當機器導盲犬用之有視覺自動車在人行道上導航之研究

A Study on Guidance of a Vision-based Autonomous Vehicle on Sidewalks for Use as a Machine Guide Dog

研 究 生：周彥翰

指導教授：蔡文祥　教授

中 華 民 國 一 百 年 六 月

當機器導盲犬用之有視覺自動車在人行道上導航之研究

A Study on Guidance of a Vision-based Autonomous Vehicle on Sidewalks for Use as a Machine Guide Dog

研 究 生：周彥翰　　　Student：Yen-Han Chou

指導教授：蔡文祥　　　Advisor：Wen-Hsiang Tsai

國 立 交 通 大 學
資 訊 科 學 與 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 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年六月

當機器導盲犬用之有視覺自動車在人行道上導航之研究

研究生：周彥翰　　　指導教授：蔡文祥 博士

國立交通大學資訊科學與工程研究所

摘要

本研究提出了一個利用有視覺的自動車在戶外人行道上作機器導盲犬應用的系統，該系統利用一部搭載雙鏡面環場攝影機的自動車當作實驗平台，能在環場影像中直接求出實際物體的立體資訊。首先，利用環境學習的技術建立導航地圖，此地圖包含自動車導航路徑、沿途路標的位置，以及相關的導航參數。接著，利用人行道上特定的路標(人行道路緣、消防栓和電線杆)作定位來輔助導航，本研究整合上述兩項技術提出一個擁有自動定位和自動導航功能的自動車系統。

此外，本研究亦利用空間映射的方法提出新的直線偵測技術，能夠在環場影像上直接偵測出直線特徵，並計算出人行道上垂直形狀路標的位置，進而提出偵測以及定位消防栓和電線杆的方法。最後利用已定位的路標位置，來校正機械誤差，並算出正確的自動車位置。接著，本研究也提出自動跟隨人行道路緣線的技術，以及一項新的動態障礙物偵測技術，利用一「地板配對表」定出障礙物位置，讓自動車穩定且不間斷地完成導航，並在導航路徑中閃避障礙物。此外，本研究亦提出動態調整曝光值以及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。實驗結果顯示本研究所提方法完整可行。

A Study on Guidance of a Vision-based Autonomous Vehicle on Sidewalks for Use as a Machine Guide Dog

Student: Yen-Han Chou　　Advisor: Wen-Hsiang Tsai

Institute of Computer Science and Engineering
National Chiao Tung University

ABSTRACT

A vision-based autonomous vehicle system for use as a machine guide dog in outdoor sidewalk environments is proposed. A vehicle equipped with a two-mirror omni-camera system, which can compute 3D information from acquired omni-images, is used as a test bed. First, an environment learning technique is proposed to construct a navigation map, including a navigation path, along-path landmark locations, and relevant vehicle guidance parameters. Next, a vehicle navigation system with self-localization and automatic guidance capabilities using landmarks on sidewalks including curb lines, hydrants, and light poles is proposed. Based on a space-mapping technique, a new space line detection technique for use on the omni-image directly is proposed, which can compute the 3D position of a vertical space line in the shape of a sidewalk landmark.

Moreover, based on the vertical space line detection technique just mentioned, hydrant and light pole detection and localization techniques are proposed. Also proposed accordingly is a method for vehicle self-localization, which can adjust an imprecise vehicle position caused by incremental mechanic errors to a correct one. In addition, for the purpose to conduct stable and continuous navigation, a curb line following technique is proposed to guide the vehicle along a sidewalk. To avoid obstacles on the navigation path, a new dynamic obstacle detection technique, which uses a ground matching table to localize an obstacle and then avoid it, is proposed. Furthermore, dynamic techniques for exposure and threshold adjustments are 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.

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

Guide dogs provide special services to blind people. Formally training methods for guide dogs have been adopted for over seventy years. Besides leading blind people to correct destinations, guide dogs can assist them to avoid obstacles and negotiate street crossings, public transportations, and unexpected events when navigating on the road. For blind people, guide dogs not only strongly enhance their mobility and independence, but also improve the quality of their lives.

However, according to the information provided by Taiwan Foundation for the Blind [13] and Taiwan Guide Dog Association [14], there are more than fifty thousand blind people and just thirty trained guide dogs in Taiwan. Therefore, not all of the blind people have opportunities to adopt their own guide dogs, and so they have to utilize some other mobility aids likes blind canes instead. At least the following problems cause difficulties in training more guide dogs for the blind:

1. it costs at least one million NT dollars to train one guide dog;
2. only certain breeds of dogs can be trained as guide dogs;
3. after carefully bred for over one year, they still need be trained for four to six months before navigation tasks can be assigned to them for specific blind persons;
4. personality and individual differences between the master and a guide dog are problems which should be solved;

In order to overcome the problem of insufficient guide dogs, it is desirable to employ a *machine* guide dog in replacement of traditional one for each blind person. A vision-based autonomous vehicle with high mobility and being equipped with an omni-camera can assume this task if it can be designed to automatically navigate in outdoor environments by monitoring the camera's field of view (FOV) automatically. When the vehicle detects the existence of a risk area, it must safely bring the blind person through the dangerous condition *by itself*; and when the vehicle arrivals at the goal according to the instruction, it should give him/her a notice immediately. This study aims to design a machine guide dog with these functions using autonomous vehicle guidance techniques.

For this purpose, the most important issue is how to construct the autonomous vehicle to navigate successfully and securely in complicated conditions in outdoor environments. Usually, an autonomous vehicle is equipped with an odometer, and we could obtain the current position with respect to the initial position. However, the location of the autonomous vehicle could become imprecise because the vehicle might suffer from incremental mechanic errors. One solution is to continually localize the vehicle by monitoring obvious natural or artificial landmarks in the environment using computer vision techniques.

Usually, there must be some regular scenes like sidewalks that a blind person has to pass frequently, so we may train the autonomous vehicle in advance just like training a guide dog in these places. Simply speaking, we may design the autonomous vehicle to "memorize" along-path landmarks in advance, and instruct the vehicle system during navigation to retrieve the current location information by the use of the learned landmarks and set foot on the expected destination in the end.

In general, a visual sensor could yield undesired effects in acquired images under varying lighting conditions in outdoor environments, and to solve this problem we might also train the vehicle to adapt to these different conditions. Moreover, the autonomous vehicle should also be required to prevent itself and the blind people from dangerous events in the guide process. Some suitable strategies like following a line and avoiding obstacles should be adopted in the navigation sessions.

In summary, the goal of this study is to develop an autonomous vehicle for use as a machine dog with the following abilities:

1. learning the path on sidewalks;
2. navigating to the goal successfully in a learned path;
3. detecting obstacles and avoiding them;
4. adapting itself to different weather conditions in outdoor environments.

1.2 Survey of Related Works

In recent years, more and more research results about developing walking aids for the blind have emerged, and some of them are reviewed here. As an improvement of the blind cane, a simple aid is to install a sensor device on a blind cane in order to detect obstacles at a certain distance. Other aids may also be designed to be worn by the blind like the NavBelt [1], which has the function of continually detecting front obstacles automatically. In general, we call these devices electronic travel aids (ETA) which cannot automatically guidance the blind but only help them to find obstacles. Therefore, some more helpful navigation systems were proposed. Borenstein and Ulrich [2] developed the “GuideCane” which has a shape similar to widely used blind canes and can find obstacles by ultrasonic sensors to help blind people to pass them automatically. A guide dog robot called Harunobu-5 [3] was proposed by Mori and Sano which can follow a person using a visual sensor. Also, Hsieh [4] utilized two cameras installed on a cap to find accessible regions and obstacles in unknown environments and alert the blind by auditory outputs. In addition, Lisa et al. [5] utilized a DGPS (differential GPS) device to localize a blind person in indoor and outdoor environments.

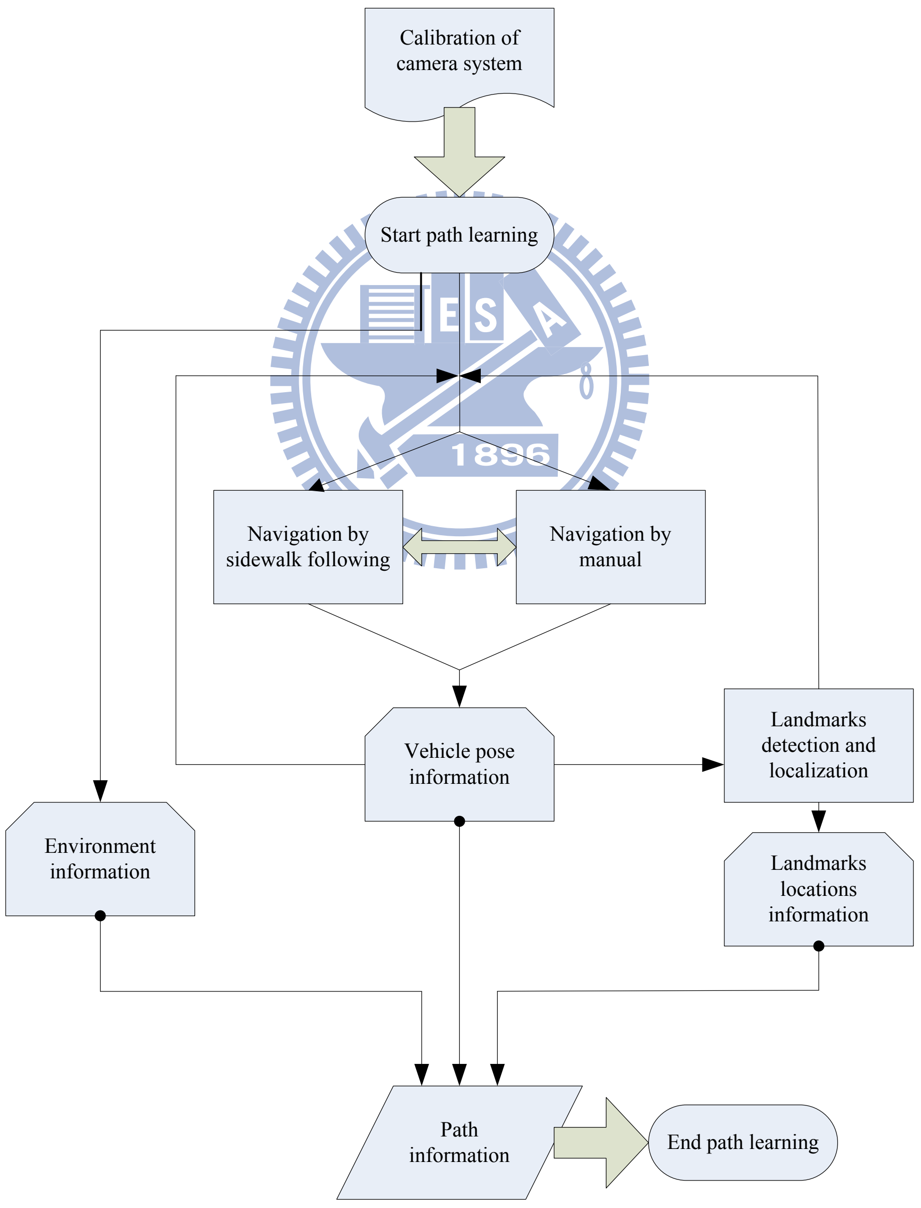
On the above-mentioned autonomous vehicle systems used for navigation, usually installed are some visual sensors or other equipments in order to give assistance to the blind. An autonomous vehicle system mounted with a tri-aural sensor and an infrared range scanner was proposed by Kam et al. [6]. Also, Chen and Tsai [7] proposed an indoor autonomous vehicle navigation system using ultrasonic sensors. In outdoors, the GPS can be used as a localization system for the vehicle [8]. Likewise, visual sensors have also been used widely for vehicle navigation. Chen and Tsai [9] proposed a vehicle localization method which modifies the position of the vehicle by monitoring learned objects. Another technique of vehicle localization by recognizing house corners was proposed by Chiang and Tsai [10]. Besides, in some other applications, cameras with other devices were combined as the sensing device. Tsai and Tsai [11] used a PTZ camera and an ultrasonic sensor to conduct vehicle patrolling and people following successfully. What is more is the use of cameras and laser range finders together for environment sensing, like Pagnottelli et al. [12] who performed data fusion for autonomous vehicle localization.

In contrast with a traditional CCD camera, an omni-camera has the advantage of having a larger FOV, and so they can monitor a larger environment area. Because of this advantage, in this study we exploit the use of a stereo omni-camera which is also useful for acquiring omni-images to retrieve range information. In the following, we review some studies about vehicle navigation systems using omni-cameras. One way of localizing a vehicle is to detect landmarks in environments. Yu and Kim [15] detected particular landmarks in home environments and localized the vehicle by the distance between the vehicle and each landmark. The technique proposed by Tasaki et al. [16] conducted vehicle self-localization by tracking space points with scale- and rotation-invariant features. Wu and Tsai [17] detected circular landmarks on ceilings to accomplish vehicle indoor navigation. Siemiątkowska and Chojecki [18] used the wall-plane landmarks to localize a vehicle. Another method proposed by Courbon et al. [19] conducted vehicle localization by memorizing key views in order along a path and compared the current image with them in navigation. The vehicle system proposed by Merke et al. [20] used omni-cameras to recognize lines on the ground to conduct self-localization in a Robocup contest environment.

Except for self-localization, the autonomous vehicle has to own more capabilities when navigating in more complicated environments. Obstacle avoidance is an essential ability for vehicle navigation [21]. In outdoor environments, estimation of traversability of a terrain is another important topic. Fernandez and Price [22] proposed a method which can find traversable routes on a dirty road using color vision. By training a classifier with autonomous training data, Kim et al. [23] could estimate the traversability of complex terrains. A mobile robot proposed by Quirin et al. [24] not only can navigate by sidewalk following in the urban area, but also can interact with the people.

1.3 Overview of Proposed System

In this study, our goal is to conduct the autonomous vehicle to navigate in outdoor environments. As discussed previously, vehicle localization is one of the important works we have to complete to implement a machine guide dog. The method of vehicle localization we propose is to detect landmarks along the path and to localize a vehicle's position by these landmarks. Also, some other strategies for reliable navigation are proposed in this system. In this section, we will roughly introduce the proposed vision-based autonomous vehicle system. The system process may be into two stages: the learning stage and the navigation stage. What is done in the learning stage is mainly training of the autonomous vehicle before navigation. Then, in the navigation stage we conduct vehicle navigation along the pre-selected path using the learned information. More details of the two stages are illustrated in Figure 1.1 and Figure 1.2, respectively, and discussed in the following.



Calibration of camera system
Start path learning
Navigation by sidewalk following
Navigation by manual
Vehicle pose information
Landmarks detection and localization
Environment information
Landmarks locations information
Path information
End path learning

Figure 1.1 Flowchart of learning stage

A. The learning stage

First, the learning stage consists of two steps. The first step, as a prior work, is to *train* the camera system equipped on the vehicle. The system operator conducting this step is called the *trainer* of the system subsequently. In general, a camera system has to be calibrated for the purpose of knowing the relation between the image and the real space. In this system, we use a two-mirror omni-camera as a visual sensor. Because of the difficulty in retrieving intrinsic and extrinsic parameters of the omni-camera, we adopt a space mapping technique [25], called *pano-mapping*, to calibrate our camera system instead. After the calibration work is done, we construct a space mapping table, called *pano-table*. Then, we can obtain range data from an omni-image directly using the pano-table and continue the navigation process.

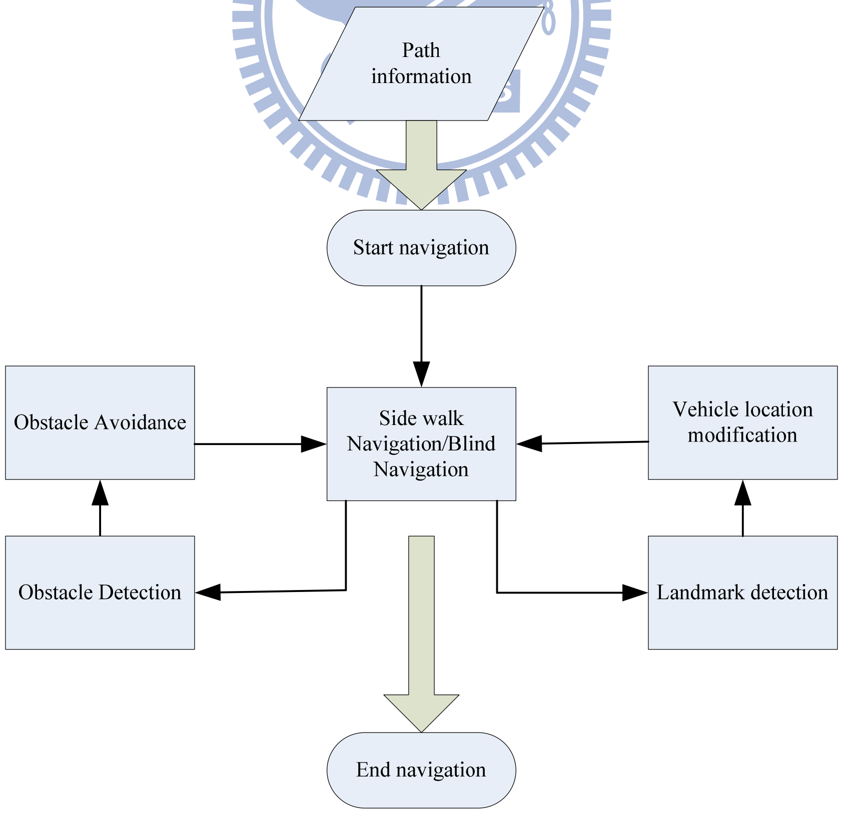


Figure 1.2 Flowchart of navigation stage

The second step of the learning stage is to guide the autonomous vehicle to *learn* path information, including vehicle poses in the navigation path, the location of each landmark, and some other environment information. After brining the vehicle to a chosen scene spot, a path learning work is started. Two autonomous navigation modes are designed in this study, which are applied alternatively in the navigation process. One mode is *navigation by following the sidewalk*, and the other is *manual control by the trainer*. After being assigned the first mode, the vehicle starts to navigate toward the goal and the information of the vehicle pose is continually recorded. If a specific landmark need be memorized in the path, the trainer may guide manually the vehicle using the second mode to an appropriate position and record the location of the landmark after the landmark is detected by the camera. In addition, other information about the outdoor environment is also recorded constantly when navigating. Finally, all of these data are integrated into the path information, which is stored in the memory and can be retrieved during the navigation stage.

B. The navigation stage

In the navigation stage, with the path information learned in advance, an automatic navigation process is started. Three major works are conducted by the vehicle in the navigation process — moving forward, obstacle detection, and vehicle location modification.

In principle, the autonomous vehicle constantly move forward toward the goal node by node based on the learned path information. In the movement between any two nodes in the path, the vehicle chooses one of two navigation modes — *navigation by following the sidewalk* or *navigation just by the use of an odometer* (called the *blind navigation mode*). When navigating in the first mode, the sidewalk curb with a prominent color is detected continuously and the line following technique is adopted to guide the vehicle. When no curb can be used for line-following guidance as often encountered on sidewalks, the second mode is adopted in which the vehicle navigates blindly according to the information of the odometer reading and the learned path.

Also, as a rule, the autonomous vehicle tries to find obstacles at any time and can take a proper obstacle avoidance strategy when desired. By the use of a stereo omni-camera, we develop a new method to detect obstacles on the ground using computer vision techniques. Moreover, when reaching a particular location in the learned path, the autonomous vehicle will detect the appointed landmark to localize itself automatically.

Furthermore, in this study we use some objects such as hydrants and light poles, which often can be found on sidewalks, as landmarks for vehicle localization. That is, we modify the vehicle position with respect to each located landmark to eliminate cumulated mechanical or vision-processing errors during the navigation process. Specifically, we propose a new space line detection technique to detect the along-path hydrant and the light pole and then calculate the locations of them. By these techniques, the autonomous vehicle can navigate safely and smoothly to the destination at the end of the navigation stage.

1.4 Contributions of This Study

Some contributions of this study are described as follows.

1. A method of training an autonomous vehicle for outdoor navigation using commonly-seen objects on sidewalks is proposed.
2. A new space line detection and localization technique using the pano-mapping table is proposed.

3. Techniques for detecting hydrants and light poles as landmarks for vehicle localization are proposed.
4. A technique of following sidewalk curbs for vehicle navigation is proposed.
5. A new obstacle avoidance technique and a new camera calibration method for it are proposed.
6. Dynamic camera exposure adjustment and dynamic thresholding methods for use in outdoor environments are proposed.

1.5 Thesis Organization

The remainder of this thesis is organized as follows. In Chapter 2, we introduce the configuration of the proposed system and the system process in detail. In Chapter 3, the proposed training methods for vehicle to learn the guidance parameters and the navigation path are described. In Chapter 4, we introduce the navigation strategies including the ideas, the proposed guidance techniques, and detailed navigation algorithms. In Chapter 5, a new space line technique is proposed and the proposed techniques of hydrant and light pole detections are described. In Chapter 6, the navigation techniques of line following and obstacle avoidance are introduced. In Chapter 7, some experimental results to show the feasibility of the proposed techniques for vehicle navigation are shown. At last, conclusions and some suggestions for future works are given in Chapter 8.

Chapter 2
System Design and Processes

2.1 Idea of System Design

For a blind person to walk safely on a sidewalk, a vision-based autonomous vehicle system is a good substitute for a guide dog, as mentioned previously. Because of the advantages of possessing good mobility and long-time navigation capabilities, autonomous vehicles have become more and more popular in recent years for many applications. Equipped with cameras, an autonomous vehicle is able to "see" like a human being. Moreover, both the autonomous vehicle and the cameras may be connected to a main control system, which have the capabilities to integrate information, analyze data, and make decisions. In this study, we have designed an autonomous vehicle system of this kind for use as a machine guide dog. The entire configuration of the system will be introduced in detail in Section 2.2, and 3D data acquisition using the camera will be described in Section 2.3.

In an unknown environment, the autonomous vehicle system still has to be "trained" before it can navigate by itself. Specifically, it should be "taught" to know the information of the navigation path; how to navigate in this path; and how to handle different conditions on the way. Moreover, secure navigation strategies should also be established for the vehicle to protect the blind and itself. In the end, the vehicle should be able to navigate in the same path repetitively with the learned data and the navigation strategies. The system processes designed to achieve these functions on the proposed autonomous vehicle system will be described in Section 2.3, including the learning process described in Section 2.3.1 and the navigation process in Section 2.3.2.

2.2 System Configuration

In this section, we will introduce the configuration of the proposed system. We use Pioneer 3, an intelligent mobile vehicle made by MobileRobots Inc. as shown in Figure 2.1, as a test bed for this study. The autonomous vehicle and other associated hardware devices will be introduced in more detail in Section 2.2.1. In addition, a particularly-designed *stereo omni-camera* is employed in this study and equipped on the autonomous vehicle. We will describe the structure of the camera system in Section 2.2.2. Finally, the configuration of the software we use as the development tool will be introduced in Section 2.2.3.

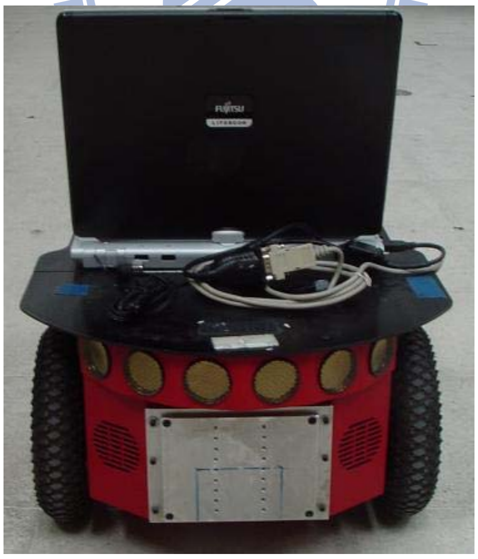
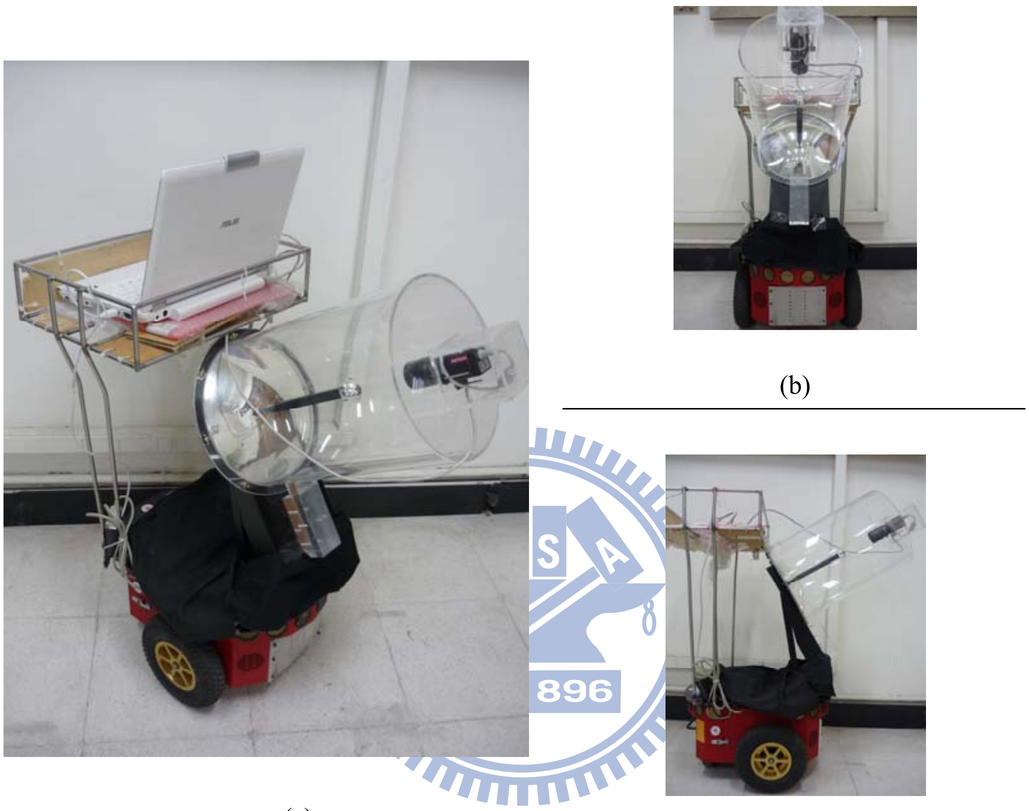


Figure. 2.1 Autonomous vehicle, Pioneer 3 produced by MobileRobots Inc., used in this study.

2.2.1 Hardware configuration

The hardware architecture of the proposed autonomous vehicle system can be divided into three parts. The first is the *vehicle system*; the second the *camera system*, and the third the *control system*. The latter two are installed on the first, the *vehicle system*, as shown in Figure 2.2. We will introduce these systems one by one subsequently.

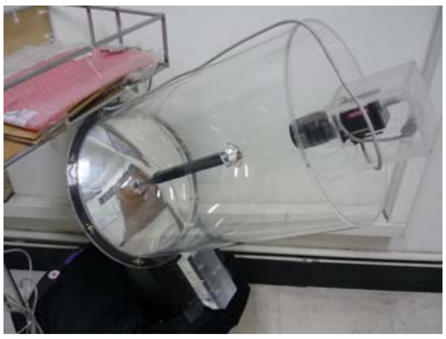


(c)

Figure. 2.2 Three different views of the used autonomous vehicle, which includes a vehicle, a stereo camera, and a notebook PC for use as the control unit. (a) A 45^{o} view. (b) A front view. (c) A side view.

The vehicle, Pioneer 3, has an aluminum body with the size of 44cm×38cm× 22cm, two wheels of the same diameter of 16.5cm, and one caster. Also, 16 ultrasonic sensors are installed on the vehicle, half of them in front of the body and the other half behind. When navigating on flat floors, Pioneer 3 can reach its maximum speed 1.6 meters per second. Also, it has the maximum rotation speed of 300 degrees per second, and can climb up a ramp with the largest slope of 30 degrees. The vehicle is able to carry payloads up to 23kg at a slower speed. It has three 12V rechargeable lead-acid batteries and can run constantly for 18 to 24 hours if all of the batteries are fully charged initially. The vehicle also provides the user some parameter information of the system, such as the vehicle speed, the battery voltage, etc.

The *camera system*, called a *two-mirror omni-camera*, consists of one perspective camera, one lens, and two reflective mirrors of different sizes. The perspective camera, ARCAM-200SO, is produced by the ARTRAY company with a size of 33mm×33mm×50mm and the maximum resolution of 2.0M pixels. With the maximum resolution, the frame rate can reach 8 fps. The CMOS visual sensor in the camera has a size of 1/2 inches (33mm×33mm). 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. A detailed view of the entire camera system is shown in Figure. 2.3, and the camera and the lens are shown in Figures 2.3(a) and 2.4(b), respectively. Other details about the camera structure will be described in the next section.


(a)

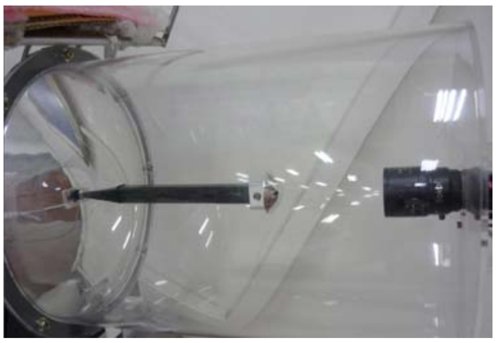

(b)

Figure. 2.3 The two-mirror omni-camera used in this study. (a) A full view of the camera equipped on the vehicle. (b) A closer view.

