

以二維物體影像比對與三維電腦視覺分析作自動車航行之研究以及  
其於室內安全巡邏之應用

A Study on Autonomous Vehicle Navigation by 2D Object Image  
Matching and 3D Computer Vision Analysis for Indoor Security  
Patrolling Applications

研 究 生：陳冠傑

Student: Kuan-Chieh Chen

指 導 教 授：蔡文祥

Advisor: Prof. Wen-Hsiang Tsai

國 立 交 通 大 學  
多 媒 體 工 程 研 究 所  
碩 士 論 文



Submitted to Institute of Multimedia Engineering  
College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年六月

# **A Study on Autonomous Vehicle Navigation by 2D Object Image Matching and 3D Computer Vision Analysis for Indoor Security Patrolling Applications**

Student: Kuan-Chieh Chen    Advisor: Prof. Wen-Hsiang Tsai, Ph. D.

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

## **ABSTRACT**

A vision-based vehicle system for security patrolling in indoor environments using an autonomous vehicle is proposed. A small vehicle with wireless control and a web camera which has the capabilities of panning, tilting, and zooming is used as a test bed. At first, an easy-to-use learning technique is proposed, which has the capability of extracting specific features, including navigation path, floor color, monitored object, and vehicle location with respect to monitored objects. Next, a security patrolling method by vehicle navigation with obstacle avoidance and security monitoring capabilities is proposed. The vehicle navigates according to the node data of the path map which is created in the learning phase and monitors concerned objects by a simplified scale-invariant feature transform (simplified-SIFT) algorithm proposed in this study. Accordingly, we can extract the features of each monitored object from acquired images and match them with the corresponding learned data by the Hough transform. Furthermore, a vehicle location estimation technique for path correction utilizing the monitored object matching result is proposed. In addition, techniques for obstacle avoidance are also proposed, which can be used to find the clusters of floor colors, detect obstacles in environments with various floor colors, and

integrate a technique of goal-directed minimum path following to guide the vehicle to avoid obstacles. Good experimental results show the flexibility and feasibility of the proposed methods for the application of security patrolling in indoor environments.



# 以二維物體影像比對與三維電腦視覺分析作自動車航行之研究 以及其於室內安全巡邏之應用

研究生：陳冠傑

指導教授：蔡文祥 教授

國立交通大學多媒體工程研究所

## 摘要

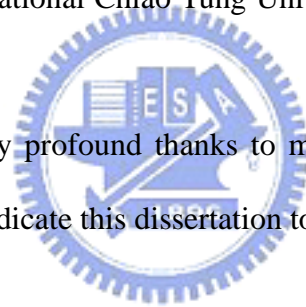
本論文基於電腦視覺技術提出了一套室內安全巡邏自動車系統，採用一台載有 PTZ 網路攝影機及具無線遙控功能的小型自動車作為實驗平台。本論文首先提出了一個易於使用的學習方法，供系統作學習巡邏環境之運用，其中包括：巡邏路線、地板顏色、監控物品，以及自動車相對於監控物品之位置。接著，本論文提出一個自動車安全巡邏的方法使自動車能進行安全監控以及障礙物自動閃避的工作。自動車會根據學習到的路徑作巡邏，並利用本論文所提出的簡化式 SIFT(Scale Invariant Feature Transform)方法進行物品監控。該方法從含有監控物品的影像中抽取特徵點，並與學習所得版本進行比對，再藉由霍夫轉換找出兩版本間的仿射轉換，據以測定自動車相對於物品之位置，以及修正自動車航行路徑的偏移。此外，本論文也利用地板的代表顏色來偵測障礙物，讓自動車適應於花色地板的環境，並整合目標導向路徑追隨之策略，使車子能自動閃避障礙物。最後我們利用一實際的室內環境來測驗本論文所提出的系統，結果自動車能順利自動修正路徑以及監控物品，顯示出本論文所提方法的完整性以及可行性。

## ACKNOWLEDGEMENTS

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

Thanks are due to Mr. Chih-Jen Wu, Miss Kuan-Ting Chen, Mr. Jian-Jhong Chen, Mr. Tsung-Chih Wang, Mr. Yi-Fan Chang, and Mr. Shang-Huang Lai for their valuable discussions, suggestions, and encouragement. Appreciation is also given to the colleagues of the Computer Vision Laboratory in the Institute of Computer Science and Engineering at National Chiao Tung University for their suggestions and help during my thesis study.

Finally, I also extend my profound thanks to my family for their lasting love, care, and encouragement. I dedicate this dissertation to my beloved parents.



# CONTENTS

<b>ABSTRACT</b> .....	<b>i</b>
<b>摘要</b> .....	<b>iii</b>
<b>ACKNOWLEDGEMENTS</b> .....	<b>iv</b>
<b>CONTENTS</b> .....	<b>v</b>
<b>LIST OF FIGURES</b> .....	<b>viii</b>
<b>Chapter 1 Introduction</b> .....	<b>1</b>
1.1 Motivation.....	1
1.2 Survey of Related Studies .....	3
1.3 Overview of Proposed Approach .....	5
1.4 Contributions.....	8
1.5 Thesis Organization .....	8
<b>Chapter 2 System Configuration and Navigation Principles</b> .....	<b>10</b>
2.1 Introduction.....	10
2.2 System Configuration .....	12
2.2.1 Hardware configuration .....	12
2.2.2 Software configuration.....	15
2.3 Learning Principle and Proposed Process.....	15
2.4 Vehicle Guidance Principle and Proposed Process.....	18
<b>Chapter 3 Learning Strategies for Indoor Navigation by Manual Driving</b> .....	<b>20</b>
3.1 Ideas of Proposed Techniques Used in Learning .....	20
3.1.1 Learning camera parameters .....	20
3.1.2 Environment features for learning .....	21
3.2 Camera Calibration .....	22
3.2.1 Calibration of location mapping .....	22
3.2.2 Calibration of intrinsic parameters.....	23
3.3 Learning of Specific Features .....	24
3.3.1 Learning of navigation paths composed of nodes.....	24
3.3.2 Learning of floor colors by k-means clustering.....	27
3.3.3 Learning of monitored objects by simplified SIFT.....	29
3.3.4 Learning of vehicle locations with respect to monitored objects	32
<b>Chapter 4 Security Patrolling by Vehicle Navigation in Indoor Environments</b> .....	<b>36</b>

4.1	Introduction to Proposed Ideas .....	36
4.2	Ideas of Navigation Process .....	37
4.2.1	Guidance by learned paths .....	37
4.2.2	Obstacle avoidance in various floor environments .....	39
4.2.3	Path correction by monitored object image matching .....	41
4.3	Purposes of Security Patrolling.....	44
4.3.1	Proposed technique for monitoring of objects .....	44
4.3.2	Types of monitored objects .....	46
4.4	Detailed Process for Security Patrolling by Vehicle Navigation .....	47
<b>Chapter 5 Detection of Monitored Objects by 2D Object Image Matching .....</b>		<b>50</b>
5.1	Introduction.....	50
5.2	Review of Method of Matching by Scale-Invariant Feature Transform (SIFT) .....	51
5.2.1	Scale-invariant keypoint localization.....	52
5.2.2	Feature descriptor generation.....	54
5.3	Proposed Simplified SIFT Features for Detection of Monitored Objects	56
5.3.1	Necessity of simplification of original concept .....	56
5.3.2	Detailed process of simplified SIFT feature generation .....	57
5.4	Proposed Matching Technique Using Simplified SIFT Features.....	62
5.4.1	Concept of proposed matching algorithm.....	62
5.4.2	Detailed algorithm .....	63
5.5	Experimental Results .....	65
<b>Chapter 6 Vehicle Guidance by Location Estimation Based on 2D Object Image Matching Results .....</b>		<b>69</b>
6.1	Introduction.....	69
6.2	Review of Vehicle Location Estimation Techniques .....	70
6.3	Vehicle Location Estimation by Object Image Matching Results .....	72
6.3.1	Coordinate systems .....	72
6.3.2	Idea of proposed method.....	73
6.3.3	Detailed process of location estimation .....	75
6.4	Path Correction by Vehicle Location Estimation Results .....	79
6.4.1	Direction angle of vehicle and coordinates of path nodes .....	79
6.4.2	Method of proposed path correction .....	81
6.5	Experimental Results .....	86
<b>Chapter 7 Obstacle Avoidance in Various Floor Environments .....</b>		<b>90</b>
7.1	Overview of Obstacle Avoidance Methods.....	90

7.2	Idea of Proposed Obstacle Avoidance Method .....	91
7.2.1	Coordinate system and direction angle of vehicle .....	91
7.2.2	Coordinate transformation .....	92
7.3	Obstacle Avoidance Techniques.....	93
7.3.1	Finding floor colors by k-means clustering .....	93
7.3.2	Obstacle detection by alert line scanning.....	96
7.3.3	Computation of goal-directed minimum path.....	99
7.4	Experimental Results .....	102
<b>Chapter 8 Experimental Results and Discussions.....</b>		<b>105</b>
8.1	Experimental Results .....	105
8.2	Discussions .....	111
<b>Chapter 9 Conclusions and Suggestions for Future Works .....</b>		<b>113</b>
9.1	Conclusions.....	113
9.2	Suggestions for Future Works.....	114
<b>References.....</b>		<b>116</b>





# LIST OF FIGURES

Figure 1.1 An overall framework of proposed system.....	6
Figure 2.1 The vehicle Pioneer3-DX used in this study. (a) A perspective view of the vehicle. (b) A front view of the vehicle. (c) A left-side view of the vehicle. ....	11
Figure 2.2 Structure of proposed system. ....	13
Figure 2.3 The pan-tilt-zoom camera used in this study. (a) A perspective view of the camera. (b) A front view of the camera. (c) A left-side view of the camera. ....	14
Figure 2.4 Flowchart of proposed learning process.....	17
Figure 2.5 Flowchart of proposed navigation process. ....	19
Figure 3.1 An illustration of attaching grids on the floor. A point set attached on the floor with coordinates known in the global coordinate system, as the red points.....	22
Figure 3.2 CAMcal user interface with the extracted grids result. ....	23
Figure 3.3 The checkerboard pattern used for the intrinsic parameters. (a) The printable checkerboard pattern. (b) The image of checkerboard pattern taken by the camera. ....	24
Figure 3.4 Control user interface. ....	25
Figure 3.5 An illustration of learned data. ....	27
Figure 3.6 A red rectangle including the monitored object as an interesting region. ..	30
Figure 3.7 A flowchart of the learning monitored object process.....	31
Figure 3.8 The relative position $C(x_r, y_r)$ of the vehicle with respect to the start point in the world coordinate system. ....	32
Figure 3.9 A user interface to specify the horizontal line. (a)A cyan line specified the horizontal line of the world coordinate system. (b) A zoom-in window for specifying the start point of the line. (c) A zoom-in window for specifying the end point of the line. ....	33
Figure 3.10 A flowchart of learning the vehicle locations process. ....	35
Figure 4.1 Computation of the turning angle and move distance. ....	38
Figure 4.2 Flowchart of obstacle avoidance process. ....	40
Figure 4.3 A cyan line (on the top of the poster) generated by applying the affine transform. ....	41
Figure 4.4 Flowchart of path correction process. ....	43
Figure 4.5 Flowchart of object monitoring process. ....	45
Figure 4.6 An example of a concerned object, specified as the blue region, in a complex background environment.....	46

Figure 4.7 Flowchart of proposed navigation process. ....	49
Figure 5.1 For each octave of scale space, the set of scale space images shown at the left and the computation of the difference-of-Gaussian images at the right. ....	53
Figure 5.2 Maxima and minima of the difference-of-Gaussian images are detected by comparing a pixel, as marked with <b>X</b> , with its 26 neighbors, as marked with the blue circles, in 3×3 regions of the current and adjacent scales (from [8]). ....	54
Figure 5.3 For each image, the gradient magnitude and orientation are computed in a region around the keypoint location, and weighted by a Gaussian window, indicated by the overlaid circle, as shown on the left. Four orientation histograms summarize their contents into 8 bins, as shown on the right. (from [8]). ....	56
Figure 5.4 A flowchart of the simplified SIFT generation. ....	58
Figure 5.5 The experimental results of the monitored object matching process. (a) is the monitored object learned in the learning phase. The location of features are marked as green crosses in (b). (c) is the successful matched result which is specified by the blue region. (d) and (e) are the matching results which fail to match with the learned object.....	66
Figure 5.6 The experimental results of the monitored object matching process with the comparison between the successful matching results by the use of the simplified SIFT algorithm and the original one. (a) is the matching result conducted by the use of simplified SIFT algorithm with reducing the number of octaves and (b) is the one conducted by the use of original SIFT algorithm.....	67
Figure 5.7 The experimental results of the monitored object matching process with the comparison between the failed matching results by the use of simplified SIFT and original SIFT algorithm. (a) and (c) are matching result conducted by the use of simplified SIFT algorithm with reducing the number of octaves, and (b) and (d) are the one conducted by the use of original SIFT algorithm. ....	68
Figure 6.1 Some landmarks used in previous approaches. (a) A diamond-shaped standard mark used in [9]. (b) A sphere used for robot location in [10]. (c) The perspective projection of a colored rectangle signboard used in [11]. (d) The perspective projection of a house corner used in [12]. (e) The two red edges of the perspective projection of a house corner used in [13]...	71
Figure 6.2 The relations of the reference coordinate system, the camera coordinate system, and the image coordinate system. ....	73

