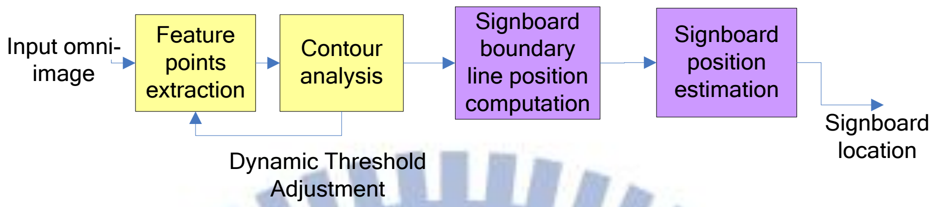

Input omni-image
Feature points extraction
Contour analysis
Signboard boundary line position computation
Signboard position estimation
Signboard location
Dynamic Threshold Adjustment

Figure 6.6 Proposed method of signboard localization.

Due to the obvious color of the signboard, we use the HSI color model to extract the signboard from an image. In order to handle the varying lighting condition in the outdoor environment, we utilize the hue and saturation values to obtain the signboard feature. The transformation of a color (*R*, *G*, *B*) in the RGB color space into a corresponding color (*H*, *S*, *I*) in the HSI color space is as follows:

$$H = \begin{cases} \theta, & if \;\; B <= G \\ 360-\theta, & if \;\; B < G \end{cases};$$

$$S = 1 - \frac{3}{(R+G+B)}[\min(R,G,B)];$$

$$I = \frac{1}{3}(R+G+B), \qquad (6.6)$$

where

$$\theta = \cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{1/2}}\right\}.$$

We define two hue values, donated as H_{min} and H_{max}, as the hue threshold values for extracting the blue feature of the signboard. We also define a saturation threshold value to choose the feature of the signboard. These parameters are used together to classify the signboard feature points.

Besides, varying lighting conditions will influence the hue and saturation features. Based on the learned signboard contour, we propose a dynamic color thresholding scheme to adjust the saturation threshold value of S_{th} in a fixed range [S_0, S_1], where S_0 and S_1 are learned in advance in different lighting conditions in the learning stage. Detailed algorithm to extract signboard feature points is described as follows.

Algorithm 6.2 ***Signboard detection by dynamic thresholding.***

Input: an input image I_{input} including a signboard; the learned five signboard contour parameters, m_{th}, ω_{min}, ω_{max}, η_{min}, and η_{max}; two hue threshold values H_{min} and H_{max}; a saturation threshold S_{th}; and a set of environment windows Win_{SB}.

Output: a bi-level image I_{bi} with feature points of the signboard, and an adjusted saturation threshold S_{th}.

Steps.

Step 1. Initialize an empty bi-level image I_{bi} for labeling feature points and set all pixel values as zero.

Step 2. Scan each pixel I_{uv} with coordinates (u, v) in Win_{SB}, compute its hue value h_{uv} and saturation value s_{uv} by Equation (6.6), and if h_{uv} is between H_{min} and H_{max} and s_{uv} is larger than S_{th}, then label I_{uv} by “1” in I_{bi}.

Step 3. Apply erosion and dilation operations to the bi-level image I_{bi}.

Step 4. Conduct connected component labeling, and find a maximum connected component M in I_{bi}.

Step 5. Apply Algorithm 5.3 to M in I_{bi} to obtain three contour parameters, the center of feature points $m_x = (u_x, v_x)$, the rotational angle ω and the length ratio η of M.

Step 6. If $\mathrm{v}_{\mathrm{th}} < \mathrm{v}_{\mathrm{x}}$, $\omega_{min} < \omega < \omega_{max}$ and $\eta_{min} < \eta < \eta_{max}$, then take M in I_{bi} and S_{th} as outputs; else, adjust the threshold S_{th} in the range [S_0, S_1] and go to Step 1.

6.2.2 Signboard position computation

In this section, we describe how to compute the signboard position. The method proposed is based on Chou and Tsai [23]. We utilize morphological operations including erosion and set difference to conduct boundary extraction. Then, we use the L_Y line detection method to find one vertical boundary line based on a 1D Hough transform technique. The detailed algorithm is described as follows.

Algorithm 6.3 ***Signboard boundary line detection.***

Input: an input image I_{input}, two pano-mapping tables for *Mirrors S* and *B*, and an environment window Win_{SB}.

Output: two parameters B_{B1} and B_{S1} representing the two space planes going through the boundary line of the signboard and the *Mirror B* center and the *Mirror S* center, respectively.

Steps.

Step 1. For I_{input}, use morphological operations to conduct edge detection to extract the feature points of the boundary lines of the signboard, and obtain an edge-point image I_{edge}.

Step 2. Set a 1D space S with parameter B and initialize all cell counts to be zero.

Step 3. For each edge point I at coordinates (u, v) in win_B of Win_{SB}, look up the pano-mapping table to obtain an azimuth θ and an elevation angle α.

Step 4. Compute B by Equation (5.13) using θ and α, and increment by 1 the value of the cell with parameter B in S.

Step 5. Repeat Steps 3 and 4 until all edge points in win_B of Win_{SB} are computed.

Step 6. Find the cell, denoted as B_{B1}, with the maximum value in space S

Step 7. Take B_{B1} as the output.

Step 8. In the same way, repeat Steps 2 through 7 in win_S of Win_{SB} for *Mirror S* and take the obtained two corresponding parameters B_{S1} as outputs.

After we obtain the space plane which goes through its boundary line, we use the resulting two parameters to compute the signboard location. In addition, we also can localize the middle line position to check whether the distance is close to the known width of the signboard. The detailed algorithm is described in the following.

Algorithm 6.4 ***Signboard position computation.***

Input: two corresponding space plane parameters B_{B1} and B_{S1} obtained from Algorithm 6.3, of a signboard appearing in an omni-image.

Output: the signboard position G_{SB} in the CCS.

Steps.

Step 1. By B_{B1} and B_{S1}, compute one boundary space line L_1 of the signboard by Equation (5.26) and obtain its equation as follows:

$$X = X_1; \quad Z = Z_1. \tag{6.7}$$

Step 2. Apply Algorithm 5.4 to compute the middle line signboard location and obtain its equation as follows:

$$X = X_2;\ Y = -H,\ Z = Z_2. \tag{6.8}$$

Step 3. Compute the distance d between the two lines by the following equation:

$$d = \sqrt{(X_1 - X_2)^2 + (Z_1 - Z_2)^2}\,. \tag{6.9}$$

Step 4. If $|d - D_{diameter}| \leqq Th_D$ where $D_{diameter}$ represents the pre-measured width of the signboard and Th_D is a pre-defined threshold, then go to Step 5; else, show a message saying that there is no signboard and exit.

Step 5. Compute the coordinates (x_{SB}, y_{SB}, z_{SB}) of the signboard position G_{SB} in the

CCS as follows:

$$X_{SB} = X_1; \quad y_{SB} = -H; \quad z_{SB} = Z_1. \tag{6.10}$$

where H is the height of the camera center.

Step 6. Take G_{SB} as the output.

6.2.3 Experimental results of signboard detection

Some experimental results for signboard detection are shown in this section. The input image with a signboard in the regions of *Mirrors S* and *B*, respectively, is shown in Figure 6.7. The result of signboard segmentation using the initial threshold values is shown in Figure 6.8(a). Next, the result of signboard segmentation by dynamic thresholding is shown in Figure 6.8(b). Finally, the result of detecting the vertical axis line of the signboard and the obtained signboard position are shown in Figure 6.9.