About the final part, we use a laptop computer as the *control system*. It is of the model of ASUS W7J produced by ASUSTek Computer Inc. as shown in Figure. 2.5. For the computer to communicate with the other parts, we connected it with the autonomous vehicle by an RS-232, and with the camera system by a USB.

(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.

Figure. 2.5 The laptop computer of model ASUS W7J used in this study.

2.2.2 Structure of used two-mirror omni-camera

The structure of the two-mirror omni-camera used in this study and the projection of a space point P onto the system are illustrated in Figure 2.6. We call the bigger and higher mirror *Mirror A*, and the smaller and lower one *Mirror B* subsequently, and both of them are made to be of the shape of a hyperboloid with parameters shown in Table 2.1. One of the focal points of *Mirror A* is on f_a and one of the focal points of *Mirror B* is on f_b. Both mirrors have another focal point on the same position f_c which is the center of the lens. In addition, the line segment $\overleftrightarrow{\mathbf{f_a f_b}}$, which we call the *baseline*, has a length in 20cm.

Table 2.1 Specifications of the used two hyperboloidal-shaped mirrors.

	radius	Parameter a	Parameter b
Mirror A	12 cm	11.46cm	9.68cm
Mirror B	2cm	2.41cm	4.38cm

An important optical property of the hyperboloidal-shaped mirror is: if a light ray goes through one focal point, it must be reflected by the mirror to the other focal point. As illustrated in Figure 2.7, two light rays which go through f_a and f_a are both reflected to the same focal point f_c by the specially-designed mirrors, *Mirrors A* and *B*, respectively. Based on these property of the omni-camera and as illustrated in Figure 2.6, a space point *P* will be projected onto two different positions in the CMOS sensor in the camera along the blue light ray and the red light ray reflected by *Mirrors A* and *B*, respectively, so that the range data of *P* can be computed according to the two resulting distinct image points.

Furthermore, the way of placement of the two-mirror omni-camera on the vehicle has been carefully considered. The camera was originally placed in such a way that the optical axis going through *Mirrors A* and *B* is parallel to the ground as shown in Figure 2.8(a). In the omni-image acquired by the two-mirror omni-camera, because of the existence of the image region caused by *Mirror B*, the image region of the field of view (FOV) reflected by *Mirror A* became smaller. We can see that the overlapping region on the ground where range data can be computed was not large enough in this situation (as shown in the green dotted region in the figure). However, for a navigation system, the front FOV is very important for the vehicle to avoid collisions. Moreover, it is desired that the vehicle can find obstacles at distances as large as possible in the navigation process. Due to these reasons, the camera was later slanted up for an angle of γ as shown in Figure 2.8(b). We can see in the figure that the region of overlapping is now bigger than before.

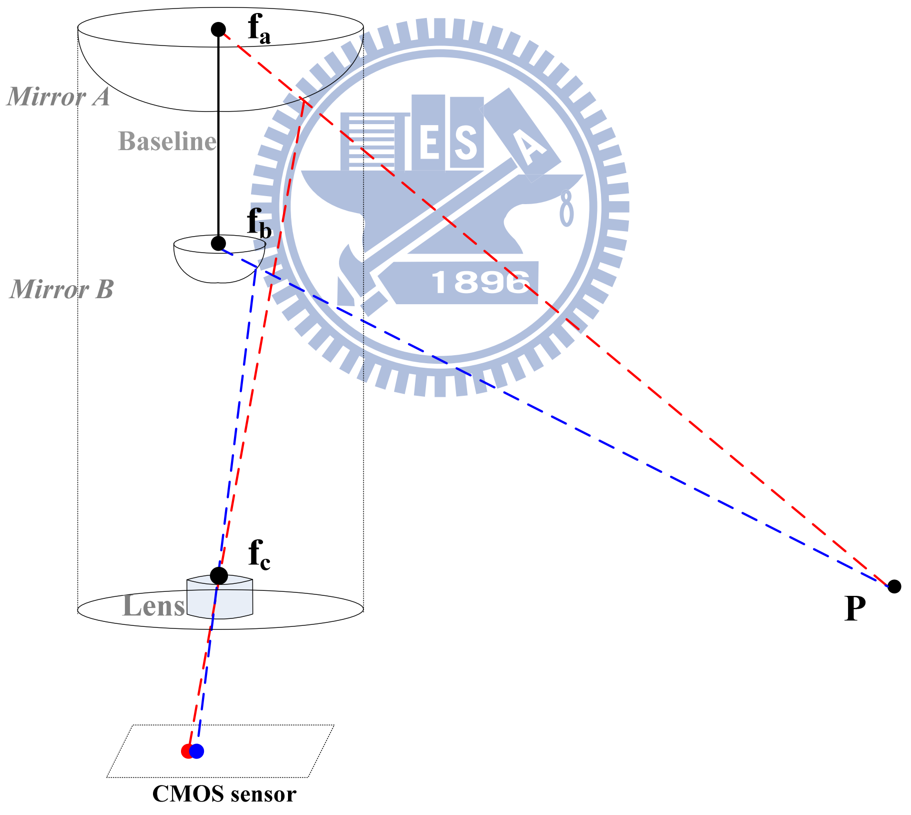


Figure 2.6 The prototype of the two-mirror omni-camera and a space point projected on the CMOS sensor of the camera.

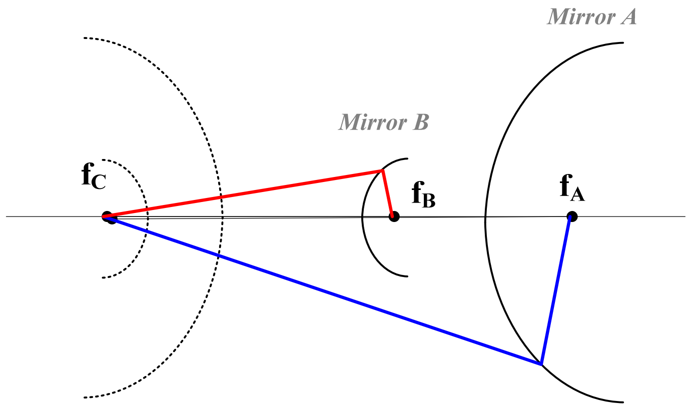
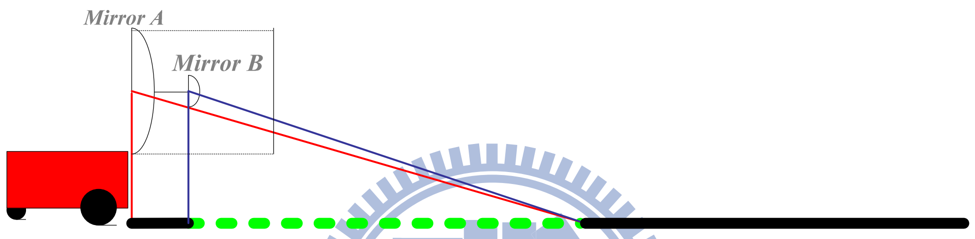
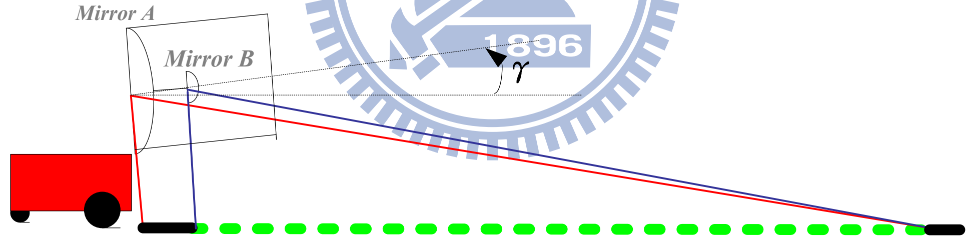


Figure 2.7 The reflection property of the two hyperboloidal-shaped mirrors in the camera system.



(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 .

2.2.3 Software configuration

The producer, MobileRobots Inc., of the autonomous vehicle used 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 Win32 or Linux operating system using the C^{++} language. Therefore, we can utilize the ARIA to communicate with the embedded sensor system in the vehicle and obtain the vehicle state to control the pose of the vehicle.

For the camera system, the ARTAY provides a development tool called *Capture Module Software Developer Kit (SDK)*. This SDK 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 can change many parameters of the camera, such as the exposure. In the control system, we use Borland C++ Builder 6, which is a GUI-based interface development environment, to develop our system processes on the Windows XP operating system.

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

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

Before derivation of the formulas for range data computation by the use of the two-mirror omni-camera, we review first the imaging principle of a simple omni-camera consisting of a hyperboloidal-shaped mirror and a projective camera. First of all, we introduce two coordinate systems as shown in Figure 2.9 where the *image coordinate system* (ICS) is a two-dimensional coordinate system coincident with the omni-image plane with its origin being the center of the omni-image. The *camera coordinate system* (CCS) in Figure 2.9 is a three-dimensional coordinate system with the origin being located at a focal point of the mirror.

According to the optical property of the hyperboloidal-shaped mirror, a space point *P* at coordinates (*x*, *y*, *z*) in the CCS is projected onto an omni-image point *I* at coordinates (*u*, *v*) in the ICS as shown in Figure 2.9. In more detail, assume that a light ray from *P* goes through a focal point in the mirror center O_m. Then, reflected by the hyperboloidal-shaped mirror, the light ray goes through another focal point at the lens center O_c. It is finally projected onto an image point *I* at coordinates (*u*, *v*) on the image plane. In this way of imaging space points, therefore, for each image point *I*, we can find a corresponding light ray with a specific elevation angle α and a specific azimuth angle θ (shown in the figure by the red and the green characters, respectively) to represent the image point *I*.

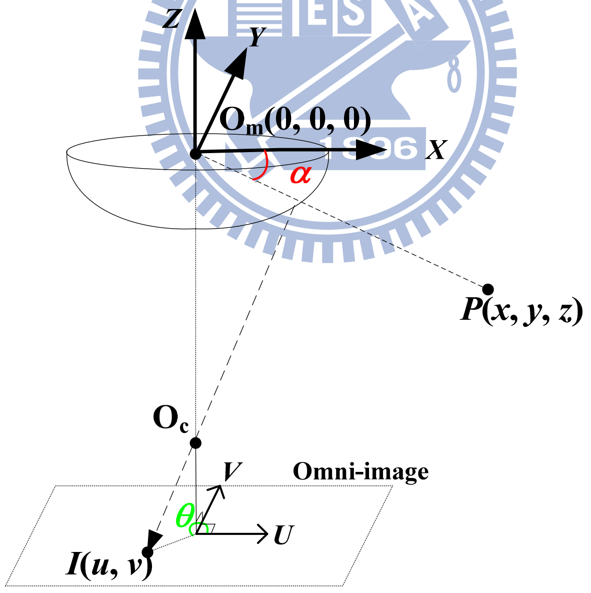


Figure 2.9 Imaging principle of a space point *P* using an omni-camera.

2.3.2 3D data computation for used two-mirror omni-camera

Before deriving the 3D data from the omni-image acquired by the two-mirror omni-camera, we define a camera coordinate system (CCS) CCS_{local} as shown in Figure 2.10. The origin of CCS_{local} is the focal point of *Mirror A*, and the *Z*-axis coincident with the optical axis going through the two mirrors. As shown in the figure, there is a space point Q at coordinates (X, Y, Z) in CCS_{local} which is projected respectively by the two mirrors onto two image points, I_s at coordinates (u_1, v_1) and I_b at coordinates (u_2, v_2), in the omni-image. By the geometry of the camera oprtics, we may compute the 3D position of Q by the following way.

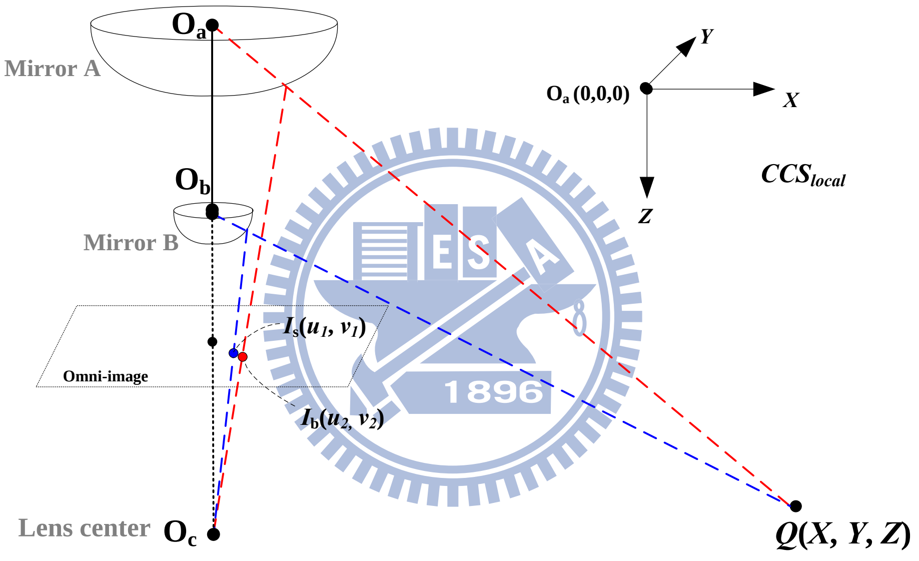


Figure 2.10 The cameras coordinate system CCS_{local} , and a space point Q projected on the omni-image acquired by the two-mirror omni-camera.

Firstly, following the two light rays which go through *Mirror A*'s center and *Mirror B*'s center, respectively, we obtain two different elevation angles α_a and α_b as shown in Figure 2.11(a). Also, the points O_a, O_b, and Q form a triangle ΔO_aO_bQ which we especially illustrate in Figure 2.11(b). The distance between O_a and O_b, which is the length of the baseline defined previously, is known to b, while the distance d between O_a and Q is an unknown parameter. By the *law of sines* based on geometry, we can compute d as follows:

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

$$d = \frac{b \times \sin(90^o + \alpha_b)}{\sin(\alpha_\alpha - \alpha_b)}. \tag{2.2}$$

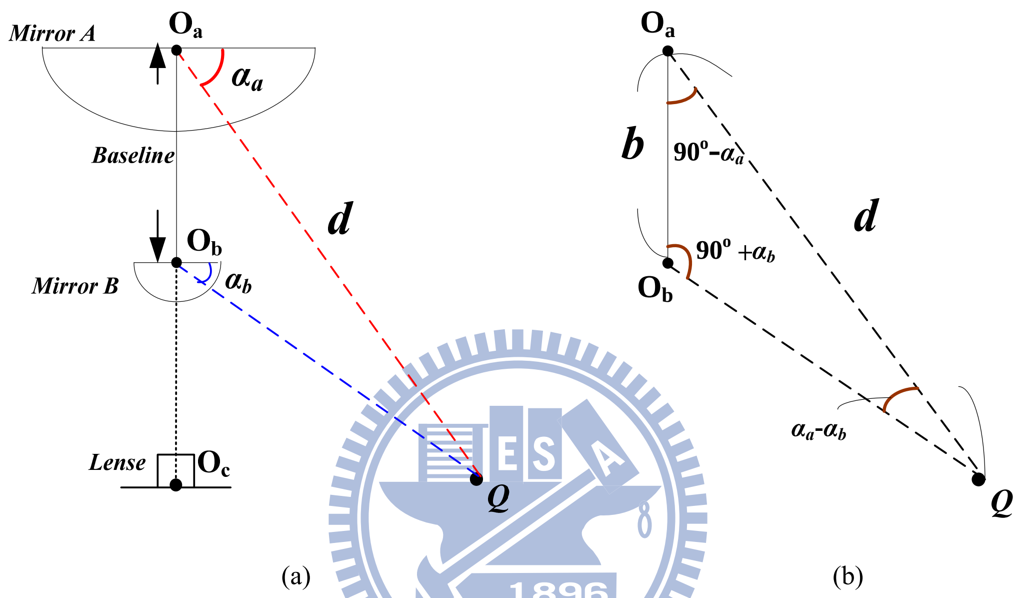


Figure 2.11 An illustration of the relation between a space point Q and the two mirrors in the used camera. (a) A side view of Q projected onto the two mirrors. (b) A triangle ΔO_aO_bQ used in deriving 3D data.

Secondly, we may compute 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, by the use of the image point I_s at coordinates (u_1, v_1), we can derive θ by the following equations:

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

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

Thirdly, with the distance d derived by Equation (2.2) and the azimuth angle θ obtained by Equation (2.3), we can compute the position of Q in CCS_{local} according to geometry illustrated in Figure 2.12 as follows:

$$\begin{aligned} X &= d \times \cos\alpha_a \times \sin\theta, \\ Y &= d \times \cos\alpha_a \times \cos\theta, \\ Z &= d \times \sin\alpha_a. \end{aligned} \tag{2.4}$$

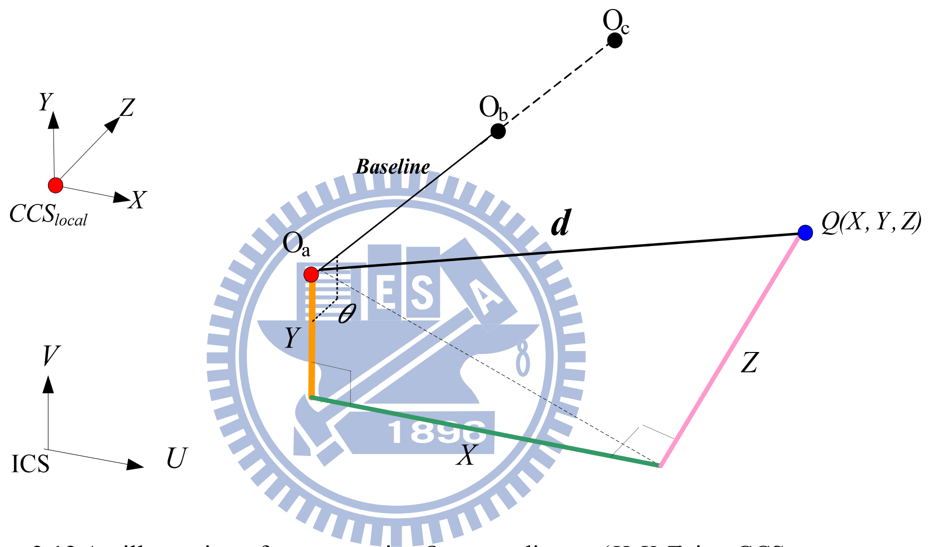


Figure 2.12 An illustration of a space point Q at coordinates (X, Y, Z) in CCS_{local}.

However, as mentioned previously, the optical axis going through the two mirrors is slanted up so that it is not parallel to the ground. It is desired that the Z-axis of CCS_{local} could be parallel to the ground. As shown in Figure 2.13, we define another camera coordinate system CCS, which coincidences with CCS_{local} except that the Z-axis is slanted for an angle of γ toward the Y-axis along the X-axis. Finally, the coordinates of Q is translated to a new coordinates (X', Y', Z'), which we want to obtain, in the CCS by the use of a rotation matrix R by following equations:

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

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

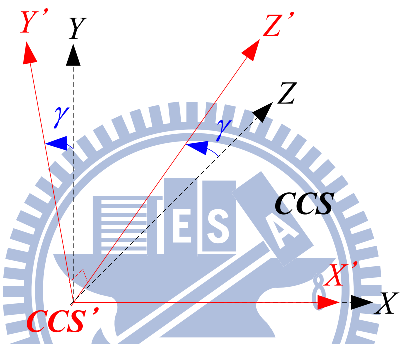


Figure 2.13 The relation between the two camera coordinate systems *CCS* and CCS_{local} .

2.4 System Processes

2.4.1 Learning process

The goal in the learning process is to “teach” the autonomous vehicle to know how to navigate automatically in a pre-specified path. The entire learning process proposed in this study is shown in Figure 2.14. Discussed in the following is some information which the vehicle should “memorize.” First, as mentioned in Chapter 1, the autonomous vehicle has to conduct self-localization by some pre-selected landmarks in the specified path, so the first type of information the vehicle have to record is the landmark locations along the path. Next, for our study, the experimental environment is on the sidewalk, and this has an advantage that the vehicle may navigate along the curb of the sidewalk. For this reason, line following along the sidewalk curb, called *sidewalk following*, is a proper navigation method instead of using the odometer only. Thus, the second type of information which has to be recorded is the vehicle navigation information along the path (path nodes, the navigation distance between two nodes, etc.). Finally, the environment information at different locations on the navigation path also has to be recorded.

For the purpose of training an autonomous vehicle easily, a user learning interface is constructed for the *trainer* and can be used to control the autonomous vehicle as well as construct learning navigation information. First, at the beginning of each section of the navigation path, the trainer should establish a set of corresponding navigation rules in advance, and the vehicle will follow them and conduct navigation in the learning process as well as in the navigation process. Then, the current vehicle pose obtained from the odometer and some current environment information like the illumination are also recorded. Next, when the mode, *navigation by following the sidewalk,* is selected, a *semi-automatic* learning process will proceed until reaching the next node assigned by the *trainer*. Otherwise, the trainer is required to guide the vehicle manually to the next path node by the use of the learning interface.

In addition, the trainer can decide where to localize the vehicle by a selected landmark in the learning process. After guiding the vehicle to a proper pose for detecting the landmark (close enough to the landmark, "looking" at the landmark from the right direction, etc.), the trainer then has to establish relevant rules for landmark detection. Some parameters for landmark detection can be appropriately adjusted by the trainer before the detection work is started. Next, landmark localization is conducted by a space line detection technique described in Chapter 5. After possibly multiple times of detecting and collecting adequate information of the landmark, its position is finally computed automatically and recorded.

At last, after bringing the autonomous vehicle to the destination, the learning process is finished, and the learned information is organized into a *learned path* composed of several path nodes with guidance parameters. Combining it with landmark information and environment information, we obtained an integrated *path map* which finally is stored in the memory of the vehicle navigation system.

2.4.2 Navigation process

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

According to the learned information data retrieved from the storage, the autonomous vehicle continually analyzes the current environment *node by node* to navigate to the goal. At first, before starting to navigate to the next node, the autonomous vehicle checks if the image frame is too dark or too bright according to the learned environment parameter data, and then dynamically adjusts the exposure of the camera if necessary.

After that, 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. Then, based on the learned navigation rules, the autonomous vehicle checks the corresponding navigation mode and follows it to navigate forward. In the meantime, the vehicle checks whether it has arrived at the next node; whether the node is the destination; or whether the vehicle has to localize its current position.

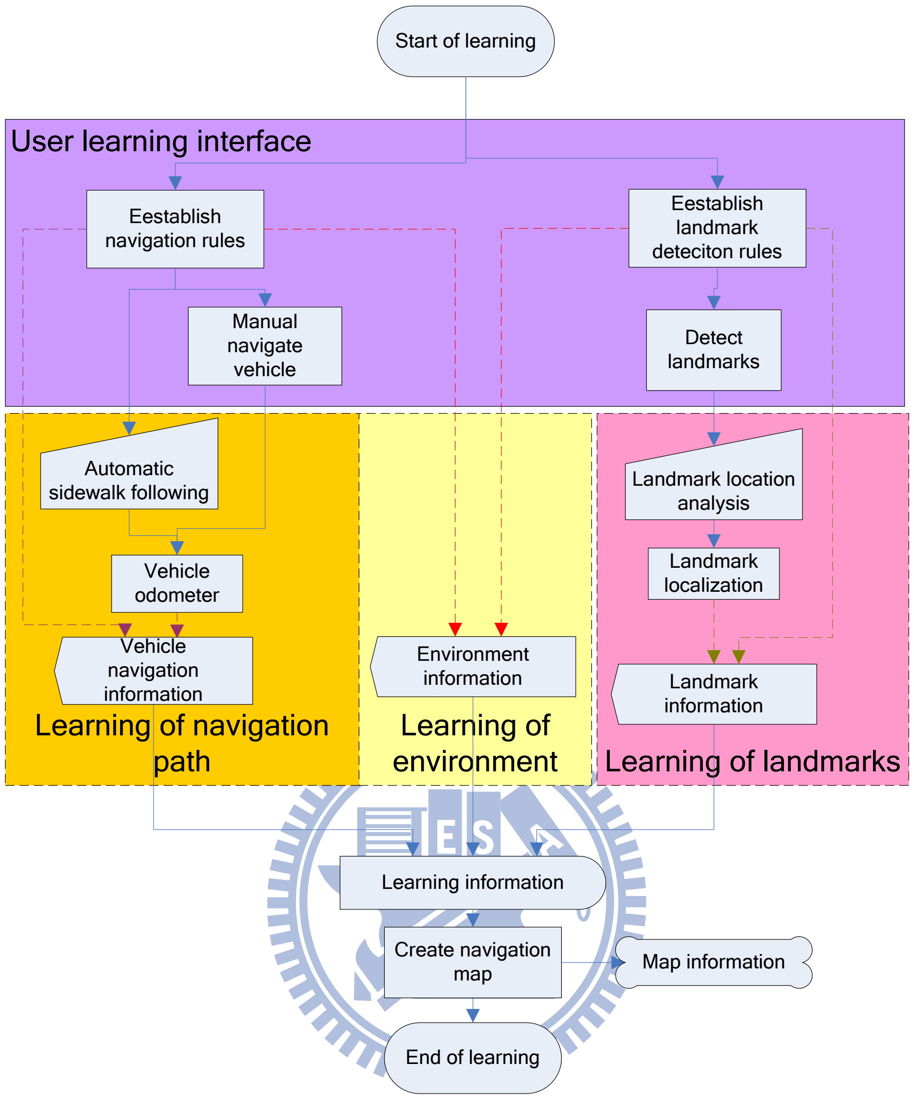


Figure 2.14 Learning process.

In addition, if a self-localization node is expected, the autonomous vehicle will adjust its pose and relevant parameters into an appropriate condition and conduct landmark detection. For landmark detection, the autonomous vehicle uses the corresponding technique in accordance with the property of the landmark. If a desired landmark is found and localized successfully, its location then is used to modify the position of the vehicle; if not, some remedy for recovering the landmark will be conducted, such as changing the pose of the vehicle to detect the landmark again.

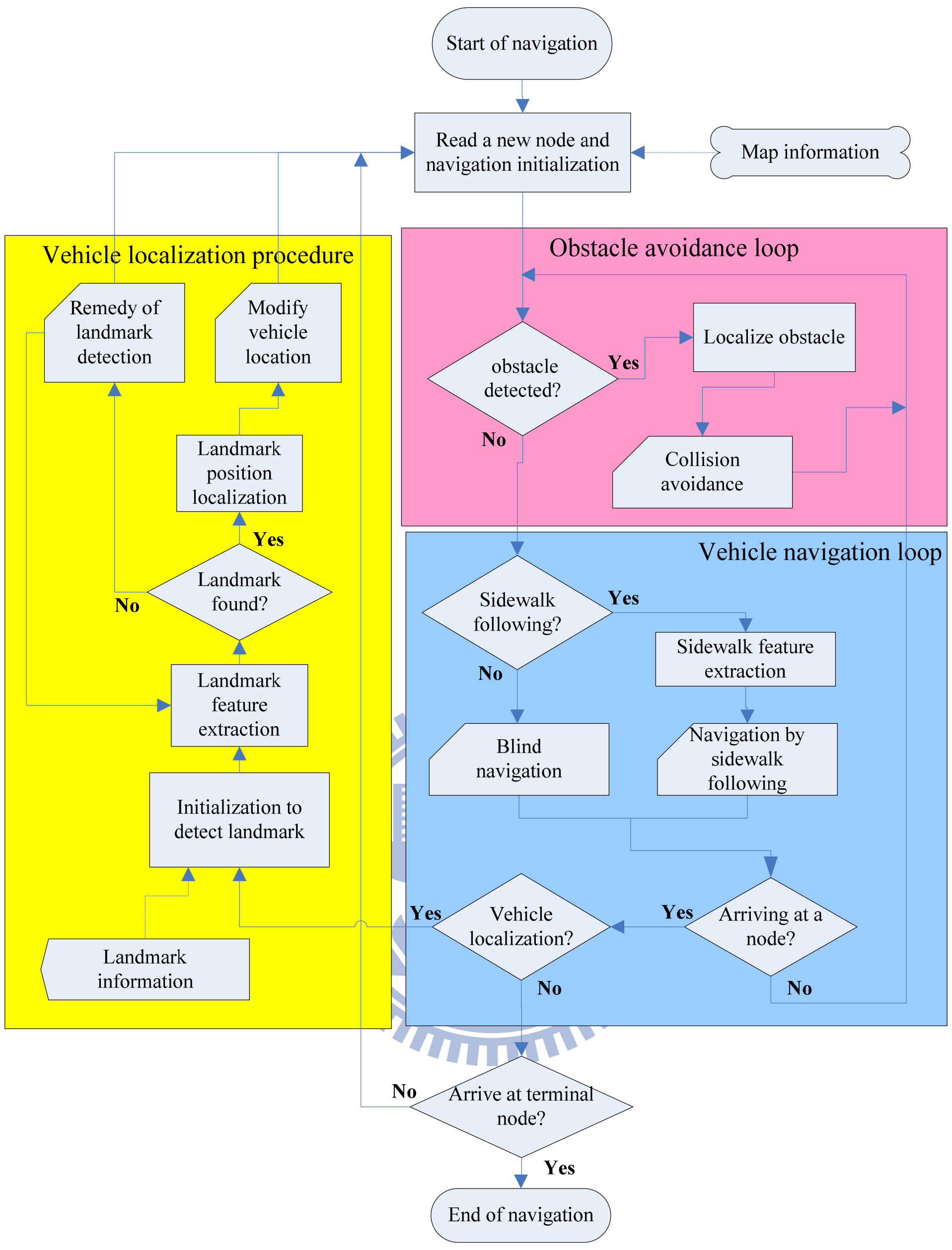


Figure 2.15 Navigation process.