Figure 6.3 A diagram of the virtual house corner which is specified by the given horizontal line (the cyan line on the top of the poster), and the start point (the red point on the left-top of the poster). .....	74
Figure 6.4 Another two coordinate systems used in this study. (a) The global coordinate system. (b) The vehicle coordinate system. ....	80
Figure 6.5 The rotate angle $\theta_v$ of the vehicle and the pan angle $\theta_c$ of the PTZ camera. (a) $0 \leq \theta_v \leq \pi$ . (b) $-\pi \leq \theta_v \leq 0$ . (c) $0 \leq \theta_c \leq \pi$ . (d) $-\pi \leq \theta_c \leq 0$ .....	81
Figure 6.6 The relation among the vehicle, the camera, and the reference coordinate system. ....	82
Figure 6.7 The relation among the RCS, the VCS, and the GCS, and the corresponding angle the vehicle. The learned location of the vehicle is denoted as a pastel vehicle with the location $(X_l, Y_l)$ in the RCS, and the current location of the vehicle is denoted as the colored vehicle with the location $(X_v, Y_v)$ in the RCS. ....	84
Figure 6.8 An experimental navigation path including a monitoring node and a path node. (a) A diagram of the navigation path. (b) A photo of the navigation path. (c) The user interface of learning the monitored object and the location estimation data. ....	86
Figure 6.9 The experimental result of path correction by the navigation starting from the original learned start position. (a) The vehicle arrived at the monitoring node and performed the matching and path correction. (b) The vehicle navigated to the next path node after correcting the navigation path. (c) The vehicle successfully matched the monitored object and estimated the location. ....	87
Figure 6.10 The experimental result of path correction by the navigation starting from a different start position. (a) The vehicle started the navigation at a position different to the start position in the learning phase. (b) The vehicle arrived to a wrong place where should be the monitoring node, and performed the matching and path correction. (c) The vehicle still successfully matched the monitored object and estimated the location. (d)-(e) The vehicle navigated to next path node after correcting the navigation path.....	88
Figure 7.1 The coordinate transformation between the VCS and the GCS. ....	92
Figure 7.2 A flowchart of finding the floor color clusters by the k-means clustering. ....	96
Figure 7.3 An alert line specified by three red line segments in the image. ....	97
Figure 7.4 The alert scanning results. (a) The mask of the region growing result of the candidate obstacle seeds. (b) The mask generated by applying morphological operations on (a). (c) The green parts of the alert line	

represent the floor and the red parts of the alert line represent the obstacles. ....	98
Figure 7.5 An illustration of computation of goal-direction minimum path (a modified version from [3]). ....	100
Figure 7.6 Two kinds of floors. ....	102
Figure 7.7 The experimental result of obstacle detection by k-means clustering. (a) The image contains the floor and an obstacle. The red parts of the alert line specify the obstacle and the green parts of the alert specify the floor. (b) The clustering result which is shown in the $u'v'$ chromaticity plane. (c) The colors of the clusters. (d) The non-floor colors on the alert line. (e) The mask created by region growing. (f) The mask after applying the morphological operations. ....	103
Figure 7.8 Another experimental result of detecting an obstacle by k-means clustering. (a) The image contains the floor and an obstacle. (b) The clustering result. (c) The colors of the clusters. (d) The non-floor colors on the alert line. (e) The mask created by region growing. (f) The mask after applying the morphological operations. ....	104
Figure 8.1 A user interface of the experiment. User can control the vehicle through this interface to learn a path and the monitored objects on the path. ....	105
Figure 8.2 An illustration of learned data and navigation path. ....	106
Figure 8.3 The experimental result of object monitoring and navigation path correction. (a) Monitored object labels. (b) The vehicle monitors the monitored objects. (c) The matching result and the horizontal line used for path correction. (d) The image of learned monitored objects. (continued) ....	109
Figure 8.4 The vehicle detects an obstacle and changes the navigation path to avoid the obstacle. (a) The mask of detected obstacle region. (b) The image contains the floor, an obstacle, and the alert line specified the distribution of the obstacle. (c)~(h) show the vehicle avoiding an chair as an obstacle. (continued) ....	111

# Chapter 1

## Introduction

### 1.1 Motivation

In recent years, studies on vision-based autonomous vehicle navigation are in high prominence because of its great potential in various applications and the developments of computer vision techniques [1]. They give autonomous vehicles the ability to perform a great variety of dangerous or dreary works in replacement of human beings, for example, interoffice document delivering, unmanned transportation, house cleaning, security patrolling, etc.

Today, for security surveillance, we usually install lots of stationary cameras to monitor indoor environments. Success of environment security monitoring depends on the operator's endeavor to keep track of the videos taken by cameras. This way spends much time and manpower and lacks efficiency. Moreover, it results in weakness of the surveillance system because of its fixed field of view. These obvious disadvantages of conventional security surveillance systems prohibit more intelligent environment monitoring and surveillance applications.

In order to overcome the above problem, we can employ a vision-based autonomous vehicle which has high mobility and is equipped with more capable cameras. It can be utilized to patrol in indoor environments to assist the security surveillance system to monitor the area uncovered by still cameras' fields of view. And we can use it in replacement of patrolling guards to conduct security patrolling ceaselessly. When the vehicle detects an abnormal condition, it can send an alert

message to the security center. This provides more efficient and reliable security protection.

To develop autonomous vehicle systems for indoor security patrolling applications, the most critical requirement is that the vehicle has to be guided *smartly* when patrolling in indoor environments. Facing this challenge, learning artificial landmarks or specific scene features in the environment and locating the vehicle by landmark or feature matching are feasible solutions. Although many works based on these ideas have been developed in the past decade, most of them can only learn landmarks with special shapes or in ideal backgrounds like pure-colored ones, resulting in unreasonable restrictions on environments in which the vehicle can navigate. Therefore, it is desired in this study to design a method utilizing the technique of monitored-object image matching for vehicle location estimation. The idea, simply speaking, is to analyze the 3D geometric transformation of different monitored object views to estimate the vehicle location.

More specifically, in a traditional vision-based autonomous vehicle navigation system, the vehicle is usually equipped with a fixed pinhole camera, and the view of the vehicle is restricted to be a lower area. Hence, instead of using a fixed pinhole camera, we equip the vehicle with a *pen-tilt-zoom camera*, called *PTZ camera* in the sequel. With the PTZ camera and its movement, the view of the vehicle is extended to a wider range. Thus we can monitor objects which are located higher than the camera or detect obstacles which are placed lower than the camera, by the images taken with the PTZ camera.

In summary, our research goal in this study is to develop an autonomous vehicle security patrolling system with the following capabilities:

1. navigating in desired environments automatically;
2. monitoring concerned objects;

3. detecting obstacles and dodging automatically; and
4. calibrating vehicle locations against incremental mechanical errors.

## 1.2 Survey of Related Studies

In order to achieve the mission of security patrolling in indoor environments, the design of algorithms for learning navigation paths and recording the features of monitored objects is required at first. While the vehicle patrols routinely, some unexpected situations might be encountered, such as an obstacle blocking a road which the vehicle has to pass. To detect such situations, it is necessary to measure the distance between the vehicle and an object. Lai and Tsai [2] proposed a 2D-to-3D distance transformation by using a curve fitting technique and a modified interpolation technique. The distances between the vehicle and the surroundings in the real world were measured accordingly through captured images. For obstacle avoidance, Chiang and Tsai [3] proposed a goal-directed minimum-path following technique to guide the vehicle in order to avoid collisions with obstacles. Besides, a fuzzy guidance technique with two navigation modes was proposed by Chen and Tsai [4]. It creates a navigation map with two kinds of learned data. Then, a fuzzy guidance technique is applied to accomplish the navigation work according to the navigation map. Moreover, a learning method using manual driving was proposed by Chen and Tsai [5]. Before the vehicle navigation stage, a user drives the vehicle to learn the navigation path and monitoring objects. Then, the vehicle with mechanic error correction and visual object monitoring capabilities can accomplish specified navigation works.

For object monitoring, the vehicle has to learn the features of the concerned



object and match the features to determine whether the object is exactly the same as the previous learned one. There are many progresses made in the use of invariant features for object shape recognition or matching in the past decade. Schmid and Mohr [6] proposed a local feature detector for general image recognition problems. Mikolajczyk and Schmid [7] extended this idea to the Harris-Laplace detector and used it to detect points of interests at several scales and to select the right scale by computing the maximum Laplace function. Lowe [8] used a scale-invariant detector to find the extrema in the difference-of-Gaussian scale-space. He then created a *scale-invariant feature transform* (SIFT) descriptor to match key points using a Euclidean distance metric in an efficient best-bin first algorithm where a match is rejected if the ratio of the best and the second best matches is greater than a threshold.

While the vehicle patrols, the vehicle location is the most vital information to keep the navigation in track. Traditionally, an autonomous vehicle is equipped with an odometer to measure the current location of the vehicle. However, it usually suffers from incremental mechanic errors. Thus we need a technique of vision-based vehicle location estimation to reset the mechanic error.

Fukui [9] proposed a method for vehicle location by utilizing a diamond shape whose boundary consists of four identical thick line segments with known lengths. Magee and Aggarwal [10] used a sphere on which two perpendicular great circles were drawn as a standard landmark for vehicle location. Huang et al. [11] also used a colored rectangle signboard and obtained the relative position between the signboard and the vehicle by calculating the vanishing points in the image of the signboard. Moreover, Chou and Tsai [12] proposed a method which utilizes a house corner existing naturally in the house to estimate the vehicle location. And Chiang and Tsai [13] simplified the Chou's formula and applied the resulting technique to the application of indoor vehicle guidance with a PTZ camera.



## 1.3 Overview of Proposed Approach

In this study, it is desired to develop an effective vision-based autonomous vehicle system for security patrolling in indoor environments. With this purpose, an easy-to-use learning method which processes the odometer data of manual driving obtained in learning and the features of monitored objects extracted in navigation automatically is proposed. Secondly, a vision-based navigation technique with obstacle avoidance and specific object monitoring capabilities is proposed. Finally, a vehicle location calibration technique based on monitored-object image matching results is proposed.

We use the odometer to provide the position of the vehicle and the image captured by the PTZ camera equipped on the vehicle to monitor higher-located objects as well as the surrounding environment. An overall framework of the proposed system is illustrated in Figure 1.1. Here we divide the work conducted by the system into two phases: the *learning phase* and the *navigation phase*.

In the *learning phase*, the proposed learning method consists roughly of four steps. The first step is *navigation path learning*. Since the vehicle must know where to patrol, we control the vehicle to move to desired places through a user interface. Then, the vehicle processes the odometer data to get the positions of desired places and then records these data as *navigation nodes* along the learned path. While the vehicle is entering a different floor environment, in order for the system to avoid misrecognizing a new floor as an obstacle, the vehicle has to record the features of the new floor as part of the navigation node data.

The second step is *monitored object learning*. In this step, the vehicle has to record the features of the objects to be monitored. With the use of the PTZ camera, the vehicle can capture the images of the monitored objects which are located higher than

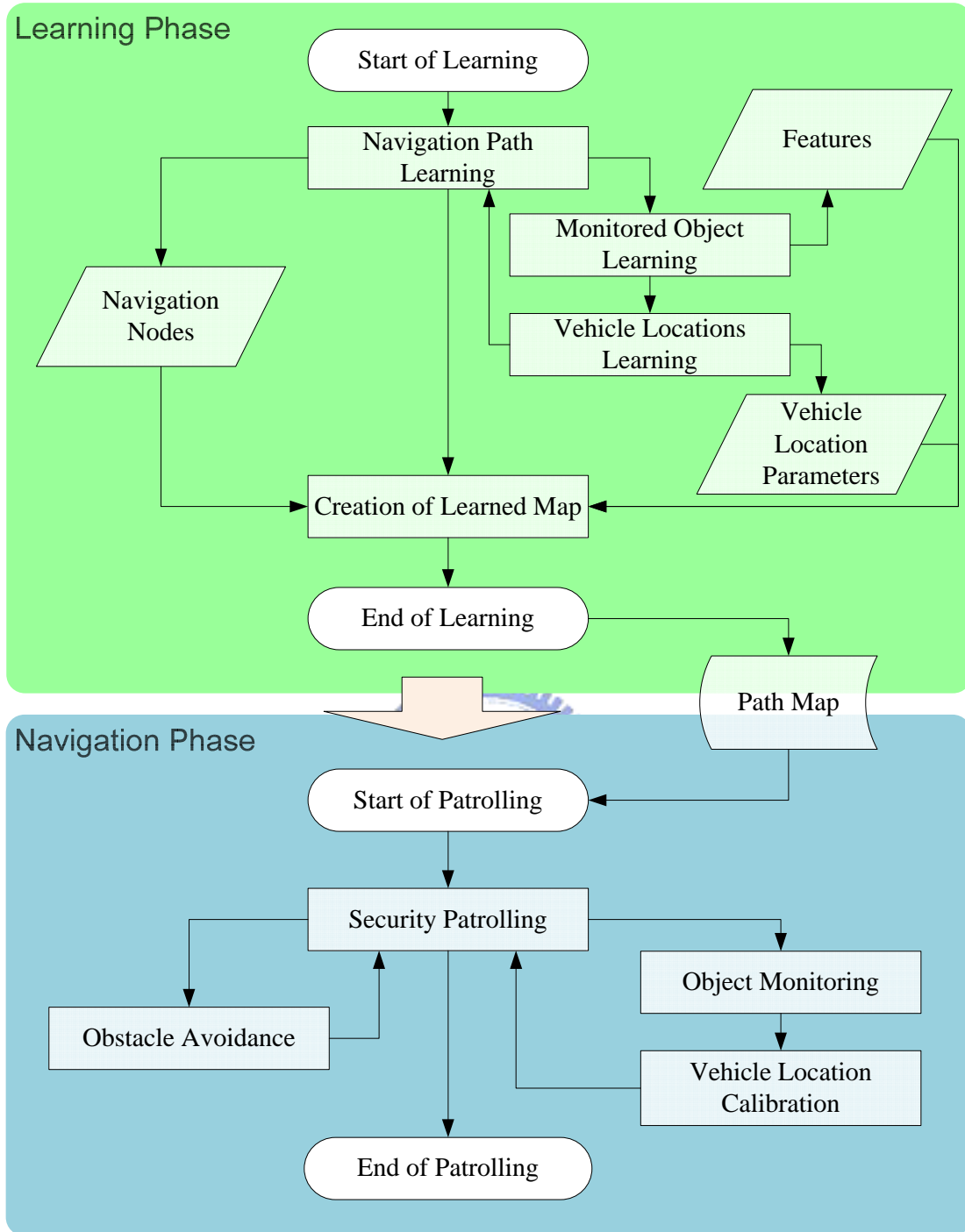


Figure 1.1 An overall framework of proposed system.

the camera. For object recognition and matching, Lowe [8] proposed the SIFT to extract features from given images as *SIFT descriptors* and used a best-bin first algorithm for SIFT descriptor matching, as mentioned previously. Since in the navigation phase, the position of the same monitored object will be *just close to*,

instead of exactly at, the one found in the learning phase, resulting in a slight variation on the scale of the taken images, we propose in this study a *simplified SIFT* which reduces the difference of Gaussian scale layers. It is faster than the original SIFT to meet real-time security monitoring needs.

The third step is *vehicle locations learning*. While using the monitored object matching results for vehicle location adjustment, the system requires the knowledge of the vehicle position with respect to the monitored object. Thus a position relative to the monitored object is recorded.

The final step of the learning phase is *creation of a learned path map*. The navigation nodes and the features of the monitored objects recorded in the previous step are processed to create a *path map* in this step. This path map is designed to be in a form of graph which is composed of a set of nodes connected with edges. The path map can then provide the patrolling data for use in the navigation phase.

In the *navigation phase*, the vehicle reads the path map and moves along the path nodes orderly in accordance with the map data. While the vehicle navigates, we analyze the floor images taken by the camera continuously. If an obstacle is detected on the patrolling path, a new path for obstacle avoidance and destination approaching is planned. If the vehicle navigates to a node where a monitored object is located, it detects the object and adjusts the vehicle location to move to the right navigation path. If the vehicle finds the concerned object missing, a warning message will be issued.

In summary, an autonomous vehicle navigation system for indoor security patrolling applications with capabilities of novel learning and self-adjustment of navigation paths is proposed. The vehicle can utilize the navigation map acquired by the proposed learning process to navigate in desired indoor environments and monitor concerned objects.

## 1.4 Contributions

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

- (1) A simplified SIFT algorithm for automatic matching of monitored object images captured by PTZ cameras is proposed.
- (2) A vision-based vehicle location estimation method utilizing monitored-object image matching to avoid mechanic error accumulation is proposed.
- (3) A method for security monitoring of various types of concerned objects is proposed.
- (4) A method based on k-means clustering of floor colors for real-time obstacle avoidance in environments with various floor colors is proposed.
- (5) A vision-based detection of obstacles by learned floor colors is proposed.