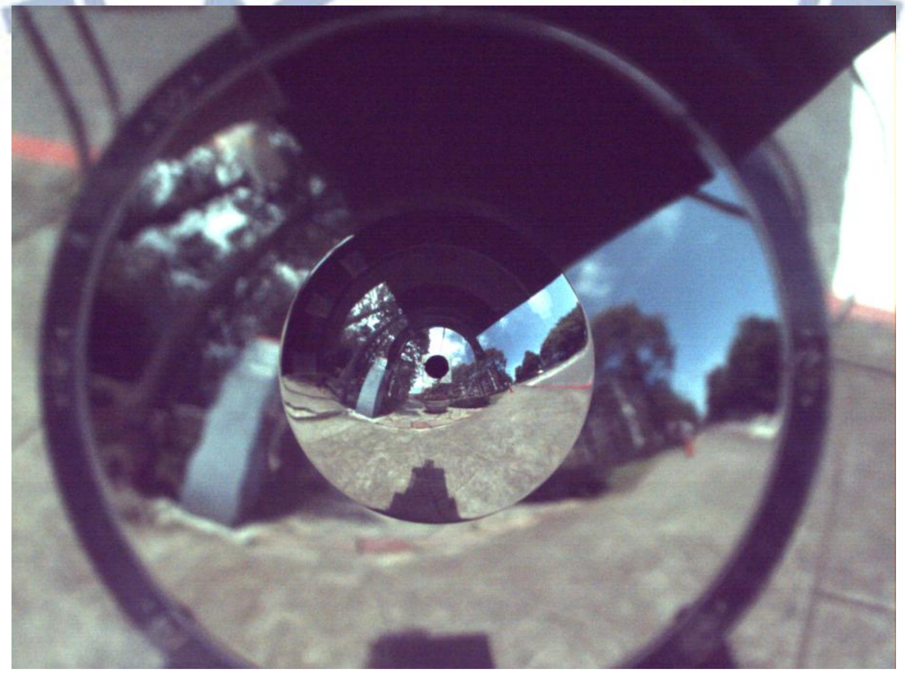
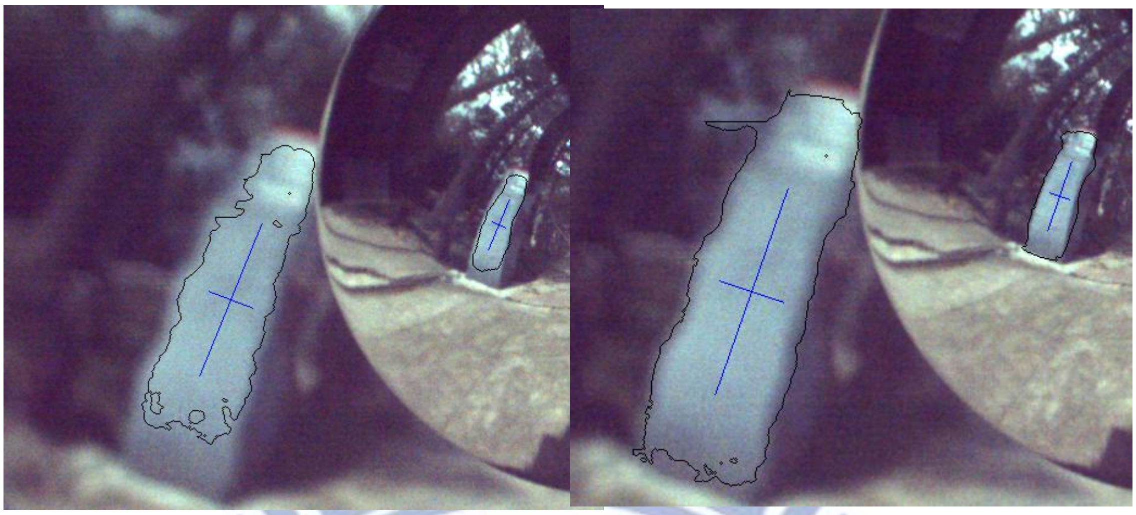
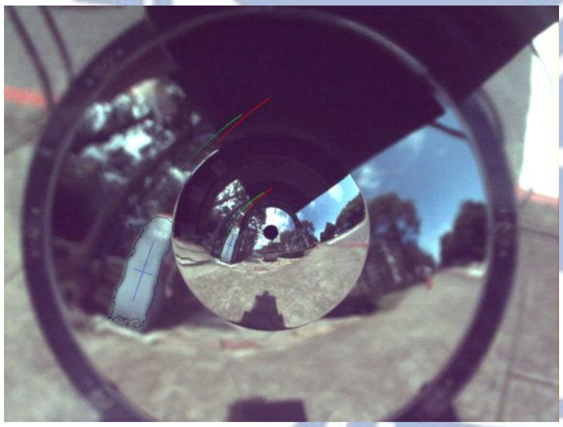


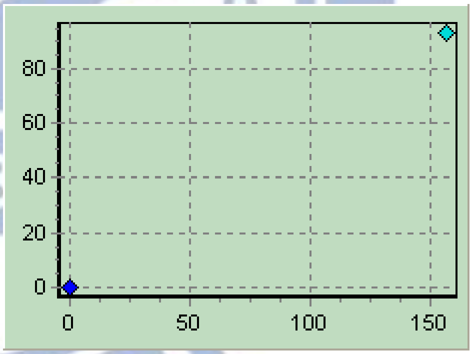
Figure 6.7 The omni-image with a signboard.



(a) (b)

Figure 6.8 Two result images of signboard segmentation with different threshold values (a) The result of signboard segmentation with original threshold value. (b) The result image of signboard segmentation by dynamic thresholding.





(a) (b)

Figure 6.9 The result of signboard detection and obtained signboard position. (a) The result image of extracting the L_Y line of the signboard (b) The related signboard position with respect to the vehicle position.

6.3 Proposed Method for Detecting Stop Lines on Roads

It is advantageous for the vehicle to be able to detect the landmark and calibrate its odometer. Except for the landmarks on the *sidewalk*, we also can use those landmarks on the *road path* for vehicle localization. There are some landmarks which are usually seen on the road. One of them is the stop line. In this section, we introduce the method we propose in this study for detecting a stop line on a road and localizing its position.

6.3.1 Detection and localization of stop lines on roads

As shown in Figure 3.2(b), because the stop line on roads is artificial, it has the obvious color information. We can utilize it to extract its boundary lines. Specifically, we also use the HSI color model to extract the feature points of the stop line on roads. Then, we utilize the Canny edge detector to detect the boundary line. Finally, we detect one L_Z line and two L_X lines to obtain the entire boundary information. A detailed algorithm for implementing the above idea is described as follows.

Algorithm 6.5 ***Detection of boundary lines of stop lines on roads.***

Input: a bi-level image I_{input} which includes the feature points of a stop line on roads appearing in an omni-image, two pano-mapping tables for *Mirrors S* and *B*, and an environment window Win_{WL}.

Output: six parameters A_{B1}, A_{B2}, A_{S1}, A_{S1}, K_{B1},and K_{S1} representing the parameters of six space planes through the boundary line of the stop line and the *Mirror B* center and the *Mirror S* center, respectively.

Steps.

Step 1. For I_{input} in the environment window Win_{WL}, use the Canny edge detector to conduct edge detection to extract the feature points of the boundary lines of the stop line, and obtain an edge-point image I_{edge}.

Step 2. Set up a 1D Hough space S_1 with parameter A, and initialize all cell counts to be zero.

Step 3. For each edge point I at coordinates (u, v) in win_B of Win_{WL}, look up the pano-mapping table to obtain the corresponding azimuth angle θ and elevation angle α.

Step 4. Compute A by Equation (5.16) using θ and α, and increment by 1 the value of the cell with parameter B in S_1.

Step 5. Repeat Steps 3 and 4 until all edge points in win_B of Win_{SB} are computed.

Step 6. Find the two cells, denoted as A_{B1} and A_{B2}, with two maximum values in space S_1

Step 7. Set a 1D Hough space S_2 with parameter K, and initialize all cell counts to be be zero.

Step 8. For each edge point I at coordinates (u, v) in win_B of Win_{WL}, look up the pano-mapping table to obtain the corresponding azimuth angle θ and elevation angle α.

Step 9. Compute K by Equation (5.19) using θ and α, and increment by 1 the value of the cell with parameter K in S_2.

Step 10. Repeat Steps 8 and 9 until all edge points in win_B of Win_{SB} are computed.

Step 11. Find the cells, denoted as K_{B1}, with the maximum values in space S_2

Step 12. Take A_{B1}, $A_{B2,}$ and K_{B1} as outputs.

Step 13. In the same way, repeat Steps 2 through 12 in win_S of Win_{SB} for *Mirror S* and take the obtained three corresponding parameters A_{S1}, A_{S2}, and K_{S1} as outputs.

By the use of the six known corresponding space planes obtained in the above algorithm, we can compute two corners of the stop line on, denoted as C_{in} and C_{out}, respectively, in the CCS as illustrated in Figure 6.10. Next, we check whether the distance between C_{in} and C_{out} is close to the known width of the stop line. If not, we assume that the detected two L_X lines are not the boundary lines of a stop line. Finally, we compute the center position between C_{in} and C_{out} for use as the position G_{WL} of the stop line on the road. The detailed algorithm to estimate the position of the stop line is described in the following algorithm.

Algorithm 6.6 ***Computing the position of a stop line on the road.***

Input: two corresponding space plane parameters A_{B1} and A_{S1}, four other corresponding parameters A_{B2}, A_{S2}, K_{B1}, and K_{S1} obtained from Algorithm 6.5, of the stop line on roads appearing in an omni-image.

Output: the position of the stop line on the road, G_{WL}, in the CCS.

Steps.

