

國立交通大學

資訊科學研究所

碩士論文

利用電腦視覺與泛晰導航技術作室內環境

自動車航行之研究



Autonomous Vehicle Navigation in Indoor Environments  
by Computer Vision and Fuzzy Guidance Techniques

研究生：陳逸傑

指導教授：蔡文祥 博士

中華民國九十三年六月

利用電腦視覺與泛晰導航技術作室內環境自動車航行之研究

Autonomous Vehicle Navigation in Indoor Environments by Computer  
Vision and Fuzzy Guidance Techniques

研究生：陳逸傑

Student : Yi-Chieh Chen

指導教授：蔡文祥

Advisor : Dr. Wen-Hsiang Tsai

國立交通大學  
資訊科學研究所  
碩士論文



Submitted to Institute 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 2004

Hsinchu, Taiwan, Republic of China

中華民國九十三年六月

# 利用電腦視覺與泛晰導航技術作室內環境自動車航行之研究

研究生：陳逸傑

指導教授：蔡文祥博士

國立交通大學資訊科學研究所

## 摘要

本論文提出了一套基於電腦視覺及泛晰控制理論的導航技術，可讓自動車航行於室內環境中，並且具備避碰障礙物的能力。在實驗中我們利用一可無線遙控且擷取影像的迷你自動車作為研究平台。我們設計了一套人性化、有效而且沒有繁瑣規則與限制的學習方法，能自動分析使用者學習時的操作行為，並自動建立導航路線地圖。在使用此方法得到學習路線資料之後，我們也運用一套完整的導航策略來完成學習路線之航行。此策略主要是建構在泛晰控制導航的技術之下，並且可以搭配學習時記錄的資料，完整地走完所有的學習路線。我們最後亦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。

# **Autonomous Vehicle Navigation in Indoor Environments by Computer Vision and Fuzzy Guidance Techniques**

Student : Yi-Chieh Chen

Advisor: Dr. Wen-Hsiang Tsai

Institute of Computer and Information Science  
National Chiao Tung University

## **ABSTRACT**

A vision-based fuzzy-guidance approach to autonomous vehicle navigation in indoor environments with a capability of avoiding obstacles appearing in planned paths is proposed. A small vehicle with wireless control and image grabbing capabilities is used as a test bed. A simple learning strategy is designed for flexible and effective non-visual learning of reachable spots in rooms without rules or restrictions. Through this simple learning strategy, a planned path is obtained by analyzing user-driving commands and odometer data. And following the learned path, the vehicle can accomplish specified navigation sessions by proposed collision-avoidance strategies based on the use of fuzzy-control guidance techniques and input images acquired by an onboard vision system. And some strategies for navigation accuracy maintenance are also proposed. Finally, experimental results showing flexibility of the proposed approach for navigations in indoor environments are also included.

# 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. Yen-Chung Chiu, Mr. Nan-Kun Lo, Mr. Wei-Liang Lin, Mr. Kuei-Li Huang, Mr. Cheng-Jiun Lai, and Miss Yen-Lin Chen 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

ABSTRACT (in Chinese) .....	i
ABSTRACT (in English).....	ii
ACKNOWLEDGEMENTS .....	iii
CONTENTS.....	iv
LIST OF FIGURES .....	vii
LIST OF TABLES .....	ix
<b>Chapter 1 Introduction.....</b>	<b>1</b>
1.1 Motivation.....	1
1.2 Survey of Related Studies .....	2
1.3 Overview of Proposed Approach .....	3
1.4 Contributions.....	6
1.5 Thesis Organization .....	7
<b>Chapter 2 System Configuration and Navigation Principles.....</b>	<b>8</b>
2.1 Introduction.....	8
2.2 System Configuration .....	9
2.2.1 Hardware Configuration .....	10
2.2.2 Software Configuration.....	10
2.3 Proposed Learning Principle and Process.....	11
2.4 Proposed Vehicle Guidance Principle and Process.....	13
<b>Chapter 3 Proposed Fuzzy Guidance Techniques for Indoor Navigation .....</b>	<b>16</b>
3.1 Introduction.....	16
3.2 Review on Fuzzy Control Concepts .....	17
3.3 Reasons of Using Fuzzy Guidance Techniques .....	19
3.4 Parameters for Proposed Fuzzy Guidance System .....	20
3.4.1 Features for Vehicle Guidance in Rooms.....	21
3.4.2 Features for Vehicle Guidance in Corridors.....	24
3.5 Proposed Fuzzy-Control Systems and Their Uses in Guidance .....	28
3.5.1 Proposed Fuzzy Control System for Room-Type Guidance.....	28
3.5.2 Proposed Fuzzy Control System for Corridor-Type Guidance.....	30

<b>Chapter 4 Simple Learning Strategies for Indoor Navigation by User-Driving and Odometer Parameters .....</b>	<b>33</b>
4.1 Introduction.....	33
4.2 Two Navigation Modes.....	34
4.3 Manual Learning Algorithm .....	35
4.3.1 Proposed Control Interface for manual learning.....	36
4.3.2 Proposed Simple User-Driving Rules for Learning.....	37
4.3.3 Data Structures of Learned Data.....	38
4.4 Automatic Path Map Creation Process of Learned Data .....	41
4.4.1 Learned Data Analysis .....	41
4.4.2 Identification of Nodes for Single Path Mode .....	42
4.4.3 Identification of Nodes for Area Mode.....	45
<b>Chapter 5 Vehicle Navigation in Indoor Environment.....</b>	<b>48</b>
5.1 Introduction.....	48
5.2 Navigation in Single Path Mode .....	49
5.2.1 Navigation Strategy for Straight-Line Sections .....	50
5.2.2 Navigation Strategy for Turning Sections.....	51
5.2.3 Process of Proposed Navigation in Single Path Mode.....	55
5.3 Navigation in Area Mode.....	56
5.3.1 Generation of essential data for path planning .....	56
5.3.2 Dynamic Path Planning using Dijkstra Algorithm.....	60
5.3.3 Navigation Strategy among Room Nodes.....	61
5.3.4 Process of Proposed Navigation in Area Mode .....	63
5.4 Strategy for Fuzzy Navigation Accuracy Maintenance .....	65
5.4.1 Comparison of Navigation Distances .....	65
5.4.2 Comparison of Directions .....	66
5.4.3 Smooth Curve Following Process.....	66
5.4.4 Adjustment of Beginning Position.....	68
<b>Chapter 6 Person Following in Corridors .....</b>	<b>69</b>
6.1 Introduction.....	69
6.2 Image Preprocessing for Unexpected Object Detection in Corridors.....	70
6.3 Human Identification .....	73
6.4 Human Tracking by Feet Positions .....	74
6.5 Process of Person following.....	75
<b>Chapter 7 Experimental Results and Discussions.....</b>	<b>78</b>

7.1 Experimental Results .....	78
7.1.1 Navigation in Single Path Mode.....	78
7.1.2 Navigation in Area Mode.....	79
7.2 Discussions .....	85
<b>Chapter 8 Conclusions and Suggestions for Future Works .....</b>	<b>86</b>
8.1 Conclusions.....	86
8.2 Suggestions for Future Works.....	87
<b>References.....</b>	<b>89</b>





# LIST OF FIGURES

Figure 1.1 : Flowchart of four stages of proposed system.....	6
Figure 2.1 : The Amigo vehicle used in this study.....	9
Figure 2.2 : Structure of proposed system.....	11
Figure 2.3 : Illustration of the proposed learning process.....	13
Figure 2.4 : An illustration of vehicle navigation process.....	15
Figure 3.1 : Illustration of a fuzzy inference system.....	18
Figure 3.2 : An image of an obstacle appearing in navigation path.....	20
Figure 3.3 : Illustration of image processing results (a) An input image. (b) Result of candidate route point classification. (c) Result of route area extraction. (d) Result of guidance line computation. (e) Another input image. (f) Result of guidance line computation of (e).....	24
Figure 3.4 : Illustration of image processing results (a) An input image. (b) Result of binarization process. (c) Result of applying Sobel operator to (b). (d) Result of finding connected components. (e) An another input image. (f) Result of applying binarization process and Sobel operator to (e).....	27
Figure 3.5 : Membership functions for guidance in rooms.....	30
Figure 3.6 : Membership functions for guidance in corridors.....	32
Figure 4.1 : An illustration of the learned user control interface.....	37
Figure 4.2 : An illustration of the learned data.....	40
Figure 4.3 : An example to illustrate a created graph for single path mode.....	44
Figure 4.4 : An example to illustrate created graph for area mode.....	47
Figure 5.1 : An illustration of computing two wheel speeds for the case of turning rightward in curve following.....	53
Figure 5.2 : A figure illustrating curve following process.....	53
Figure 5.3 : A figure illustrating the check process before curve following process (a) A case that the curve following process can be executed (b) A case that the curve following process can not be executed.....	54
Figure 5.4 : A figure to illustrate a process of proposed navigation procedure in single path mode.....	57
Figure 5.5 : An example to illustrate result of performing the proposed algorithm.....	59
Figure 5.6 : An illustrative example to show the direction creation of two turn nodes, where 1 represents node $N^1$ and 2 represents node $N^0$ .....	63
Figure 5.7 : A figure to illustrate proposed navigation process in the area mode.....	64
Figure 5.8 : An example to illustrate the usability of smooth curve following for navigation accuracy maintenance.....	67

Figure 6.1 : Images of several image processing results (a) A corridor image. (b) An unexpected object is appeared. (c) Binary thresholding of (b). (d) Two approximation lines are found. (e) Scan the lines vertically from bottom to the approximation lines. (f) An *Object Foot* is found (yellow points). (g) Another example. (h) Two *Object Foots* are found.....72

Figure 6.2 : An example to illustrate the position of tracking target.....75

Figure 6.3 : A process of person following.....77

Figure 7.1 : An experimental result of navigation in single path mode.....81

Figure 7.2 : An experimental result in area mode (1).....82

Figure 7.3 : An experimental result in area mode (2).....83

Figure 7.4 : An experimental result in area mode (3).....84



# LIST OF TABLES

Table 1 : User-Driving commands and parameters for Learned Data.....40



# Chapter 1

## Introduction

### 1.1 Motivation

Recently, vision-based autonomous vehicles or mobile robots have been used in more and more human environments, especially in security patrolling and home service applications. A difficulty encountered in such vehicle navigation applications is the complicated environment faced by an autonomous vehicle, resulting in a challenge of designing a general and flexible learning strategy for various navigation environments. Facing this challenge, we use a small vehicle with wireless control and image grabbing capabilities as a test bed for the development of indoor navigation in this study.

Most former works focused on *vision-based* learning. Though this approach is useful for constructing visual environment data, which can be used for locating a vehicle in the navigation stage, yet certain restrictions are usually imposed on the learning process, resulting in inconvenience or difficulty in conducting the learning work. Furthermore, certain artificial landmarks or specific scene features are often forced to appear in the environment to be learned, in order to accomplish the work of locating the vehicle by landmark or feature matching in the navigation stage. This often makes the learning method inapplicable or less flexible in certain applications. Finally, landmark or feature matching often yields imprecise vehicle location results, leading possibly to unstable navigation performances. In this study, we propose a

*non-visual* approach to learning which solves all the above-mentioned problems encountered in the conventional vision-based learning approach.

After the learning process is finished, the remaining task is to guide the vehicle to navigate along the learned path safely. That is, two goals must be achieved, one is that the vehicle should traverse along the learned path correctly, and the other is that it should avoid possible obstacles appearing in the path. Therefore, we propose to use fuzzy-control techniques to guide the vehicle and avoid possible collisions in the meanwhile. Navigation accuracy maintenance is also considered. With these capabilities, the vehicle can be used in many applications, such as security patrolling, inter-building transportation, cleaning service among rooms, intelligent toys, and so on.



## **1.2 Survey of Related Studies**

In many applications for vehicle navigation in indoor environments, learning navigation paths is required before a navigation process can be started. Since scenes of indoor environments are often complicated and may consist of several kinds of rooms and corridors, such learning works are usually difficult and heuristic. Design of general learning algorithms for complicated indoor environments is still a difficult research topic. In Davison and Murray [1], some preset landmarks placed manually are used to train a mobile robot or vehicle to recognize its location. After a few times of training, the vehicle can detect the landmarks accurately and follow them to finish the learned path stably. In Hayet, Lerasle, and Devy [2], a vehicle was designed to identify obvious square blocks as landmarks and recorded its color and size as feature

sets in the learning stage. Then the vehicle can check the color and the size of each landmark to locate its position in the navigation stage. In Gaussier, Joulain, and Zrehen [3], a mobile robot can be controlled by visual information retrieved from particular goal landmarks. And their model does not need a precise map nor learn all the possible positions in the environment.

Besides, many visual navigation methods have been developed in recent years. In Li and Tsai [4], a graph-based navigation method was proposed. The vehicle traverses among the nodes in a graph which is composed of learned paths. But certain complex learning rules were adopted to the learning process. In [5, 6], route areas in the navigation path are obtained and the vehicle keeps navigating on the central path of each route area. In Desouza and Kak [7], a survey of visual navigation techniques for mobile robots is given. Some techniques including optical flows, landmarks tracking, simultaneous localization and map-building, and visual feature matching for indoor or outdoor environments were introduced. Cho and Nam [8] proposed a method of vision-based navigation for mobile robots using fuzzy-logic control techniques. They used the wavelet transform to detect the edges of guidelines or obstacles as fuzzy input parameters and found fuzzy output parameters to guide the robot to move forward. And a collision avoidance algorithm was also proposed. Ayanna and Edward [9] proposed a technique using real-time terrain feature analysis for outdoor-environment navigation with a mobile robot. They used terrain features as fuzzy inputs to get crisp fuzzy outputs to move the robot to navigate within terrains safely.

## **1.3 Overview of Proposed Approach**

In this study, we try to design a vision-based vehicle navigation method which uses fuzzy-control guidance techniques in indoor environments. With this purpose, a simple and flexible learning method that processes user-driving parameters and odometer data automatically is proposed. Secondly, a fuzzy-control guidance technique with obstacle avoidance capability is also proposed. Finally, a person following application by tracking human feet positions is proposed.

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

- (1). Configure the setting of the parameters of a small vehicle as a test.
- (2). Design a fuzzy control system for the guidance purpose
- (3). Analyze images grabbed by the vision system of the vehicle to obtain visual parameters as fuzzy input.
- (4). Record user-driving parameters and odometer data during the learning process.
- (5). Process roughly recorded data and transform them into usable data for navigation.
- (6). Guide the vehicle to traverse along learned paths safely.
- (7). Detect human feet in corridors.
- (8). Track human feet to follow their movement.

The proposed approach for vision-based vehicle navigation consists roughly of four stages. The first stage is *Fuzzy-Control Guidance*. Since the principle of fuzzy theory is intuitive and experiential, we utilize the characteristic of fuzzy theory to implement the guidance of navigation. We propose two fuzzy control systems for guiding the vehicle to move forward and avoid possible collisions. The second stage is *path learning for navigation by user control*. In this stage, the vehicle is controlled to move to desired places through a designed user interface. During the path learning process, a sequence of user-control actions, called user-driving parameters, and

odometer data are recorded, respectively. These two types of parameters will be recorded as a temporary list for the processing of the next stage.

The third stage is *creation of learned path map for navigation*. In this stage, the parameters recorded in the previous stage will be automatically processed into a path map by the proposed path map creation process. The path map is in a form of graph, and is composed of a set of nodes connected with inter-node edges. The data of each node in the path map are recorded into a text file orderly. Then the data of this text file is a basis for autonomous navigation in the next stage.

The fourth stage is *vehicle navigation in indoor environments*. In this stage, the vehicle traverses along the learned path node by node orderly in accordance with node data in the text file. Two navigation modes are also proposed for different purposes. One is the *single path* mode for one path navigation from a starting point to an ending point; the other is the *area* mode for navigation among rooms. And two strategies are proposed for a vehicle to navigate in indoor environments. The first is line following, which is useful for the vehicle to navigate in straight-line sections. And the second strategy is smooth curve following, which is used for turning the vehicle leftward of leftward in turning sections. Furthermore, a navigation accuracy maintenance strategy is also proposed.

A possible application of the proposed approach is *person following in corridors*. In this application, an image processing technique is employed for detecting objects in corridors. And a human identification method by a position checking method is proposed. Since the positions of human feet are known, the person following action will be triggered for tracking the positions of human feet.

In summary, an innovative learning strategy with simple and flexible capabilities without using a video camera is proposed. The vehicle can use a path map acquired by the simple learning process to navigate in desired environments. And the proposed



fuzzy-control guidance technique is utilized in autonomous navigation method. Using these hybrid techniques, the vehicle can navigate in any unknown indoor environment if the learning process was finished before.

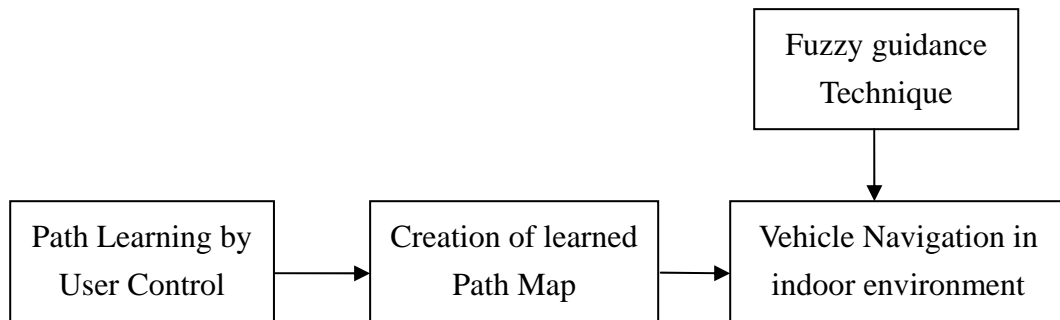


Figure 1.1 Flowchart of four stages of proposed system.

## 1.4 Contributions



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

- (1) A fuzzy guidance technique using 2D visual parameters is proposed
- (2) A simple and flexible learning strategy using user-driving parameters and odometer data is proposed.
- (3) A path map creation method which transforms rough learned data into a path map for navigation is designed.
- (4) A graph-based navigation method using node data in the path map is proposed.
- (5) A curve following method which can reduce accumulative errors for navigation is proposed.

- (6) Two kinds of navigation modes that can be applied in many applications are designed.
- (7) A person following method for tracking the positions of human feet is proposed.

## 1.5 Thesis Organization

The remainder of this thesis is organized as follows. The system configuration of the vehicle used as a test bed and the principles of learning and navigation are described in Chapter 2. The proposed fuzzy-control techniques and the designed fuzzy control system are described in Chapter 3. In Chapter 4, the proposed innovative learning strategy with simple and flexible capabilities, and the proposed path map creation method are described. In Chapter 5, the proposed vehicle navigation method with obstacle avoidance capability, a smooth curve following process for turning sections, and some strategies for navigation accuracy maintenance are described. The application of person following in corridors by tracking the positions of human feet is described. Satisfactory experimental results are shown in Chapter 7. Finally, some conclusions and suggestions for future works are given in Chapter 8.

# Chapter 2

## System Configuration and Navigation Principles

### 2.1 Introduction

Recently, the applications of indoor navigation of mobile robots or autonomous vehicles become more and more popular, especially in home services and security patrolling. Hence, the size of the vehicle was designed to be smaller and smaller in order to achieve the purpose of dexterous movement in indoor environments. With this goal, a small vehicle with wireless control and image grabbing capabilities is used as a test bed for our research and development in this study. Because of the dexterous property of this small vehicle, it can be used in many kinds of applications.

In order to navigate in any unknown indoor environment, a learning process is essential for the vehicle. Therefore, a simple and flexible learning strategy which learns user-driving parameters and odometer data is developed in this study. Through the learning process, desired navigation paths can be learned by the vehicle and useful data are recorded at in the mean time.

After navigation paths are known by the vehicle after the learning process, a fuzzy-control guidance technique and a vehicle navigation method, both deigned in this study, are applied for the vehicle to navigate along the learned paths to finish desired trips.

## 2.2 System Configuration

In this study, we use the Amigo robot, a mini-vehicle made by ActivMedia Robotics Technologies Inc., as shown in Figure 2.1 as the test bed. There are eight ultrasonic sensors, an odometer, and a wireless 2.4G camera on this mobile vehicle. The ultrasonic sensors are disabled in this study because they are useless for our research work. The maximum advance speed and rotation speed of the vehicle are 75 cm/sec and 300 degrees/sec, respectively, and the speed encoders encodes 39000 ticks per wheel revolution with 124 ticks per mm. The length, width, and height with the camera and the body of the Amigo vehicle are 33 cm, 28cm, and 21cm, respectively.



Figure 2.1 The Amigo vehicle used in this study.

## 2.2.1 Hardware Configuration

The proposed system is composed of five parts, as shown in Figure 2.2. The first part is the vehicle with an embedded control system. By a user's commands, this system can control the two DC motors to move forward or backward or turn around, and return some status parameters of the vehicle to the user. The second part is a vision system which consists of a camera with a wireless transceiver, a wireless video signal receiver, and an imaging frame grabber. Because the signal obtained from the camera and sent to the video signal receiver is analog, a still digital image can be obtained by the imaging frame. The image grabbed in our experiments are of the resolution of 320x240 pixels for the reason of raising image processing efficiency. The third part is a power system which consists of two kinds of batteries. One is a 9V battery to supply the power of the wireless camera; and the other is a 12V battery to supply the power of the vehicle system. The fourth part is a remote control system in a desktop computer or a notebook PC. The kernel program can be executed on this remote control system to command the vehicle through a wireless transmission system, which is the fifth part of the proposed system, consisting of a wireless access point and a wireless signal receiver. The command of the remote control system is transmitted to the wireless signal receiver on the vehicle by an access point that meets the 802.11b standard.