Chapter 3
Learning Guidance Parameters and Navigation Paths

3.1 Introduction

Before the autonomous vehicle can navigate, some works has to be conducted in the learning process. First, the camera system should be calibrated. Then, a path and a set of landmarks should be selected, and each landmark location should be recorded into the path, resulting in a *learned path*. Finally, adopted guidance parameters have to be "trained" and then recorded.

3.1.1 Camera calibration

As mentioned in Chapter 1, instead of calibrating the camera's intrinsic and extrinsic parameters, we adopt a space-mapping technique [25], called pano-mapping, to calibrate the two-mirror omni-camera used in this study. We will describe the adopted technique in Section 3.2.

3.1.2 Selection of landmarks for navigation guidance

For the purpose to localize the position of the vehicle during the navigation process, some objects should be selected as landmarks to conduct vehicle localization. Two types of objects, hydrant and light pole, as shown in Fig 3.1 are selected in this study as landmarks for vehicle localization during sidewalk navigation. By the use of proposed hydrant and light pole localization techniques, introduced later in Chapter 5, we can guide the vehicle to learn the positions of pre-selected hydrants and light poles in the learning process.

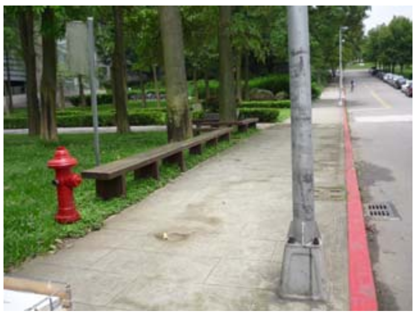


Figure 3.1 The hydrant (left) and the light pole (right) used as landmarks in this study.

3.1.3 Learning of guidance parameters

For complicated outdoor environments, the trainer should train some parameters for use in vehicle guidance, such as environment parameters and image segmentation thresholds, in the learning process. We will introduce the proposed techniques for learning environment parameters in Sections 3.4. Also, some image segmentation parameters for landmark image analysis and the techniques proposed to learn them will be introduced in Sections 3.5. Finally, a scheme proposed to create the learned navigation path will be described in Section 3.6.

3.2 Camera Calibration by Space-mapping Approach

We utilize the pano-mapping technique proposed by Jeng and Tsai [25] for image unwarping to calibrate the camera system used in this study. The main idea is to establish a so-called *pano-mapping table* to record the relation between image points and corresponding real-world points. More specifically, as illustrated in Figure 3.2, 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 image point *p* and the azimuth-elevation angle pair (*θ*, *α*) of the corresponding world-space point *P*. The table is established in advance and can be looked up to retrieve 3D information forever. Accordingly, we construct two pano-mapping tables for *Mirrors A* and *B*, respectively, by the following steps, assuming an omni-image *I* has been taken as the input.

Algorithm 3.1 Construction of pano-mapping tables.

Step 1. Manually select in advance six known image points p_i at coordinates (u_i, v_i,) on the *Mirror A region* in omni-image *I* and the six corresponding known world-space points P_i at coordinates (x_i, y_i, z_i), where *i* is 1 through 6.

Step 2. Select similarly six known image points q_j at coordinates (U_j, V_j,) on the *Mirror B* region in omni-image *I* and the six corresponding known world-space points Q_j at coordinates (X_j, Y_j, Z_j), where *j* is 1 through 6.

Step 3. For image points p_i and q_j, compute the radial distances r_i and R_j in the image plane with respect to the image center respectively by the following equations:

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

Step 4. Compute the elevation angles α_i and β_i for the corresponding world-space points P_j and Q_j 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 radial distances and corresponding elevation angles for *Mirrors A* and *B*, respectively.

Step 5. Under the assumption that the surface geometries of *Mirrors A* and *B* are radially symmetric in the range of 360 degrees, use two *radial stretching functions*, denoted as f_A 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:

$$r_i = f_A(\alpha_i) = a_0 + a_1 \times \alpha_i^{\,1} + a_2 \times \alpha_i^{\,2} + a_3 \times \alpha_i^{\,3} + a_4 \times \alpha_i^{\,4} + a_5 \times \alpha_i^{\,5};$$

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

Step 6. Solve the above 6-th degree polynomial equations f_A and f_B by the use of the six radial-distance pairs for *Mirrors A* and *B*, respectively, obtained in Step 4 using 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_A with the known coefficients a_0 through a_5, construct the pano-mapping table for *Mirror A* in a form as that shown in Figure 3.3(a) according to the following rule:

for each world-space point P_{ij} with the azimuth-elevation pair (θ_i, α_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 the pano-mapping table for *Mirror B* by the use of the function f_B with the known coefficients b_0 through b_5 in a form as that shown in Figure 3.3(b).

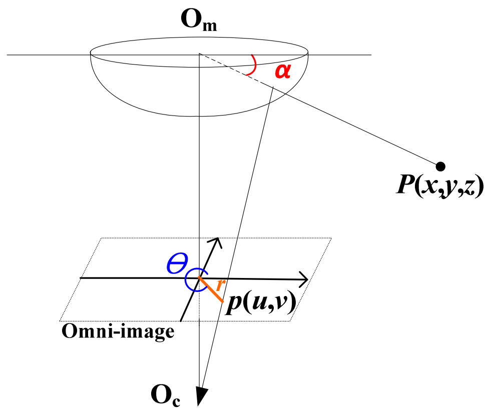


Figure 3.2 The relation between a space point *P* and the relevant elevation angle and azimuth.

	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})

(a)

	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})

(b)

Figure 3.3 Two pano-mapping tables used for the two-mirror omni-camera used in this study. (a) Pano-mapping table used for *Mirror A*. (b) Pano-mapping used for *Mirror B*.

3.3 Coordinate Systems

In this section, we will introduce the coordinate systems used in this study, which describe the relations between the used devices and concerned landmarks in the navigation environment. Furthermore, the used odometer and some involved coordinate transformations are introduced also. The following are four coordinate systems used in this study.

(1). Image coordinate system (ICS): denoted as (u, v). The u-v plane coincides with

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

(3). Global coordinate system (GCS): denoted as (M_X, M_Y). The M_X-M_Y plane coincides with the ground. The origin O_G of this system is always placed at the start position of the vehicle in the navigation path.

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

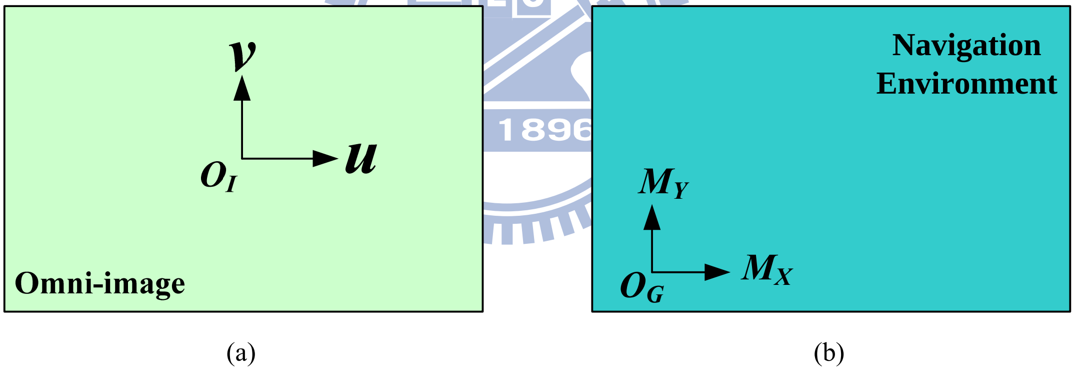


Figure 3.4 Two coordinate systems used in this study. (a) The ICS. (b) The GCS.

In this study, the navigation path is specified by the GCS as shown in Figure 3.4(b). The relationship between the GCS and the VCS is illustrated in Figure 3.5. At the beginning of the navigation, the VCS coincides with the GCS, and then the VCS follows the movement of the current vehicle position as well as the CCS. In addition, it is emphasized that the vehicle uses an odometer to localize its position in the GCS. As illustrated in the figure, the reading of the vehicle odometer is denoted as (P_x, P_y, P_{th}) where P_x and P_y represent the current vehicle position with respect to its original position on the ground, and P_{th} represents the rotation angle of the vehicle axis with respect to the GCS.

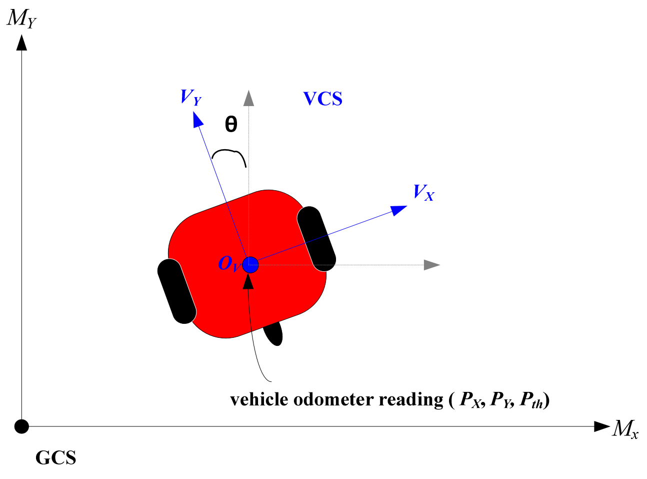
As shown in Figure 3.6, assume that the vehicle is at a position V at world coordinates (C_x, C_y) with a rotation angle θ. We can derive the coordinate transformation between the coordinates (M_X, M_Y) of the VCS and the coordinates (V_X, V_Y) of the GCS by the following equations:

$$M_X = V_X \times \cos\theta - V_Y \times \sin\theta + C_x;$$

$$M_Y = V_X \times \sin\theta - V_Y \times \cos\theta + C_y. \tag{3.5}$$

In addition, the relationship between the CCS and the VCS is illustrated in Figure 3.7. As shown in the figure, the projection of the origin of the CCS onto the ground does not coincident with the origin of the VCS, and there is a horizontal distance between the two origins, which we denote as S_y. Thus, the coordinate transformation between the CCS and the VCS can be derived in the following way:

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



jiFigure 3.5 An illustration of the relation between the GCS and the VCS.

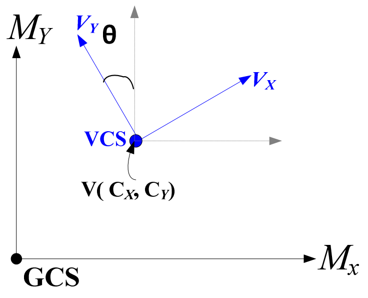


Figure 3.6 A vehicle at coordinates (C_x, C_y) with a rotation angle θ with respect to the GCS.

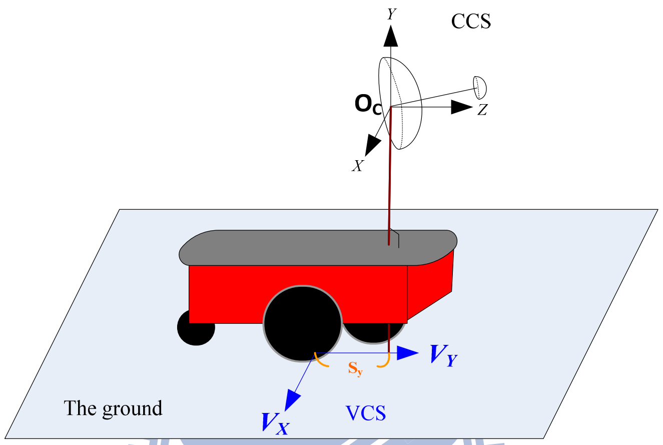


Figure 3.7 An illustration of the relation between the GCS and the VCS.

3.4 Learning of Environment Parameters

3.4.1 Definition of environment windows on images

In the process of navigation, the vehicle conducts several works including following sidewalk curbs, finding landmarks, obstacle detection, etc. In general, each desired landmark is projected onto a specific region in the image. By this property, we can consider only the region of interest in the image instead of the whole image, and two advantages can be obtained from this approach as follows:

1. reducing computation time for each navigation cycle;

2. the environment appearing within the image region is similar in each navigation cycle so that we can analyze the environment information in this region with a fixed scheme.

To be more specific, an *environment window,* as we call hereafter, is predefined by the trainer, which specifies a rectangular region in the image. Two *environment windows* including a small one in the region of *Mirror B* and a big one in the region of *Mirror A,* denoted as win_S *and* win_B, respectively, are considered to form a set for use in conducting specific image analysis works when the vehicle is navigating. For instance, a set of environment windows is defined for the hydrant detection work, as shown is Figure 3.8 (the blue rectangles). Besides, it is pointed out that this scheme of defining a pair of search windows for use in this study follows the property of rotational invariance of the omni-image, which is useful to reduce the redundancy region where we cannot get relevant 3D information, as will be elaborated later in this section.

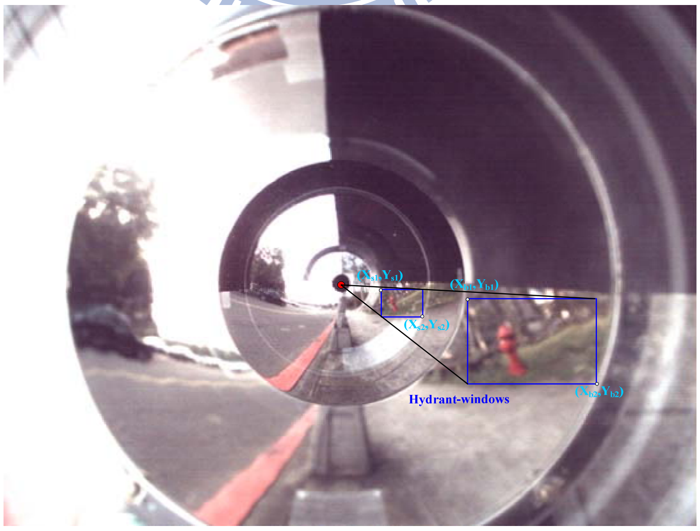


Figure 3.8 An example of a pair of environment windows for hydrant detection

3.4.2 Learning of environment intensity by environment windows

In outdoor environments, varying lighting conditions influence the results of our navigation environment analysis work, according to our experimental experience. For example, as shown in Figure 3.9(a), the feature of the curb is not obvious enough to be recognized because of the overexposure due to the lighting condition. Also, using a fixed value of the exposure, some landmark detection works cannot be completed successfully. For example, the hydrant and the light pole become undetectable in images because of the overexposure and underexposure, as shown the examples in Figures 3.10(a) and 3.11(a), respectively.

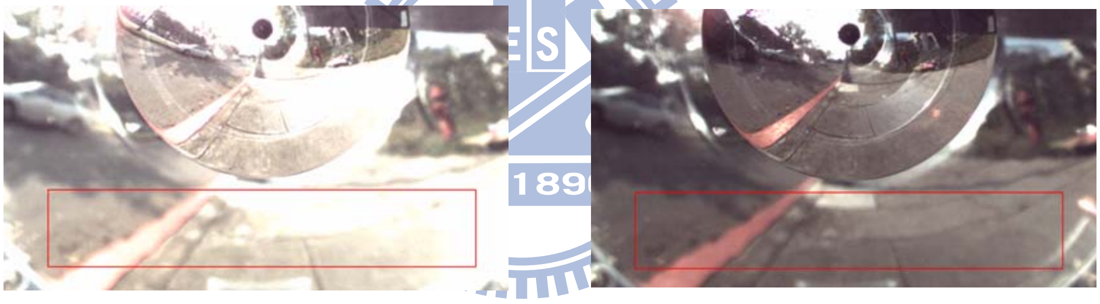

(a) (b)

Figure 3.9 Two different illuminations in the image for curb detection and the environment windows. (a) An instance of overexposure. (B) A suitable case.

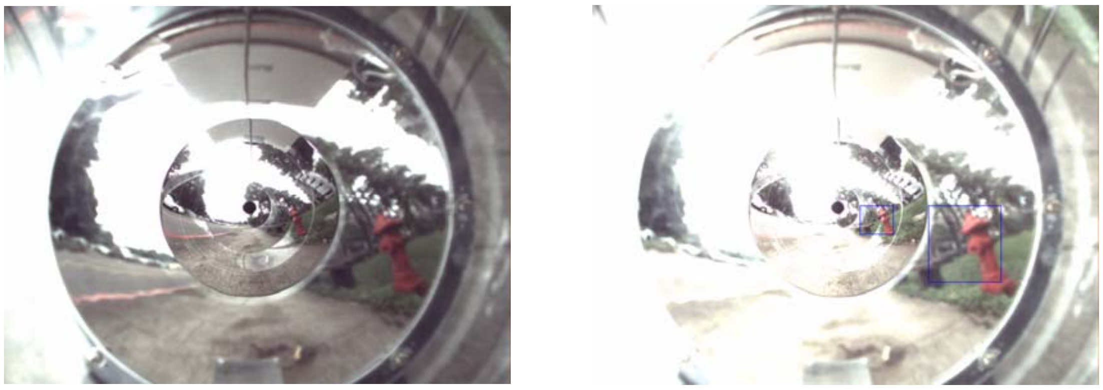
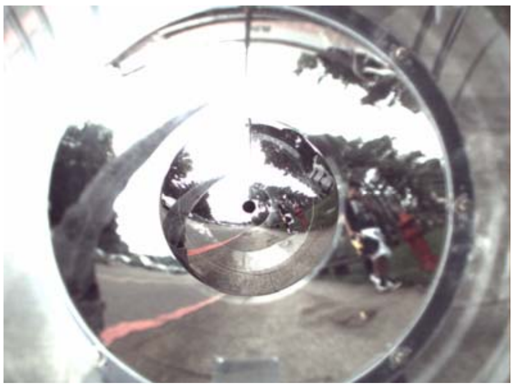

(a) (b)

Figure 3.10 Two different illuminations in the image for hydrant detection and the environment windows. (a) An unclear case. (B) An appropriate case.


(a)

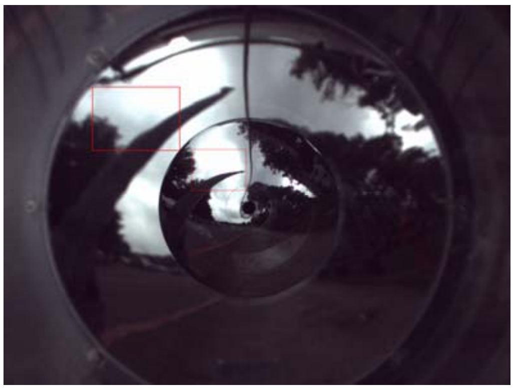

(b)

Figure 3.11 Two different illuminations in the image for light pole detection and the environment windows. (a) A blurred case. (B) A proper case.

For this reason, in this study we design the system in such a way to allow the trainer in the learning phase to determine a suitable illumination parameter by manually adjusting the exposure of the camera for the purpose to detect desired objects successfully. By adjusting the illumination parameter to a suitable value, we mean that the desired landmark feature can be extracted well in the same illumination afterward. Then, this image illumination parameter is recorded into the path information as part of the learning result. To be more specific, for each environment analysis work using a landmark, we learn a value of suitable image intensity, called *environment intensity* hereafter, on the image in relevant environment windows during the path learning process. The detail of this scheme of getting proper environment intensity parameters is described in the following algorithm.

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

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

Output: an environment intensity parameter I_{en}.

Steps.

Step 1. Adjust the exposure of the camera and acquire an appropriate image I_{cur}.

Step 2. Check if the desired landmark feature is well imaged in the resulting illumination. If not, go to Step 1.

Step 3. For each pixel I_i in I_{cur} with color (R, G, B) in win_B of Win_{en}, 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. Compute the value I_{en} as output by the use of the data in S_Y in the following way 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 3.9(b), 3.10(b), and 3.11(b), and 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.

3.5 Learning of Landmark Segmentation Parameters

In this study, we utilize some segmentation methods for image analysis in landmark detection which we describe in detail later in Chapters 5 and 6. In this section, we introduce the process for learning the parameters used in landmark segmentation. Firstly, we introduce three sets of segmentation parameters for landmark segmentation which we propose for use in this study as follows.

(1) In 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. The thresholds for hue and saturation values are collected as a set of *curb segmentation parameters*.

(2) In hydrant segmentation, just like what we do in sidewalk curb detection, we use the HSI color model to extract the hydrant shape. The threshold values for hue and saturation and also the contour of the hydrant described by the principal components obtained from principal component analysis is collected as a set of *hydrant segmentation parameters*.

(3) In light pole segmentation, we adopt the Canny edge detection technique to extract the light pole shape. The threshold values used to detect the light pole in the image are collected as a set of *light pole segmentation parameters*.

Next, as shown in Figure 3.12, when conducting landmark learning, the trainer can detect a desired landmark by the use of a user interface of the system, and adjust the values of the related set of segmentation parameters. After obtaining a proper result from the landmark detection process, the used set of segmentation parameters and the learned landmark information are recoded together as part of the learned path.

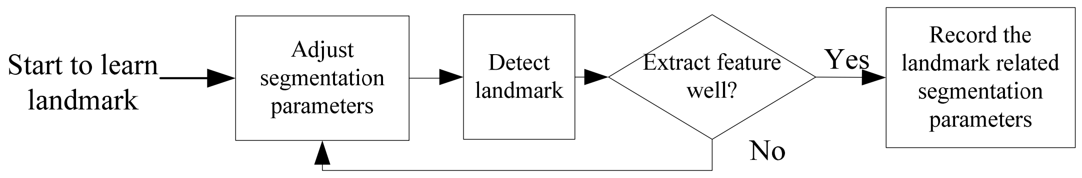

Start to learn landmark
Adjust segmentation parameters
Detect landmark
Extract feature well?
Yes
No
Record the landmark related segmentation parameters

Figure 3.12 The process for learning landmark segmentation parameters.

3.6 Learning Processes for Creating a Navigation Path

In this section, we introduce the proposed method for learning a navigation path in the learning process. As usual, we use the odometer to localize the vehicle position and estimate the position of a detected landmark in the learning process. The proposed strategy for learning landmarks for vehicle localization is introduced in Section 3.6.1. In addition, in the navigation path, some obstacles on the sidewalk, which may not be recognized easily by the camera system, could also block the vehicle. An example of obstacles, a sewer cover with uprising handles, which might hinder autonomous vehicle navigation, is shown in Figure 3.13. Thus, we propose in this study a method to learn the positions of such obstacles, called *fixed obstacles* hereafter. The method is described in Section 3.6.2. Finally, the entire proposed procedure to learn a navigation path is described in Section 3.6.3.

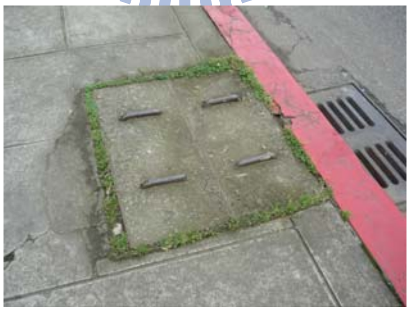


Figure 3.13 A fixed obstacle in a navigation path which may block the autonomous vehicle.

3.6.1 Strategy for learning landmark positions and related vehicle poses

In this section, we introduce the proposed strategy for learning a landmark and its position. Simply speaking, for a landmark to be learned well, we guide the vehicle to appropriate positions to detect it. To increase the accuracy of the learned landmark position, we take images of the landmark a number of times from a number of different directions after guiding the vehicle to a number of different locations. The reason why we take multiple images from a fixed direction at a fixed position is that the weather 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 analyzing the collected multiple images, a more precise landmark position can be obtained, which, together with the corresponding vehicle pose (including the vehicle position and orientation on the path), is recorded as part of the learned navigation path.

To be more specific, after detecting the landmark in acquired omni-images for a multiple times with the vehicle in a number of poses, we 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 all zero, 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 landmark type, localize the landmark by the use of the corresponding localization technique (described in subsequent chapters) 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:

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

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 pose $P_{vehicle}$ as $P_{vehicle} = (Px_c, Py_c, Th_c)$.

3.6.2 Learning of fixed obstacles in a navigation path

In this study, we propose a function in the learning interface which can be used to learn fixed obstacles. After guiding the vehicle to a proper location where a fixed obstacle is projected onto the image region of both *Mirrors A* and *B*, we utilize this function to learn the fixed obstacle. As shown in Figure 3.14, we can select the fixed obstacle in the image by using the mouse to click two corresponding fixed obstacle positions on the regions of *Mirrors A* and *B*. In the mean time, a pair of selected mutually-corresponding image points is recorded into a set for use later to analyze the learned position of the fixed obstacle. Finally, after selecting sufficient obstacle points in the image, the fixed obstacle position W_{obs} is computed automatically. This fixed obstacle position W_{obs} and some parameters for avoiding it are recorded together as part of the learned path information. The detailed algorithm for learning a fixed obstacle's position W_{obs} is described in the following.

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

Input: an image I_{input}, and a set S_{obs} of N pairs of corresponding image points, denoted as $a_i = (u_{1i}, v_{1i})$ and $b_i = (u_{2i}, v_{2i})$, where $i = 1$ through N.

Output: a fixed obstacle position W_{obs}.

Steps.

Step 1. Select manually a fixed obstacle point a_i at coordinates (u_{1i}, v_{1i}) in the region of *Mirror A* in I_{input} and record a_i.

Step 2. Manually select the corresponding fixed obstacle point b_i at coordinates (u_{2i}, v_{2i}) in the region of *Mirror B* in I_{input} and record b_i.

Step 3. Repeat Steps 1 and 2 for N times.

Step 4. For a pair of the corresponding points a_i and b_i, compute the 3D position (c_{xi}, c_{yi}, c_{zi}) of the corresponding point C_i in the CCS by the derivations

Step 5. Compute the position (x_i, y_i) of the corresponding point V_i on the ground in the WCS by 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), and record V_i into a set V_{obs}.

Step 6. Repeat Steps 4 and 5 for N times.

Step 7. Derive the position (obs_x, obs_y) of point W_{obs} in the WCS as the location of the obstacle by the following equations:

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

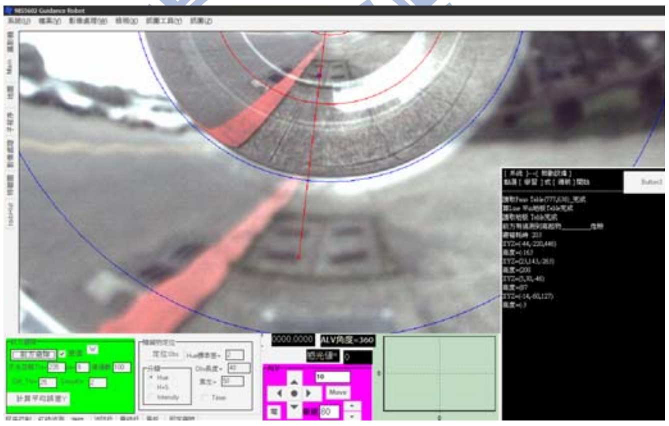


Figure 3.14 A learning interface for the trainer to learn the position of a fixed obstacle by clicking the mouse on a pair of corresponding obstacle points in the image regions of *Mirrors A* and *B*.

3.6.3 Learning procedure for navigation path creation

In this section, we describe how we establish a navigation path in the learning process. Firstly, we define eight types of navigation nodes as listed in Table 3.1, where each navigation node includes a set of different appointed works which have to be conducted by the vehicle, or a set of data representing a landmark position in the navigation path. We guide the vehicle to learn a pre-selected navigation path as well as some pre-selected landmarks by the use of these navigation nodes to construct a learned navigation path. In addition, while each navigation node is recorded, some relevant guidance parameters are also recorded into the learning result. At the end of the learning process, a navigation path consisting of a series of navigation nodes and relevant guidance parameters is recorded, which then can be utilized for vehicle navigation in the navigation process. A flowchart of the process for navigation path creation is shown in Figure 3.15, and the detailed algorithm to implement it is described in the following.

Table 3.1 Eight 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	Light-pole landmark node
Type 6	Hydrant landmark node
Type 7	Fixed obstacle node

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 vehicle direction, 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 guide the vehicle to navigate forward until arriving at a desired destination 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 four following *additional* learning tasks.

(1) Learn a hydrant landmark by the method mentioned in Section 3.3, obtain a hydrant position N_{hyd} and the related vehicle pose N_{car}, and record N_{car} in *Type* 4 and N_{hyd} in *Type* 6 into N_{path}.

(2) Learn a light pole landmark by the method mentioned in Section 3.3 and obtain a light pole position N_{lp} and the related vehicle poses N_{car}, and record N_{car} in *Type* 4 and N_{lp} in *Type* 5 into N_{path}.

(3) Learn a fixed obstacle N_{obs} using the proposed function discussed in Section 3.4, and record N_{obs} in *Type* 7 into N_{path}.

(4) 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 (with the detail introduced in Section 4.2.2), 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}.