Step 1. By A_{B1} and A_{S1}, compute one boundary space line L_X of the stop line by Equation (5.33) and obtain its equation as follows:

$$Y = Y_1; \quad Z = Z_1. \tag{6.11}$$

Step 2. By A_{B1} and A_{S1}, compute one boundary space line L_X of the stop line by Equation (5.33) and obtain its equation as follows:

$$Y = Y_2; \quad Z = Z_2. \tag{6.12}$$

Step 3. By K_{B1} and K_{S1}, compute one boundary space line L_Z of the stop line by Equation (5.40) and obtain its equation as follows:

$$X = X_1; \quad Y = Y_3. \tag{6.13}$$

Step 4. Compute the width between the two lines by the following equation:

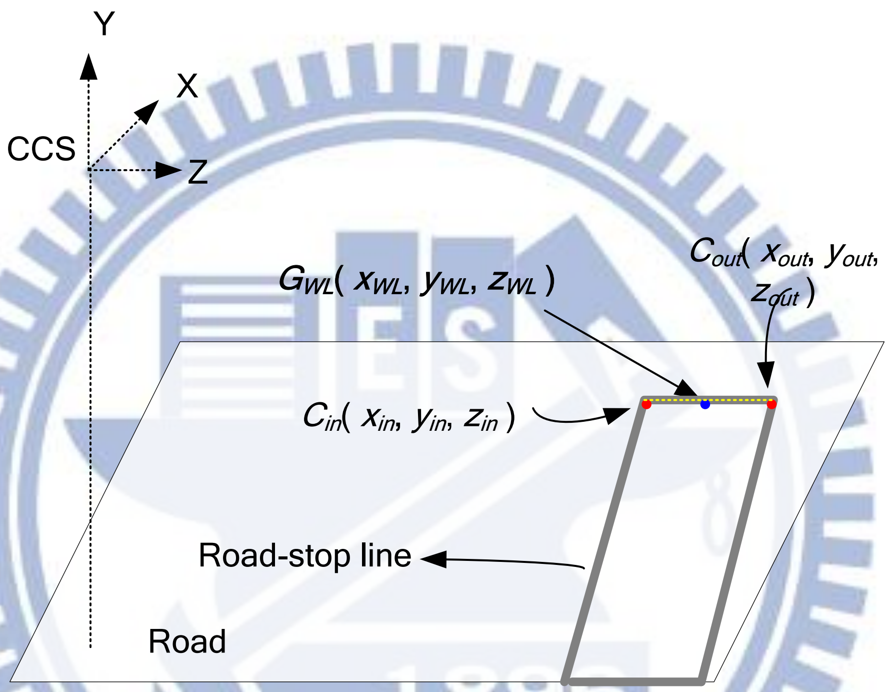
$$d = |Z_1 - Z_2|. \tag{6.14}$$

Step 5. If $|d - D_{width}| \leqq Th_W$ where D_{width} represents the pre-measured width of the stop line and Th_W is a pre-defined threshold, then go to Step 6; else, show a message saying that there is no stop line and exit.

Step 6. Compute the coordinates (x_{WL}, y_{WL}, z_{WL}) of the position of the stop line on roads G$_{WL}$ in the CCS as follows:

$$X_{WL} = (X_2 + X_1)/2; \quad y_{WL} = (Y_1 + Y_2 + Y_3)/3; \quad z_{WL} = (Z_1 + Z_2)/2. \qquad (6.15)$$

Step 7. Take G_{WL} as the output.



Y
X
CCS
Z
C_{out}(x_{out}, y_{out}, z_{out})
G_{WL}(x_{WL}, y_{WL}, z_{WL})
C_{in}(x_{in}, y_{in}, z_{in})
Road-stop line
Road

Figure 6.10 Two obtained corners of C_{in} and C_{out} of a stop line on the road in the CCS.

6.3.2 Experimental results of detection of stop lines on roads

An input image with the projection of a stop line on the regions of *Mirrors S* and *B* is shown in Figure 6.11. After conducting the feature extraction and Canny edge detection processes, we obtain an edge-point image as shown in Figure 6.12. The result of stop line detection is shown in Figure 6.13(a) and the relative stop line position with respect to the vehicle is shown in Figure 6.13(b).

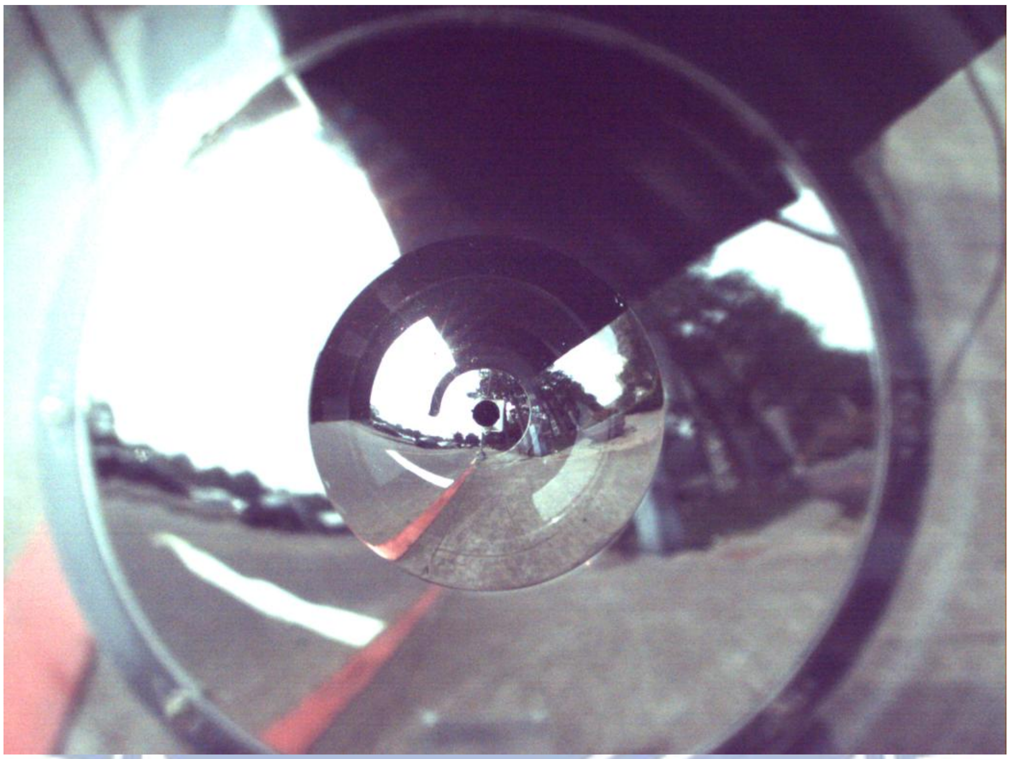


Figure 6.11 The omni-image with a stop line on the road.

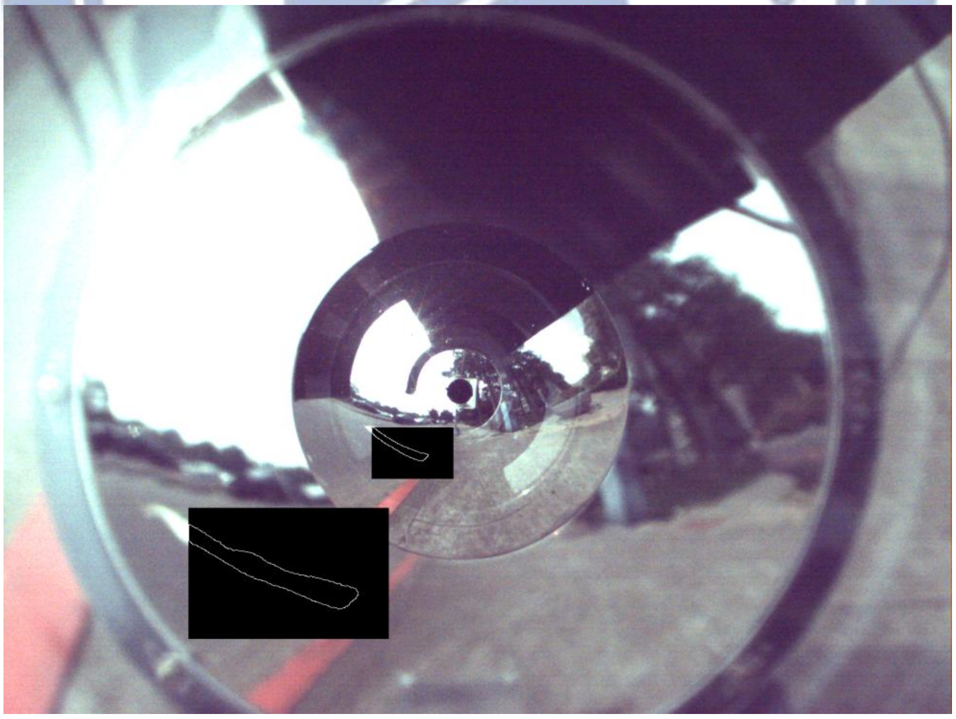
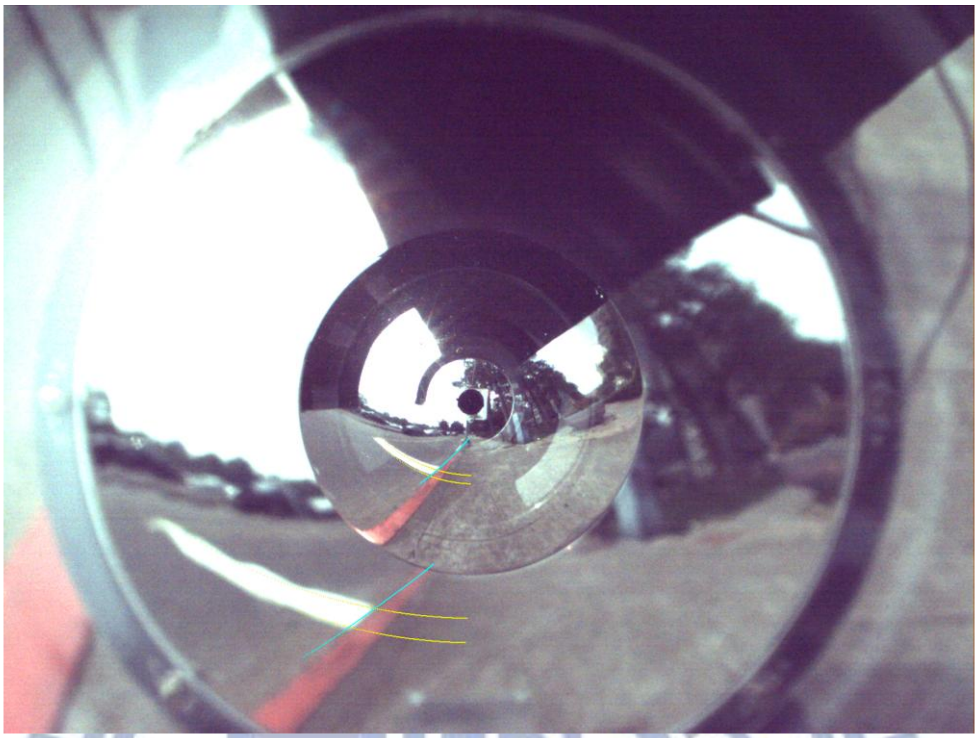
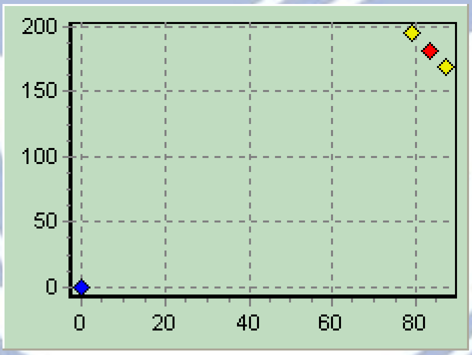