## 1.5 Thesis Organization

The remainder of this thesis is organized as follows. In Chapter 2, we describe the system configuration of the vehicle used as a test bed in this study, as well as the principle of vehicle learning and guidance. In Chapter 3, the proposed techniques for extraction of specific features such as navigation path, floor color, monitored object, vehicle location with respect to monitored objects are described. In Chapter 4, the proposed method for security patrolling by vehicle navigation with obstacle avoidance capability and the security monitoring processes is described. In Chapter 5, the proposed method for detecting monitored objects by object image matching is described. In Chapter 6, the proposed vision-based vehicle location estimation by object image matching results to correct the odometer records in the vehicle is

described. In Chapter 7, the proposed method for obstacle avoidance in various floor environments is described. Some experimental results are shown in Chapter 8. Finally, some conclusions and suggestions for future works are given in Chapter 9.



# Chapter 2

## System Configuration and Navigation Principles

### 2.1 Introduction

For security surveillance, a vision-based autonomous vehicle system in indoor environments might face many unexpected conditions, such as colliding with obstacles, passing through narrow paths, finding missing valuables, etc. In order to achieve the objective of security patrolling by an autonomous vehicle, a small and agile vehicle is the best choice for this study. It is suitable to navigate in the large patrolling area and monitor the open space such as corridors, lobbies, or exhibition halls.

The autonomous vehicle system used in this study is composed of a small vehicle and a pan-tilt-zoom camera. For users to control the vehicle, some communication and control equipments are required. The entire system configuration including hardware equipments and software is introduced in Section 2.2.

To navigate in an unknown indoor environment, a learning strategy is necessary to teach the vehicle where to navigate, what to monitor, and how to adjust the vehicle locations before starting security patrolling. In the following sections, we describe the vehicle navigation principles and the overviews of the detailed processes. In Section 2.3, the principle and process of learning navigation paths and environment features are described. In Section 2.4, the principle and process of the security patrolling in

which the vehicle monitors concerned objects and patrols in desired indoor environments automatically are described.



(a)



(b)



(c)

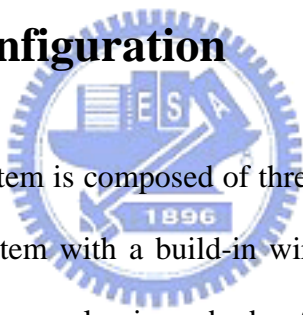
Figure 2.1 The vehicle Pioneer3-DX used in this study. (a) A perspective view of the vehicle.

(b) A front view of the vehicle. (c) A left-side view of the vehicle.

## 2.2 System Configuration

In this study, we use the Pioneer 3-DX, an agile, versatile intelligent vehicle made by MobileRobots Inc., as a test bed. The vehicle is equipped with a pan-tilt-zoom camera, as shown in Figure 2.1. Because the whole system is controlled by users remotely, some wireless communication equipments are necessary. The hardware architecture and used components of the test bed are described in Section 2.2. Besides, the software including application programming interfaces and development tools we used in the study to help us develop the system and provide an interface for users to control the vehicle is described in Section 2.2.2.

### 2.2.1 Hardware configuration



The entire navigation system is composed of three parts, as shown in Figure 2.2. The first part is a vehicle system with a build-in wireless device and an embedded control system. The vehicle has an aluminum body of the size of 44cm×38cm×22cm with two 19cm wheels and a caster. The vehicle can climb a 25% grade and sills of 2.5cm. On flat floors, the vehicle can reach a forward speed of 160cm per second and a rotation speed of 300 degrees per second. There are three 12V rechargeable lead-acid batteries in the vehicle which supply the power. The vehicle can run 18-24 hours with the three fully charged batteries. By a user's command, the embedded control system can control the vehicle to move forward or backward or to turn around. The system is also able to return some status parameters of the vehicle to the user.



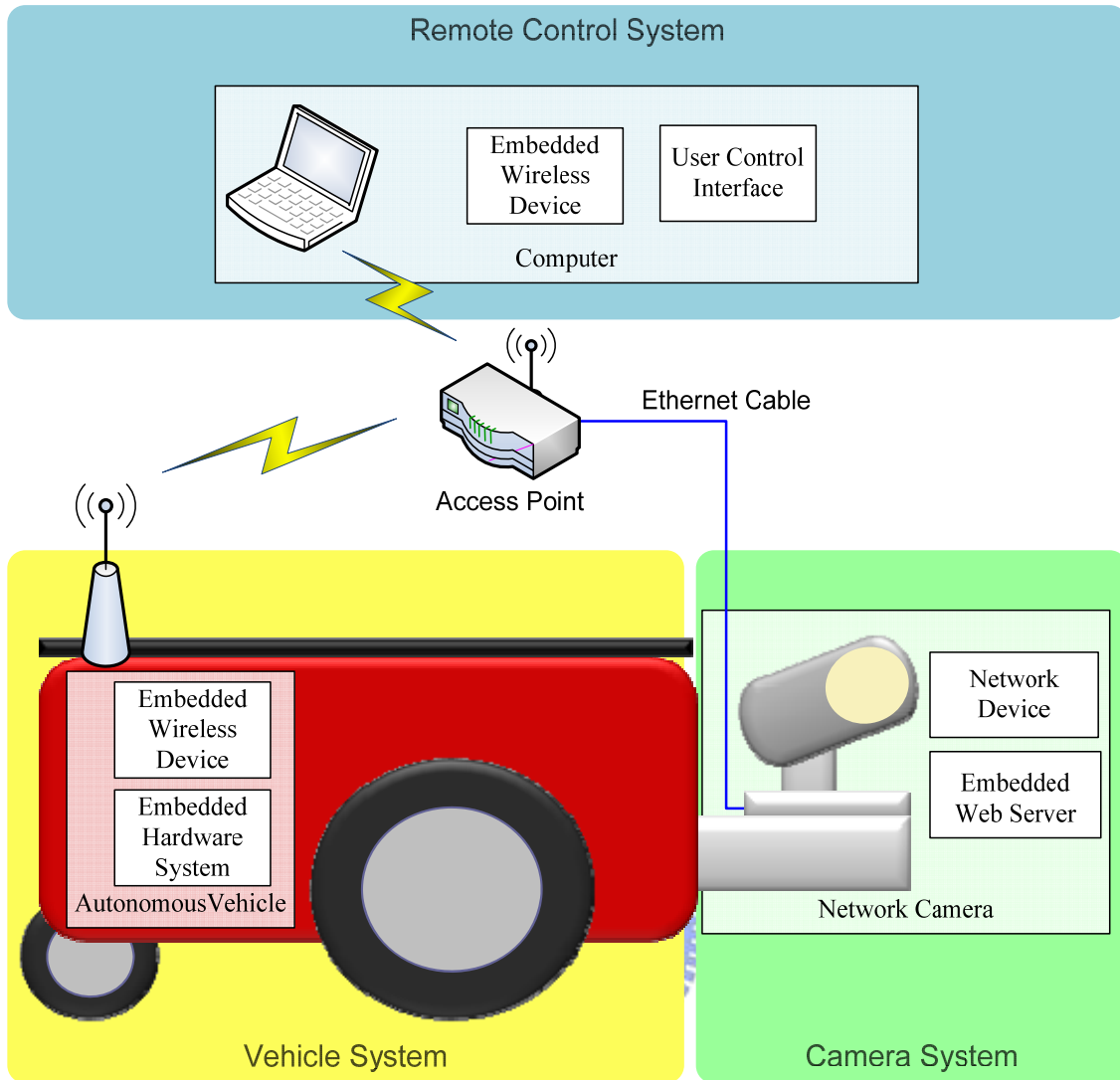


Figure 2.2 Structure of proposed system.

The second part is a digital IP camera with panning, tilting, and zooming (PTZ) capabilities. The PTZ IP camera used in this study is an AXIS 213 PTZ made by AXIS, as shown in Figure 2.3. This is a camera with a height of 130mm, a width of 104 mm, a depth of 130mm, and a weight of 700g. The pan angle range is 340 degrees and the tilt angle range is 100 degrees. It has 26x optical zoom and 12x digital zoom. The image captured in our experiments is of the resolution of 320×240 pixels for the reason of raising image processing efficiency. Moreover, the camera is directly connected to an access point by a network cable for transmission of the captured image.



(a)



(b)



(c)

Figure 2.3 The pan-tilt-zoom camera used in this study. (a) A perspective view of the camera. (b) A front view of the camera. (c) A left-side view of the camera.

The third part is a remote control system using a notebook PC. A control program can be executed on the remote control system to issue commands and get the status information from the vehicle and the PTZ camera. All commands transmitted to the vehicle or to the camera are through the wireless network. There is an access point in our test environment which meets the IEEE 208.11g standard to offer a bandwidth for the remote control system to communicate with the vehicle and the camera. Both the vehicle and the remote control system own wireless devices to connect to the access point, and the camera connects to the access point via a network cable. In other words,

we use the access point as a medium to connect the three parts of the proposed navigation system.

## 2.2.2 Software configuration

The MobileRobots Inc. provides an Advanced Robotics Interface for Applications (ARIA) which is an object-oriented programming interface in the C++ language to control the mobile robot. The lowest-level data and information of the vehicle are also retrieved easily by means of the ARIA. In other words, using the ARIA as an interface makes developers to communicate with the embedded system of the vehicle. And we use the Borland C++ builder as the development tool in our experiments.

For PTZ camera controlling, the AXIS Company also provides a development tool called AXIS Media Control SDK for the AXIS 213 PTZ. With the SDK, we can preview the image of the camera's view and get the current image data from the camera. We can also perform the panning, tilting, and zooming actions easily through the SDK. It is convenient for users to develop any function with the images grabbed from the camera.

## 2.3 Learning Principle and Proposed Process

To perform security patrolling in an unknown environment, a learning process is necessary. It is desired to develop an easy-to-use learning process which can learn the knowledge of desired navigation paths and concerned objects. As soon as the learning

process ends, the navigated paths, the features of the monitored objects, and the vehicle locations' data are recorded in advance. The entire learning process in this study is shown in Figure 2.4.

In order to help users teach the vehicle, a user control interface is designed for use in controlling the vehicle and specifying the objects to be monitored. The user drives the vehicle to navigate in indoor environments and move to the front of the concerned objects. The main recorded data include two categories, namely, path-related data and object-related ones. As soon as the learning process ends, all data are stored in the storages of the computer such that the learning process is only executed once and the data can be used repeatedly.

More specifically, while the vehicle navigates in an open space by the control of a user, it records the path data provided by the odometer, and denotes them as *navigation nodes*. It also analyzes the images taken by the camera continuously to detect whether the colors of the floor change. If the colors change, the vehicle records the colors of the new floor and the navigation node data which represents the start position of the new floor.

When the vehicle arrives at the front of a concerned object, the user can control the PTZ camera to move toward the object and select the object in the image captured by the camera. Then, the features of the object are computed automatically from the images by performing the *simplified SIFT*. And the relative position between the vehicle and the monitored object is also computed automatically from the image subsequently. With such manners, the user can specify the concerned objects continuously along the path until finishing a learning process.

After finishing the learning process, a navigation map which consists of the path and monitored objects data is created and saved into a text file for use in the navigation phase.

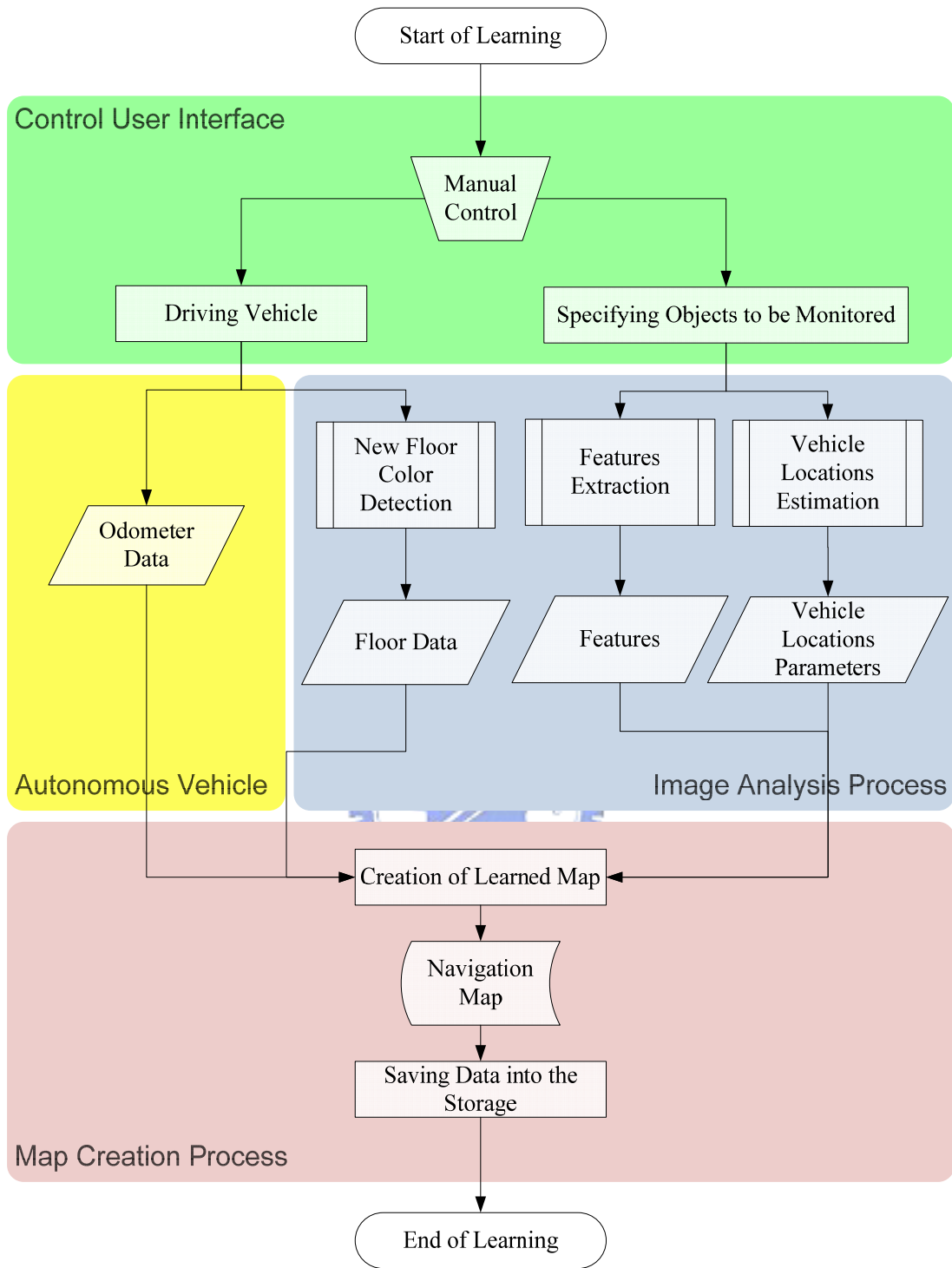


Figure 2.4 Flowchart of proposed learning process.