## 2.2.2 Software Configuration

In order to transmit commands to the vehicle and to retrieve status data about the vehicle, we use the ARIA which is a C++ based open-source development environments and provides a robust interface to the vehicle. Lowest-level details or

information of the vehicle (e.g., odometer data) can be easily retrieved by means of the ARIA, and moving commands can be sent by means of the ARIA also. In other words, any developer can use the ARIA as an interface to communicate with the embedded system of the vehicle. And we use the Borland C++ builder as the development tool in our experiments.

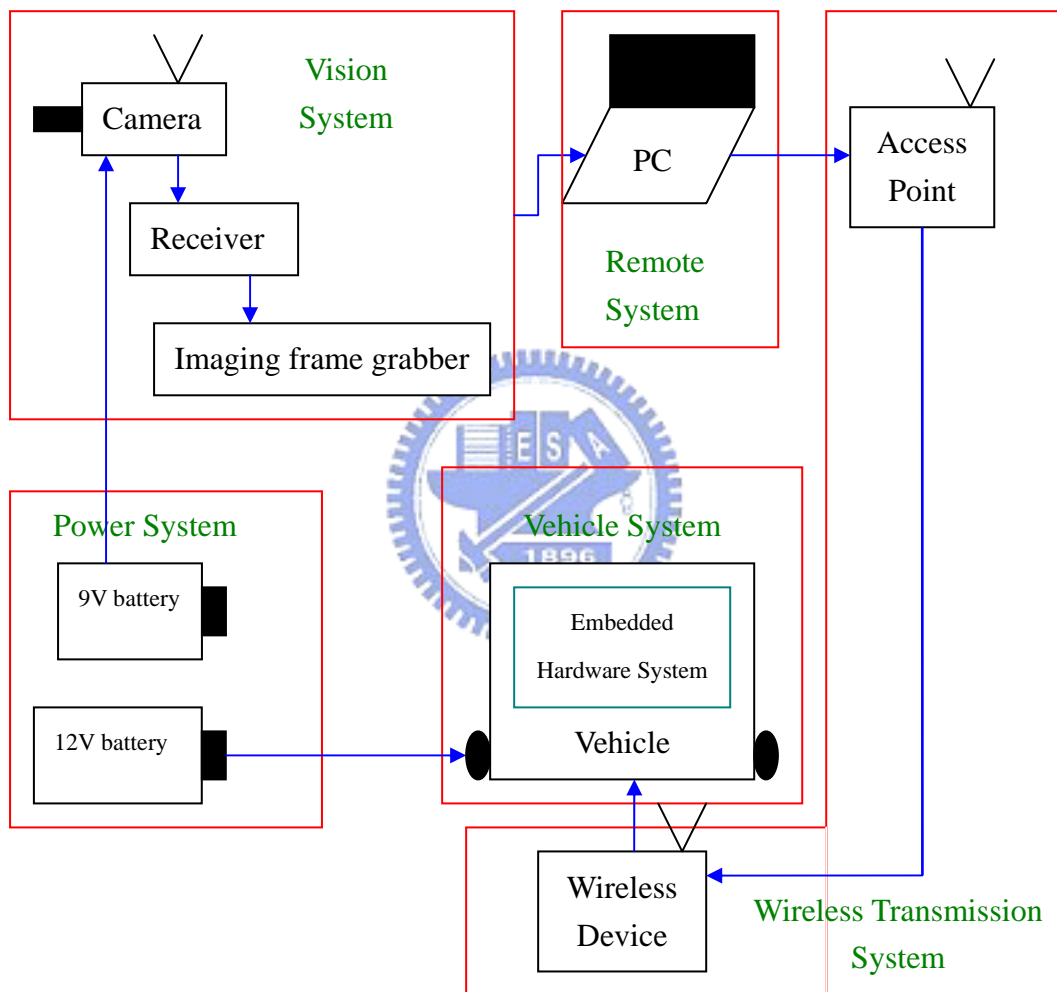


Figure 2.2 Structure of proposed system.

## 2.3 Proposed Learning Principle and Process

To navigate in an unknown environment, learning is necessary. Then, it is desired to develop an efficient learning strategy that can learn the knowledge of desired navigation paths. Nevertheless, in most developed learning strategies, visual environment features are the most important data that are used as the basis for localization or other purposes in navigation. Even though the knowledge of visual environments data is very useful, the processes for obtaining and analyzing these visual data are usually complicated. Therefore, we want to design a learning strategy without using visual data.

The proposed learning strategy consists of three processes: a user-control process, a data-collection process, and a data-transformation process. The user-control process is designed to control the vehicle to learn desired paths. Users can press the buttons on a graphic user interface designed in this study at will to accomplish the learning session.

As soon as the user-control process is executed by the user, the data-collection process is activated. Two types of data are collected orderly in the process, one being user-driving parameter, and the other odometer data. The former is a set of parameters collected upon the user' actions in controlling the vehicle. And the latter is the odometer data which include the global vehicle coordinates and azimuths of the vehicle.

In addition, the path map creation process is activated after both of the user-control process and the data-collection process are finished. The collected data are then transformed into a simple graph with two types of rules, and this simple graph represents the learned path. The data of each node in the simple graph is recorded as an order list and saved into a text file for use in navigation sessions. An illustration of the learning process is shown in Figure 2.3.

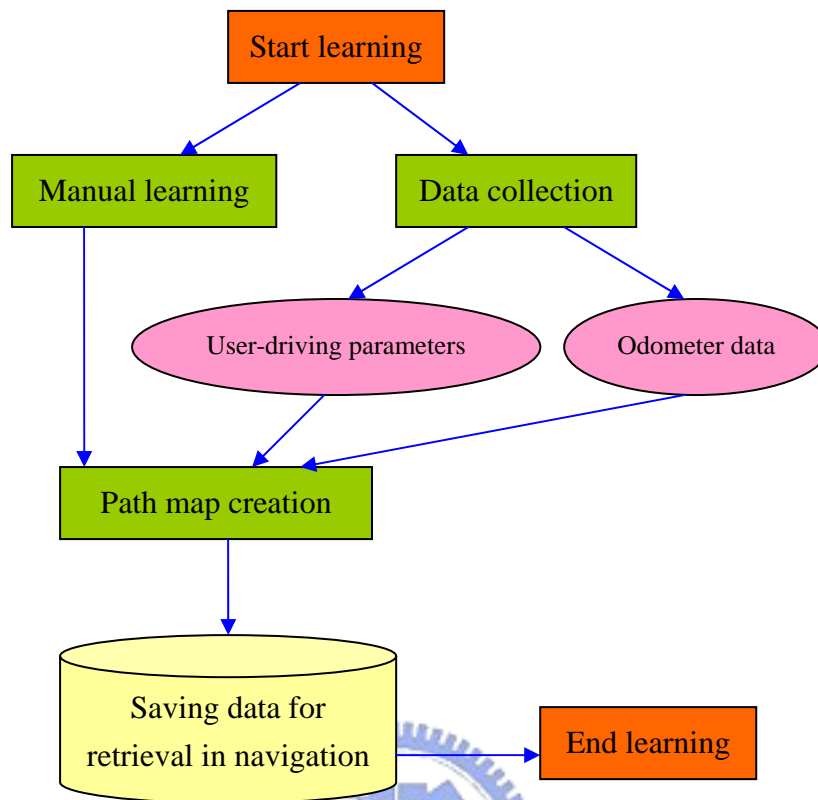


Figure 2.3 Illustration of the proposed learning process

## 2.4 Proposed Vehicle Guidance Principle and Process

The scene in each kind of indoor environment is different and usually complicated, and furniture and decorations within rooms also have difference appearances. Therefore, it is hard to differentiate between obstacles and non-obstacles in guiding the vehicle to correct and safe paths. Hence, a fuzzy-control guidance technique is proposed in this study which guides the vehicle in a natural manner. In every navigation cycle, a suitable steering angle can be obtained by this fuzzy-control



guidance technique and the goal of obstacle avoidance can be achieved in the meanwhile. In other words, the proposed fuzzy-control guidance technique can nominally reduce the complexity of scenes in indoor environments for choosing a suitable path to navigate.

Furthermore, a vehicle navigation process with two navigation modes for different purposes is proposed. One is the *single path* mode for navigating along a learned path with only a starting point and a destination. In the single path mode, the learned data are retrieved from a saved file first, and all the nodes within this navigation path are set in accordance with the nodes in the learned graph exactly. Then the navigation trip can be accomplished by traversing each node in the learned graph orderly. For this reason, two navigation strategies are proposed for the vehicle to navigate in two types of sections during the navigation process. The first strategy is used along each straight-line section between two nodes connected with a line approximately. And the second strategy is used for turning sections where the vehicle navigates around a corner.

The other mode is the *area* mode for navigating among multiple nodes collected from several paths. In the area mode, a desired navigation path may not be learned before. First, the learned data are retrieved from a saved file, too. After that, the desired navigation path is determined by a given starting point and a given destination which are learned in the learning process. The nodes within this navigation path are found dynamically using Dijkstra shortest path searching algorithm. After the desired navigation path is determined, a similar navigation strategy is adopted for traversing along the nodes along the navigation path. An illustration of the vehicle navigation process is shown in Figure 2.4.

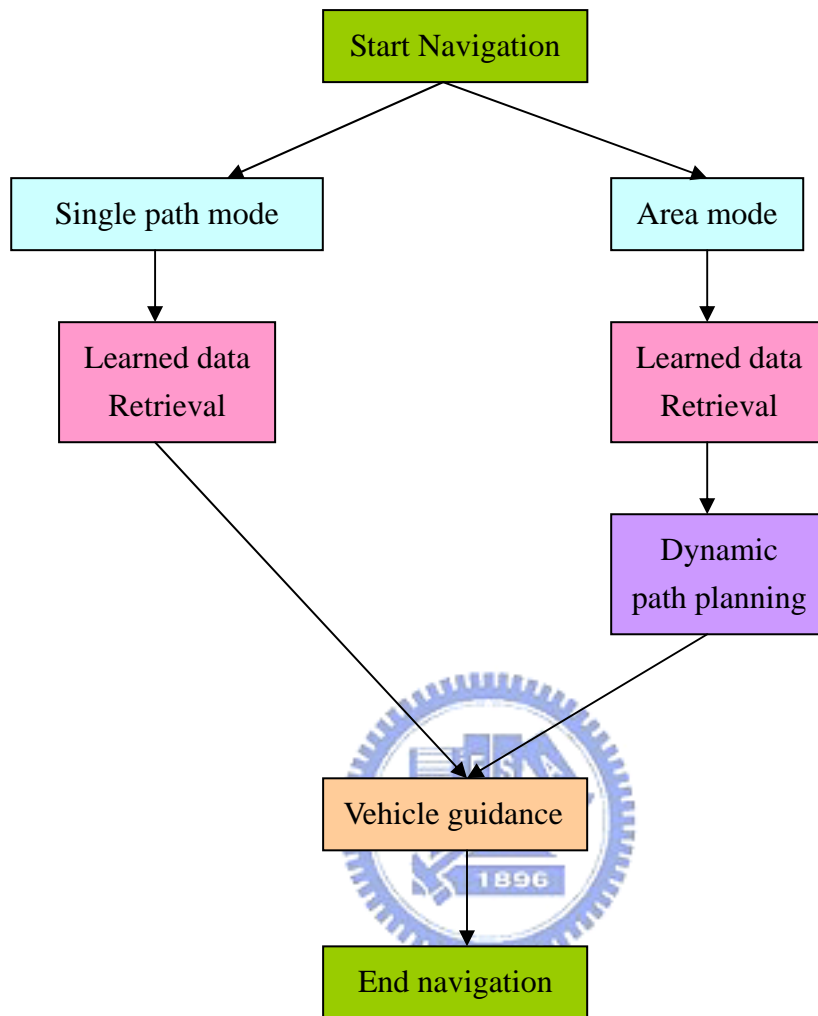


Figure 2.4 An illustration of vehicle navigation process

# Chapter 3

## Proposed Fuzzy Guidance Techniques for Indoor Navigation

### 3.1 Introduction

In order to guide a small vehicle to navigate in indoor environments with complicated scenes, two fuzzy-control guidance techniques are proposed in this study for safe indoor navigation. Since the principle of fuzzy control is intuitive and experiential, we utilize fuzzy-guidance techniques for vehicle navigation. The essence of this approach is to keep equilibrium between navigation accuracy and fuzzy control.

Generally speaking, fuzzy theory can be implemented in a wide variety of fields because of its multidisciplinary nature. A system based on a fuzzy will react to triggered events autonomously in an intuitive way. Due to this characteristic of fuzzy theory, many researchers use fuzzy theory in robot-control applications. A review of the fuzzy-control concept will be described in Section 3.2. And the reason of using fuzzy-control techniques for guidance of small vehicles in this study will be described in Section 3.3.

Besides, some parameters for the proposed fuzzy model must be acquired through some image processing works. Since appearances of different kinds of indoor environments can be classified roughly into two types, namely, room and corridor, two processes of parameter acquirement for the two types of indoor environments are

proposed and will be described in Section 3.4. After necessary parameters are acquired, desired fuzzy outputs as meaningful commands for vehicle control can be obtained by a computation process based on the proposed fuzzy model. Similarly to the need of two parameter acquirement processes, two different fuzzy models are designed for vehicle guidance in the two types of indoor environments. The details of the fuzzy models will be described in Section 3.5.

## 3.2 Review on Fuzzy Control Concepts

The proposed fuzzy control system is derived from fuzzy inference techniques using fuzzy if-then rules and fuzzy reasoning. According to C.T. Sun [20], the basic structure of a fuzzy inference system consists of three conceptual components: a *rule base*, which contains the detailed fuzzy rules; a *database*, which defines the membership functions used in the fuzzy rules; and a *reasoning mechanism*, which performs the fuzzy reasoning process using the rules and certain given facts to obtain a reasonable output.

In a fuzzy inference system, when fuzzy or crisp inputs are taken into the system, the outputs are fuzzy sets in most cases. But fuzzy-set outputs are not suitable for use in our vehicle control system. Therefore, a method of defuzzification is needed to extract a crisp value which best represents a fuzzy set. Besides, in the case with crisp inputs and outputs, the fuzzy inference system just implements a nonlinear mapping from its input space to its output space. And this mapping is accomplished by a number of fuzzy if-then rules, each of which describes the local behavior of the

mapping. The proposed fuzzy control techniques belong to this case.

The procedure of a fuzzy inference system with a crisp output is shown in Figure 3.1. The first part is an input process which takes some fuzzy or crisp values as inputs into this system. And the second part is a fuzzy reasoning process which processes the inputs with fuzzy rules to yield fuzzy-set outputs. The third part is a defuzzification process which transforms the output fuzzy sets into a single crisp value. The final crisp value will be used as the basis for generating commands to guide the vehicle in the proposed fuzzy guidance techniques.

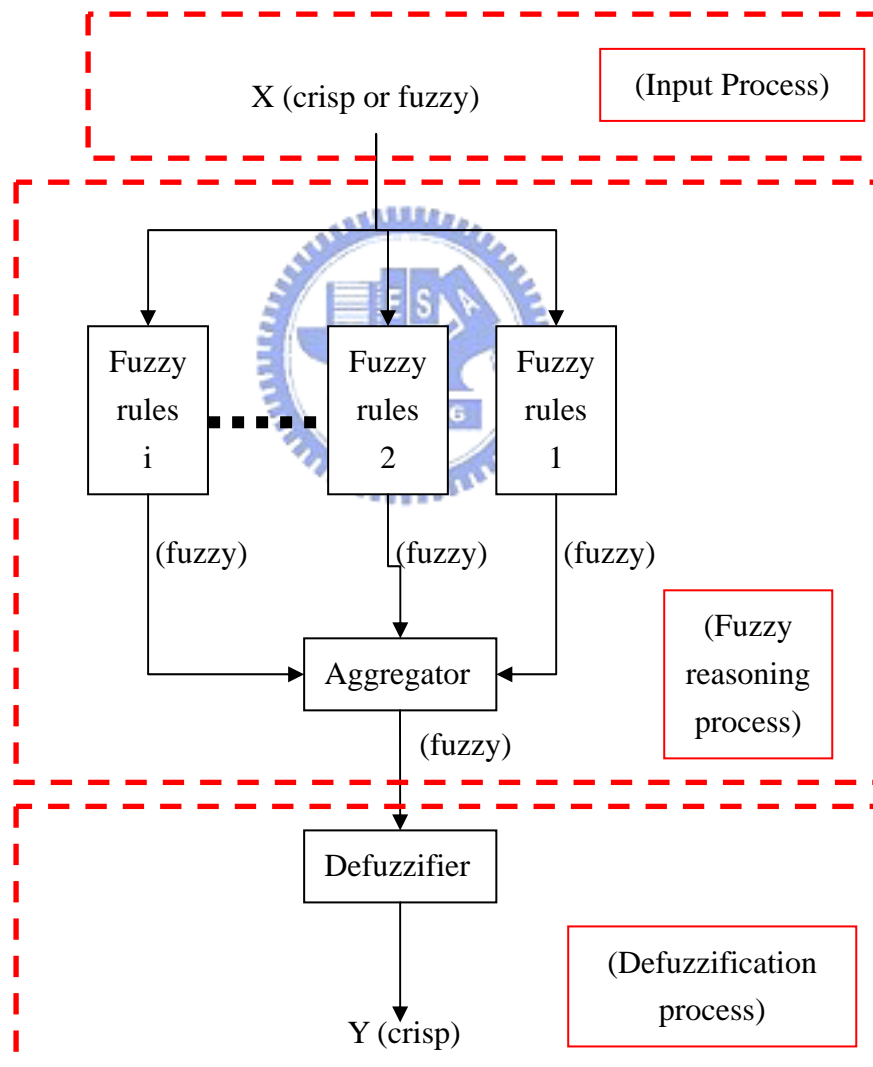


Figure 3.1 Illustration of a fuzzy inference system.

### 3.3 Reasons of Using Fuzzy Guidance Techniques

The reason for using fuzzy guidance techniques in this study is three-fold, as described in the following.

1. *To reduce the complexity of guidance in indoor environments.*

Since the scene structures of indoor environments are usually complicated, it is generally difficult to guide a vehicle to navigate in it. One way to solve this problem is to use fuzzy guidance techniques, which presumably are more proper for use to find vehicle navigation paths. That is, it is expected that complicated scene structures will not influence vehicle guidance when fuzzy guidance techniques are employed.

2. *To create obstacle avoidance capability for the vehicle.*

When a vehicle is navigating in indoor environments using the proposed fuzzy guidance techniques, obstacles may appear in the navigation path. An example of images grabbed by the wireless camera in our vehicle system is shown in Figure 3.1. Obstacles are regarded in this study as furniture in indoor environments or simply as part of environments because of the use of fuzzy guidance techniques. Therefore, our method for finding safe navigation paths is simple with no necessity to differentiate between objects in navigation paths and furniture or walls in environments. In the situation shown in Figure 3.1, the vehicle is steered to the left side in our experiment to avoid possible collision by considering the paper box as part of the environments instead of as an obstacle.

3. *To create an adaptive capability for the proposed vehicle system to navigate in different indoor environments.*

Some vehicle systems use visual environments features (e. g., baseline, corner, manual landmark, etc.) in the guidance process. Since these kinds of visual features can be found only in common indoor environments, these vehicle navigation systems are useless when navigating in certain environments with no such visual feature. This weakness will not be found in our navigation system using the proposed fuzzy guidance techniques because any kind of object will be regarded as part of the environment. Therefore, the proposed vehicle navigation system can be used in any kind of indoor environments without any restriction on the scene structures of the navigation environments.



Figure 3.2 An image of an obstacle appearing in navigation path.

## **3.4 Parameters for Proposed Fuzzy Guidance System**

According to the review on fuzzy-control concepts in Section 3.2, in order to

acquire a crisp output value, certain crisp values must be taken into consideration in the fuzzy guidance system first. And these crisp values are related to the designed fuzzy rules which are determined in the phase of fuzzy model development. In the proposed fuzzy guidance techniques, still images captured from the wireless camera will be processed to obtain appropriate features as crisp inputs for the fuzzy guidance system in every navigation cycle. And the proposed processes of parameter acquirement for use as inputs to the proposed fuzzy models will be detailed in the following.

Because two different fuzzy guidance techniques are designed in this study, two different processes, as mentioned previously, which acquire the demanded parameters for the two fuzzy guidance techniques, are proposed. They are described in Sections 3.4.1 and 3.4.2, respectively.



### **3.4.1 Features for Vehicle Guidance in Rooms**

In every navigation cycle, an input image captured with the wireless camera is processed to obtain two kinds of features, namely, *collision-free direction*, and *degrees of collisions of the left and the right route sides*. The values of these features are taken as inputs into the proposed fuzzy guidance system, which yields an angle as the output for use to steer the vehicle in each navigation cycle. In the following, we describe how we extract the two kinds of features by image processing techniques in the captured images.

#### **A. Computing collision-free direction**

The feature of *collision-free direction* means the direction into which the vehicle may be driven with no collision with the objects along the path traversed by the



vehicle. Such a feature is computed in every navigation cycle in the following way. First, we divide an input image into  $4 \times 4$  blocks and classify each image block into two classes, namely, *route area* and *non-route one*. As a preliminary step to achieve this goal, we utilize an algorithm presented in Li and Tsai [6] to locate image blocks of possible route areas. The essence of the algorithm is to use two pixel features, namely, *a pixel's grayscale value* and *its Sobel edge value*, to identify *candidate route-area pixels*. A pixel in the input image with its grayscale value close to a pre-learned grayscale value of the room ground and with its Sobel edge value smaller than a pre-selected threshold is classified as a candidate route-area pixel. An example of such image pixel classification results is shown in Figs. 3.3(a) and 3.3(b). Because of color and uniformity similarities, some wall or furniture regions may be misclassified as route areas, as can be seen in the example shown in Fig. 3.3(b). Nevertheless, we propose an algorithm in this study to remove such erroneous areas in the following, which in addition computes the above-mentioned feature of collision-free direction from the remaining correct route areas in the input image.