Figure 6.12 The result of stop line segmentation by the Canny edge detector.



(a)



(b)

Figure 6.13 The result of stop line on detection and position computation. (a) The resulting image of extracting the L_X and L_Z lines of the stop line. (b) The relative position of the stop line with respect to the vehicle position.

6.4 Proposed Method for Detecting and Localizing Traffic cones

6.4.1 Detection and localization of traffic cones

When engineering works are conducted on sidewalks, the workers usually put traffic cones near the working area to warn people. For this situation, we propose to detect traffic cones and use them as landmarks. The proposed method for traffic cone detection is similar to that for stop line detection. But here we detect one L_Z line and one L_X lines to carry out the detection of the traffic cone. After successfully detecting the boundary lines of the traffic cone by Algorithm (6.5), we can utilize two parameters, A_{B1} and K_{B1}, to compute its position. Because the traffic cone is located on the sidewalk, we can use it to compute the position of the traffic cone. The detailed algorithm to estimate the position of a traffic cone is described in the following algorithm.

Algorithm 6.6 ***Computing the traffic cone position.***

Input: the height of the camera center H, one space plane parameters A_{B1} and another parameter K_{B1} obtained from Algorithm 6.5, of the traffic cone appearing in an omni-image.

Output: the traffic cone position G_{TC} in the CCS.

Steps.

Step 1. By A_{B1} and H, compute the position of the traffic cone by Equation (5.33) and obtain its equation as follows:

$$Y = -H; \quad Z = Z_1. \tag{6.16}$$

Step 2. By K_{B1} and H, compute the position of the traffic cone by Equation (5.40) and obtain its equation as follows:

$$X = X_2, \quad Y = -H. \tag{6.17}$$

Step 3. Compute the coordinates (x_{TC}, y_{TC}, z_{TC}) of the position G_{TC} of the traffic cone in the CCS as follows:

$$X_{TC} = X_2; \quad y_{TC} = -H; \quad z_{TC} = Z_1. \tag{6.18}$$

Step 4. Take G_{TC} as output.

Besides, we also utilize this corner to compute the L_Y line. We solve the Equations (5.16) and (5.19) to obtain the equation:

$$A = \frac{-a_1}{-K} \tag{6.19}$$

which is equivalent to

$$A \times K = -B \tag{6.20}$$

where

$$B = -a_1 \tag{5.21}$$

Finally, we can illustrate the L_Y line which goes through the corner point and is perpendicular to the ground by the Equation (6.20).

6.4.2 Experimental results of traffic cone detection

Some experimental results of detecting the traffic cone using the proposed method are given in this section. An input omni-image with a traffic cone is shown in Figure 6.14. After conducting the feature extraction and Canny edge detection processes, we obtain an edge-point image as shown in Figure 6.15. The final result of traffic cone detection is shown in Figure 6.16(a) and the relative position of the traffic cone with respect to the vehicle is shown in Figure 6.16(b).

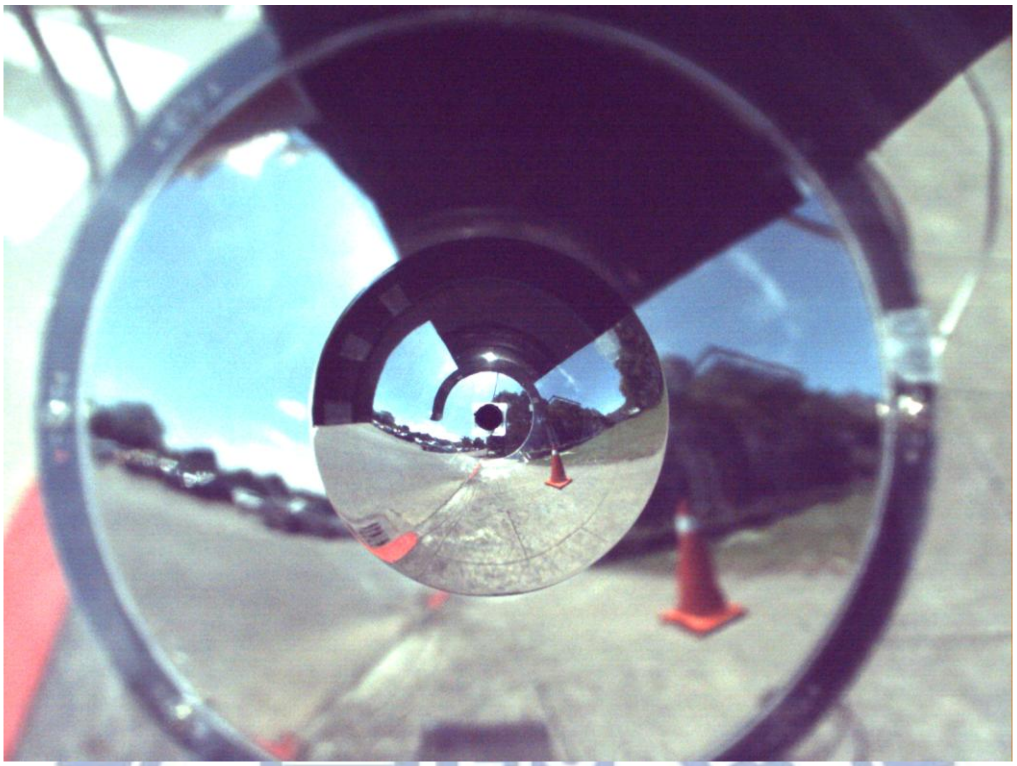


Figure 6.14 The omni-image with a traffic cone.