## 2.4 Vehicle Guidance Principle and Proposed Process

Before the vehicle starts to patrol, the system reads the path map created in the learning phase, as mentioned previously. In order to guide the vehicle along the learned path, the vehicle moves sequentially from one node to another according to the path map. While the vehicle patrols in indoor environments, two things should be paid attention to. One is whether the vehicle reaches the next node or not. The other is whether the vehicle encounters an obstacle or not. An illustration of the vehicle navigation process is shown in Figure 2.5.

When the vehicle reaches the next node, it checks first whether the node includes the monitored object data or the color data of the new floor. If the node includes the color data of the new floor, the vehicle updates the reference color of the floor which the obstacle detector uses for recognizing an obstacle on the floor. If the node includes the monitored objects data, the vehicle uses the learned data to detect whether the object still exists or not. If the detection or matching process of the object fails, the system will issue an alarm message to the user. Otherwise, the vehicle uses the learned vehicle locations' data to adjust the vehicle's location.

Besides, the vehicle might encounter an obstacle blocking the navigation path in the patrolling process. In order to detect such a condition, the system analyzes the images taken by the camera continuously by the use of the floor color to recognize an obstacle on the floor. If an obstacle is detected on the patrolling path, a new path for obstacle avoidance and destination approaching is planned. With such navigation processes, the vehicle can patrol along the learned path to accomplish specified security patrolling works.

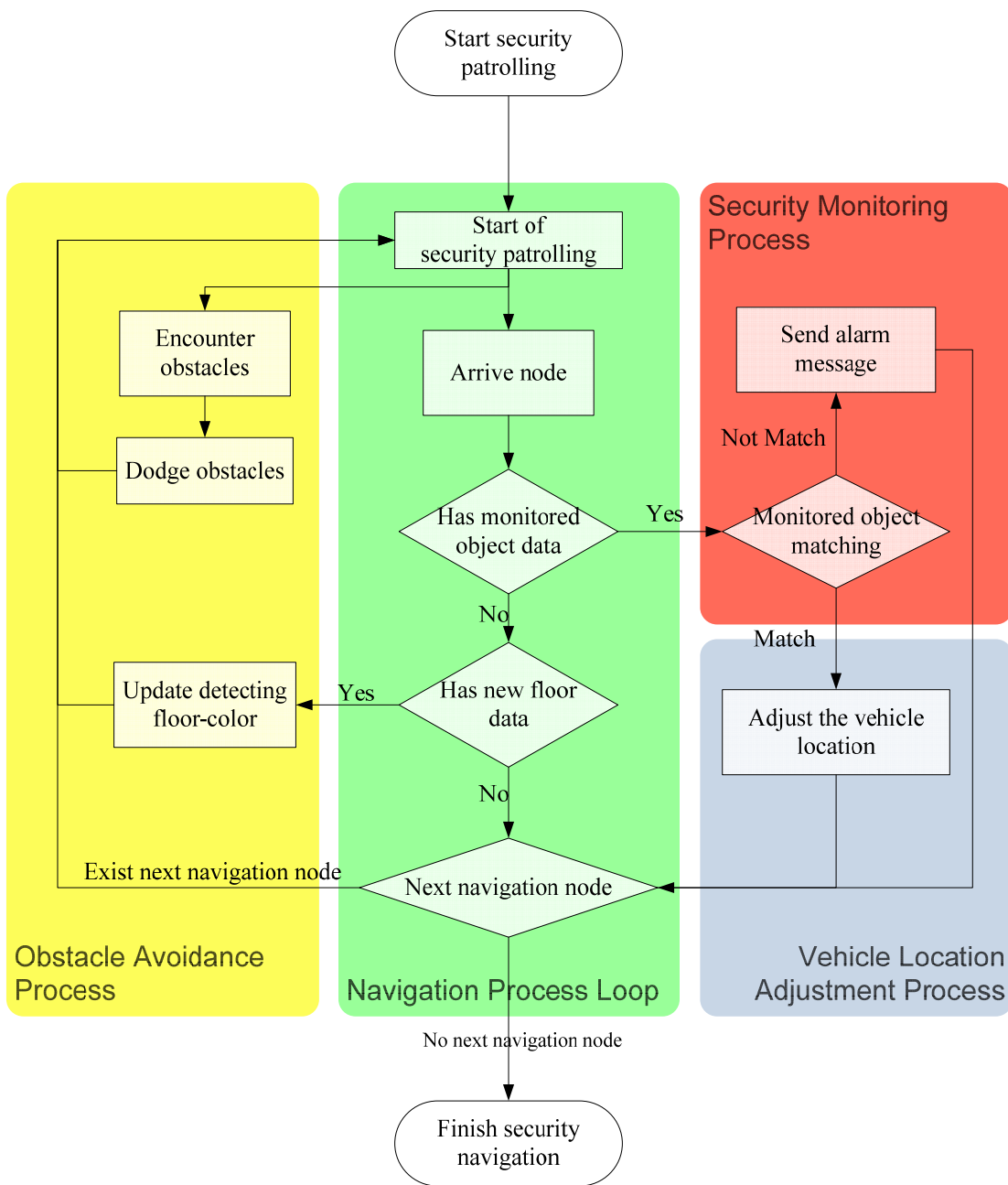


Figure 2.5 Flowchart of proposed navigation process.

# Chapter 3

## Learning Strategies for Indoor Navigation by Manual Driving

### 3.1 Ideas of Proposed Techniques Used in Learning

In general, indoor environments are usually complicated and the concerned objects are placed at different positions, and so building a complete path map to guide the patrolling vehicle is indispensable. Thus, to create the path map and select monitored objects is a primary work for security patrolling by vehicle navigation. In this chapter, we divide the data to be learned into two categories, namely, the *camera parameters* and the *environment features*. After learning whole data, we utilize the data to build the *path map* for security patrolling and save it into the storage of the computer, as described in Section 2.3.

#### 3.1.1 Learning camera parameters

While the vehicle navigates, many unexpected situations might be encountered, such as facing an obstacle in front. It is desired to measure the distance between the vehicle and an object. Since the camera is the only sensor to gather the features of the environment, the calibration of location mapping and image analysis techniques are necessary in this study. We utilize a real-world location data acquisition method [5] by the calibration of location mapping to obtain the relative position between the vehicle



and an obstacle precisely. The detailed process is described in Section 3.2.1

Besides, while calculating the relative position between the vehicle and the monitored object, it is necessary to know the camera intrinsic parameters, such as the focus length, the coordinates of the image center, etc. Hence, we adopted a method proposed by Shu et al. [14] for calibrating the camera based on planar checkerboard patterns in this study. The detailed calibration process is described in Section 3.2.2

### **3.1.2 Environment features for learning**

In order to navigate in an unknown environment, four kinds of environment features for learning are used in this study. The first is navigation path data. Although we can get the position of the vehicle by the odometer value at any time, it is desired to represent the entire path by simple and useful values. In Section 3.3.1, we describe how to gather path data when the user controls the vehicle to navigate in an indoor environment. The second feature is floor color. For obstacle avoidance, how to detect an obstacle in various floor environments is the key issue. Hence, we propose a detection method which uses the learned floor color based-on k-means clustering, as illustrated in Section 3.3.2. The third feature is the monitored objects' feature. In order to monitor concerned objects, the vehicle must firstly learn the features of them. Then the vehicle can perform object monitoring by matching the features computed in the navigation phase. We propose a simplified SIFT in this study to transform monitored-object images into the features. The detailed learning process is described in Section 3.3.3. The last feature is the vehicle location with respect to each monitored object, as illustrated in Section 3.3.4.

## 3.2 Camera Calibration

### 3.2.1 Calibration of location mapping

In order to measure the distance between the vehicle and an object, we map the tessellated points in the image coordinate system to the ones in the global coordinate system, as shown in Figure 2.2.

Using a point set attached on the floor with coordinates known in the global coordinate system, as the red points shown in Figure 2.2, and their corresponding point set appearing in the image taken by the camera, we can get a point-to-point coordinate transformation from the image coordinate system to the global coordinate system. And for any point that is not exactly at the tessellated point, an interpolation method is performed to gather their corresponding coordinates in the global coordinate system.

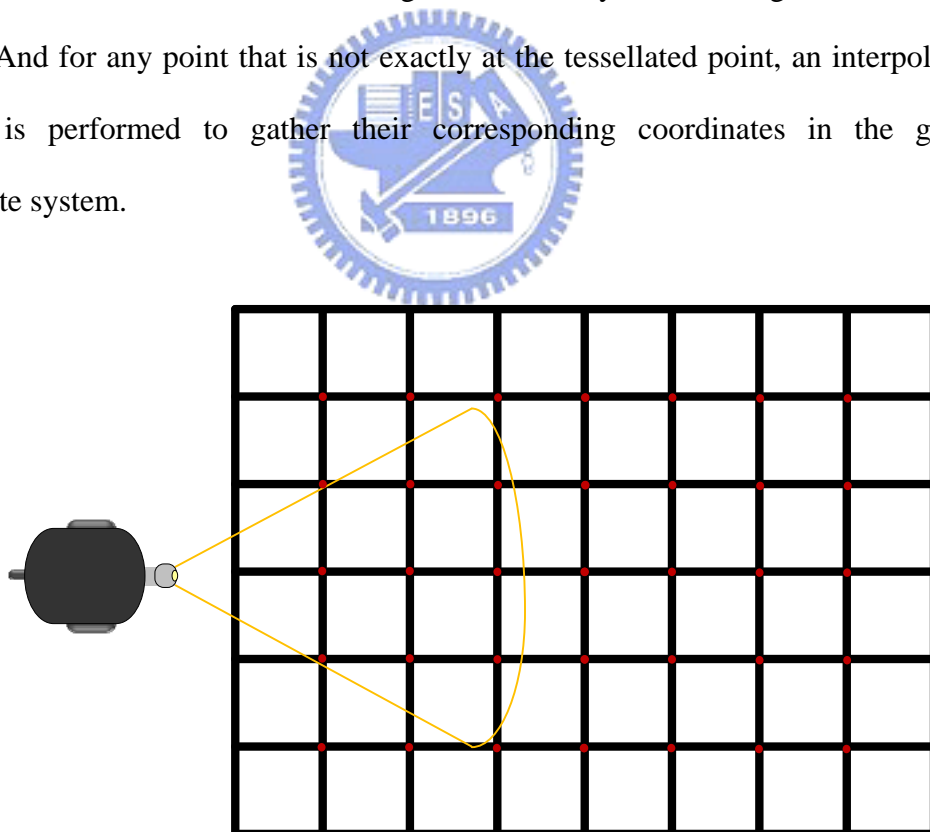


Figure 3.1 An illustration of attaching grids on the floor. A point set attached on the floor with coordinates known in the global coordinate system, as the red points.

## 3.2.2 Calibration of intrinsic parameters

In order to calibrate the camera, we use a program named CAMcal [14] to calibrate the intrinsic parameters of the camera. The program reads a sequence of images of planar checkerboard patterns. The checkerboard pattern consists of tiled black or white squares with 2cm edges, as shown in Figure 3.3(a). After reading the images, as shown in Figure 3.3(b), the program extracts the features from the images of patterns and matches them with those of the patterns themselves, as shown in Figure 3.2. Once the correspondences between the points in the images and those in the pattern are established, the camera's intrinsic parameters such as the focus length, the origin of the image in the image coordinate system, and the distortion coefficients can be computed automatically.

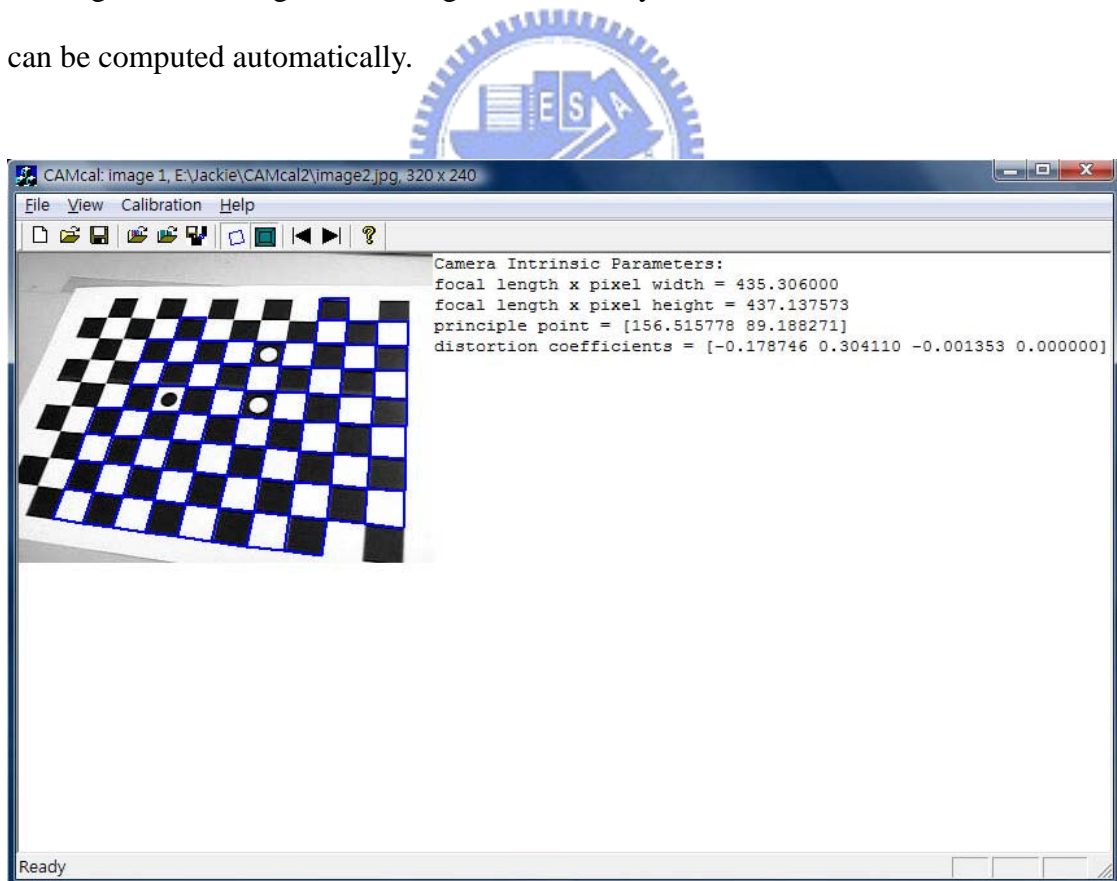


Figure 3.2 CAMcal user interface with the extracted grids result.