**Algorithm 1.** *Computation of route areas and collision-free direction.*

- Step 1. Perform region growing to find as the desired route area the largest bottom region in the candidate route-area pixels extracted from the input image using [6]. (The region growing result of the above example is shown in Fig. 3.3(c).)
- Step 2. Put two parallel horizontal scanning lines in a fixed lower part of the route area. If any non-route area crosses either scanning line and cuts it into several line segments, pick out the longest segment and call it a *non-obstacle segment*; otherwise, select the original scanning line as the non-obstacle segment. (At the end of this step, two non-obstacle segments

will be obtained.)

- Step 3. Find the middle points of the two non-obstacle segments, connect them to form a line segment, and call it *route segment*. (The found middle points for the above example are A and B as shown in Fig. 3.3(c).)
- Step 4. Compute the middle point of the route segment, and call it *route center*. (The result of this step for the last example is point C in Fig. 3.3(c).)
- Step 5. Connect the route center to the middle point of the bottom line of the image to form a line segment, and call it *guidance line* (like the line labeled L in Fig. 3.3(d).)
- Step 6. Find as the desired collision-free direction  $\theta_g$  the angle of the guidance line with respect to the bottom line of the image.

In the above algorithm, no complicated 3D computer vision technique is used in computing the collision-free direction, as contrasted with most conventional methods. Also, notice that by the above algorithm, the vehicle automatically has the capability of obstacle avoidance. Actually, obstacles in the path are treated as outside-path objects (like furniture) in this study.

## **B. Computation of degrees of collisions**

The degrees of collisions of the left and the right route sides are defined and computed in this section. We first define a rectangular window in each captured image with two sub-windows separated by a centerline, as shown in Fig. 3.3(e). A region in each rectangular window of captured images represents an area which is two meters in front of the small vehicle. Then the proportion of the route area in the left sub-window is defined to be the degree of collision of the left route side, which will be denoted by  $P_L$ . And that of the right route side can be defined similarly, and will be denoted by  $P_R$ .

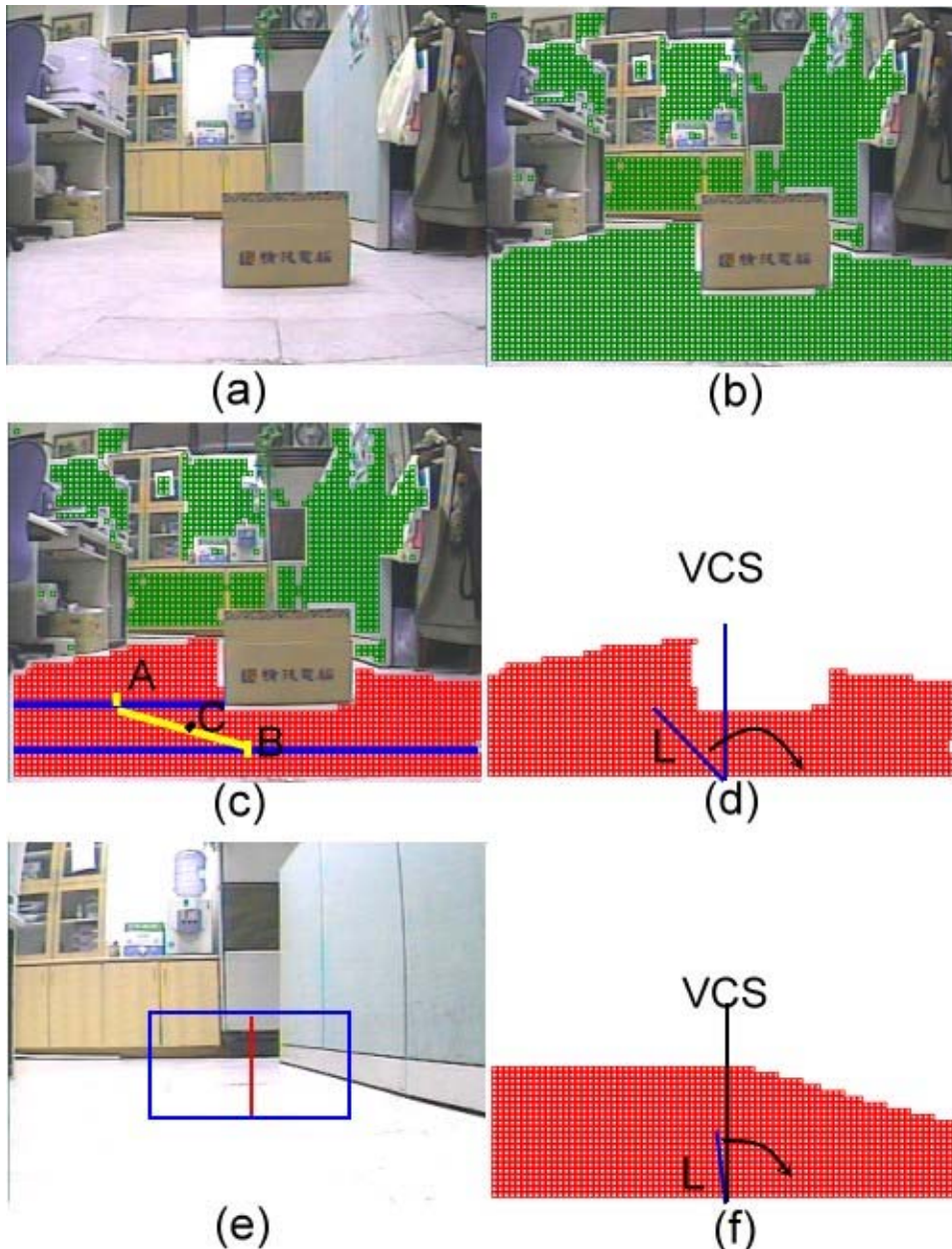


Figure 3.3 Illustration of image processing results (a) An input image. (b) Result of candidate route point classification. (c) Result of route area extraction. (d) Result of guidance line computation. (e) Another input image. (f) Result of guidance line computation of (e).

### 3.4.2 Features for Vehicle Guidance in Corridors

The demanded features for the fuzzy guidance system for the corridor type of

indoor environments are different from those used for the room type. Two kinds of features, namely, *baseline angle at the right side or at the left side*, and *baseline height at the right side or at the left side*, are computed by processing an input image captured with the wireless camera in every navigation cycle. These features are taken as inputs into the fuzzy guidance system, which yields an angle as the output for use to steer the vehicle in each navigation cycle. In the following, we describe how we extract the two kinds of features by image processing techniques in captured images for use in vehicle guidance.

#### **A. Computing baseline angle at right side or at left side**

The feature of baseline angle either at the left or the right side means the angle between the lower edge of the baseline and the left or right image boundary in the captured image in every navigation cycle. Since baselines are more common in building corridors, we utilize their visual characteristics in the field of view of the camera on the vehicle to adjust the vehicle's moving direction in the navigation process. We propose an algorithm to compute the value of the baseline angle feature at either side in the input image in the following.

**Algorithm 2.** *Computation of baseline angle at either side.*

- Step 1. Convert the input image into a binary image with a threshold  $T_b$  that is defined in advance. (The result of the above example is shown in Fig. 3.4(b).)
- Step 2. Apply the Sobel operator to the processed binary image to find all edges in the input image. (The result of the above example is shown in Fig. 3.4(c).)
- Step 3. Find the point of the lower edge of the baseline intersecting the image boundary at either side, and find the other end point of the baseline in the

image using a connected component computation method [21].

- Step 4. Calculate the line equations of the two edges (upper and lower), denoted as  $L_r$  and  $L_\ell$ , of the baseline at either side according to the two points found in the last step. Denote also the two vertical image boundaries as  $V_r$  and  $V_\ell$ . (The result of the above example is shown in Fig. 3.4(d) and (e).)
- Step 5. Calculate the angle between  $L_r$  and  $V_r$ , and that between  $L_\ell$  and  $V_\ell$ , and denote them as  $\theta_r$  and  $\theta_\ell$ , which are the desired features as inputs to the fuzzy guidance system. (The result of the above example is shown in Fig. 3.4(d) and (e).)

In the above algorithm, we use conventional image processing techniques to compute the two baseline angles. It is pointed out here that the two baselines are not both seen at every navigation cycle, because the direction of the vehicle may be directed to the left or right side. Therefore, a reasonable assumption is made in advance, that is, if no baseline angle can be found due to failure of finding the baseline, then the baseline angle is set to 90 degrees.

## **B. Computing baseline height at right or left side**

The feature of baseline height means the  $y$  image coordinate value of the start point of each baseline in the input image, as shown in Figs. 3.4(d) and 3.4(e). This  $y$  coordinate value can be obtained easily by performing Steps 1 through 3 of Algorithm 2. The baseline height at the right side will be denoted by  $H_r$ , while that at the left side by  $H_\ell$ . Generally speaking, the value of the baseline height is larger when the direction of the vehicle tends to the right or left wall, and on the contrary, the value is smaller when the direction of the vehicle tends to the midway of the corridor.

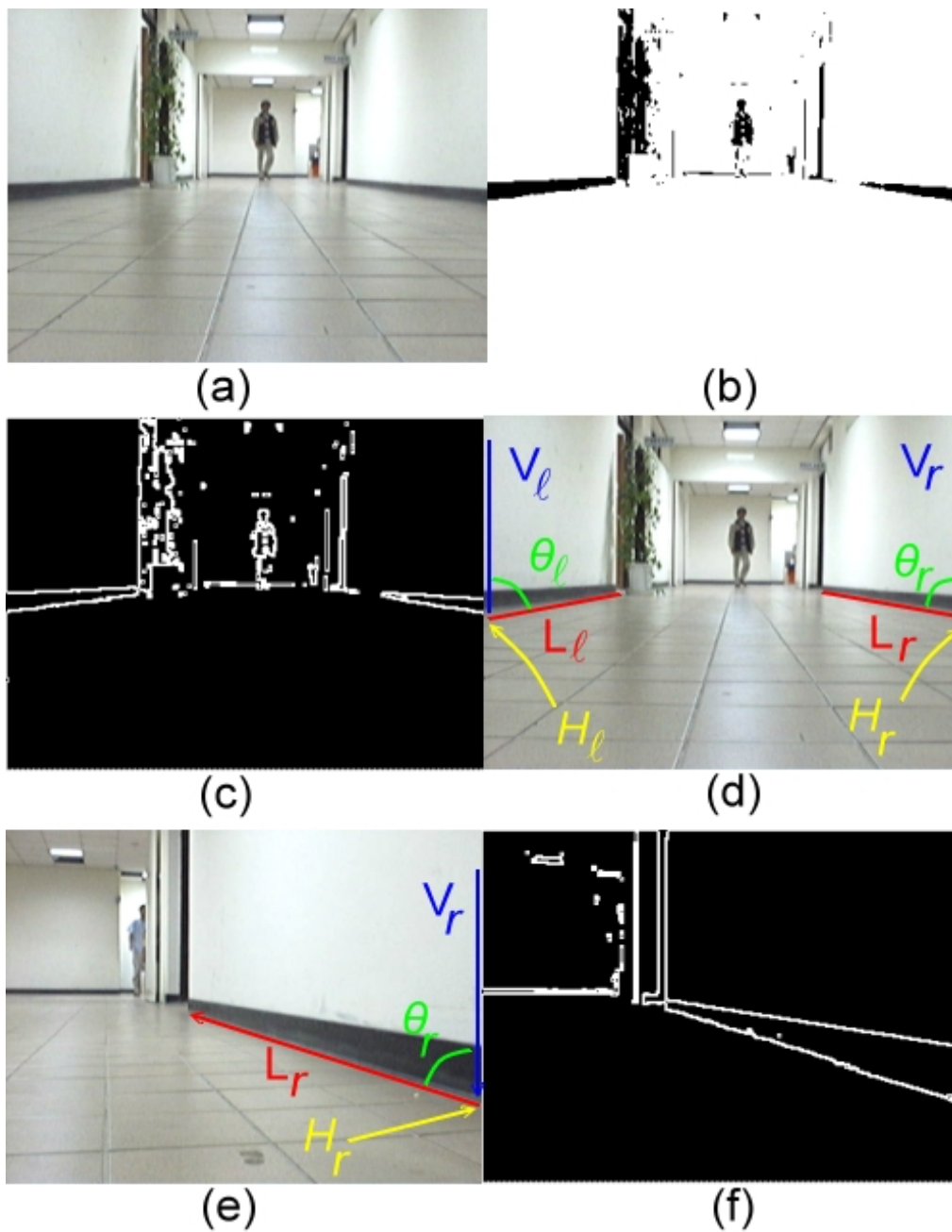


Figure 3.4 Illustration of image processing results (a) An input image. (b) Result of binarization process. (c) Result of applying Sobel operator to (b). (d) Result of finding connected components. (e) An another input image. (f) Result of applying binarization process and Sobel operator.



## 3.5 Proposed Fuzzy-Control Systems and Their Uses in Guidance

After the demanded features described in Section 3.4 are acquired, the input process of the fuzzy control system is finished and the next process is started. In the following, the fuzzy reasoning process that consists of fuzzy rule design, fuzzy aggregation, and defuzzification will be discussed. Because of the difference between the previously-mentioned two indoor environments types, two different fuzzy reasoning processes and their respective defuzzification processes are designed in this study and will be detailed in Section 3.5.1 and Section 3.5.2, respectively.

### 3.5.1 Proposed Fuzzy Control System for Room-Type Guidance

In the proposed fuzzy control system, when an acquired feature is processed, a *steering angle*  $\theta_s^i$  is computed for use in turning the vehicle in the  $i$ th navigation cycle. If  $\theta_s^i$  is positive, it means that the vehicle should turn leftward for the angle of  $\theta_s^i$ ; if  $\theta_s^i$  is negative, it means that the vehicle should turn rightward for  $\theta_s^i$ . The proposed fuzzy control system is based on two fuzzy rules in terms of three linguistic variables LESS, LEFT, and RIGHT that describe the values of the input features. The rules are described as follows:

**Rule 1:** if the collision-free direction  $\theta_g$  trends to LEFT and the degree of collision of the right route side  $P_R$  is LESS, then turn the vehicle rightward for the

angle of  $\theta_s^i$ ;

**Rule 2:** if  $\theta_g$  trends to RIGHT and  $P_L$  is LESS, then turn the vehicle leftward for  $\theta_s^i$ .

To implement the above two rules, six membership functions  $\mu_{LEFT}(\theta_g)$ ,  $\mu_{RIGHT}(\theta_g)$ ,  $\mu_{LESS-R}(P_R)$ ,  $\mu_{LESS-L}(P_L)$ ,  $\mu_{f1}(\theta_s^i)$ , and  $\mu_{f2}(\theta_s^i)$  are defined as shown in Fig. 3.5, in which the units of  $\theta_g$ , and  $P_R$  and  $P_L$  are degree and percentage, respectively.

The fire strengths of Rules 1 and 2 can be calculated accordingly as follows:

$$m_1 = \mu_{LEFT}(\theta_g) \wedge \mu_{LESS-R}(P_R);$$

$$m_2 = \mu_{RIGHT}(\theta_g) \wedge \mu_{LESS-L}(P_L),$$

where  $\wedge$  denotes the AND operator which is defined as the minimum function. After the fuzzification stage, the conclusion of each rule can be derived according to fuzzy reasoning. Because the membership function of the conclusion of each rule is monotonic, the Tsukamoto defuzzification method [7] can be applied to the fuzzy reasoning process here. The crisp output  $\theta_s^i$ , which is the desired steering angle, is calculated by the following equation:

$$\theta_s^i = \frac{\sum_{j=1}^2 m_j \mu_{\theta_s^j}^{-1}(m_j)}{\sum_{j=1}^2 m_j}$$

where  $m_k$  is the firing strength of Rule  $k$ , and  $\mu_{\theta_s^j}^{-1}(m_j)$  is the inverse function of  $\mu_{\theta_s^j}(\theta_s^i)$ . The previous discussion results have been followed to design a fuzzy guidance algorithm in this study. The details are omitted.



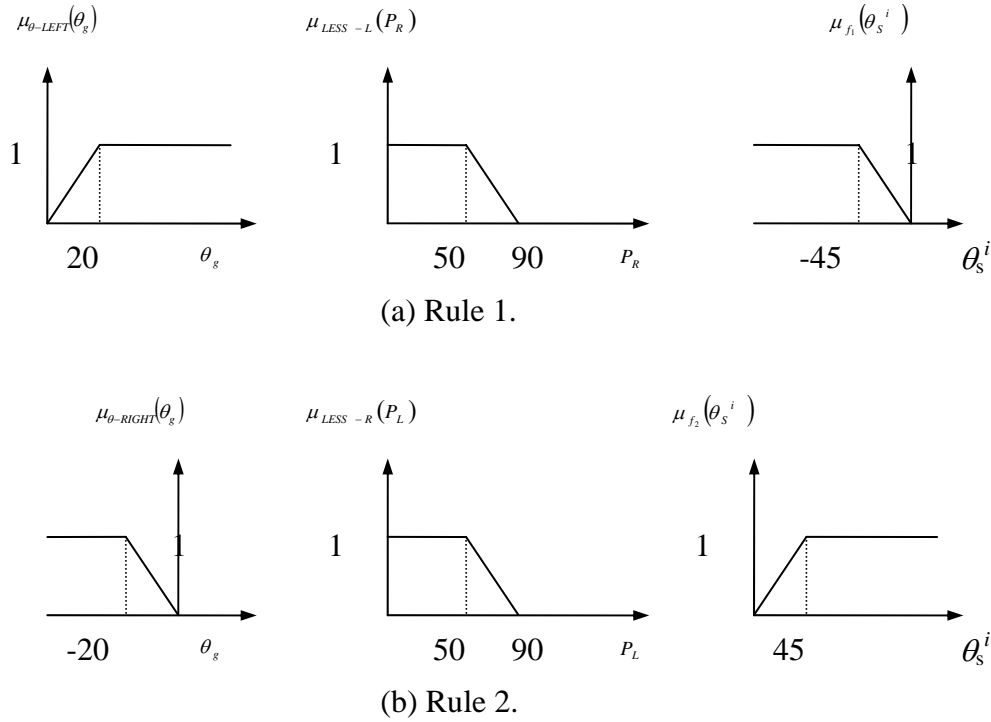


Figure 3.5 Membership functions for guidance in rooms.

### 3.5.2 Proposed Fuzzy Control System for Corridor-Type Guidance

In contrast with the fuzzy control system for the room type described above, the fuzzy control system for the corridor type is different in fuzzy rules and input features. The *steering angle*  $\theta_s^i$  is an output of the fuzzy control system in the  $i$ th navigation cycle, which is the same as the condition of room-type guidance. If  $\theta_s^i$  is positive, it means that the vehicle should turn leftward for the angle of  $\theta_s^i$ ; if  $\theta_s^i$  is negative, it means that the vehicle should turn rightward for  $\theta_s^i$ . The proposed fuzzy control system is based on two fuzzy rules in terms of two linguistic variables, LARGE and SMALL, that describe the values of the input features. The rules are described as follows:

**Rule 3:** if the baseline angle  $\theta_l$  is SMALL and the baseline height at the left side  $H_l$  is LARGE, then turn the vehicle rightward for the angle of  $\theta_s^i$ ;

**Rule 4:** if  $\theta_r$  is SMALL and  $H_r$  is LARGE, then turn the vehicle leftward for  $\theta_s^i$ .

To implement the above two rules, six membership functions  $\mu_{SMALL-R}(\theta_r)$ ,  $\mu_{SMALL-L}(\theta_l)$ ,  $\mu_{LARGE-R}(H_r)$ ,  $\mu_{LARGE-L}(H_l)$ ,  $\mu_{f_3}(\theta_s^i)$ , and  $\mu_{f_4}(\theta_s^i)$  are defined as shown in Fig. 3.6, in which the units of  $\theta_r$  and  $\theta_l$ , and  $H_r$  and  $H_l$  are degree and integer, respectively.

The fire strengths of Rules 3 and 4 can be calculated accordingly as follows:

$$m_3 = \mu_{SMALL-L}(\theta_l) \wedge \mu_{LARGE-L}(H_l);$$

$$m_4 = \mu_{SMALL-R}(\theta_r) \wedge \mu_{LARGE-R}(H_r),$$

where  $\wedge$  denotes the AND operator defined again as the minimum function. After the fuzzification stage, the conclusion of each rule can be derived according to fuzzy reasoning. Because the membership function of the conclusion of each rule is monotonic, the Tsukamoto defuzzification method again can be applied to the fuzzy reasoning here. The crisp output  $\theta_s^i$ , which is the desired steering angle, is calculated by the following equation:

$$\theta_s^i = \frac{\sum_{j=3}^4 m_j \mu_{\theta_s^j}^{-1}(m_j)}{\sum_{j=3}^4 m_j}$$

where  $m_k$  is the firing strength of Rule  $k$ , and  $\mu_{\theta_s^j}^{-1}(m_j)$  is the inverse function of  $\mu_{\theta_s^j}(\theta_s^i)$ . The previous discussion results have been followed to design a fuzzy

guidance algorithm in this study. The details are omitted here.

