國立交通大學

資訊科學系

碩 士 論 文

利 用 自 動 車 作 基 於 視 覺 之 室 內 安 全 巡 邏

Vision-Based Security Patrolling in Indoor Environments Using Autonomous Vehicles

研 究 生：陳明哲

指導教授：蔡文祥　教授

中 華 民 國 九 十 四 年 六 月

利 用 自 動 車 作 基 於 視 覺 之 室 內 安 全 巡 邏
Vision-Based Security Patrolling in Indoor Environments Using Autonomous Vehicles

研 究 生：陳明哲　　Student：Ming-Che Chen
指導教授：蔡文祥　　Advisor：Wen-Hsiang Tsai

國 立 交 通 大 學
資 訊 科 學系
碩 士 論 文

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

Computer and Information Science

June 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

利用自動車作基於視覺之室內安全巡邏

研究生：陳明哲　　　　　　　　　指導教授：蔡文祥 博士

國立交通大學資訊科學研究系

摘 要

本研究主要是提出一套基於電腦視覺技術，可讓自動車航行在室內環境中且具有安全監控的能力。我們利用一台小型自動車為實驗平台，並且利用無線操控的方式讓自動車航行在室內的環境中。我們設計了一套人簡單且有效的學習方式，讓使用者可以操控自動車行走且自由的選擇要被監控的物品或是房門，自動車將會自動的建立導航的地圖。在學習完路線、物品和門的資料之後，我們運用了一套完整的導航策略來完成安全巡邏的任務。這個策略包括了監控物品的安全偵測和房門情況的識別，利用自動車判斷物品是否遭竊或者是門被開起，我們能夠立刻發出警報給使用者，已達成安全巡邏的任務。最後我們以成功的學習與導航實驗結果證明本系統的完整性與可行性。

Vision-Based Security Patrolling in Indoor Environments Using Autonomous Vehicles

Student: Ming-Che Chen Advisor: Dr. Wen-Hsiang Tsai

Department of Computer and Information Science
National Chiao Tung University

ABSTRACT

A vision-based approach to security patrolling in indoor environments using autonomous vehicles is proposed. A small vehicle with wireless control and image grabbing capabilities is used as a test bed. Three stages of security patrolling are proposed. First, a simple learning strategy is designed for flexible and effective learning of reachable spots and monitored objects in indoor environments. Accordingly, a planned path is obtained, and monitored objects and doors are specified by analyzing user commands. Next, following the learned path, the vehicle can accomplish specified navigation sessions. Two different kinds of methods, mechanic error correction modeling and vehicle position modification by positions of monitored objects, are proposed for navigation accuracy maintenance. Finally, an object matching algorithm is used for checking the existence of monitoring objects and the opening status of doors. All the experimental results show flexibility and feasibility of the proposed approach for the application of indoor security patrolling.

ACKNOWLEDGEMENTS

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

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

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

CONTENTS

LIST OF FIGURES

Chapter 1
Introduction

1.1 Motivation

The development of autonomous vehicles or mobile robots is paid much attention recently. Many researchers are devoted to developing functions of vehicles or robots such that they can do works conducted by human beings. Among applications of the autonomous vehicle, security patrolling is a practical function for human beings.

We often install cameras in the house to monitor the indoor situation nowadays. We only depend on looking at videos to find out thieves when we discover that thieves broke into the house or objects were stolen. It takes much time and manpower, and lacks efficiency. Moreover, the positions of cameras are always fixed such that there might exist corners where the camera view cannot cover.

The use of a vision-based autonomous vehicle system is a good choice to solve the above problem. Because the vehicle only spends electric power or fuel, it can substitute for manpower to do security patrolling cheaply. By computer control, the vehicle can repeat identical steps, never feels tired, and makes errors. Hence, the vehicle is able to patrol the whole day and will not rest like human beings. Another advantage is that when the vehicle detects an unusual situation; the monitoring system can send warning messages to guards or hosts immediately through signal transmission equipment. So the works of monitoring will cost less time and money.

So the research goal of this study is to design an intelligent system for security patrolling by means of a vehicle. It is desired to design the system to be capable of indoor navigation and security checks, including object detection and door situation recognition.

1.2 Survey of Relative Studies

To achieve the mission of security patrolling in indoor environments, learning navigation paths and recording features of monitored objects is required before the vehicle can navigate automatically. Since scenes of indoor environments consisting of rooms, corridors, and objects are usually complicated, the vehicle must have the ability of computing the distance between objects. Lai [9] proposed a curve fitting technique and a modified interpolation technique to perform 2D-to-3D distance transformations. By this method, we can know the distances between the vehicle and the surroundings in real space through captured images. Li [10] proposed learning methods for dealing with complicated surroundings as well as strategies for autonomous vehicle navigation techniques. The user controls the vehicle to patrol and analyze the captured image in the learning process, and a navigation map is created dynamically. The vehicle can keep away from obstacles when it navigates automatically. Moreover, Chen [12] proposed two navigation modes and a fuzzy guidance technique. A navigation map is created by two kinds of learned data and the fuzzy guidance technique is applied to achieve obstacle avoidance. About security patrolling navigation, Liu [13] proposed a method for use in building corridors by multiple-camera vision and automatic vehicle navigation techniques.

In the learning process, the vehicle has to record the features of monitored objects. It is usually difficult to segment objects from complicated background in images. Kass, WitKin, and Terzopoulos [1] proposed a prototype of the *snake algorithm* to segment patterns in the image. However, there are some drawbacks in the snake algorithm. Some researches [2-6] tried to improve this algorithm by using more complicated techniques. After detecting an object from the image, the features of the object is recorded in the learning process. In order to represent the shapes of objects, Sussman [7] mentioned the technique of fitting an ellipse to a set of data points by using the least square fitting technique. And Pilu, Fitzgibbon, and Fisher [8] proposed the ellipse representation of object pixels by using the edge pixels of the object. There are methods of coordinate transformations in the space and vector operations techniques for use in Fraleigh and Beauregard [14]. There are some image processing techniques consulted in Gonzalez and Woods [15].

1.3 Overview of Proposed Approach

In this study, we try to design a vision-based vehicle system for security patrolling in indoor environments. In order to achieve the mission of security patrolling, to recognize surrounding objects and know the vehicle positions are necessary. The chief tools are the images captured by the camera and the odometer in the vehicle hardware system. An overall framework of the proposed system is illustrated in Figure 1.1.

The proposed vehicle system for security patrolling includes three major processes: *setup of the vehicle system*, *learning of paths and objects by user control,* and *vehicle navigation*.

Setup of the vehicle system is the basis of navigation. It includes *location mapping calibration* and *mechanic error correction*. After fixing the camera on the vehicle, a mapping calibration technique for using 2D image analysis is used to calculate the distance between the vehicle and an object. On the other hand, because the vehicle is a machine, it is unavoidable that there exist mechanic errors. Although the odometer provides the position of the vehicle, the real position of the vehicle is different from the value provided by the odometer due to the mechanic error. By the way of correcting the value provided by the odometer, navigation can be made more accurate. A mechanic error correction model is proposed to solve the problem. Moreover, the mechanic error causes the vehicle moving far away from original straight path. We propose a straight navigation technique to deal with this problem.

Before the vehicle navigates, how to detect objects from the image is essential. Because the computer can not identify objects likes people, *an improved snake algorithm* is proposed in this study to detect objects in captured images. Although some researches [2-6] improve the snake algorithm, these methods are complicated and slow. We design a faster method to meet the real time requirement. In order to recognize objects, we utilize three kinds of object features in this study; they are color, shape, and coordinates. We propose three methods to compute these three features, respectively.

In the learning process, the vehicle is controlled to move to desired places by using a user interface designed in this study. Some coordinates of the vehicles which are denoted as *nodes* are recorded along the learning path. The user may identify objects or doors to be monitored from the images when the vehicle moves to neighborhood of objects. The system will record the features of the objects which are illustrated in last paragraph automatically.

According to the node data and the position of the objects and doors, a navigation map is created for use of the vehicle navigating automatically. The vehicle moves along the path and detects the objects and doors one by one by the positions of them. As soon as the vehicle can not detect the monitored object, a warning message will be sent out from the system right away. When detecting the object, the system compares it with the learned data further. If the features are not correct, the same warning is announced. In the same manner, the vehicle moves to the neighborhood of the door to check whether the door is opened or not.

Due to the mechanic error, the vehicle might move far away from the original path gradually. We propose two methods to improve the stability of navigation. One is to use the coordinates of learned objects as an auxiliary tool to adjust the position and direction of the vehicle. The other is a line following technique. After completing the check of the objects and doors, one circle of security patrolling is finished.

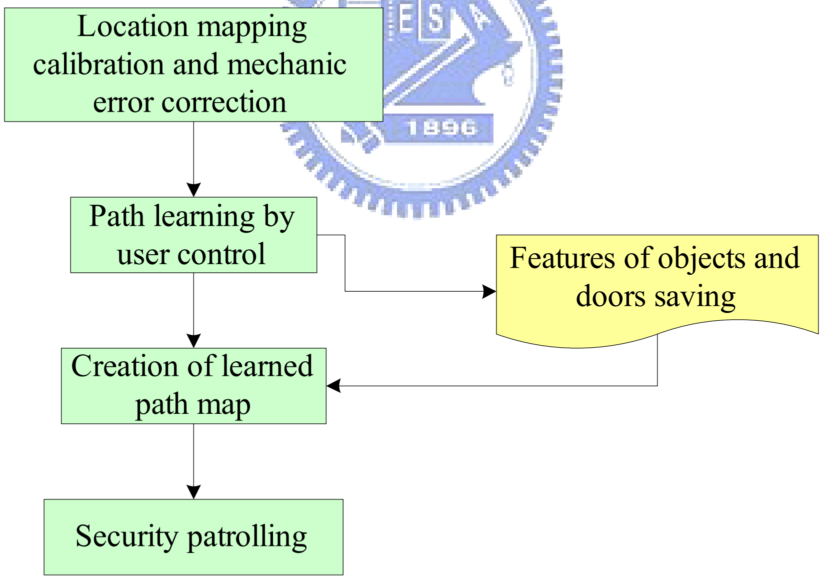

Location mapping calibration and mechanic error correction
Path learning by user control
Features of objects and doors saving
Creation of learned path map
Security patrolling

Figure 1.1 A flowchart of proposed system.

1.4 Contributions

Some major contributions of this study are listed as follows.

(1) A mechanic error correction method is designed.

(2) A technique for reducing mechanic error effect on straight navigation is proposed.

(3) A fast object detection method in indoor environments by using an *improved snake algorithm* is proposed.

(4) Three methods for object feature computing are proposed.

(5) A strategy for learning paths, objects, and doors is proposed.

(6) A method for path map creation by learned data is designed.

(7) A line following method which can reduce accumulative errors for navigation is proposed.

(8) Methods for object matching and door situation recognition are proposed.

(9) A method for correcting vehicle positions in the navigation session is proposed.

1.5 Thesis Organization

The remainder of this thesis is organized as follows. In Chapter 2, the system configuration of the vehicle and the principles of learning and navigation are described. In Chapter 3, the proposed methods for location mapping calibration and mechanic error correction are described. The proposed methods for object detection in the images and feature computing are described in Chapter4. The proposed learning strategy and path map creation method are described in Chapter 5. The proposed vehicle navigation method, the strategies for navigation accuracy maintenance, object recognition, and judgment of the door situation are described in Chapter 6. Some experimental results are given in Chapter 7. Finally, conclusions and some suggestions for future works are included in Chapter 8.

Chapter 2
System Configuration And Navigation Principles

2.1 Introduction

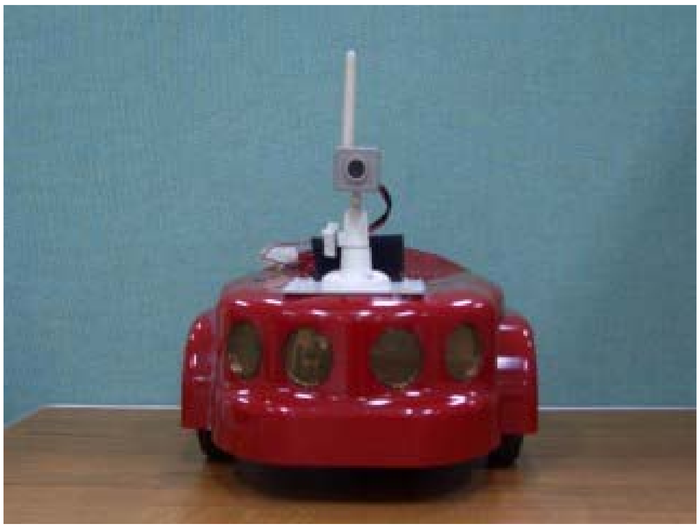
In order to achieve the goal of security patrolling by an autonomous vehicle, conditions around the vehicle are investigated. There are usually many obstacles and narrow paths in indoor environments. Hence, it is a wonderful choice that the size of a vehicle is small and its action is dexterous. The smaller the size of the vehicle, the larger range the vehicle can navigate. Moreover, if the size of the vehicle is small enough, it can monitor the space under tables, beds, cabinets, etc.

In this study, a small vehicle is used as a test bed. Beside the vehicle structure, a camera is installed on the vehicle. For a user to achieve the function of controlling the vehicle, some communication and control equipments are required. The entire hardware equipment and software used in this study are introduced in Section 2.2.

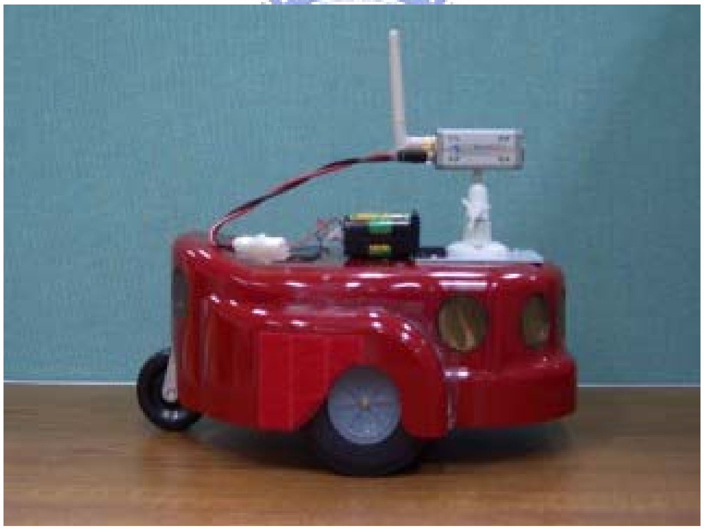
After introducing the equipment, the vehicle navigation principles and some detailed relevant processes are described in the following sections. In Section 2.3, we will describe the process of learning. Path and object data which are necessary for security patrolling are recorded in the learning process. In Section 2.4, we will describe the process of security patrolling in which the vehicle monitors concerned objects and checks whether a door is opened or not. As soon as an unusual situation is detected, a warning is issued.

2.2 System Configuration

The vehicle is shown in Figure 2.1.



(a)



(b)

Figure 2.1 The vehicle used in this study. (a) The front of the vehicle. (b) The flank of the vehicle.

Recently, wireless networking and communication techniques advance quickly. More and more communication products are made with wireless transmission. In this study, we use an Amigo robot, a mini-vehicle made by ActiveMedia Robotics

Technologies Inc., on which a camera is installed, as the testbed of our research. We utilize wireless transmission equipment to control the vehicle and get images captured by the camera. We will describe the hardware system in Section 2.2.1 and the software used in this study in Section 2.2.2.

2.2.1 Hardware Configuration

The entire system is illustrated in Figure 2.2. The first part is the vehicle system. The length, width, and height with the camera of the vehicle are 33cm, 28cm, and 21cm, respectively. There are two larger wheels and one auxiliary small wheel at the rear of the vehicle. The maximum speed of the vehicle is 75cm/sec and the maximum rotation speed is 300 degrees/sec. There are eight ultrasonic sensors, an odometer, and an embedded hardware system in this mobile vehicle. The ultrasonic sensors are not used in this study. The odometer provides the coordinates and direction of the vehicle in the navigation. The origin of coordinates is the starting position of the vehicle. There is a 12V battery in the vehicle to supply the power of the vehicle system.

The second part is the vision system. There is a wireless 2.4GHz camera which can transmit analog signals on the vehicle. A receiver and an imaging frame grabber are connected to the computer. The receiver receives analog signals from the camera and a digital image can be obtained by the imaging frame. The image grabbed in our experiments is of the resolution of 320×240 pixels.

A personal computer with Centrino 1.4GHz CPU, a 768MB DDR RAM, and a 5400 rpm 40GB HDD is used as a remote control and monitoring system; a kernel program can be executed on it to give commands to the vehicle through a wireless transmission system.

There is one wireless device in the vehicle and another in the PC. The commands of the remote system are transmitted to the wireless signal receiver by an access point that meets the IEEE 802.11b standard. By using the access point as a medium, the commands can be transmitted from the PC to the vehicle.

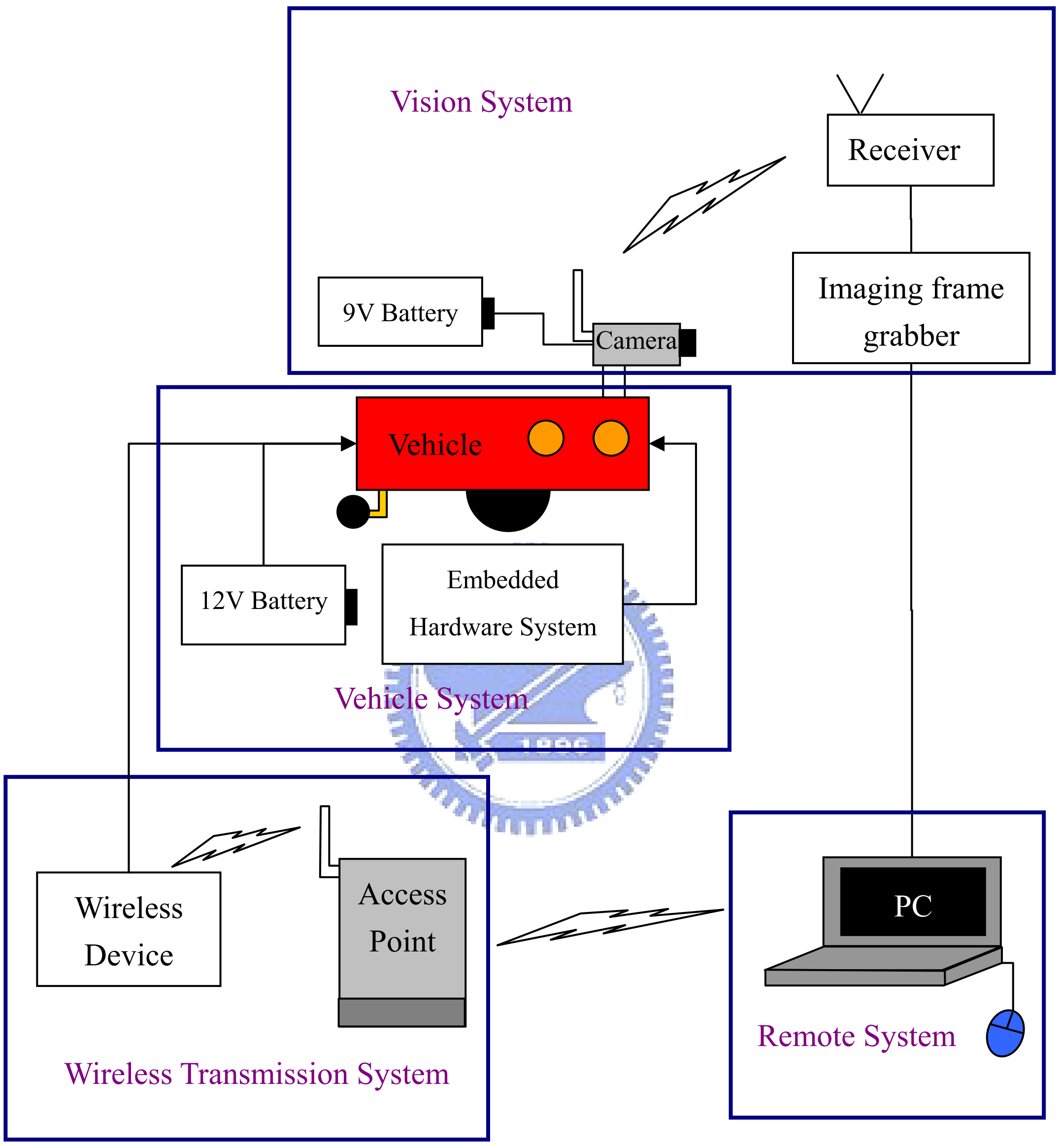


Figure 2.2 Structure of system.

2.2.2 Software Configuration

The ARIA is an object-oriented control applications-programming interface for

ActivMedia Robotics' line of intelligent mobile robots. The ARIA is written in the C++ language. We use it to control and retrieve the status of the vehicle. The operation system in the PC is Microsoft Windows XP and we use Borland C++ Builder as the development tool in our experiments.

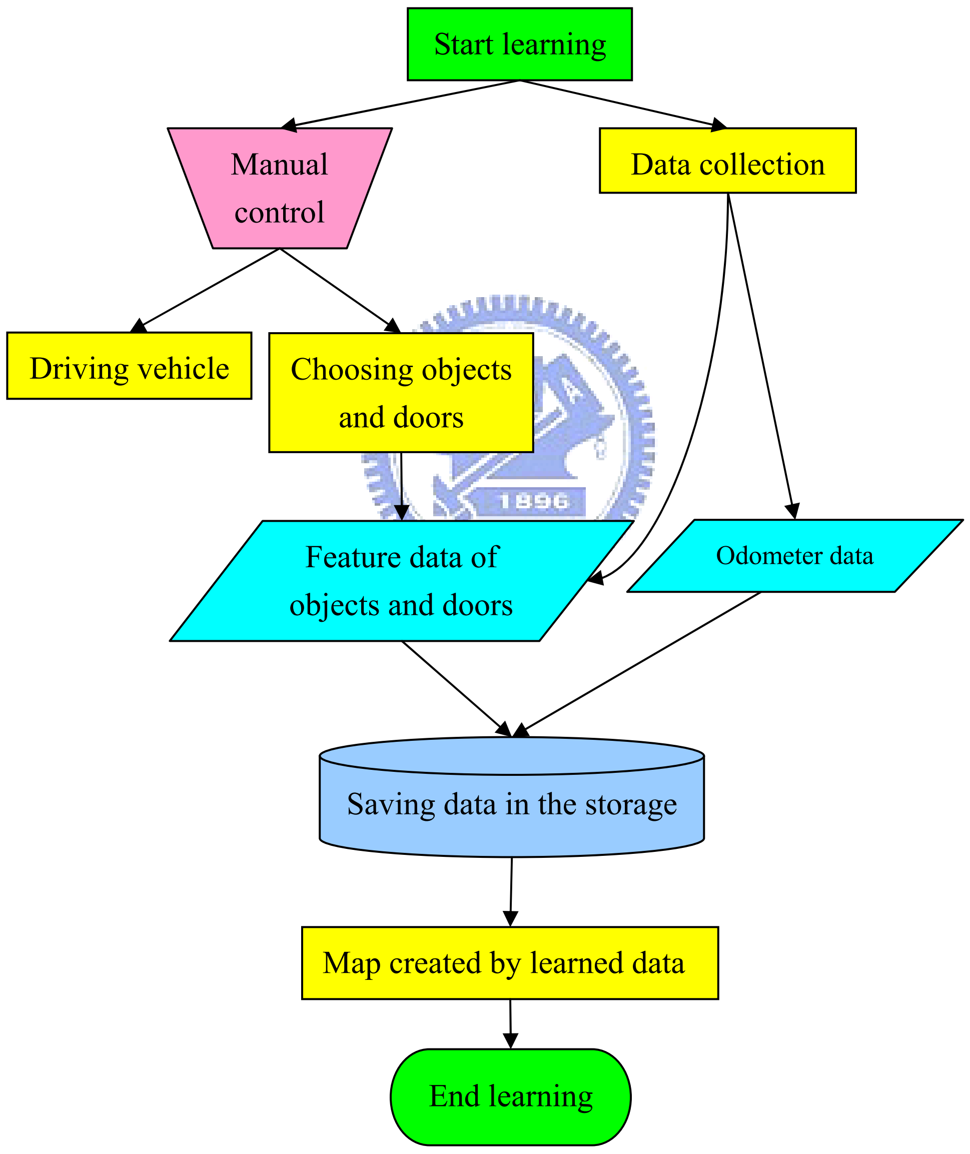
2.3 Learning Principle And Proposed Process of Learning

Before security patrolling, a learning process is necessary. Navigation path, object, and door data are recorded in advance. The entire learning process in this study is shown in Figure 2.3.

A user control interface is designed for use in controlling the vehicle and choosing monitored objects and doors. The user controls the vehicle to navigate in indoor environments and move to the front of objects for choosing monitored objects. The main recorded data include two categories, path and object data. The door is considered as an object in this study. As soon as the learning process ends, all data are stored in the storages of the computer such that the learning process is only executed once and data can be used repeatedly.