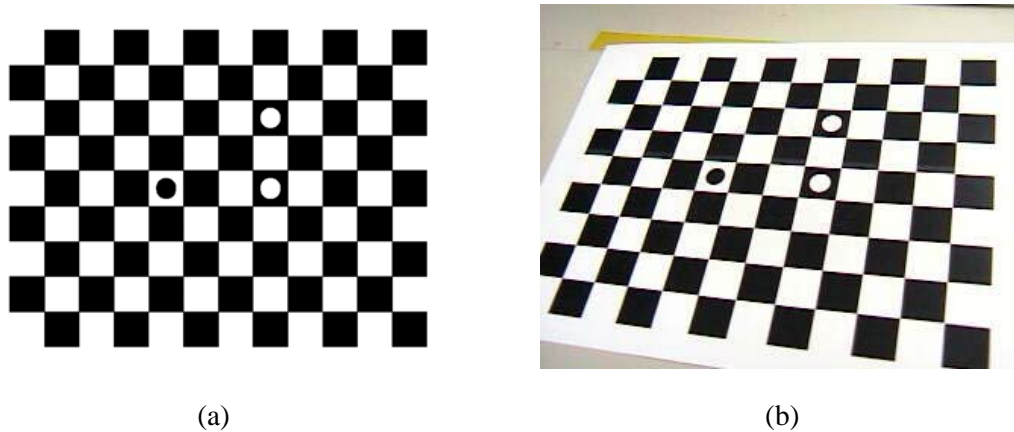


Figure 3.3 The checkerboard pattern used for the intrinsic parameters. (a) The printable checkerboard pattern. (b) The image of checkerboard pattern taken by the camera.

## 3.3 Learning of Specific Features

### 3.3.1 Learning of navigation paths composed of nodes

When the vehicle navigates in an open space by the control of a user, the odometer provides the current position data continuously. The position data consist of the vehicle coordinates  $(x, y)$  in the vehicle coordinates system and the direction angles  $\theta$  with respect to the vehicle's position and direction of the navigation starting point, respectively. We record both the vehicle coordinates and the direction angle as path data in this study.

In order to simply the data of learned paths, we only save the coordinates  $(x, y)$  which are called *node*  $N_i$  while the vehicle at one of the following three situations:

- (1) when the user controls the vehicle to turn;

- (2) when the vehicle detects a new color of the floor;
- (3) when the user controls the vehicle to learn a concerned object.

Each node includes the coordinates and the direction angle values and is labeled with a serial number. These nodes then form a graph of the learned path.

A control user interface is designed for user to drive the vehicle, as shown in Figure 3.4. When the user controls the vehicle to move to a desired place, the vehicle system will automatically collect the node data. After finishing learning, we have a set of notes, denoted as  $N_{path}$ . The process of recording the path data is described as an algorithm in the following.



Figure 3.4 Control user interface.

**Algorithm 3.1.** *Path node collection.*

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

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

*Steps:*

- Step 1. Record the first node as  $(x_0, y_0, \theta_0) = (0, 0, 0)$  into the set  $N_{path}$  and mark the node as  $N_0$  with index 0, when the vehicle is at the starting position.
- Step 2. Record the node  $N_i(x_i, y_i, \theta_i)$  into the set  $N_{path}$  by taking the values of the odometer  $(x, y)$  and the direction angle  $\theta$ , and label the node  $N_i$  with the next index number, when the vehicle is at one of the following three situations:
  - (4) when the user controls the vehicle to turn;
  - (5) when the vehicle detects a new color of the floor;
  - (6) when the user controls the vehicle to learn a concerned object.
- Step 3. Repeat Step 2 until the learning process is finished.
- Step 4. Record the finally node  $N_t$  into the set  $N_{path}$  and label it as  $N_t$  by the next index number.
- Step 5. Save all the nodes of the set  $N_{path}$  into the storage of the computer.

As an illustration of the result of applying Algorithm 3.1, we show an example of recorded nodes in Figure 3.5. It is shown that all nodes of the three situations are recorded in addition to the start and the end nodes. Each node is labeled with index numbers according to the order of patrolling. The index numbers are useful for path map creation and object monitoring.

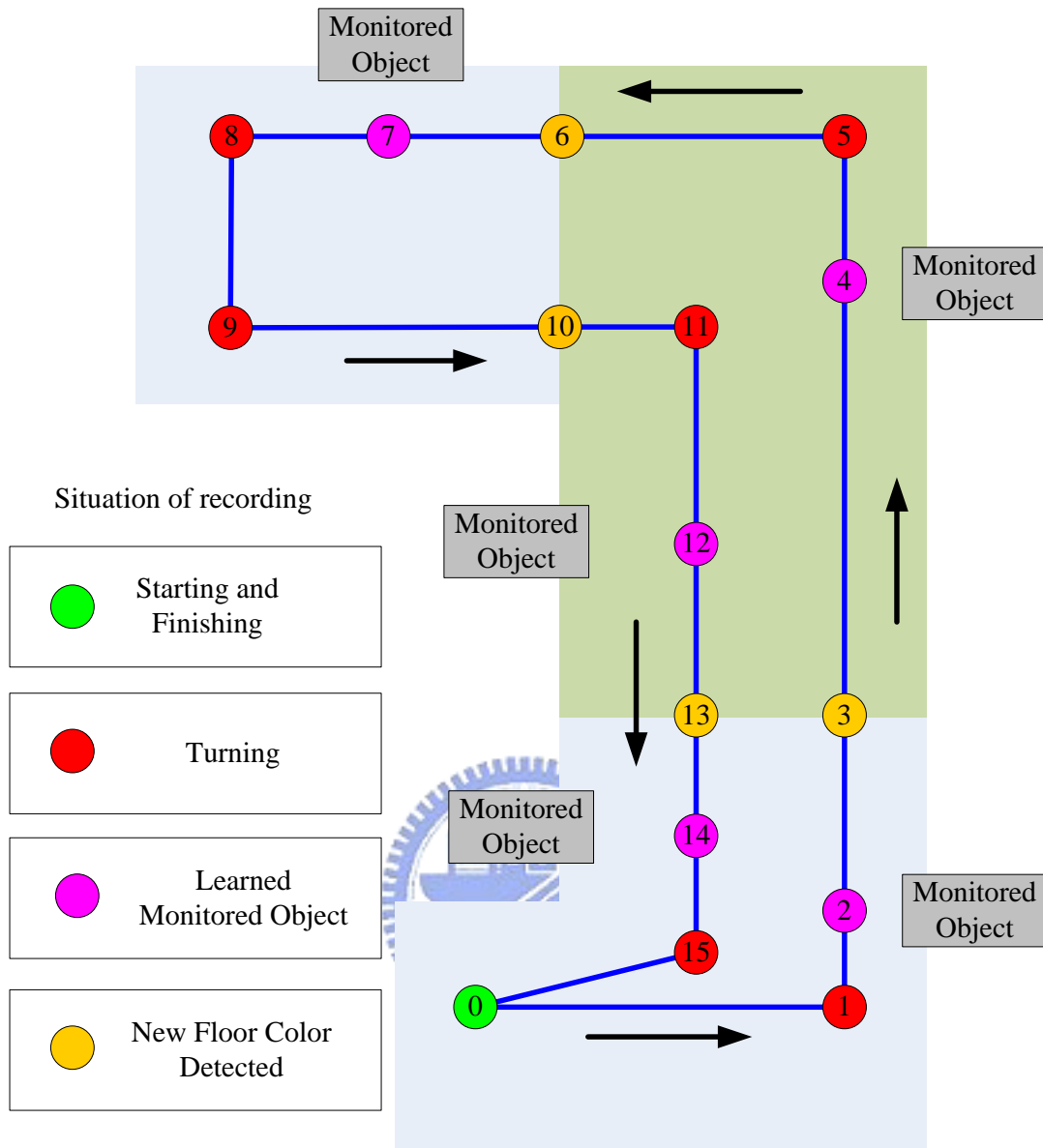


Figure 3.5 An illustration of learned data.

### 3.3.2 Learning of floor colors by k-means clustering

In order to detect an obstacle on various floors, we propose a method by learning the colors of the floors to distinguish between an obstacle and the floor. We design an algorithm to cluster the colors of the floor as features. One reasonable assumption is that there is no obstacle in the initial image captured by the camera. Under this assumption, we can apply a k-means clustering algorithm to compute the floor-color

features of the initial floor environment. Traditional clustering techniques usually choose the RGB space as the feature space. But the color distance in the RGB space usually can not represent the similarity of the similar color in the real world. Thus we first transform the image from the RGB space into the  $Lu'v'$  space. Then we can find the clusters in the 2D plane containing the chromatic value of the color and associate them with appropriate clusters in the 1D luminance space. The detailed algorithm will be described in Section 7.3. After clustering the colors of the floor successively, we can retrieve a set of the color features of the floor, denoted as  $C_{floor} = \{c_i\}_{i=1, \dots, n}$ , as shown in Equations (3.1) below, defining the main colors within the image. The color features set of the floor is useful in determination of obstacle existence and finding a collision-free path later.

$$\begin{aligned}
 C_{floor} &= \{c_i\}_{i=1, 2, \dots, n}; \\
 c_i &= \{L_i, u'_i, v'_i\}.
 \end{aligned}
 \tag{3.1}$$

While a user controls the vehicle to learn the navigation paths, the vehicle performs the above clustering algorithm periodically. In realistic environments, the colors of the floor and the illumination might change slightly while the vehicle moves, resulting in the changes of the color features; Therefore, we threshold the differences of the color features. If the difference of the color features is higher than a threshold parameter, a new *node* which represents this new floor is created automatically. And the vehicle can use this node to know where the new floor begins in the navigation phase. The process of detecting and recording the color data of a new floor is described as an algorithm in the following.



**Algorithm 3.2.** *The color of the new floor detection and recording by node creation.*

*Input:* A color image  $I$  and the threshold of the color feature differences  $\tau$

*Output:* A node which represents the beginning of a new floor.

*Steps:*

- Step 1. Apply the proposed *k-means clustering algorithm* on the image  $I$  to find the color feature set, denoted as  $C_{floor}$  in the initial state.
- Step 2. Capture a new image of the current environment, denoted as  $I_{new}$ .
- Step 3. Apply the *k-means clustering algorithm* to find the new color feature set of the image  $I_{new}$ , denoted as  $C_{new}$ .
- Step 4. Compute the difference between  $C_{new}$  and  $C_{floor}$  as  $C_{diff}$ .
- Step 5. If  $C_{diff}$  is greater than  $\tau$ , create a node with  $C_{new}$  and record it into the path by Algorithm 3.1 and update  $C_{floor}$  as  $C_{new}$ .
- Step 6. Repeat Steps 2 through 5 until the learning process is finished.

### 3.3.3 Learning of monitored objects by simplified

#### SIFT

In order to learn concerned objects, we design a user interface to help users specify the object which they want to monitor. While the user controls the vehicle to the front of the object to be monitored, they can move the PTZ camera toward the object. Then, they can select the object in the image by the use of the mouse connected to the computer to drag a rectangle as an interesting region to cover the object which appears in the image, as shown in Figure 3.6. Now, we apply the simplified SIFT algorithm which is described in Chapter 5 to obtain the feature set of the interesting region. Then, we save the vehicle location, the PTZ position, the feature set, and the interesting region into the storage of the computer. A flowchart is

illustrated in Figure 3.7, and the detail process is described in the following.



Figure 3.6 A red rectangle including the monitored object as an interesting region.

**Algorithm 3.3.** *Learning of a monitored object.*

*Input:* The position  $P$  of a monitored object.

*Output:* A monitored object information data.

*Steps:*

- Step 1. Drive the vehicle to the monitored object position  $P$ .
- Step 2. Move the PTZ camera toward the object and take an image  $I$ .
- Step 3. Drag a rectangle on the image  $I$  as an interesting region.
- Step 4. Apply the simplified SIFT on the interesting region to extract the feature set.
- Step 5. Save the vehicle location, the PTZ position, and the feature set of the interesting region as a monitored object information data.
- Step 6. Save the interesting region in the image.

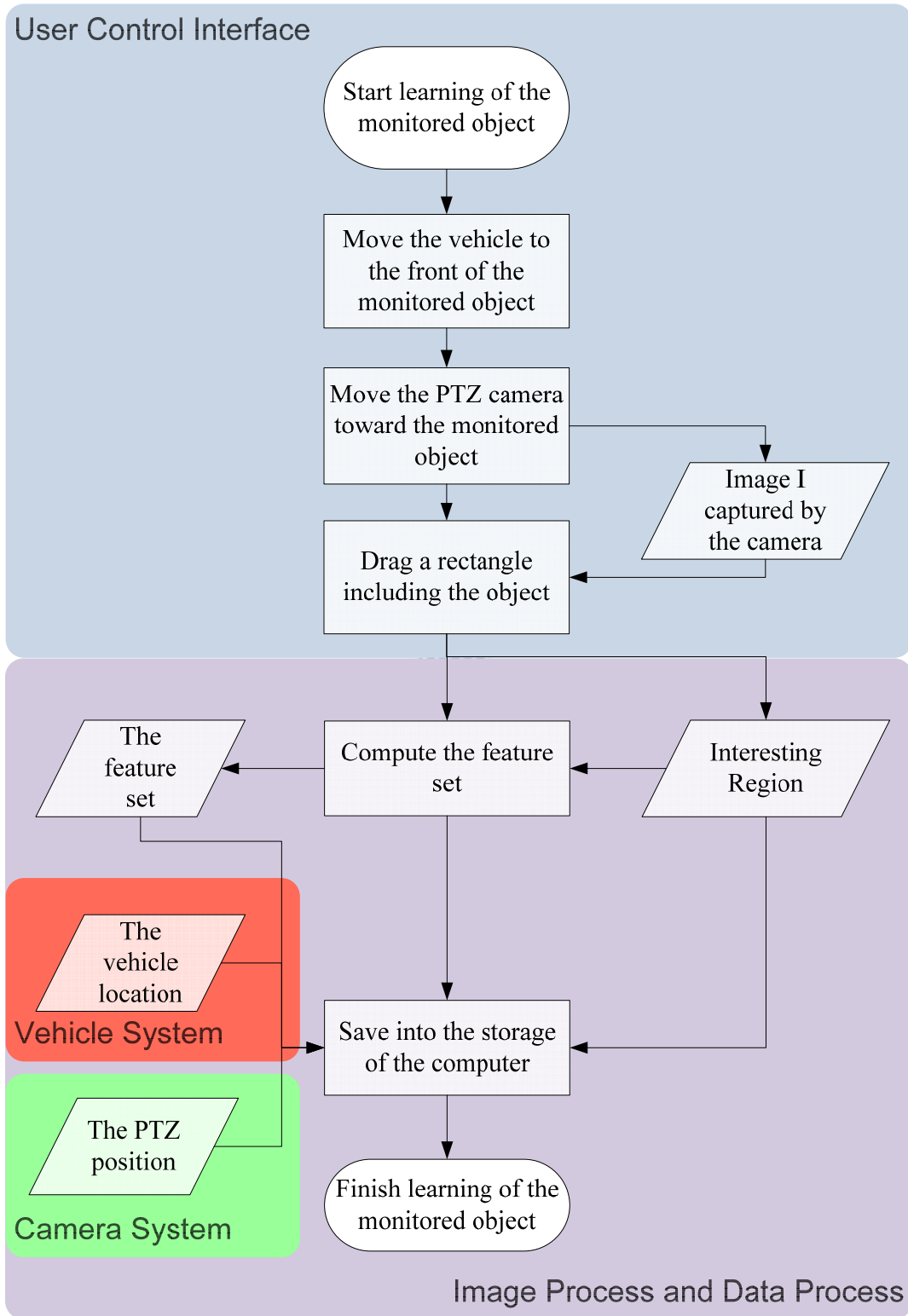


Figure 3.7 A flowchart of the learning monitored object process.