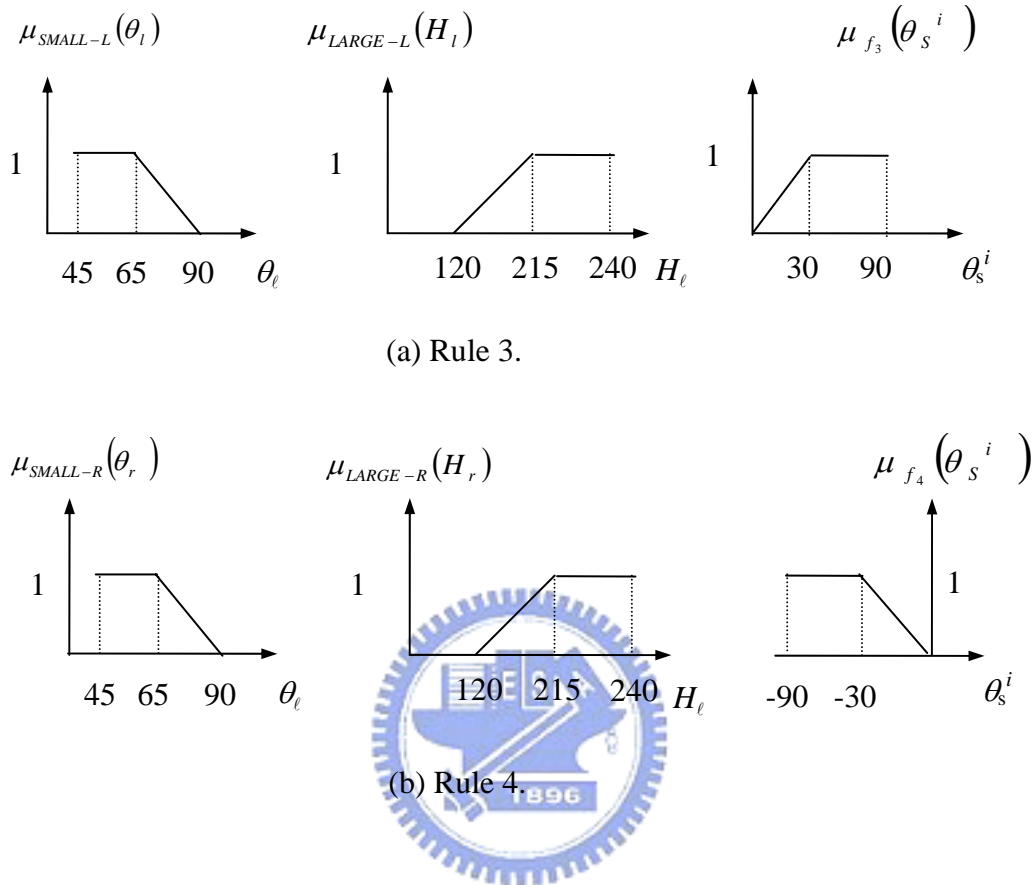


Figure 3.6 Membership functions for guidance in corridors.

# Chapter 4

## Simple Learning Strategies for Indoor Navigation by User-Driving and Odometer Parameters

### 4.1 Introduction

In this chapter, we propose a *non-visual* approach to learning which solves all the problems encountered in the conventional vision-based learning approach. This non-visual learning process is just a sequence of *free* actions of vehicle driving without using a vision system. The vehicle is driven by the user from one room spot to another as wished. Only simple data consisting of the records of user-driving actions as well as the odometer values of each spot visited along traversed paths are collected. To achieve this goal of simple learning process, we design a convenient user-control interface and simple user-driving rules, and these will be described in Section 4.2.

After the manual learning process is finished, an automatic transformation method is proposed for creating path maps with the learned data as input. The generated path map consists of a set of connected nodes, with each inter-node edge being a traversed trajectory in the learning process. Since the navigation process is classified into two modes, namely *Single path mode* and *Area mode*, we also propose two kinds of path creation processes for this purpose. In the path creation process for

each navigation mode, we propose an algorithm for identifying two types of path nodes, *check node* and *turn node*, in the path data collected in the manual learning process to create a path map of the visited room spots in the form of a graph with undirected edges specifying the navigated routes. The details of the automatic path map creation processes in the single path mode and the area mode are described in Section 4.3.

## 4.2 Two Navigation Modes

Two navigation modes are designed, which are aimed to fit the vehicle for different applications. One is the *single path mode*, the other the *area mode*. Since the characters and functions of each navigation mode are so different, the details of each navigation mode are also different, as described in the following.

The *single path mode* means that only one navigation path is learned in the manual learning process, and the vehicle just complies with the processed data of the learned path to navigate along this path exactly. This navigation mode is applicable to applications where only one-way unceasing navigation is required. For instance, in certain kinds of “security patrolling” applications, an autonomous vehicle system is requested to navigate along a fixed path in a building. For this reason, the autonomous vehicle system can be controlled to learn a single path in advance, and then it can navigate along this learned path back-and-forth automatically, instead of hiring a security officer to guard the building. An important characteristic of the *single path mode* for the vehicle is that it can accomplish its navigation works autonomously without issuing any user command during the navigation process.

The *area mode* means that multiple navigation paths are learned in the manual learning process, and the vehicle can choose a shortest navigation path dynamically according to a pair of “start point” and “end point,” which is given by the user. This navigation mode is applicable to applications where “flexible point-to-point” navigation is required. For instance, in certain kinds of “home service” applications, an autonomous vehicle system might be requested to navigate among rooms to accomplish some service works (e. g., morning calls or article transportations). Nevertheless, to learn all the routes among these rooms sometimes is too much for a vehicle (e. g., there are total 60 routes among five rooms). Therefore, we design the *area mode* for navigation to solve this problem. In the proposed manual learning process for the area mode, the number of necessary learned paths is the same as the number of rooms that the user wants the vehicle to visit. And in the navigation stage, according to the proposed navigation strategy for the *area mode* with the corresponding learned data, the vehicle will choose a shortest path dynamically to navigate among any two rooms after a user-defined command is ordered. In short, the complexity of the manual learning process can be reduced significantly by using the *area mode* learning process for navigation.

## 4.3 Manual Learning Algorithm

Some schemes are proposed in this study to accomplish the manual learning process by a user. For a user to control the vehicle easily and correctly, a plain control interface and certain simple control rules are designed. Furthermore, when a user is controlling the vehicle to learn a desired navigation path, some employed features

consisting of the user-driving actions and the odometers data are collected by the proposed system and stored into three types for later processing. Besides, the vision system on the vehicle is disabled during the learning process, because the proposed learning algorithm is non-visual and uses no landmark or special features for vehicle location in the navigation environment.

### **4.3.1 Proposed Control Interface for Manual Learning**

Since the main concept of the proposed learning strategy is simple, we hope the implementation of the proposed learning algorithm can be used for general users, even for those who do not know much about computer techniques. As a result, a plain control interface is designed for this purpose, as shown in Fig 4.1. Every user can drive the vehicle through this simple control interface to learn a desired navigation path. In the following, illustrations of this control interface will be described.

Firstly, the user can press “move forward” or “move backward” to control the vehicle to move a straight line until the “stop” button is pressed. Either the user can adjust the “angle scroll bar” to a suitable position followed by pressing the “turn leftward” or “turn rightward” to turn the vehicle for a desired angle. In the single path mode, the user can utilize only the above-mentioned control actions to move the vehicle to learn the desired path easily.

In addition, in the area mode two works should be done by the user. Firstly, before starting the learning process, the user must define a certain location as a start point and bring the vehicle to this start point. Secondly, when a one-way learning is finished at a room spot, the user must attach a certain description for this end place as

a label. This work can be done also on the simple control interface by entering a text description in a corresponding field and pressing the “attach” button. Followed by the label attaching action, the user must bring the vehicle back to the start point. After that, the user needs to press the “another trip” button to start another one-way learning, or press the “end” button to finish the learning session in the area mode.

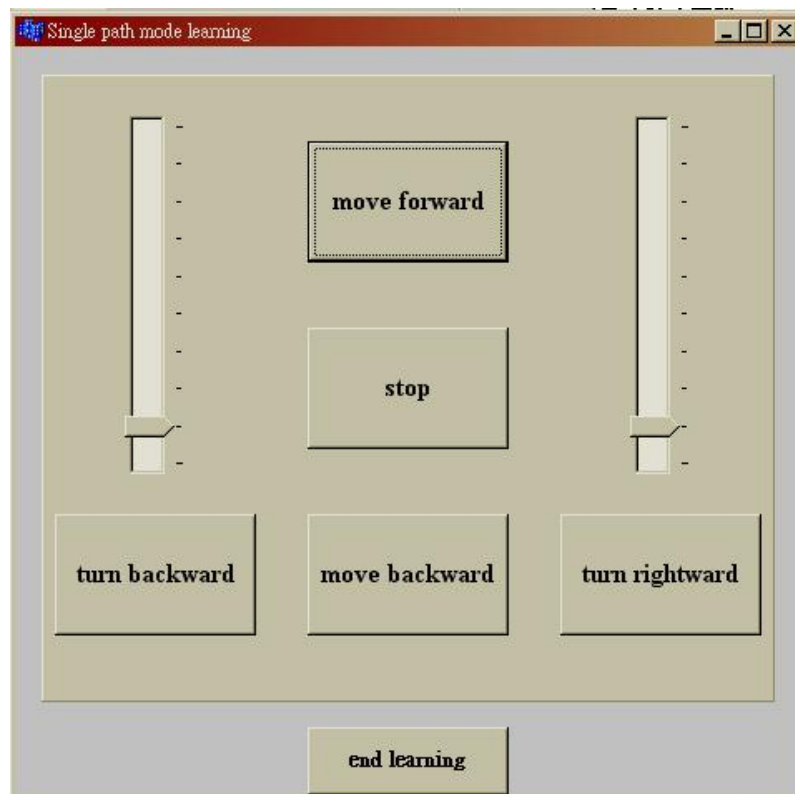


Figure 4.1 An illustration of the proposed user control interface.

### 4.3.2 Proposed Simple User-Driving Rules for Learning

Generally speaking, a learning interface is an important part of the learning



system because it will affect the pleasure of a user operating the system. Nevertheless, a set of effective learning rules without complicated restrictions is another important part of the learning system. Only the use of a combination of a simple user learning interface and a set of simple user learning rules can achieve the goal of simple learning. In the following, some simple user driving rules proposed in this study for learning will be described.

- (1) Control the vehicle to move on the ground in indoor environments, and keep at least 50 centimeters or more to the left- or to the right-side obstacle (e. g., a piece of furniture or anything else).
- (2) When the distance between the vehicle and its front object is approximately smaller than 1 meter, stop the vehicle and turn rightward or leftward.
- (3) When the user wants to turn rightward or leftward and the vehicle is still moving, stop the vehicle right away.
- (4) Issue only necessary commands and avoid changing actions all the time.

### 4.3.3 Data Structures of Learned Data

During the previously-mentioned manual driving step in the learning stage, the proposed system allows a user to control the vehicle to move freely in an unknown environment through the simple control interface. And certain data about the learning process are recorded for later processing. All the recorded data are call *learned data*, and it can be classified into three types of data structure, namely *user-driving actions*, *odometer values*, and *user-defined data*. The detail of these data structures will be described as follows.

A *user-driving action* means an action that is requested by the user by pressing a corresponding button on the user control interface in a learning process, including the

“move forward”, “move backward”, “move leftward”, “move rightward”, “stop”, “attach”, “another trip”, and “end”. Each action taken by the user is recorded as a command together with its execution time with respect to the start time of the learning process. A list of the labels for data representing the *user-driving actions* is shown in Table 4.1.

And the location of the vehicle, including its position and direction, is recorded every fixed time period (say, every second). The position data consists of the *x-axis* values and *y-axis* values for the global coordinate system (GCS), and the range of the direction data is about 360 degrees. These location data of the vehicle are called *odometer values* which are retrieved from the embedded system on the vehicle through the ARIA API. A list of the labels for data representing the *odometer values* is also shown in Table 4.1.

The remaining data structure is *user-defined data* which represent information of corresponding room spots in area-mode learning. Each set of *user-defined data* is composed of one or two meaningful texts or several ones given by the user at will. But even if the number of meaningful texts is more than one, they will be regarded as a single set of *user-defined data* at each room spot.

Furthermore, each recorded command is associated with three or four parameters. More specifically, either of the “turn leftward” or the “turn rightward” command is associated with the parameters of turn angle, position coordinates, direction angle, and action time; and each of the “move forward”, the “move backward”, and the “stop” command with the parameters of position coordinates, direction angle, and action time (without the turn angle). An illustrative example is shown in Figure 4.2.

TABLE 1  
USER-DRIVING COMMANDS AND PARAMETERS FOR LEARNED DATA

DATA STRUCTURE	TYPE	COMMAND & PARAMETERS	LABEL OR VALUE
User-driving actions	Command	move forward	$f$
User-driving actions	Command	move backward	$b$
User-driving actions	Command	turn leftward	$l$
User-driving actions	Command	turn rightward	$r$
User-driving actions	Command	stop	$s$
User-driving actions	Command	attach	$c$
User-driving actions	Command	another trip	$z$
User-driving actions	Command	end	$q$
User-driving actions	Parameter	turn angle	$\theta_t$
Odometer values	Parameter	position coordinates	$(x,y)$
Odometer values	Parameter	direction angle	$\theta_d$
Odometer values	Parameter	action time	$t$
User-defined data	Parameter	meaningful description	text

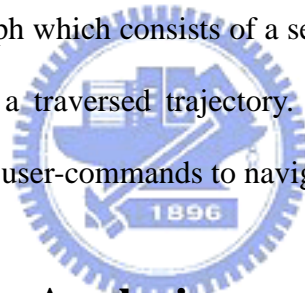


Figure 4.2 An illustration of the learned data.

## 4.4 Automatic Path Map Creation

### Process of Learned data

After the manual learning process is finished, the learned data are recorded as a list. Although this list of learned data collects all the features about user-driving information, it cannot be used for the navigation process directly. The reason is that these learned data are too high-level for the vehicle, it needs more detail data about the learned path (e.g. global coordinates data). Therefore, in this section, certain transformation algorithms are proposed and applied to perform the *path map creation process* automatically. This *path map creation process* transforms the high-level learned data into a simple graph which consists of a set of connected nodes, with each inter-node edge representing a traversed trajectory. Then, the vehicle can use this low-level processed data with user-commands to navigate in the learned environment.



#### 4.4.1 Learned Data Analysis

In essence, the proposed automatic path map creation process uses certain rules to analyze the simple learned data that consist of user-driving actions, odometer data, and user-defined data, which are the only information about the learned path. And we want to use a simple graph to represent the final result of the path map creation process, because a elementary graph with certain inter-node information is clear for the vehicle can accomplish a given navigation task by traversing each inter-node in the graph. Therefore, a critical job of the path map creation process is to identify each inter-node edge with its attached data in the elementary graph. In the following, the principle of this job will be described.

We think of the user-driving actions as useful information in the learned data for determining the location of each inter-node edge. Each user-driving action has its corresponding action time, and since the location of the vehicle is recorded every fixed time period, then we can determine the location relatively where the user-driving action is commanded. By using this transformation concept, the majority of all *inter-nodes* can be determined automatically. Nevertheless, a problem is encountered followed by this concept that must be solved. In certain cases, a time period between two user-driving actions is too long, so the distance between these two determined inter-nodes may be too long, and this kind of case is not suitable for navigation. Therefore, a function need be designed for the path map creation process that can automatically determines more inter-nodes with suitable locations among any two inter-nodes with too-far distance.

As the location of each inter-node edge is determined, the attached data of these inter-node edge can be also determined by other inputted learned data, odometer values and user-defined data. After that, a more integrated simple graph is created by the accomplishment of path map creation process. More details of the proposed algorithm will be described in the next two sections.

#### **4.4.2 Identification of Nodes for Single Path Mode**

With the difference between two proposed navigation modes, as mentioned previously, two kinds of path map creation processes are proposed to create its corresponding simple graph. Here, the proposed automatic path map creation process for the single path mode is described, which is designed to create a graph composed of two kinds of nodes, check node and turn node, from the learned data. The meaning of a check node is to check relevant information when the vehicle arrives at this node

during navigation. And the meaning of a turn node is to launch or end a turning session when the vehicle arrives at this node during navigation. An algorithm is proposed as follows for this purpose.

**Algorithm 1.** *Automatic path map creation process for single path mode.*

Step 1. Take the learned data as input, and identify check nodes and turn nodes by processing the commands sequentially in the input in the following way. Let the currently processed command be denoted as  $C^0$ , the previously processed one as  $C^{-1}$ , and the next processed one as  $C^{+1}$ .

Step 1.1. If  $C^0$  is ‘move forward’, then

- a. if  $C^{-1}$  is ‘turn leftward’ or ‘turn rightward,’ then create a turn node at the spot visited by the vehicle three seconds after the action time of  $C^0$  (i.e., three seconds after  $C^0$  was executed);
- b. if  $C^{+1}$  is executed at a spot farther than a meter from  $C^0$ , then create check nodes at spots with distance intervals of one meter with  $C^0$  as the first created check node.

Step 1.2. If  $C^0$  is ‘stop’, then

if the  $C^{+1}$  is ‘turn leftward’ or ‘turn rightward’, then create a turn node at a spot visited by the vehicle three seconds before the action time of  $C^0$  (i.e., three seconds before  $C^0$  was executed); otherwise, create a check node at  $C^0$ .

Step 2. Repeat Step 1 until all commands in the input are processed.

Step 3. Mark the first produced node as a start point with its corresponding location kept.

Step 4. Mark the last produced node as an end point with its corresponding location

kept.

In the above algorithm, the two threshold values of three seconds and one meter may be varied to meet other application needs. It can be seen from the algorithm that the resulting path map is a simple attributed directed graph, which we found sufficient for this study. An example to illustrate this graph is shown in Figure 4.3.

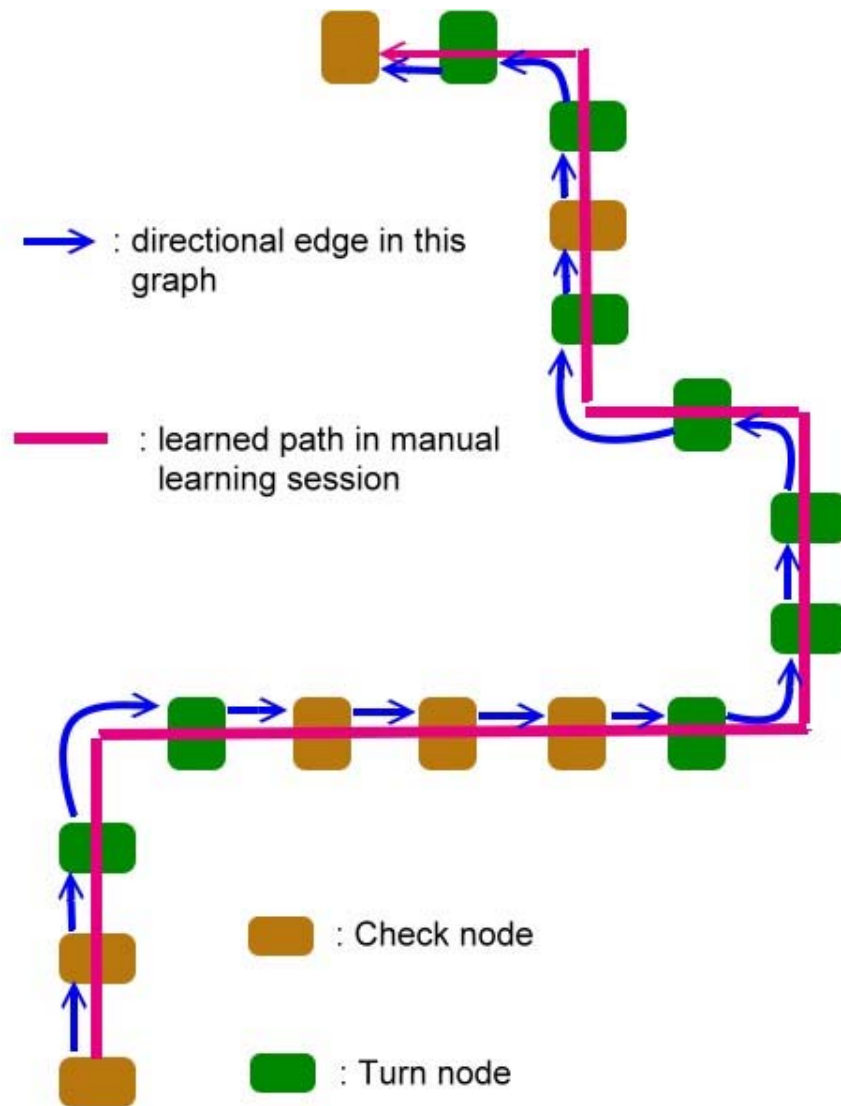
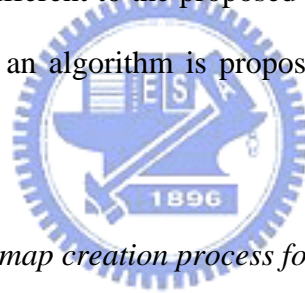


Figure 4.3 An example to illustrate a created graph for single path mode.

### 4.4.3 Identification of Nodes for Area Mode

Here, the proposed path map creation process for the area mode is described, which is designed to create a graph composed of four kinds of nodes, check node, turn node, room node, and initial node, from the learn data. The meanings of the check node in the two kinds of created graphs for two navigation modes are the same, but the property of the turn node is not the same exactly. The initial node is a node with a fixed location defined by the user in each manual learning process for the area mode. And the room node is a node with user-defined data and location that can be used to represent a special place for the use of navigation. Therefore, the proposed navigation strategy in the area mode is different to the proposed navigation strategy in the single path mode. In the following, an algorithm is proposed for identifying two kinds of nodes in the area mode.