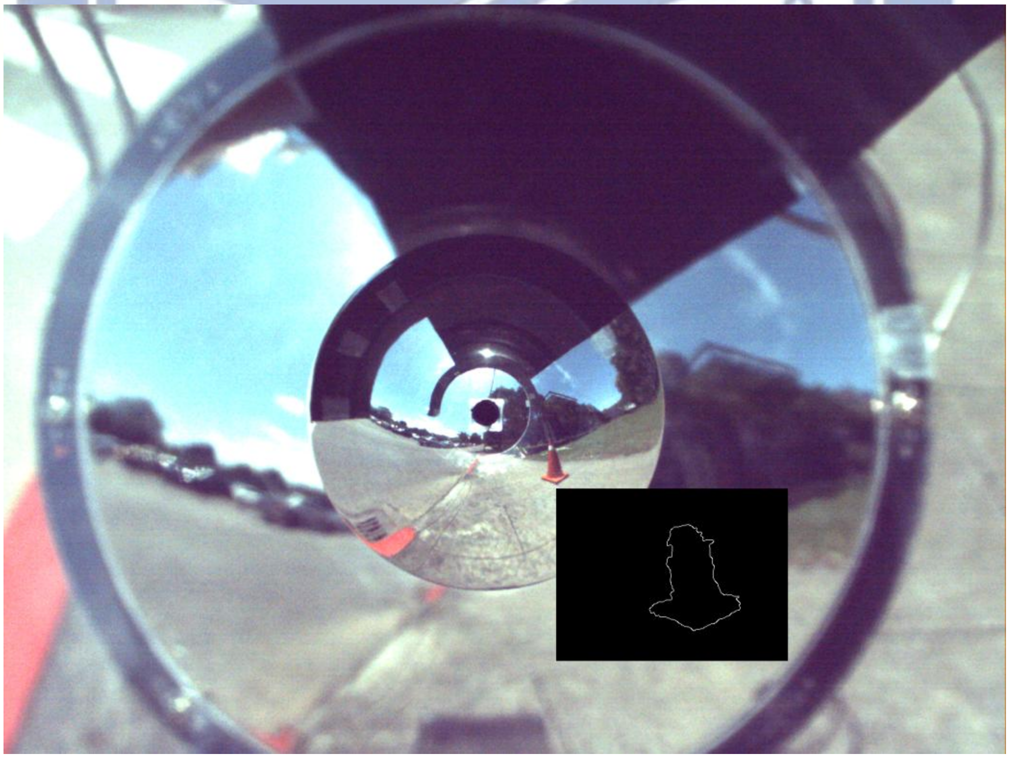
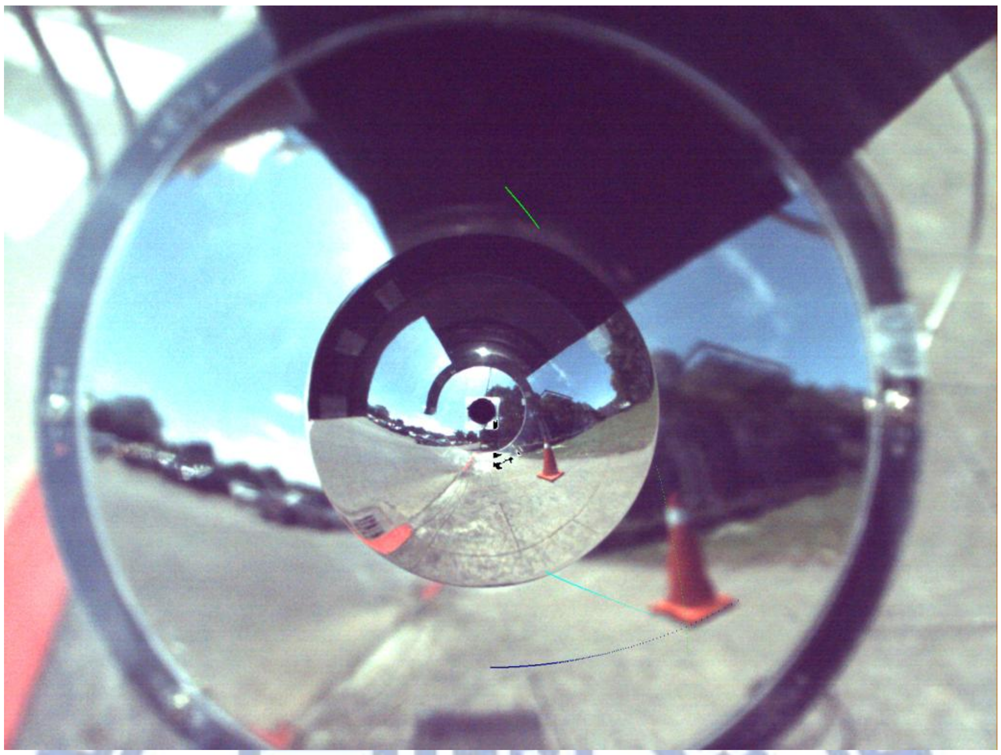
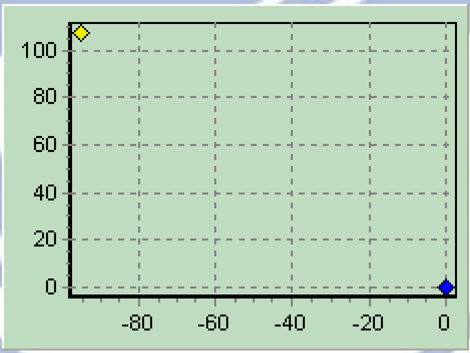


Figure 6.15 The result of traffic cone segmentation using the Canny edge detector.



(a)



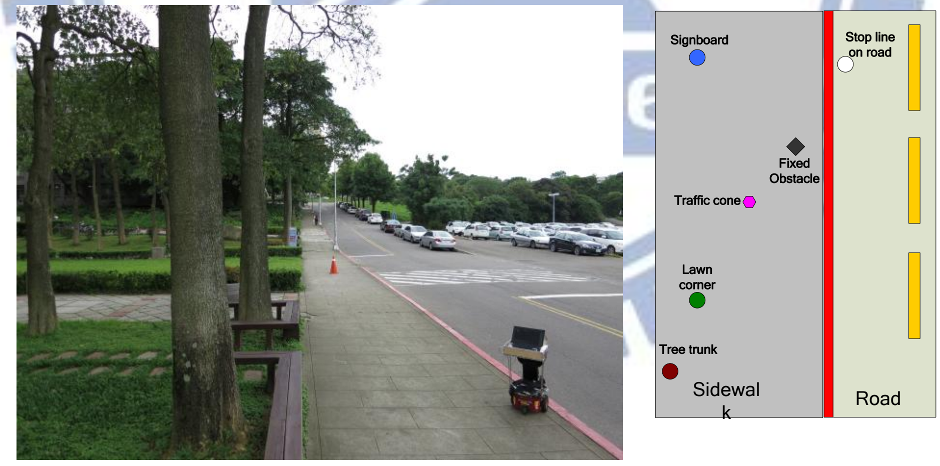
(b)

Figure 6.16 The result of traffic cone detection and the obtained position of the traffic cone. (a) The result image of extracting the L_X, L_Y, and L_Z lines of the traffic cone (b) The relative position of the traffic cone with respect to the vehicle position.

Chapter 7
Experimental Results and Discussions

7.1 Experimental Results

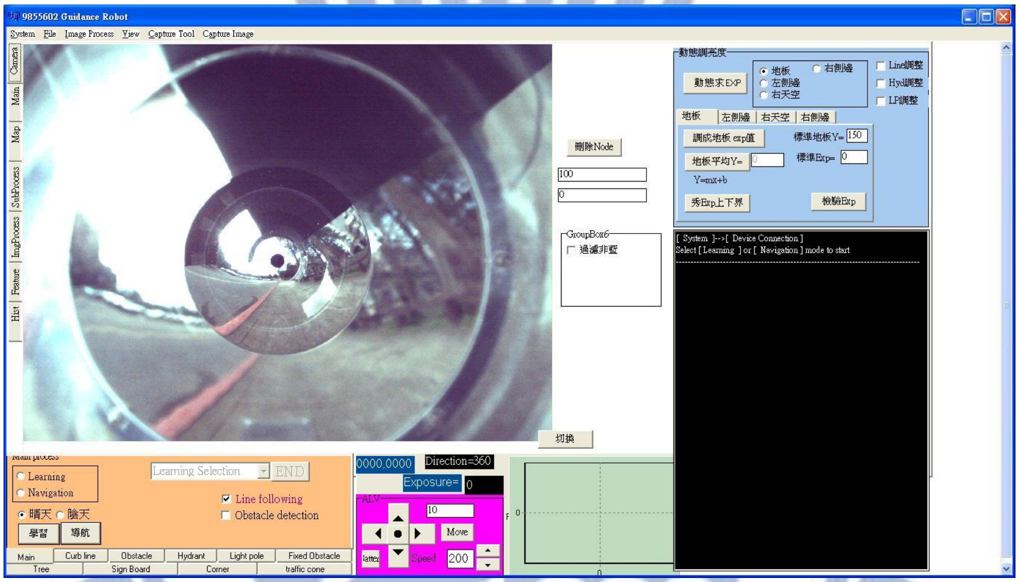
In this section, we will show some experimental results of the proposed vehicle navigation system for use as a machine guide dog in the learning and navigation processes. The experimental environment was an outdoor sidewalk in National Chiao Tung University as shown in Figure 7.1(a). We illustrate the outdoor environment including a gray sidewalk, a red curb line, and some landmarks as shown in Figure 7.1(b). The portion to the right of the red curb line is part of an around-campus road.



(a) (b)

Figure 7.1 The experimental environment. (a) A side view. (b) Illustration of the environment.

In the learning process, a trainer guided the vehicle by the use of a learning interface as shown in Figure 7.2 to construct a navigation path. The trainer navigated the vehicle to conduct learning tasks on the vehicle system along the path. After arriving at appropriate locations on the sidewalk, the vehicle was commanded to learn the positions of specific landmarks like signboard, tree trunk, stop line, …, etc. In addition, the position of the fixed obstacle was recorded manually by localizing its position on the omni-image as shown in Figure 7.3. At the end of the learning process, the trainer obtained a navigation map with a navigation path and other environment landmarks as illustrated in Figure 7.4.