When the vehicle patrols in rooms and corridors, the computer records the path data provided by the odometer, and denote them as *nodes*. The monitored objects are selected in the image captured by the camera when the user controls the vehicle to patrol. The features of the objects are also computed automatically from the images. There are three kinds of object features are recorded in this study. They are color, shape, and coordinates respectively.

The process of door learning is similar to that of ordinary object learning. The computer can record more than one object when the vehicle is at a single position. By means of turning the vehicle at same position, we can learn neighboring objects of the vehicle. The user can choose objects continuously along the path until finishing a learning process.



Start learning
Manual control
Data collection
Driving vehicle
Choosing objects and doors
Feature data of objects and doors
Odometer data
Saving data in the storage
Map created by learned data
End learning

Figure 2.3 A flowchart of proposed learning process.

After finishing learning process, we can get a navigation map by way of combining the path data and object positions. The vehicle utilizes this map to monitor the doors and objects consecutively along the navigation path.

2.4 Vehicle Guidance Principle And Proposed Process

In the navigation process, the vehicle monitors the objects and doors consecutively by using the map created from learned data. The vehicle moves to a fixed position and stops to check the existence of the monitored objects. As soon as an unusual situation is occurred, a warning is issued. After checking the objects at a location, the vehicle moves to the next position where there are monitored objects or doors, and then continues checking the existence of the monitored objects or whether the door is opened. An illustration of the vehicle navigation process is shown in Figure 2.4.

In order to patrol along the learning path, the vehicle moves along each node in the map one by one. To move from a node to another, a line following technique is used to reduce navigation errors.

When the vehicle moves to a node where there is a nearby monitored object, the computer detects the object by using the position of the learned object. As soon as the computer detects an object from the image, its features are compared with those of the learned objects. If the features are incorrect, a warning is issued. Of course, when the computer can not detect the object, the same warning is also issued. Because the vehicle might navigate far away from the original path gradually, we use a vision-based technique to reduce accumulative errors for navigation. When the vehicle detects an object and consider it as a learned object, we use its position recorded in the learning process to correct the position of the vehicle for reducing accumulative errors.

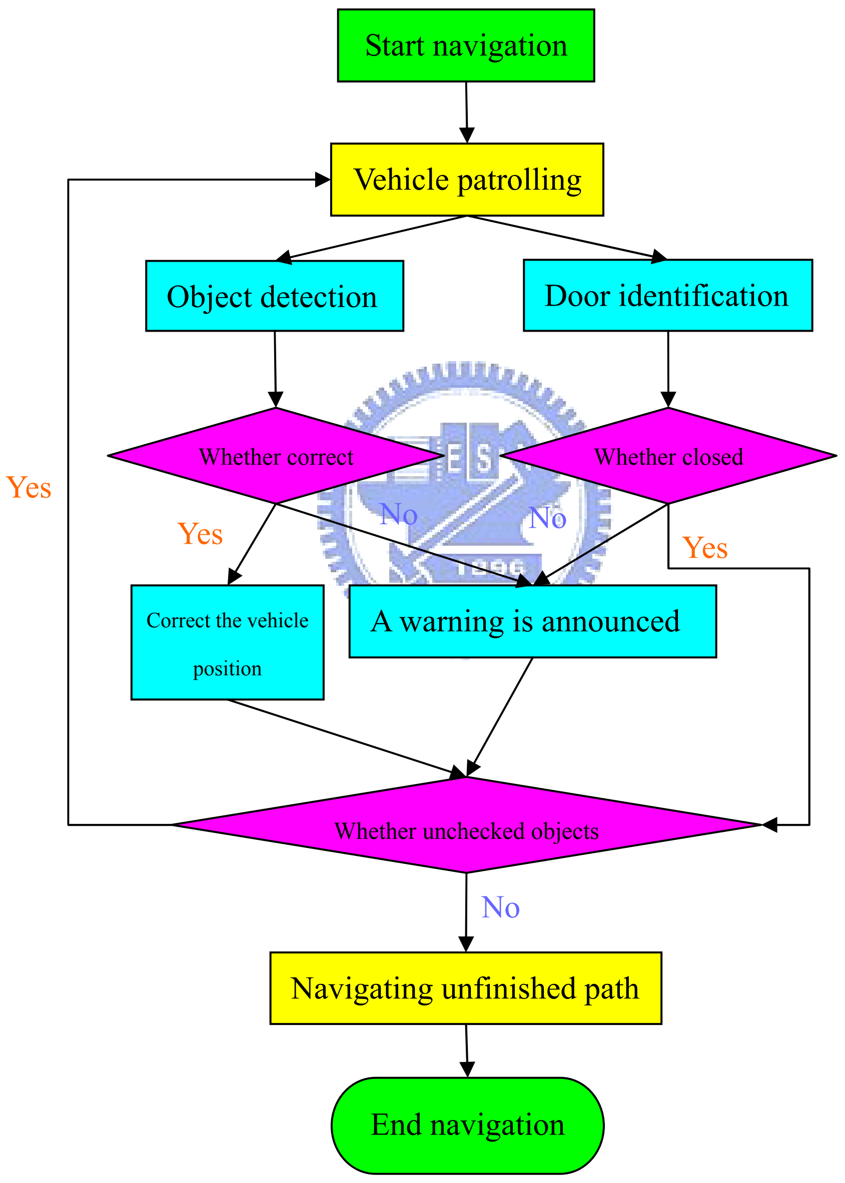


Figure 2.4 A flowchart of proposed navigation process

Although the door is considered an object, the method of recognizing the door situation is different from the method of object matching. After the vehicle moves to a suitable position recorded in the learning process, the computer checks the door situation to see if it is opened or closed. After the vehicle patrols the entire route and checks all objects and doors, a security patrolling process is finished.

Chapter 3
Location Mapping Calibration and Mechanic Error Correction

3.1 Introduction

When a vehicle navigates in indoor environments, the position and distance are important data for the navigation. By using these data, the vehicle can arrive at suitable positions to search monitored objects or doors and navigate along a learned path. In order to get position data and distance values, a camera and an odometer are used in this study, which are equipped on the vehicle.

At first, we utilize an image captured by the camera to obtain the relative position between an object and the vehicle. We use a 2D mapping method to achieve this goal. The detailed process is described in Section 3.2.

The odometer provides the positions of the vehicle in the environment. It records the coordinates and the direction angles of the vehicle when the vehicle navigates. Possible mechanic errors might cause the real positions of the vehicle to be unequal to the expected values provided by the odometer. Hence, in Section 3.3, we will propose a correction method to reduce the mechanic error so that the vehicle can navigate more stably.

Before describing the above-mentioned methods, we first introduce some coordinates system and the definition of the direction angle of the vehicle for use in this study. We introduce the coordinate systems in Section 3.1.1 and the definition of the direction angle of the vehicle in the space is described in Section 3.1.2. By using the direction angle, a transformation function between the coordinate systems is described in Section 3.1.3.

3.1.1 Coordinate Systems

Three coordinate systems are utilized in this study to describe the locations of the vehicle and objects. The coordinate systems are shown in Figure 3.1. The definitions of these systems are stated in the following.

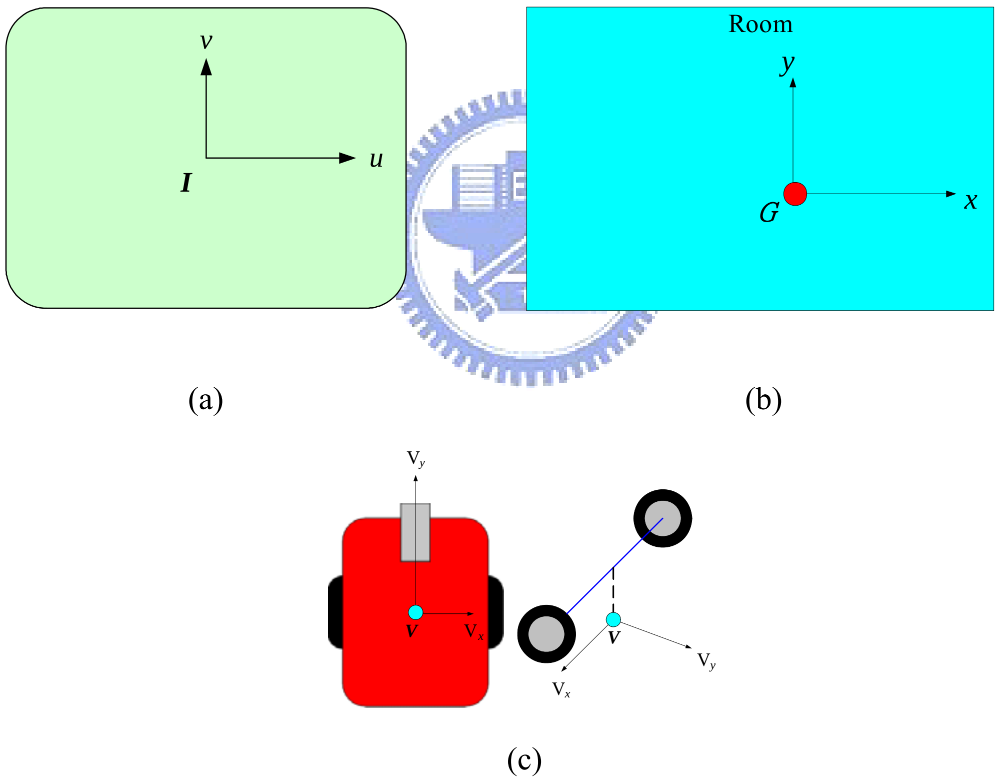


Figure 3.1 Three coordinate systems in this study. (a) ICS (b) GCS (c) VCS.

(1) The image coordinate system (ICS): denoted as u–v. The u–v plane is coincident with the image plane and the origin ***I*** of the ICS is placed at the center of the

image plane.

(2) The global coordinate system (GCS): denoted as *x*-*y*. The *x*-axis and the *y*-axis are defined to lie to on the ground, and the origin ***G*** of the global coordinate system is a pre-defined point on the ground. In this study, we define ***G*** as the starting position of the navigation.

(3) The vehicle coordinate system (VCS): denoted as V_x-V_y. The V_x-V_y plane is coincident with the ground. The V_x-axis is parallel to two wheels of the vehicle. The V_y-axis is parallel to the body of the vehicle. And the origin ***V*** is placed at the middle of the line segment that connects the two contact points of the two driving wheels with the ground.

3.1.2 The Direction Angle

The direction angle of the vehicle is defined in the global coordinate system, the *x*-*y* plane. The angle denoted as θ represents the rotation degree of the vehicle in the global system and plays a very important role in the coordinate transformations.

θ is the angle between the positive direction of the *x*-axis and the front of the vehicle. The direction angle θ is set to be zero at the beginning of navigation. The range of θ is between 0 and π if θ is in the first and second quadrants, as illustrated in Figure 3.2(a) and (b). It is between 0 and $-\pi$ if θ is in the third and forth quadrants, as illustrated in Figure 3.2(c) and (d).

3.1.3 Coordinate Transformation

By using the direction angle and the coordinates of the vehicle in the GCS, we can describe the coordinate transformation between the vehicle coordinate system and the global coordinate system by the following equations.

$$x = \mathrm{V}_x \times \cos\theta - \mathrm{V}_y \times \sin\theta + x_p$$

$$y = \mathrm{V}_x \times \sin\theta + \mathrm{V}_y \times \cos\theta + y_p$$

where x_p and y_p are the coordinates of the vehicle in the GCS. It is shown in Figure 3.3.

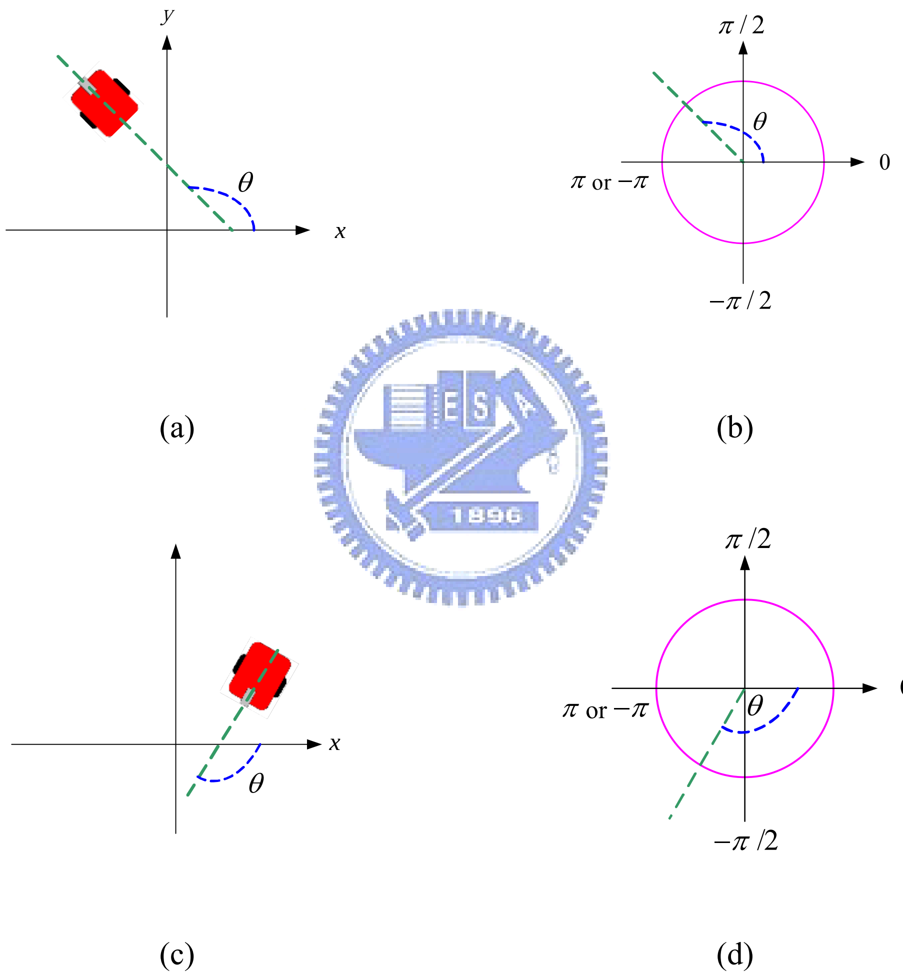


Figure 3.2 Definition of the direction angle (a)(b) θ is positive. (c)(d) θ is negative.

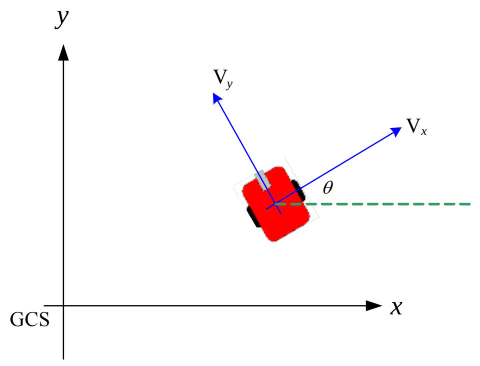


Figure 3.3 The coordinate transformation between the GCS and VCS.

3.2 Location mapping Calibration

The camera is the only sensor of the vehicle to gather the features of the environment, and the techniques of learning and guidance are based on visual perception. Location mapping calibration and image analysis techniques are indispensable in this study. A real location data acquisition method by Location mapping calibration and image mapping is proposed to obtain the relative positions of the vehicle and the surrounding environment precisely.

We use a point set $\boldsymbol{P} = \{P_{00}, P_{01}, \ldots, P_{mn}\}$ whose coordinates are known in advance in the VCS attached on the floor, and their corresponding point set $\boldsymbol{V} = \{V_{00}, V_{01}, \ldots, V_{mn}\}$ appearing in the image, to compute the VCS coordinates of a set of pixels. The detailed process is described in the following algorithms.

Algorithm 3.1. *Real location data acquisition by image taking and mapping.*

Input: An image I, as shown in Figure 3.5(a).

Output: A point set $\boldsymbol{V} = \{V_{00}, V_{01}, \ldots, V_{mn}\}$ and another point set $\boldsymbol{P} = \{P_{00}, P_{01}, \ldots, P_{mn}\}$

Steps:

Step 1. Attach some straight lines on the floor, as shown in Figure 3.4. Set the length between every two vertical lines to be 30 cm and the length between every two horizontal lines to be 30 cm, too, where every rectangle is a 30×30 square. There exists a distance D_{iv} equal to 80cm in the front of the vehicle, which the camera can not view.

Step 2. For those tessellated regions shown in the image which are not surrounded by the lines, extend the lines such that every quadrilateral is complete. Mark red points on the intersections of the lines, as shown in Figure 3.5(b).

Step 3. Record the coordinates of each red point $P_{ij}(u_{ij}, v_{ij})$ in the ICS and group all such points into the set $\boldsymbol{P}$.

Step 4. Measure relative coordinates $V_{ij}(x_{ij}, y_{ij})$ manually in the VCS of the point P_{ij} and group such points into the set $\boldsymbol{V}$.

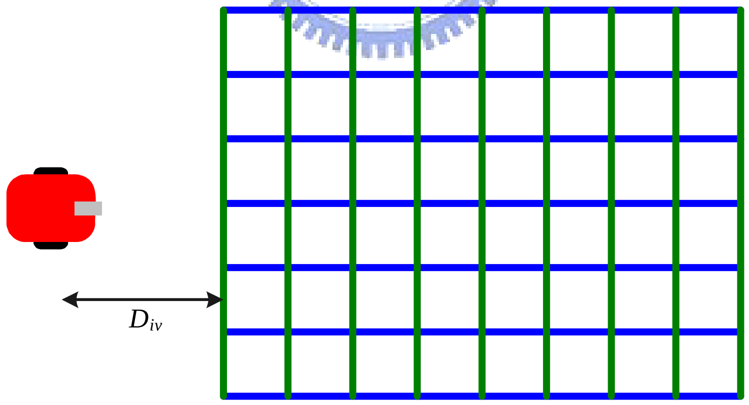
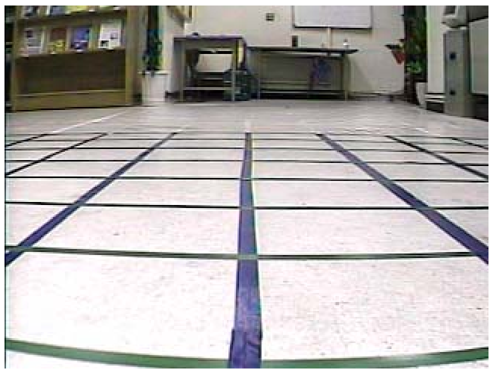
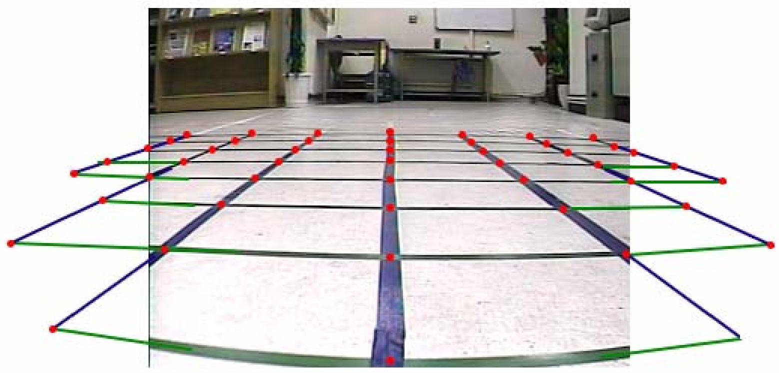


Figure 3.4 An illustrated of attaching the lines on the floor.



(a)



(b)

Figure 3.5 A method of finding image coordinates of tessellated points in the grabbed image. (a) A grabbed image with tessellated points. (b) The tessellated points marked by red points.

We now have known the VCS coordinates of red points. As for the VCS coordinates of the other pixels, we use an interpolation method to obtain them, as described in the following algorithm.

Algorithm 3.2. *Interpolation for computing lateral distances.*

Input: A point $I(u,v)$ in the ICS, a point set $\boldsymbol{P}$, and another point set $\boldsymbol{V}$.

Output: The coordinates of I in the VCS.

Steps:

Step 1. Compute the varibales a and b of the line equation $y = ax + b$ for lines L_0, L_1, L_2, and L_3 by using ICS coordinates of four endpoints, P_{ij}, $P_{(i+1)j}$, $P_{i(j+1)}$, and

$P_{(i+1)(j+1)}$ in the following ways, as illustrated in Figure 3.6.

$$a = \frac{v_2 - v_1}{u_2 - u_1}. \tag{3.1}$$

$$b = \frac{v_1 \times u_2 - v_2 \times u_1}{u_2 - u_1}. \tag{3.2}$$

Step 2. Decide whether the point I is in the region surrounded by four endpoints, P_{ij}, $P_{(i+1)j}$, $P_{i(j+1)}$, $P_{(i+1)(j+1)}$ by substituting (u, v) for (x, y) of the line equation in the following ways.

$$(a_0 \cdot u + b_0 - v) \cdot (a_2 \cdot u + b_2 - v) \leq 0. \tag{3.3}$$

$$(a_1 \cdot u + b_1 - v) \cdot (a_3 \cdot u + b_3 - v) \leq 0. \tag{3.4}$$

Step 3. If the inequalities (3.3) and (3.4) are satisfied, the point I is in this region. Else, repeat Step 2 to check the next region.

Step 4. By using a horizontal line equation $y = v$ which passes the point I, we obtain two intersections $H(u_h, v_h)$ and $K(u_k, v_k)$ as shown in Figure 3.6(a).

Step 5. By using a vertical line equation $x = u$ which passes the point I, we obtain two intersections $S(u_s, v_s)$ and $T(u_t, v_t)$.

Step 6. Use an interpolation method to obtain the VCS coordinates (x_I, y_I) of I by the following equations:

$$x_I = \begin{cases} 30 \times (j-3) + 30 \times (1 - \frac{u_h - u}{u_h - u_k}), & \text{if } j \leq 3; \\ 30 \times (j-3) + 30 \times (\frac{u - u_k}{u_h - u_k}), & \text{if } j > 3. \end{cases} \tag{3.5}$$

$$y_I = D_{iv} + 30 \times |i - 7| + 30 \times (\frac{v_s - v}{v_s - v_t}). \tag{3.6}$$

Where i is serial number of the horizontal lines and j is serial number of the vertical lines.

Although the image is distorted slightly, we consider every line as a straight line. Every region is not rectangular as appearing in the image; therefore, we use the interpolation method to compute x_I and y_I respectively.

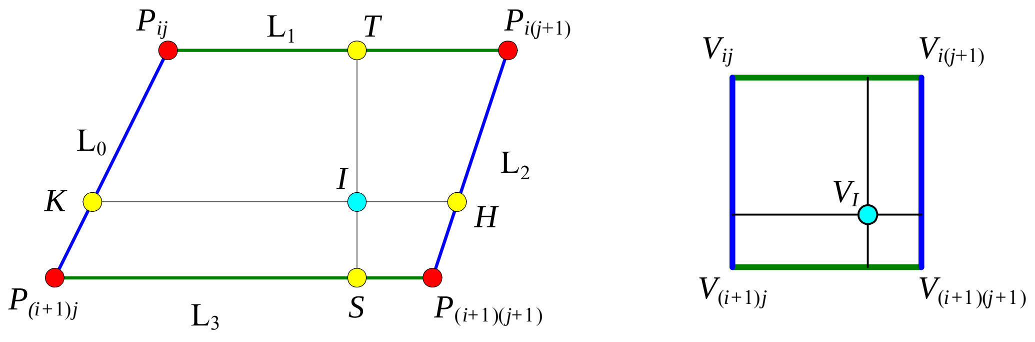


Figure 3.6 An illustration of the interpolation method. (a) A region contains the point *I* in the ICS. (b) The projection of the region in (a) onto a floor region.

3.3 Proposed Mechanical Error Correction Method

There exist mechanic errors when the vehicle navigates back and forth, no matter how people control the vehicle or how the vehicle navigates automatically.

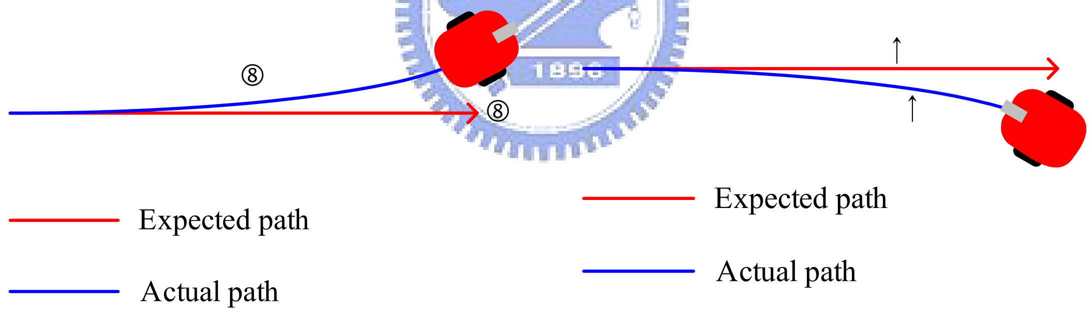
Whenever the control instruction is moving ahead or backing off, the vehicle will move away from the original straight path gradually and the coordinates and the direction angle of the vehicle provided by the odometer will be different from the actual position and angle of the vehicle.