**Algorithm 2.** *Automatic path map creation process for area mode.*

Step 1. Attach a counter value to the initial node, and set the initial value of this counter to be '1'.

Step 2. Take the learned data as input, and identify check nodes and turn nodes by processing the commands sequentially in the following way. Let the currently processed command be denoted as  $C^0$ , the previously processed one as  $C^{-1}$ , and the next processed one as  $C^{+1}$ .

Step 2.1. If  $C^0$  is 'move forward', then

if  $C^{+1}$  is executed at a spot farther than a meter from  $C^0$ , then create check nodes at spots with distance intervals of one meter with  $C^0$  as the first created check node.



Step 2.2. If  $C^0$  is 'stop', then

if the  $C^{+1}$  is 'turn leftward' or 'turn rightward', then create a turn node at a spot visited by the vehicle at the action time of  $C^0$ ; otherwise, create a check node at  $C^0$ .

Step 2.3. If  $C^0$  is 'attach', then

create a room node at  $C^0$ .

Step 2.4. If  $C^0$  is 'another trip', then

increment the value of the counter of the initial node by one.

Step 3. Repeat Step 2 until all commands in the input are processed.

In the above algorithm, the threshold value of one meter may be varied to meet other application needs. And the counter value of the initial node is used to represent the number of times of one-way learning during a learning session for the area mode. Besides, it can be seen from the algorithm that the resulting path map is a simple attributed undirected graph, in contrast to the created graph for the single path mode. An example to illustrate this graph is shown in Figure 4.4.



# Chapter 5

## Vehicle Navigation in Indoor Environment

### 5.1 Introduction

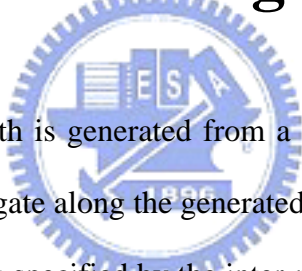
In the navigation stage, the main task is to guide the vehicle to follow a selected path generated from a learned path map. The path consists of a set of connected nodes. In this chapter, two vision-based navigation strategies based on fuzzy guidance are proposed to traverse among these nodes in two types of navigation modes.

Firstly, two navigation methods for the single path mode, namely line following and smooth curve following, are proposed for straight-line and turning-section navigations, respectively. The vehicle can use these navigation methods to traverse among the nodes within a created path map generated from the learning process for the single path mode. These topics will be described in Section 5.2.

Secondly, in order to navigate in the area mode, certain methods are proposed. A transformation method is proposed to convert high-level user commands into a data list with useful information for navigation, according to the learned data generated from the learning process for the area mode. After that, a dynamic path planning method using the Dijkstra algorithm is proposed to generate a navigation path dynamically. Finally, a navigation method is proposed for the vehicle to traverse along the generated path. These topics will be described in Section 5.3.

In addition, a strategy for navigation accuracy maintenance is proposed in Section 5.4. Since the proposed collision-avoiding guidance strategy is based on a computer vision technique, the proposed navigation method regards objects appearing along navigation routes as obstacles, which should be avoided. Then certain path deviations may occur after the vehicle avoids some obstacles in navigation sessions. For this reason, a navigation accuracy maintenance method should be designed for correcting this kind of error, as is done in this study. More details about this method will be described later.

## 5.2 Navigation in Single Path Mode



After a single learned path is generated from a learning process, the remaining work is for the vehicle to navigate along the generated path by visiting each path node sequentially through the routes specified by the inter-node edges. Since a created path is composed of two kinds of nodes, check node and turn node, the vehicle will navigate along the path in four distinct cases, check node to check node, check node to turn node, turn node to check node, and turn node to turn node. Except the case of turn node to turn node, which is a turning section, the others are all straight-line sections. In the following, two navigation methods, line following and smooth curve following, are proposed for straight-line sections and turning sections, respectively. Finally, the detail procedure for accomplishing a navigation task in single path mode will be described.

## 5.2.1 Navigation Strategy for Straight-Line Sections

The navigation strategy of line following is adopted when the vehicle passes a check node in its navigation path. The vehicle uses mainly the fuzzy guidance technique described in the Section 3 during the line following process. The details are described as an algorithm as follows.

**Algorithm 3.** *Line-following navigation process.*

- Step 1.** Turn the vehicle toward the check node  $N_c$  to be reached.
- Step 2.** Calculate the distance from the current vehicle location  $L_v$  (recorded in the odometer) to the location of  $N_c$  (recorded in the learned data), and denote the distance by  $d_1$ .
- Step 3.** Move the vehicle forward.
- Step 4.** Turn accordingly if the output steering angle of the fuzzy guidance algorithm is larger than zero.
- Step 5.** Read the odometer to get the current vehicle location  $L_v'$  and compute how far the vehicle has moved as  $d_2 = |L_v' - L_v|$ .
- Step 6.** If the difference  $|d_1 - d_2|$  is smaller than 5 centimeters, then end this navigation session, and turn the vehicle to the original direction of  $N_c$  specified in the learned data; otherwise, repeat Step 3 through Step 5.

By this line following algorithm, the vehicle can accomplish a navigation process in a straight-line section. But in certain situations, an unexpected obstacle might appear only just in front of the vehicle and its size might be too big, leading to the result that the vehicle cannot avoid collision with this obstacle by the proposed

fuzzy guidance technique. Then the vehicle will stop right away to terminate the navigation task and return an error message to the system. On the contrary, as long as a steering angle can be acquired from the proposed fuzzy guidance technique, the vehicle will correct its direction according to this steering angle to find a suitable way to avoid the obstacle.

## 5.2.2 Navigation Strategy for Turning Sections

The navigation strategy of curve following is adopted when the vehicle passes a turn node (called *initial* turn node) and moves toward another turn node (called *end* turn node) in its navigation path. The proposed curve following method uses the information of the global coordinates of the two turn nodes as well as their direction angles to make a smooth turn. An illustration of the proposed curve following technique is shown in Fig. 4. The main concept behind is to move the vehicle along a well-planned curve joining the two turn nodes by setting different speeds for the two wheels individually. An algorithm for the method is described as follows

**Algorithm 4.** *Smooth Curve following navigation process.*

**Step 1.** When the vehicle arrives at the initial turn node denoted as  $N_t^1$ , get the global coordinates  $(x_1, y_1)$  and the direction angle  $\theta_1$  from the odometer, and convert  $\theta_1$  into a vector  $\vec{v}_1$  using the following equation:

$$\vec{v}_1: \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} = \begin{pmatrix} \sin(-\theta_1) \\ \cos(-\theta_1) \end{pmatrix}.$$

**Step 2.** Get the global coordinates  $(x_2, y_2)$  and the direction angle  $\theta_2$  of the end turn node  $N_t^2$  from the learned data, and convert  $\theta_2$  into a vector  $\vec{v}_2$  in a way

similar to that for computing  $\vec{v}_1$ . Also, compute a vector  $\vec{v}_3 = \begin{pmatrix} c \\ d \end{pmatrix}$  which is perpendicular to  $\vec{v}_2$ .

**Step 3.** Compute the middle point  $(X_c, Y_c)$  between  $(X_1, Y_1)$  and  $(X_2, Y_2)$ .

**Step 4.** According to geometry theory, compute a unique circle with center  $(X_o, Y_o)$  using the two points  $(X_1, Y_1)$  and  $(X_2, Y_2)$  by the following equation:

$$X_o = X_2 + cS, \quad Y_o = Y_2 + dS$$

where

$$S = \frac{(X_c - X_2)(X_2 - X_1) + (Y_c - Y_2)(Y_2 - Y_1)}{c(X_2 - X_1) + d(Y_2 - Y_1)}.$$

**Step 5.** Compute a vector  $\vec{v}_4$  by the following equation:

$$\vec{v}_4 = \begin{pmatrix} -\left(\frac{Y_o - Y_1}{X_o - X_1}\right) \\ 1 \end{pmatrix} = \begin{pmatrix} X \\ Y \end{pmatrix}.$$

**Step 6.** Convert  $\vec{v}_4$  into an angle  $\theta_3$  by the following equation:

$$\text{if } Y < 0 \text{ and } X > 0, \text{ then set } \theta_3 = -180 - \tan^{-1}\left(\frac{X}{Y}\right);$$

$$\text{if } Y < 0 \text{ and } X < 0, \text{ then set } \theta_3 = 180 - \tan^{-1}\left(\frac{X}{Y}\right);$$

$$\text{otherwise, set } \theta_3 = -\tan^{-1}\left(\frac{X}{Y}\right).$$

**Step 7.** Turn the vehicle leftward for the angle of  $\theta_3$  if  $\theta_3$  is larger than zero; otherwise, turn the vehicle rightward for the angle of  $\theta_3$ .

**Step 8.** Set the speeds of the two wheels individually by the following equations:

$$\frac{R_1}{R_2} = \frac{V_L}{V_R}, \quad R_2 - R_1 = w$$

where  $V_L$  and  $V_R$  are the speeds of the left wheel and the right one,

respectively;  $R_2$  and  $R_1$  are the distances from the center of the circle to the left wheel and the right one, respectively; and  $w$  is the width of the vehicle.

(An illustration of the situation and the notations is shown in Fig. 5.1.)

**Step 9.** Start the curve following action until the end turn node is reached.

**Step 10.** Turn the vehicle to the original direction specified in the learned data, and finish the turning session.

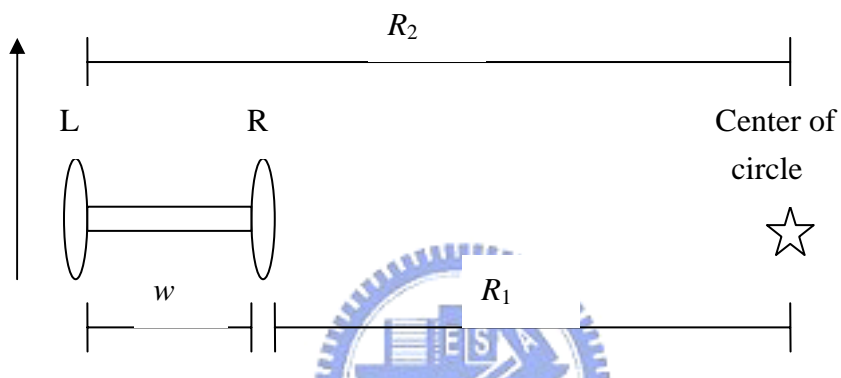


Fig. 5.1 : An illustration of computing two wheel speeds for the case of turning rightward in curve following.

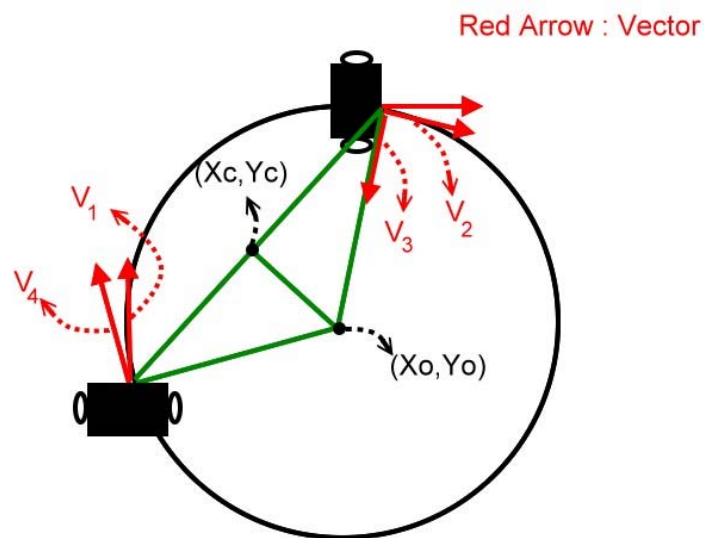


Figure 5.2 A figure illustrating curve following process.



Since the smooth curve following process uses no computer vision technique, the vehicle will lose the collision avoidance capability during a curve following process. Therefore, we use an additional method to improve this process before entering a turning section.

An image will be captured before a curve following process is executed. We first define a rectangular window in a captured image, as shown in Fig. 5.2. Then the proportion of the route area in the window is defined to be an indicator, which is used to indicate whether the curve following process can be executed or not. If the value of the indicator is higher than 90%, it means that the curve following process can be executed; otherwise, cannot.

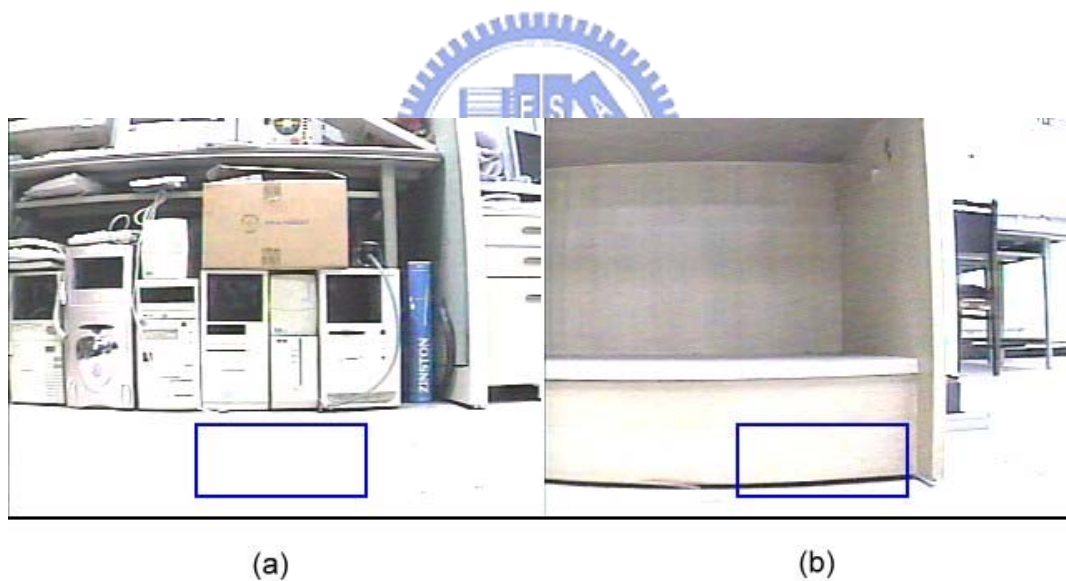
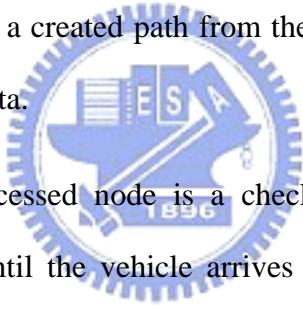


Figure 5.3 A figure illustrating the check process before curve following process  
(a) A case that the curve following process can be executed (b) A case that the curve following process can not be executed

## 5.2.3 Process of Proposed Navigation in Single Path Mode

With the methods proposed previously, including the line following method for straight-line sections and the smooth curve following method for turning sections, a navigation procedure for the single path mode can be designed for the purpose. Several steps of the process proposed in this study are described as follows, and its corresponding figure is shown in Figure 5.3.

**Algorithm 5.** *Proposed process of navigation in single path mode.*

- 
- Step 1.** Scan the node list of a created path from the starting point sequentially and read its associated data.
- Step 2.** If the currently processed node is a check node, then perform the line following process until the vehicle arrives at this node. And perform the corresponding navigation accuracy maintenance strategy to fix certain navigation errors.
- Step 3.** If the currently processed node is a turn node, then take the following action.
- If the next processed node is a turn node also, perform the line following process until the vehicle arrives at this turn node. And perform the smooth curve following process until the vehicle arrives at the next turn point.
  - Otherwise, perform the navigation accuracy maintenance strategy for the smooth curve following process to fix certain navigation errors.
- Step 4.** If an ending point flag is found with the currently processed node, then end

this navigation trip.

Note that the navigation accuracy maintenance strategy performed in Step 3 and 4 will be described in Section 5.4.

## 5.3 Navigation in Area Mode

After a learning process for navigation in the area mode is completed, a generated path map is only composed of a set of connected nodes with undirected inter-node edges. Without a starting spot and an ending one, a navigation path cannot be generated for the vehicle to accomplish the navigation task. Therefore, a process is designed for this purpose, which generates essential data for later path planning, including a set of starting and ending spots and a set of weights of the inter-node edges. When these essentials data are acquired from the above transformation process, the Dijkstra algorithm is used to plan a shortest path for later navigation task. After that, the proposed navigation method can be employed to traverse the navigation path.

### 5.3.1 Generation of essential data for path planning

When users want to order the vehicle to carry out a navigation task in the area mode (e.g. from a bedroom to a living room), he/she only needs to press two buttons on the user interface designed in this study to choose a start room and an end room. Afterward, before a navigation path is planned using the Dijkstra algorithm, a generation process is designed for computing a weight for each inter-node edge and identifying the connectivity between each node pair. If the weight of an inter-node edge between two nodes is equal to '0', that means the two nodes are disconnected,

otherwise, the two nodes are connected. An algorithm designed for this purpose is described as follows.

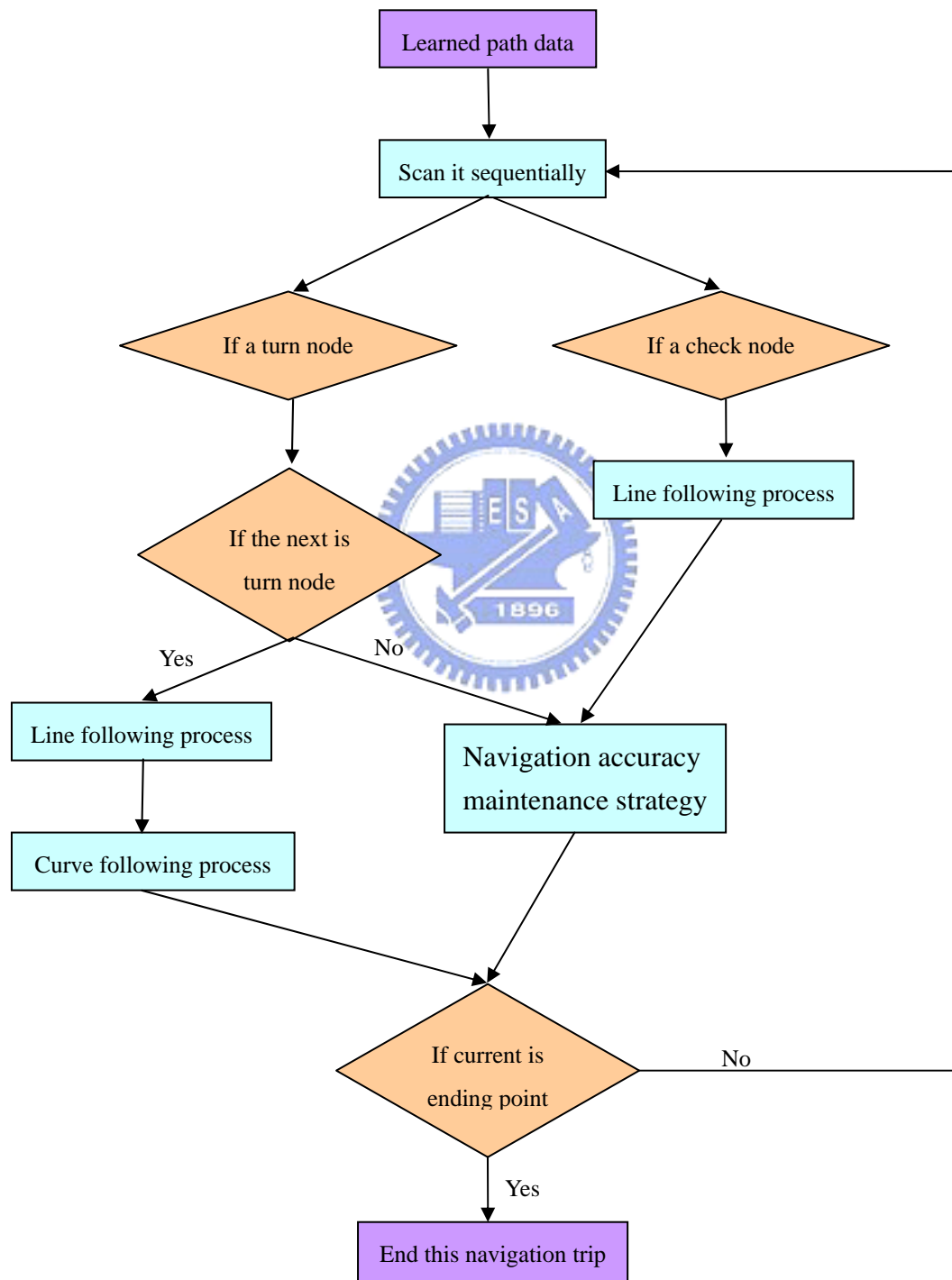


Figure 5.4 A figure to illustrate a process of proposed navigation procedure in single path mode.

**Algorithm 6.** *A process of essential data generation.*

**Step 1.** Take a learned path map as input, and identify the weight and connectivity of each inter-node edge by processing all nodes within the learned path map sequentially. Let the currently processed node be denoted as  $N^0$ , the previously processed one as  $N^{-1}$ , and the next processed one as  $N^{+1}$ .

**Step 1.1.** If  $N^0$  is a “check node”, then do the following.

- a. If  $N^{-1}$  is an “initial node” or a “check node”, calculate the distance between  $N^{-1}$  and  $N^0$  and mark this distance as a weight of the edge between the two nodes. Otherwise, set the weight of the edge between  $N^{-1}$  and  $N^0$  to be 0.
- b. If  $N^1$  is an “initial node” or a “check node”, calculate the distance between  $N^1$  and  $N^0$  and mark this distance as a weight of the edge between the two nodes. Otherwise, set the weight of edge between  $N^1$  and  $N^0$  to be 0.