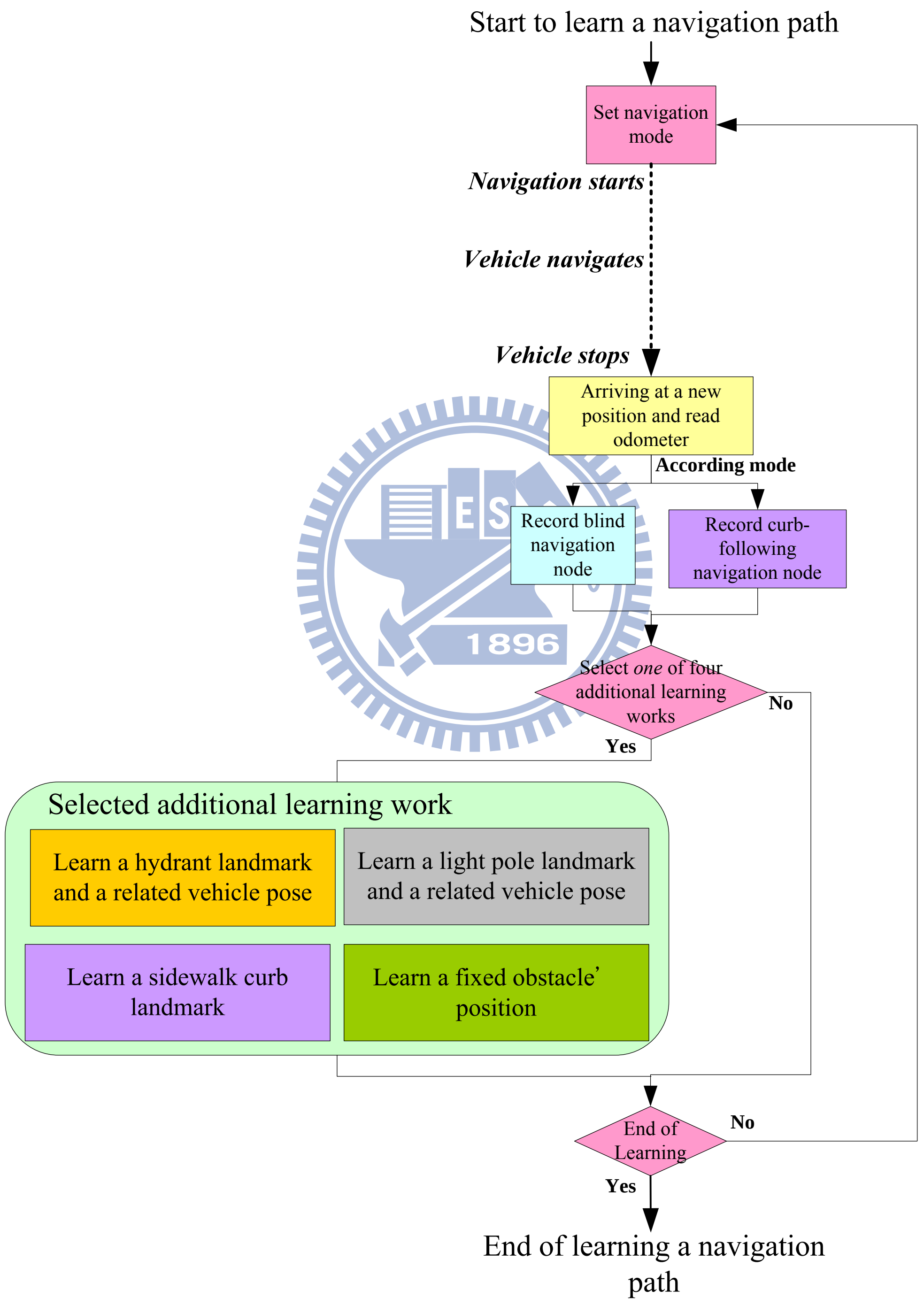

Start to learn a navigation path
Set navigation mode
Navigation starts
Vehicle navigates
Vehicle stops
Arriving at a new position and read odometer
According mode
Record blind navigation node
Record curb-following navigation node
Select one of four additional learning works
No
Yes
Selected additional learning work
Learn a hydrant landmark and a related vehicle pose
Learn a light pole landmark and a related vehicle pose
Learn a sidewalk curb landmark
Learn a fixed obstacle' position
End of Learning
No
Yes
End of learning a navigation path

Figure 3.15 The process for navigation path creation.

Chapter 4
Navigation Strategy in Outdoor Environments

4.1 Idea of Proposed Navigation Strategy

After successfully learning the navigation environment, we acquire the learned environment information including a navigation path and other guidance parameters. In this chapter, we introduce the proposed strategies for vehicle navigation in complicated outdoor environments by use of this information. The proposed principles to conduct the navigation work are introduced in Section 4.2.1. The process for navigation is described in Section 4.2.3. In addition, three main ideas to guide the vehicle to navigate on the learned path in this study follows.

4.1.1 Vehicle localization by alone-path objects

As mentioned previously, the vehicle navigation process usually suffers from incremental mechanic errors, resulting in imprecise computations of vehicle positions, so the vehicle should be guided to constantly localize its position by the learned landmark position. After localizing a landmark by the use of proposed localization techniques introduced later in Chapter 5 and obtaining the relative vehicle position with respect to the landmark, we can adjust the vehicle posture by changing its position and orientation using vehicle commands and correcting the odometer readings. In addition, we also used the learned straight curb line segment on the sidewalk to calibrate the vehicle posture. Theses proposed techniques to adjust the vehicle posture in the navigation path are introduced in Section 4.2.2.

4.1.2 Dynamic adjustment of guidance parameters

In complicated outdoor environments, we cannot only adopt fixed guidance parameters recorded in the learning process to conduct image analysis works, resulting in varying lighting. Thus, we taught the vehicle in the learning process to analyze environment data and then utilize learned methods to adjust guidance parameters. Some techniques for dynamic guidance parameters adjustment are proposed in this study. First, the learned contour of the hydrant helps the vehicle to adjust the segmentation parameters by principal component analysis (PCA). Also, by estimating the result of curb contour extraction, we can adjust the curb segmentation parameters. The above two techniques of dynamic adjustment of thresholds for hydrant and curb detection are introduced later in Chapters 5 and 6, respectively.

In addition, we use a dynamic exposure adjustment scheme to deal with the varying lighting condition in the outdoor environment during the vehicle navigation process. An advantage of dynamic exposure is the possibility to preserve more usable color information of the object in the image. According to the environment intensity parameter learned in the learning process for each work, we can determine whether the current luminance of the image frame is suitable, and the technique will be automatically enforced if necessary. The proposed technique for dynamic exposure adjustment is introduced in Section 4.2.3.

4.1.3 Obstacle avoidance by 3D information

For vehicle navigation in outdoor environments, encountering an obstacle is unavoidable and must be found to dodge it. By using a stereo camera in this study, we propose a dynamic obstacle detection technique via the use of 3D information. We use this technique to conduct secure navigation. The detail about this technique is described later in Chapter 6.

4.2 Guidance Technique in Navigation Process

4.2.1 Principle of navigation process

In this section, we introduce the principles of the proposed vehicle navigation method on the learned path. At the beginning, the vehicle retrieves a navigation path and related guidance parameters which were recorded in the vehicle system in the learning process. The obtained navigation path consists of several navigation nodes labeled in a sequential order. The vehicle is guided to visit each node sequentially in the navigation process. Four principles are proposed in this study to guide the vehicle to navigate to a desired destination. They are described as follows.

(1) The vehicle always keeps its navigation safe by avoiding collusions along the navigation path. By the use of the proposed obstacle detecting method, the vehicles always check if there is any dynamic obstacle in front and dodge it if necessary. In addition, by localizing a nearby light pole and the learned position of fixed obstacles, the vehicle conducts a specific procedure to dodge these static obstacles.

(2) The vehicle always adjusts guidance parameters based on the learned rules when detecting a landmark using techniques such as dynamic thresholding and

(3) The vehicle always follows the sidewalk curb if possible. After detecting and localizing the curb line, the vehicle modifies its direction to maintain a safe distance and orientation with respect to the curb on the sidewalk.

(4) The vehicle localizes its position and corrects the odometer readings at a constant time interval along the navigation path. According to the learned landmark information, the vehicle detects and then localizes an appointed landmark by the use of the proposed techniques. Then, calibration of the vehicle pose is conducted.

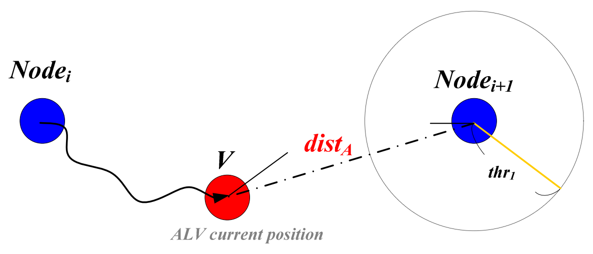
As a rule, the vehicle always localizes itself by the odometer readings to conduct *node-base navigation*. With the learned path information, we establish two principles to judge whether the vehicle has arrived at the next node in node-based navigation. The principles are described in the following.

(1) As shown in Figure 4.1(a), the distance $dist_A$ between the current vehicle position V and the position of the next node $Node_{i+1}$ is smaller than a threshold thr_1.

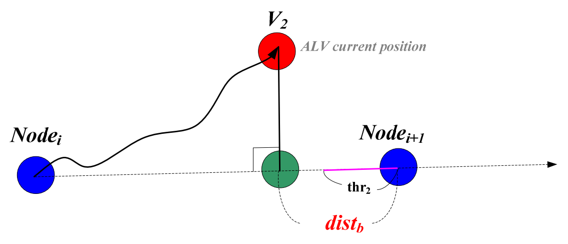
(2) As shown in Figure 4.1(b), if the distance $dist_B$ between the next node $Node_{i+1}$ and the position of the projection of the vehicle on the vector formed by $Node_i$ and $Node_{i+1}$, is smaller than a threshold thr_2.

By the mentioned navigation principles, the vehicle can be expected to navigate to the goal in the end. A flowchart illustrating the proposed node-based navigation is shown in Figure 4.2.

4.2.2 Calibration of vehicle odometer readings by sidewalk curb and particular landmarks

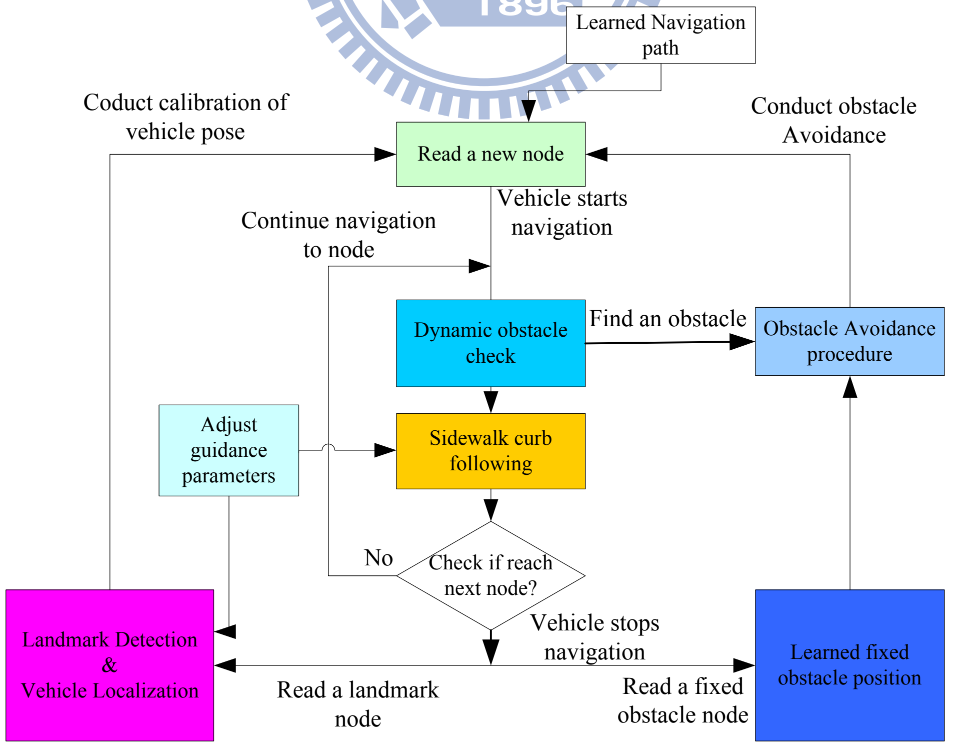


(a)



(b)

Figure. 4.1 Two proposed principles to judge 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.



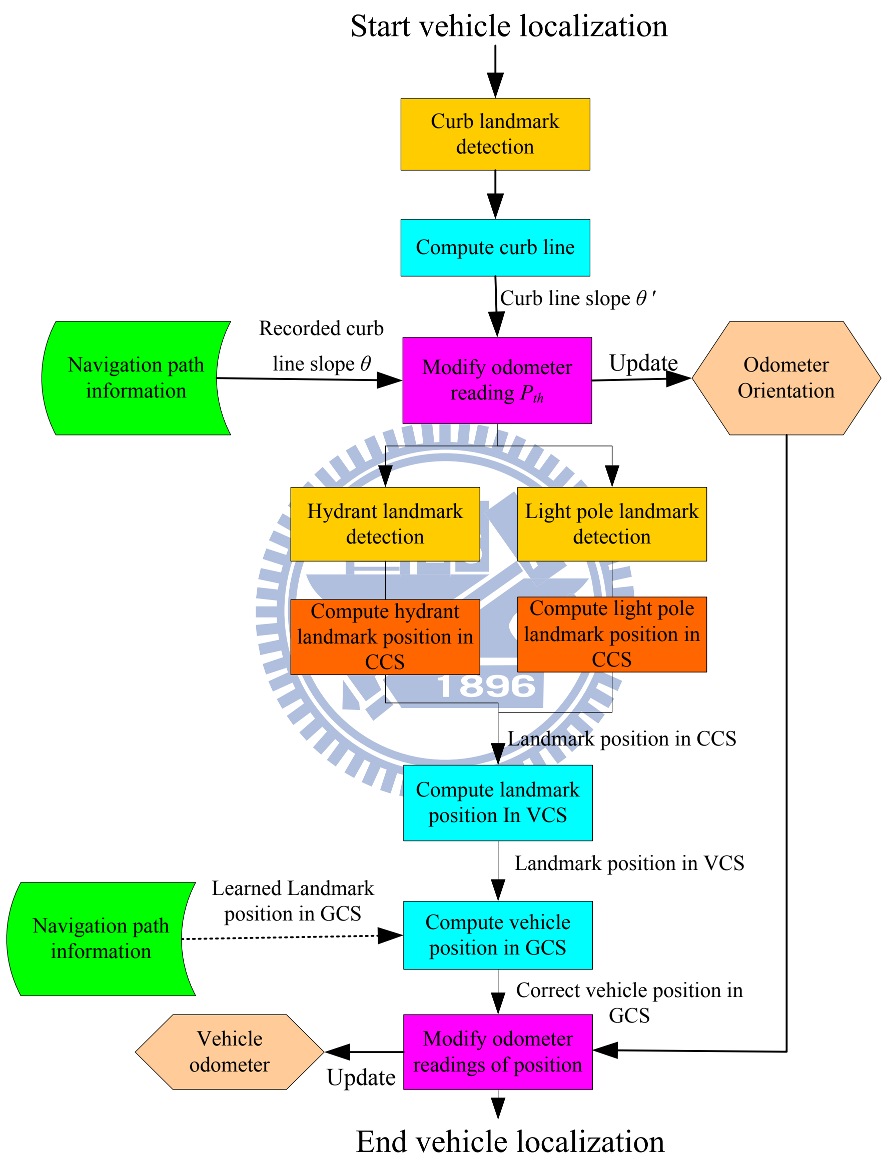
Learned Navigation path
Coduct calibration of vehicle pose
Conduct obstacle Avoidance
Read a new node
Vehicle starts navigation
Continue navigation to node
Dynamic obstacle check
Find an obstacle
Obstacle Avoidance procedure
Adjust guidance parameters
Sidewalk curb following
No
Check if reach next node?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

Figure 4.2 Proposed node-based navigation process

As mentioned in Chapter 3, the odometer readings provide three values P_x, P_y, and P_{th} for the vehicle to know its position (P_x, P_y) and moving direction P_{th}. Unfortunately, all of them become imprecise owing to incremental mechanic errors after the vehicle navigates for a period of time. In this section, we describe the proposed schemes to calibrate the odometer readings. The process of odometer reading calibration is illustrated in Figure 4.3. At first, we use recorded curb line segment information to calibrate the orientation reading P_{th} of the odometer. Second, by the recoded hydrant and light pole positions, we use the proposed hydrant and light pole detection method to obtain its position and then calibrate the position readings (P_x, P_y) of the vehicle. The reason why we have to combine a hydrant or light pole position with the curb information is that in the odometer reading calibration method we propose in this study, we have to calibrate the orientation odometer reading in advance using the detected curb line before the computed position of the hydrant and light pole can be used to localize the vehicle position.

(A) Odometer calibration by the hydrant and the sidewalk curb line

Two different positions of the vehicle at two nodes in the navigation path and the relation between the vehicle, the curb, and the hydrant are illustrated in Figure 4.4. The calibration process consists of two steps. Firstly, after adjusting the vehicle to the direction specified by the current odometer readings, we detect the nearby straight curb line segment seen in the omni-image, and obtain the slope angle with respect to the vehicle. From the learned navigation path, we can obtain the recorded slope angle of the curb line, and then analyze the two different slope angles to estimate the *correct* direction of the vehicle. Second, we conduct the vehicle to detect the hydrant and obtain its location. According to the recorded hydrant position from the learned navigation path, we use the correct vehicle orientation to compute the correct vehicle position by the relation between the hydrant position and the vehicle position in the GCS as shown in Figure 4.5. We describe the proposed method to calibrate the odometer readings in detail in the following algorithm.



Start vehicle localization
Curb landmark detection
Compute curb line
Curb line slope θ'
Navigation path information
Recorded curb line slope θ
Modify odometer reading P_{th}
Update
Odometer Orientation
Hydrant landmark detection
Light pole landmark detection
Compute hydrant landmark position in CCS
Compute light pole landmark position in CCS
Landmark position in CCS
Compute landmark position In VCS
Landmark position in VCS
Navigation path information
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 calibration process.

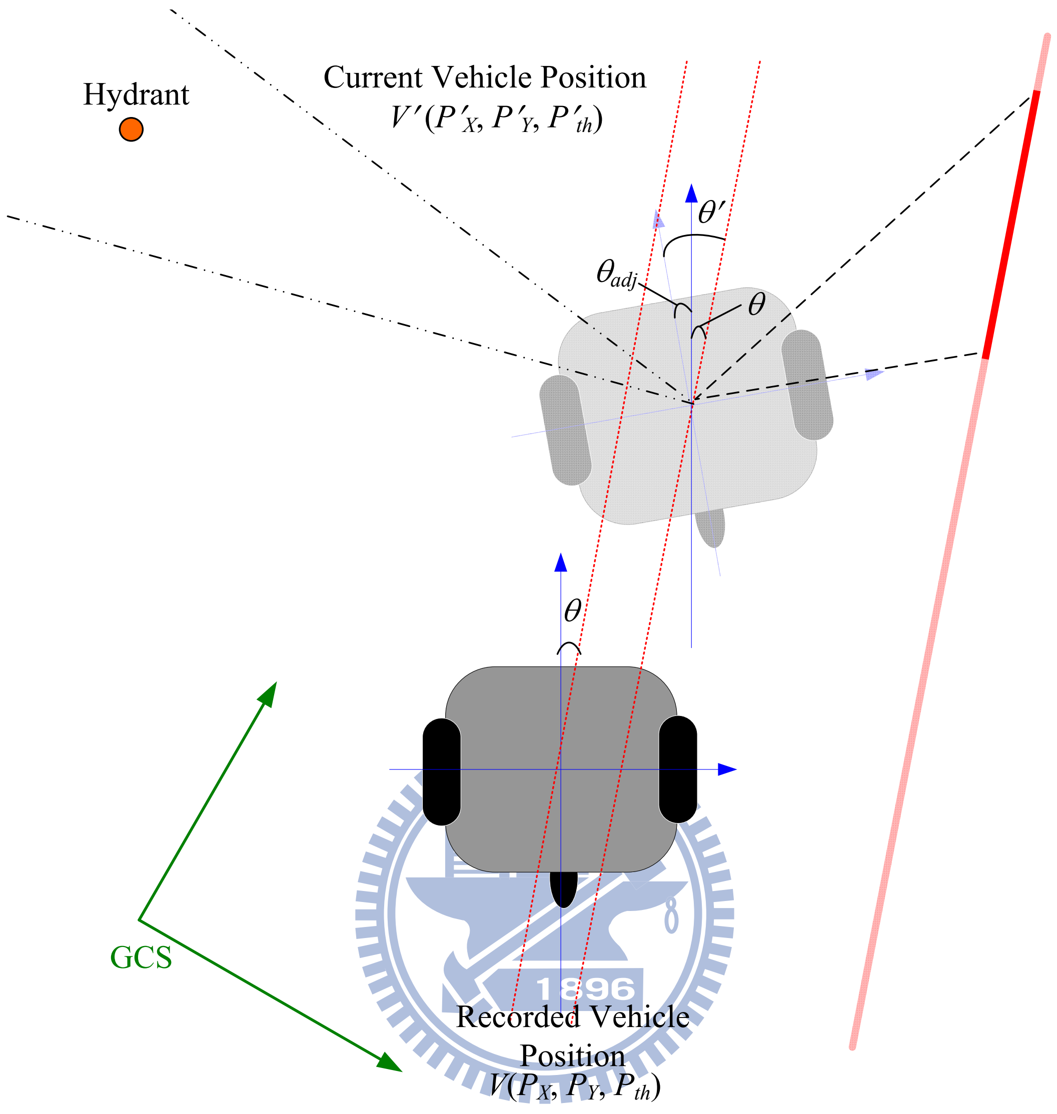
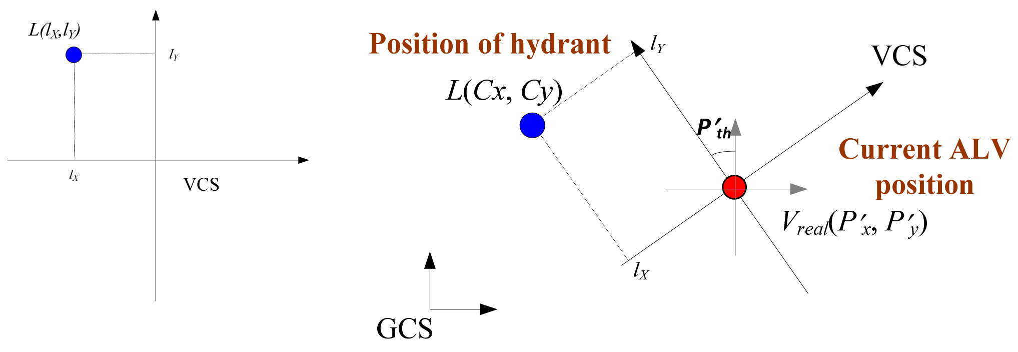


Figure 4.4 A recoded vehicle position *V* and the current vehicle position *V'* in the GCS.



(a) (b)

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

Algorithm 4.1 ***Odometer readings calibration by a hydrant and a curb line segment.***

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

Output: None.

Step.

Step 1. Turn the vehicle to the recorded direction P_{th}, conduct the curb line detection process 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 hydrant 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 as 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 imprecise position readings of the odometer, (P_X', P_Y'), by the computed vehicle position (X_{cali}, Y_{cali}).

(B) Odometer calibration by the light pole and the sidewalk curb line

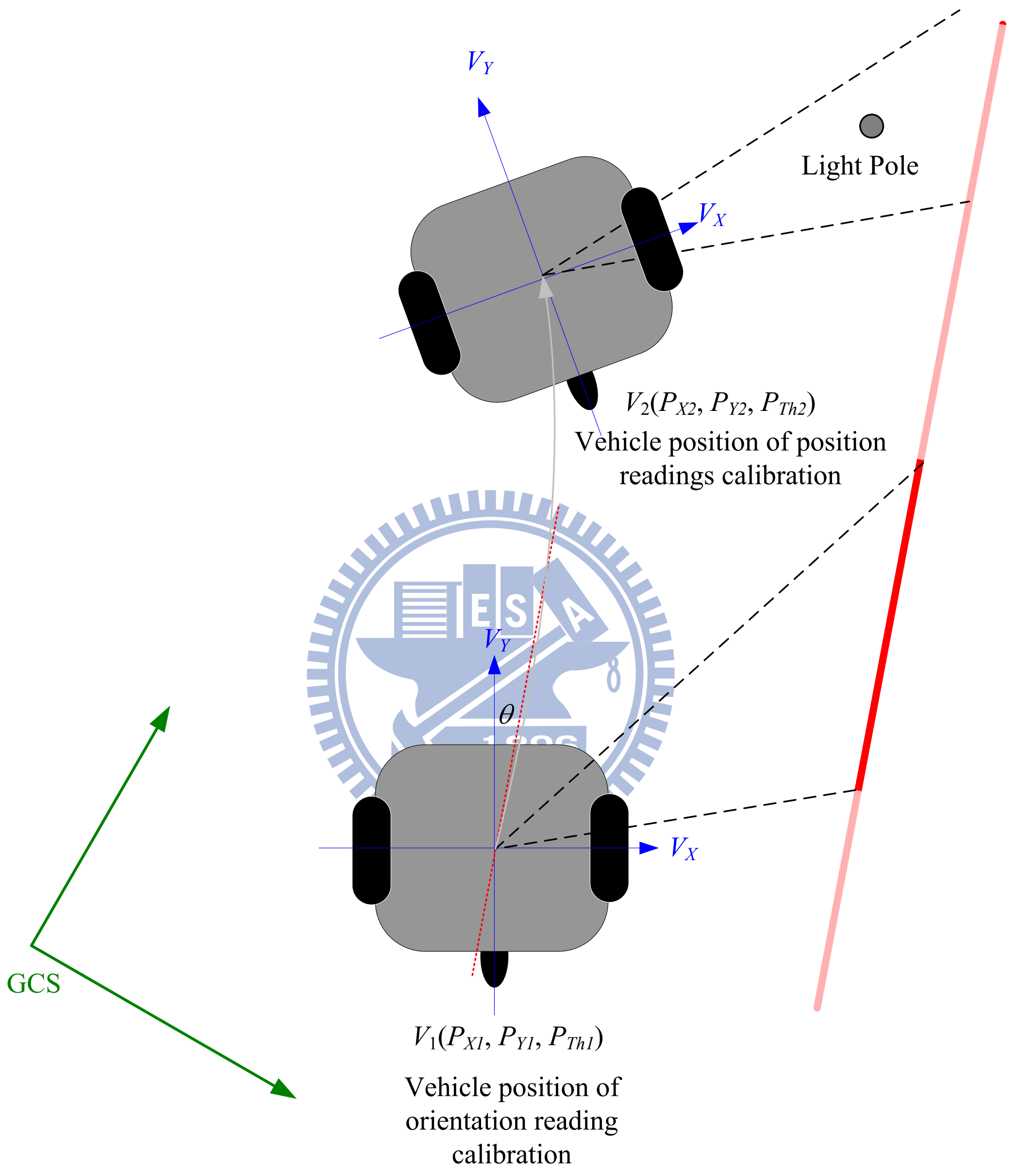
The process for calibration by the light pole and the sidewalk curb is similar to the above-mentioned method for odometer calibration by a hydrant and a sidewalk curb line segment. First, we detect and localize a nearby curb line segment for the purpose to calibrate the orientation reading in a similar way as described previously at a node V_1 in the learned path. Next, we conduct a slight difference task, i.e., we navigate the vehicle a step further to another node V_2, which is a location recoded in the navigation path with a light pole nearby, in order to detect the light pole at a closer location. The process is shown in Figure 4.4. It is noted that here the mechanical error of the orientation reading is assumed slight after the movement of the vehicle from node V_1 to node V_2. Then, after detecting and localizing the light pole position, we use the same method to compute the current vehicle position and modify the position odometer as that used for the calibration work using the hydrant described previously.

4.2.3 Dynamic exposure adjustment for different tasks

In the navigation process, by the recorded relevant environment intensity information in the learned navigation path, we can adjust the luminance into an appropriate value for different works. According to the experimental result as shown in Figure 4.7, we find that there exits a specific range of exposure values in which the exposure value has an approximate linear relation with the image intensity in a specific area in the image. Thus, we can estimate an appropriate exposure value *Exp* using the following polynomial function f_{exp}:

$$Exp = f_{exp}(Y) = m \times Y + b, \tag{4.3}$$

where Y is the average intensity in a specific region in the image, and a and b are two parameters.



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

Figure. 4.6 Process of odometer calibration by the light pole and 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 reading by a detecting light pole.

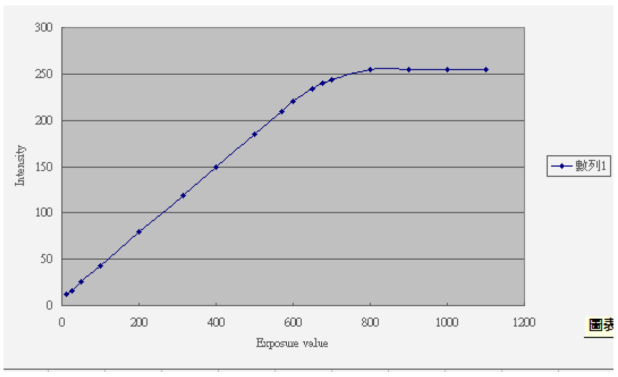


Figure. 4.7 A relationship between the exposure value and intensity in an experimental result.

However, under different light sources in outdoor environments, the specific range will be different, so is the linear function, f_{exp}. Thus, we propose an efficient method consisting of two stages to automatically obtain an appropriate exposure value which can be utilized to obtain an appointed illumination in an appointed region in an omni-image. First, we use a bisection scheme to adjust the exposure to find the specific range. It is desired to obtain two approximate bounds of the exposure value between which we can get proper intensities. Next, by the two bounds, we utilize linear interpolation to adjust the exposure value and then obtain the desired illumination. An algorithm to describe the proposed method is as follows.

Algorithm 4.2 ***Dynamic exposure adjustment.***

Input: an input image I_{input}; desired environment intensity Y_{base} and relevant environment window Win_{en}; and the minimum lower bound Exp_1 and the maximum upper bound Exp_2 of the camera exposure value.

Output: None.

Step.

Step 1. Initialize two parameters $Y_1 = -1$ and $Y_2 = -1$.

Step 2. Compute an exposure value Exp_{bi} by the following equation:

$$Exp_{bi} = \frac{(Exp_1 + Exp_2)}{2}. \tag{4.4}$$

Step 3. Use Exp_{bi} to acquire an image I_{input} with the system camera, and compute the average intensity Y_{cur} in Win_{en} in I_{input}.