A method is proposed for solving the above problem. A mathematical model is set up for correcting mechanic errors. In Section 3.3.1, we will describe the concept of the method briefly. Utilizing this mathematical model; we can adjust the coordinates and the direction angle of the vehicle provided by the odometer, as described in Section 3.3.2. Finally, we hope the vehicle can move along straight path automatically; hence, we propose a method to solve this problem, as described in Section 3.3.3.

Because there exists a similar problem when the vehicle backs off, we use the same concept to correct navigation errors. It is described in Section 3.3.4.

3.3.1 Brief Description of Proposed Error Correction Model

Due to mechanic errors, a straight path of the vehicle navigation will become a curve path. A detailed situation is illustrated in Figure 3.7. In Figure 3.7(a), the vehicle starts moving and the expected path is a straight line in advance. After moving a distance, the vehicle moves along path ② instead of the expected path ①. The same situation occurs when the vehicle backs off. In Figure 3.7(b), the vehicle backs along path ④ instead of path ③.



(a) (b)

Figure 3.7 Illustrations of navigation path deviation. (a) Moving ahead. (b) Backing off.

Besides the last problem, the coordinates and direction angle provided by the odometer do not show the actual position of the vehicle in the GCS due to the mechanic error. As shown in Figure 3.8, the vehicle moves from the starting position P_0 and arrives at position P_c finally. But the coordinates provided by the odometer are the point P_1 instead of P_c and the direction angle is the original angle of the starting position. Actually, because the vehicle moves along the curve path, the vehicle has rotated for an angle θ.

Because the path of the vehicle is a curve, we compute a curve equation to represent the path. We measure the deviation distances in advance to build a mathematical equation as a correction model such that the vehicle can modify navigation errors dynamically. The concept is described as follows.

At first, we get some the vertical deviation distance values measured manually when the vehicle navigates forward a distance. According to the values, a second-order equation $y = ax^2 + bx + c$ is obtained by a curve fitting technique. Among the equation, x is the distance between the starting position P_0 and the position P_1, and y is the deviation distance from the expected path. Moreover, In Section 3.3.2, we will describe how to get the coefficients a, b, and c of the equation. This correction equation always accompanies the vehicle whether the vehicle is controlled by a user or navigates automatically.

To correct the direction angle of the vehicle, we use the first derivative equation to compute the rotation angle. It is like a concept in physics mentioning that when the object is in a curvilinear motion, by using a tangent line, the direction of the vehicle can be computed. Based on such a principle, we can find a tangent line L_1 as shown in Figure 3.8. We compute the slope of line L_1 by using the first derivative of $y = ax^2 + bx + c$. The process of modifying the values of the vehicle position is shown in Figure 3.9 and a brief computing process is described as follows.

A distance x_d is gathered by computing the length between P_0 and P_1. Substituting x_d into the equation $y = ax^2 + bx + c$, we can get the deviation value y_d. By using x_d and the slope $\frac{dy}{dx} = 2ax + b$, we can get an deviation angle θ the vehicle had turned in the GCS. Hence, by using coordinate transformation techniques, we can use x_d, y_d and θ to get actual position P_c. The details are described in Section 3.3.3.

Every time the vehicle starts moving, the above method is carried out. As long as the vehicle stops moving, the correct values of x_d, y_d and θ must be computed again in the next moving circle.

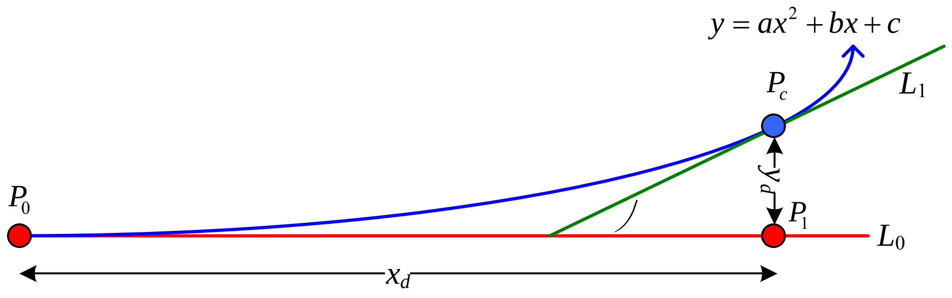
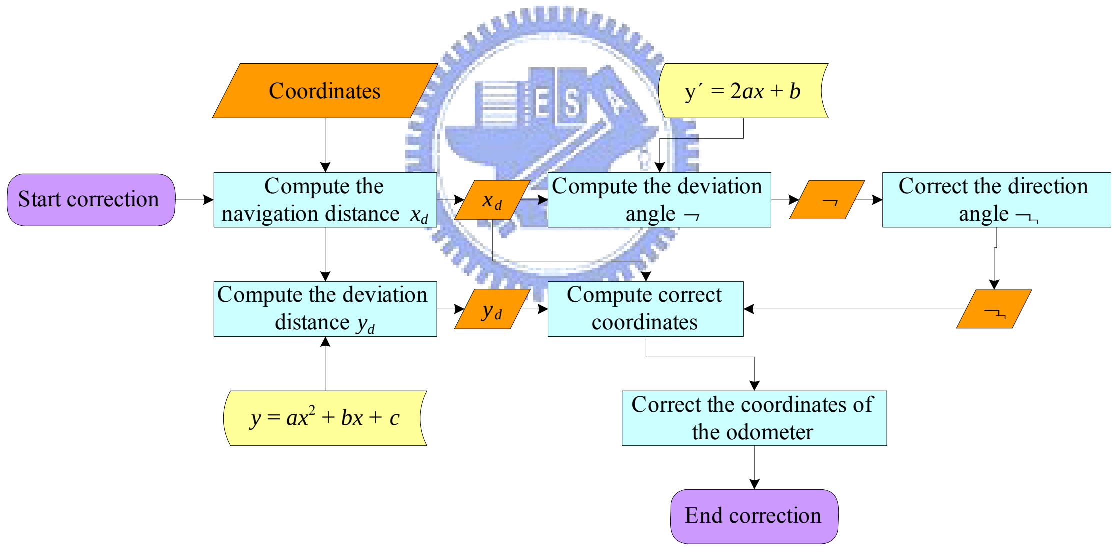


Figure 3.8 An illustration the correction model.



Coordinates
y′ = 2ax + b
Start correction
Compute the navigation distance x_d
x_d
Compute the deviation angle ¬
¬
Correct the direction angle ¬
Compute the deviation distance y_d
y_d
Compute correct coordinates
¬
$y = ax^2 + bx + c$
Correct the coordinates of the odometer
End correction

Figure 3.9 Flowchart of correction process.

3.3.2 Curve Fitting for Navigation Path

In Section 3.3.1, we have described the concept of the correction model. According to the discussion in the last section, to build the second-order curve equation is the most basic requirement. All correction computation is based on this curve equation, so we describe how we compute the coefficients of the equation first in this section. A detailed process is described in the following algorithm.

Algorithm 3.3. *Curve fitting for navigation path.*

Input: A navigation path.

Output: An equation $y = ax^2 + bx + c$.

Steps:

Step 1. Measure a deviation distance y_{dv} every 0.5m when the vehicle navigates, as shown in Figure 3.10.

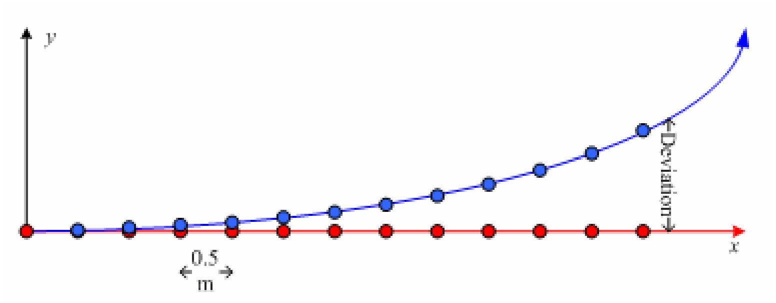


Figure 3.10 An illustration of recording the coordinates.

Step 2. Record the deviation distance y_{dv} when the vehicle navigates a distance x, and denote the data as $P_{1j}(x_j, y_{1,j})$ where x_j is navigation distance computed between the starting position and the new position provided by the odometer and $y_{1,j}$ is a deviation distance. We record thirteen data when the vehicle moves a 6m distance. Finally, we group the data into a point set $P_1 = \{P_{1,0}, P_{1,1}, \ldots, P_{1,12}\}$.

Step 3. Repeat Step 1 two times and group the data into two sets P_2 and P_3.

Step 4. There are three deviation values at each distance x_j. Compute a mean set $\overline{P}$ in the following way;

$$\overline{P} = \frac{1}{3}\left\{\sum_{i=1}^{3} P_{i,0}, \sum_{i=1}^{3} P_{i,1}, \sum_{i=1}^{3} P_{i,2}, \ldots, \sum_{i=1}^{3} P_{i,12}\right\}. \tag{3.7}$$

where the data in the set $\overline{P}$ are denoted as $\overline{P}_j(x_j, \overline{y}_j)$.

Step 5. By using the least square method, we compute the coefficients of the optimum curve $y = ax^2 + bx + c$ according to x_j and $\overline{y}_j$, as shown in Figure 3.11.

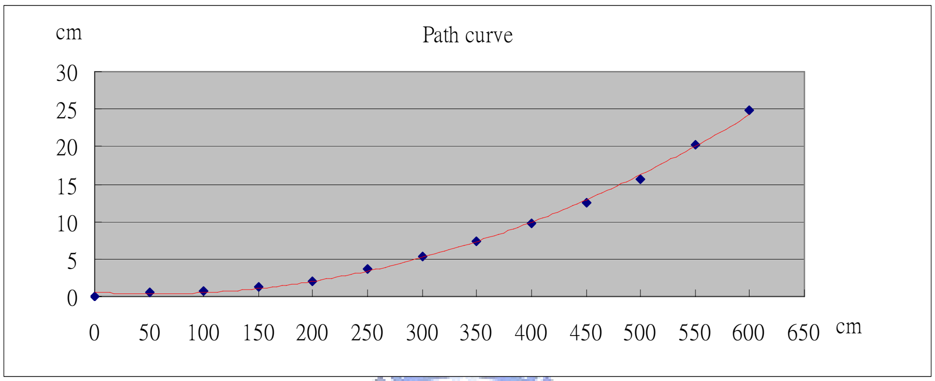


Figure 3.11 A figure of the curve.

Because the deviations are slightly different in each navigation path, we measure each deviation three times and use their means to compute the curve equation. In this study, *a*, *b*, and *c* are computed to be *a* = 0.00008, *b* = –0.0089, and *c* = 0.5828. The distance unit we use is cm.

3.3.3 Coordinates And Angle Correction

After computing the path curve equation, the main goal is to adjust the odometer values of a navigating vehicle. Utilizing the curve equation; we can conjecture that the vehicle has shifted how much distance from a straight line when it is still to move a certain distance. A detailed description of the adjustment is described as follows.

In Figure 3.12, the vehicle starts moving from point $P_0(x_0, y_0)$ toward the goal point $P_1(x_1, y_1)$ in the GCS. The direction of the vehicle in the starting position is *θ*.

Due to mechanic errors, the final position the vehicle arrives at is point $P_1'(x_1', y_1')$. Although the position of the vehicle is P_1' and the angle has become $\theta + \alpha$, the odometer shows that the position and angle are P_1 and θ. The point P_1' and angle α is the most important data we want to modify the error of the odometer. We compute the distance D as follows:

$$D = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}\,. \tag{3.8}$$

Utilizing D and the curve equation $y = ax^2 + bx + c$, the deviation distance d_y is derived to be as follows:

$$d_y = aD^2 + bD + c\,. \tag{3.9}$$

Then we use the coordinate transformation technique to get the point P_1' as follows:

$$x_1' = D \cdot cos\theta - d_y \cdot sin\theta + x_0 \tag{3.10}$$

$$y_1' = D \cdot sin\theta + d_y \cdot cos\theta + y_0 \tag{3.11}$$

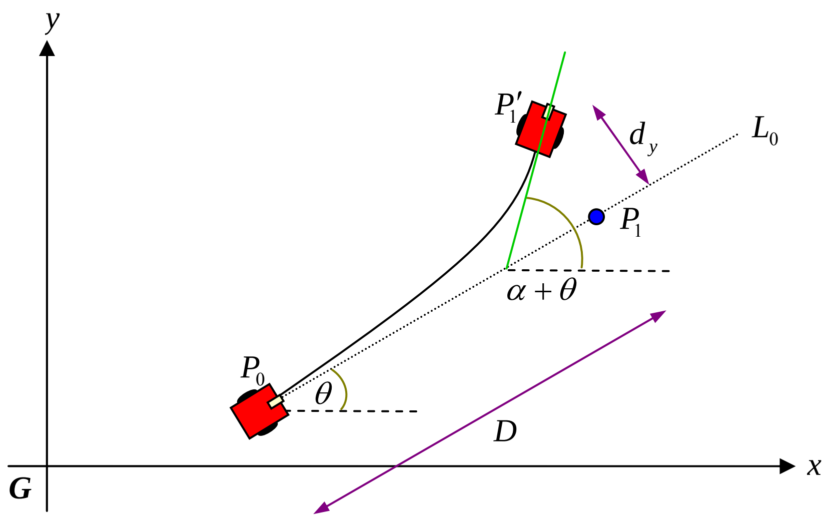


Figure 3.12 An illustration of correction detailed.

In the following, we want to get the rotation angle α, as shown in Figure 3.13. Utilizing the tangent slope $y' = 2ax + b$, we can get vector $\vec{b} = [1, 2aD + b]$ in the error correction model. Using the unit vector $\vec{a} = [1, 0]$, we can get the angle α by the following formula:

$$\alpha = \cos^{-1}\left(\frac{\vec{a} \bullet \vec{b}}{|\vec{a}||\vec{b}|}\right). \tag{3.12}$$

Finally, we correct the coordinates of the vehicle to be P_1' and correct the angle to be $\theta + \alpha$.

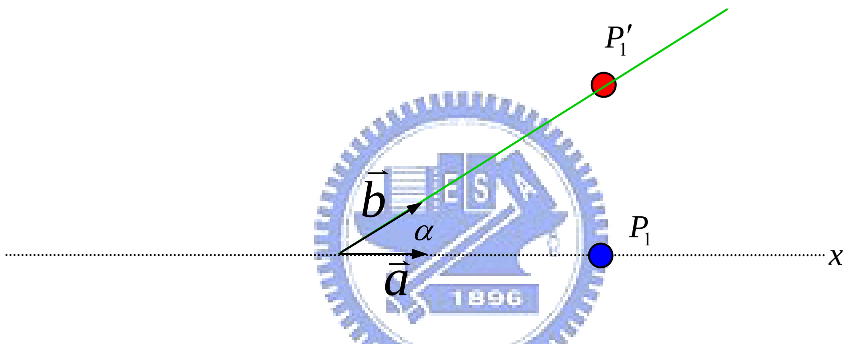


Figure 3.13 An illustration of computing correct angle.

3.3.4 Straight Navigation Technique

Because the vehicle always navigates along a curve path, in this section, we propose a technique to control the vehicle to navigate along a straight line automatically. The most important concept behind our technique is that we control the vehicle to return to the original straight path, as illustrated in Figure 3.14. The path of the vehicle is shown as a red line in Figure 3.14.

Assume that the vehicle starts from point P_0, and plans to navigate along path L_0. According to the curve equation $y = ax^2 + bx + c$, we can get a distance d_y when the vehicle has moved a distance D_t. We use these data to compute a moving distance and a rotation angle such that the vehicle can turn toward path L_0 and go back to this path. Although we can correct the values of the odometer at any time, it is impossible to do this in the practical operation. Hence, we correct the mechanic error when the vehicle stops moving.

When the vehicle goes to point P_1', the vehicle stops and corrects the values of the odometer. To utilize the tangent line, we compute the rotational angle α. Point $P_2'(x_2', y_2')$ which is on the path L_0 is the goal for the vehicle to go back to path L_0. We compute the distance $\left|\overline{P_1'P_2'}\right|$ by the following equation:

$$\left|\overline{P_1'P_2'}\right| = \frac{d_y}{sin\alpha}. \tag{3.13}$$

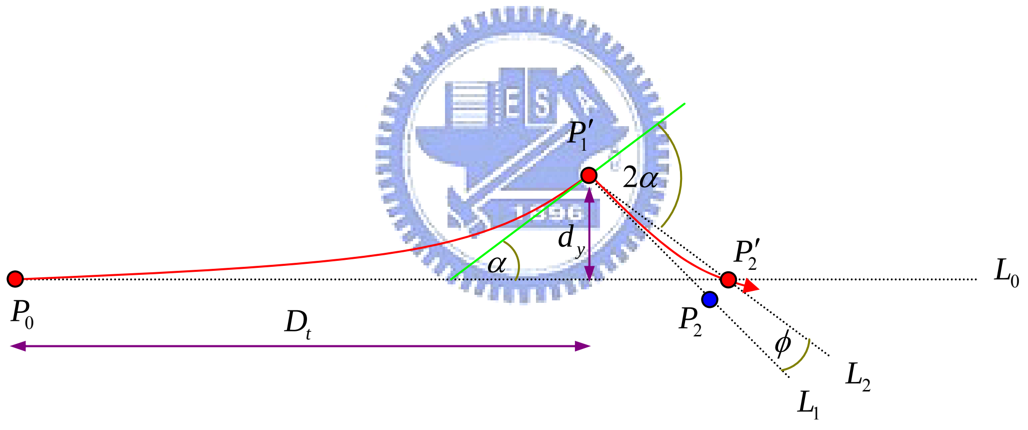


Figure 3.14 An illustration of the correction path

The vehicle should turn 2α toward path L_0 and go through the distance $\left|\overline{P_1'P_2'}\right|$ back to L_0, as the path L_2 shown in Figure 3.3.4.1. Unfortunately, the vehicle does not arrive at point P_2' because of the curve path. If the vehicle goes toward point P_2', the path will be like the blue line shown in Figure 3.15 and the terminal will be P_t finally, as shown in Figure 3.15. Hence, the vehicle has to turn an angle of ϕ such that it can navigate along the curve and arrive at P_2' finally. As shown in Figure 3.15, the planed path is L_1 and the expected goal is $P_2(x_2, y_2)$, such that the vehicle can arrive at P_2' finally.

From the triangle $\triangle P_1'P_2'P_t$, we have known the length $\left|\overline{P_1'P_2'}\right|$, hence we can compute the length $\left|\overline{P_tP_2'}\right|$ by the following curve equation:

$$\left|\overline{P_tP_2'}\right| = a\left|\overline{P_1'P_2'}\right|^2 + b\left|\overline{P_1'P_2'}\right| + c. \tag{3.14}$$

And the angle $\angle P_tP_1'P_2'$ can be computed as follows:

$$\angle P_tP_1'P_2' = \tan^{-1}\left(\frac{\left|\overline{P_tP_2'}\right|}{\left|\overline{P_1'P_2'}\right|}\right). \tag{3.15}$$

Let $\angle P_tP_1'P_2'$ be equal to $\angle P_2'P_1'P_2$ which is denoted as ϕ, then we can get $\triangle P_tP_1'P_2' \approx \triangle P_2'P_1'P_2$. Hence, we can get the following relation:

$$\frac{\left|\overline{P_1'P_t}\right|}{\left|\overline{P_1'P_2'}\right|} = \frac{\left|\overline{P_1'P_2'}\right|}{\left|\overline{P_1'P_2}\right|}. \tag{3.16}$$

If the angle $\angle P_tP_1'P_2'$ is small enough, then we can get $\left|\overline{P_1'P_t}\right| \approx \left|\overline{P_1'P_2'}\right|$. Hence, we can get $\left|\overline{P_1'P_2'}\right| \approx \left|\overline{P_1'P_2}\right|$. Therefore, we get a navigation distance by computing the distance between P_1' and P_2'.

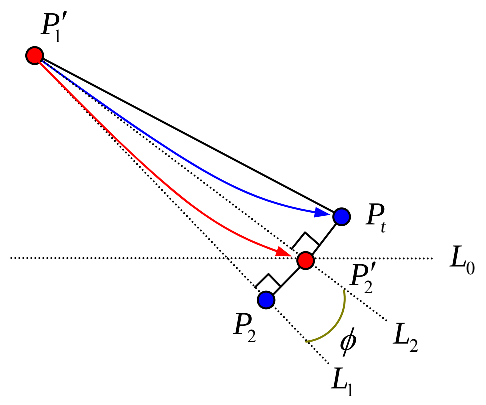


Figure 3.15 Computation of corrected path

Summarizing the above description, we describe the correction process as follows:

Step 1. Start the vehicle from point P_0. According to value of the odometer, the vehicle stops at the point P_1' after going through a distance of D_t.

Step 2. Correct the coordinates and the angle of the vehicle.

Step 3. Compute the turn angle ϕ by using Equation (3.15) and navigate a distance $\left|\overline{P_1'P_2'}\right|$.

Step 4. Turn the vehicle for the angle of $2\alpha + \phi$ toward the straight path L_0 and let it navigate for a distance of $\left|\overline{P_1'P_2'}\right|$ to arrive at point P_2' and stop.

Step 5. Correct the coordinates and the angle of the vehicle again.

Step 6. Turn the vehicle such that the direction of the vehicle is the same as the direction in the beginning position P_0 at the point P_0 in the GCS.

The above process is just only one correction circle among the navigation. If the path is longer, the vehicle has to repeat the above process many times. The final path which is the red line is shown in Figure 3.3.4.3. It is periodic and looks like a trigonometric function of mathematics.

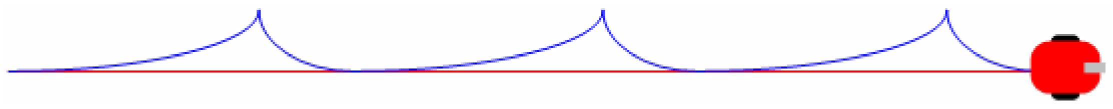


Figure 3.16 An Illustration of corrected navigation path

3.3.5 Correction of Backing off

Due to the mechanic error the vehicle also backs off along a curve path. The navigation path when the vehicle backs off is shown in Figure 3.17. We use the same method as mentioned in last sections to correct the errors in this section.

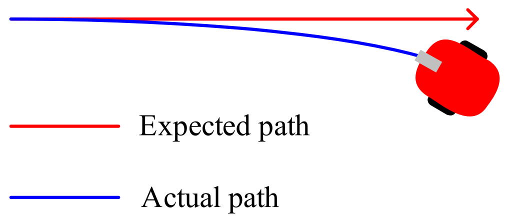


Figure 3.17 An illustration of Backing path.

As first, using curve fitting to compute the curve equation y = $ax^2 + bx$ + c, as shown in Figure 3.18. The equation is different from the equation computed when the vehicle moves ahead. The distance between the starting position and the now position is record as a negative value. The coefficients of *a*, *b*, and *c* are computed to be a = 0.0001, b = 0.002, and c = 0.0216, respectively, in this study.

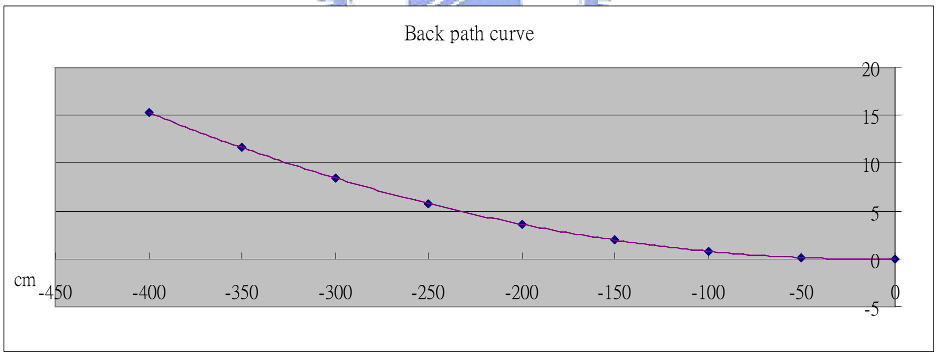


Figure 3.18 The path curve of backing off.

We still use the same method to correct the coordinates and direction angle of the vehicle. The correction equations are different from those of (3.10) and (3.11). The equation of computing the actual position (x_1', y_1') of the vehicle become as:

$$x_1' = -D \cdot cos\theta - d_y \cdot sin\theta + x_0\,; \tag{3.17}$$

$$y_1' = -D \cdot sin\theta + d_y \cdot cos\theta + y_0\,. \tag{3.18}$$

Finally, the process of backing along the straight line is the same with the process of moving ahead.

Chapter 4
A Method for Detection of Monitored Objects by Image Analysis

4.1 Introduction

We use a vision-based vehicle to monitor concerned objects and doors in this study. How to recognize objects and doors are an important problem in this study. Hence, before describing the vehicle navigation process, we will first describe the image processing and pattern recognition techniques we use for solving this problem in this chapter.

There are two main processes to recognize an object in an image. One is to detect the object region, and the other is to compute the feature data of the object. To detect an object, we propose the use of an *improved snake algorithm* to accomplish the task. The details of the method are described in Section 4.2.