### 3.3.4 Learning of vehicle locations with respect to monitored objects

With successively monitoring of an object, we can extract an affine transformation between the image of the same monitored object in the navigation phase and the one found in the learning phase. During the learning phase, if we give a horizontal line, which is parallel to the floor plane in the 3D global coordinate system in the image, we can acquire the same line found in the image taken in the navigation phase by applying this affine transformation. Then, by some analytic mathematics analysis on this horizontal line found in the image taken in the navigation phase, we can obtain the relative position  $C(x_r, y_r)$  and the relative angle  $\theta_r$  of the vehicle in the world coordinate system with respect to the monitored object, as shown in Figure 3.8. The detailed algorithm is described in Chapter 6.

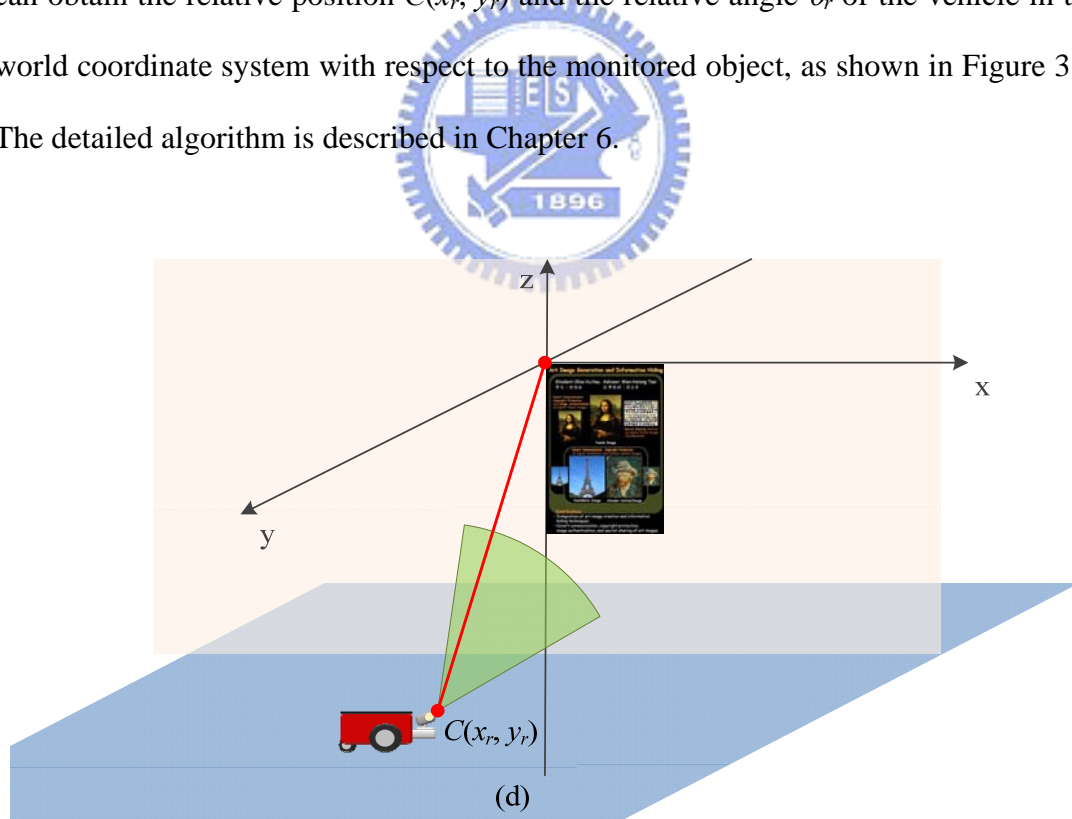
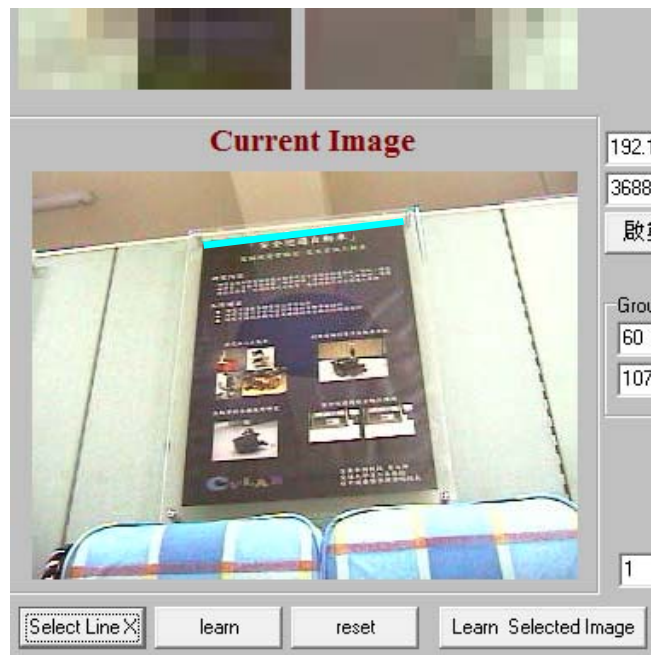


Figure 3.8 The relative position  $C(x_r, y_r)$  of the vehicle with respect to the start point in the world coordinate system.



(a)



(b)

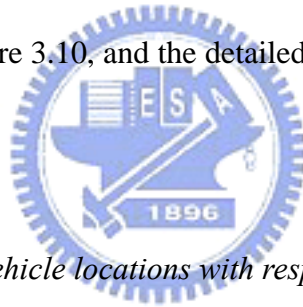
(c)

Figure 3.9 A user interface to specify the horizontal line. (a) A cyan line specified the horizontal line of the world coordinate system. (b) A zoom-in window for specifying the start point of the line. (c) A zoom-in window for specifying the end point of the line.

Hence, in order to gather the horizontal line, a user interface to help users specify the line is necessary. Based on the user interface designed for learning of monitored objects, we add two zoom-in windows for users to point out the start point  $(u_1, v_1)$  and

the end point  $(u_2, v_2)$  of the horizontal line in the image, as shown in Figure 3.9. Successively specifying the line, we can compute the linear equation's coefficient  $b$  and  $c$  by solving the equation  $u + bv + c = 0$  with  $(u_1, v_1)$  and  $(u_2, v_2)$ , and apply the proposed location estimation algorithm described in Chapter 6 to find the relative position  $C(x_r, y_r)$  and the relative angle  $\theta_r$ .

Now we have the relative position  $C(x_r, y_r)$  and the relative angle  $\theta_r$  between the monitored object and the vehicle location. The relative position is useful in adjusting the vehicle location while the vehicle patrols to the monitored object in the navigation stage. Thus, the last thing is to save the calibration information data including the start point  $(u_l, v_l)$ , the coefficients  $b$  and  $c$ , the relative position  $(x_r, y_r)$ , and the relative angle  $\theta_r$  for the use in adjusting the vehicle location in the navigation phase. A flowchart is illustrated in Figure 3.10, and the detailed learning process is described in the following.



**Algorithm 3.4.** *Learning of vehicle locations with respect to monitored objects.*

*Input:* A color image  $I$  including a monitored object and an interesting region selected during learning of the monitored object.

*Output:* A calibration information data.

*Steps:*

- Step 1. Select the start point  $(u_1, v_1)$  and the end point  $(u_2, v_2)$  of the horizontal line in the image  $I$  which is parallel to the floor plane in the 3D global coordinate system.
- Step 2. Compute the coefficients  $b$  and  $c$  by solving the equation  $u + bv + c = 0$  with  $(u_1, v_1)$  and  $(u_2, v_2)$ .
- Step 3. Apply the proposed location estimation algorithm described in Chapter 6 to

find the relative position  $C(x_r, y_r)$  and the relative angle  $\theta_r$  with respect to the start point in the 3D global coordinate system as an origin.

Step 4. Save the start point  $(u_1, v_1)$ , the coefficients  $b$  and  $c$ , the relative position  $(x_r, y_r)$ , and the relative angle  $\theta_r$  as calibration information data for a monitored object.

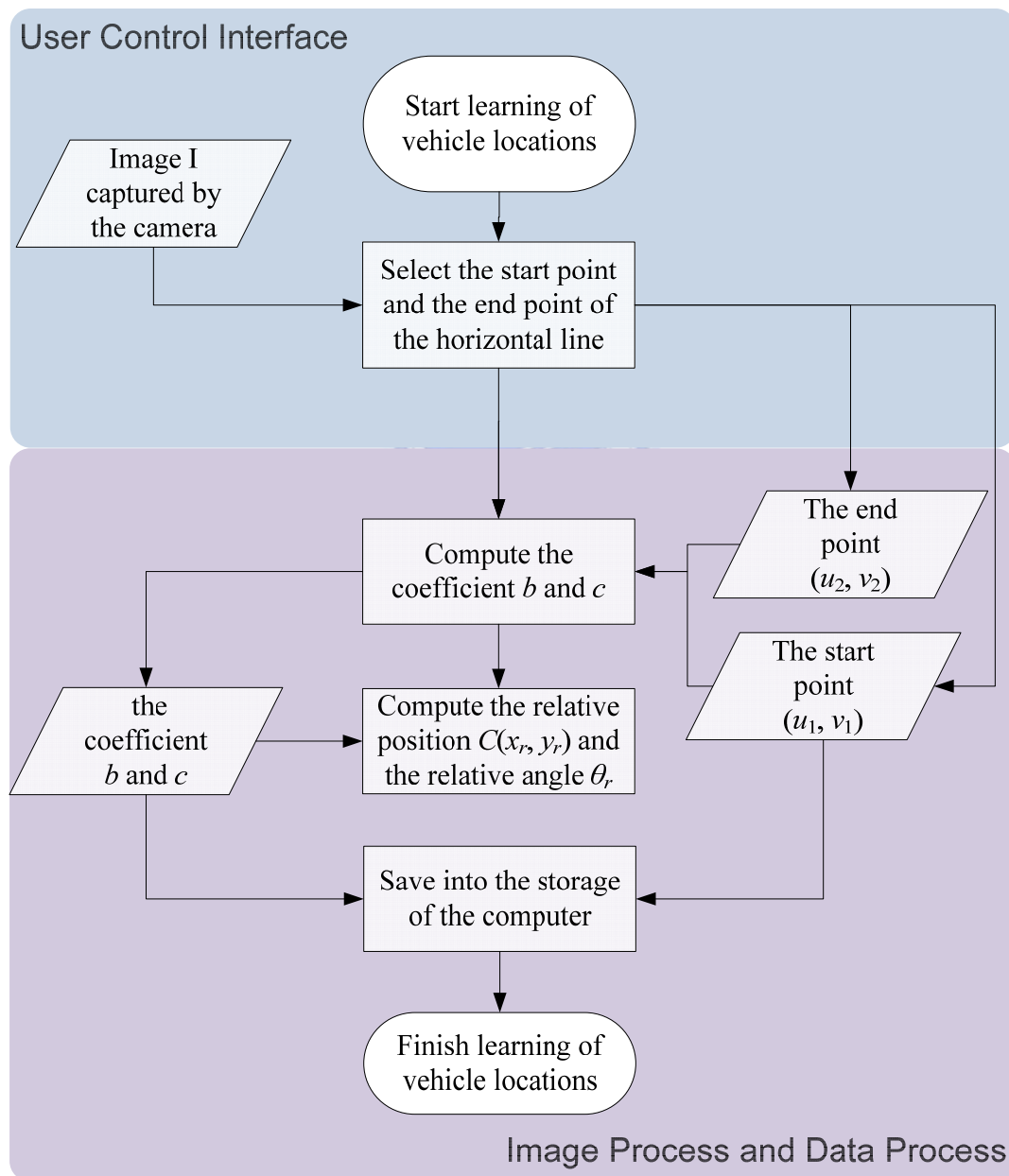


Figure 3.10 A flowchart of learning the vehicle locations process.

# Chapter 4

## Security Patrolling by Vehicle

### Navigation in Indoor Environments

#### 4.1 Introduction to Proposed Ideas

With successfully learned features mentioned in the previous chapter, the vehicle can navigate accordingly. In this chapter, we describe in more detail how security patrolling is accomplished by the proposed vehicle navigation scheme.

Firstly, security patrolling must be based on an effective navigation process. In Section 4.2.1, we describe how the vehicle is guided by the learned path. According to the node data of the path map which is created in the learning phase, the vehicle decides which direction to turn or how long it should advance. Secondly, in real applications, the environment is usually complicated and the floor typically consists of textures of various colors. In order to navigate in such environments, we propose an obstacle avoidance technique for various floor environments. We describe the proposed technique in Section 4.2.2. Thirdly, while the vehicle navigates, it usually suffers from incremental mechanic errors of the vehicle location provided by the odometer. Hence, we propose a location estimation method based on the monitored-object matching result to adjust the vehicle location in each navigation cycle. The detailed adjustment process is described in Section 4.2.3.

By the effective navigation process, the vehicle now can perform security patrolling to monitor concerned objects. In Section 4.3.1, we propose an object security monitoring process based on simplified SIFT using the learned feature set of



the monitored object. We also suggest some possible types of monitored objects in Section 4.3.2. Lastly, we give a summary in Section 4.

## 4.2 Ideas of Navigation Process

### 4.2.1 Guidance by learned paths

In a security patrolling process, the vehicle navigates along a learned path by visiting each path node consecutively. The learned navigation path, which consists of a set  $N_{path}$  of nodes is firstly read by the vehicle at the beginning. The first node of  $N_{path}$  is the starting node of the navigation path and specifies the current position of the vehicle. Then, the vehicle reads the next node data  $N_{i+1}(x_{i+1}, y_{i+1})$  and computes a turning angle and a moving distance by Equations (4.1) and (4.2) in the following algorithm, for the vehicle to move to the next position  $N_{i+1}(x_{i+1}, y_{i+1})$ , as shown in Figure 4.1. Repeating the same actions cycle after cycle, the vehicle can navigate along the learned path until all nodes have been visited.

**Algorithm 4.1.** *Process of vehicle guidance by a learned path.*

*Input:* A set  $N_{path}$  of nodes.

*Output:* Navigation cycles.

*Steps:*

- Step 1. Start vehicle navigation from the starting node  $N_0$  in  $N_{path}$ .
- Step 2. Scan  $N_{path}$  to read the next node  $N_{i+1}(x_{i+1}, y_{i+1})$ .
- Step 3. Read the position data provided by the odometer to gather the current vehicle location coordinates  $(x_{odo}, y_{odo})$  and the current direction angle  $\theta_{odo}$ .