Step 4. Compare Y_{cur} with Y_{base}:

(1). if $Y_{base} < Y_{cur}$, set $Exp_2 = Exp_{bi}$ and $Y_2 = Y_{cur}$;

(2). if $Y_{base} \geqq Y_{cur}$, set $Exp_1 = Exp_{bi}$ and $Y_1 = Y_{cur}$.

If Y_1 and Y_2 are between 10 and 245, go to Step 5; else, go to Step 2.

Step 5. Compute the exposure value Exp_{linear} by the following equation:

$$Exp_{linear} = \frac{Y - Y_1}{Y_2 - Y_1} \times (Exp_2 - Exp_1) + Exp_1 . \tag{4.5}$$

Step 6. Use Exp_{linear} to acquire an image I_{input} and compute the average intensity Y_{cur} of I_{input} in Win_{en}. If $|Y_{cur} - Y_{base}|$ is smaller than a threshold Thr_Y, then exit.

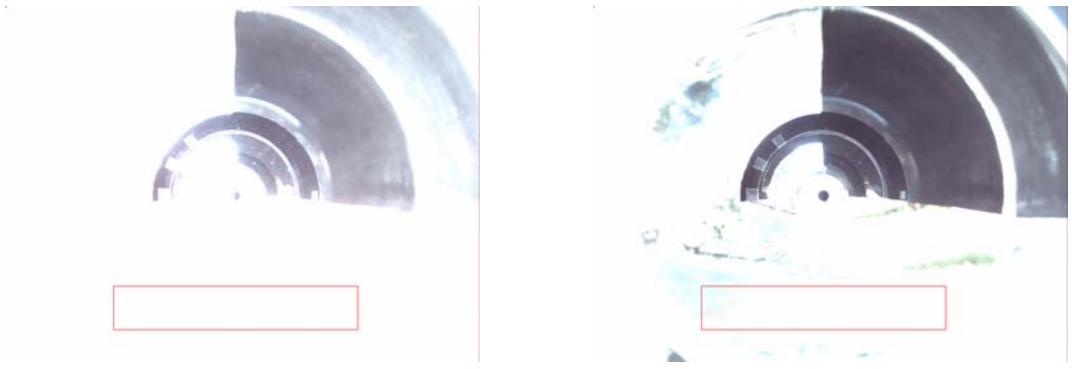
Step 7. Compare Y_{cur} with Y_{base}:

(1). if $Y_{base} < Y_{cur}$, set $Exp_2 = Exp_{linear}$ and $Y_2 = Y_{cur}$;

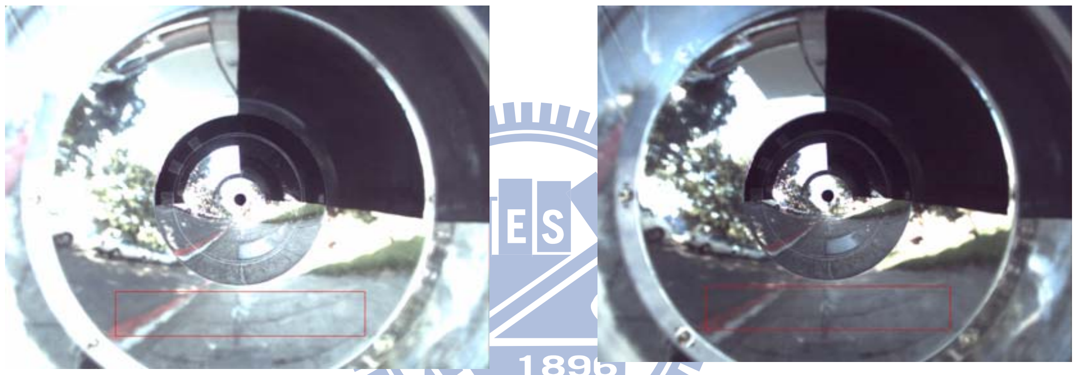
(2). if $Y_{base} \geqq Y_{cur}$, set $Exp_1 = Exp_{linear}$ and $Y_1 = Y_{cur}$,

and then go to Step 6.

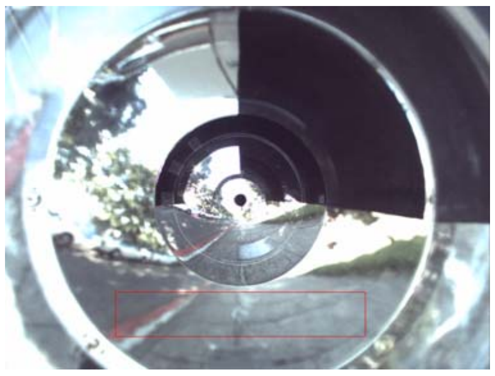
An experimental result for dynamically adjusting the exposure in the sidewalk curb detection task in the outdoor environment is illustrated in Figure 4.8. By the use of the leaned environment window for curb detection as illustrated by a red rectangular shape on the image in each figure, we compute the average intensity in this region. In the first stage, for the purpose to finding the exposure bounds, we conduct the bi-section scheme to adjust the exposure value as shown in Figures 4.8(a) through 4.8(d). After that, using the obtained exposure lower bound 50 and upper bound 100, we can use a linear interpolation scheme to obtain a suitable intensity on the image as illustrated in Figures 4.8(e).



(a) (b)



(c) (d)



(e)

Figure 4.8 Process of the proposed method to dynamically adjust the exposure for the sidewalk detection task. (a) With exposure value 400. (b) With exposure value 200. (c) With exposure value 100. (d) With exposure value 50. (e) A suitable illumination for sidewalk detection with exposure value 79.

4.3 Detail Algorithm of Navigation Process

In this section, we describe the detail process for vehicle navigation in the navigation process. The flowchart of the entire navigation process is shown in Figure 4.8. With the learned information, the vehicle navigates along the learned path by the way of visiting each recorded node consecutively and conducts appointed works at specific positions until reaching the destination of the learned path. The entire navigation process is described in the following algorithm.

Algorithm 4.3 ***Navigation Process***.

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

Output: Navigation process.

Step.

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

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

Step 3. Check the illumination by the recoded environment intensity and conduct the dynamic exposure adjustment procedure if necessary, and then conduct the vehicle to navigate forward.

Step 4. Try to find obstacles; and if an obstacle is founded and located at a position which is too close to the vehicle, stop the vehicle and insert avoidance nodes (see Section 6.2 for the detail) into the navigation path for the purpose of obstacle avoidance and go to Step 1.

Step 5. If a sidewalk following mode is adopted, modify the vehicle direction after localizing the curb landmark by the curb detection method using the

Step 6. Check whether the next node N_{next} is reached by the mentioned two principles in Section 4.2.1; and if not, go to Step 4.

Step 7. If a fixed obstacle is read from N_{path}, insert dodging nodes into the navigation path and go to Step 10.

Step 8. If a hydrant or light pole landmark is read from N_{path}, take the following steps and then go to Step 10.

8.1 Check the illumination in the relevant environment windows in the image for the appointed landmark by the recoded environment intensity, and then dynamically adjust the exposure if necessary.

8.2 Detect the appointed landmark, a light pole or a hydrant, and obtain the landmark position as illustration in Sections 5.3 and 5.4, respectively.

8.3 Use the landmark position to localize the vehicle position and modify the odometer position as described in Section 4.2.

Step 9. If a curb line calibration node is read from N_{path}, modify the orientation reading of the odometer by detecting and localizing a curb line segment, as illustrate as described in Section 4.2.

Step 10. Repeat Steps 1 through 9 until there exists no remaining nodes in N_{path}.

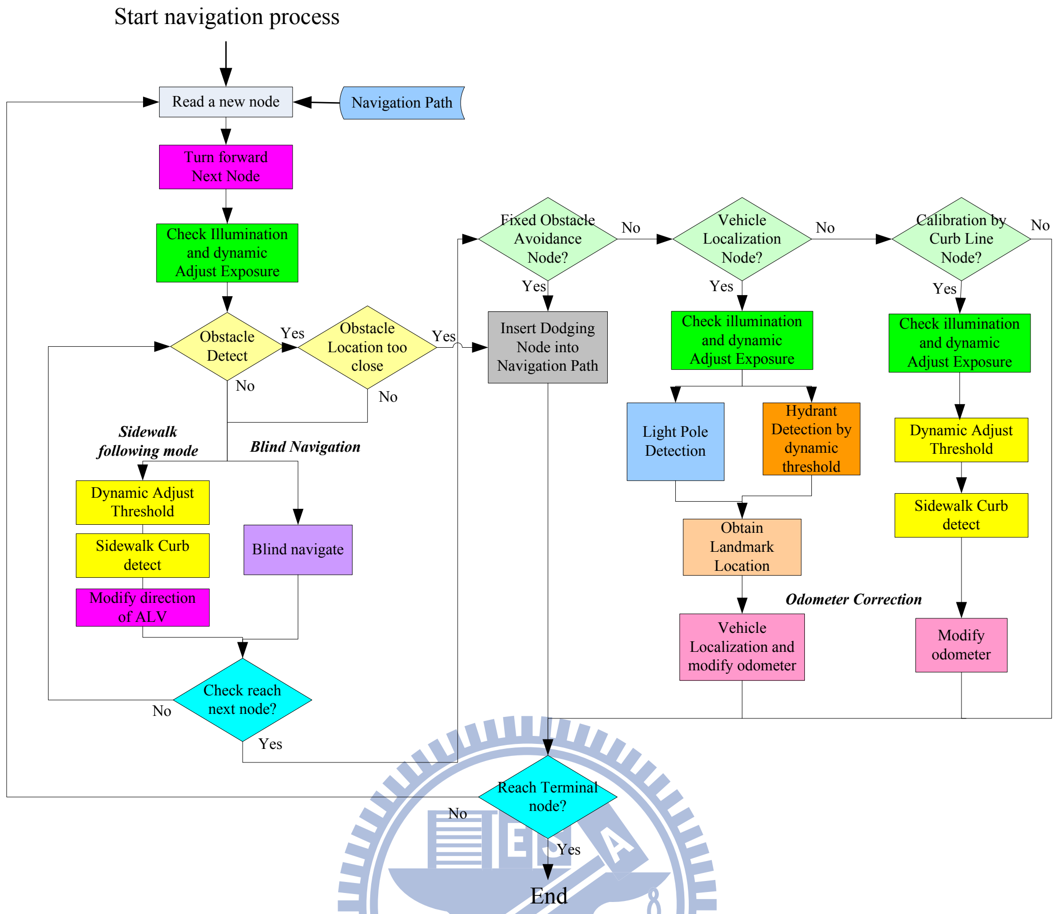


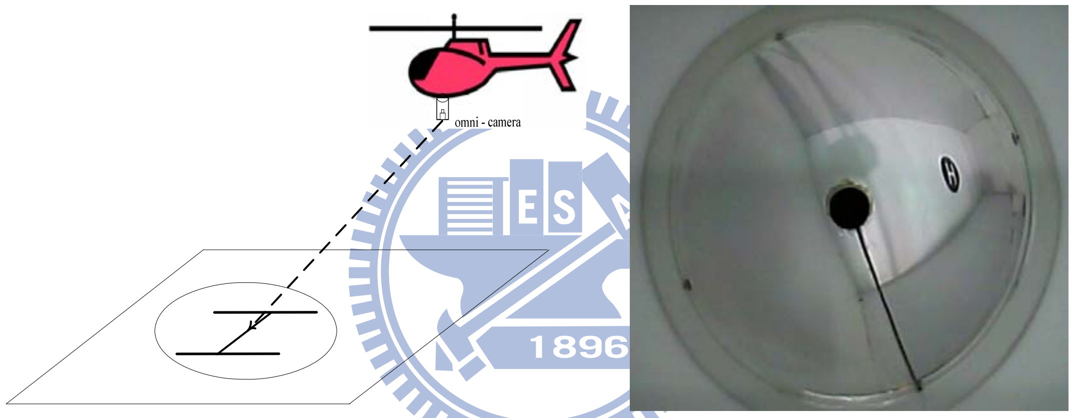
Figure 4.9 Flowchart of detailed proposed navigation process.

Chapter 5
Light Pole and Hydrant Detection in Images Using a 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 technique to localize each light pole or hydrant landmark on the navigation path for vehicle navigation. However, in contrast to the function of a traditional projective camera, the projection of a space line on an omni-image using an omni-camera is not a line shape any more but a *conic-section curve* [26]. Some techniques have been proposed for line detection in an omni-image, among which is Wu and Tsai's method [26] which detects lines in an H-shaped landmark for use in automatic helicopter landing, as illustrated in Figure 5.1. By the use of the parameters of a hyperboloidal mirror and some geometric relationship, they proved that the projection of a space line onto an omni-image is a conic section curve. Then, by the use of a simple technique using the 2D Hough transform, they extracted the conic section curve in the omni-image and localized the boundary lines of the H shape for conducting helicopter localization.

However, the above-mentioned method is based on the condition that the parameters of the hyperboloidal mirror are known, but in fact retrieving the parameters of a hyperboloidal mirror is not an easy work. Hence, by the use of the *pano-mapping* method which is a more convenient omni-camera calibration method, we propose a new space line detection technique in this study. Instead of directly obtaining the projected conic section cure of a space line in the omni-image, we obtain the *space plane* which goes through the desired space line and the mirror center. The detail of the proposed line detection method by the use of the two-mirror omni-camera is introduced in Section 5.2.1. Furthermore, for the specific space line which is *perpendicular to the ground*, we derive in this study a method to obtain its 3D information directly based on the results of the proposed line detection method.



(a) (b)

Figure 5.1 Wu and Tsai [26] 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, the light pole and hydrant localization works can be completed for vehicle navigation in both the learning and the navigation processes. We introduce the proposed hydrant and light pole detection and localization techniques in Sections 5.3 and 5.4, respectively.

5.2 Proposed Technique for Space Line Detection

5.2.1 Line detection using pano-mapping table

In this section, we introduce the proposed space line detection technique for use on omni-images taken by the two-mirror omni-camera. As mentioned previously, it is desired to detect the space plane, which goes through a specified space line and the mirror center, instead of detecting a space line projected on an omni-image in other methods. The process is described in the following. It is emphasized that the pano-mapping table has be established in advance for the use in this process.

Suppose that the space line L to be detected is projected by *Mirror A* onto the omni-image, and that P is an arbitrary space point on L. Firstly, we consider a way to represent a vector which goes through P and the mirror center in the camera system used in this study. As shown in Figure 5.2, a light ray going through the space point P is projected by *Mirror A* onto an image point I. The mirror center O_A and P together form a vector V_p', denoted as (P_x', P_y', P_z') in the CCS CCS_{local}. This vector V_p' can be described using the elevation and azimuth angles α and θ by the following equations:

$$P_x' = \cos\alpha \times \cos\theta\,;\quad P_y' = \cos\alpha \times \sin\theta\,;\quad P_z' = \sin\alpha\,. \tag{5.1}$$

Next, owing to the slant-up placement of *Mirror A* discussed previously in Chapter 2, we rotate the camera coordinate system CCS_{local} by a specific slant angle, denoted as γ. By the use of the rotation matrix described in Equation (2.5), the transformation between the coordinates (X', Y', Z') of the original CCS CCS_{local} and the coordinates (X, Y, Z) of the rotated CCS can be described as follows:

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

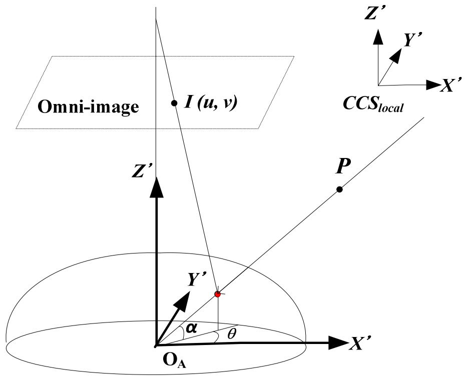


Figure 5.2 A space point with a elevation angle α and an azimuth θ.

By the above coordinate transformation described by Equation (5.2), we can convert vector V_p' into a new one V_p, 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:

$$V_p = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} \cos\alpha \times \cos\theta \\ \cos\alpha \times \sin\theta \times \cos\gamma + \sin\alpha \times \sin\gamma \\ -\cos\alpha \times \sin\theta + \sin\alpha \times \cos\gamma \end{bmatrix}. \tag{5.3}$$

Next, considering the space line L projected onto the omni-image the one I_L as shown in Figure 5.3, we can find a space plane Q which goes through L and the mirror center O_A. For this, suppose that the normal vector of Q is denoted as $N_Q = (l, m, n)$. Then, we can derive the following equation to describe the coordinates (X, Y, Z) of a pixel on the space plane Q:

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

On the other hand, it is noted that vector V_P is perpendicular to N_Q, so that the inner product between V_P and N_Q becomes zero, leading to the following equation:

$$N_Q \cdot V_P = (l, m, n) \cdot (P_x, P_y, P_z) = lP_x + mP_y + nP_z = 0. \tag{5.5}$$

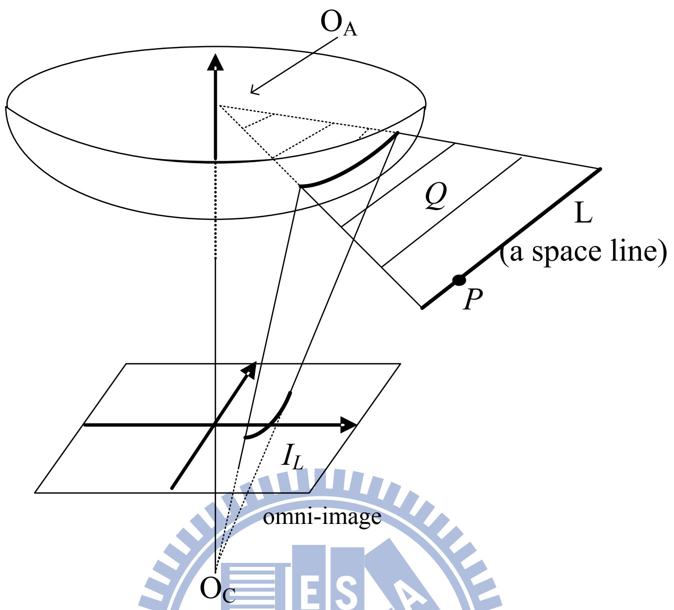


Figure 5.3 A space line L projected on I_L in an omni-image.

By Equation (5.3), we can transform Equation (5.5) into an alternative form as follows:

$$l + m \times \frac{\cos\alpha \times \sin\theta \times \cos\gamma + \sin\alpha \times \sin\gamma}{\cos\alpha \times \cos\theta} + n \times \frac{-\cos\alpha \times \sin\theta + \sin\alpha \times \cos\gamma}{\cos\alpha \times \cos\theta} = 0. \tag{5.6}$$

From Equation (5.6), it is desired to obtain the three unknown parameters l, m, and n which represent the normal of the space plane Q. For this purpose, we divide Equation (5.6) by n to get the following form:

$$B + A \times \frac{\cos\alpha \times \sin\theta \times \cos\gamma + \sin\alpha \times \sin\gamma}{\cos\alpha \times \cos\theta} + \frac{-\cos\alpha \times \sin\theta + \sin\alpha \times \cos\gamma}{\cos\alpha \times \cos\theta} = 0 \tag{5.7}$$

where $A = m/n$, $B = l/n$. We may rewrite the above equation further to obtain

$$B + A\, a_0 + a_1 = 0 \tag{5.8}$$

where $A = m/n$, $B = l/n$, and

$$a_0 = \frac{\cos\alpha \times \sin\theta \times \cos\gamma + \sin\alpha \times \sin\gamma}{\cos\alpha \times \cos\theta}, \quad a_1 = \frac{-\cos\alpha \times \sin\theta + \sin\alpha \times \cos\gamma}{\cos\alpha \times \cos\theta}.$$

In the above equation, we use two parameters A and B to represent the original three ones l, m, n. By this form, we can use a simple 2D Hough transform technique to obtain the parameters A and B, as described in detail in the following algorithm.

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

Input: an input edge-point image I_{edge} which includes the points of the projection I_L of a space line L, and the pano-mapping table for *Mirror A*.

Output: two parameters, A_{max} and B_{max}, representing a normal vector of the space plane described by Equation (5.8).

Steps.

Step 1. Set a 2D Hough space S with the parameters A and B, and initialize all cell counts 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 values A and B by Equation (5.7) using θ and α, and 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.

After the algorithm is conducted, we can obtain the normal vector (l, m, n) of the desired space plane Q in another form represented by the two parameters $A = m/n$ and

$B = l/n$.

Furthermore, if L is a vertical space line which means that the normal vector of the space plane Q is parallel to the ground, then it is easy to figure out that m is equal to zero. Thus Equation (5.8) can be reduced to the following equation:

$$B = -a_1 \tag{5.9}$$

where $B = l/n$ and

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

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 a vertical space line and the mirror center.

5.2.2 3D data computation using a vertical space line

In this section, based on the proposed space line detection technique described above, we can derive the 3D data of a vertical space line (such as the boundary lines of a light pole or the vertical axis of a hydrant) from the omni-image, as described subsequently.

As shown in Figure 5.4, a vertical space line L is projected onto I_{L1} and I_{L2} on the regions of *Mirrors A* and *B*, respectively. The center O_A of *Mirror A* is located at coordinates (0, 0, 0) in the CCS as we previously assumed. Thus, with the slant angle denoted as γ and the length of the *baseline* denoted as b as shown in Figure 5.4, we can derive the position of the center O_B of *Mirror B* to be at coordinates $(0, b\sin\gamma, b\cos\gamma)$. Next, according to Equation (5.4), the equations of the two space planes Q_1 and Q_2 going through L and the mirror centers, O_A and O_B, respectively, can be described in the following:

$$l_1X + m_1Y + n_1Z = 0; \tag{5.10}$$

$$l_2X + m_2(Y - b\sin\gamma) + n_2(Z - b\cos\gamma) = 0 \tag{5.11}$$

where (l_1 , m_1 , n_1) represents the normal vector of Q_1 and (l_2 , m_2 , n_2) represents that of Q_2.

In addition, by the reason that the space line L is perpendicular to the ground, we know that m_1 and m_2 are both zero. Thus, the above two space plane equations can be reduced into the following forms:

$$l_1X + n_1Z = 0; \tag{5.12}$$

$$l_2X + n_2(Z - b\cos\gamma) = 0 \tag{5.13}$$

which are equivalent to

$$B_1X + Z = 0; \tag{5.14}$$

$$B_2X + (Z - b\cos\gamma) = 0 \tag{5.15}$$

where $B_1 = l_1/n_1$ and $B_1 = l_2/n_2$.

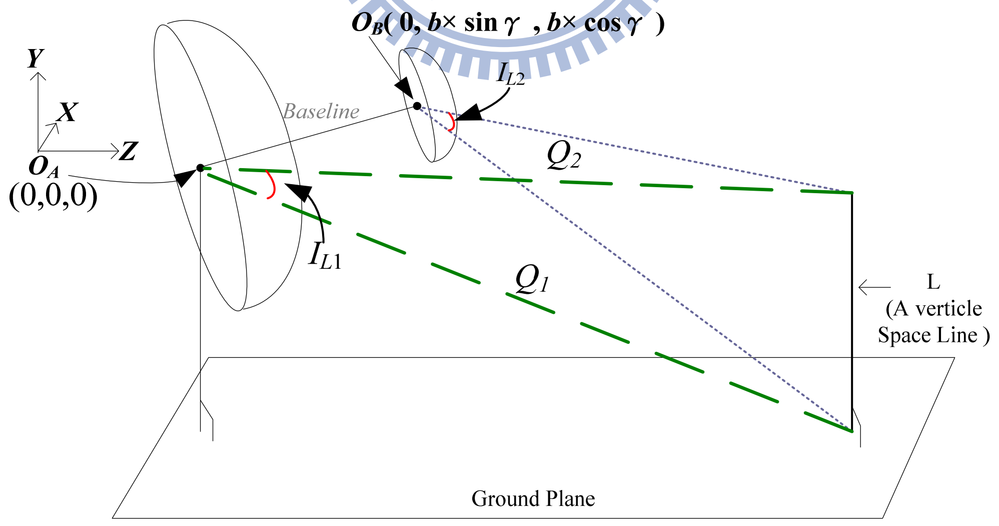


Figure 5.4 A space line projected onto I_{L1} and I_{L2} on two mirrors in the used two-mirror omni-camera.

By solving Equations (5.14) and (5.15), we can obtain the following equations to describe the position of the vertical space line L:

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

$$Z = -B_1 X = -B_2 X + (b + \cos\gamma) \tag{5.16}$$

where $B_1 = l_1/n_1$, $B_1 = l_2/n_2$. It is noted that Equation (5.15) cannot be solved when B_1 is equal to B_2, resulting in the parallelism between the two space planes Q_1 and Q_2.

In conclusion, for a vertical space line projected on both of the regions of *Mirrors A* and *B* in the omni-image, after conducting the proposed line detection on the regions of *Mirrors A* and *B* in the omni-image and finding a pair of the corresponding space planes using Algorithm 5.1, we can use Equation (5.16) to compute the location of the vertical space line directly.

5.3 Method of Light Pole Detection

The idea of the proposed method for light pole localization is to use *two vertical boundary lines* of the light pole to estimate the position of the light pole. The entire process for light pole position computation is shown in Figure 5.5. Firstly, the proposed technique to detect two boundary lines of the light pole is introduced in Section 5.3.1. Then, the computation of the light pole location is described in Section 5.3.2. Finally, some experimental results for light pole detection are given in Section 5.3.3.

5.3.1 Light pole boundary detection

In this section, we describe how to detect the two boundary lines of a light pole in an omni-image. The proposed method consists of two steps. Firstly, we conduct light pole segmentation by the Canny edge detection technique to obtain the boundary points of the light pole. Then, by the resulting edge-point image, we use the above-mentioned space line detection technique to find the two vertical boundary lines based on a 1D Hough transform technique. Finally, we can obtain two specific space planes which go through one of the two light pole boundary lines as well as two other space planes which go through the other of the two light pole boundary lines; and use these results to compute the light pole location, as described in the next section. The detailed algorithm for the just-mentioned idea of light pole detection is described as follows.

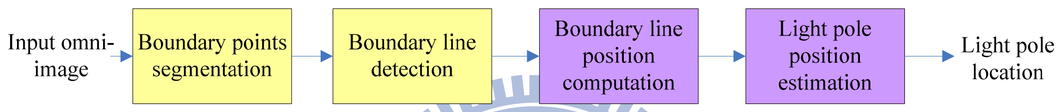

Input omni-image
Boundary points segmentation
Boundary line detection
Boundary line position computation
Light pole position estimation
Light pole location

Figure 5.5 Proposed method of light pole localization.

Algorithm 5.2 ***Light pole boundary line detection.***

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

Output: two parameters B_{A1} and B_{B1} representing the parameters of two space planes through one of the two boundary lines of the light pole and then through the *Mirror A* center and the *Mirror B* center, respectively; and two other parameters B_{A2} and B_{B2} representing the parameters of two space planes through the other one of the two boundary lines of the light pole and then through the *Mirror A* center and the *Mirror B* center, respectively.

Steps.

Step 1. For I_{input}, use the Canny edge detector to conduct edge detection to extract the feature points of the boundary lines of the light pole, and obtain an

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_{lp}, look up the pano-mapping table to obtain an azimuth θ and an elevation angle α.

Step 4. Compute B by Equation (5.9) using θ and α, and increment by 1 the value of the cell with parameter B in S.

Step 5. Repeat Steps 2 and 3 until all edge points in win_B of Win_{lp} are computed.

Step 6. Find two cells, denoted as B_1 and B_2, with the two maximum values in space S

Step 7. If $B_1 > B_2$, set $B_{A1} = B_1$ and $B_{A2} = B_2$; else, set $B_{A1} = B_2$ and $B_{A2} = B_1$.

Step 8. Take B_{A1} and B_{A2} as outputs.

Step 9. In the same way, repeat Steps 2 through 8 in win_S of Win_{lp} for *Mirror B* and take the obtained two corresponding parameters B_{B1} and B_{B2} as outputs.

5.3.2 Light pole position computation

After successfully detecting two boundary lines of a light pole, we can use them to compute the light pole location. The proposed technique for this is described in this section. At first, by the use of two known corresponding space planes obtained in the previous section, we compute the locations of the two light pole boundary lines, denoted as L_{in} and L_{out}, respectively, in the CCS as illustrated in Figure 5.6. Then, two corresponding points, G_{in} and G_{out}, on the ground can be obtained by the obtained equations of L_{in} and L_{out}. Next, we check whether the distance between G_{in} and G_{out} is close to the known diameter of the light pole. If not, we assume that the detected two vertical space lines are not the boundary lines of the light pole. Finally, we compute the center position between G_{in} and G_{out} for use as the light pole position G_{lp}. The detailed algorithm to estimate the light pole position is described in the following algorithm.