After detecting objects, we have to compute useful feature data and save them for use in the learning process such that we can use them to monitor the objects in the security patrolling process. We will describe how to compute the feature data in Section 4.3.

4.2 Process of Monitored Objects Detection

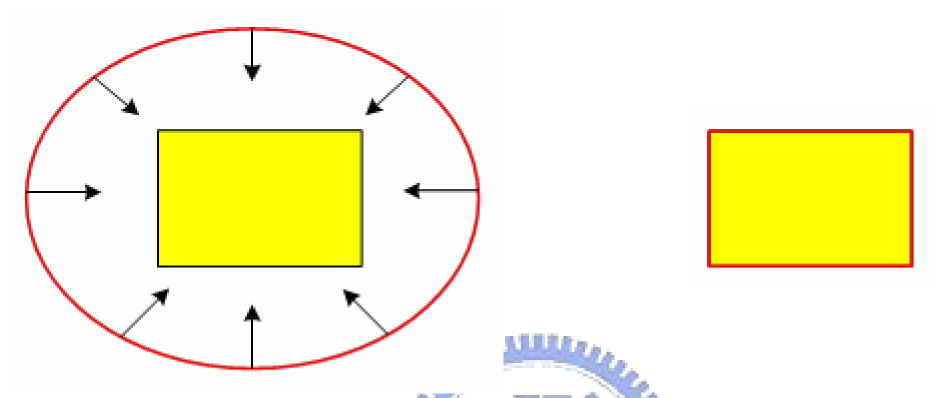
In this section, we will describe how to detect an object region in an image. We tried to use a famous object detection technique, called "*Snake Algorithm*" [1] in this study. In Section 4.2.1, we will briefly describe the basic concept of the snake algorithm. But the snake algorithm has many weaknesses when it is used in the complicated environments; hence, we will propose an improved version of the method to detect objects in indoor environments in Section 4.2.2.

Some complicated environment backgrounds could influence the result of object detection; we will describe a feasible method proposed in this study in Section 4.2.4. In Section 4.2.3, we will describe the details of the proposed improved snake algorithm.

4.2.1 Brief Description of Snake Algorithm

The performance of the snake algorithm is like the behavior of an elastic band. Imagine that there is an elastic band in the image and an object exists in it. By repeating continuous operations, the elastic band becomes narrow and approaches the object edge. Finally, the elastic band will enclose the edge of the object and stop computing, as illustrated in Figure 4.1. We now describe the principle of the snake algorithm below.

The snake algorithm uses a set of control points, effectively connected by straight lines. Each control point has a position, specified by coordinates (x, y) in the ICS, and the snake is entirely specified by the number and coordinates of its control points. By means of adjusting the positions of the control points and computing the energy of the snake at every moment, computation stops finally when the energy achieves the minimum value. The strength of traction is determined by the energy. When the control points are moved toward their center continuously, some image properties will influence energy computation and cause the control points to stop moving.



(a) (b)

Figure 4.1 Performance of the snake algorithm. (a) The behavior of elastic band and an object in it. (b) Final result of repeating continuous computations.

The energy for a snake exists in two parts, the *internal* and *external energies*. That is,

$$E_{snake} = \int E_{internal}\mathbf{v}(s) + \int E_{external}\mathbf{v}(s) \qquad (4.1)$$

where $\boldsymbol{v}(s) = (x(s), y(s))$ which specifies image coordinates. The *internal* energy $E_{internal}$ is the part that depends on the intrinsic properties of the snake, such as its length or curvature. The *external* energy $E_{external}$ depends on factors such as the image structure, and the particular constraints the user has imposed. Each control point has its *internal* and *external* energies and E_{snake} is the total energy of all the control points.

The equation of the *internal* energy is written as

$$E_{internal} = \alpha \cdot |\mathbf{v}_s(s)|^2 + \beta \cdot |\mathbf{v}_{ss}(s)|^2 . \tag{4.2}$$

The subscript s denotes the derivative. The above equation is the sum of the first derivative and the second derivative of the control point. Also, the values α and β are the coefficients determined by the designer.

The first derivate can be written as

$$\left|\frac{d\mathbf{v}_i}{ds}\right|^2 \approx |\mathbf{v}_i - \mathbf{v}_{i-1}|^2 = (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2,\ i = 1, 2, ..., n. \tag{4.3}$$

From the above equation, we can understand that when two control points are more close, the value of the first derivate will be smaller. Hence, the first derivate accounts for stretching of the snake and will have a large value when there is a gap between the control points. So, when the distribution of the control points becomes smaller, the *internal* energy will have a smaller value, as shown in Figure 4.2 (a) and (b).

The second derivate can be written as

$$\left|\frac{d^2\mathbf{v}_i}{ds^2}\right|^2 \approx |\mathbf{v}_{i-1} - 2\mathbf{v}_i + \mathbf{v}_{i+1}|^2 = (x_{i-1} - 2x_i + x_{i+1})^2 + (y_{i-1} - 2y_i + y_{i+1})^2. \tag{4.4}$$

The second order term accounts for bending and will be large in a region of high curvature. When the edge of the object is smooth, the *internal* energy will have a smaller value, as shown in Figure 4.2(c) and (d). Therefore, when the control points are closer to each other and their distribution shape is smooth, a smaller *internal* energy is obtained.

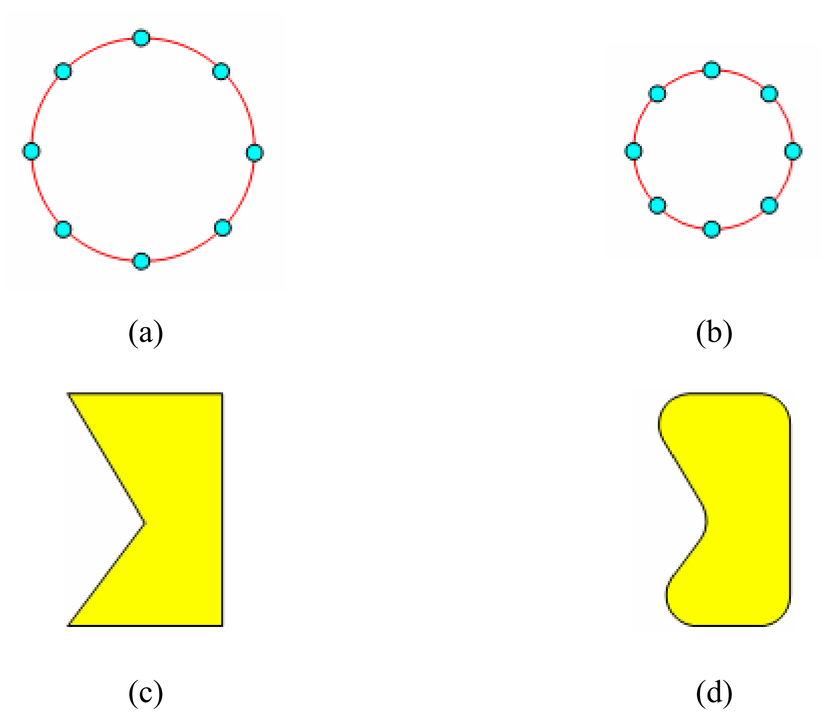


Figure 4.2 A illustration of *internal* energy. (a) A pattern has a larger *internal* energy. (b) A pattern has a smaller *internal* energy. (c) A pattern has a larger *internal* energy. (d) A pattern has a smaller *internal* energy.

Now we will focus on the *external* forces on the snake. These determine its relationship to the image. Suppose that we want a snake to latch on to bright structures in the image. We usually use the magnitude of the gradient in image processing.

4.2.2 Description of Improved Snake Algorithm

In the last section, we have described the principle of the snake algorithm. We will now describe the proposed method in this section.

Before computing the E_{snake} energy, the locations of the control points have to be decided at first. In this study, the shape distributed by the control points is a rectangle at the beginning in the image plane. A larger object in the image will have a larger rectangle, and vice versa. The terminal of control point shrinking is the center of the locations of the control points, as shown in Figure 4.3. The center of the control point $\boldsymbol{P}_{center}(x_0, y_0)$ is computed as:

$$\boldsymbol{P}_{center}(x_0, y_0) = \left(\frac{\sum_{i=1}^{n} x_i}{n}, \frac{\sum_{i=1}^{n} y_i}{n}\right). \tag{4.5}$$

where n is the number of the control points, and x_i and y_i are the coordinates of the control points in the image plane. Each control point moves toward the center point $\boldsymbol{P}_{center}(x_0, y_0)$ in each cycle of computation. The moving path is computed as a vector $\vec{V}(x_v, y_v)$ in the following way:

$$\vec{V} = \begin{bmatrix} x_v \\ y_v \end{bmatrix} = \begin{bmatrix} x_i - x_0 \\ y_i - y_0 \end{bmatrix}. \tag{4.6}$$

After computing E_{snake} each time, the control points move to the center point $\boldsymbol{P}_{center}(x_0, y_0)$ gradually. The new position of the control points in each moving step becomes:

$$\begin{aligned} x_i &= \left|\frac{x_v - 1}{x_v}\right| \times x_v + x_0, \\ y_i &= \left|\frac{y_v - 1}{y_v}\right| \times y_v + y_0. \end{aligned} \tag{4.7}$$

Through repeated computations, the shape of the control point distribution is no longer a rectangle. The distribution will be deformed gradually and the control point is close to the center point $P_{center}(x_0, y_0)$. Until the E_{snake} reaches a minimum value, .the process of the snake algorithm ends.

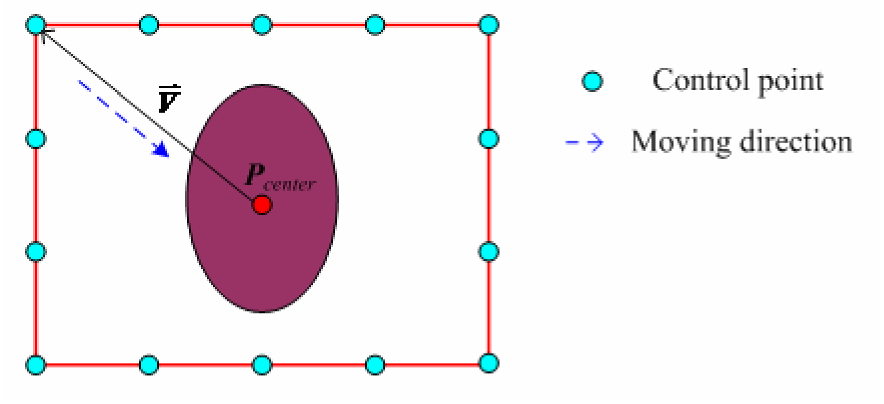


Figure 4.3 Contribution of control point.

The strength of shrinking comes from *internal* energy computation. Besides the method that Section 4.2.1 mentions, we add the so-called *centripetal force* into the *internal* energy. The improved *internal energy* is computed in the following way:

$$E_{internal} = \alpha \cdot \left|\frac{d\mathbf{v}_i}{ds}\right|^2 + \beta \cdot \left|\frac{d^2\mathbf{v}_i}{ds^2}\right|^2 + \gamma \cdot \left|\overline{V}\right|^2. \tag{4.8}$$

The detailed mathematical operation is specified by:

$$\begin{aligned} E_{internal} = {} & \alpha(x_i - x_{i-1})^2 + \alpha(y_i - y_{i-1})^2 + \beta(x_{i-1} - 2x_i + x_{i+1})^2 + \\ & \beta(y_{i-1} - 2y_i + y_{i+1})^2 + \gamma(x_i - x_0)^2 + \gamma(y_i - y_0)^2. \end{aligned} \tag{4.9}$$

We utilize the distance between a control point and $P_{center}(x_0, y_0)$ as an element for computing *internal* energy. When the control point is closer to the point $P_{center}(x_0, y_0)$, the distance becomes smaller gradually so that the *internal* energy has a smaller value. Hence, using centripetal force causes the control points to gather together.

The internal energy causes the control points to gather, so we have to compute the *external* energy so that the moving of the control points can be stopped. The final result is that the control points surround an object. The *Sobel operator* is shown in Figure 4.4 and its equation is shown in Eq.(4.10). The following specifies its operation formula:

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Figure 4.4 Sobel operators.

$$\begin{aligned} I(u,v) = &\left|(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)\right| \\ &+ \left|(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)\right| \end{aligned} \tag{4.10}$$

where z_5 denotes gray value in the position (u,v), z_1 denotes gray value in (u − 1,v − 1), and so on.

By using the *Sobel operator*, the edge of an object becomes obvious, as shown in Figure 4.5. Hence, utilizing the value of the *Sobel operator* of the pixels on the control point's position, we can get the *external energy* of a control point, as illustrated in the following way:

$E_{external}$ = $\lambda \times$ [minus (value of *Sobel operator* of the control point in the image)^2] (4.11)

where λ is a coefficient. Finally, the total energy, i.e., the snake energy, is computed as follows:

$$E_{snake} = \sum_{i=1}^{n} E^{i}_{internal} + \sum_{i=1}^{n} E^{i}_{external} \tag{4.12}$$

where n is the number of control points.

Because the data are discrete, the integration is substituted by a summation. The snake energy E_{snake} is the sum of each control point's *internal* and *external* energy. The *internal* energy is smaller when the control points are closer to the center such that the snake energy has smaller value. As soon as the control points bump into the edges, the *external energy* has a larger value such that the snake energy value becomes larger; in the mean while the snake energy computation is stopped because the minimum snake energy is reached.

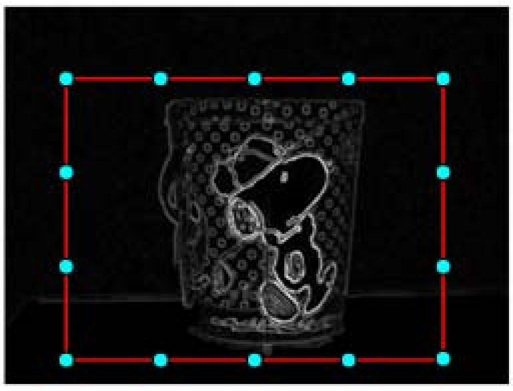


Figure 4.5 The result of using the Sobel operator.

The coefficients, α, β, γ, and λ play important roles in the snake energy. The magnitudes of the coefficients α and β influence the control points' pulling force. The magnitude of coefficient γ influences the cohesion of the control points, and the magnitude of coefficient λ influences the result of the captured object edge. In this study, the magnitudes of α, β, γ, and λ are set to be 2, 3, 2 and 1, respectively.

4.2.3 Computing Object Region

After detecting an object, we have to recognize the object region from the image. Through the above repeated calculations, the control points surround the object edge pixels finally. Utilizing the coordinates of two adjacent control points to compute a straight equation $y = ax + b$, we can find out the pixels of the object edge by using this equation. A detailed process is described in the following algorithm.

Algorithm 4.1. *Object region extraction*.

Input: A set of control points ***P***.

Output: An object region *ObjR*.

Steps:

Step 1. For any two adjacent points $P_s(x_s, y_s)$ and $P_t(x_t, y_t)$, do the following steps.

Step 1.1 Compute a straight line equation $y = ax + b$ which crosses P_s and P_t by the following equation:

$$(a,b) = \left(\frac{y_t - y_s}{x_t - x_s}, \frac{y_s x_t - y_t x_s}{x_t - x_s}\right), \text{ if } (x_t - x_s) \neq 0. \quad (4.13)$$

If $x_t - x_s = 0$, the equation becomes $x = x_t$.

Step 1.2 Compute the value y of the edge point coordinates between P_s and P_t by using the value of x in the following way:

$$y = ax + b,\ x = x_t \text{ to } x_s. \quad (4.14)$$

Step 2. Find out the leftmost pixel and rightmost pixel denoted as (x_L, y_L) and (x_R, y_R) after comparing the x coordinate of each edge pixel.

Step 3. Find out the maximum and minimum y coordinates of each value of x_i, denoted as (x_{ui}, y_{ui}) and (x_{di}, y_{di}).

Step 4. For each pixel, $P_i(x_i, y_i)$, if its x coordinate is between x_L and x_R and its value of y satisfies the following inequality, then P_i is considered as an object pixel:

$$y_{di} \leq y_i \leq y_{ui}. \quad (4.15)$$

4.2.4 Eliminating Noise of Background

Because of the effect of the noise and edges in the background, the experimental result of object detection is usually not satisfactory, as shown in Figure 4.6. If a wall or the floor under the object is not clear, it usually influences the result of the moving of the control points. Hence, we have to improve the method for object detection.

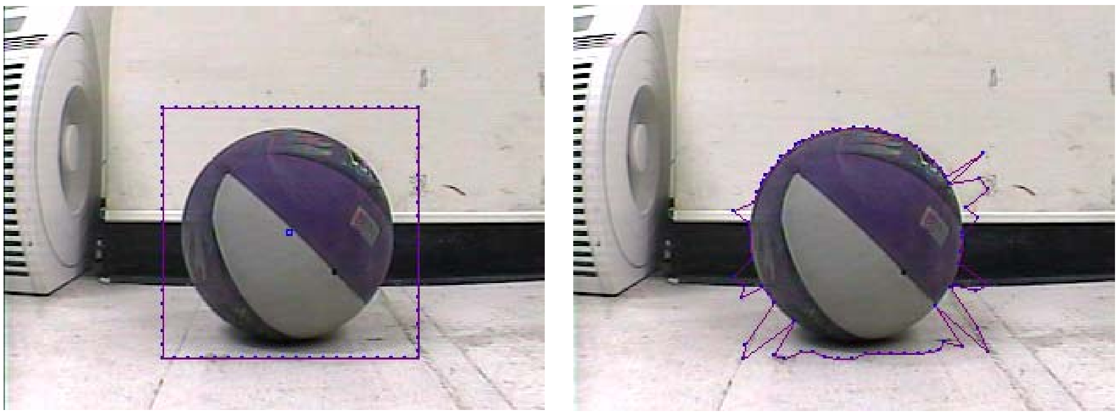

(a) (b)

Figure 4.6 Effect of background on object detection (a) Before detection. (b) Result of detection.

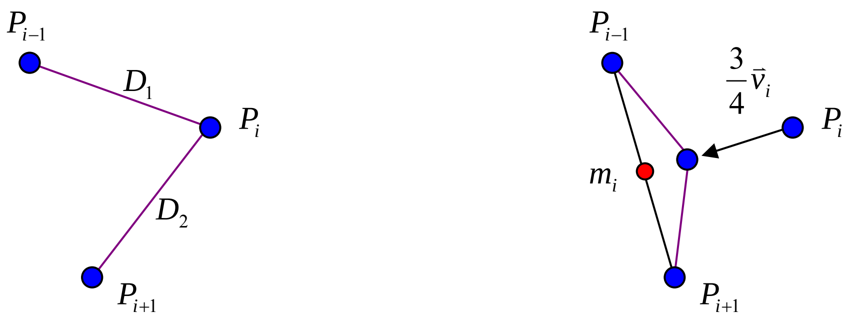
Besides the methods of Section 4.2, we add the strength of pull and drag between the control points. We hope that the distance of adjacent control points can keep inside a reasonable domain. In order to achieve this goal, the number of the control points is determined by a beginning rectangle size. According to the length of a rectangle's boundary and the coordinates of the four endpoints of the rectangle, we compute every control point position on the edge of the rectangle so that every control point can keep a fixed distance to each other. Utilizing the distance of the control points, we can adopt a threshold D_t such that the distance between two adjacent control points are kept inside D_t when the control points are moving along the path.

As illustrated in Figure 4.7(a), the control point $P_i(x_i, y_i)$, and the two points, $P_{i-1}(x_{i-1}, y_{i-1})$ and $P_{i+1}(x_{i+1}, y_{i+1})$ are adjacent points of P_i. We compute the distances D_1 and D_2 by the following equations:

$$D_1 = |x_i - x_{i-1}| + |y_i - y_{i-1}|; \qquad (4.16)$$

$$D_2 = |x_i - x_{i+1}| + |y_i - y_{i+1}|. \qquad (4.17)$$

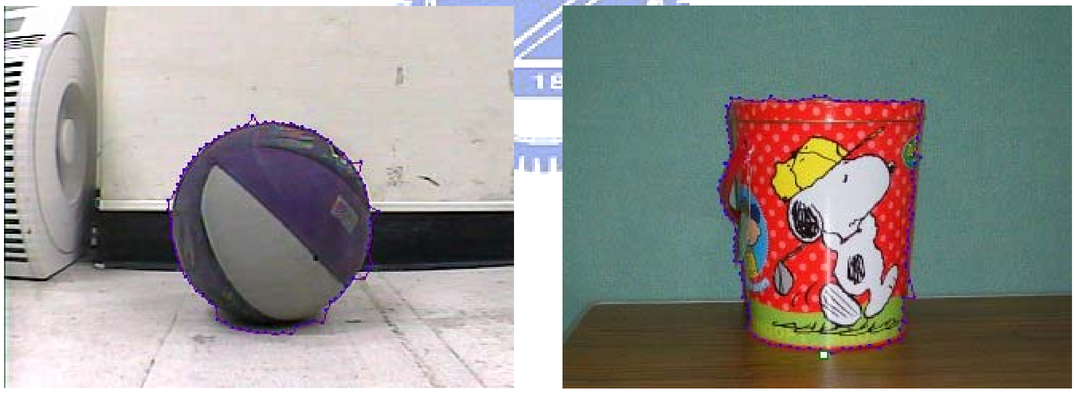
If the inequality $D_1 + D_2 > D_t$ is true, then we will modify the position of the control point P_i, as shown in Figure 4.7(b). Utilizing the coordinates of the control points P_{i-1} and P_{i+1}, we can get a center position m_i between them by using the equation of $m_i = \frac{P_i + P_{i+1}}{2}$. After computing a vector $\vec{v}_i = m_i - P_i$, the position of the control point P_i is changed to the new position specified by $P_i + \frac{3}{4}\vec{v}_i$.



(a) (b)

Figure 4.7 Correction of control points. (a) Before correction. (b) After correction.

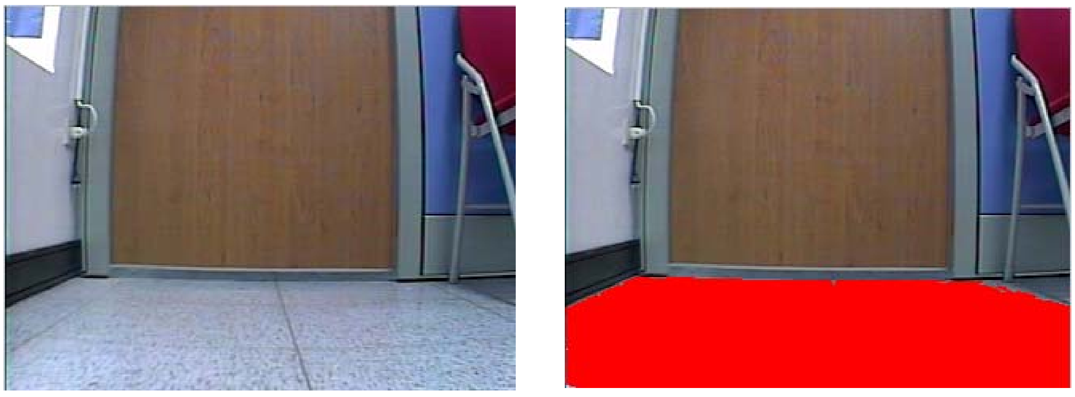
An experimental result of the above-mentioned method is shown in Figure 4.8.



(a) (b)

Figure 4.8 Results of improved method. (a) Detecting a ball. (b) Detecting a tub

The vehicle sometimes will navigate on a decorative floor, as shown in Figure 4.9(a). Before detecting the object, we have to filter the floor region by using a region growing technique. A result is shown in Figure 4.9(b). When the position of control point is on the floor region, it is moved without being stopped until it leaves the floor region.



(a) (b)

Figure 4.9 Elimination of a floor region. (a) A decorative floor. (b) An experimental result.

4.2.5 Detailed Object Detection

We have already described the concept behind the proposed improved snake algorithm in the above sections. The detail of the proposed object detection process is described in the following algorithm.

Algorithm 4.2. *Object detection.*

Input: An object image *I*.

Output: An object region *ObjR*.

Steps:

Step 1. Specify manually four end points of a rectangle in the ICS as initial control points and compute the coordinates of the four points.

Step 2. Compute the coordinates of the other control points according to the coordinates of the four endpoints.

Step 3. Compute the center point P_{center} by using the coordinates of all the control points.

Step 4. Use a region growing technique to filter out the floor region.

Step 5. Compute the internal energy $E_{internal}$ and the external energy $E_{external}$ of every control point. And compute the original snake energy E_{snake} by summing up all the values of $E_{internal}$ and $E_{external}$.

Step 6. For each control point, do the following steps.

Step 6.1. Compute next position of the control point on the shrinking path and its internal energy $E_{internal}$ and external energy $E_{external}$ and the new snake energy $newE_{snake}$ by summing up $E_{internal}$ and $E_{external}$.

Step 6.2. Compare the original snake energy E_{snake} with $newE_{snake}$.

Step 6.3. If original energy E_{snake} is smaller than $newE_{snake}$, the control point is not moved to the next position. Else moving the control point to the next position and substitute E_{snake} with the new snake energy $newE_{snake}$.