9855602 Guidance Robot
System File Image Process View Capture Tool Capture Image
Camera Main Map SubProcess ImgProcess Feature Hist
刪除Node
切換
動態調亮度
動態求EXP
Line調整
Hyd調整
LP調整
地板 左側邊 右天空 右側邊
調成地板 exp值
秀Exp上下界
檢驗Exp
Main process
學習 導航
Learning Selection
END
ALV
Move
Speed
Main Curb line Obstacle Hydrant Light pole Fixed Obstacle
Tree Sign Board Corner traffic cone

100
0

GroupBox6
☐ 過濾非藍

(● 地板) 左側邊 右天空 右側邊
標準地板Y= 150
地板平均Y= 0
標準Exp= 0
Y=mx+b

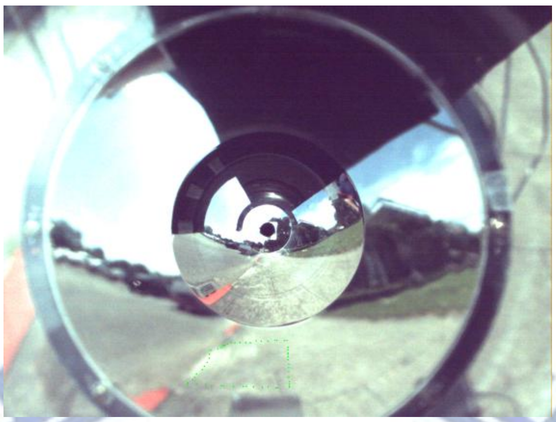
```
[ System ]-->[ Device Connection ]
Select [ Learning ] or [ Navigation ] mode to start
--------------------------------------------------------------------------------
```

Learning Navigation
(● 晴天) 陰天
☑ Line following
☐ Obstacle detection

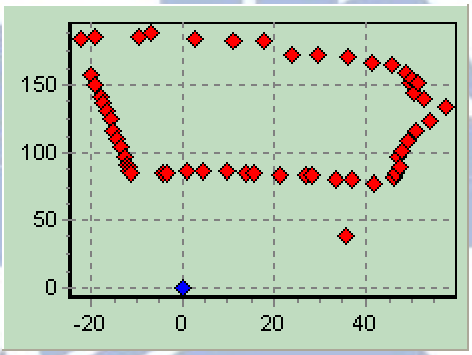
0000,0000 Direction=360
Exposure= 0
10
Speed 200

Figure 7.2 The Learning interface of the proposed vehicle system.

In the navigation process, the vehicle started from the same position just like in the learning process and navigated along the recorded navigation path nodes with the curb line following technique. Then, the vehicle detected many types of landmarks and localized its position. Some results of landmark detection are shown in Figure 7.5. By conducting curb detection, the vehicle kept its path parallel to the curb. A result of curb detection is given in Figure 7.6. Besides, after detecting the fixed obstacle in the navigation path, the vehicle adopted the obstacle avoidance procedure to avoid it as shown in Figures 7.7. Finally, the vehicle reached the appointed terminal node successfully, and the path map with a record of each vehicle position in the navigation process is illustrated in Figure 7.8.



(a)



(b)

Figure 7.3 Learning of a fixed obstacle. (a) The position of fixed obstacle on the omni-image (Lime-colored points clicked by the trainer). (b) Computed fixed obstacle positions in the real world.

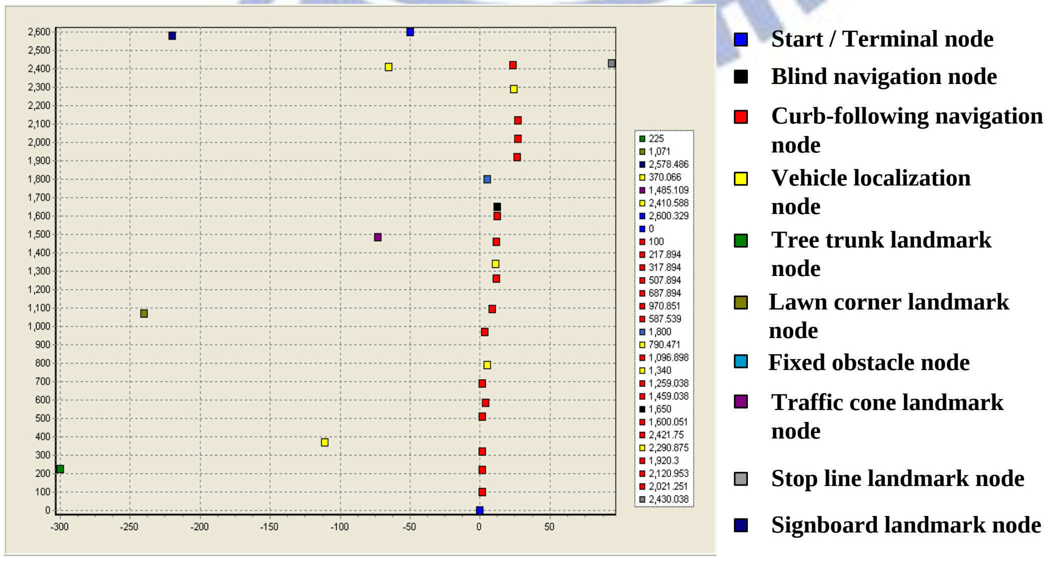
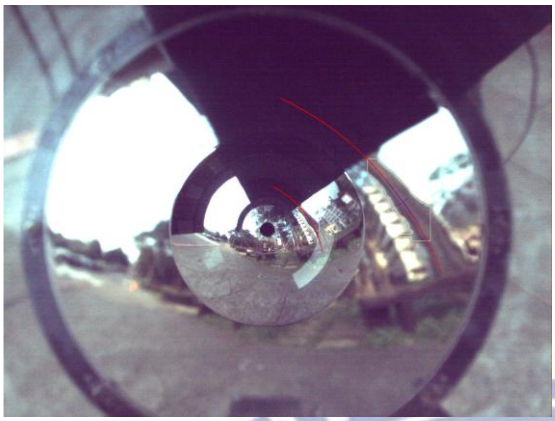
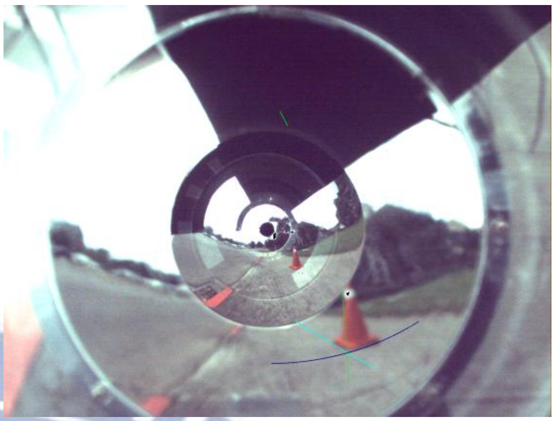
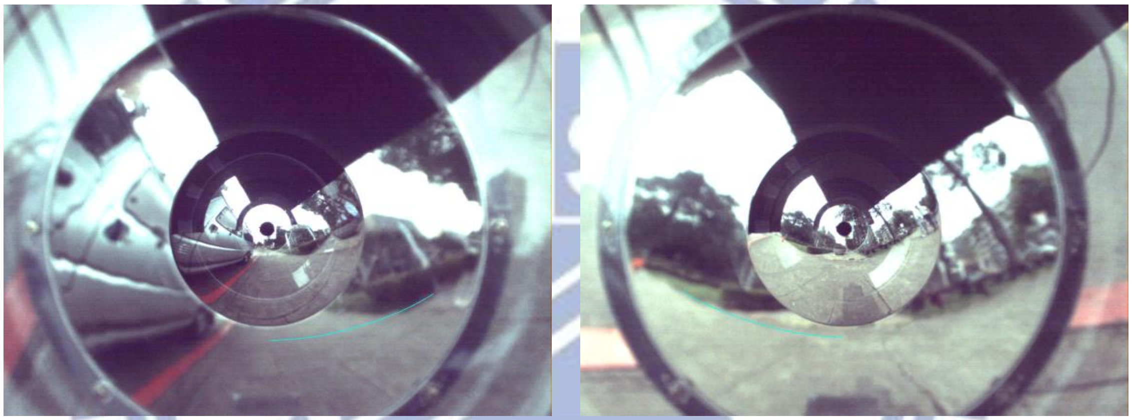
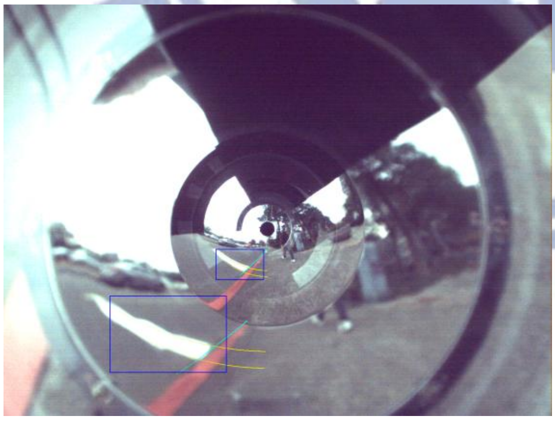


Figure 7.4 Illustration of the learned navigation path.


(a)


(b)


(c)


(d)

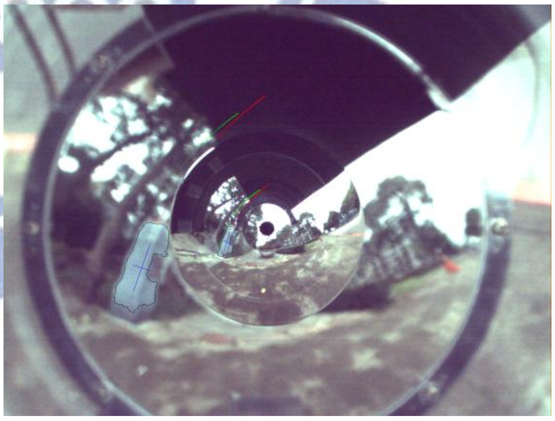

(e)

Figure 7.5 Some results of landmark detection. (a) A tree trunk detection result with L_Y line drawn in red. (b) A traffic cone detection result with L_X, L_Y, and L_Z line drawn in dark blue, lime, and navy blue, respectively. (c) A lawn corner detection result with two boundary lines drawn in navy blue. (d) The result of stop line with three boundary lines drawn in yellow and navy blue, respectively. (e) A signboard detection result with L_Y line drawn in red and lime, respectively.