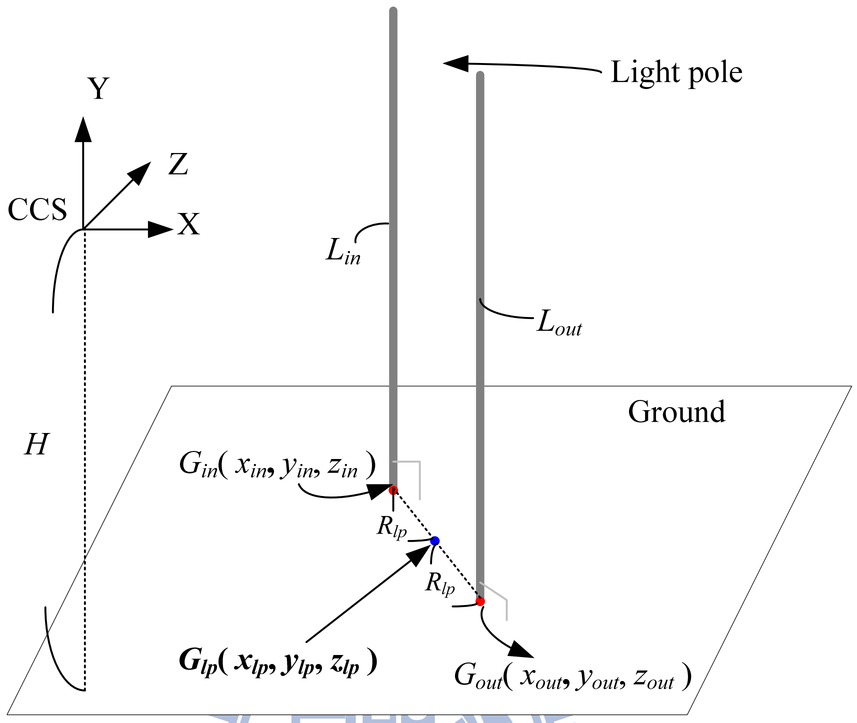

Light pole
Y
Z
CCS
X
L_{in}
L_{out}
Ground
H
$G_{in}(x_{in}, y_{in}, z_{in})$
R_{lp}
R_{lp}
$G_{lp}(x_{lp}, y_{lp}, z_{lp})$
$G_{out}(x_{out}, y_{out}, z_{out})$

Figure 5.6 Two obtained boundary lines L_{in} and L_{out} of the light pole in the CCS.

Algorithm 5.3 ***Light pole position computation.***

Input: two corresponding space plane parameters B_{A1} and B_{B1}, and two other corresponding parameters B_{A2} and B_{B2} obtained from Algorithm 5.2, of a light pole appearing in an omni-image.

Output: a light pole position G_{lp} in the CCS.

Steps.

Step 1. By B_{A1} and B_{B1}, compute one boundary space line L_1 of the light pole by Equation (5.16) and obtain its equation as follows:

$$X = X_1; \quad Z = Z_1. \tag{5.17}$$

Step 2. By B_{A2} and B_{B2}, compute another boundary space line L_2 of the light pole by Equation (5.16) and derive its equation as follows:

$$X = X_2; \quad Z = Z_2. \tag{5.18}$$

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

$$d = \sqrt{(X_1 - X_2)^2 + (Z_1 - Z_2)^2}\,. \tag{5.19}$$

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

Step 5. Compute the coordinates (x_{lp}, y_{lp}, z_{lp}) of the light pole position G_{lp} in the CCS as follows:

$$x_{lp} = (X_2 + X_1)/2; \quad y_{lp} = -H; \quad z_{lp} = (Z_1 + Z_2)/2 \tag{5.20}$$

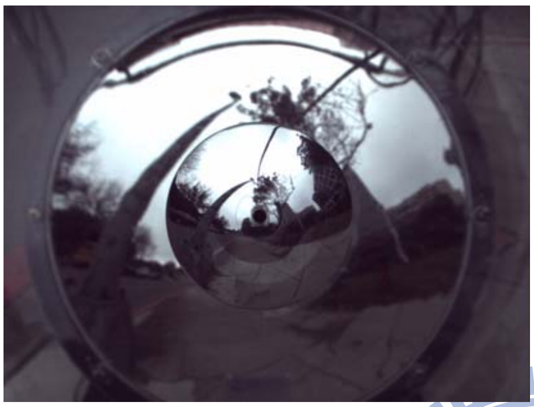
where H is the height of the camera center, and take G_{lp} as output.

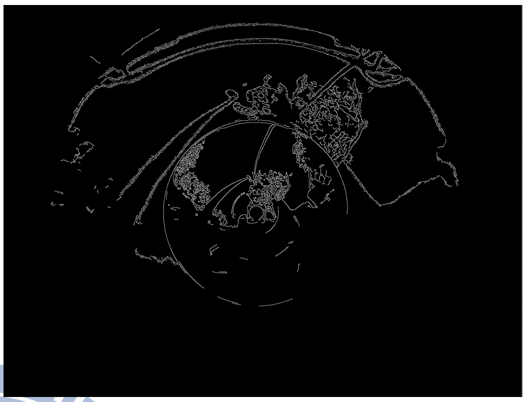
5.3.3 Experimental results for light pole detection

An input image with the projection of a light pole on the regions of *Mirrors A* and *B* is shown in Figure 5.7(a). After conducting Canny edge detection, we obtain an edge-point image as shown in Figure 5.7(b). By this edge image, we use the proposed line detection method to extract two light pole boundary lines, and the two 1D Hough spaces of the parameter B for *Mirrors A* and B are shown in Figures 5.8(a) and 5.8(b), respectively. The result of light pole detection is shown in Figure 5.9 and the relative light pole position with respect to the vehicle is shown in Figure 5.10.

5.4 Method of Hydrant Detection

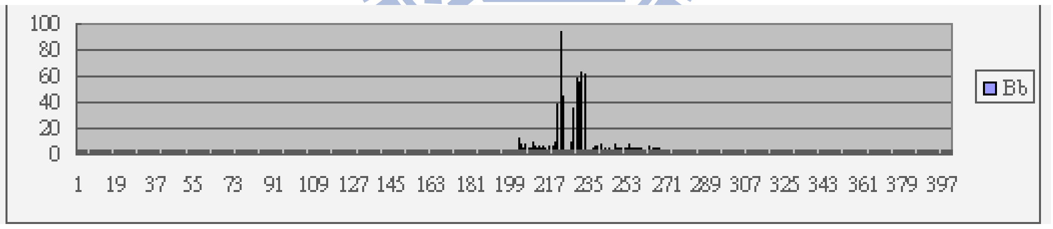
In this section, we introduce the proposed method to localize a hydrant. At first, we introduce the used method to describe a hydrant contour and the learning of the hydrant contour in Section 5.4.1. Next, by the use of dynamic threshold adjustment and vertical line localization techniques, we can extract a representative structural feature of the hydrant, namely, its axis, and then estimate the position of the axis, as described in Section 5.4.2. Also, some experimental results for hydrant detection by the proposed method are given in Section 5.4.3.



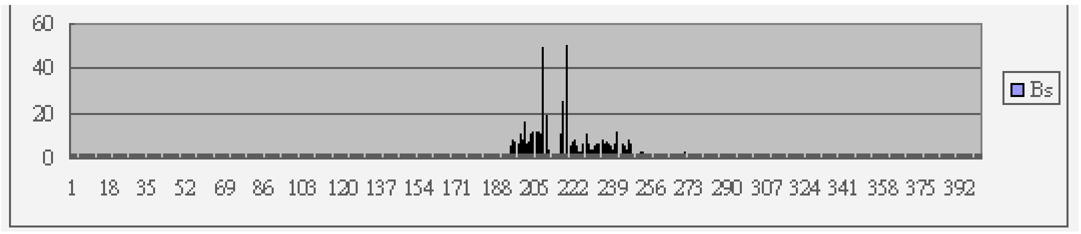


(a) (b)

Figure 5.7 Two omni-images with the projection of the light pole. (a) The input image. (b) The result edge image after doing Canny edge detection.



(a)



(b)

Figure 5.8 Two 1D accumulator spaces with parameters *B*. (a) For *Mirror A*. (b) For *Mirror B*.

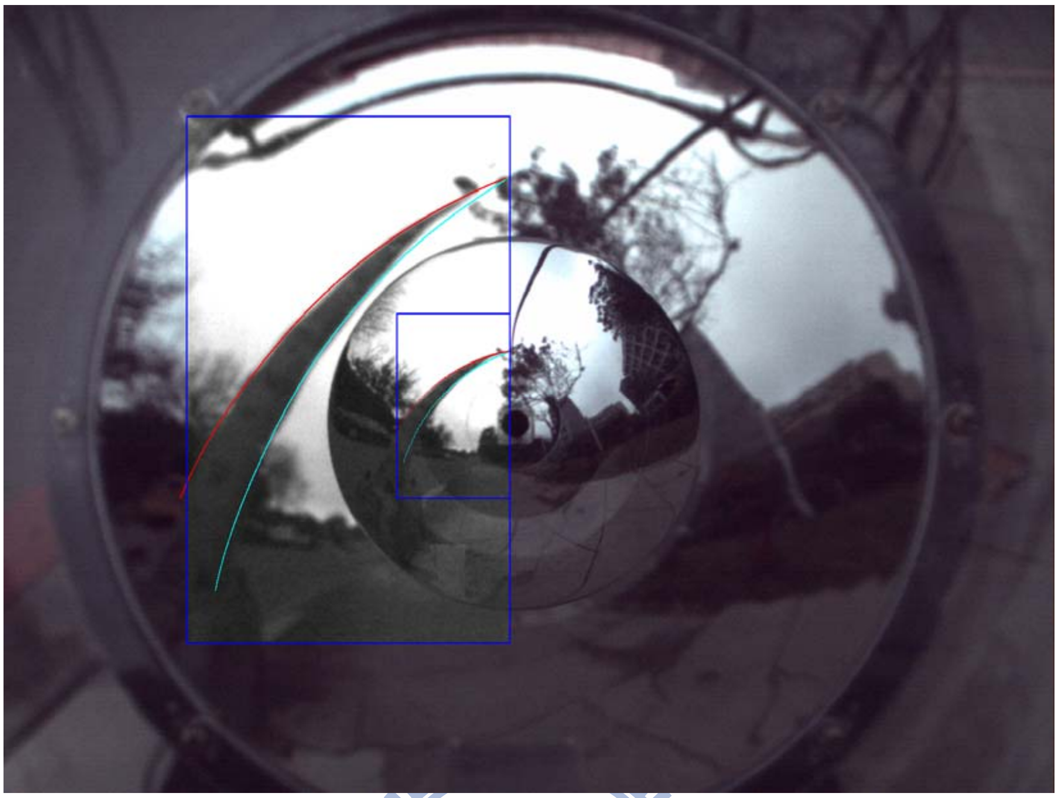


Figure 5.9 The result image of light pole detection. Two boundary lines are illustrated as the red and blue curves.

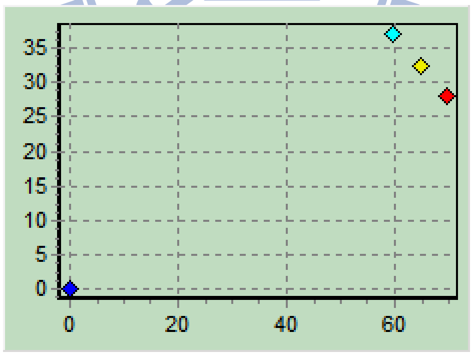


Figure 5.10 A computed light pole position, the yellow point, with respect to the vehicle position, the blue point, in the VCS. Two boundary lines are located at the blue and red positions.

5.4.1 Hydrant contour description

In hydrant detection, for the purpose to inspect the results of hydrant segmentation on the image, we use a simple description with two specific parameters obtained by *principal component analysis*. Specifically, after obtaining the hydrant segmentation results, we compute the covariance matrix C_x by the feature point positions in the image. After obtaining the two eigenvalues and the two corresponding eigenvectors of the matrix C_x, we compute 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 use ω and η to describe the hydrant contour as shown in Figure 5.11. The detail to obtain these two parameters is described in the following algorithm.

Algorithm 5.4 ***Hydrant contour parameter computation.***

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

Output: two hydrant contour parameters, a rotational angle ω, and a length ratio η.

Steps.

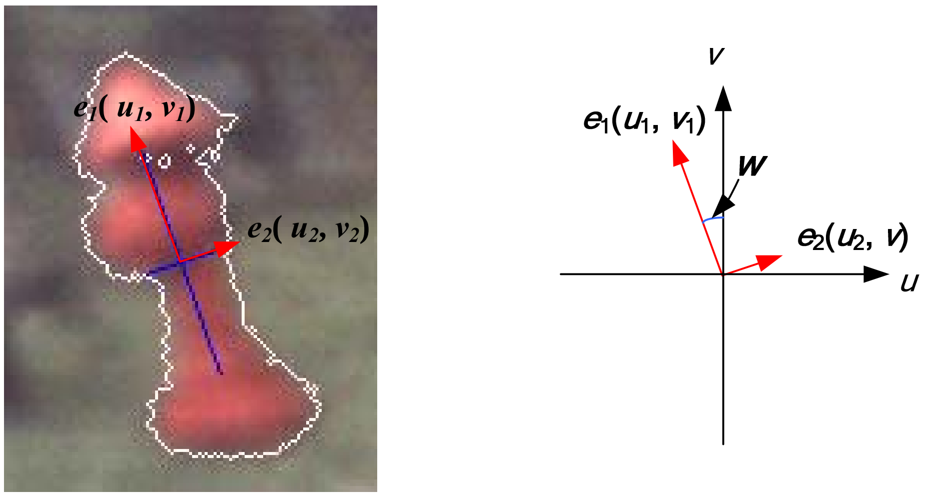
Step 1. Scan each feature point p with coordinates (u, v) in I_{input}, compute the center $m_x = (u_x, v_x)$ of all the feature points using their coordinates, and calculate the covariance matrix C_x of these feature points using their coordinates and m_x.

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 principle 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:

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

Step 4. Take ω and η as outputs.



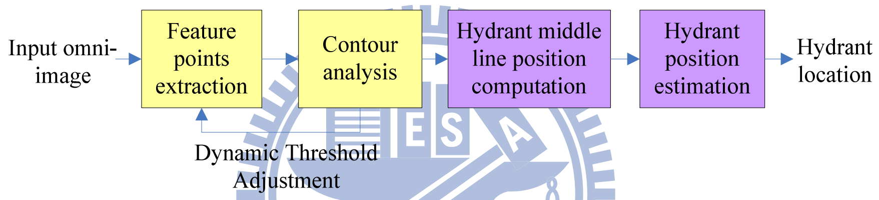
(a) (b)

Figure 5.11 Principal component analysis for the hydrant 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.

In addition, because different projections of the same hydrant on omni-images taken at different positions are usually similar, we can record as many different hydrant contours as possible in the learning process in order to "learn" the hydrant contour more precisely. More specifically, for the learning of a specific hydrant contour in the navigation path, we guide the vehicle to take a number of omni-images from different directions at different positions. For each obtained image, we compute two parameters ω_i and η_i by the above-described algorithm after extracting the hydrant feature points. Then, from all ω_i and η_i, we select a minimum angle $\omega_{\min}$ and a maximum angle $\omega_{\max}$ as well as a minimum ratio $\eta_{\min}$ and a maximum ratio $\eta_{\max}$ to compose the ranges of the hydrant contour parameters. Then, we record the four parameters $\omega_{\min}$, $\omega_{\max}$, $\eta_{\min}$, and $\eta_{\max}$ as the *hydrant contour thresholds*. After this learning process, if the computed rotational angle ω and the ratio η in hydrant detection are not in the learned ranges, we decide that the result of detection is not a hydrant.

5.4.2 Hydrant detection and localization

By the symmetric shape of a common hydrant, the idea of the proposed method for hydrant localization is to detect the vertical axis line of the hydrant using principal component analysis, and localize the hydrant by this line. The entire process to localize a hydrant in this way is shown in Figure 5.12, and two stages of works conducted in this process are described in the following.



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

Figure 5.12 Proposed method of light pole localization.

(A) Hydrant feature extraction by dynamic color thresholding

Due to the special color of the hydrant, we utilize the color information to extract the hydrant contour from an image. Specifically, by the use of the HSI color space, we use only the hue and saturation values to classify the hydrant feature in order to ignore the influence of the varying image intensity caused by the time-changing lighting condition in the outdoor environment. The conversion of color values from the RGB color space to the HSI color space is as follows:

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

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

$$I = \frac{1}{3}(R + G + B) \tag{5.22}$$

where

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

According to our experimental experience, we define two hue values, denoted as $H_{\min}$ and $H_{\max}$, as the hue threshold values of the upper and lower bounds for extracting the red feature of the hydrant. Similarly, we define two saturation values, denoted as $S_{\min}$ and $S_{\max}$, as the saturation threshold values of the upper and lower bounds for extracting the surface feature of the hydrant. These threshold values are used together to classify the hydrant feature points.

Furthermore, varying lighting conditions will influence the hue and saturation features. Based on the learned hydrant contour, we conduct dynamic color thresholding to adjust the recorded saturation threshold value of $S_{\min}$ 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. We describe the overall method to extract the feature points of the hydrant in detail in the following algorithm.

Algorithm 5.4 ***Hydrant detection by dynamic thresholding.***

Input: an input image I_{input} including a hydrant; the learned four hydrant contour parameters, ω_{min}, ω_{max}, η_{min}, and η_{max}; two hue threshold values H_{min} and H_{max}; two saturation thresholds S_{min} and S_{max}; and a set of environment windows Win_{hyd}.

Output: a bi-level image I_{bi} with feature points of the hydrant, and an adjusted saturation threshold $S_{\min}$.

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_{hyd}, compute its hue value h_{uv} and saturation value s_{uv} by Equation (5.22), and if h_{uv} is between H_{min} and H_{max} and s_{uv} is between S_{min} and S_{max}, 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 image 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 two contour parameters, the rotational angle ω and the length ratio η of M.

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

(B) Hydrant position computation by the vertical axis line of the hydrant

By using the results obtained by the hydrant contour extraction process described above, it is desired further to find the vertical axis line of the hydrant to localize the hydrant. We assume that the desired vertical axis line goes through both centers of the hydrant appearing in the regions of *Mirrors A* and *B* in the omni-image. After extracting the two center positions of the hydrant in regions of *Mirrors A* 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. Finally, we can obtain the hydrant position by the located axis line using the information of the two space planes. The detailed process is described as follows.

Algorithm 5.5 ***Hydrant location computation.***

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

Output: a hydrant position G_{hyd} in the CCS.

Steps.

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

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 (5.9), 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_A and B_B, compute the position coordinates X and Z of the axis line L of the hydrant by Equation (5.16).

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

$$x_{hyd} = X; \quad y_{hyd} = -H; \quad z_{hyd} = Z \tag{5.23}$$

where H is the height of the camera center.

Step 6. Take G_{hyd} as output.

5.4.3 Experimental results for hydrant detection

Some experimental results for hydrant detection are shown in this section. The input image with a hydrant on the regions of *Mirrors A* and *B*, respectively, is shown in Figure 5.13. The result of hydrant segmentation using the initial threshold values is shown in Figure 5.14(a). Next, the result of hydrant segmentation by dynamic thresholding is shown in Figure 5.14(b). We can see that the extracted contour in

Figure 5.14(b) is more similar to the real shape of the hydrant. Finally, the result of detecting the vertical axis line of the hydrant and the obtained hydrant position are shown in Figure 5.15.

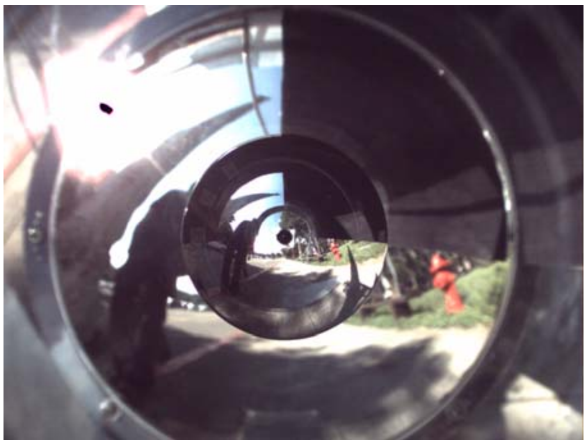
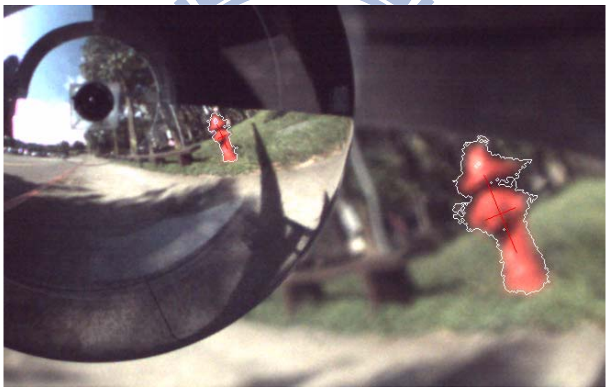
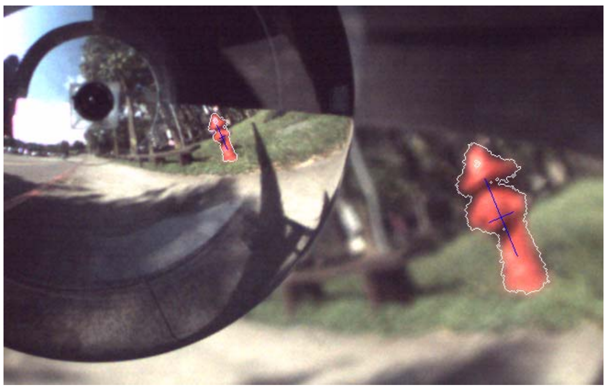


Figure 5.13 The input omni-image with a hydrant.

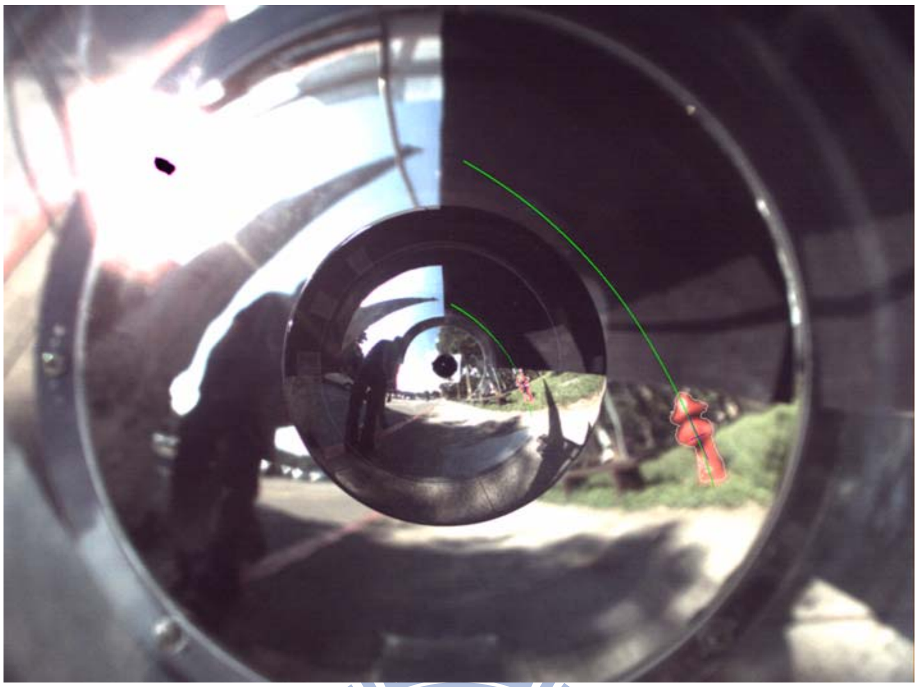


(a)

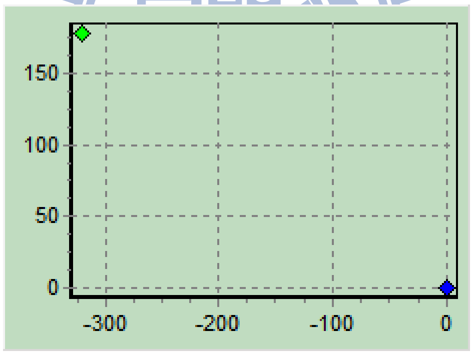


(b)

Figure 5.14 Two result images of hydrant segmentation with different threshold values (a) The result of hydrant segmentation with original threshold values. (b) The result image of hydrant segmentation by dynamic thresholding.



(a)



(b)

Figure 5.15 The result of hydrant detection and obtained hydrant position. (a) The result image of extracting the vertical axis line of the hydrant (b) The related hydrant position with respect to the vehicle position.

Chapter 6
Curb Line Following and Obstacle Avoidance in Navigation

6.1 Proposed Technique of Curb Line Following

To conduct vehicle navigation on sidewalks, we propose a technique to detect a curb line and compute its location with respect to the vehicle. Then, by a localized curb line, we can guide the vehicle to follow the curb and also calibrate the orientation odometer reading in the navigation process. In this system, we detect the curb line by the use of the projection of a curb line on the region of *Mirror A* in the omni-image. We know that the detected curb image points are on the floor in the real world, so the position of the curb line can be computed directly by the use of a single camera, i.e., the one with *Mirror A* in the proposed camera system.

In the remainder of this chapter, the proposed method to extract curb boundary points is introduced in Section 6.1.1. By the use of the method for curb boundary extraction, we conduct curb line localization by the proposed dynamic threshold adjustment technique described in Section 6.1.2. Finally, after deriving the location of the curb line, we propose a method for the vehicle to navigate by following the curb line in the navigation process as introduced in detail in Section 6.1.3. Some experimental results for curb detection are shown in Section 6.1.4.

6.1.1 Curb line boundary points extraction

For curb line detection on an omni-image, we define an environment window, denoted as Win_{curb}, to specify a specific region of *Mirror A* on the image. By an input omni-image obtained from the omni-camera, we perform the following four steps to compute the relative position of a detected curb boundary point with respect to the vehicle.

(1) Curb feature detection by the use of color information

Because of the special color of the curb (which is red in our experimental environment), we use the color information to extract the curb feature using the HSI color model. Like the method for hydrant feature extraction as discussed previously in Section 5.4.2, we classify the curb feature in the image by the use of two hue threshold values, denoted as H_{min} and H_{max}, and two saturation threshold values, denoted as S_{min} and S_{max}, as the lower and upper bounds for thresholding the hue and saturation value, respectively. Then, by thresholding the hue and saturation values for each point on the image, we can obtain a set of curb feature points and then label their positions on a bi-level image I_{bi} for the use in the next step.

(2) Curb boundary point detection

With the bi-level image I_{bi} which includes the curb feature points, we can start to find the *inner* boundary points of the curb line. In image I_{bi}, we scan each pixel from top to down and from right to left in Win_{curb} as illustrated in Figure 6.1, and record the first found feature point as a curb boundary point. After scanning each row, we derive the curb boundary point position in the image and label them as red points, as shown in Figure 6.1.

(3) Computation of curb boundary point positions

After deriving the curb boundary points in the omni-image, we compute the boundary positions in the CCS. Suppose that a curb point is found on the ground, such as the point P illustrated in Figure 6.2. By the use of *Mirror A*, point P at coordinates (X, Y, Z) is projected onto the omni-image with an elevation angle α and an azimuth angle θ. As described previously in Section 5.2.1, we can represent the vector from the mirror center O_A to a space point P using the related elevation and azimuth angles by the use of Equation (5.3) which is repeated in the following:

$$V_p = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} \cos\alpha \times \cos\theta \\ \cos\alpha \times \sin\theta \times \cos\gamma + \sin\alpha \times \sin\gamma \\ -\cos\alpha \times \sin\theta + \sin\alpha \times \cos\gamma \end{bmatrix}. \qquad (5.3)$$

In other words, we can represent the position of P in the CCS with the form described by Equation (5.3). Besides, by the reason that the height H of the center of *Mirror A* is known in advance, we can further derive $Y = -H$. Hence, by the proportions among P_x, P_Y, and P_Z and known parameter Y, the position of the ground point P can be computed by dividing Equation (5.3) by P_y and then multiplying the result by $-H$, leading to the following equations which describe the position of P:

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

$$Y = -H;$$

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

Finally, the details of the above three major steps for curb line boundary extraction is described in the following algorithm, and the obtained curb boundary points positions in the VCS will be used to estimate the location of the curb line, as introduced in the next section.

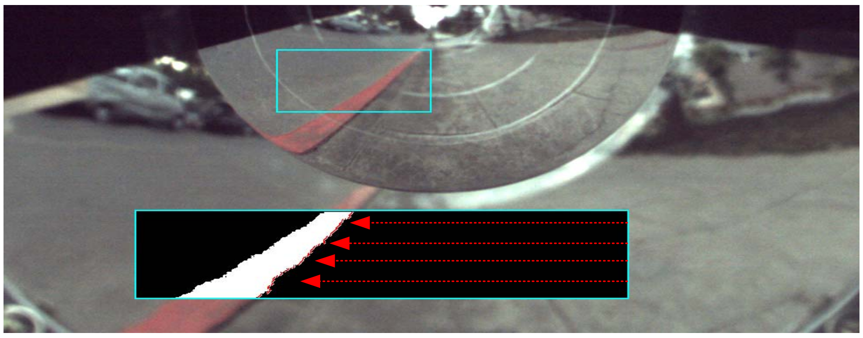


Figure 6.1 A detected curb line and the inner boundary points of the curb line on the omni-image.

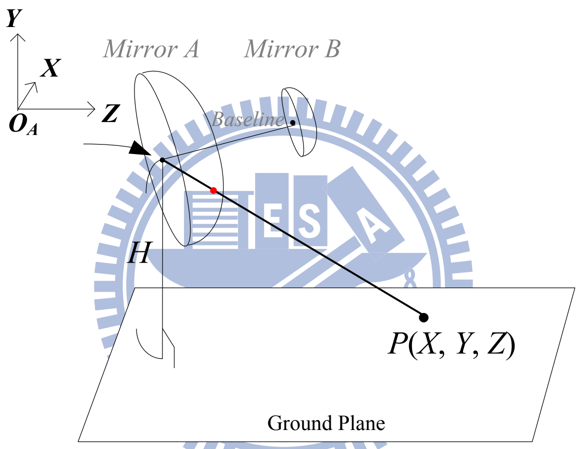


Figure 6.2 A ground point *P* projected onto *Mirror A*

Algorithm 6.1 ***Extraction of curb boundary points.***

Input: an input image I_{input}, two hue threshold values $H_{\min}$ and $H_{\max}$, two saturation threshold values $S_{\min}$ and $S_{\max}$, and an environment window Win_{curb}.