Step 6.4. Compute the distances D_1 and D_2 between the control point and its two adjacent points.

Step 6.5. Sum up D_1 and D_2. If their sum is larger than a threshold D_t, then adjust the coordinates of the control point by using the method described in Section 4.2.3 and compute the new snake energy.

Step 7. When reaching the minimum energy, stop moving the control points.

Step 8. Compute the object edges using the coordinates of the control points.

Step 9. Detect object region *ObjR* using the coordinates of the object edge points.

4.3 Object Feature Extraction

After detecting an object from an image, we hope to gather useful features of the object for object matching. Through comparing the object features with those of a learned object, we can achieve the purpose of security patrolling. The object features for use in this study are color, shape, and coordinates in the GCS. Among the features, the coordinates are used as vehicle navigation data, and the others are used for object matching. In Section 4.3.1, we will illustrate how to use simple mathematics to describe color features. We will use an ellipse to fit the shape of each object, as illustrated in Section 4.3.2. Finally, a coordinate transformation is described in Section 4.3.3 for computing the coordinates of objects in the GCS.

4.3.1 Color Feature

Each object has its own colors which are the most useful feature. We can use it for object matching in the security patrolling process. About color data, we adopt the RGB color model and choose certain statistics to describe color features in this study.

After segmentation of object pixels, we compute the means of the R, G, and B colors, respectively, and denote them as $\overline{R}_{obj}$, $\overline{G}_{obj}$, and $\overline{B}_{obj}$. The means can be used to describe the whole color situation and are influenced slightly by noise or some fragmented edges. The equation is shown as follows:

$$(\overline{R}_{obj}, \overline{G}_{obj}, \overline{B}_{obj}) = \left(\frac{\sum_{i=1}^{n} R_i}{n}, \frac{\sum_{i=1}^{n} G_i}{n}, \frac{\sum_{i=1}^{n} B_i}{n}\right) \tag{4.18}$$

where R_i, G_i, and B_i are the R , G, and B values of object pixels, and n is the number of the object pixels.

Representing the color deviation of the object, we use the standard deviation to show this feature. We compute the standard deviations of the R, G, and B values, respectively, which are denoted as R_{obj}^{sd}, G_{obj}^{sd}, and B_{obj}^{sd}. The equation for this is shown as follows:

$$(R_{obj}^{sd}, G_{obj}^{sd}, B_{obj}^{sd}) = \left(\sqrt{\frac{\sum_{i=1}^{n}(R_i - \overline{R}_{obj})^2}{n}}, \sqrt{\frac{\sum_{i=1}^{n}(G_i - \overline{G}_{obj})^2}{n}}, \sqrt{\frac{\sum_{i=1}^{n}(B_i - \overline{B}_{obj})^2}{n}}\right) \qquad (4.19)$$

4.3.2 Ellipse Fitting for Shape Representation

Shapes of objects are usually different, such as circle, triangle, ellipse, rectangle, etc. Moreover, many object shapes are irregular. Hence, if all shapes can be represented by using only one shape, it will be convenient and fast. In this study, we use the ellipse shape to represent all object shapes. Although the shapes of objects are different, we always can compute an ellipse shape to fit the object shape.

According to the ellipse equation, $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$, there are two unknown variables a and b which are the horizontal and the vertical axis length. We can distinguish different objects by different values of a and b. Because the center of the ellipse is usually taken to be the origin of the coordinate system, we must change the coordinates of the object pixels and the origin is moved to the center of object region such that the equation $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$ can be used.

The center of the object region is denoted as $P_{obj}^0(x_{obj}^0, y_{obj}^0)$, and calculated by the following equation:

$$(x_{obj}^0, y_{obj}^0) = \frac{1}{n}\left(\sum_{i=0}^{n} x_i, \sum_{i=0}^{n} y_i\right) \qquad (4.20)$$

where variables, x_i and y_i, are the coordinates of the object pixels in the ICS. For all the object pixels (x_i, y_i), they are moved to the new coordinates, denoted as x_i' and y_i', by using the equation $(x_i', y_i') = (x_i - x_{obj}^0, y_i - y_{obj}^0)$ and the least square method to solve variables a and b by the following equation:

$$a = \sqrt{\frac{(\sum_{i=1}^{n} x_i'^2 y_i'^2)^2 - (\sum_{i=0}^{n} x_i'^4)(\sum_{i=0}^{n} y_i'^4)}{(\sum_{i=0}^{n} x_i'^2 y_i'^2)(\sum_{i=0}^{n} y_i'^2) - (\sum_{i=0}^{n} x_i'^2)(\sum_{i=0}^{n} y_i'^4)}}; \tag{4.21}$$

$$b = \sqrt{\frac{(\sum_{i=0}^{n} x_i'^2 y_i'^2)^2 - (\sum_{i=0}^{n} x_i'^4)(\sum_{i=0}^{n} y_i'^4)}{(\sum_{i=0}^{n} x_i'^2 y_i'^2)(\sum_{i=0}^{n} x_i'^2) - (\sum_{i=0}^{n} x_i'^4)(\sum_{i=0}^{n} y_i'^2)}}. \tag{4.22}$$

Finally, we take *a* and *b* to represent this object. An experimental result is shown in Figure 4.10.

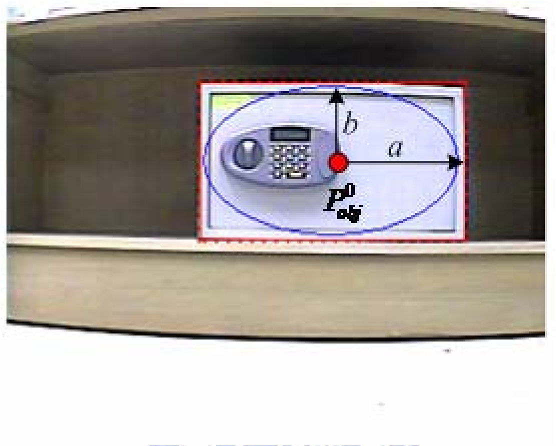


Figure 4.10 The ellipse representation of a safe.

4.3.3 Coordinate Transformation

The coordinates of an object are useful for navigation guidance. When the vehicle navigates along a route, it can detect the object according to coordinates of the learned object to help navigation on the right path. In this study, we use the coordinates of the object in the GCS as navigation data.

After detecting an object in an image, we use Equation (4.20) to compute the center of object pixels, such as the red point shown in Figure 4.11. By using a region-growing technique to find out the floor region in the image, the object can then be located on the floor, such as the green point shown in Figure 4.11. Computing the coordinates of green points in the VCS by the 2D mapping technique as illustrated in Chapter 3, we can get relative positions of the vehicle and the object.

Using the coordinates and the direction angle of the vehicle in the GCS, we can compute the coordinates of the object in the GCS by the following equations:

$$Gx_{obj} = Vx_{obj} \cdot \cos\theta - Vy_{obj} \cdot \sin\theta + Gx_{car}; \tag{4.23}$$

$$Gy_{obj} = Vx_{obj} \cdot \sin\theta + Vy_{obj} \cdot \cos\theta + Gy_{car}, \tag{4.24}$$

where Gx_{obj} and Gy_{obj} are the coordinates of the object in the GCS, Vx_{obj} and Vy_{obj} are the coordinates of the object in the VCS, and Gx_{car} and Gy_{car} are the coordinates of the vehicle in the GCS. Finally, θ is the direction angle of the vehicle in the GCS.

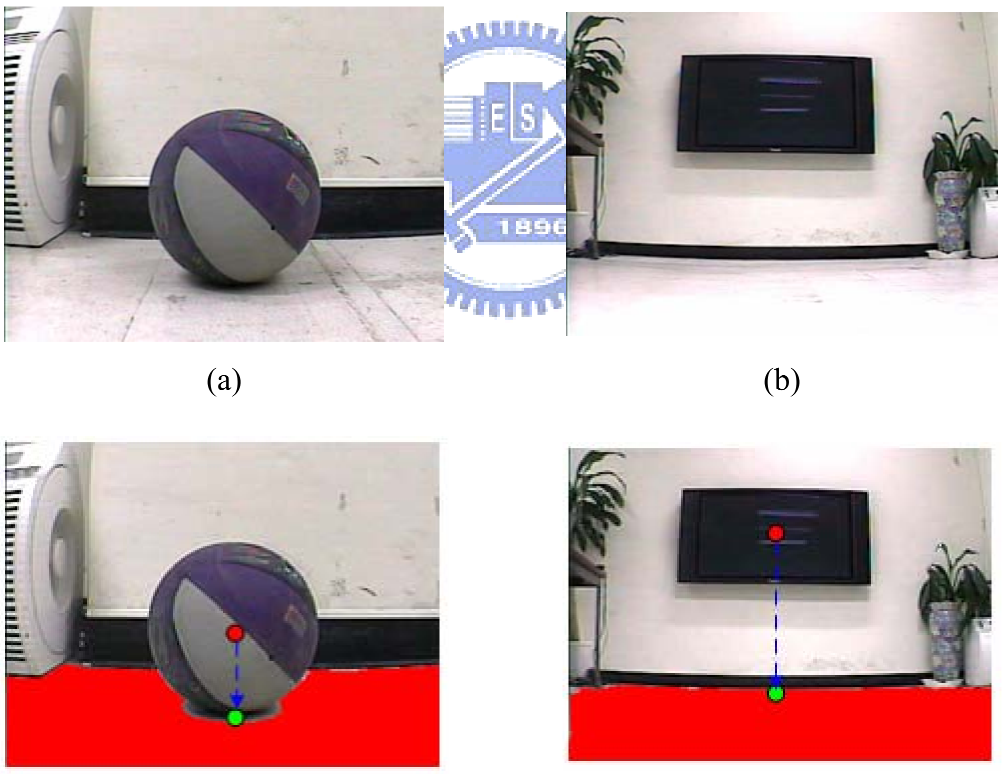

(a) (b) (c) (d)

Figure 4.11 The locations of objects (a) An original image in which a ball is on the floor. (b) An original image in which a TV is on the wall. (c) An experimental result of (a). (d) An experimental result of (b).

Chapter 5
Learning Strategies for Indoor Navigation by Manual Driving

5.1 Introduction

Before the vehicle navigates, it is a necessary process that we control the vehicle to record the paths and objects. Because indoor environments are usually complicated and objects are placed at different positions, building a complete navigation map is necessary. Hence, to create the navigation path and choose monitored objects is a primary work of security patrolling by vehicle navigation. We will describe how to build navigation data by manual driving in this chapter.

In Section 5.2, we will first describe the control rules and the entire manual learning process simply. It includes all the steps that the user may use a control interface to control the vehicle to patrol in indoor surroundings and point out which object has to be monitored by using the control system.

Two kinds of navigation data are used in this study. One is path data and the other object data. Although we can get the position of the vehicle by the odometer value any time, how to represent the entire path by using useful and simple values is a problem. In Section 5.3, we will describe how to gather path data when the user controls the vehicle to navigate in indoor environment. Every object has its own color and shape; if different kinds of objects use different learning methods, the work will be annoying to the user. We design a simple object learning method to solve this problem in this study, as illustrated in Section 5.4. The door is considered as an object, and monitoring of it is also described in Section 5.4.

After learning whole data, we have to utilize the data to build information for security patrolling. In Section 5.5, we will describe how to use path data and the positions of learned objects to create a navigation path which is then used when the vehicle navigates automatically.

5.2 Control Rules And Entire learning Process

In this study, the user controls the vehicle to navigate by the following fives types of actions.

(1) Moving forward.

(2) Moving backward.

(3) Turning left at the original position.

(4) Turning right at the original position.

(5) Stop.

Because the vehicle only owns three wheels and the rear wheel is an auxiliary wheel, rotation of the vehicle depends on the two side-wheels. Also, the vehicle can rotate at any position. Therefore, we can control the vehicle to move to the neighborhood of an object and a door, and let it turn to the front of the object or the door to learn relative object features.

The control rules for the vehicle are described as follows.

(1) When a user wants to turn rightward or leftward, the vehicle should be made still

first. If the vehicle is moving, the user has to stop the vehicle and then turn the vehicle.

(2) Because the view of camera is fixed, there is a limit of distance in the front of the vehicle, 60cm, in which the camera can not take clear images of the scene. Hence, the distance between the vehicle and a learning object has to be kept larger than 60cm.

(3) The process of object learning should be done when the vehicle is still.

(4) The object region should be around the center of an image.

The entire learning process is described as following algorithm and a example of learning process is shown in Figure 5.1.

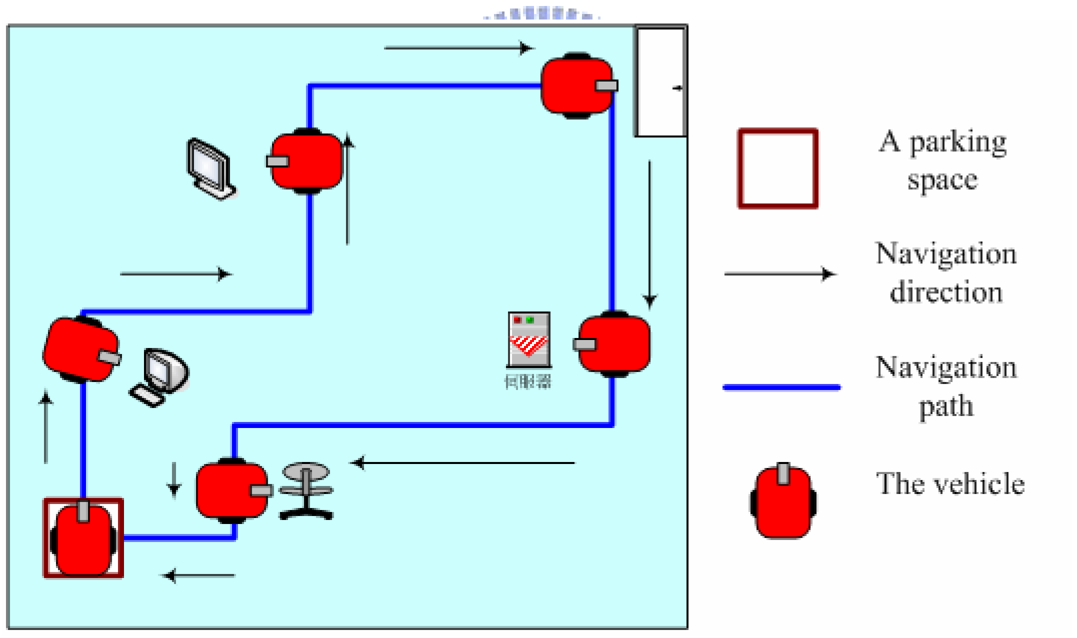


Figure 5.1 Proposed learning process.

Algorithm 5.1. *Learning process*.

Input: The user control interface in the PC.

Output: Learned data.

Steps:

Step 1. Control the vehicle to move from a start position using the control interface.

Step 2. Let the vehicle move to the front of a monitored object or a door.

Step 3. Choose the monitored object by using the user control interface in the control system, a personal computer in this study. Compute the features of objects or doors, and show the result on the control interface.

Step 4. If the learning result of the object and door are not satisfactory, repeat Step 3 until satisfactory, and then save the features of the objects and doors.

Step 5. Repeat Steps 2 through 4 until all the features of the monitored objects are grasped and saved.

Step 6. If another object is to be learned, repeat Steps 2 through 5; else, continue following steps.

Step 7. Decide whether the learning process should be continued, and continue driving and repeat Step 2 through 6 if so; else control the vehicle to move to the destination and finish the learning process.

Step 8. Save the learned data.

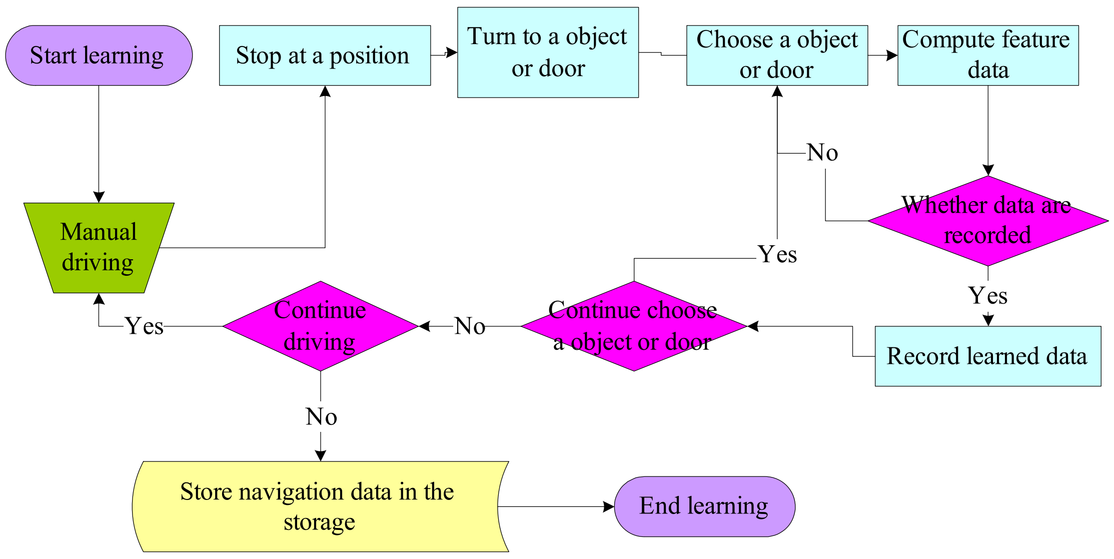


Figure 5.2 Flowchart of manual learning process.

5.3 Process of Learning Navigation Paths

When the vehicle moves along a path, the odometer values are changed continuously. The odometer provides the rotation angles and the vehicle coordinates (x, y) with the coordinate origin being the beginning position of the vehicle. We only record vehicle coordinates as path data in this study.

Although the coordinates are changing all the time, we just save some coordinates (x, y) which are called *node* N_i in this study. Two types of data are stored in node N_i. Besides the coordinates, a number denoted as *NNumber* and used to mark the order of the object is also saved. The number is computed from 0.

Saving the path data is different from saving the object data. The user has to point out an object manually in the acquired image on the interface in the object learning process, but the user does not have to do so in the path learning process. When the user controls the vehicle to move ahead or back, the vehicle system will automatically collect values of the coordinates (x, y) and the moving direction.

Each node of a path is marked with a serial number. After finishing learning, we have a set of notes, denoted as $\boldsymbol{N_{path}}$. The process of recording the path data is described as an algorithm in the following.

Algorithm 5.2. *Path node collection.*

Input: The coordinates provided by the odometer in the vehicle.

Output: A set of nodes denoted by $\boldsymbol{N_{path}} = \{N_0, N_1, N_2, \ldots, N_t\}$.

Steps:

Step 1. Record the first node as $(x_0, y_0) = (0, 0)$ into the set $\boldsymbol{N_{path}}$ and mark the node as N_0 with index 0, when the vehicle is at the starting position.

Step 2. Record the node $N_i(x_i, y_i)$ into the set $\boldsymbol{N_{path}}$ by taking the values of the odometer (x, y) and mark the node P_i with the next index number, when the vehicle is at one of the following three situations:

(1) when the direction angle and coordinates are corrected, as described in Section 3.3;

(2) when the user controls the vehicle to turn;

(3) when the user controls the vehicle to learn the data of certain objects.

Step 3. Record the finally node N_t into the set $\boldsymbol{N_{path}}$ and mark it as N_t by the next index number when the learning process is finished.

Step 4. Save all the nodes of the set $\boldsymbol{N_{path}}$ into the PC.

According to Section 3.3, the navigation path of vehicle is a curve when the vehicle moves ahead. In the learning process, the direction angle and coordinates of the vehicle are corrected automatically. In this study, we design the direction angle and coordinates to be corrected once each time when the vehicle moves for the distance of 250cm. We record a node each time the vehicle comes to a stop in a learning course. Besides the start position which the vehicle begins to move ahead, there are two types of nodes which should be recorded. One is the position where the vehicle turns back to a straight path, as denoted by node *s*1 in Figure 5.3(a). The other is the position where the vehicle already comes back to a straight path, as denoted by node *s*2 in Figure 5.3(a). Recorded nodes are shown in Figure 5.3(b) when the vehicle navigates along a straight path. And we use the same method to record nodes when the vehicle moves backward.

As an illustration of the result of applying Algorithm 5.2, we show an example of recorded nodes in Figure 5.4 that a navigation path shown in Figure 5.1 mentioned in Section 5.2. We can see that all critical nodes of the three situations are recorded in addition to the start and the end nodes. All nodes are marked with index numbers according to the order of patrolling. And the start node and the end node are the same node in this example. The index numbers are useful for path map creation and object detection. We will describe them in detail in Section 5.5.

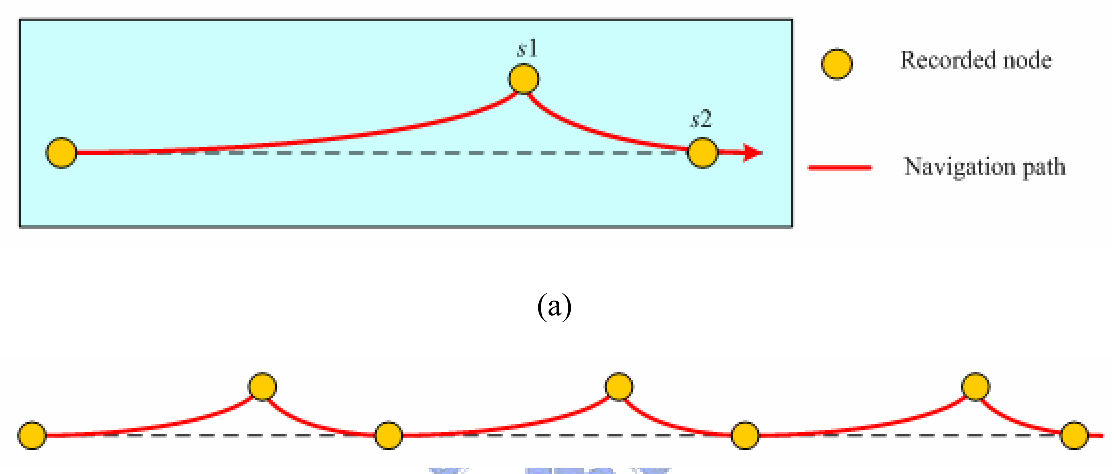


Figure 5.3 Recorded nodes when the vehicle moves straightly. (a) Nodes in one circle of the navigation correction. (b) Nodes in a longer path.

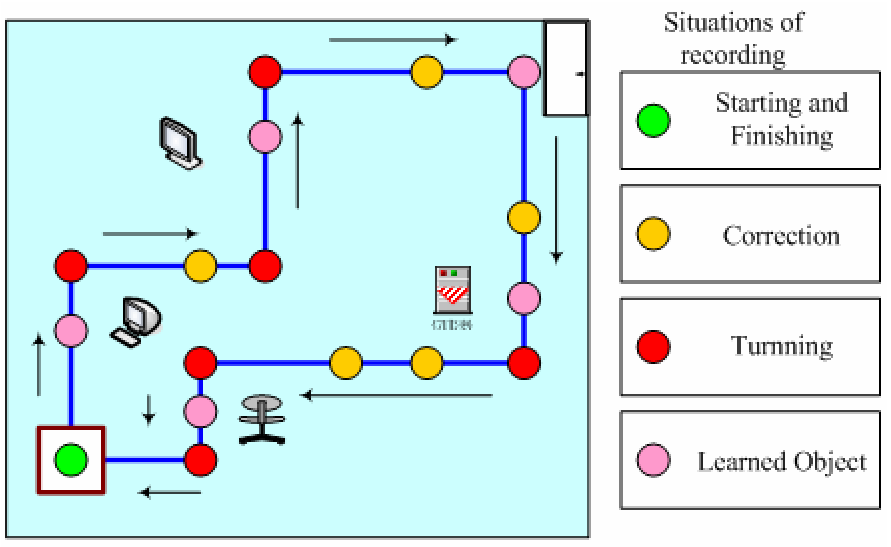


Figure 5.4 An experimental result of path learning with critical nodes recorded.

5.4 Process of Learning Monitored Objects and Doors

5.4.1 Process of Learning Objects

When the user controls the vehicle to move to the front of objects, the user must use the mouse connected to the PC to choose an object which appears in the image. As soon as the user chooses the object, the object data are computed automatically and saved. The set of object data is denoted by $LearnO_i$ and $i = 1, 2, 3, \ldots, n$. There are six kinds of data to be saved in $LearnO_i$. They are:

(1) The color set denoted as $LearnC_{obj,i}$.

(2) The shape set denoted as $LearnS_{obj,i}$.

(3) The GCS coordinate set denoted as $LearnGC_{obj,i}$.

(4) The ICS coordinate set of object centers denoted as $LearnCIC_{obj,i}$.

(5) The set of light source of the floor, denoted as $LearnC_{floor,i}$.

(6) The number of path nodes *NodeNumber*.

Hence, we have $LearnO_i = \{LearnCIC_{obj,i}, LearnC_{obj,i}, LearnS_{obj,i}, LearnGC_{obj,i}, LearnC_{floor,i}, NodeNumber\}$.