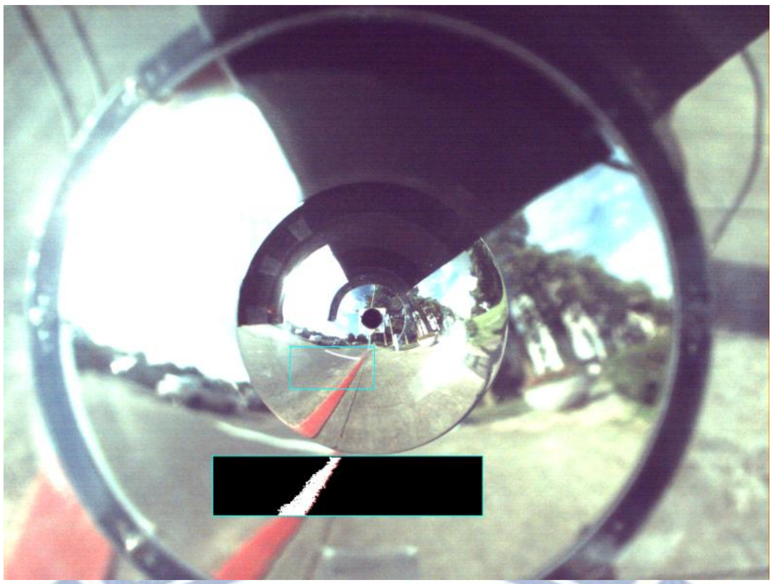


Figure 7.6 The result of curb line detection.

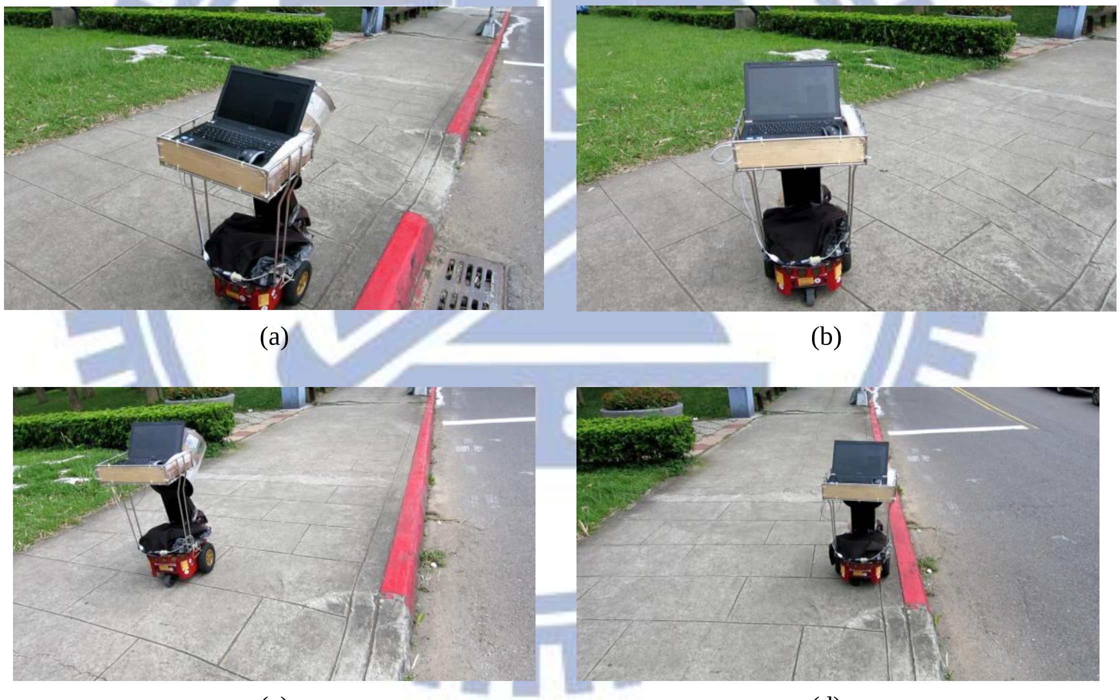


Figure 7.7 The vehicle reads the fixed obstacle position from the navigation path and change the path to avoid it. (a)~(d) show the process of fixed obstacle avoidance.

In Table 7.1, we show the errors in percentage between the actual position of the landmarks and the estimated positions of the landmarks of 8 times of navigations using the proposed system. From the table, we see that the average error of the landmark position is 7.52%. These small error percentages show that the precision of the proposed system is satisfactory for real applications.

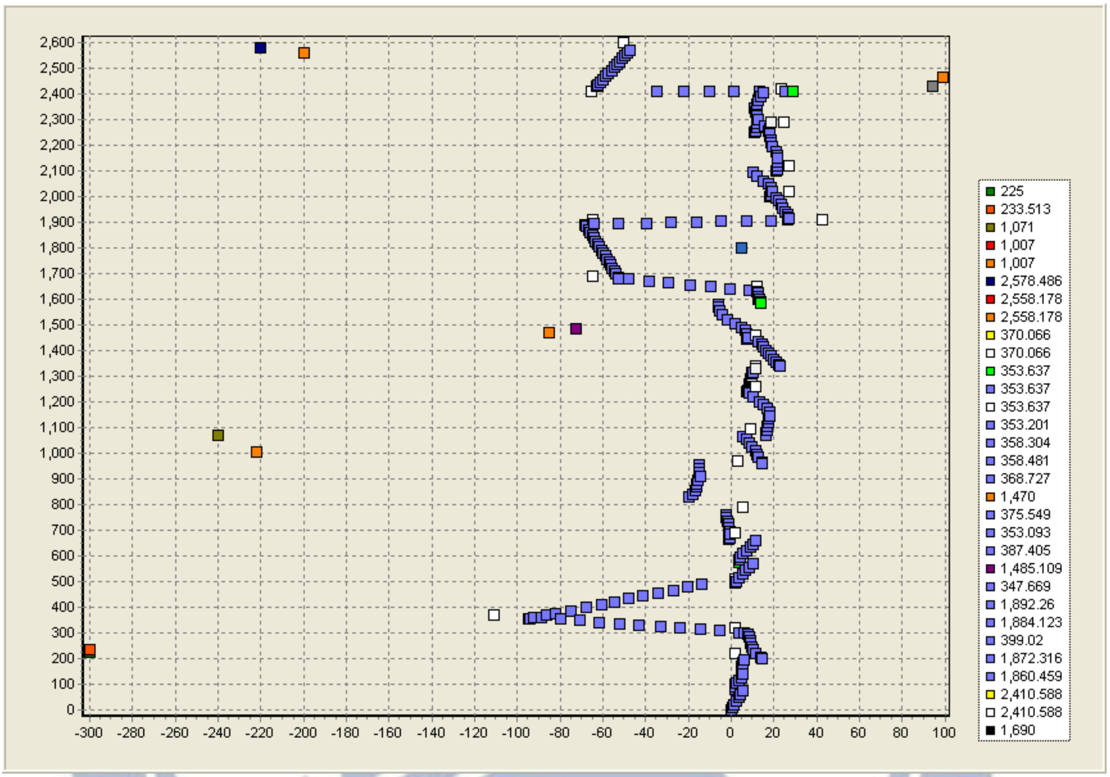


Figure 7.8 The recorded path map in the navigation process. (Blue points represent the vehicle path and other points with different color represent different localized landmark positions in different detections).

7.2 Discussions

By analyzing the experimental results of the vehicle navigation, we find some problems. Firstly, for sidewalk curb detection, we detect the curb with a specific surface in the campus of National Chiao Tung University. More kinds of curb lines with different colors should be learned for the line following technique. Also, the light reflection caused by the plastic camera enclosure created in the omni-image also causes ill effects in image analysis. A possible solution is to learn these specific regions in advance and ignore them when conducting image processing. Furthermore, we may spend much time to detect the L_X line and localize it. A possible solution is to implement an embedded system to speed up the calculation. Finally, more experiments in different environments should also be conducted to test our system more thoroughly.

Table 7.1 Precision of estimated landmark positions and their error percentages.

	navigation No.	Real position	Estimated position	Estimated landmark error	Average estimated landmark error
Tree trunk	1	239.01	224.19	6.20%	3.11%
	2		237.20	0.76%	
	3		234.24	1.99%	
	4		235.42	1.50%	
	5		236.60	1.01%	
	6		214.71	10.17%	
	7		233.08	2.48%	
	8		237.20	0.76%	
Lawn corner	1	372.81	331.44	11.10%	7.81%
	2		314.04	15.77%	
	3		336.40	9.77%	
	4		354.98	4.78%	
	5		336.77	9.67%	
	6		338.92	9.09%	
	7		366.90	1.59%	
	8		369.98	0.76%	
Traffic cone	1	167.14	161.79	3.20%	5.08%
	2		150.65	9.86%	
	3		161.68	3.27%	
	4		154.42	7.61%	
	5		168.73	0.95%	
	6		157.33	5.87%	
	7		165.56	0.94%	
	8		182.05	8.92%	
Stop line on road	1	178.28	175.28	1.68%	12.69%
	2		207.40	16.33%	
	3		189.44	6.26%	
	4		212.74	19.33%	
	5		199.89	12.12%	
	6		219.26	22.98%	
	7		185.30	3.94%	
	8		211.96	18.89%	
Signboard	1	227.85	251.93	10.57%	8.91%
	2		221.76	2.67%	
	3		198.92	12.70%	
	4		263.36	15.59%	
	5		222.30	2.44%	
	6		242.62	6.48%	
	7		186.00	18.37%	
	8		222.19	2.48%	
Average					7.52%

Chapter 8
Conclusions and Suggestions for Future Works

8.1 Conclusions

Construction of a machine guide dog using a two-mirror omni-camera and an autonomous vehicle has been proposed in this study. To implement such as a system, several methods have been proposed.