Output: a set S_{curb} of the positions of the curb boundary points in I_{input} in the VCS.

Steps.

Step 1. Initialize a bi-level image I_{bi}.

Step 2. Scan each point I_{uv} at coordinates (u, v) in Win_{curb} in I_{input}, and by Equation (5.22) compute its hue value h_{uv} and saturation value s_{uv}. If h_{uv} is between

Step 3. Conduct the erosion and dilation processes to the bi-level image I_{bi}.

Step 4. Scan each row from right to left in Win_{curb} in I_{bi} and find the first labeled point B_j at coordinates (u, v).

Step 5. Look up the pano-table to obtain the corresponding elevation and azimuth angle pair (α,θ), and compute the boundary point position B_{CCS} in the CCS by the use of Equation (6.1).

Step 6. Calculate the corresponding position B_{VCS} of the point in the VCS by the coordinate transformation described by Equation (3.6) with B_{CCS} as input, and record B_{VCS} into the set S_{curb}.

Step 7. Repeat Steps 4 through 6 until all rows in Win_{curb} have been scanned.

6.1.2 Curb line localization by dynamic color thresholding

To localize a detected curb line segment, firstly assume that the curb line segment in the image is a straight line. This is reasonable because the projection of the curb line on the omni-image is a small part of the whole curb line. Thus, we may approximate the detected curb line using a liner function by a line fitting technique. Specifically, using the boundary point positions by the method discussed previously in the last section, we can fit the data to a line L and obtain the equation of L as follows:

$$Y = ax + b, \tag{6.5}$$

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

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

$$b = \frac{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i y_i \sum_{i=1}^{n} x_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \quad (6.6)$$

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

Furthermore, by the use of this model, we can estimate a more precise position of the curb line using the proposed dynamic color thresholding technique mentioned previously. To be more specific, we conduct dynamic threshold adjustment for curb detection by adjusting the saturation threshold $S_{\min}$ in a pre-defined fixed range $[S_0, S_1]$. After using all possible pre-selected threshold values in this range to extract curb boundary points, we select the saturation threshold value with the minimum sum of errors in the result of fitting the curb boundary points with the computed line. The entire process for curb line location computation is shown in Figure 6.3 and the detailed algorithm is described in the following.

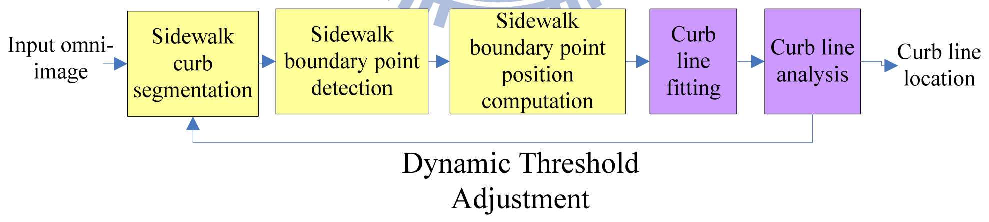

Input omni-image
Sidewalk curb segmentation
Sidewalk boundary point detection
Sidewalk boundary point position computation
Curb line fitting
Curb line analysis
Curb line location
Dynamic Threshold Adjustment

Figure 6.3 Process of curb line location computation

Algorithm 6.2 ***Curb line detection by dynamic color thresholding.***

Input: an input image I_{input}, and an environment window Win_{curb}.

Output: a slope angle θ of the curb line, and the distance d of the vehicle to the curb line.

Steps.

Step 1. Conduct curb boundary point extraction by the use of Algorithm 6.1 with I_{input}, Win_{curb}, two hue threshold values $H_{\min}$ and $H_{\max}$ and two saturation threshold values $S_{\min}$ and $S_{\max}$ as inputs to obtain a set S_{curb} of N boundary points, each denoted as c_i with coordinates (x_i, y_i) in the VCS.

Step 2. Use the line regression scheme to compute a line L by Equation (6.6) with c_i as inputs, where i is 1 through N and derive the equation of the best-fit line L as follows:

$$Y = aX + b \tag{6.4}$$

where the coefficients a and b are as described by Eqs. (6.6).

Step 3. Compute the sum of the errors S_e of fitting the boundary points c_i with L by the following equation:

$$S_e = \sum_{i=1}^{n}\left[y_i - \left(ax_i + b\right)\right] \tag{6.5}$$

Step 4. Adjust the threshold value S_{min} in the range $[S_0, S_1]$ and repeat Steps 1 through 3 until all possible pre-selected threshold values in $[S_0, S_1]$ have been computed.

Step 5. Select the fitting line L_{best} with the minimum sum of errors from the computed fitting lines obtained in Step 4.

Step 6. Compute the slope angle of L_{best} and the distance d to the vehicle by the following equation:

$$\theta = \tan^{-1}\left(\frac{1}{a}\right); \quad d = \frac{|b|}{\sqrt{1+a^2}}. \tag{6.6}$$

Step 7. Take θ and d as outputs.

6.1.3 Line following in navigation

By the obtained curb line location, the vehicle can conduct line following on sidewalks in the navigation process. The proposed scheme for line following aims to keep the navigation path at an appreciate distance to the curb line. As shown in Figure 6.4, we define the range [$Dist_1$, $Dist_2$] as the safe limits between the vehicle and the curb line. When the vehicle is at a position with a safe distance to the curb, we guide the vehicle to adjust its direction to be parallel to the curb. However, if the distance to the curb line is not in this range, we slow down the speed of the vehicle and turn the vehicle forward to get into the safe region progressively. The proposed line following process for vehicle navigation is described in the following algorithm.

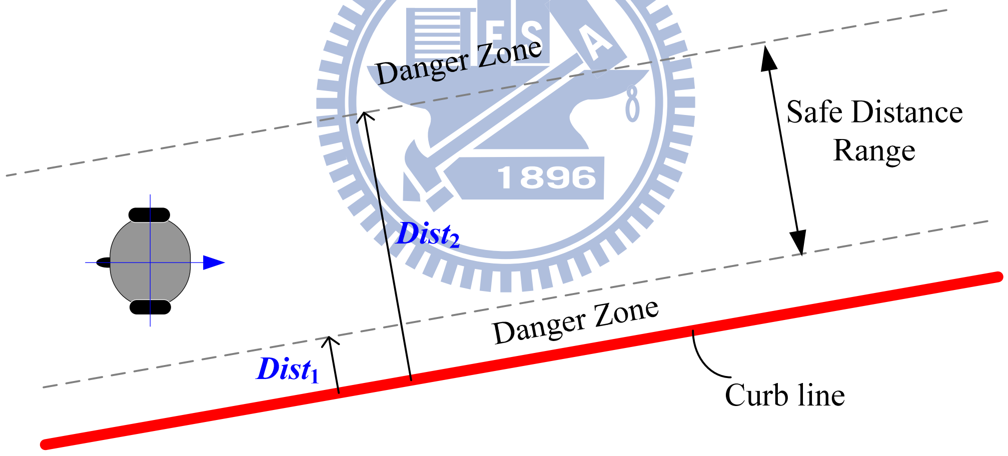


Figure 6.4 Illustration of line following strategy.

Algorithm 6.3 ***Curb Line following.***

Input: an input image I_{input}.

Output: none.

Steps.

Step 1. By the use of Algorithm 6.2, obtain the slope angle θ of the curb line and a distance d to the curb line.

Step 2. According to the distance d between the vehicle and the curb line, perform the following steps.

(1) If $d > Dist_2$, slow down the speed of the vehicle; and if the current vehicle direction is toward the safe region, exit; else, turn to the right for an angle of 5^o toward the safe region.

(2) If $d < Dist_1$, slow down the speed of the vehicle; and if the current vehicle direction is toward the safe region, exit; else, turn to the left for an angle of 5^o forward the safe region.

(3) If $Dist_1 \leqq d \leqq Dist_2$, modify the vehicle direction by the use of θ to make it parallel to the curb line.

6.1.4 Experimental results of curb detection

Some experimental results of curb detection using the proposed method are given in this section. An input omni-image with curb line is shown in Figure 6.5. By the proposed method, the curb segmentation result with original threshold parameters is shown in Figure 6.6(a). In addition, a better curb segmentation result adopting the dynamic threshold adjustment technique is shown in Figure 6.6(b). Finally, the extracted curb boundary points and computed best-fit line from Figure 6.6(b) are shown in Figure 6.7.

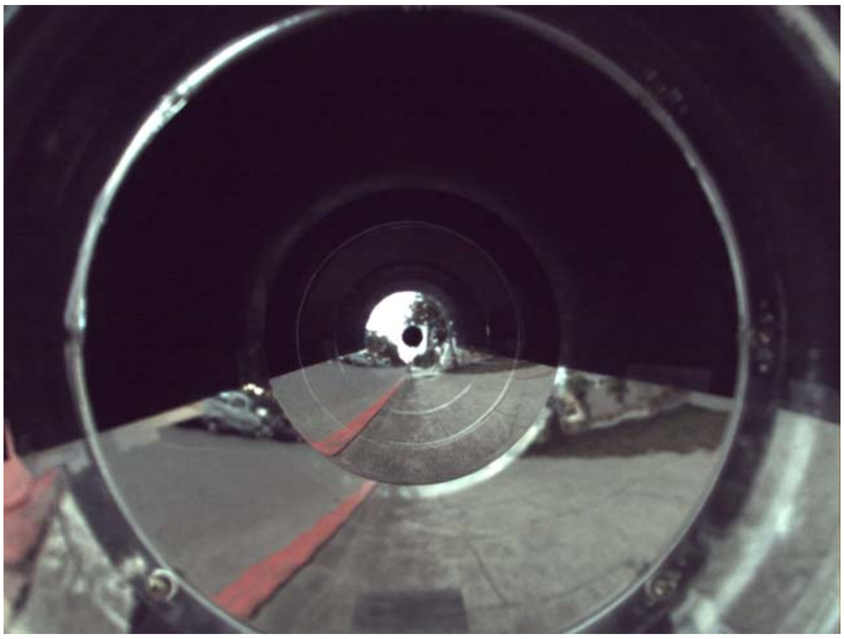


Figure 6.5 An input omni-image with curb line landmark.

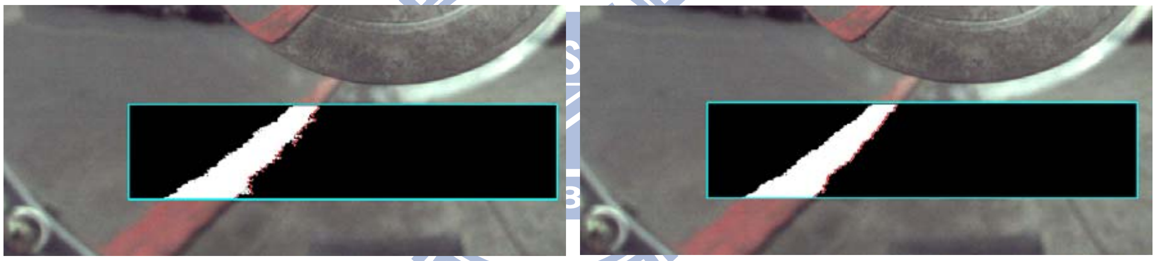

(a) (b)

Figure 6.6 Two result images of curb segmentation with different threshold values (a) The segmentation result with original threshold values. (b) The segmentation result image by dynamic thresholding.

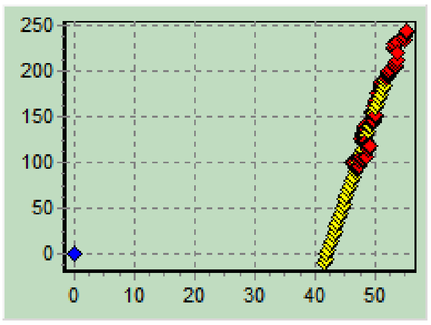


Figure 6.7 Illustration of extracted curb boundary points and a bet fitting line (the yellow dots).

6.2 Proposed Technique of Obstacle Avoidance

The idea of the proposed obstacle detection technique is based on the use of the disparity resulting from the separation of *Mirrors A* and *B*. Because the used two-mirror omni-camera is placed at a fixed position and slanted up for a fixed angle on the autonomous vehicle, a ground point *P* will be projected by the two mirrors onto the camera at two specific different image positions as shown in Figure 6.8(a). In other words, we can find the same space point *P* at these two image positions simultaneously. Thus, we can record in advance the relation between two *corresponding ground points* in the two mirrors and use them to inspect an object which is not flat on the ground. More specifically, if an object with a height is projected by the two mirrors onto the omni-images, we can detect it by looking up recoded corresponding ground positions on the two mirrors. As shown in Figure 6.8(b), instead of the ground point *G* we find out another space point *F* on the obstacle which is projected by *Mirror A* onto the image.

Simply speaking, for obstacle detection in this study, our purpose is to construct a specific table, we call *ground matching table*, which records the relationship between the ground points on the image region of *Mirror A* and the corresponding ground points on the image region of *Mirror B* in the omni-image as shown in Figure 6.9. The proposed method for creating a ground matching table is introduced in Section 6.2.1. Next, by the use of the established ground matching table, we can conduct obstacle detection and localization conveniently for vehicle navigation, as described in Section 6.2.2. Finally, the procedure of obstacle avoidance is introduced in Section 6.2.3.

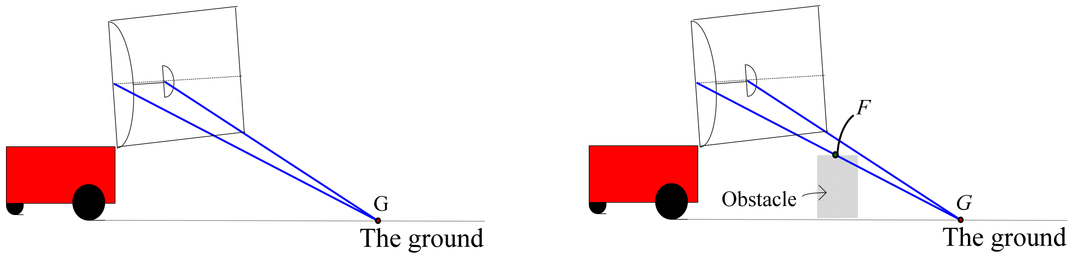


Figure 6.8 Two side view of the vehicle system and a ground *G*. (a) Without obstacles. (b) With an obstacle in front of the vehicle.

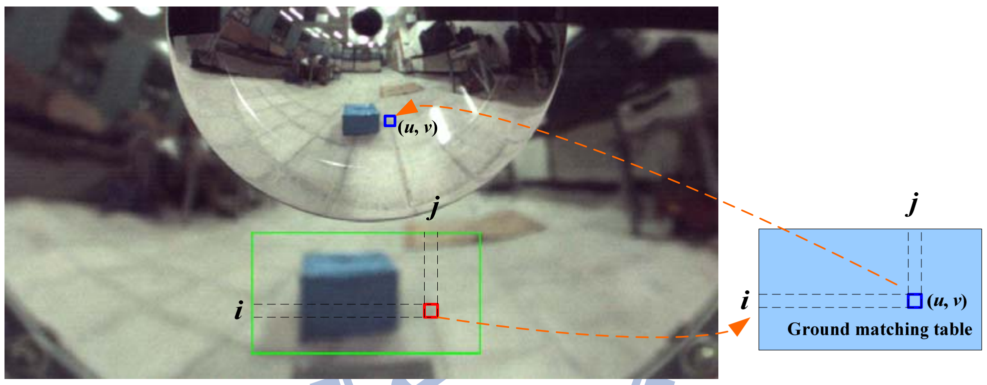


Figure 6.9 Illustration of the ground matching table.

6.2.1 Calibration process for obtaining corresponding ground points in two mirrors

At the beginning of the calibration process, we specify a set of environment windows Win_{obs} for use in the calibration process as well as in the obstacle detection process in the navigation process. The purpose of the calibration process is to construct a ground matching table for the use in win_B of Win_{obs} and the entry in the table records the image point position of a corresponding ground point in the image region of *Mirror B*. In this study, we propose a *semi-automatic* calibration method for obtaining corresponding ground points in *Mirrors A* and *B*.

To be more specific, a specific line, which we call a *calibration line*, with a special color is placed on the ground in front of the vehicle as shown in Figure 6.10 and can be seen on both regions of the two mirrors in the omni-image. Next, by the property of rotational invariance of omni-imaging, we can obtain the corresponding ground points in the two mirror regions from an input omni-image automatically as illustrated in Figure 6.11, and record them on the relevant entries in the ground matching table by the following algorithm.

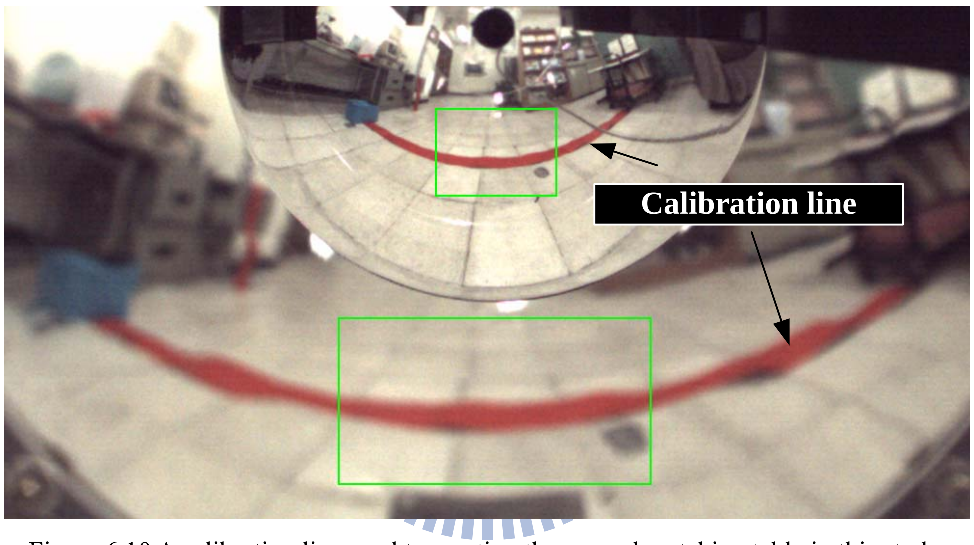


Figure 6.10 A calibration line used to creating the ground matching table in this study.

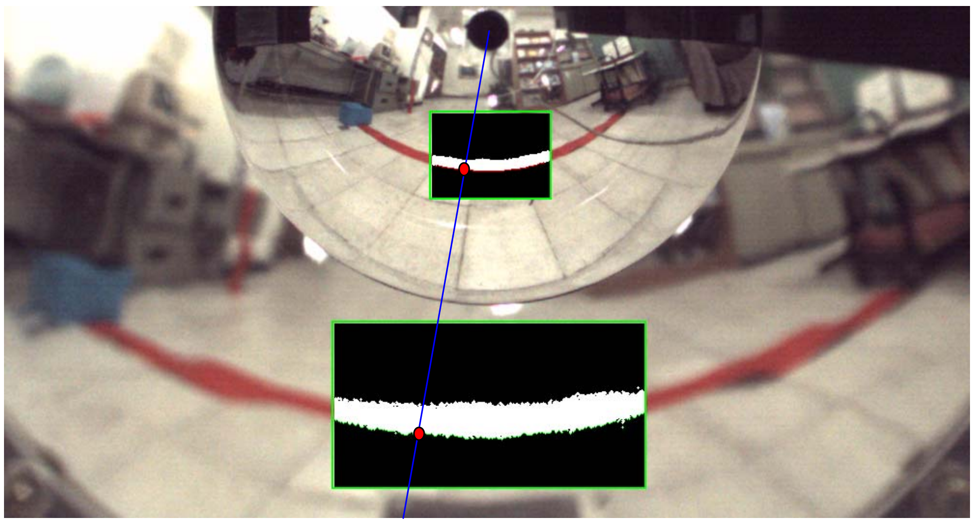


Figure 6.11 Illustration of detecting corresponding ground points on a calibration line by the use of rotational property on the omni-image.

Algorithm 6.4 ***Ground matching table recording.***

Input: an input omni-image I_{input}, and a ground matching table.

Output: Updated ground matching table.

Steps.

Step 1. By the use of the previously-mentioned HSI classification method, classify the feature points of the calibration line on Win_{obs} in I_{input} and label the feature points in a bi-level image I_{bi}.

Step 2. Scan each pixel from bottom to top for all columns on win_{B} of Win_{obs} in I_{bi} to find the first feature point f_{Bi} at coordinates (u_i, v_i); and record f_{Bi} into a set S_B.

Step 3. For each point f_{Bi} in S_B, compute its azimuth angle θ by Equation (2.3); scan accordingly on the same radial direction from far to near in the region of *Mirror B* to find the first feature point f_{Si}.

Step 4. Record the image position of f_{Si} into the relevant entire in the ground matching table; and go to Step 3 if the elements in S_B have not been exhausted.

By conducting the above algorithm continually while the vehicle is moving around, we can gradually create the ground matching table. With sufficient input images taken at different positions by manually moving the vehicle, a complete ground matching table can be obtained at the end of the calibration process.

6.2.2 Obstacle detection process

For obstacle detection during vehicle navigation, it is assumed that each obstacle which blocks the vehicle has a surface with a single color. Also, the vehicle should be close enough to "see" it on the two regions of *Mirrors A* and *B* in the image. The idea of the proposed method to detect an obstacle is to “learn” the color information of the obstacle points which are detected by the use of the ground matching table. Then, by analyzing the color information of these points, we can derive the color of the obstacle and extract the entire obstacle contour. Finally, we can localize the detected obstacle position by extracting the boundary points on the bottom of the obstacle on the floor. The above-mentioned obstacle detection process consisting of four major steps are described in the following, and a flow chart illustrating the process is shown in Figure 6.12.

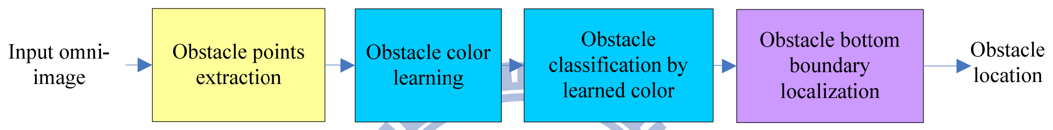

Input omni-image
Obstacle points extraction
Obstacle color learning
Obstacle classification by learned color
Obstacle bottom boundary localization
Obstacle location

Figure 6.12 Proposed process for obstacle detection.

(1) Obstacle point extraction by the use of the ground matching table

By the use of the ground match table, we can obtain the corresponding ground points on the two mirror regions in the omni-image. By the difference of the intensity, we can find out a space point like F on the obstacle surface projected on the region of *Mirror A* in the omni-image as shown in Figure 6.13. We collect all detected obstacle points on win_B of Win_{obs} and label them in a bi-level image I_{bi}. After reducing noise by conducting some image processing on I_{bi}, we record all remaining obstacle points in I_{bi} into a set S_{obs}.

(2) Obstacle color information learning

By RGB-to-HSI conversion, we compute the average hue value H_{obs} and a hue variance var_{obs} from all image points in S_{obs}. We record H_{obs} and var_{obs} as the color information of the obstacle.

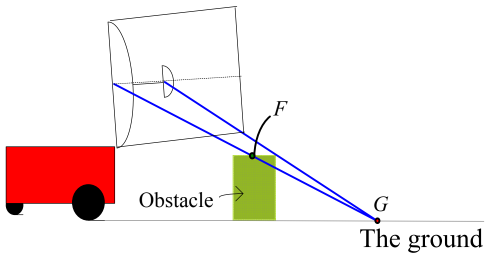


Figure 6.13 An obstacle point F found dun to the difference of the intensity.

(3) Obstacle detection by the learned color information

By the use of the HSI color model, we classify the obstacle feature points by the learned color information of the obstacle. After that, we obtain the obstacle feature points in the omni-image.

(4) Obstacle location computation

With the image which includes the detected obstacle points, we detect outlier feature points on the bottom of the obstacle by a method which scans each radial line in win_B of Win_{obs}, and takes the first found feature point as the outlier point as shown in Figure 6.14. By the reason that these bottom boundary points are on the ground, we use the previously-mentioned method using Equation (6.1) to compute their positions. Finally, we calculate the average position of the found bottom boundary points as the obstacle location G_{obs}.

The detailed algorithm of the above four steps for conducting obstacle localization is described in the following. At the end of the algorithm, we can obtain the obstacle location and guide the vehicle to dodge the found obstacle as discussed in the next section.

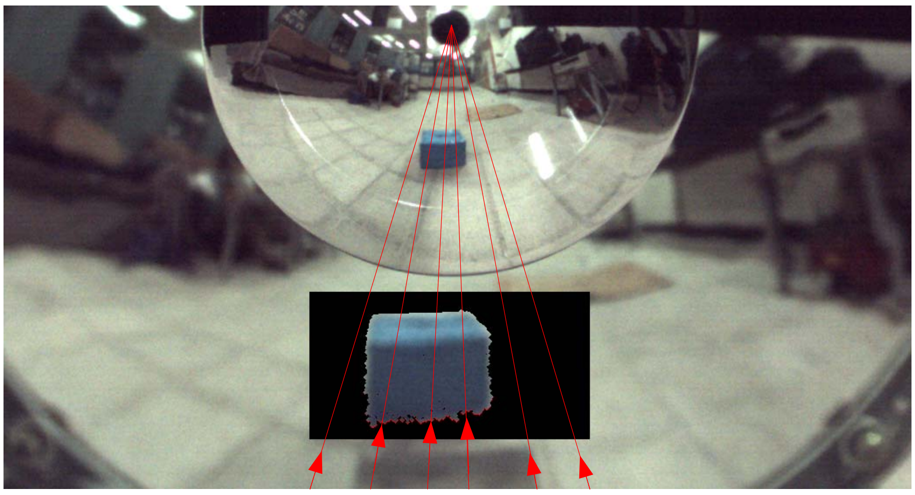


Figure 6.14 Illustration of the method to find boundary points on the bottom of obstacle.

Algorithm 6.5 ***Obstacle detection and localization.***

Input: an input omni-image I_{input}, a set of environment windows Win_{obs}, and a ground match table.

Output: an obstacle location G_{obs} in the CCS.

Steps.

Step 1. Scan each point I_B at coordinates (u_B, v_B) in win_B of Win_{obs} in I_{input}, and look up the ground matching table to find the corresponding ground point I_S at coordinates (u_S, v_S).

Step 2. (*Detecting obstacle points*) Compute the intensity Y_B of I_B and Y_S of I_S; if $|Y_B - Y_S|$ is larger than a threshold, label I_B in a bi-level image I_{bi} which keeps obstacle pixels; go to Step 1 until all points in win_B of Win_{obs} are scanned.

Step 3. Apply the operations of erosion, dilation, and connected component labeling to the bi-level image I_{bi}.

Step 4. (*Deciding whether an obstacle has been found*) If the number of all labeled points in I_{bi} is larger than a threshold, then regard the points as an obstacle and record them into a set S_B.

Step 5. By the image points in S_B, compute the mean hue value H_{obs} and the hue variance value var_{obs} by the use of the conversion described by Equation (5.22).

Step 6. (*Finding obstacle points more precisely*) For each pixel on win_B of Win_{obs} in the original input image I_{input}, classify the obstacle feature points by the following rule:

if the computed hue value of a pixel is between the range $[H_{obs} - 2\times var_{obs}, H_{obs} + 2\times var_{obs}]$, then take the pixel as an obstacle point;

and label all obstacle points in another bi-level image I_{Hue}.

Step 7. Scan each radial line in win_B of Win_{obs} from far to near in I_{Hue}, and record the first found obstacle point as a bottom boundary point I_{obs} into a set S_{obs}.

Step 8. For all points I_{obs} in S_{obs}, look up the pano-mapping table to find corresponding elevation and azimuth pair (α, θ).

Step 9. Compute the related boundary point position C_{obs} in the CCS by Equation (6.1).

Step 10. Calculate the average position of the bottom boundary points as the obstacle position G_{obs} and take G_{obs} as output.

6.2.3 Obstacle avoidance process

In this section, we introduce the method of obstacle avoidance for vehicle navigation. The obstacle avoidance process is conducted in the cases of finding a localized light pole, reading a fixed obstacle position from the navigation path, or detecting a dynamic obstacle in the navigation process. The proposed strategy of the obstacle avoidance is to insert additional path nodes for obstacle avoidance into the navigation path to guide the vehicle to change its path to pass the obstacle. As shown in Figure 6.15, after obtaining the obstacle location, the path node $Node_{avoid}$ is placed at the left-hand side with a pre-defined distance *Dist* with respect to the obstacle position as shown in the figure. Finally, the vehicle will navigate to $Node_{avoid}$ firstly and then the original destination node $Node_{i+1}$.