The methods for computing the color, shape, and coordinates of the object have already been described in Section 4.4. After the vehicle records all the data of monitored objects $LearnO_i$, we save the data into the set $\boldsymbol{LearnO_{object}}$. The entire learning process of the object is described as follows.

Algorithm 5.3. *Learning of object features.*

Input: A color image I captured by the camera on the vehicle.

Output: A set of object data $\boldsymbol{LearnO_{object}} = \{LearnO_0, LearnO_1, LearnO_2, \ldots,$

$LearnO_n\}$.

Steps:

Step 1. Control the vehicle to move to the front of the first object and turn toward it such that the image of the object can be taken.

Step 2. Use the mouse to choose an object from the image, as shown in Figure 5.5(a) and enclose the object by a rectangle.

Step 3. Release the button of the mouse for the computer to perform the improved snake algorithm to capture the object, as shown in Figure 5.5.

Step 4. Compute the elements of the object data in $LearnO_0$ as follows:

Step 3.1. Record the object center coordinates in the ICS into $LearnCIC_{obj,i}$ by using Equation (4.5) with $LearnCIC_{obj,i}$ computed by

$$LearnCIC_{obj,i} = \{ Learn\_u_{obj\_centerj,i}, Learn\_v_{obj\_centerj,i} \}. \tag{5.1}$$

Step 3.2. Record the mean and standard deviation values of the R, G, and B values into the color set $LearnC_{obj,i}$ by using Equations 4.13 and 4.14 with $LearnC_{obj,i}$ computed as

$$LearnC_{obj,i} = \{Learn\overline{R}_{obj,i}, Learn\overline{G}_{obj,i}, Learn\overline{B}_{obj,i}, LearnR^{sd}_{obj,i}, LearnG^{sd}_{obj,i}, LearnB^{sd}_{obj,i}\}. \tag{5.2}$$

Step 3.3. Record the shape data of the object into $LearnS_{obj,i}$ in terms of the horizontal axis $Learn\_a_{obj,i}$ and the vertical axis $Learn\ \_b_{obj,i}$ as illustrated in Section 4.4.2:

$$LearnS_{obj,i} = \{Learn\_a_{obj,i}, Learn\ \_b_{obj,i}\}. \tag{5.3}$$

Step 3.4. Record the coordinates of the object in the GCS into the set $LearnGC_{obj,i}$ in terms of $Learn\_x_{obj,i}$ and $Learn\_y_{obj,i}$ as illustrated in Section 4.4.3:

$$LearnGC_{obj,i} = \{ Learn\_x_{obj,i}, Learn\_y_{obj,i}\}. \tag{5.4}$$

Step 3.5. Record the means of the R, G, and B values of the floor into the set

$LearnC_{floor,i}$, denoted by $Learn\overline{R}_{floor,i}$, $Learn\overline{G}_{floor,i}$, and $Learn\overline{B}_{floor,i}$.

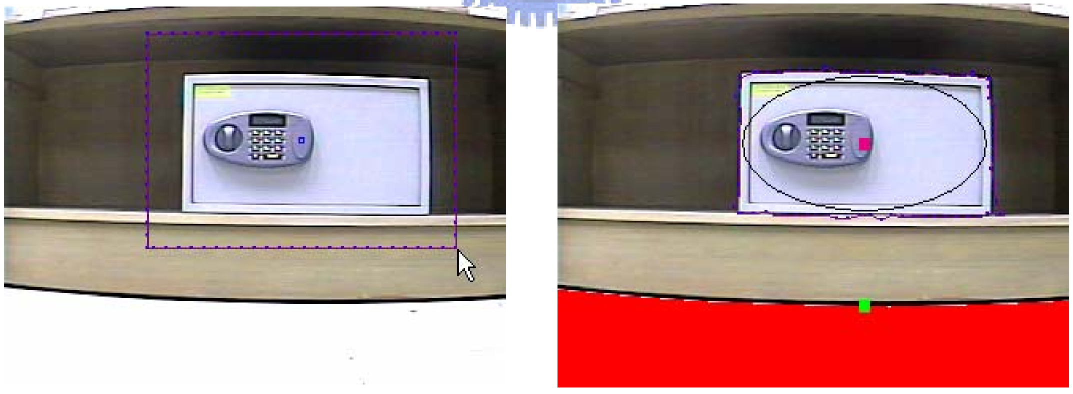
$$LearnC_{floor,i} = \{Learn\overline{R}_{floor,i}, Learn\overline{G}_{floor,i}, Learn\overline{B}_{floor,i}\}. \tag{5.5}$$

Step 3.6. Record the number of path nodes *NodeNumber* which the vehicle has located.

Step 5. Decide whether the data of object $LearnO_0$ should be saved into $\boldsymbol{LearnO_{object}}$ or not. If the result of object segmentation is not satisfactory, repeat Steps 2 through 4.

Step 6. Save the object data $LearnO_0$ into $\boldsymbol{LearnO_{object}}$ and control the vehicle to move to the next object and repeat Steps 2 through 5 to save object data $LearnO_i$ into $\boldsymbol{LearnO_{object}}$ until all data of monitored objects are collected and saved.

Step 7. Save $\boldsymbol{LearnO_{object}}$ into the PC and finish the learning process.



(a) (b)

Figure 5.5 Learning process of choosing an object manually. (a) Choosing an object. (b) An experimental result of computing object features data.

Because of the lighting effect of the environment, the R, G, and B values of an object are not always the same when the camera takes the object images at different times. It causes erroneous results when the vehicle navigates using the object color feature to conduct object matching. We utilize an offsetting technique to solve this problem. We record the color features of a floor area in an object learning process. In Figure 5.6, the red area is the floor area, and we compute the R, G, and B means of the interior area of the rectangle. As the object matching process is done when the vehicle navigates, we utilize the differences of the R, G, and B values of the floor region to modify the thresholds used in the object matching process.

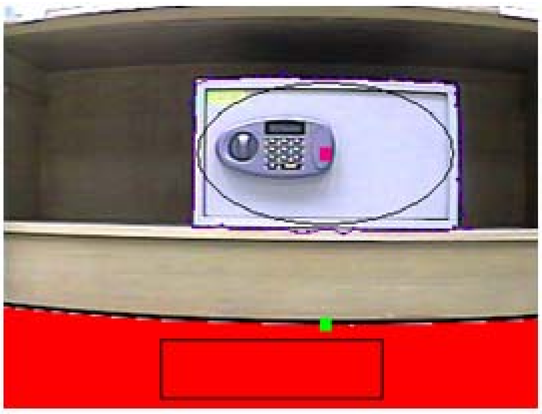


Figure 5.6 A selected floor region.

5.4.2 Process of Learning Doors

When a user controls the vehicle to move, he/she can choose doors as monitored objects. After the features of a door are recorded, the vehicle can check their situations in a security patrolling navigation. When the vehicle learns the features of a door, we record its color data (denoted as $LearnC_{door,i}$) and coordinates (denoted as $LearnGC_{door}$).

All door data are saved in a set D_i and the date sets of all doors in the navigation are saved in a set $LearnD = \{D_0, D_1, D_2, \ldots, D_n\}$. The entire learning process for a door is described in the following.

Algorithm 5.4. *Learning of a door.*

Input: An image *I*, and the number of nodes *NNumber*.

Output: A set of door data $D_i = \{LearnC_{door,i}, LearnGC_{door,i}, NodeNumber, baseline\}$.

Steps:

Step 1. Point out a door on the image *I* using a cursor, as shown in Figure 5.7(a).

Step 2. Use a region growing technique to find out the door region *DoorR*, as shown in Figure 5.7(b).

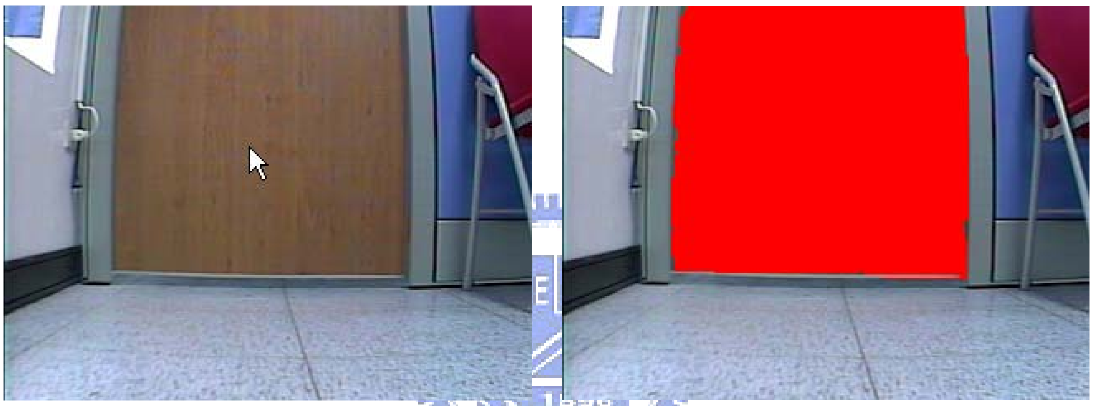

(a) (b)

Figure 5.7 A learning process of choosing a door manually. (a) Choose a door. (b) A door region is shown.

Step 3. Compute the means of the R, G, and B values of *DoorR* and save them in the set $LearnC_{door}$ as follows:

$$LearnC_{door,i} = \{Learn\overline{R}_{floor,i},\ Learn\overline{G}_{floor,i},\ Learn\overline{B}_{floor,i}\}. \tag{5.6}$$

step 1. Compute the coordinates of the door in the GCS and save them in the set $LearnGC_{door,i}$ as follows:

$$LearnGC_{door,i} = \{Learn\_x_{door,i},\ Learn\_y_{door,i}\}. \tag{5.7}$$

Step 4. Record the index number of the node as *NodeNuber* = *NNumber*.

Step 5. Detect the edges of *I* by applying the *Sobel operator* to get an edge image *S*.

Step 6. Detect the door edge e_{door}, the right baseline edge e_r, and the left baseline edge e_l of the door, as shown in Figure 5.8.

Step 7. Compute the slopes a_{door}, a_r, and a_l of e_{door}, e_r, and e_l, using a line fitting technique by the following equation:

$$a = \frac{\sum_e u \sum_e v - n \sum_e uv}{(\sum_e x)^2 - n \sum_e x^2} \tag{5.8}$$

Step 8. Compare a_r and a_l with a_{door} by the following equation:

$$| a_r - a_{door} | \le | a_l - a_{door} |. \tag{5.9}$$

If the above equation is true, then set *baseline* = 1; else, set *baseline* = 0. The variable *baseline* is used to illustrate which baseline is used in the navigation process.

As shown in Figure 5.8, we utilize a baseline of a wall and compare it with the edge of the door to test whether it is open or not, as done in Step 9 in the above algorithm. Hence, if the edge of the wall is not parallel to the door edge when the door is closed, we can not use its baseline to compare with the door edge. So we compute the slopes of two baselines and the door edge, and choose a baseline whose slope is close to the door edge for use in the comparison. The slope of e_r is closer to the slope of e_{door} than that of e_l, as seen in Figure 5.8.

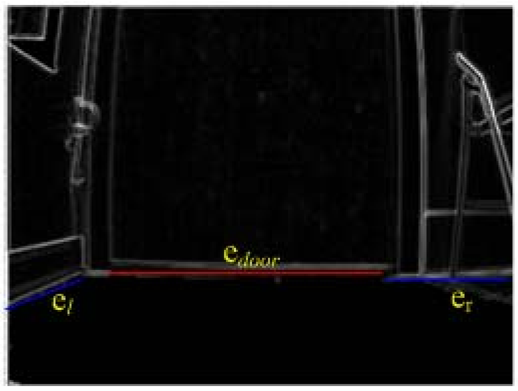


Figure 5.8 An experimental result of detection of three edges.

5.5 Process of Automatic Path Map Creation from Learned Data

After ending the learning process, the path data $\boldsymbol{N_{path}}$, the object data $\boldsymbol{LearnO_{object}}$, and the door data $\boldsymbol{LearnD_{door}}$ are already saved. We use the index number *NNumber* of each node and the total number of nodes *NodeNumber* to create a path map for later navigation sessions, as shown in Figure 5.9. By using the index numbers *NNumber* of the nodes, the vehicle can move along the navigation path.

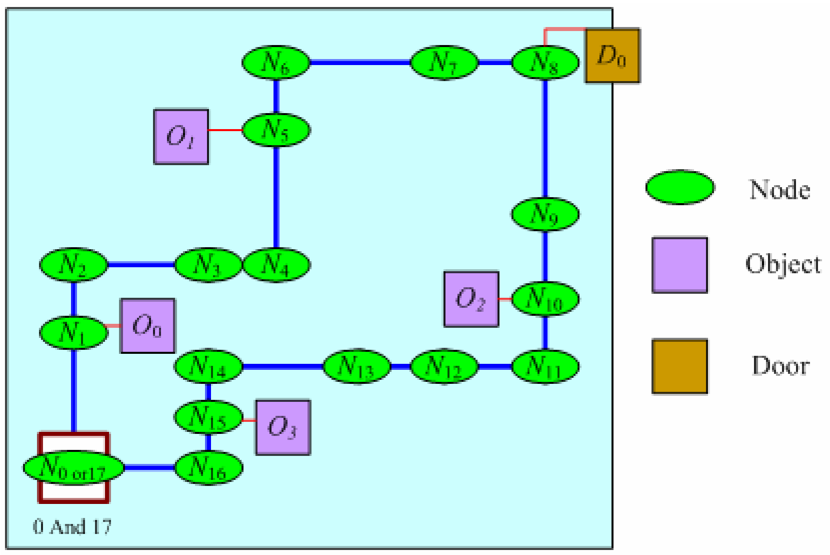


Figure 5.9 An example of navigation map.

Chapter 6
Security Patrolling in Indoor Environments

6.1 Introduction

By using the learning strategies mentioned in the previous chapters, path data and object data are saved in the PC. The vehicle can navigate according to the information. In this chapter, we will describe the entire process of security patrolling in more detail.

In Section 6.2, we will first describe a navigation process briefly. The vehicle navigates according to the node data and checks the existence of the monitored objects by using the coordinates of the learned object during the navigation process.

In Section 6.3, a line following technique is proposed. Since the vehicle navigates along the nodes one by one, the entire navigation path can be divided into many straight sections. A line following technique is used for correcting mechanic errors generated in each straight section when the vehicle is moving from one node to another consecutively.

In Section 6.4, we will propose an object security monitoring process. First, the vehicle moves forward to a node where there is a monitored object nearby, and the object should be detected according to the previous learned data in the learning stage. Next, a rotation angle is computed such that the vehicle can turn accordingly toward the monitored object. Also, by using the improved snake algorithm, the vehicle detects an object by using the improved snake algorithm. Finally, a recognition process is conducted to compare the images with the learned ones to decide whether the original object exists or not.

In Section 6.5, we will describe the proposed door situation recognition process. The door situation will be checked to see if it is opened or closed when the vehicle moves automatically to a suitable position recorded in the learning process.

In Section 6.6, we will describe a vehicle coordinate correction method. Since the vehicle might navigate a long distance in the original path gradually, the method is basically based on a vision-based technique which is proposed for reducing the accumulative mechanical errors for precise path navigation. When the vehicle detects a previously learned object from the image, our method will use this information to correct position errors.

6.2 Navigation Process

In the security patrolling process, the vehicle navigates along the generated path by visiting each path node consecutively through the routes specified by the node edges and checks the existence of the learned objects. A simple security patrolling process is described in the following algorithm.

Algorithm 6.1. *Security patrolling.*

Input: The set of nodes $\boldsymbol{N_{path}}$, object data $\boldsymbol{LearnO_{object}}$, and doors data $\boldsymbol{LearnD_{door}}$.

Output: Navigation process.

Steps:

Step 1. The vehicle starts navigating from a starting node N_0.

Step 2. Scan the node list $\boldsymbol{N_{path}}$ to read the next node data.

Step 3. Perform the line following process until the vehicle arrives at the next node.

Step 4. Check whether the vehicle has to monitor objects or doors.

Step 5. If there exists a monitored object O_j or a door D_k in the current node, take the following action; else, continue the remaining navigation.

Step 5.1. If the learned data are object data, do the following steps.

Step 5.1.1. Turn toward to the learned object according the coordinates recorded in the learning process.

Step 5.1.2. Do the object matching process.

Step 5.2. If the learned data is a door data, recognize the door situation.

Step 6. Read the next node data. If there exists the remaining nodes, repeat step 3 to step 5. Else, finish the navigation.

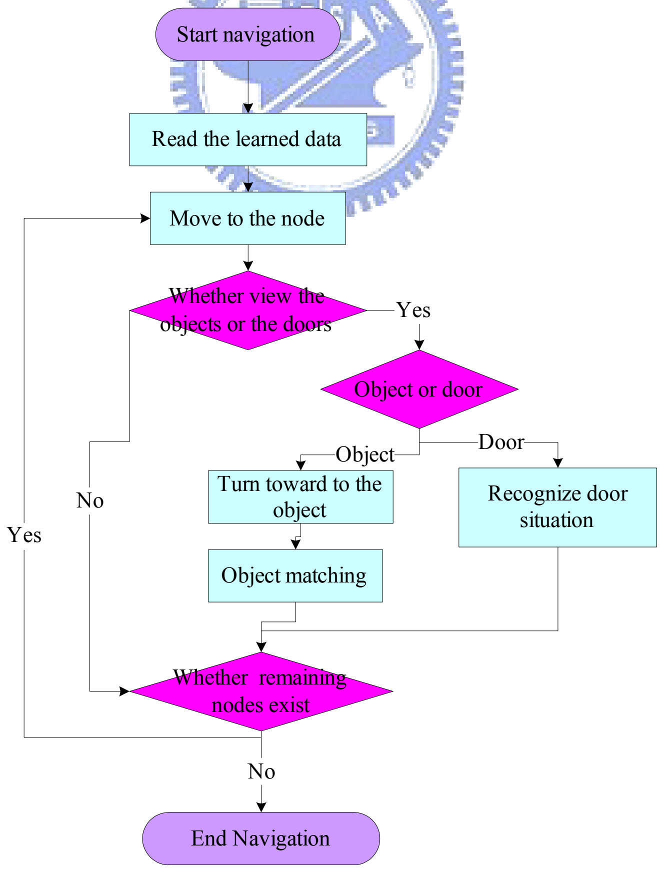


Figure 6.1 Flowchart of security patrolling process

6.3 Navigation Strategy for Straight-Line Sections

The navigation strategy of line following is adopted to reduce deviations from the route when the vehicle passes a node in its navigation path. The line following process will ensure that the vehicle passes each node during crossing two adjacent nodes. The details are described as an algorithm as follows and a related figure is shown in Figure 6.2.

Algorithm 6.2. *Line following navigation.*

Input: Coordinates $L(x_{\text{odo}},y_{\text{odo}})$ and direction angle θ_0 of the vehicle provided by the odometer, next node $N_i(x_{i+1},y_{i+1})$, and a unit vector $\vec{i}=[1,0]$.

Output: A navigation path between two adjacent nodes.

Steps:

step 1. Compute a vector $\vec{V_i}$ by using the following equation.

$$\vec{V_i}=\begin{bmatrix}X_i\\Y_i\end{bmatrix}=\begin{bmatrix}x_{i+1}-x_{odo}\\y_{i+1}-y_{odo}\end{bmatrix}. \tag{6.1}$$

step 2. Compute the direction angle θ_1 of the vehicle in the GCS after the vehicle turns toward the node N_{i+1} by using the following equations:

$$\theta_1=\begin{cases}\cos^{-1}(\frac{\vec{V_i}\bullet\vec{i}}{|\vec{V_i}||\vec{i}|}), & \text{if } Y_i\geq 0;\\ (-1)\times\cos^{-1}(\frac{\vec{V_i}\bullet\vec{i}}{|\vec{V_i}||\vec{i}|}), & \text{if } Y_i<0.\end{cases} \tag{6.2}$$

step 3. Compute the rotation angle as $\alpha = \theta_1 - \theta_0$ and the navigation distance as $d=|\vec{V_i}|$.

step 4. Because of the mechanic error, the vehicle can not move to the correct position as shown in Figure 6.2. A correction angle β is computed as follows by using the curve equation $y = ax^2 + bx + c$ as illustrated in Section 3.3:

$$\beta = \tan^{-1}\left(\frac{a\left|\overline{V_i}\right|^2 + b\left|\overline{V_i}\right| + c}{\left|\overline{V_i}\right|}\right). \qquad (6.3)$$

step 5. Compute the real rotation angle as $r = \alpha - \beta$.

step 6. Turn the vehicle leftward for the angle of r if r is larger than zero; otherwise, turn the vehicle rightward for the angle of r. Therefore, the direction of the vehicle become $\theta_0 - \gamma$.

step 7. Move the vehicle forward. Read the odometer to obtain the current vehicle location L_v and compute how far the vehicle has moved as $d_1 = |L_\text{v} - L|$.

step 8. End this navigation session if $d_1 \geq d$.

Theoretically, there are two main parameters we have to compute in the navigation session, namely, the rotation angle γ and the navigation distance d between two adjacent nodes. By using the curve built in advance which is mentioned in Chapter 3, we compute the real distance d and angle r. Although the vehicle can not move straightly due to the mechanic error, it can still arrive at the next position accurately by adjusting the direction of starting for compensation.

As shown in Figure 6.2, the starting position of the vehicle should be N_i in theory, but the vehicle will not stop at node N_i in last line following navigation from N_{i-1} to N_i due to the condition of ending a navigation session. The real position of the vehicle became L. Hence, we can compute parameters of a navigation session by using L provided by the odometer instead of N_i.

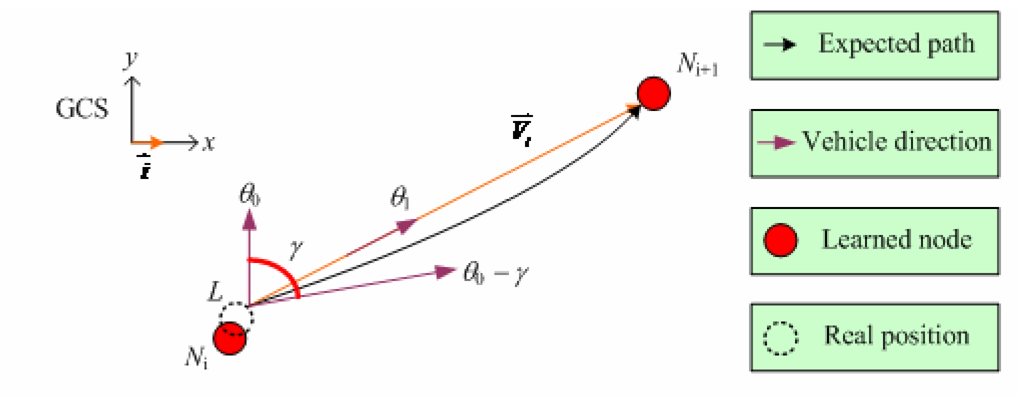


Figure 6.2 Line following navigation.

6.4 Object Security Monitoring Process

In this Section, we will describe the detailed object monitoring process. In the process, the vehicle has to search for the monitored object or the door automatically which was pre-selected by the user in the security patrolling. Then it checks the existence of the objects or the door situation. A new parameter *NodeNumber* is created which is stored in each object data for specifying the position of the object searching. Another problem to be addressed is that when the vehicle moves to the neighborhood of a monitored object, there are some unmonitored objects near the vehicle. How to distinguish a monitored object from them is a problem we have to deal with. Here, we use data of the learned objects to distinguish other objects. In Section 6.4.1, we will propose an object detection method. And in Section 6.4.2, an object matching method will be proposed.

Although the vehicle can detect the learned object according the object coordinates recorded in the learning process, in practice, the navigation deviation often causes the vehicle to be unable to search an object at once. The vehicle moves to the neighbor of the object and detects it but it can not detect an object if the object is not in the image. The last method in Section 6.4 is to decide the searching angle of the object due to the navigation deviation. Also in this section, we will describe the entire object monitoring process.

6.4.1 Proposed Monitored Object Detection Method

When the vehicle moves to the neighborhood of the object O_i, the data of O_i, $LearnO_i$ = {$LearnCIC_{obj,i}$, $LearnC_{obj,i}$, $LearnS_{obj,i}$, $LearnGC_{obj,i}$, $LearnC_{floor,i}$, $NodeNumber$} which is illustrated in the learning process is used to detect the object O_i. The detailed algorithm is described as follows.

Algorithm 6.3. *Monitored object detection.*

Input: A color image I, coordinates $L(x_{\text{odo}}, y_{\text{odo}})$, and direction angle θ_0 provided by the odometer, and a learned object data $LearnO_i$.

Output: A set of object region $ObjR$.