At first, by the pano-mapping technique proposed by Jeng and Tsai [25], we calibrate the two-mirror omni-camera used in this study by recording the relationship between the image pixels and the real-world azimuth and elevation angles. Next, by the use of a learning interface designed in this study, a trainer can guide the vehicle to navigate on a sidewalk and construct a navigation path conveniently including the path nodes, alone-path landmarks, and relevant guidance parameters.

Next, two new space line detection techniques based on the pano-mapping technique have been proposed. Each space line, which when projected on an omni-image becomes a conic-section curve, is detected by the use of analytic formulas and the Hough transform technique. In addition, for the three types of space line which exists in landmarks like the tree trunk, the lawn corner, the signboard, the stop line on roads, and the traffic cone, we can further compute its position directly using omni-images according to the pano-mapping technique.

Also, several landmark detection techniques have been proposed for conducting vehicle navigation. Firstly, a curb line detection technique has been proposed for use to guide the vehicle on a safe path as well as to adjust the odometer reading of the vehicle orientation. Next, some natural and artificial landmark detection techniques have been proposed as well. The three types of space lines found in these landmarks using the techniques can be used to localize the vehicle in the navigation process. Furthermore, to conduct the landmark detection works more effectively in the outdoor environment, techniques for dynamic threshold adjustments have also been proposed, which can be used to handle different lighting conditions.

Good landmark detection results and successful navigation sessions on a sidewalks in the National Chiao Tung university campus show the feasibility of the proposed methods.

8.2 Suggestions for Future Works

According to our experience obtained in this study, several suggestions and related interesting issues worth further investigations in the future are stated in the following:

(1) it seems necessary to develop some techniques to detect moving objects, like pedestrians walking on the sidewalk or people riding bikes;

(2) it is a challenge to detect natural landmarks which have no obvious color information to conduct vehicle navigation in more complicated outdoor environments;

(3) it is desired to design a new camera system which has a smaller size for more convenient uses by the blind people;

(4) it is a challenge to develop additional techniques to guide the vehicle to pass crossroads, like recognizing traffic signals and following zebra crossings, etc.;

(5) it may be necessary to add the capability of warning the user via sound in danger conditions.

(6) It is interesting to combine other facilities like range finders to implement the system for more complicated applications.

(7) It is desired to utilize properties of trigonometric functions to reduce the range of the Hough space to speed up the computation time.

References

[1] J. Borenstein and I Ulrich, “The GuideCane — a computerized travel aid for the Active guidance of blind pedestrians,” *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1283–1288, Albuquerque, NM, USA, Apr. 1997.

[2] V. Ivanchenko, J. Coughlan, W. Gerrey, and H. Shen, “Computer vision-based clear path guidance for blind wheelchair users” *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*, pp. 291–292, New York, NY, USA, 2008.

[3] Hideo Mori, Shinji Kotani, and Noriaki Kiyohiro, "A robotic travel aid “HITOMI”," *Proceedings of the IEEE/RSJ/GI International Conference onIntelligent Robots and Systems '94*, vol. 3, pp. 1716–1723, Munich, Bavaria, Germany, 1991.

[4] Y. Z. Hsieh and M. C. Su, “A stereo-vision-based aid system for the blind,” *M. S. Thesis*, Department of Computer Science and Information Engineering, National Central University, Jhongli, Taoyuan, Taiwan, June 2006.

[5] S. Wenqin, J. Wei, and C. Jian, “A machine vision based navigation system for the blind,” *IEEE International Conference on Computer Science and Automation Engineer*ing, vol. 3, pp. 81-85, Shanghai, People's Republic of China, July 2011.

[6] S. Willis and S. Helal, “RFID information grid for blind navigation and wayfinding,” *Proceedings of the 9th IEEE International Symposium on Wearable Computers*, pp. 34–37, Washington, DC, USA, Oct. 2005.

[7] L. Ran, S. Helal, and S. Moore, “Drishti: an integrated indoor/outdoor blind navigation system and service,” *Proceedings of 2nd IEEE Annual Conference on Pervasive Computing and Communications*, pp. 23–31, Orlando, Florida, USA,

2004.

[8] M. F. Chen and W. H. Tsai, "Automatic learning and guidance for indoor autonomous vehicle navigation by ultrasonic signal analysis and fuzzy control techniques," *Proceedings of 2009 Workshop on Image Processing, Computer Graphics, and Multimedia Technologies, National Computer Symposium*, pp. 473–482, Taipei, Taiwan, Nov. 2009.

[9] E. Abbot and D. Powell, "Land-vehicle navigation using GPS," *Proceedings of the IEEE,* vol. 87, no. 1, pp. 145–162, Jan. 1999.

[10] Sami Atiya and Gregory D. Hager, "Real-time vision-based robot localization," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 6, pp. 785–800, Dec. 1993.

[11] M. C. Chen, "Vision-based security patrolling in indoor environments using autonomous vehicles," *Proceedings of 2005 Conference on Computer Vision, Graphics and Image Processing*, pp. 811–818, Taipei, Taiwan, Republic of China.

[12] K. L. Chiang and W. H. Tsai. "Vision-based autonomous vehicle guidance in indoor environments using odometer and house corner location information," *Proceedings of 2006 IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP-2006),* pp. 415–418, Pasadena, California, USA, Dec. 18-20, 2006.

[13] M. Agrawal and K. Konolige, "Real-time localization in outdoor environments using stereo vision and inexpensive GPS," *Proceedings of 18th International Conference on Pattern Recognition*, Hong Kong, People's Republic of China, pp. 1063–1068, vol. 3, Aug, 2006.

[14] S. Y. Tsai and W. H. Tsai, "Simple automatic path learning for autonomous vehicle navigation by ultrasonic sensing and computer vision techniques,"

Proceedings of 2008 International Computer Symposium, vol. 2, pp. 207-212, Taipei, Taiwan, Dec. 2008.

[15] D. L´opez, K. Sj¨o, C. Paul, and P. Jensfelt, "Hybrid laser and vision based object search and localization," *Proceedings of IEEE International Conference on Robotics and Automation*, pp.2636–2643, Pasadena, CA, USA, May 19-23, 2008.

[16] W. Lui and R. Jarvis, "Eye-Full Tower: A GPU-based variable multibaseline omnidirectional stereovision system with automatic baseline selection for outdoor mobile robot navigation," *Robotics and Autonomous Systems*, pp. 747-761, vol. 58, no. 6, Apr. 2010.

[17] H. Fu, Z. Cao, and X. Cao, "Embedded omni-vision navigator based on multi-object tracking," *Machine Vision and Applications*, pp. 349-358, vol. 22, no. 2, 2011.

[18] D. Ishizuka, A. Yamashita, R. Kawanishi, T. Kaneko, and H. Asama, "Self-localizaion of mobile robot equipped with omnidirectional camera using image matching and 3D-2D edge matching," *Proceedings of IEEE International Conference on Computer Vision Workshops*, Barcelona, Spain, pp. 272-279, Nov. 6-13, 2011.

[19] C. J. Wu and W. H. Tsai, "Location estimation for indoor autonomous vehicle navigation by omni-directional vision using circular landmarks on ceilings," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 546-555, May 2009.

[20] Taiwan Guide Dog Association, "About Taiwan Guide Dogs Information," Available online:
http://www.guidedog.org.tw/.

[21] Taiwan Foundation for the Blind, "Get to know blind people," Available online:
http://www.tfb.org.tw/english/index.html.

[22] J. K. Huang, “Autonomous vehicle navigation by two-mirror omni-directional imaging and ultrasonic sensing techniques,” *M. S. Thesis*, Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, June 2010.

[23] Y. H. Chou and W. H. Tsai, “Guidance of a vision-based autonomous vehicle on sidewalks for use as a machine guide dog,” *Proceeding of 2011 Conference on Computer Vision, Graphic, and Image Processing*, Chiayi, Taiwan, 2011.

[24] S. W. Jeng and W. H. Tsai, "Using pano-mapping tables to unwarping of omni-images into panoramic and perspective-view Images," *Proceeding of IET Image Processing*, vol. 1, no. 2, pp. 149–155, June 2007.

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

[26] C. J. Wu and W. H. Tsai, "An omni-vision based localization method for automatic helicopter landing assistance on standard helipads," *Proceedings of 2nd International Conference on Computer and Automation Engineering*, vol. 3, pp. 327–332, Singapore, 2010.