Step 4. Compute a vector  $\vec{V}_i$  by using the following equation:

$$\vec{V}_i = \begin{bmatrix} X_i \\ Y_i \end{bmatrix} = \begin{bmatrix} x_{i+1} - x_{odo} \\ y_{i+1} - y_{odo} \end{bmatrix}. \quad (4.1)$$

Step 5. Compute the direction angle  $\theta_{new}$  for the vehicle to turn toward the node  $N_{i+1}$  by using the following equation:

$$\theta_{new} = \tan^{-1}\left(\frac{Y_i}{X_i}\right). \quad (4.2)$$

Step 6. Compute the rotation angle for the vehicle as  $\theta_{turn} = \theta_{new} - \theta_{odo}$  and the navigation distance for the vehicle to advance as  $d = |\vec{V}_i|$ .

Step 7. Turn the vehicle leftward for the angle  $\theta_{turn}$  if  $\theta_{turn}$  is greater than zero; otherwise, turn the vehicle rightward for the angle  $\theta_{turn}$ .

Step 8. Move the vehicle forward for the distance  $d$ .

Step 9. Read the next node data. If there exist remaining nodes, repeat Steps 3 through 8; else, finish the navigation.

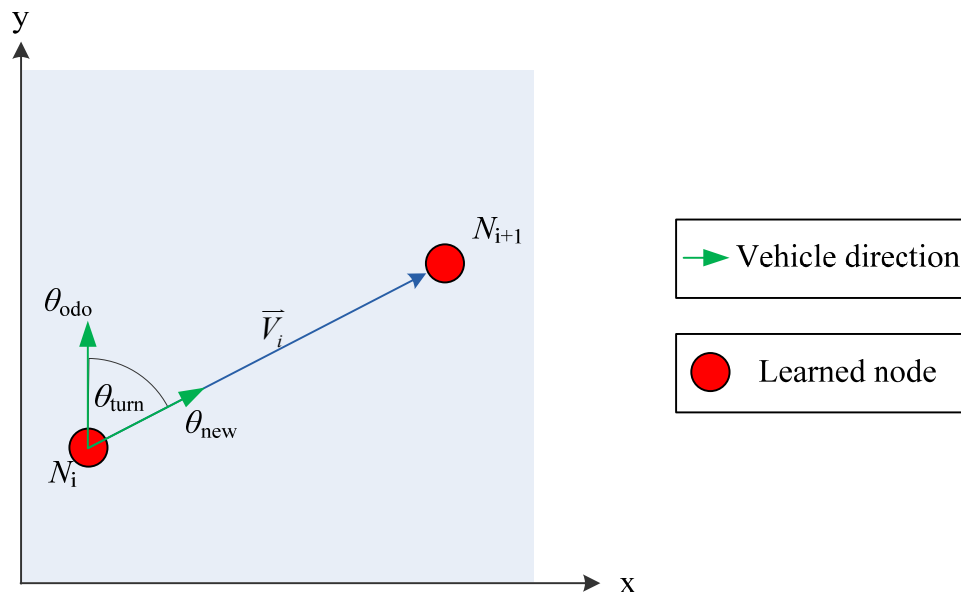


Figure 4.1 Computation of the turning angle and move distance.

## 4.2.2 Obstacle avoidance in various floor environments

The first issue involved in obstacle avoidance is obstacle detection. For obstacle detection in various floor environments, we utilize the colors of the floor. More specifically, we adopt *k-means clustering* and a so-called *alert line scanning* technique to recognize an obstacle on floors of various colors. The detail of the proposed obstacle detection method is described in Section 7.3.2, and the detail of the adopted k-means clustering algorithm is described in Section 7.3.1. If an obstacle is detected on a navigation path, a new local path is computed to avoid the obstacle. In this study, we utilize the goal-directed minimum path technique [3] to create the new local path. The goal of this new local path is to guide the vehicle toward the next node of the navigation path without colliding the obstacle. The process of obstacle avoidance is described in the following algorithm and a corresponding flowchart is shown in Figure 2.2.

**Algorithm 4.2.** *Process of obstacle avoidance.*

*Input:* A color image  $I$ .

*Output:* Navigation cycles.

*Steps:*

- Step 1. Apply the proposed *k-means clustering* algorithm described in Section 7.3.1 to the colors of the pixels in image  $I$  to get color clusters as the color feature set of  $I$ .
- Step 2. Scan the alert line to detect if an obstacle exists.
- Step 3. If an obstacle exists, find the distribution of the obstacle; otherwise, repeat

Steps 1 through 2 until an obstacle is found.

Step 4. Detect if the navigation path is impeded by any obstacle.

Step 5. If an obstacle impedes the navigation path, compute the minimum path for the vehicle to avoid the obstacle; otherwise, repeat Steps 1 through 4 until an obstacle is found to impede the navigation path.

Step 6. Drive the vehicle according to the new path.

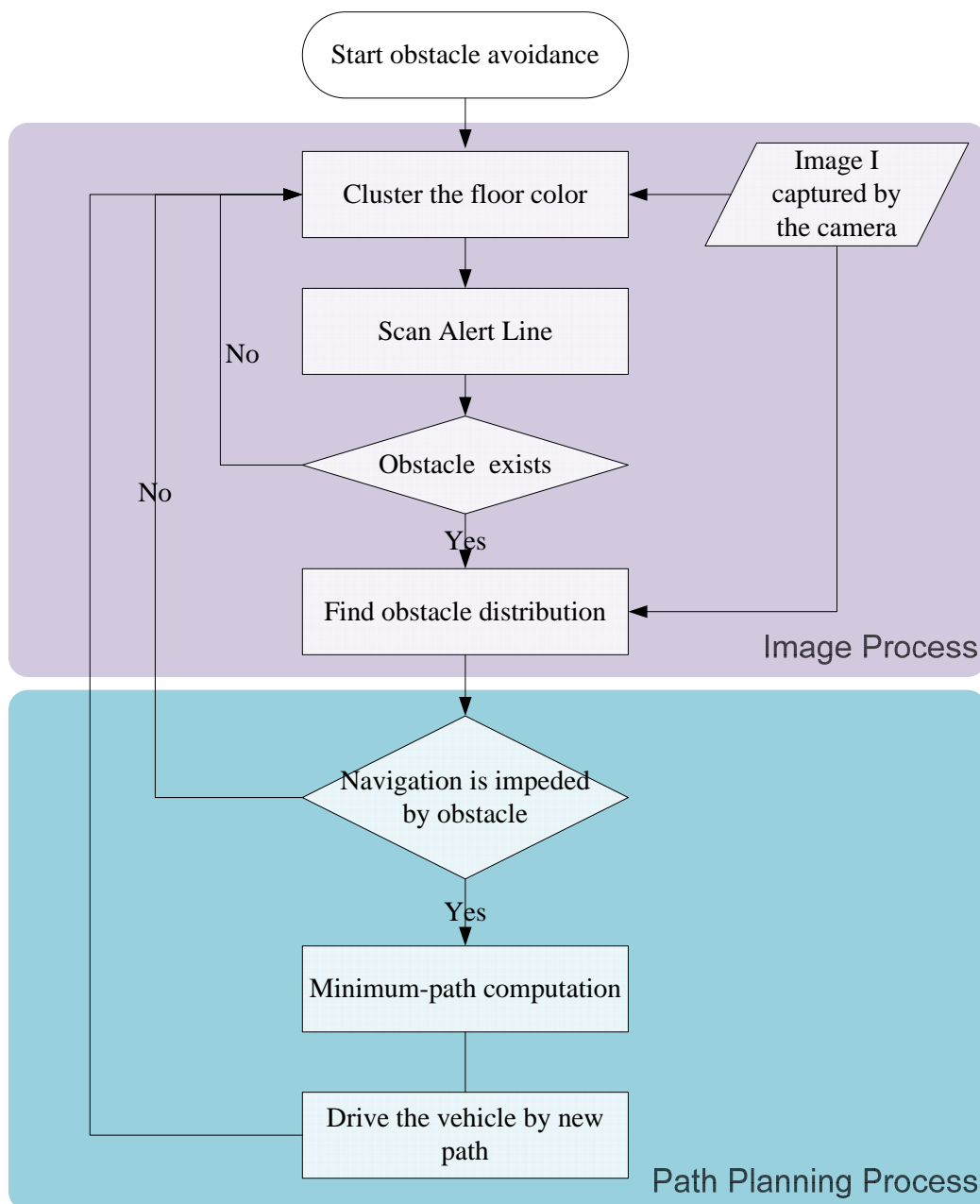


Figure 4.2 Flowchart of obstacle avoidance process.

## 4.2.3 Path correction by monitored object image matching

After successfully monitoring an object, we can take advantage of the matching result to adjust the vehicle location. The matching result includes a set of matched pairs of features. We define accordingly an affine transform by the use of the set of matched pairs. Then we can apply this affine transform to transform a horizontal line, which is learned in the learning phase, into the one which is found in the image taken by the camera in the navigation phase, as shown in Figure 4.3.

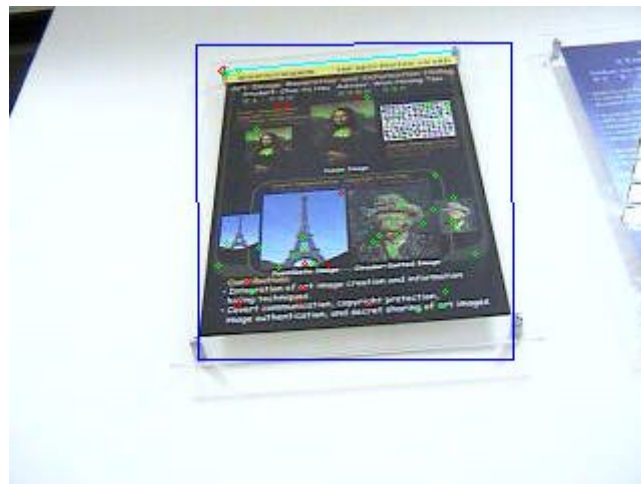


Figure 4.3 A cyan line (on the top of the poster) generated by applying the affine transform.

Now, we have the same horizontal lines which appear both in the images of the monitored object in the learning and in the navigation phases. Let the horizontal line which is found in the navigation phase be denoted as  $l_{navi}$  and the one which was found in the learning phase be denoted as  $l_{learn}$ . By applying the proposed location estimation algorithm which is described in Chapter 6, we can gather the relative position  $C(x_{r,navi}, y_{r,navi})$  and the relative angle  $\theta_{r,navi}$ , between the current vehicle location and the monitored object position. Then, we can adjust the vehicle location

by comparing the position  $C(x_{r,navi}, y_{r,navi})$  and the angle  $\theta_{r,navi}$  with the learned position  $C(x_r, y_r)$  and the learned angle  $\theta_r$  which are computed by  $l_{learn}$  in the learning phase. The detailed adjustment algorithm is described in Chapter 6. A flowchart is illustrated in Figure 4.4, and the detailed path correction process is described as an algorithm in the following.

**Algorithm 4.3.** *Process of path correction.*

*Input:* The monitored-object matching results and the learned calibration information including the horizontal line, the relative position  $C(x_r, y_r)$ , and the relative angle  $\theta_r$ .

*Output:* the correction process.

*Steps:*

- Step 1. Extract the affine transformation from the monitored-object matching results.
- Step 2. Apply the affine transformation to gather the new horizontal line in the current image.
- Step 3. Apply the proposed location estimation algorithm described in Chapter 6 by the new horizontal line and the current PTZ position, to find the relative position  $C(x_{r,navi}, y_{r,navi})$  and the relative angle  $\theta_{r,navi}$ , between the current vehicle location and the monitored object position.
- Step 4. Compute the correction data to adjust the vehicle by comparing respectively  $C(x_{r,navi}, y_{r,navi})$  and  $\theta_{r,navi}$  with  $C(x_r, y_r)$  and  $\theta_r$  which are learned in the learning phase.
- Step 5. Adjust the vehicle according to the correction data.



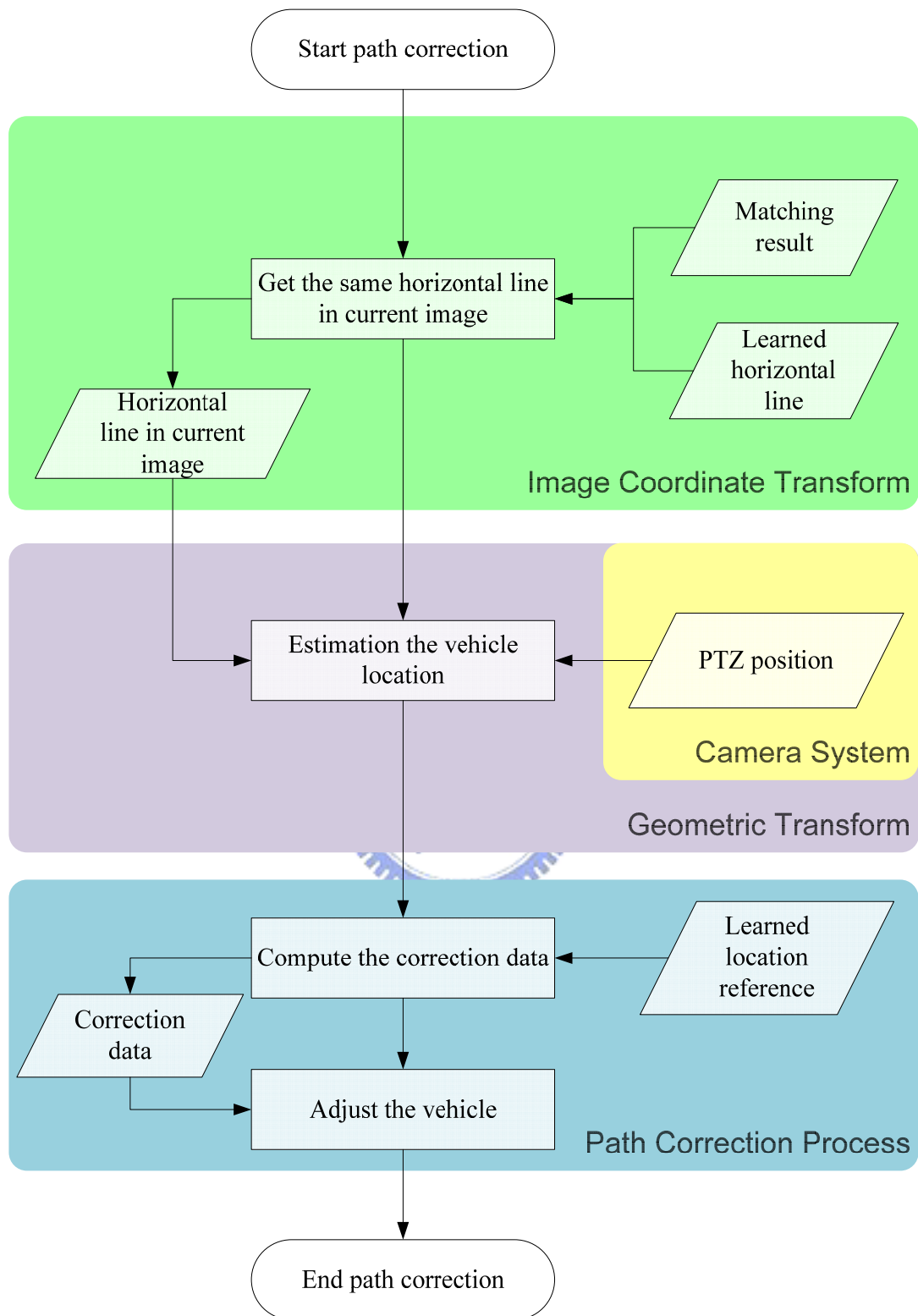


Figure 4.4 Flowchart of path correction process.