Steps:

step 1. Use coordinates of the object, $LearnGC_{obj,i}$ = { $Learn\_x_{obj,i}$, $Learn\_y_{obj,i}$}, and the coordinates of the vehicle L to compute rotation angle α by the following steps, as shown in Figure 6.3.

step 1.1. Compute a vector $\vec{V}$ by the following equation:

$$\vec{V} = \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} Learn\_x_{obj,i} - x_{\text{odo}} \\ Learn\_y_{obj,i} - y_{\text{odo}} \end{bmatrix}. \qquad (6.4)$$

step 1.2. Compute a direction angle θ_1 between the vector $\vec{V}$ and the unit vector of GCS $\vec{i}$ by the following equation:

$$\theta_1 \begin{cases} \cos^{-1}\left(\frac{\vec{V}\bullet\vec{i}}{|\vec{V}||\vec{i}|}\right), & \text{if } Y \geq 0. \\ (-1)\times\cos^{-1}\left(\frac{\vec{V}\bullet\vec{i}}{|\vec{V}||\vec{i}|}\right), & \text{if } Y < 0. \end{cases} \tag{6.5}$$

step 1.3. Compute the rotation angle α as $\alpha = \theta_1 - \theta_0$.

step 2. Turn the vehicle leftward for the angle α if α is larger than zero; otherwise, turn the vehicle rightward for the angle α.

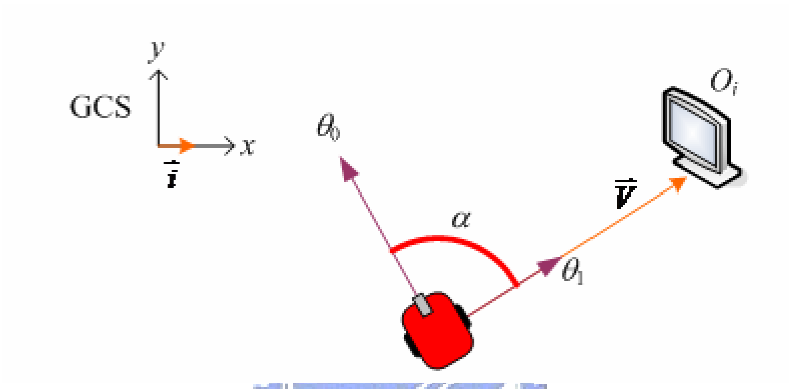


Figure 6.3 Illustration of turn angle computing.

step 3. Capture an image and decide the coordinates of the control points in the ICS of the snake algorithm by the following steps.

step 3.1. Decide four endpoints of a rectangle in the ICS in the following way, as shown in Figure 6.4.

$$\begin{aligned} &I_0(u_0,v_0) = ((-1.5) \times Learn\_a_{obj,i},\ Learn\_v_{obj\_centerj,i} + 1.5 \times Learn\_b_{obj,i}) \\ &I_1(u_1,v_1) = (1.5 \times Learn\_a_{obj,i},\ Learn\_v_{obj\_centerj,i} + 1.5 \times Learn\_a_{obj,i}) \\ &I_2(u_2,v_2) = ((-1.5) \times Learn\_a_{obj,i},\ Learn\_v_{obj\_centerj,i} - 1.5 \times Learn\_a_{obj,i}) \\ &I_3(u_3,v_3) = (1.5 \times Learn\_a_{obj,i},\ Learn\_v_{obj\_centerj,i} - 1.5 \times Learn\_a_{obj,i}) \end{aligned} \tag{6.6}$$

step 3.2. Execute the improved snake algorithm by using the four endpoints to detect an object.

step 4. Get the detected object pixels *ObjR* by using the coordinates of the control points in the ICS, as shown in Figure 6.4.

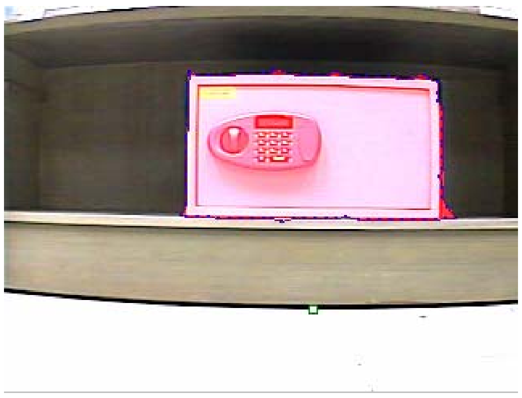


Figure 6.4 An experimental result of object region.

In this process, the vehicle turns its head forward to the front of the object. If no mechanic error occurs, the center of the object region is located on a vertical line which is at the image center. It is shown in Figure 6.5.

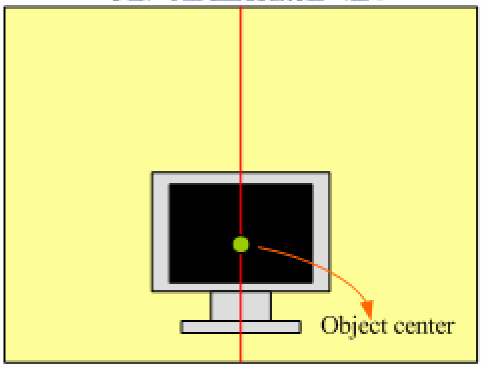


Figure 6.5 An ideal experimental result of the vehicle turning to the object.

6.4.2 Proposed Object Matching Method

After finishing the object detection process as illustrated in the last section, a matching rule is proposed to determine whether the object is exact the same as the previous learned one. A detailed matching algorithm is described as follows.

Algorithm 6.4. *Object Matching Process.*

Input: An image I and a learned object data O_i.

Output: A boolean value, true or false.

Steps:

Step 1. Perform the monitored object detection process.

Step 2. If the object cannot be detected, as shown in Figure 6.6, end this process and return false.

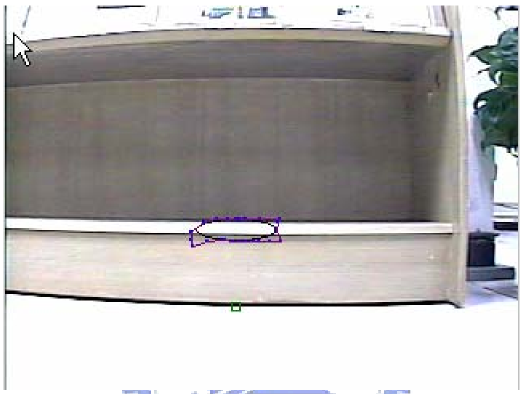


Figure 6.6 An experimental result of detecting no object.

Step 3. Else, detect an object Obj, and compute the feature data which are denoted as $ObjData = \{ C_{obj}, S_{obj}, GC_{obj}, C_{floor}\}$. All features are illustrated as follows.

(1) Color data C_{obj}.

$$C_{obj} = \{\overline{R}_{obj}, \overline{G}_{obj}, \overline{B}_{obj}, R^{sd}_{obj}, G^{sd}_{obj}, B^{sd}_{obj}\}. \tag{6.7}$$

where $\overline{R}_{obj}, \overline{G}_{obj}$, and $\overline{B}_{obj}$ are the means of the R, G, and B values respectively. R^{sd}_{obj}, G^{sd}_{obj}, and B^{sd}_{obj} are the standard deviations of the R, G, B values, respectively.

(2) Shape data S_{obj}.

$$S_{obj} = \{a_{obj}, b_{obj}\}. \tag{6.8}$$

a_{obj} and b_{obj} are the horizontal axis and vertical axis of the ellipse to represent the shape of the object.

(3) Global coordinate GC_{obj}.

$$GC_{obj} = \{x_{obj}, y_{obj}\}. \qquad (6.9)$$

x_{obj} and y_{obj} are coordinates of the object in the GCS.

(4) Floor color C_{floor}.

$$C_{floor} = \{\overline{R}_{floor}, \overline{G}_{floor}, \overline{B}_{floor}, R^{sd}_{floor}, G^{sd}_{floor}, B^{sd}_{floor}\} \qquad (6.10)$$

$\overline{R}_{floor}$, $\overline{G}_{floor}$, and $\overline{B}_{floor}$ are the means of R, G, and B values respectively. R^{sd}_{floor}, G^{sd}_{floor}, and B^{sd}_{floor} are the standard deviations of R, G, B values respectively.

Step 4. Compare the learned object with this object by using the color data in the following methods.

step 4.1. Compare the means of color data in the following way.

step 4.1.1. First compute light difference by using the floor color.

$$R_{ld} = Learn\overline{R}_{floor,i} - \overline{R}_{floor} \qquad (6.11)$$

$$G_{ld} = Learn\overline{G}_{floor,i} - \overline{G}_{floor} \qquad (6.12)$$

$$B_{ld} = Learn\overline{B}_{floor,i} - \overline{B}_{floor} \qquad (6.13)$$

R_{ld}, G_{ld}, and B_{ld} are the differences of the values R, G, and B.

step 4.1.2. Compare these means by using three inequalities below.

$$\left|\overline{R}_{obj} + \lambda \times R_{ld} - Learn\overline{R}_{obj,i}\right| \leq Ct_1 \qquad (6.14)$$

$$\left|\overline{G}_{obj} + \lambda \times G_{ld} - Learn\overline{G}_{obj,i}\right| \leq Ct_1 \qquad (6.15)$$

$$\left|\overline{B}_{obj} + \lambda \times B_{ld} - Learn\overline{B}_{obj,i}\right| \leq Ct_1 \qquad (6.16)$$

where the parameter λ is a coefficient and Ct_1 is a threshold.

step 4.2. Compare the standard deviations by using the following three inequalities:

$$\left|R_{obj}^{sd} - LearnR_{obj,i}^{sd}\right| \leq Ct_2\,, \tag{6.17}$$

$$\left|G_{obj}^{sd} - LearnG_{obj,i}^{sd}\right| \leq Ct_2\,, \tag{6.18}$$

$$\left|B_{obj}^{sd} - LearnB_{obj,i}^{sd}\right| \leq Ct_2\,. \tag{6.19}$$

where the parameter Ct_2 is a threshold.

Step 5. Compare the learned object with this object by using the shape data in the following ways.

step 5.1. Compute the ratio of the horizontal axis and vertical axis of the learned object and this object by using following equations.

$$LearnR = (Learn\_a_{obj,i} \,/\, Learn\_b_{obj,i}) \tag{6.20}$$

$$R = (a_{obj} \,/\, b_{obj}) \tag{6.21}$$

Compare ratios in the following inequality.

$$|\,(LearnR \,/\, \mathrm{R}) - 1| \leq St \tag{6.22}$$

The parameter *St* is a threshold.

Step 6. If the above inequalities are satisfied in Steps 4 step 5, then return true, else return false.

6.4.3 Detailed Object Monitoring Algorithm

Although the vehicle can detect the object by using the learned object coordinates and the vehicle location in the object detection process as described in Section 6.4.1, there are some situations in which the vehicle can not detect the monitored object. The reason is that sometimes mechanic errors cause the vehicle moveing far away from the route such that it can not view the monitored objects, as shown in Figure 6.7. Therefore, a complete object monitoring process including searching view adjustment for improving the robustness is described in the following algorithm and a detailed flowchart is shown in Figure 6.8

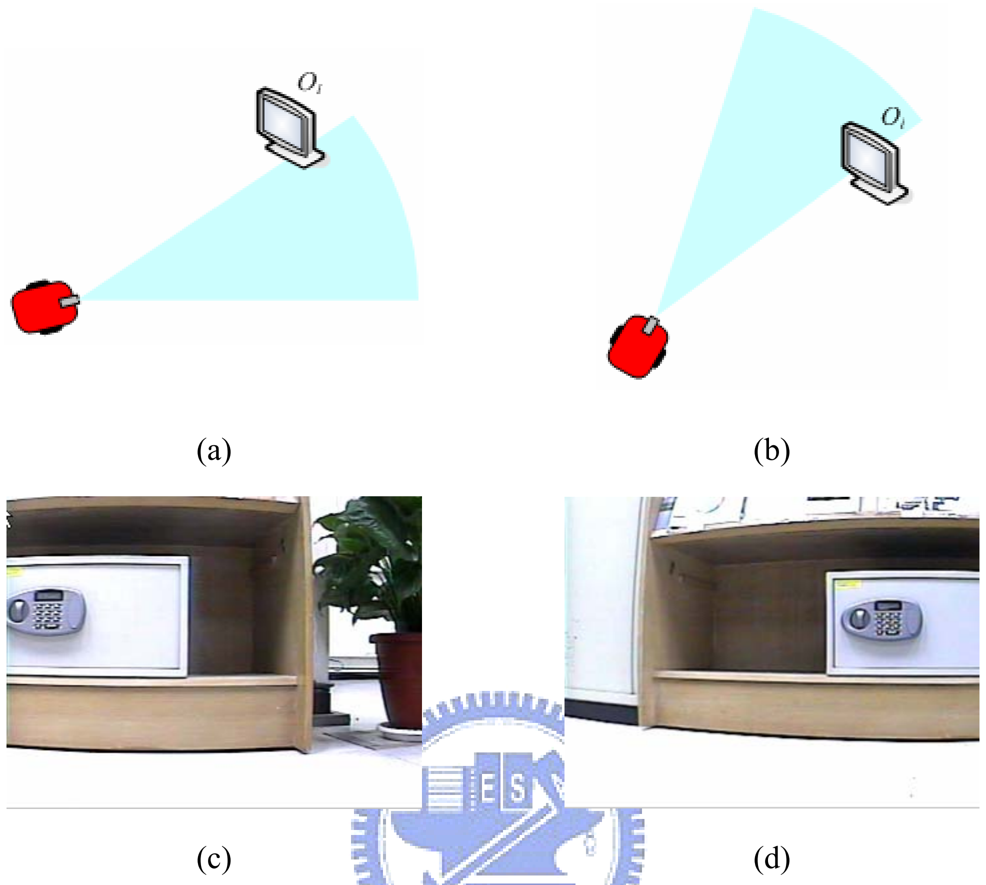


Figure 6.7 Situations of object detection.

Algorithm 6.5. *Monitored object monitoring process*.

Input: A learned object data *LearnO*$_i$, coordinates $L(x_{\text{odo}},y_{\text{odo}})$, and direction angle θ_0 provided by the odometer.

Output: A warning message or nothing.

Steps:

step 1. Perform the monitored object detection process.

step 2. Perform the object matching process.

step 3. If the return value is true, then end this process. Else, continue the following steps.

step 4. Compute a rotation angle Φ by the following equation;

$$\Phi = \tan^{-1}\left(\frac{|\overline{V}|}{10}\right). \tag{6.23}$$

The symbol $|\overline{V}|$ is the distance between the object and the vehicle as illustrated in Algorithm 6.3.

step 5. Turn the vehicle leftward for the angle Φ.

step 6. Repeat Step 2.

step 7. If the return value is still false, continue the following steps. Else, end this process

step 8. Turn the vehicle rightward for the angle 2Φ.

step 9. Repeat Step 2.

step 10. If the return value is still false, a warning is announced. Else, end this process.

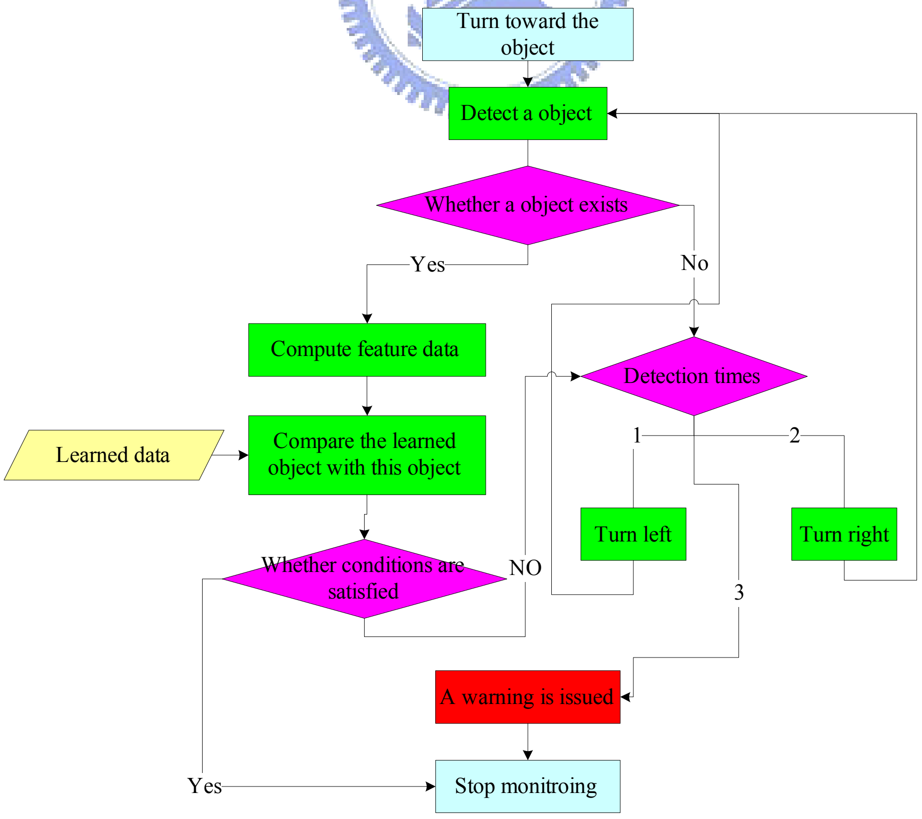


Figure 6.8 Flowchart of object matching.

The main goal of turning the vehicle is to search an object. When the vehicle cannot discover the desired object, it must turn left or turn right for the angle Φ to search the object again. The angle Φ is extracted according to the distance between the vehicle and the learned object. After finishing the searches in the three directions, if the vehicle still can not find the desired object, then a warning message is announced.

6.5 Detection of Door Opening

In this section, the detection algorithm for the determination of the current situation of a door is proposed. The algorithm is performed after the vehicle moving itself to the front of a door, and it is conducted to detect if the door is open or not. The detection process is illustrated as following algorithm.

Algorithm 6.6. *Detection of door opening*.

Input: An image I and a set data of the door D_i.

Output: A Boolean value, true or false.

Steps:

step 1. Detect the edges of I by applying the *Sobel operator*.

step 2. Detect the edge of the door E_d and the edge of the baseline E_b. Choose the right or left side baseline of the door according to learned data *baseline*.

step 3. Compute the slopes of the E_d and E_b by the following equation:

$$a = \frac{\sum_e u \sum_e v - n \sum_e uv}{(\sum_e x)^2 - n \sum_e x^2}. \tag{6.24}$$

step 4. Compare a_d and a_b by the following inequity:

$$|a_d - a_b| \leq th. \tag{6.25}$$

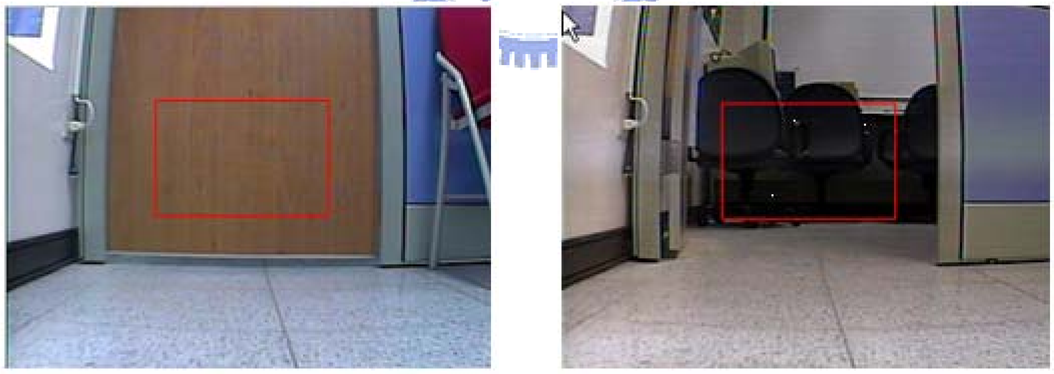
where *th* is a threshold.

step 5. If it is not satisfactory, it means the door is open, and then return true. Else continue the following steps.

step 6. Compute color data by selecting a rectangular region, as shown in Figure 6.9.

step 7. Compare the color data. If it is satisfactory, then return false, else return true.

The main idea here is to utilize the edges of the door and the baseline. We detect the edges of the door's downside and baseline. We compute their slopes by using a line fitting technique. If the door is closed, the edge of the door's downside is parallel to the baselines. Hence, the two slopes should be closer. But when the door is open completely, as shown in Figure 6.9(b), we can not detect the edge of the door. Hence, we utilize the color conditions to decide whether the door is open or not.



(a) (b)

Figure 6.9 An illustration of door detection. (a) The door is opened. (b) The door is closed.

6.6 Improved Guidance Precision in Navigation by Learned Object Location

Although the line following technique and the mechanic error correction method are helpful for the improvement of the navigation accuracy, it is still possible that the vehicle moves for a great deviation from the normal path after a long distance of security patrol. A vision-based technique is proposed to correct the navigation deviation in this section. The main idea of correcting the navigation deviation is to correct the position coordinates of the vehicle. Utilizing the known position of the monitored object which was mentioned in Chatper5, we can correct the coordinates of the vehicle in the GCS.

When the vehicle searches for the object, an ideal situation is that the vehicle turns directly to the front of the vehicle in the object detection process, as illustrated in Section 6.4.1. However, if the vehicle has turned an extra angle Φ leftward or rightward to search for the object, as illustrated in Section 6.4.3, the angle Φ has to be considered during the correction of the coordinates of the vehicle.

Algorithm 6.7. *Improved guidance precision in navigation by learned object location.*

Input: A detected object region *Obj*, the coordinates of the learned object $LearnGC_{obj,i} = \{Learn\_x_{obj,i}, Learn\_y_{obj,i}\}$, the direction angle θ_0 from the odometer, and a turn angle Φ illustrated in Algorithm 6.5 in Section 6.4.3.

Output: A set of coordinates (x_v, y_v) and direction angle θ_1.

Steps:

step 1. Compute the coordinates of the detected object $(\mathrm{V}x_{\mathrm{obj}}, \mathrm{V}y_{\mathrm{obj}})$ in the VCS now by using region *Obj* when the vehicle turns for the angle Φ to detect the object.

step 2. Compute the coordinates of the detected object ($\mathrm{V}x_{\mathrm{o}}$, $\mathrm{V}y_{\mathrm{o}}$) in the VCS when the vehicle does not turn leftward or rightward for the angle Φ by the following equations:

$$\mathrm{V}x_{\mathrm{o}} = \mathrm{V}x_{\mathrm{obj}} \times \sin\Phi - \mathrm{V}y_{\mathrm{obj}} \times \cos\Phi; \tag{6.26}$$

$$\mathrm{V}y_{\mathrm{o}} = \mathrm{V}x_{\mathrm{obj}} \times \cos\Phi + \mathrm{V}y_{\mathrm{obj}} \times \sin\Phi. \tag{6.27}$$

step 3. Compute the coordinates of the vehicle ($\mathrm{O}x_{\mathrm{v}}$, $\mathrm{O}y_{\mathrm{v}}$) in the object coordinate system by transforming the coordinates ($\mathrm{V}x_{\mathrm{o}}$,$\mathrm{V}y_{\mathrm{o}}$) by the following equation:

$$(\mathrm{O}x_{\mathrm{v}}, \mathrm{O}y_{\mathrm{v}}) = ((-1) \times \mathrm{V}x_{\mathrm{o}}, (-1) \times \mathrm{V}y_{\mathrm{o}}). \tag{6.28}$$

step 4. Correct the current direction angle of the vehicle by the following equation:

$$\theta_1 = \theta_0 - \tan^{-1}\left(\frac{\mathrm{V}y_{\mathrm{o}}}{\mathrm{V}x_{\mathrm{o}}}\right) - \frac{\pi}{2}. \tag{6.29}$$

Assume that θ_1 is the new value of the direction angle.

step 5. Compute the angle ρ between the object coordinate system and the global coordinate system by using the direction angle θ_1 by the following equation:

$$\rho = \theta_1 - \pi/2 - \Phi. \tag{6.30}$$

step 6. Correct the coordinates of the vehicle in the GCS by using the learned object data by the following equations:

$$x_{\mathrm{v}} = \mathrm{O}x_{\mathrm{v}} \times \sin\rho - \mathrm{O}y_{\mathrm{v}} \times \cos\rho + Learn\_x_{obj,i}; \tag{6.31}$$

$$y_{\mathrm{v}} = \mathrm{O}x_{\mathrm{v}} \times \cos\rho - \mathrm{O}y_{\mathrm{v}} \times \sin\rho + Learn\_y_{obj,i}. \tag{6.32}$$

step 7. The vehicle navigates according to navigation strategy for straight-line section.