**Step 1.2.** If  $N^0$  is a “turn node”, then

- a. Calculate the distance between  $N^{-1}$  and  $N^0$ ,  $N^1$  and  $N^0$ , and mark these two distances as the weights of the edges.
- b. Calculate each distance between  $N^0$  and the other nodes in the path map, and if certain distance is within a predefined range, mark this distance as the weight of the edge between  $N^0$  and the node. Otherwise, set the weight of the edge to be 0.

**Step 1.3.** If  $N^0$  is a “room node”, then

- a. Calculate the distance between  $N^{-1}$  and  $N^0$  only, and mark this distance as the weight of the edge between  $N^{-1}$  and  $N^0$ .

**Step 2.** Repeat Step 1 until all input nodes are processed.

After the above algorithm is performed, the data of the learned path map is more

complete, including a set of nodes and the weights of the inter-node edges. An example of the resulting graphs is shown as follows to illustrate the result of performing this algorithm.

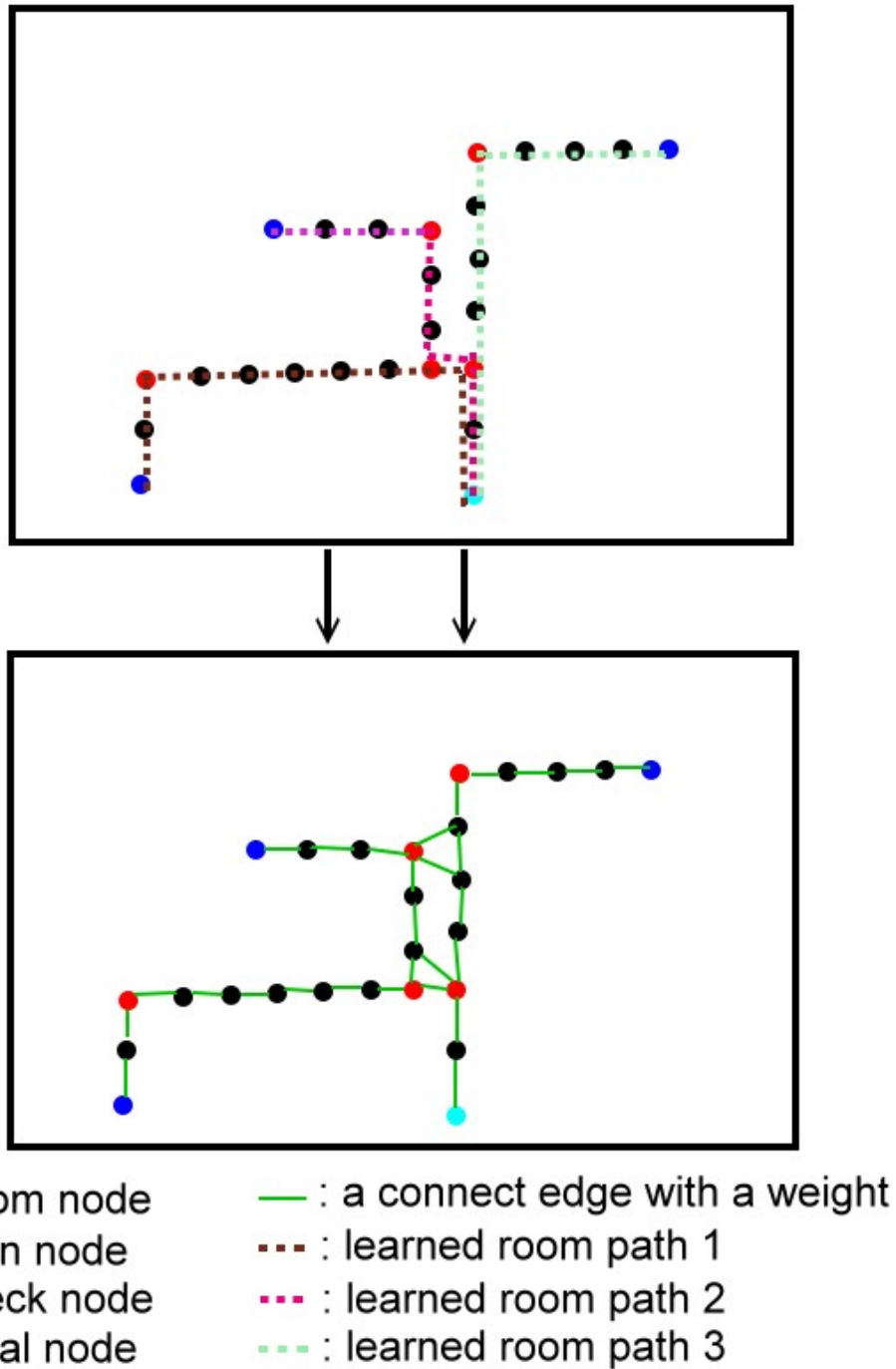


Figure 5.5 An example to illustrate result of performing the proposed algorithm.

## 5.3.2 Dynamic Path Planning using Dijkstra

### Algorithm

After the weight of each inter-node edge in a learned path map is created, regard each weight as a cost value. Then we can think the path planning problem as a minimal-cost graph search problem. The demand in our work is to search a shortest path with the minimal cost if a start point and an end point are given. For this purpose, the Dijkstra algorithm, which can solve a single source shortest path problem, is employed in our work. The Dijkstra algorithm solves the single-source shortest paths problem on a weighted directed graph, with the weights of all the edges being non-negative. Given a desired start node, the Dijkstra algorithm searches the graph using a way similar to both the breadth-first search and the Prim algorithm for computing a minimum spanning tree, and plans a shortest path from the source to each other node.

A review of the principle of the Dijkstra algorithm is as follows. The Dijkstra algorithm maintains a set  $S$  of nodes, which have shortest paths from the source  $N$  to all the nodes which have already been found. Initially, the costs of a given start node, say  $N$ , to all the other nodes are set to infinity. At the beginning,  $S$  contains  $N$  only. Then, the node  $C$ , which is not in  $S$  yet and has the minimum cost, is added into  $S$  at each search iteration. And for the nodes not in  $S$ , the costs of those *incident* nodes from  $C$ , which means those with emitting directed links to  $C$ , are updated if the following condition is satisfied:

$$Cost(N,C)+Cost(C,M)<Cost(N,M) \quad (5.1)$$

where  $M$  is one of the nodes which is incident from  $C$ . and

$Cost(N,C)$  = the cost of the path from N to C so far,

$Cost(C,M)$  = the cost of the directed link from C to M

$Cost(N,M)$  = the cost of the path from N to M so far.

If the inequality (5.1) is satisfied, update the cost of the current shortest path to M with  $Cost(N,C)+Cost(C,M)$ . Besides, add the node M to the node sequence of the current shortest path from N to C. End the search process when the desired goal is added into S.

The desired shortest path is found in a form of a node sequence. More details about the Dijkstra algorithm can be found in Cormen [9].

### 5.3.3 Navigation Strategy among Room Nodes

After a navigation demand from one room to another is given, a path is planned dynamically using the above-mentioned methods, and the remaining task is to guide the vehicle to navigate along the planned path. And this work can be accomplished using two navigation strategies proposed in this study, which are the same as those in the single path mode, namely, the strategies of line following and smooth curve following.

But, a little bit difference still exists in the smooth curve following process between two kinds of navigation modes. In the previous section, it was mentioned that both directions of the initial turn node and the ending turn node are essential data for accomplishing the smooth curve following process. And these directions are recorded during a learning process in the single path mode. On the contrary, no direction data are retrieved from the odometer during the learning process in the area mode, and so no direction data are recorded with each node in the created path map consequently. As a result, if we want to use the smooth curve following process during a navigation



task in the area mode, the direction data of each path node must be created manually.

A process of direction data creation is described as follows.

**Algorithm 7.** *Process of direction data creation*

**Step 1.** Scan each path node with its coordinate data orderly from a start node. Let the currently processed node be denoted as  $N^0$ , the previously processed one as  $N^{-1}$ , and the next processed one as  $N^{+1}$ .

**Step 1.1.** Set the direction value of the start node to zero.

**Step 1.2.** If the difference in distance between the X coordinate value and the Y coordinate value of  $N^0$  and  $N^1$  are both smaller than a predefined threshold value  $Th_c$ , regard the direction of  $N^0$  to be equaled the direction to  $N^1$ .

**Step 1.3.** If the difference in distance between the X coordinate value or the Y coordinate value of  $N^0$  and  $N^1$  is larger than a predefined threshold value  $Th_c$ , regard  $N^0$  and  $N^1$  as two “turn nodes” and that the direction value of  $N^0$  and the direction of  $N^1$  are perpendicular to each other. An illustrative example is shown in Figure 5.6.

**Step 2.** Repeat Step 1 until all input nodes are processed.

After the type of each path node and its associated direction data are created by the above algorithm, the navigation task in the area mode can use the same navigation methods as those used in the single path mode.

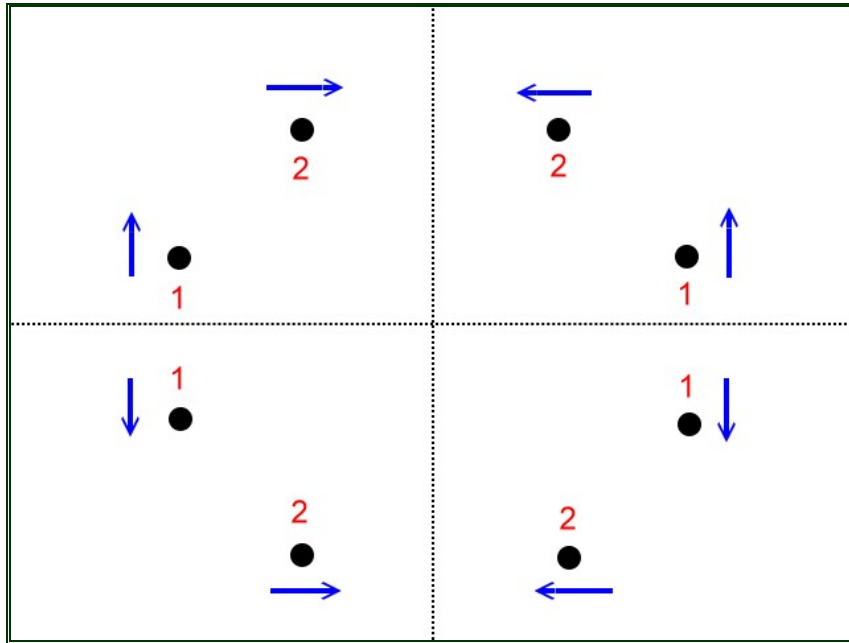


Figure 5.6. An illustrative example to show the direction creation of two turn nodes, where 1 represents node  $N^1$  and 2 represents node  $N^0$



### 5.3.4 Process of Proposed Navigation in Area Mode

With the previously-described methods, including a method for generating essential data for path planning a dynamic path planning method, and a navigation strategy, the proposed navigation process for the area mode now can be described as follows, and its corresponding figure is shown in Figure 5.7.

**Algorithm 8.** *Process of proposed navigation in area mode.*

**Step 1.** Analyze a request from users to get a start node and an end one of a given navigation task.

**Step 2.** Generate essential data, including the weight of each inter-node edge in the learned path map for later path planning.

- Step 3.** Take as input the start node and the end one to generate a shortest navigation path by the Dijkstra algorithm. If the number of the user requests is larger than one at a time, then take as input the start nodes and the end nodes orderly to get multiple paths.
- Step 4.** Perform the direction creation process to create the direction of each node in the navigation paths which are the output from Step 3.
- Step 5.** Start to navigate along the created navigation paths using Algorithm 5.
- Step 6.** Repeat Step 5 until all required rooms are visited.

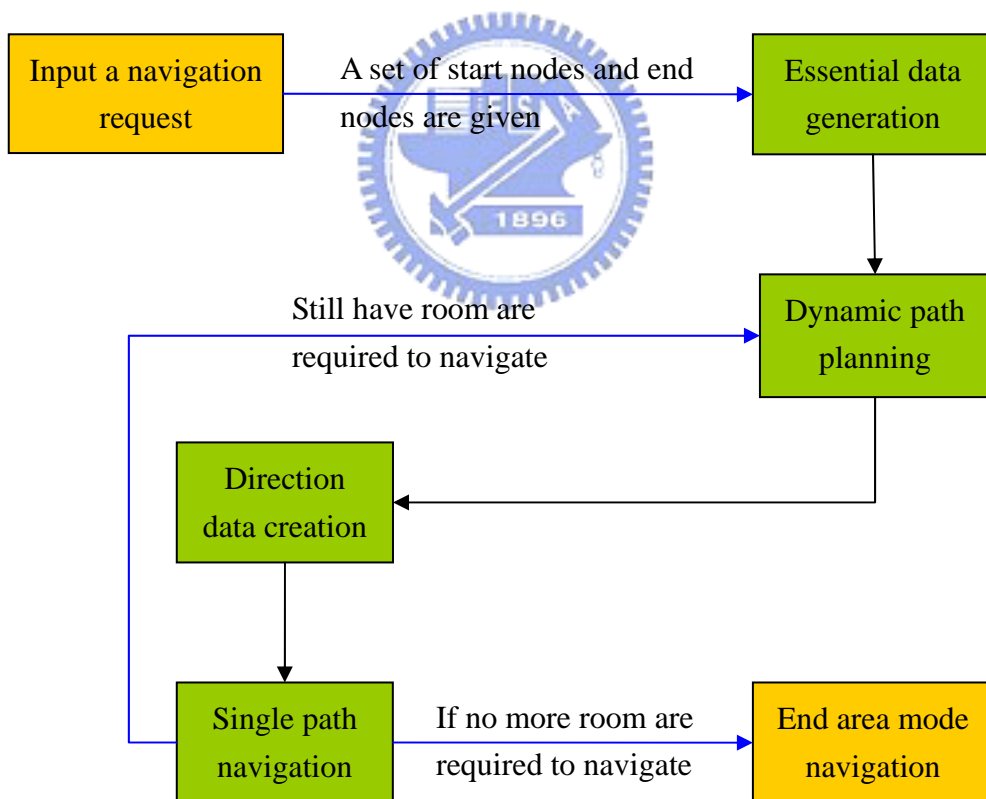


Figure 5.7 A figure to illustrate proposed navigation process in the area mode

## 5.4 Strategy for Fuzzy Navigation Accuracy Maintenance

With the navigation strategies mentioned in Section 5.2 and Section 5.3, the vehicle can accomplish any navigation task in the single path mode and the area mode. But it does not mean that the vehicle will navigate correctly all the time; certain errors may occur during the navigation session. For instance, when the vehicle tries to avoid an unexpected collision, it may deviate from its original path. If no accuracy maintenance method is performed, the navigation error will accumulate continuously with possible failures in the navigation task . Therefore, four methods for fuzzy navigation accuracy maintenance are proposed in this study as described in the following.



### 5.4.1 Comparison of Navigation Distances

When the vehicle navigates in a straight section, which is from one check node  $C^1$  to another  $C^2$ , the condition of ending the section is that the navigation distance is equal or larger than the learned distance of this section, where the navigation distance  $N_d$  is obtained from the odometer and the learned distance  $L_d$  is calculated from the distance between  $C^1$  and  $C^2$ . Therefore, a threshold  $th_d$  is defined for the comparison of the navigation distance and the learned distance, and the comparison is performed according to the following inequality:

$$|N_d - L_d| < th_d .$$

If the value of  $N_d$  satisfies this inequality, then the vehicle will stop going forward and

end this section. So we can maintain the location precision of the line-following navigation. Note that the threshold value  $th_d$  is predefined and must be a suitable one, because larger  $th_d$  is meaningless and may cause larger accumulative errors in each navigation section, and smaller  $th_d$  may cause a problem that the vehicle may never stop to enter the next navigation session.

## 5.4.2 Comparison of Directions

Before entering the next navigation session, a check of navigation accuracy maintenance will be performed when a navigation section is accomplished. The direction data of each learned node can be used to fix the direction of the vehicle at the end of every navigation session. When the vehicle arrives at one node in the navigation, the direction  $\theta_N$  of the vehicle at this time and the direction  $\theta_L$  of the learned node can be obtained from the odometer and the learned data, respectively. And in order to fix the direction  $\theta_N$  to be identical to the direction  $\theta_L$ , we use the following equation:

$$\theta_F = |\theta_N - \theta_L|$$

where  $\theta_F$  is the fixing angle to be turned. After fixing the direction of the vehicle, it will enter the next session. Without performing this check, the vehicle will deviate from its trajectory and get lost gradually.

## 5.4.3 Smooth Curve Following Process

Before entering a turning section, errors might be accumulated from the previous

navigation section, because a few errors may occur after accomplishing each straight line section. So when the vehicle arrives at the “initial turning point,” the location of the vehicle at this time may not be close to the location of the learned data at this node. But the proposed curve following process is still working at this time. Because the proposed curve following process is performed only with the requirement of correct global coordinates and the direction of the “end turning node,” it can be utilized to achieve the goal of navigation accuracy maintenance. An illustrative example is shown as Figure 5.5.

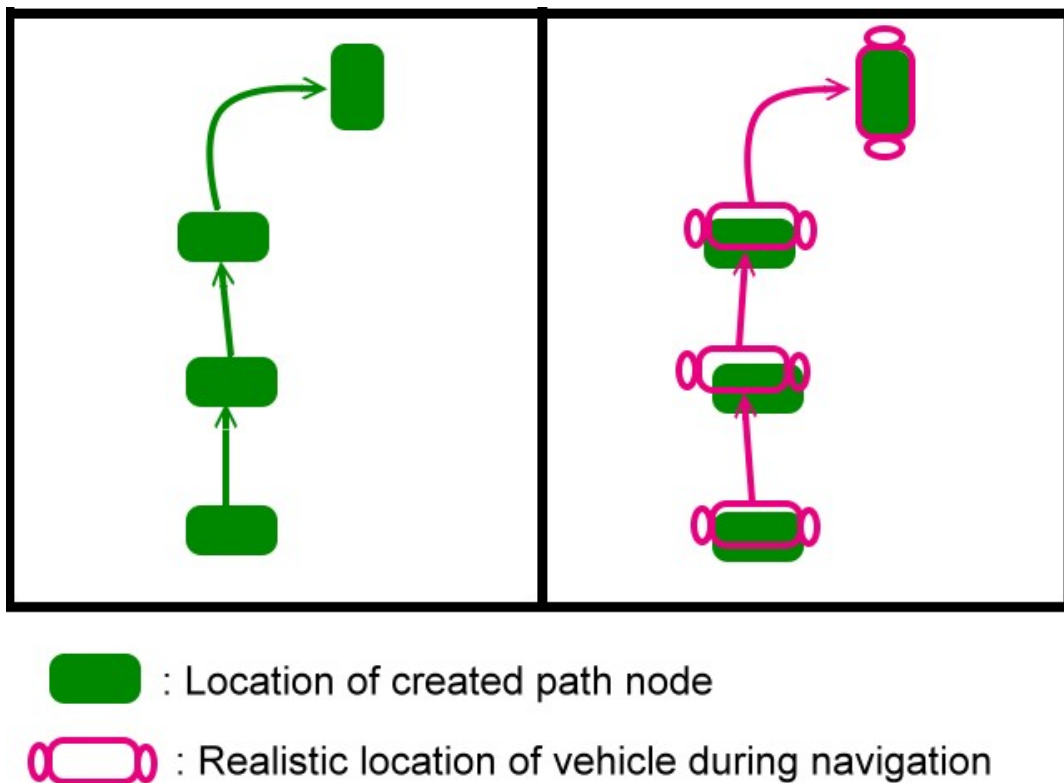


Figure 5.8 An example to illustrate the usability of smooth curve following for navigation accuracy maintenance.

## 5.4.4 Adjustment of Beginning Position

When a navigation task from one room to another is finished, the position of the vehicle may not be close to the destination of the last navigation task, because certain collision avoidance or wheel drifting may occur during the navigation process. If a navigation task is given another destination at this time, the position of the vehicle at this time will be regarded as a new beginning. And the new navigation task may fail due to a deviation of the position between the new beginning and the beginning of the newly planned path. Therefore, an adjustment method is designed to adjust the position of the beginning of the newly planned path with the result that can make the deviation to cause no influence on the next navigation task. The method is described as an algorithm in the following.



**Algorithm 9.** *Adjustment of beginning position.*

- Step 1.** Record the real position of the vehicle when a navigation task is just finished, and denote it as  $(X_{\text{real}}, Y_{\text{real}})$ .
- Step 2.** Read the position of the end node of the finished navigation task, and denote it as  $(X_{\text{old}}, Y_{\text{old}})$ .
- Step 3.** Calculate the X and Y differences between  $(X_{\text{real}}, Y_{\text{real}})$  and  $(X_{\text{old}}, Y_{\text{old}})$  in terms of the Euclidean distance, and denote them as  $(d_x, d_y)$ .
- Step 4.** Read the position of new beginning of the next navigation task, and denote it as  $(X_{\text{new}}, Y_{\text{new}})$ .
- Step 5.** Add  $(d_x, d_y)$  to  $(X_{\text{new}}, Y_{\text{new}})$  to get  $(X_{\text{new}+d_x}, Y_{\text{new}+d_y})$  as a beginning of the new navigation task.

# Chapter 6

## Person Following in Corridors

### 6.1 Introduction

In the application of person following, the proposed fuzzy guidance technique is used to guide the vehicle to navigate in corridors. Besides, an unexpected object may appear in the corridor when the vehicle navigates along the corridor. And the unexpected object might be just a static obstacle or a moving object, like a human. The goal of this application is to detect the unexpected object appearing in the corridor and identifying it as a human or not, and then perform the person following process if a human is identified.