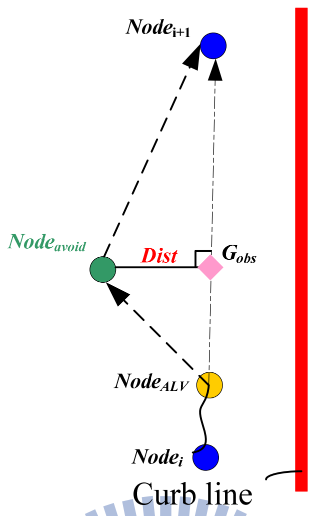
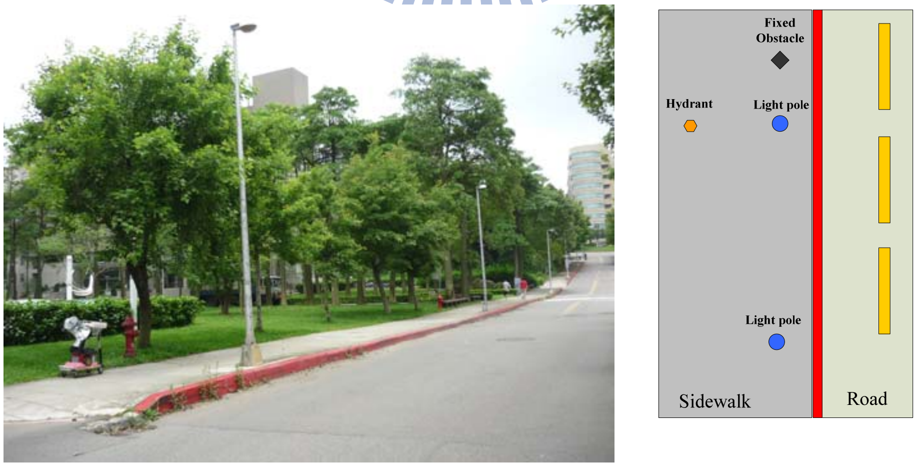


Figure 6.15 Illustration of inserting a path node $Node_{avoid}$ for obstacle avoidance in the original navigation path.

Chapter 7
Experimental Results and Discussions

7.1 Experimental Results

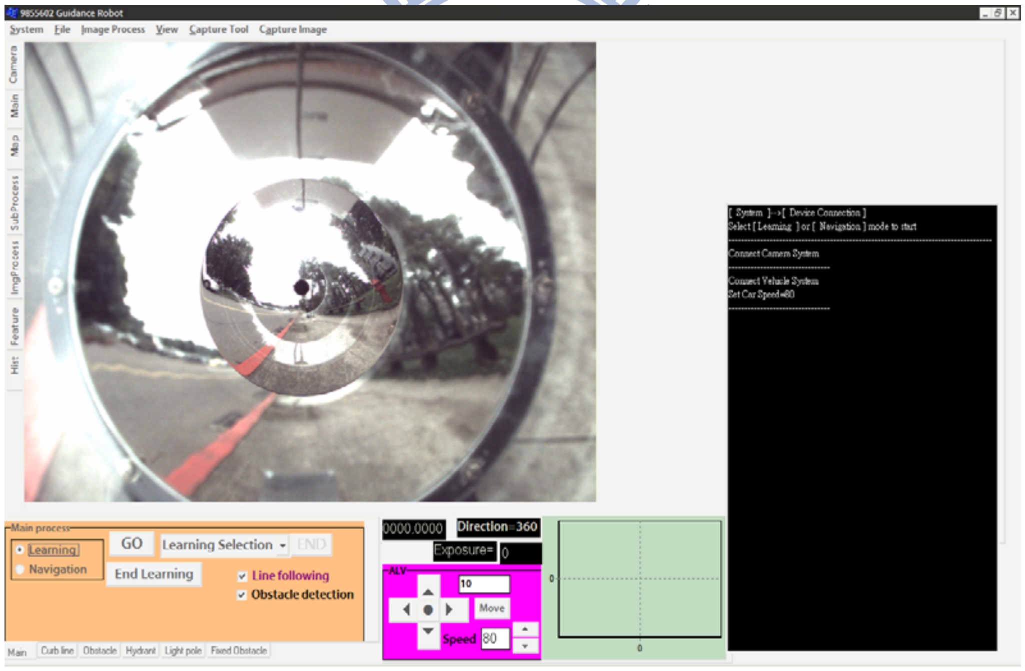
In this section, we will show some experimental results of the proposed vehicle navigation system in the learning and navigation processes. The experimental environment was an outdoor sidewalks in National Chiao Tung University as shown in Figure 7.1(a). An illustration of the environment consisting of a gray sidewalk, a red curb line, and some landmarks is 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 the learning interface as shown in Figure 7.2 to construct unknown navigation environment. The vehicle navigated forward along the detected curb line, and the trainer followed the vehicle to conduct learning tasks on the vehicle system. After arriving at appropriate locations on the sidewalk, the vehicle was instructed to learn the positions of specific landmarks such hydrants and light poles. Some landmark detection results are shown in Figure 7.3. In addition, the position of a fixed obstacle, a cover of an underground sewer, has also been recorded by the method of manually localizing its position on the omni-image as shown in Figure 7.4. At the end of the learning process, a navigation map with a navigation path and other environment landmarks was created, as illustrated in Figure 7.5.



```
[ System ]-->[ Device Connection ]
Select [ Learning ] or [ Navigation ] mode to start
------------------------------------------------------------
Connect Camera System
-----------------------
Connect Vehicle System
Set Car Speed=80
-----------------------
```

Figure 7.2 The Learning interface of the proposed vehicle system.

In the navigation process, the vehicle started from the same origin like that in the learning process and navigated alone the recorded navigation path nodes with the method of curb line following. By conducting curb detection, the vehicle kept its path to be parallel to the curb. An example of curb detection results is given in Figure 7.6.

Next, by the use of the dynamic exposure adjustment technique, the vehicle detected the appointed landmarks with a suitable illumination and localized its position. Some results of hydrant and light pole landmark detection are shown in Figure 7.7. In addition, for a detected light pole and a recoded fixed obstacle, the vehicle adopted the obstacle avoidance procedure to dodge them as shown in Figures 7.8 and 7.9, respectively. Also, after detecting a dynamic obstacle in the navigation path as shown in Figures 7.10, the vehicle created a new path with avoidance nodes to pass through the obstacle as illustrated in Figure 7.11. 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 7.12.

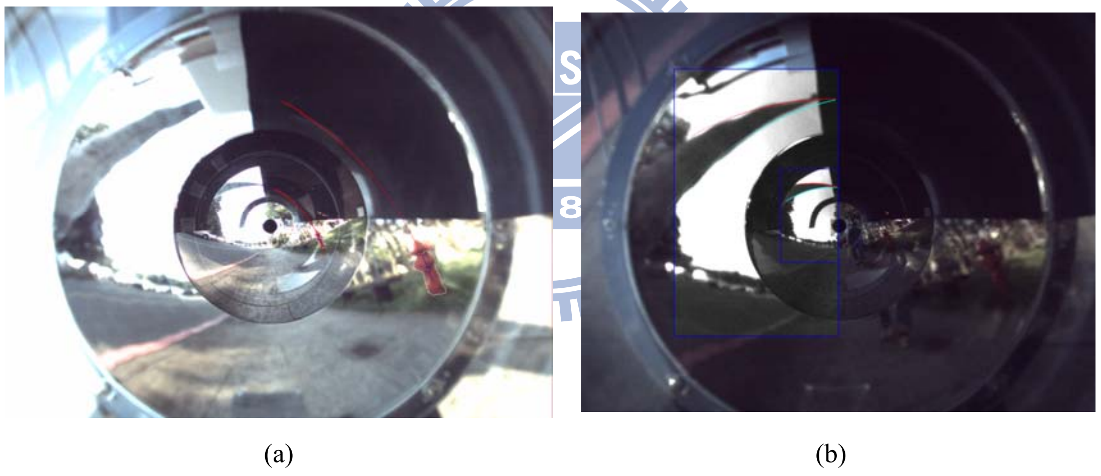
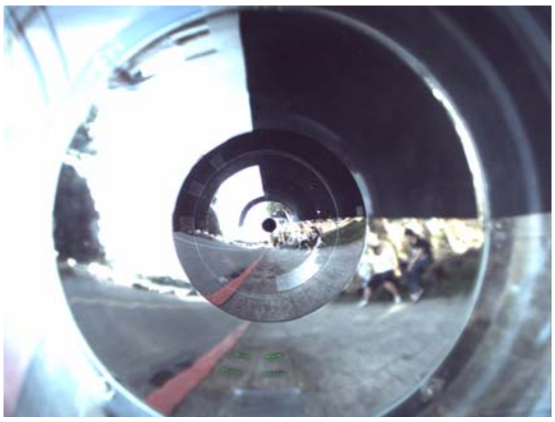
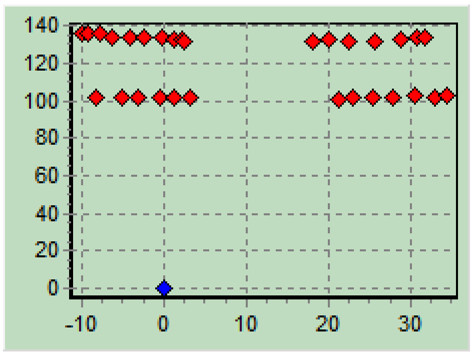


Figure 7.3 Images of some results of landmark detection in the learning process. (a) A hydrant detection result with axes of the hydrant drawn in red. (b) A light pole detection result with pole boundaries drawn in blue.





(a) (b)

Figure 7.4 Learning of the fixed obstacle. (a) The fixed obstacle position on the omni-image (Lime points clicked by the trainer). (b) Computed fixed obstacle positions in the real world.

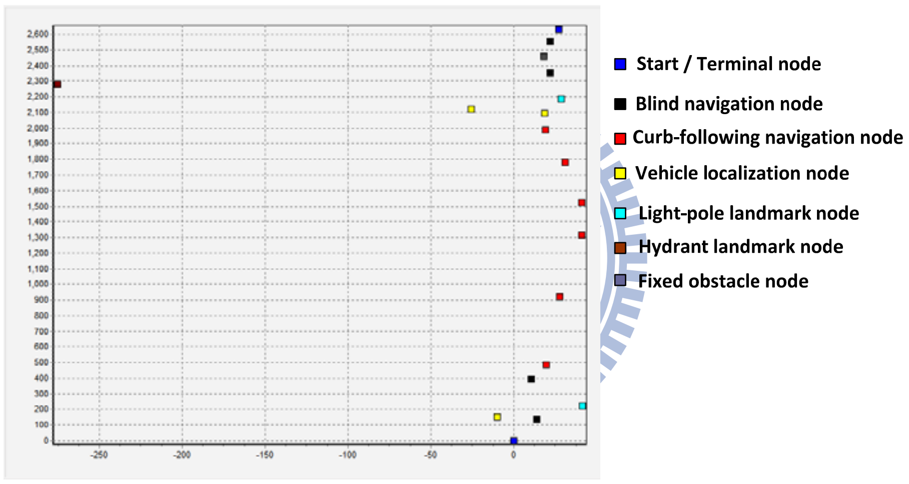


Figure 7.5 Illustration of the learned navigation map.

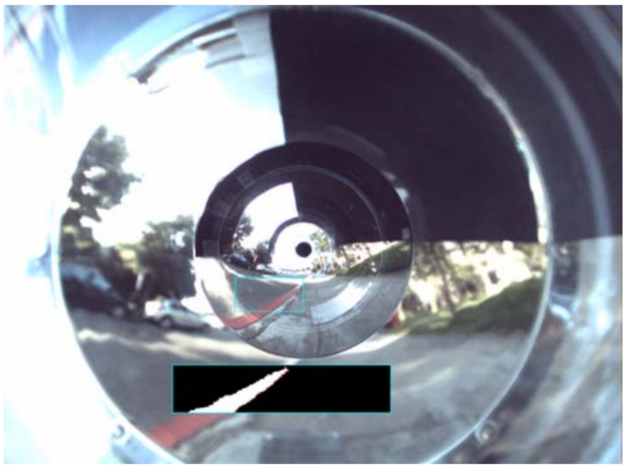
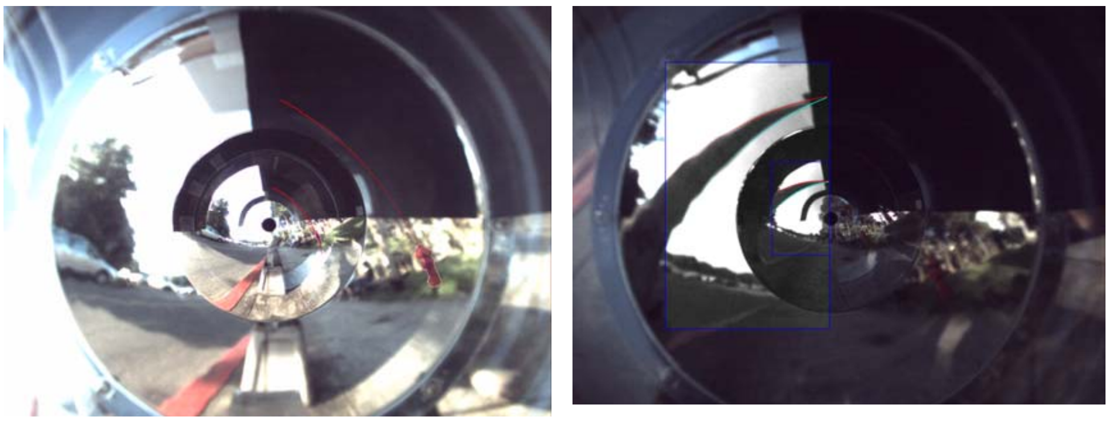
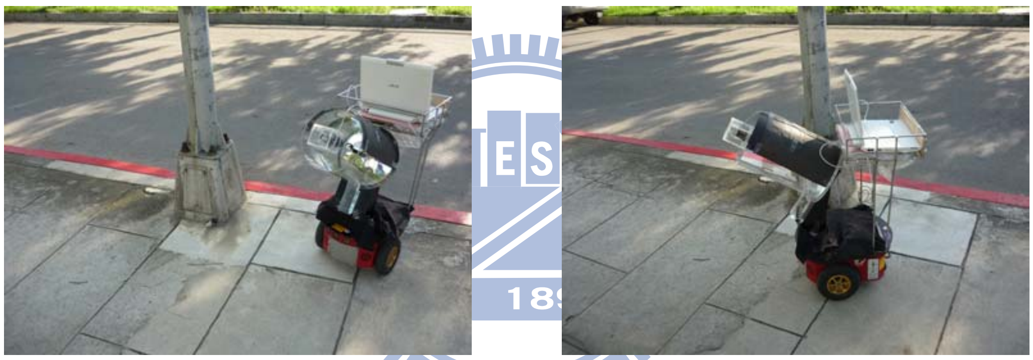


Figure 7.6 Image of a result of curb line detection.

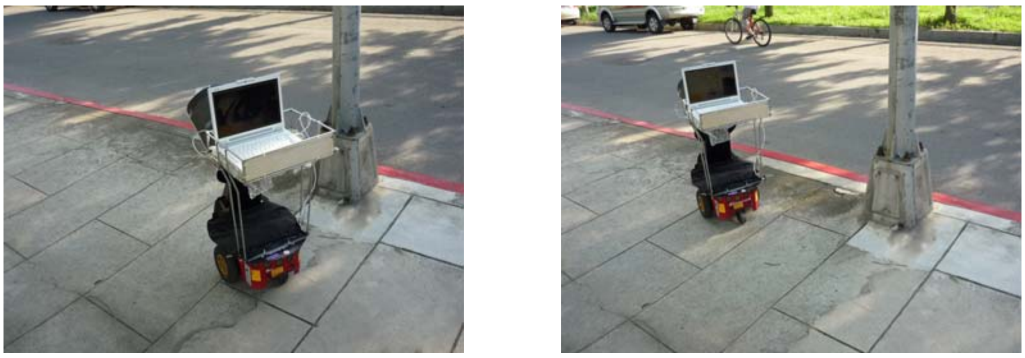


(a) (b)

Figure 7.7 Images of results of landmark detection for vehicle localization in the navigation process. (a) Hydrant detection results. (b) Light pole detection results.



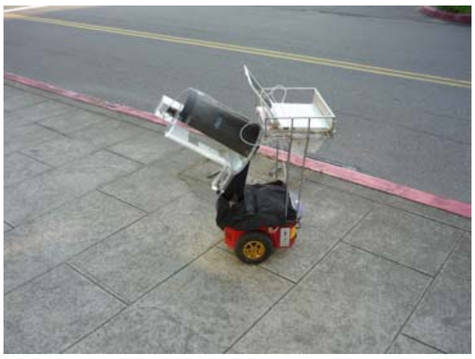
(a) (b)



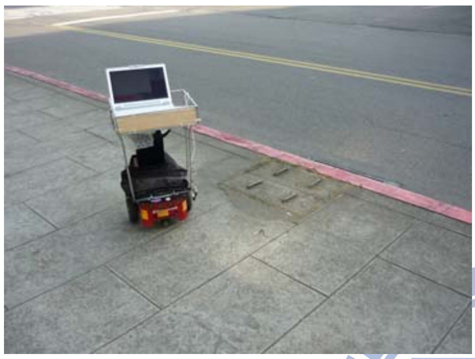
(c) (d)

Figure 7.8 The vehicle detects the light pole and conduct avoidance procedure in the navigation process. (a) ~ (d) show the process of light pole avoidance.

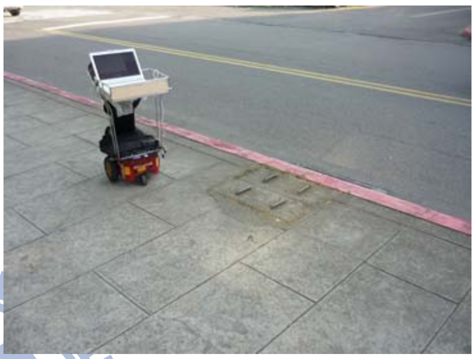
(a)



(b)



(c)



(d)

Figure 7.9 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.

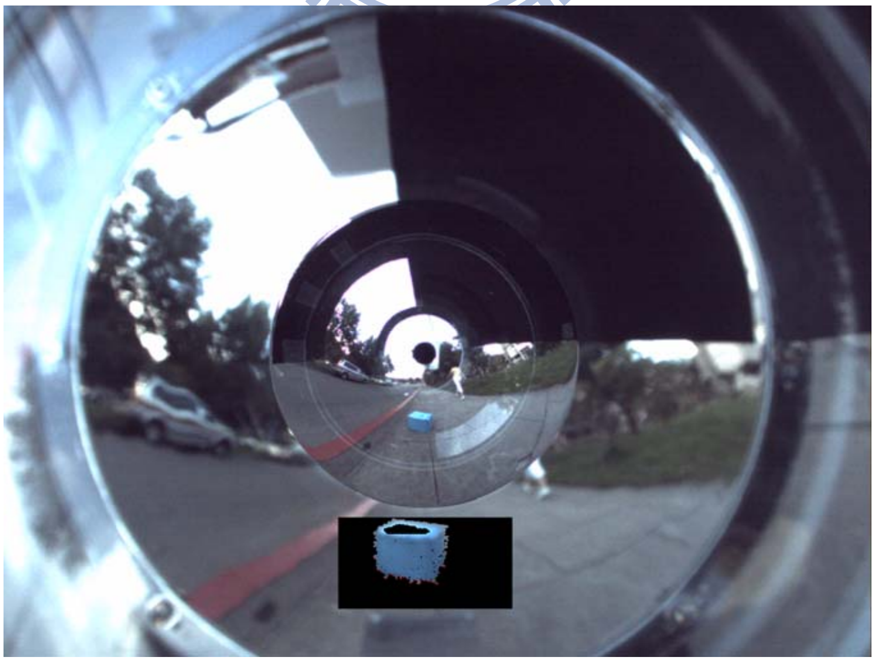


Figure 7.10 Image of a result of dynamic obstacle detection.

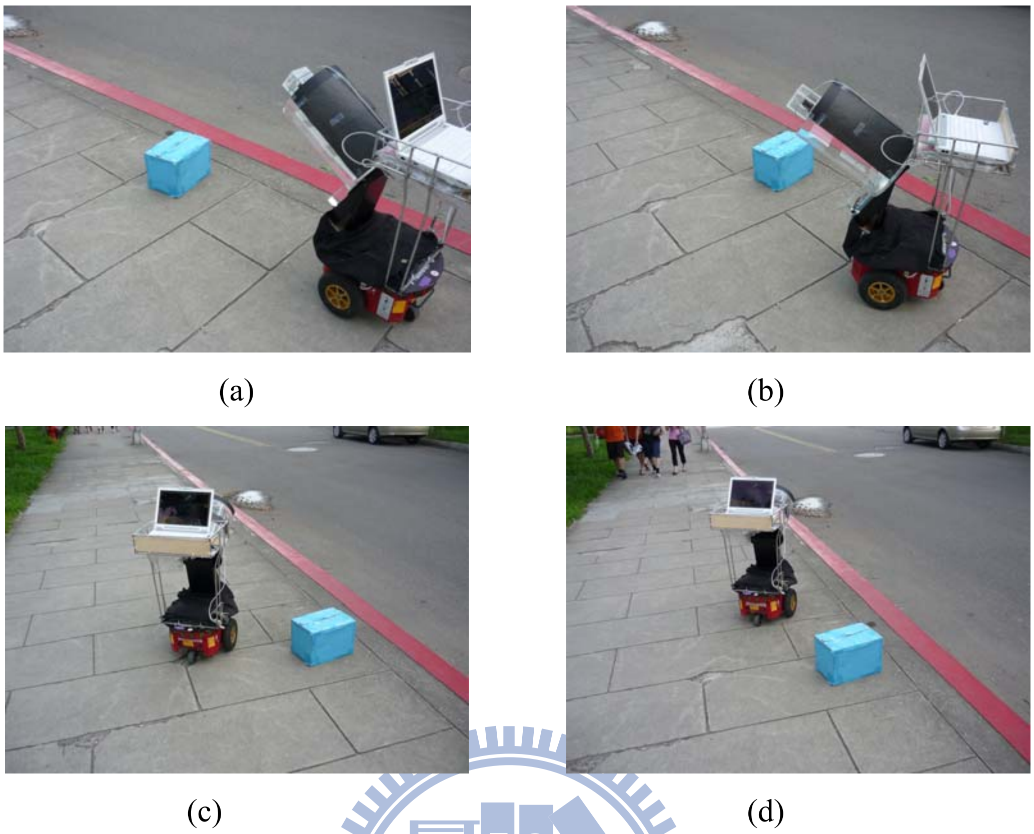


Figure 7.11 The process of dynamic obstacle avoidance in the navigation path. (a) Starting to conduct obstacle detection. (b) Turn left to dodge the localized obstacle. (c) A side view of the avoidance process. (d) Completing the avoidance process.

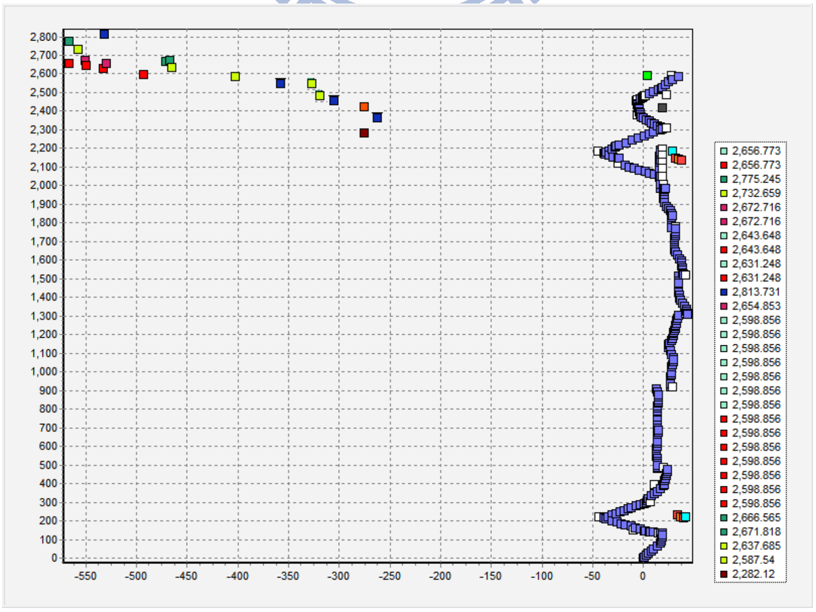


Figure 7.12 The recorded path map in the navigation process. (Blue points represent the vehicle path and other points with different color represent different localized landmark position in different detecting)

7.2 Discussions

By analyzing the experimental results of the vehicle navigation, we find some problems. Firstly, for sidewalk curb detection, we detect the specific curb with a red 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, when dynamically adjusting the exposure to obtain an appropriate exposure value for conducting different landmark detection works, it may take some time to wait the camera system to adjust to the appointed exposure value. A possible way to solve this problem is to use another camera with quicker response time in the camera parameter adjustment process. Furthermore, the light reflection caused by the plastic camera enclosure creates 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. Finally, more experiments in different environments should also be conducted to test our system more thoroughly.

Chapter 8
Conclusions and Suggestions for Future Works

8.1 Conclusions

A vision-based autonomous vehicle navigation system for use as a machine guide dog in outdoor environments has been proposed in this study. To implement such as a system, several techniques has been proposed.

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

Next, a new space line detection technique based on the pano-mapping technique has been proposed. The space line with a curve projection on the omni-image can be detected by the use of analytic formulas and the Hough transform technique. In addition, for the vertical space line which exists in landmarks like light poles and hydrants, we can further compute its position directly according to omni-imaging and pan-mapping techniques.

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 calibrate the odometer reading of the vehicle orientation. Next, hydrant and light pole detection techniques have been proposed. The vertical 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 outdoor environments, techniques for dynamic exposure and threshold adjustments have also been proposed, which can be employed to adjust the system's parameters to meet different lighting conditions. Also have been proposed is a new obstacle detection technique, which can be used to find dynamic obstacles on the sidewalk for safer vehicle navigation. Specifically, by the use of a ground matching table, the vehicle can detect obstacles on the path and localize its position for realtime path planning to conduct an obstacle avoidance process automatically.

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

8.2 Suggestions for Future Works

According to our experience obtained in this study, in the following we point out some related interesting issues worth further investigation in the future:

(1) the proposed line detection may be adopted to detect and localize other kinds of landmarks with vertical line features;

(2) it is interesting to use different artificial or nature landmarks, such as a tree, a signboard, a pillar, or a building, to conduct vehicle navigation in outdoor environments;

(3) the curb line detection technique may be improved by learning features of other

(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 seems necessary to add the capability of warning the user in danger conditions;

(6) dynamic obstacles detection technique may be improved using other techniques such as template matching;

(7) it is desired to design a new camera system which owns a smaller size.

References

[1] S. Shoval, J. Borenstein, and Y. Koren, “The Navbelt - A computerized travel Aid for the blind,” *Proceedings of the RESNA '93 Conference*, pp. 240-242, Las Vegas, Nevada, USA, June 13-18, 1993.

[2] J. Borenstein and I Ulrich, “The GuideCan — 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.

[3] H. Mori, and M. Sano, "A guide dog robot Harunobu-5 following a person," *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, vol. 1, pp. 397-402, Osaka, Japan, 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] L. Ran, S. Helal, 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.

[6] M. Kam, X. Zhu, and P. Kalata, “Sensor fusion for mobile robot navigation,” *Proceeding of the IEEE*, vol. 85, no. 1, pp. 108-119, 1997.

[7] 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*

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

[9] M. C. Chen and W. S. Tsai, "Vision-based security patrolling in indoor environments using autonomous vehicles," *M. S. Thesis*, Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, June 2005.

[10] 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.

[11] 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.

[12] S. Pagnottelli, S. Taraglio, P. Valigi, and A. Zanela, "Visual and laser sensory data fusion for outdoor robot localisation and navigation," *Proceedings of 12th International Conference on Advanced Robotics,* pp. 171-177, Seattle, Washington, USA, July 2005.

[13] Taiwan Foundation for the Blind:
http://www.tfb.org.tw/english/index.html.

[14] Taiwan Guide Dog Association:
http://www.guidedog.org.tw/.

[15] S. E. Yu and D. Kim, "Distance estimation method with snapshot landmark

[16] T. Tasaki, S. Tokura, T. Sonoura, F. Ozaki ,and N. Matsuhira, “Mobile robot self-localization based on tracked scale and rotation invariant feature points by using an omnidirectional camera”, *Proceedings of 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5202-5207, Taipei, Taiwan, Oct. 18–22, 2010.

[17] 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.

[18] B. Siemiątkowska and R. Chojecki, “Mobile robot localization based on omnicamera,” *European Conference 5TH IFAC/EURON Symposium on Intelligent Autonomous Vehicles (IAV2004)*, Lisbon, Portugal, July 5-7, 2004.

[19] J. Courbon, Y. Mezouar, and P. Martinet, “Autonomous navigation of vehicles from a visual memory using a generic camera model,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 392-402, Sept. 2009.

[20] A. Merke, S. Welker, and M. Riedmiller, “Line based robot localization under natural light conditions,” *Proceedings* of *ECAI Workshop on Agents in Dynamic and Real Time Environments*, pp. 402-409, Valencia, Spain, 2004.

[21] S. Kumar, “Binocular stereo vision based obstacle avoidance algorithm for autonomous Mobile Robots,” *Proceedings of IEEE International Advance Computing Conference*, pp. 254-259, Patiala, India, Mar. 2009.

[22] D. Fernandez and A. Price, “Visual detection and tracking of poorly structured dirt roads,” *Proceedings of 12th International Conference on Advanced Robotics*, pp. 553-560, Seattle, Washington, USA,July 2005.

[23] D. Kim, J. Sun, S. M. Oh, J. M. Rehg, and A. F. Bobick, “Traversability classification using unsupervised on-line visual learning for outdoor robot navigation,” *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pp. 518–525, May 2006.

[24] Q. Mühlbauer, S. Sosnowski, T. Xu, T. Zhang, and K. Kühnlenz, M. Buss, “Navigation through urban environments by visual perception and interaction,” *Proceeding of IEEE International Conference on Robotics and Automation*, pp. 1907–1913, Kobe, Japan, 2009.

[25] 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.

[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.

[27] J. K. Huang and W. H. Tsai, “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.

[28] K.C. Chen and W. H. Tsai, “A study on autonomous vehicle navigation by 2D object image matching and 3D computer vision analysis for indoor security patrolling applications,” *Proceedings of 2007 Conference on Computer Vision, Graphics and Image Processing*, Miaoli, Taiwan, Aug. 2007.