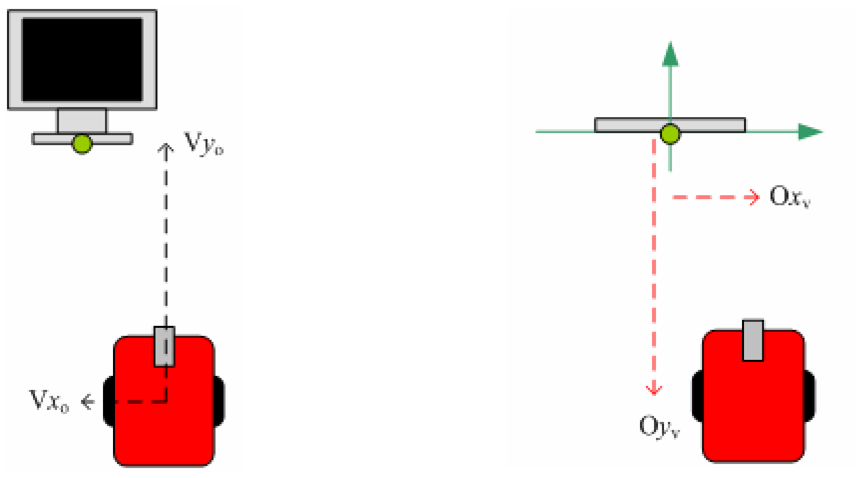
The main idea of correcting the vehicle position is to utilize the recognized monitored object GCS coordinates to modify the odometer values of the vehicle. When the vehicle detects an object and considers it as a learned object, we can use its

GCS coordinates. From the image; we can compute its VCS coordinates. The origin of the VCS is the center of the vehicle and the coordinates of the object, Vx_{obj} and Vy_{obj} are the distances relative to the vehicle, as shown in Figure 6.10(a). Since we transform the VCS to the coordinate system whose origin is the object center, we can get the vehicle coordinates Ox_{v} and Oy_{v}, as shown in Figure 6.10(b). Then using the coordinates of the learned object, we can compute the vehicle position in the GCS.

The direction angle and coordinates are considered. In order to correct the angle, we use the object detection process. When the vehicle turns to the front of the vehicle in the object detection process, the object should appear in the image center if no mechanic error occurs. However, if the error does occur, we correct the direction angle of the vehicle by using the position of the object appearing in the image. The real coordinates of the vehicle in the GCS can be obtained from our method by using the Ox_{v} and Oy_{v} and the direction angle.

As soon as we have corrected the direction and coordinates of the vehicle, the vehicle can not return the original node position, as node N_i shown in Figure 6.11. The real position of the vehicle is L. We use the real position L of the vehicle and next node N_{i+1} to compute the navigation path by using the line following technique mentioned in Section 6.3, as blue path shown in Figure 6.11. Hence, although the vehicle may not arrive at original position Ni due to navigation deviation, we use the object to correct the vehicle position and vehicle can continue navigating in the precise path without returning original position N_i. It is helpful for improving efficiency and precision of navigation.

Also, there is another situation we must consider. Although the vehicle turns toward to the object, the vehicle will turn another angle Φ additionally if the vehicle cannot detect an object. Hence, Φ is included in each correction computation.



(a) (b)

Figure 6.10 Coordinate Transform between the VCS and the object coordinate system. (a) A sidelong view of the VCS. (b) A vertical view of the object coordinates.

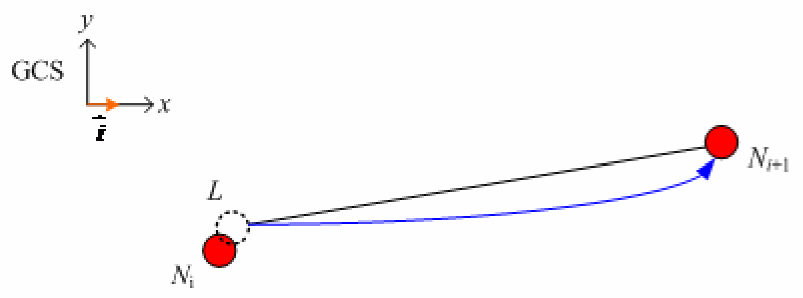


Figure 6.11 An illustration of navigation after correcting the position of the vehicle.

Chapter 7
Experimental Results And Discussions

7.1 Experimental Results

We will show some experimental results of the proposed security patrolling system in this section. The user interface of the system is shown in Figure 7.1.

At first, a user controls the vehicle to learn a path and some monitored objects and doors, as shown in Figure 7.2. The experimental images are shown in the remote system. Figure 7.2(a) illustrates the results of the learning process in which the user chose an arbitrary object to be monitored by selecting its area from the image. Figure 7.2(b) shows a door to be monitored, which has been selected by a user.

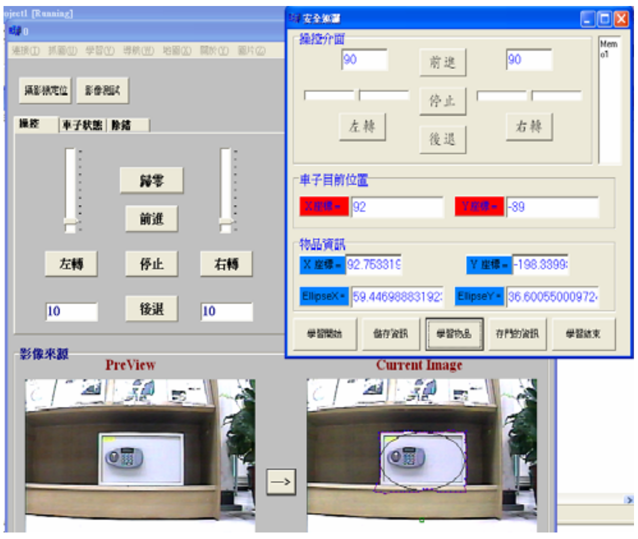


Figure 7.1 An interface of the experiment.

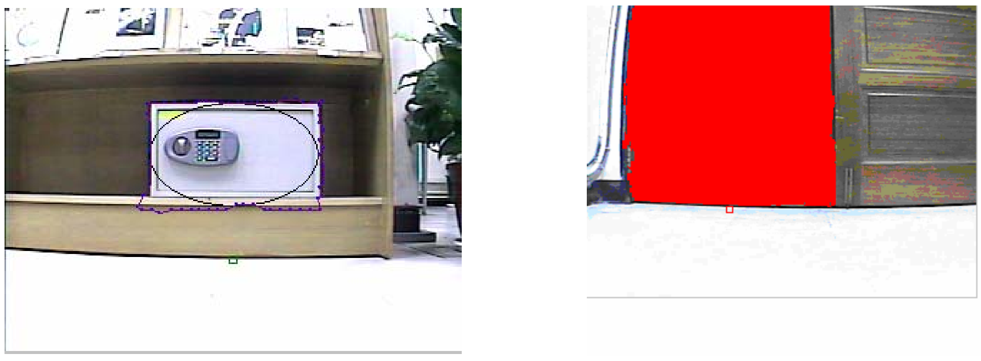

(a) (b)

Figure 7.2 The learning images.

The entire learned data is shown in Figure 7.3. It includes the path nodes and saved objects and a door. There are two safes saved in this experimental.

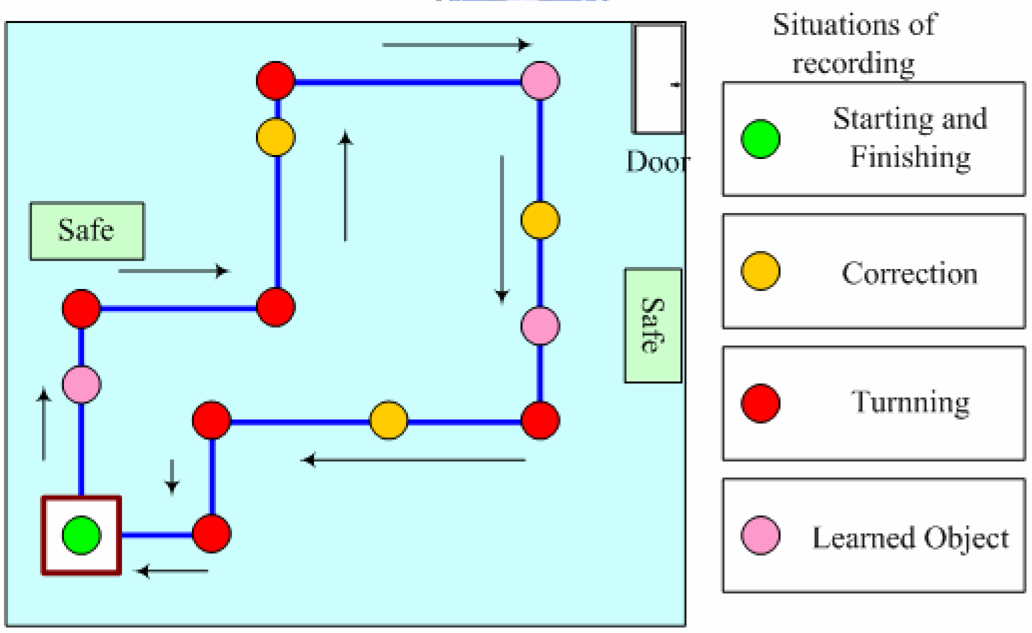


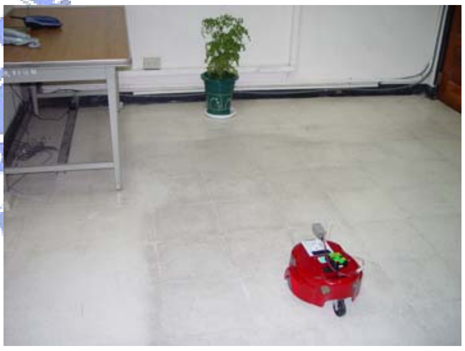
Figure 7.3 An illustration of learned data.

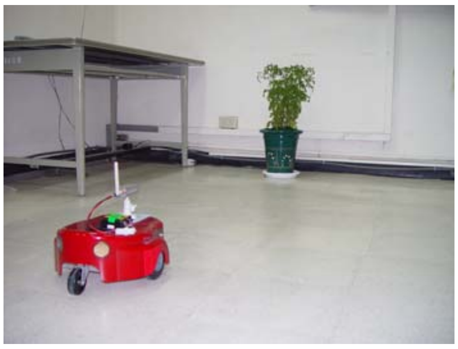
After ending the learning process, the entire vehicle navigation process is shown in Figure 7.4. Some experimental results of monitoring objects and doors are shown in Figure 7.5. There are two regions in the images; the left side is the view of the vehicle, and the right side is the image processing result. Some warning messages of monitoring results are shown in the image. Figure 7.5(a) ~ (d) demonstrate that our system successfully recognizes the existence of monitoring objects. Figure 7.5(e) and (f) include another successful example of our system successfully distinguishing different kinds of door situations.

(a)

(b)

(c)


(d)


(e)

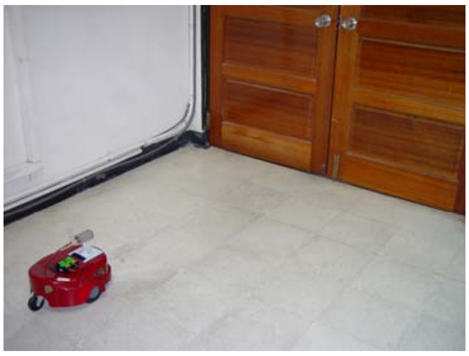
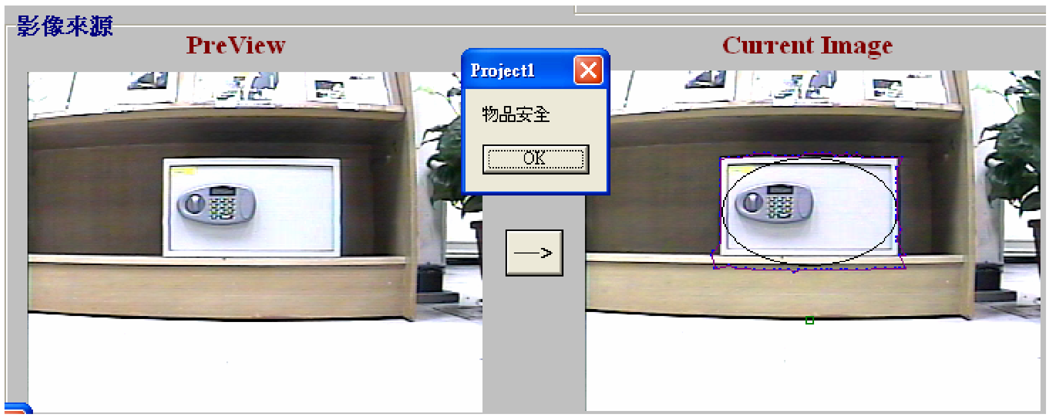
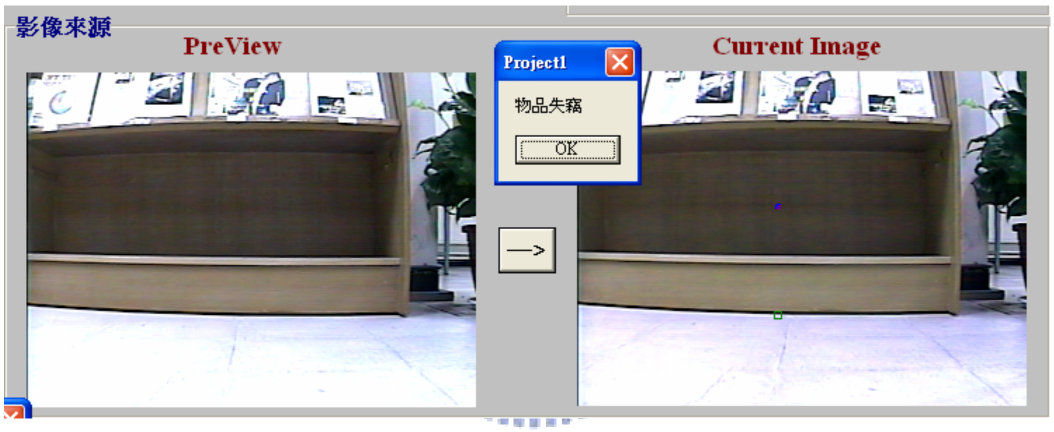

(f)

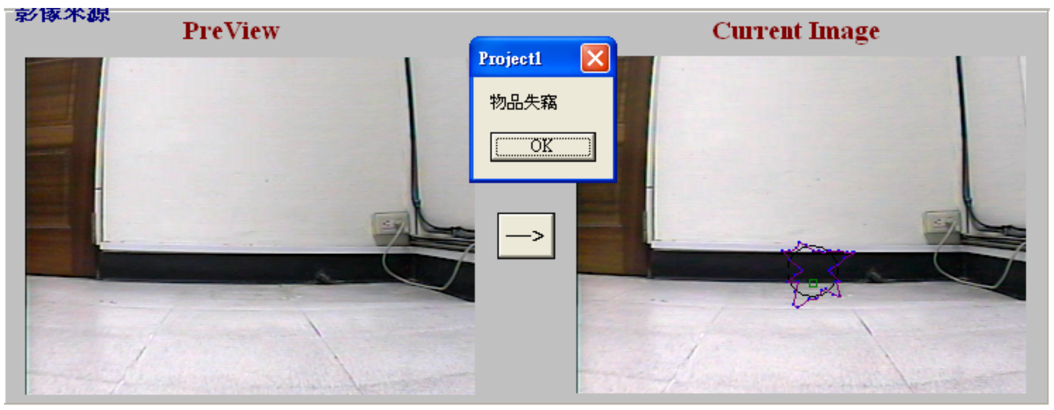
Figure 7.4 A navigation process.



(a)

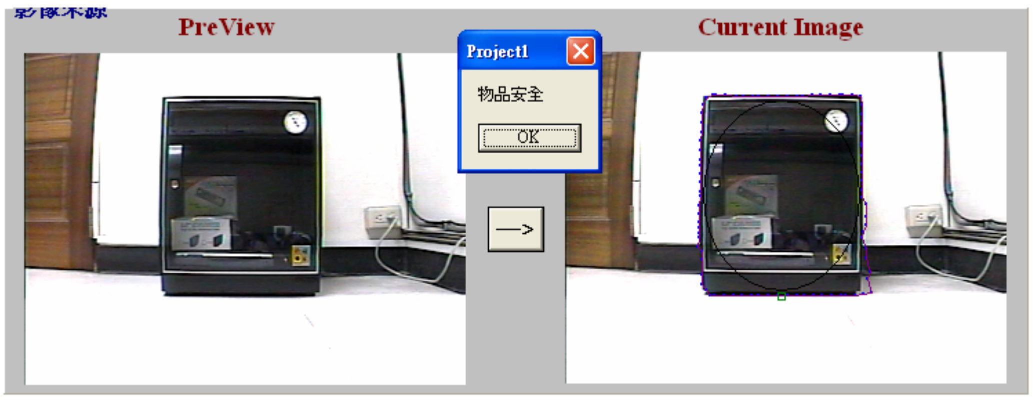


(b)



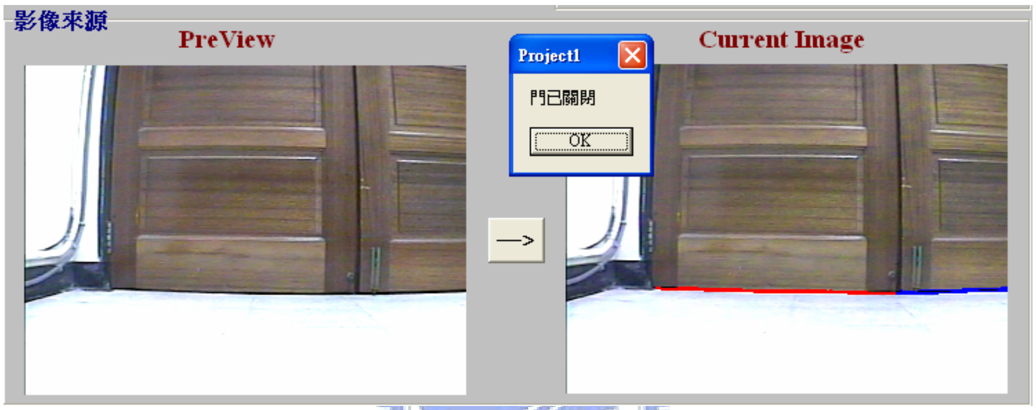
(c)

Figure 7.5 The experimental result of security monitoring.



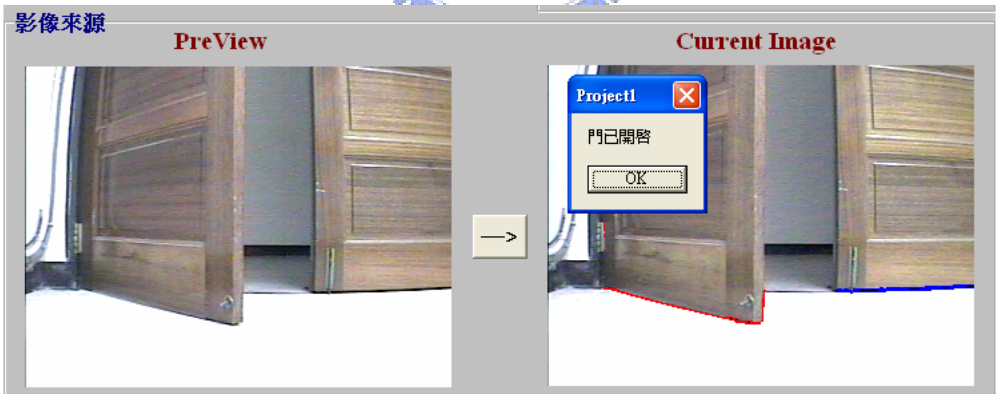
物品安全

(d)



門已關閉

(e)



門已開啓

(f)

Figure 7.5 The experimental result of security monitoring (continued).

7.2 Discussions

By analyzing the experimental results of navigation, some problems are identified as follows.

(1) The result of detecting an object by using the improved snake algorithm might become worse due to the complex background. The control points of the snake will not converge to the edge of the object since the colors of background are too complex. The control point will stop at the edge of background. In the future, the results of the object detection will more satisfactory by improving the snake algorithm.

(2) Object matching is often degraded by the varying lighting condition. Although lighting in indoor environment is more stable than outside, an image still can be affected easily due to the diaphragm of the camera. The vehicle needs to stop more than one second to wait the light steady. Since we use an offsetting technique to overcome it, an erroneous judgment sometimes will occur.

(3) That the floor has to be flat is a constraint of our system. A mechanic error correction model is used in this study but the situation of the vehicle wheel gliding can not be totally overcome. The navigation precision is affected by the roughness of the floor.

Chapter 8
Conclusions and Suggestions for Future Works

8.1 Conclusions

Several techniques and strategies have been proposed in this study and integrated into an autonomous vehicle system for security patrolling in the indoor environments with mechanic error correction and visual object monitoring capabilities.

At first, a setup strategy for the autonomous vehicle is proposed. Two kinds of tasks, namely, Location mapping calibration and mechanic error correction, have been proposed to set up the vehicle before its patrolling. Feasible 2D Location mapping calibration is proposed for acquiring the relative positions between the vehicle and the surrounding environment precisely. The mechanic error correction model which is based on a second-order curve equation is proposed to improve navigation accuracy.

Next, some learning strategies are proposed for the autonomous vehicle, including learning of the planned path and learning of monitoring objects and doors. The user can easily control the vehicle to navigate in the environment and select monitored objects in the image. And in order to make a precise navigation along a path, one method is to use the coordinates of learned objects as an auxiliary tool to adjust the position and direction of the vehicle. Another method is based on a line following technique. Both ways have been implemented in this study.

In addition, a computer vision process has been proposed for security monitoring in the navigation path. Several processes, namely, object detection, object recognition, object searching, and door opening detection, have been proposed to detect the current situation during the patrolling process.

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

8.2 Suggestions for Future Works

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

(1) Improving the object detection method --- In order to detect monitored object with a more complicated image, the object detection method need be improved, which can then be adopted for more application environments.

(2) Adding the capability of object feature extraction --- This is especially useful when the interesting image regions of an object are hollow. For example, the things are a ring, a wheel, or a flowerpot, etc.

(3) Adding the vehicle abilities of obstacle detection and avoidance such that it can navigate in complex and dynamic environments with objects or humans appearing suddenly on the navigation path.

(4) Adding the ability of human detection and tracking during the vehicle navigation.

(5) Adding the ability of conflagration detection in the house.

(6) Designing a friendlier user-machine interface and simplifying the learning strategy for object and path learning.

(7) Designing a camera system with a capability of panning, tilting, and swinging.

(8) Adding the capability of voice control in the learning process.

(9) Adding the capability of transmitting warning messages from the vehicle to the user's cell phone by using telecommunication systems.

References

[1] Michael Kass, Andrew WitKin and Demetri Terzopoulos, “Snake: Active contour models,” *International Journal of Computer Vision*, vol. 1, no. 4, Jan. 1988, pp. 321-331.

[2] P.C. Yuen, Y.Y. Wong, and C.S. Tong, “Contour detection using enhanced snakes algorithm,” *Electronics Letters*, vol. 32, issue 3, Feb. 1, 1996, pp. 202-204.

[3] Donna J. Williams and Mubarak Shah, ”A fast algorithm for active contours,” *Proceedings of Third International Conference on Computer Vision*, Osaka, Japan, Dec. 4-7, 1990 , pp. 14-26.

[4] F. Leymarie and M.D. Levine, "Tracking deformable objects in the plane using an active contour model." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 6, pp. 617-634, 1993.

[5] Dong Joong Kang, “A fast and stable snake algorithm for medical images,” *Pattern Recognition Letters*, vol. 20, issue 5, May 1999, pp. 507-512.

[6] Tianqing Li, Yi Zhang, Danya Yao, and Dongcheng Hu, “FFT Snake: a robust and efficient method for the segmentation of arbitrarily shaped objects in image sequences,” *Proceedings of the 17th International Conference on Pattern Recognition*, vol. 2, Aug. 23-26, 2004, pp. 116-119.

[7] http://members.bellatlantic.net/~vze2vrva/ellipse_fitting.html, Ellipse Fitting.

[8] Maurizio Pilu, Andrew W. Fitzgibbon, and Robot B. Fisher, “Ellipse-specific direct least-square fitting,” *Proceedings of International Conference on Image Processing*, vol. 3, Lausanne, Switzerland, Sept. 16-19, 1996, pp. 599-602.

[9] Chung-Chi Lai, “A Study on Automatic Indoor Navigation Techniques for Vision-Based Mini-Vehicle with Off-Line Environment Learning Capability,”

Master Thesis, Department of Computer and Information Science, National Chiao Tung University, Taiwan, June, 2003.

[10] Pao-Lung Li, "Path Learning, Planning, and Guidance for ALV Navigation Inside Buildings," *Master Thesis, Department of Computer and Information Science, National Chiao Tung University*, Taiwan, June, 1997.

[11] Hsien-Yi Chiu, "Automatic Vehicle Navigation and Parking in Building Corridors Using Panoramic Sensing and 2D Image Analysis Techniques," *Master Thesis, Department of Computer and Information Science, National Chiao Tung University*, Taiwan, June, 2002.

[12] Y. C. Chen and *W. H. Tsai* (2004/08). "Vision-based autonomous vehicle navigation in complicated room environments with collision avoidance capability by simple learning and fuzzy guidance techniques," *Proceedings of 2004 Conference on Computer Vision, Graphics and Image Processing,* Hualien, Taiwan, Republic of China.

[13] Rui-Chih Liu, "Security Patrolling in Building Corridors by Multiple-Camera Computer Vision and Automatic Vehicle Navigation Techniques," *Master Thesis, Department of Computer and Information Science, National Chiao Tung University*, Taiwan, June, 2001.

[14] John B. Fraleigh and Raymond A. Beauregard, "Linear Algebra," third edition, Nov. 1995.

[15] R. C. Gonzalez and R. Z. Woods, "Digital Image Processing," second edition, 2002.