The proposed person following process consists of three stages, image preprocessing, human identification, and human tracking. In the stage of image preprocessing, several image processing techniques are integrated to detect an important feature of the unexpected object appearing in the corridor. The feature is useful information to identify the object as a human or not. More detail about the stage will be described in Section 6.2.

After the essential feature is obtained from the previous stage, a strategy is proposed to detect humans in every navigation cycle. Analyzing the feature of a human's feet is the main idea of this strategy because the feet of a human are a common feature in general cases. The stage will be described in Section 6.3.

After an unexpected human is detected, the proposed person following process is performed until the human disappears within the eyesight of the vehicle. Because the



person following process is combined with the proposed fuzzy guidance technique, consideration of simultaneous use of the two proposed techniques is necessary. Otherwise, certain tracking or guidance errors might be incurred. More details about the topic will be described in Section 6.4.

## 6.2 Image Preprocessing for Unexpected Object Detection in Corridors

Generally speaking, the scene of a corridor is composed of three parts, ground, wall, and baseline. An example of corridor images grabbed in our experimental environment is shown in Figure 6.1(a). Since the camera is onboard at a very low position, the ratio of the ground region in the captured image is larger than those captured at higher views. Moreover, an object is regarded as standing on the ground usually. Therefore, we focus on searching unusual objects appearing on the ground in corridors. Nevertheless, since the eyesight of the vehicle is not so far due to the performance of the wireless camera on the vehicle, an object looks like very small when the distance between it and the vehicle is too long. Hence, a valid search distance is also defined to increase the detection probability.

In this section, an image analysis algorithm is designed for detecting an unexpected object appearing on corridor grounds. The goal of this algorithm is to detect the physical region of an unexpected object and analyze its lower part to extract a feature, called *Object Foot*. This feature can be employed to identify an unexpected object as a human or not, as described in Section 6.3. The algorithm is described as

follows.

**Algorithm 4.** *Detection of Object Foot.*

- Step 1.** Convert an input image into a binary one with a threshold  $T_b$ , which is defined in advance. An illustrative example is shown in Fig. 6.1(c).
- Step 2.** Find the *connected components* in the binary image to search two approximation lines of the lower baseline at both sides and mark them as red lines in the image. An illustrative example is shown in Fig. 6.1(d).
- Step 3.** Scan the binary image from left to right and from bottom to each of the two marked red lines vertically, respectively. Record the image coordinates of each point when the scan line encounters a white point. An illustrative example is shown in Fig. 6.1(e).
- Step 4.** At each point recorded in the previously step, find the *connect components* to search white points. If the number of found connected points is larger than the predefined value in a search time period, then the set of connected points will be regarded as an *Object Foot*. An illustrative example is shown in Fig. 6.1(f).
- Step 5.** Perform Step 1 through Step 4 recursively until all *Object Foots* are found in the image.

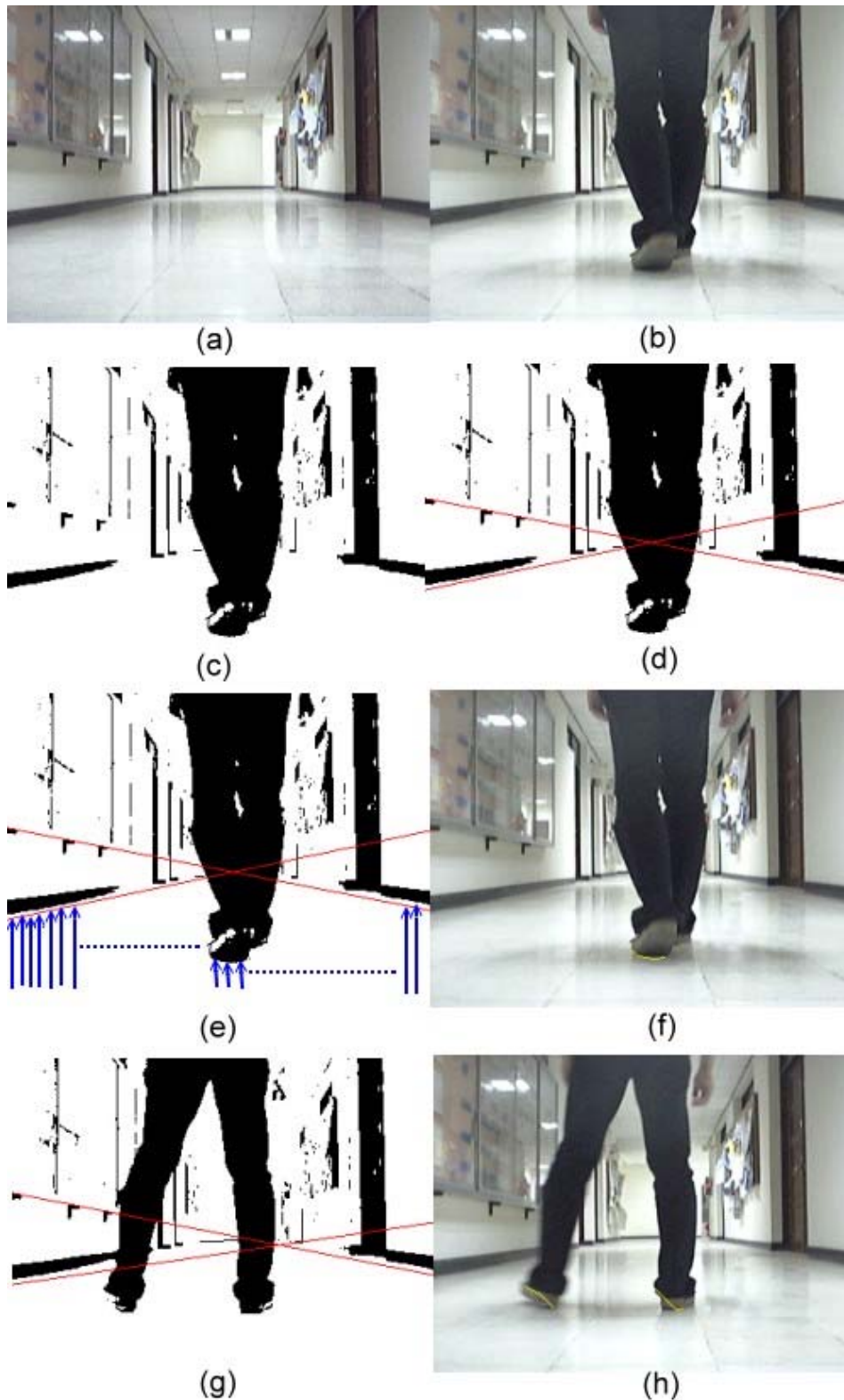


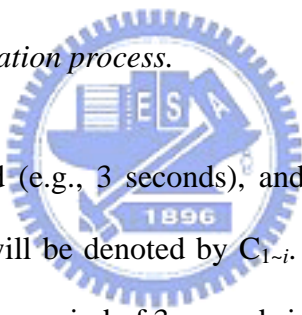
Figure 6.1 Images of several image processing results (a) A corridor image. (b) An unexpected object is appeared. (c) Binary thresholding of (b). (d) Two approximation lines are found. (e) Scan the lines vertically from bottom to the approximation lines. (f) An *Object Foot* is found (yellow points). (g) Another example. (h) Two *Object Foots* are found.

## 6.3 Human Identification

After the feature of an unexpected object, *Object Foot*, is obtained through the image analysis algorithm described in Section 6.2, the next job is to analyze the characteristic of the *Object Foot* feature for deciding the unexpected object as a human or not. Therefore, an identification algorithm is proposed for the purpose.

The concept of this identification algorithm is to analyze the variation of *Object Foot values* in a predefined time period. And the idea is based on a common sense, “humans have feet surely.” The details of the identification algorithm are described as follows.

**Algorithm 5.** *Human identification process.*

- 
- Step 1.** Define a time period (e.g., 3 seconds), and transform it into a number of cycle times, which will be denoted by  $C_{1\sim i}$ . For example, if a cycle time is 0.5 second, then a time period of 3 seconds is regarded as six cycles, denoted by  $C_{1\sim 6}$ .
- Step 2.** Perform the above-mentioned image analysis algorithm to obtain the *Object Foot* feature in each cycle in the time period  $C_{1\sim i}$ . Denote the *Object Foot* feature with its index at the corresponding cycle by  $O_i^j$ , where  $j$  denotes the index of *Object Foot* and  $i$  denotes the index of the cycle. For example, if the index of *Object Foot* at the third cycle is 2, then it will be denoted by  $O_3^2$ .
- Step 3.** If the total number of  $O_i^2$  is larger than that of  $O_i^1$  in the time period  $C_{1\sim i}$ , then decide that the unexpected object is a human; otherwise, not a human.

According to our experimental results, the index  $i$  of the  $C_{4-i}$  is suggested to be larger than 4. Then the probability of a successful human identification will become higher.

## 6.4 Human Tracking by Feet Positions

After an unexpected person is detected, a human tracking method is proposed for the vehicle to track the person until he/she disappears in the view of the vehicle. Moreover, besides the works of image analysis and human identification, human tracking is another important work which belongs to the proposed person following process. Hence, the detail of the human tracking method will be described in the following.

The essence of the method is to track the feet positions of the unexpected person. Therefore, *Object Foot*, can be employed as a tracking target. Then, an angle between two lines, a forward direction of the vehicle and a direction line from the tracking target to the vehicle, is calculated for steering the vehicle to track the person. But the number of the detected *Object foot* of a human may be one or two in each navigation cycle sometimes. The tracked target might also be changed in two different types. In the proposed tracking process, the middle point of a connected line of two *Object foots* is regarded as a type of tracking target; otherwise, the middle point of only one *Object Foot* is regarded as another type of tracking target. An illustrative example of tracking targets of the two types is shown in Figure 6.2. The tracked target is projected to a *vehicle coordinate system* before calculating the angle in between, and a

distance between the tracked target and the vehicle is also calculated.

After the steering angle is calculated, another problem is to define a suitable cycle time of navigation. As mentioned previously, the fuzzy guidance technique is proposed for vehicle navigation in corridors. Therefore, two processes, fuzzy guidance and human tracking, must be accomplished in a navigation cycle. To define a suitable cycle time is important for smoother navigation and keeping the direction of the vehicle stable. For this reason, the predefined cycle time in our system is set to 0.5 second, and satisfactory experimental results could be obtained.

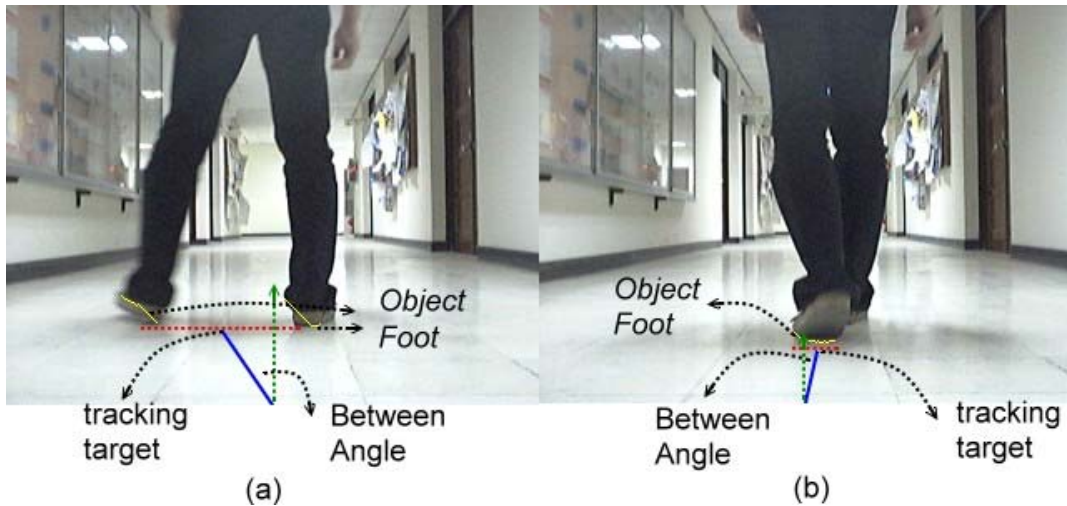


Figure 6.2 An example to illustrate the position of tracking target.

## 6.5 Process of Person Following

By integrating the methods described previously, a process of person following is designed to detect an unexpected object in corridors and identifying the object as a

human or not. Furthermore, the human tracking process is performed after a human is detected. As illustrated in Figure 6.4, the algorithm of the person following process is described as follows.

**Algorithm 6.** *Person following process.*

**Step 1.** Perform the above-mentioned image analysis algorithm to each captured image, and compute the feature *Object Foot* in each navigation cycle.

**Step 2.** Perform the above-mentioned human identification algorithm when a set of *Object Footh* is collected.

**Step 3.** If an human is detected, then do the following steps.

- a. Announce a warning message to the system if this human appears for the first time.
- b. Calculate the steering angle  $\theta_i$  according to the tracked target in each navigation cycle. If the angle is positive, turn the vehicle leftward for the angle; if the angle is negative, then turn the vehicle rightward for the angle.
- c. Calculate the distance  $d_i$  between the tracked target and the vehicle, and if the distance is smaller than a predefined threshold, end the person following process and announce a message to the system.
- d. Perform the fuzzy guidance process.
- e. Repeat Step 1 through Step 3, until the human disappears in the eyesight of the vehicle.



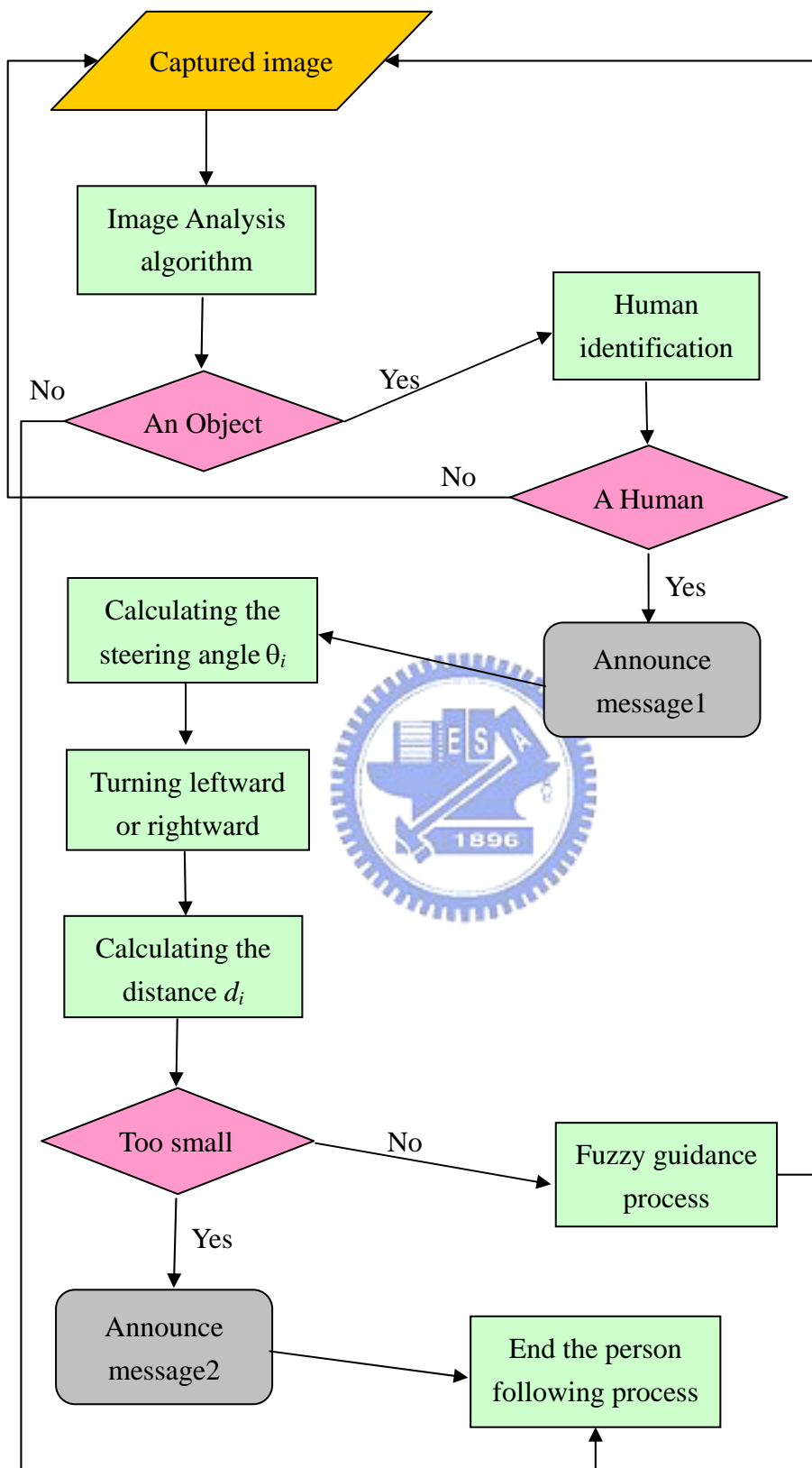


Figure 6.3 A process of person following

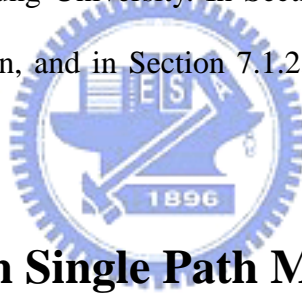


# Chapter 7

## Experimental Results and Discussions

### 7.1 Experimental Results

In this section, we will show certain navigation results in a complicated room environment. All experiments of this study were conducted in a laboratory in a building in National Chiao Tung University. In Section 7.1.1, a navigation result in the single path mode is shown, and in Section 7.1.2, a navigation result in the area mode is shown.



#### 7.1.1 Navigation in Single Path Mode

In the learning stage, we controlled the vehicle to learn a path from one spot to another in our laboratory. The learned path is marked as a yellow dotted line and an example is shown in Figure 7.1. After the manual learning process, the proposed path map creation process was performed to create a path map with a set of nodes and directed edges. The set of nodes for the previous example are also shown in Figure 7.1, in which the directed edges (marked by blue or green arrows) are connected.

In the navigation stage, the vehicle begins to navigate along the learned path autonomously. Two kinds of navigation strategies are performed in corresponding navigation sections. Line following was performed in straight-line sections, which are marked as blue lines in Figure 7.1. Curve following was performed in turning sections,

which are marked as green lines in Figure 7.1. And certain representative images are also shown in Figure 7.1 (numbered from one to sixteen), each of them showing the corresponding situation in the navigation session.

Besides, collision avoidance is another important capability of the proposed vehicle system, and also can be seen in this experimental result. In images (6) and (11) in Figure 7.1, two obstacles were put on the ground and obstructed the learned path in advance before navigation. Several suitable steering angles were calculated by the proposed fuzzy guidance techniques to avoid the obstacles.

The speed of the vehicle used in this navigation is set to be 10 cm/sec. Moreover, the computing time of each navigation cycle is about 0.5 seconds.

### **7.1.2 Navigation in Area Mode**

At the learning stage, we first selected an initial node as a start point, and controlled the vehicle to move from the start point to two spots in our laboratory. These two spots are defined as room 1 and room 2, respectively. The positions of the room nodes, the initial node, and the two learned paths are shown in Figure 7.2. The path map creation process was performed to create a map with a set of nodes and undirected edges.

At the navigation stage, a user command was given, which ordered the vehicle to move from room 1 to room 2 and back to room 1. At the beginning, the essential data generation process was performed to generate the weight of each edge in the learned path map, and the dynamic path planning process was also performed to plan a shortest path from room 1 to room 2. After these works were finished, the vehicle began to navigate along the path. When the vehicle arrived at room 2, the dynamic path planning process was performed to plan a shortest path from room 2 to room 1.

And certain representative images taken in the navigation are shown in Figure 7.3 through Figure 7.7; each of them showing a corresponding situation during the two navigation trips. In Figure 7.3, some pictures of the system interface are shown in Figures 7.3 (1) through (3). The image at the lower left part of the system interface was grabbed with the wireless camera on the vehicle at each navigation cycle. In the meanwhile, the image at the lower right part of the system interface shows a result of fuzzy guidance processing.

Besides, two situations of collision avoidance can be seen in Figures 7.3(8) through (10) and Figures 7.4(22) through (24). Especially in the latter situation, the vehicle was backing to room 1 and close to the destination. Since the position of the room node was very close to a piece of furniture (at about 40 cm aside), when the vehicle got close to the room node, it tried to turn its direction to avoid possible collisions. Therefore, the distance between room 1 and the final stop spot may be larger than expected. This kind of situation can be solved by adding a learning rule, which says that at least one meter between the position of a room node and its surrounding furniture should be maintained.

The speed of the vehicle used in this navigation is set to be 10 cm/sec. Moreover, the computing time of each navigation cycle is about 0.5 seconds.

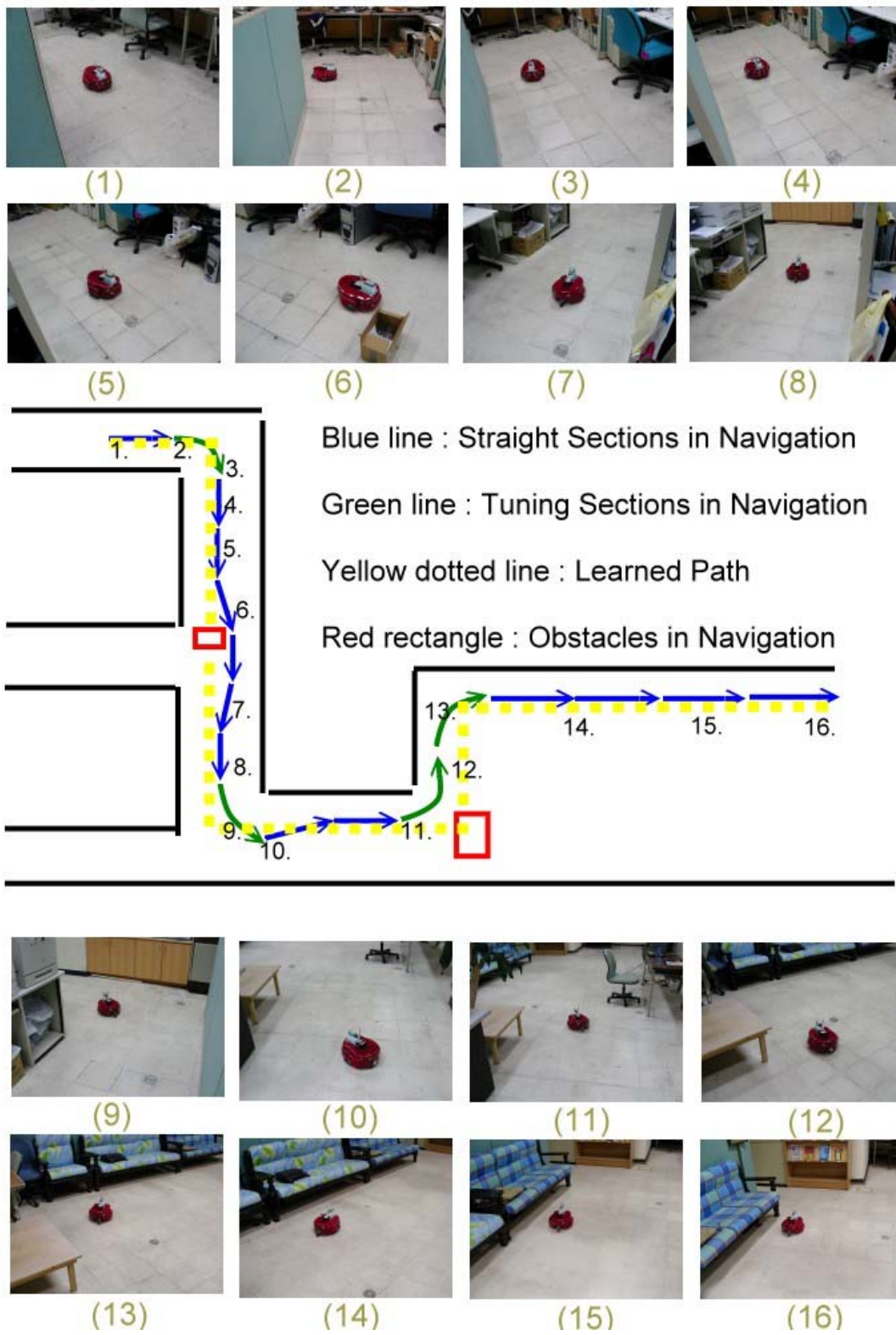


Figure 7.1 An experimental result of navigation in single path mode.

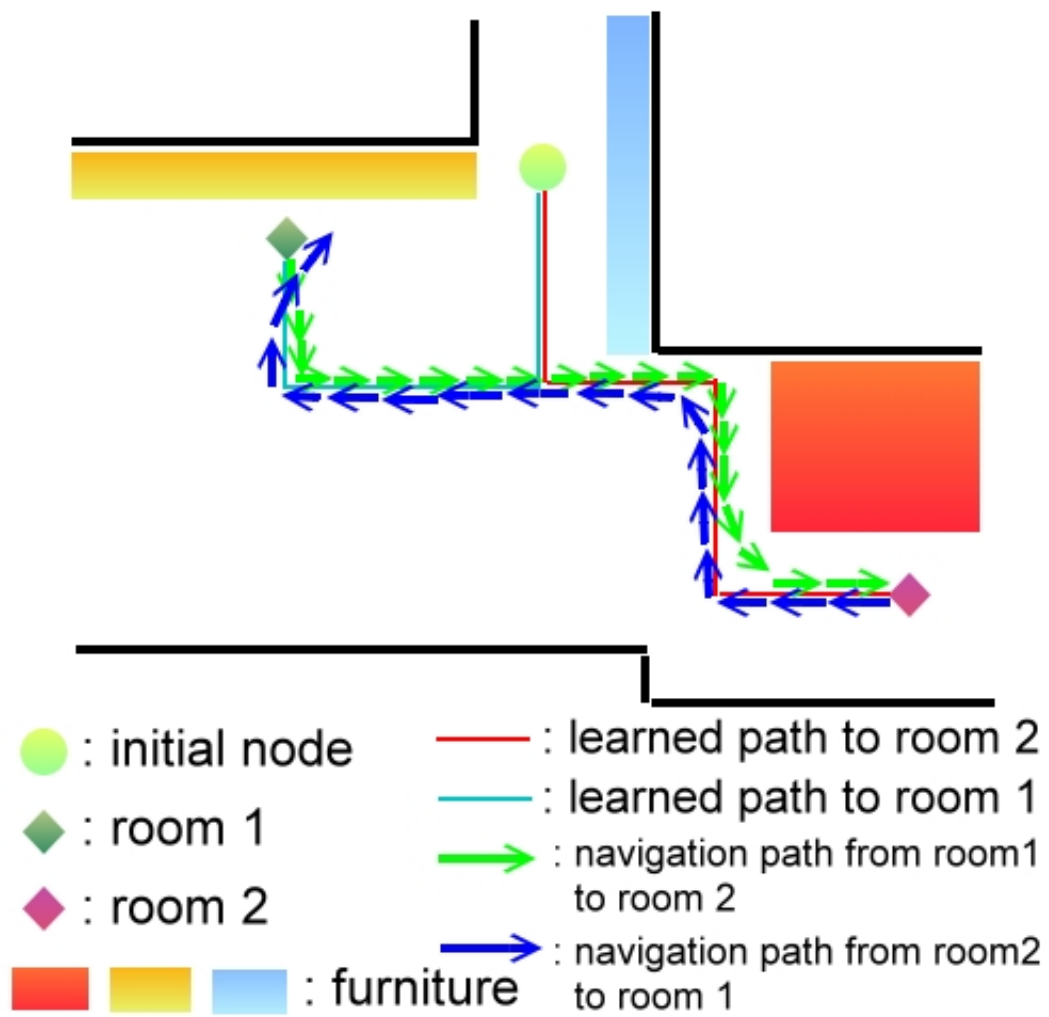


Figure 7.2. An experimental result in area mode (1).

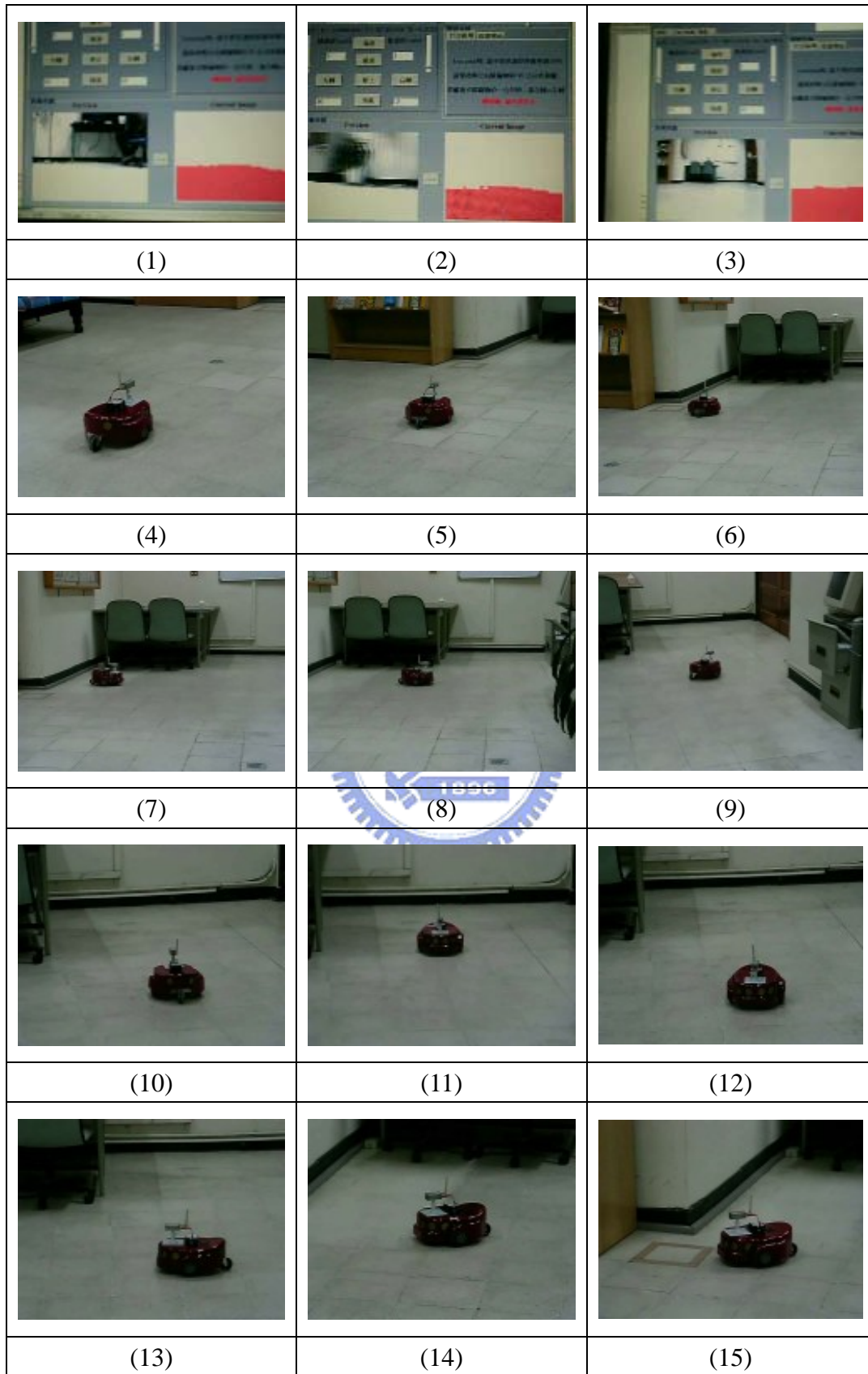


Figure 7.3 An experimental result in area mode (2).



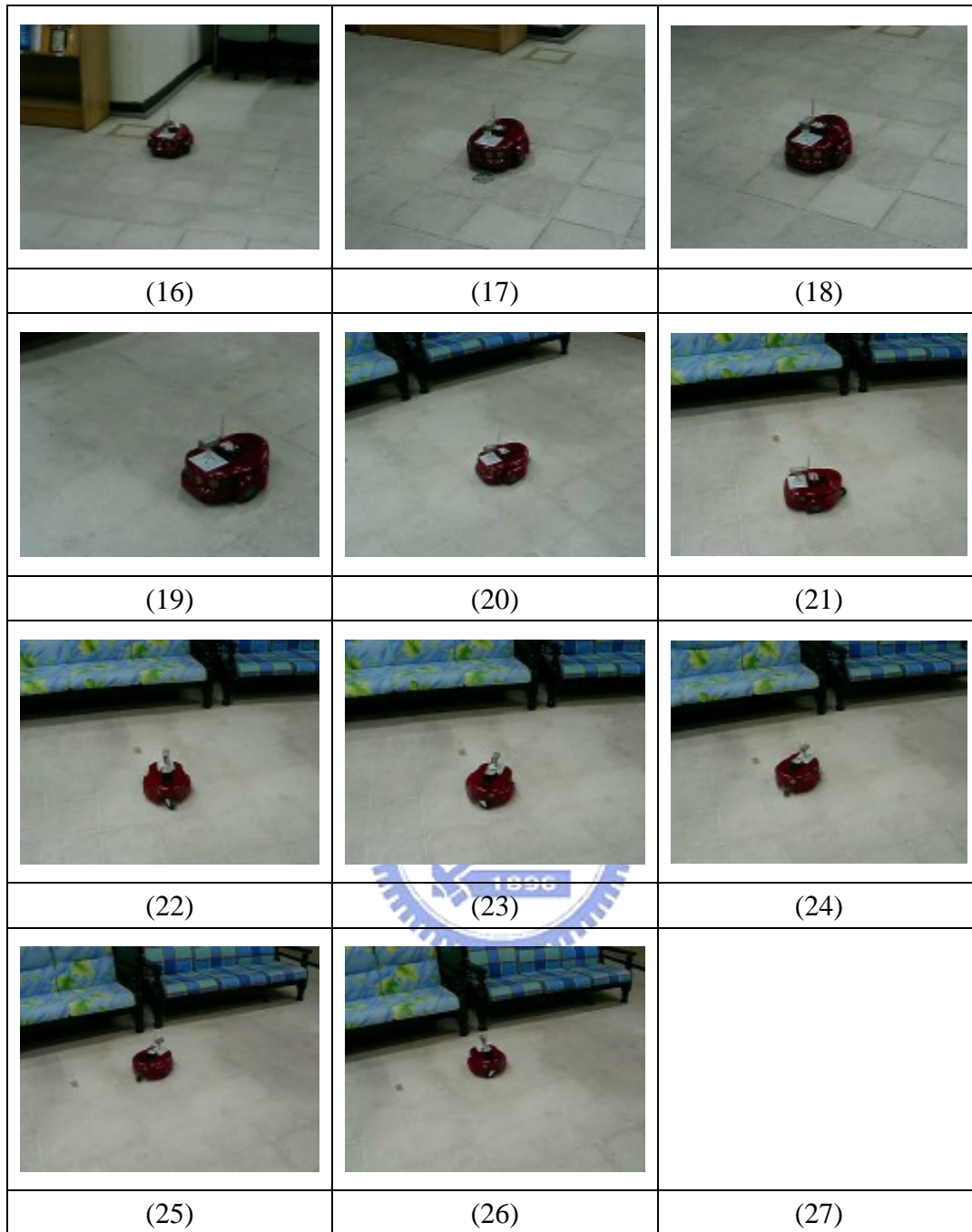


Figure 7.4 An experimental result in area mode (3).

## 7.2 Discussions

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

- (1) The floor of the indoor environment has to be as flat as possible. If certain little bulges or hollows appear on the floor that the vehicle has to pass, then the moving direction of the vehicle will diverge. And this kind of situation cannot be detected in the proposed system, so if this situation occurs too often, then the vehicle will get lost gradually.
- (2) Because of the small height of the camera on the vehicle, the eyesight of the vehicle is from about one to five meters. In other words, if an unexpected object appears suddenly in front of the vehicle and the distance between the object and the vehicle is smaller than one meter, then this object cannot be seen by the vehicle and the proposed fuzzy guidance technique cannot work to avoid the collision, either.
- (3) The color of the floor in a navigation environment must be different from those of other objects. If the color of a certain object (e. g., a piece of furniture) is too similar to the color of the floor, then this kind of object will be regarded as part of the route area. Therefore, the proposed fuzzy guidance technique may not lead the vehicle to avoid probable collisions correctly.



# 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 with fuzzy guidance and simple learning capabilities. Satisfactory navigation results have been.

At first, a simple nonvisual learning strategy with two modes has been proposed for different navigation needs. Less data are acquired in the proposed learning process without causing instable and imprecise navigation results. Next, fuzzy guidance techniques which are based on the fuzzy theory and computer vision techniques have been proposed for guiding a vehicle safely in indoor environments with complicated scenes. The navigation is based on a collision avoidance concept using fuzzy-control-rules and image analysis techniques. Two kinds of navigation strategies, namely, line following and curve following, have been proposed to guide smoothly a vehicle in two different types of navigation sessions. And the experimental results showed the feasibility of the proposed method.

In addition, a person following process has been proposed to track the detected human appearing in navigation paths. A sequence of processes, namely, object detection, human identification, and human tracking, have been proposed to detect an

unexpected object and to identify it as a human or not. After an occurrence of a human is confirmed, a human tracking process is performed to track it continuously until the human disappears in the eyesight of the vehicle.

## 8.2 Suggestions for Future Works

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

- (1) Finding a method to calibrate the odometer during the navigation process, using certain external devices like a compass. Besides, fixed cameras in the navigation environment can be used to locate the vehicle accurately. Therefore, odometer calibration can also be achieved using the location data given by the fixed cameras.
- (2) Adding the capability of starting navigation from arbitrary start points when users demand the vehicle to move to other room spots.
- (3) Adding the capability of voice control when users want to issue navigation orders to the vehicle.
- (4) Analyzing the gait of a human from his/her back view to identify him/her.
- (5) Adding the ability of detecting the situation that an unexpected human looks on the vehicle and trends to move closer to the vehicle.
- (6) Finding a wireless network camera to replace the wireless 2.4G camera, aiming at reducing electric wave interference.
- (7) Designing a camera system with a capability of panning, tilting, and

swinging.



# References

- [1] A. J. Davison and D. W. Murray, "Simultaneous localization and map-building using active vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, Issue 7, Jul 2002, pp. 865-880.
- [2] J. B. Hayet, F. Lerasle and M. Devy, "A visual landmark framework for indoor mobile robot navigation," *IEEE International Conference on Robotics and Automation*, Vol. 4, 2002, pp. 3942-3947.
- [3] Gaussier, P., Joulain, C., Zrehen, S., Banquet, J.P., Revel, A., "Visual navigation in an open environment without map," *Proceedings of 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 7-11 Sept. 1997, pp. 545 – 550, vol. 2
- [4] K. P. Li and W. H. Tsai, "Autonomous land vehicle guidance in complex room environments by computer vision techniques," *Proceedings of 1996 International Conference on Image Processing and Character Recognition (part of 1996 International Computer Symposium)*, Kaohsiung, Taiwan, Dec. 1996, pp.9-16.
- [5] P. L. Li and W. H. Tsai, "Path Learning, Planning, and Guidance for ALV navigation Inside Buildings," *M. S. Thesis, Department of Computer and Information Science, National Chiao Tung University*, 1997.
- [6] G. N. Desouza and A. C. Kak, "Vision for mobile robot navigation: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, Issue 2, Feb. 2002, pp 237-267.
- [7] Jung Tae Cho and Boo Hee Nam, "A study on the fuzzy control navigation and the obstacle avoidance of mobile robot using camera," *Systems, Man, and Cybernetics, 2000 IEEE International Conference on* , Vol: 4 , 8-11 Oct. 2000 pp.2993 - 2997 vol.4.
- [8] Howard, A., Tunstel, E., Edwards, D., Carlson, A "Enhancing fuzzy robot navigation systems by mimicking human visual perception of natural terrain traversability," *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th* , Volume: 1 , 25-28 July 2001 pp.7 - 12 vol.1
- [9] Nielsen, J. and Sandini, G, "Learning mobile robot navigation: a behavior-based approach," *1994 IEEE International Conference on Systems, Men and Cybernetics*, 2-5 Oct. 1994, pp.2809 – 2814, vol. 3.
- [10] Nayar, S.K., Murase, H., Nene, S.A., "Learning, positioning, and tracking visual appearance," *1994 IEEE International Conference on Robotics and Automation*, 8-13 May 1994, pp.3237 – 3244, vol.4.

- [11] Bianco, G, Zelinsky, A., Lehrer, M, “Visual landmark learning,” *2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 31 Oct.-5 Nov. 2000, pp. 227 – 232, vol.1.
- [12] Mata, M., Armingo, J.M., de la Escalera, A., Salichs, M.A., “Using learned visual landmarks for intelligent topological navigation of mobile robots,” *ICRA '03. IEEE International Conference on Robotics and Automation*, 14-19 Sept. 2003, pp. 1324 – 1329, vol. 1.
- [13] Y. Tsukamoto, “An approach to fuzzy reasoning method,” in M. M. Gupta, R.K. Ragade, and R. R. Yager, Eds., *Advances in Fuzzy Set Theory and Applications*, Amsterdam: North-Holland, pp. 137-149, 1979.
- [14] Baldassarri, P., Puliti, P., Montesanto, A., Tascini, G. “Visual self-localization using automatic topology construction,” *In proceedings of Image Analysis and Processing, 2003, Proceedings. 12<sup>th</sup> International Conference, 2003*, pp. 368-373.
- [15] Golovko, V., Ignatiuk, O., Sadykhov, R., “An approach to self-training of the mobile robot,” *In proceedings of the Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, International Workshop on, , 2001*, pp. 11-15.
- [16] G. Y. Chen and W. H. Tsai, “A new approach to vision-based unsupervised learning of unexplored indoor environment for autonomous land vehicle navigation,” *Robotics and Computer-Integrated Manufacturing*, Vol. 15, Issue 5, October, 1999, pp. 353-364.
- [17] Guan-Yu Chen and W. H. Tsai, “An incremental-learning-by-navigation approach to vision-based autonomous land vehicle guidance in indoor environments using vertical line information and multi-weighted generalized Hough transform technique,” *IEEE Transactions on System, Man and Cybernetics-Part B: Cybernetics*, Vol. 28, No. 5, October 1998, pp. 740-748.
- [18] P. Gaussier, C. Joulain, A. Revel, S. Zrehen, J. P. Banquet, S. Moga, and M. Quoy, “Autonomous robot learning-what can we take for free,” *Proceedings of IEEE International Symposium on Industrial Electronics*, Vol. 1, Portugal, Jul. 7-11, 1997, pp. SS1-SS6.
- [19] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithm*, McGraw-Hill, Reading, London, England, 2001.
- [20] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-Fuzzy And Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice-Hall, Reading, London, International(UK), 1997.
- [21] Rafael C. Gonzalez and Richard .E, *Digital Image Processing*, Addison-Wesley, Reading, MA, New York, U.S.A., 1993.