## 4.3 Purposes of Security Patrolling

### 4.3.1 Proposed technique for monitoring of objects

In this section, we briefly describe the proposed security object monitoring technique. With successively learned concerned objects, we have stored the feature set of each monitored object, the PTZ position which let the PTZ camera face the monitored object, and the interesting region which is specified in the learning phase. Therefore, in the navigation phase, we firstly move the vehicle to the security monitoring node and move the camera to face the monitored object according to the learned PTZ position. Then we apply the simplified SIFT algorithm to the image taken by the camera to extract a feature set  $F$ . By applying the Hough transform, we can find the matching pairs of the features between the feature sets  $F$  and  $F_{\text{learn}}$ . If no matching pair is found, the vehicle will issue an alarm message. The detailed simplified SIFT algorithms and Hough transform for matching are described in the next chapter. A flowchart of the object monitoring process is illustrated in Figure 4.5, and the detailed process is described as an algorithm in the following.

**Algorithm 4.4.** *Process of object monitoring.*

*Input:* A color image  $I$  taken by the camera, the learned PTZ position, and the learned feature set  $F_{\text{learn}}$  of the monitored object.

*Output:* An alarm message or nothing.

*Steps:*

- Step 1. Move the vehicle to the monitoring node according to the learned navigation data.
- Step 2. Move the camera to the learned position and let the camera face the



monitored object.

Step 3. Take a color image  $I$  and apply the simplified SIFT on it to extract a feature set  $F$ .

Step 4. Apply the Hough transform to find the matching pairs of the features between  $F$  and  $F_{\text{learn}}$ .

Step 5. If no matching pair is found, issue an alarm message; otherwise, finish the monitoring process.

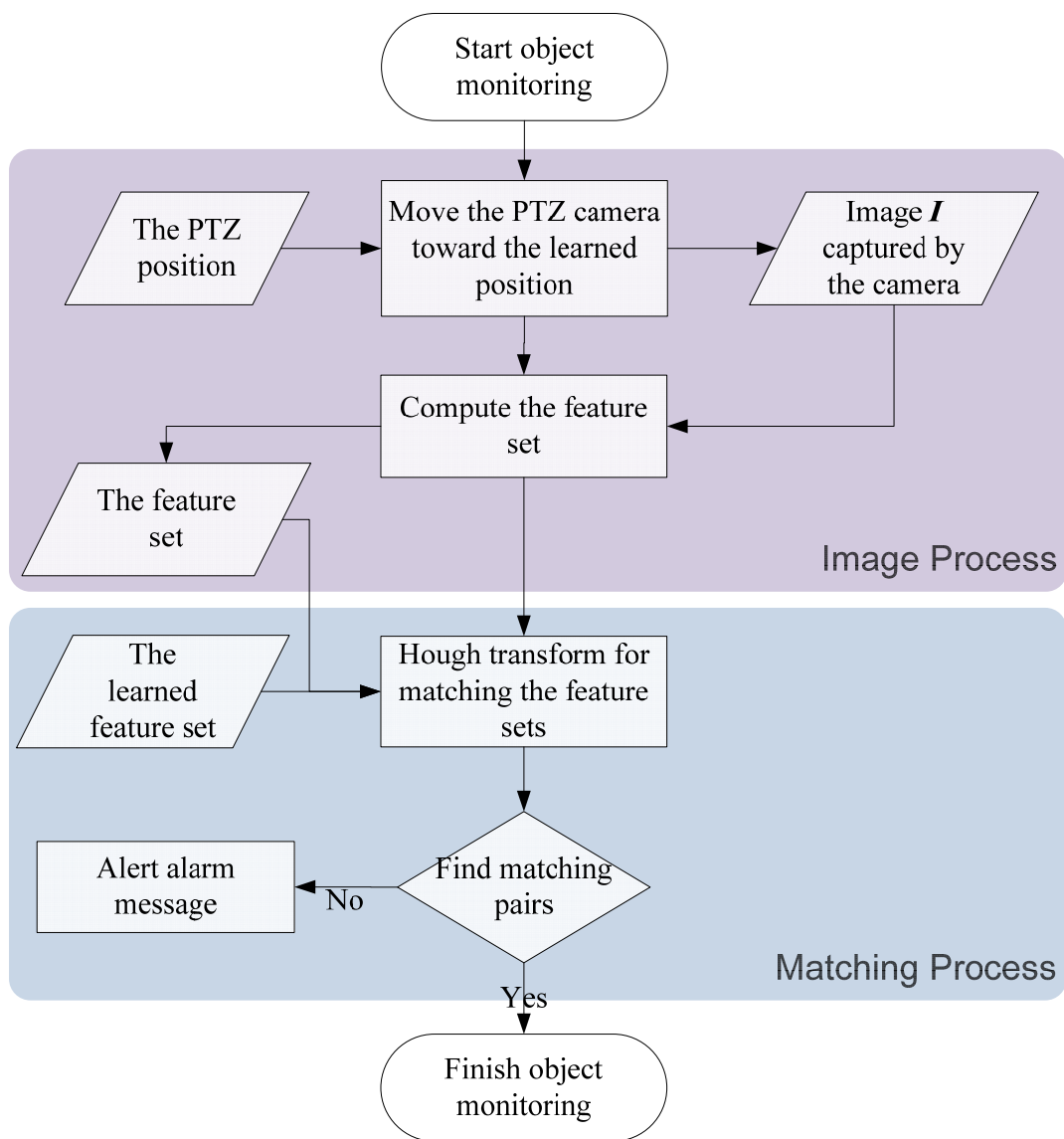


Figure 4.5 Flowchart of object monitoring process.

## 4.3.2 Types of monitored objects

In this study, we propose a simplified SIFT algorithm to transform the image into the feature set and matching the features for monitoring concerned objects. Based on the previous work of the SIFT algorithm, the used features should be invariant to image scaling, translation, and rotation, and partially invariant to illumination changes and affine or 3D projection. We transform a monitored-object image into such feature sets for monitoring. Hence, possible types of monitored objects are no longer restricted to objects with ideal shapes or objects in pure-colored background environments. An example of concerned objects in complex backgrounds is shown in Figure 4.6.



Figure 4.6 An example of a concerned object, specified as the blue region, in a complex background environment.

The main advantage of using the SIFT key as the matched features is that the SIFT algorithm only extracts the main characteristics of images. Whatever the viewing angles of the same object changes, in a tolerance, the main characteristics of the object will remain unchanged for the SIFT algorithm to extract and match. Here

we list some reasonable possible types of monitored objects in the following:

1. Valuable artworks on the wall of an exhibition halls,
2. Coffers in the cabinet,
3. Modern flat-panel TVs on walls,
4. Any planar objects.

## 4.4 Detailed Process for Security Patrolling by Vehicle Navigation

In this section, we summarize the detailed process for security patrolling. In the navigation phase, the vehicle navigates along a learned path by visiting each path node consecutively through the routes specified by the node edges and checks the existence of the learned objects to achieve security patrolling. The entire process is described in the following as an algorithm and a corresponding flowchart is shown in Figure 4.7.

**Algorithm 4.5.** *Security patrolling navigation.*

*Input:* A path map, learned object data, and learned calibration data.

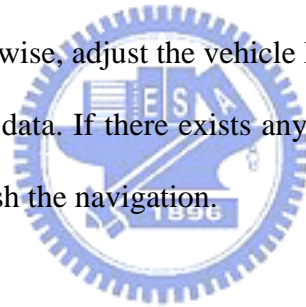
*Output:* Navigation process.

*Steps:*

- Step 1. Read the path map.
- Step 2. Start security patrolling from the starting node in the map.
- Step 3. Scan the node list of the path map to read the next node data.
- Step 4. Move the vehicle to the next node and check if an obstacle is impeding the

navigation path.

- Step 5. If there exists such an obstacle, compute a new local path and drive the vehicle accordingly to avoid the obstacle.
- Step 6. If there exist a new floor data in the current node, update the reference color of the floor.
- Step 7. If there exist a monitored object in the current node, take the following action; else, continue the remaining navigation.
  - Step 7.1. Move the camera to the learned PTZ position.
  - Step 7.2. Apply simplified SIFT to the image taken by the camera to extract a feature set and match them with the learned one.
  - Step 7.3. If the feature set does not matched the learned one, send an alarm message; otherwise, adjust the vehicle location by the matching result.
- Step 8. Read the next node data. If there exists any remaining node, repeat Steps 3 through 7; else, finish the navigation.



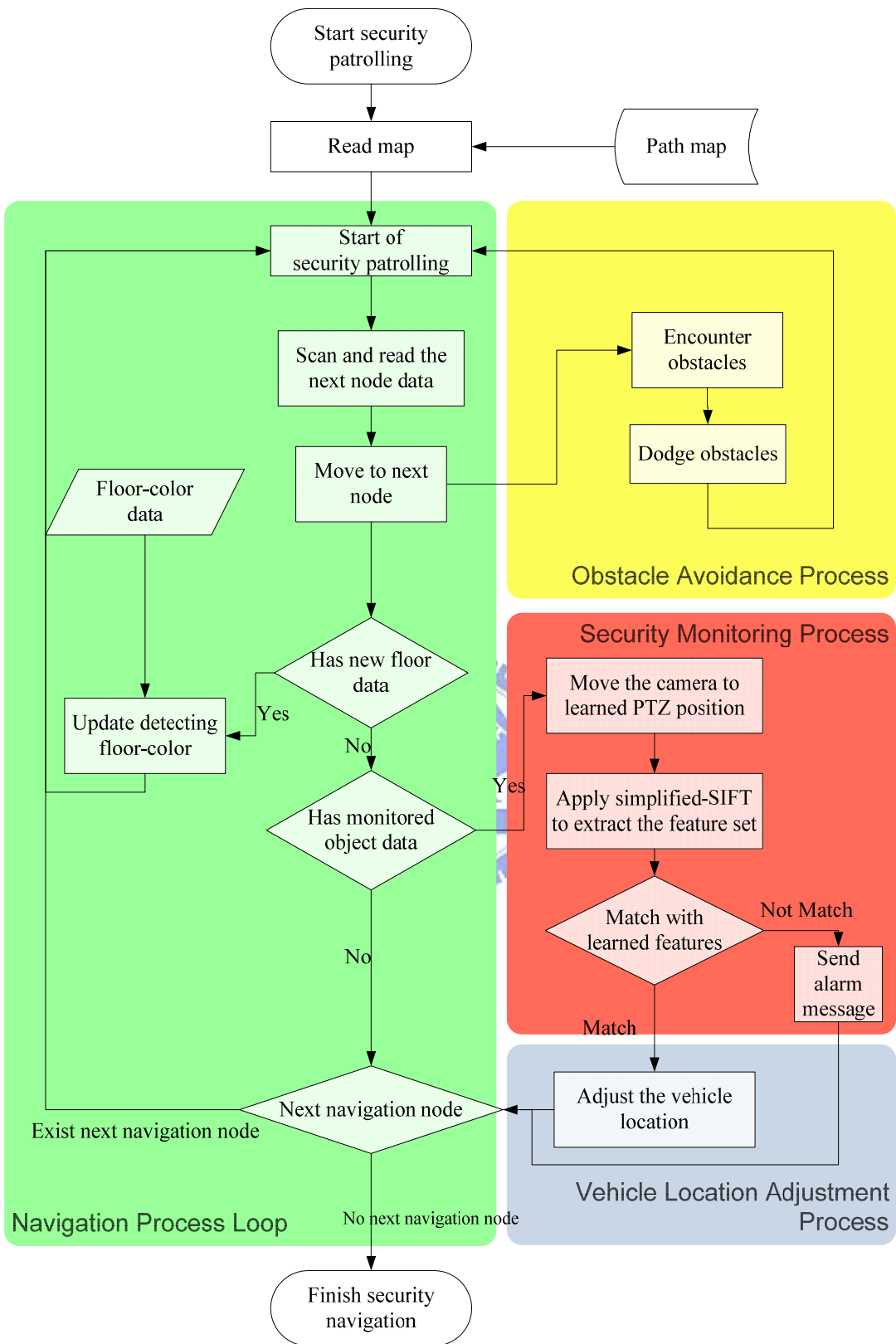


Figure 4.7 Flowchart of proposed navigation process.

# Chapter 5

## Detection of Monitored Objects by 2D Object Image Matching

### 5.1 Introduction

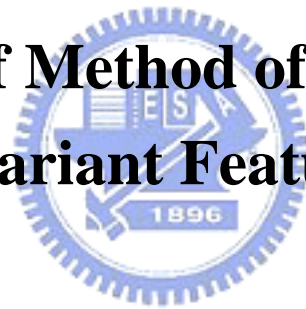
In this study, we design a vision-based security monitoring system by vehicle navigation equipped with a PTZ camera. By the means of the camera, the view of the vehicle is extended to a wider range. We can perform security monitoring on objects which are located higher than the camera. While the vehicle patrols in the navigation phase, it stops in front of the monitored object by the use of learned path nodes. But the stop position at a monitored object may not be precise every time; mostly just close to the one recorded in the learning phase. It results in slight changes in the viewing angle of the monitored object from the camera. And the image of the same monitored object will be different in scales, orientations, or positions with respect to the one taken for learning in the learning phase. Thus, a method with the ability to match corresponding objects in images taken with different illuminations and camera poses is needed.

In the past years, the Scale Invariant Feature Transform (SIFT) has been proven to be one of the most robust methods which use local invariant feature descriptors with respect to different geometrical changes [15]. In order to allow efficient matching between images, all images are represented as a set of vectors, called SIFT features. Each SIFT feature consists of local image measurements invariant to image translation, scaling, and rotation, and partially invariant to illumination and 3D

viewpoint changes. In this study, we take advantage of the SIFT to match monitored object images and propose a simplified SIFT which is faster than the original one, by reducing the difference of Gaussian scale layers to meet real-time security monitoring needs.

In Section 5.2, we firstly review the method of the SIFT proposed by Lowe [8]. Then, in Section 5.3, we describe the proposed simplified SIFT, including the necessity of simplification and the detailed process for the simplified SIFT feature generation. In Section 5.4, we describe the matching technique of the features. Lastly, some experimental results are shown in Section 5.5.

## **5.2 Review of Method of Matching by Scale-Invariant Feature Transform (SIFT)**



The SIFT proposed by Lowe [8] includes four major stages of computation to generate the set of features, which were called keypoints. In this section, we will describe a brief review of the SIFT. In summary, we divide the SIFT into two parts: the scale-invariant keypoint localization and the keypoint descriptor generation.

Firstly, the keypoint locations are efficiently detected by identifying the local maxima and minima of a difference-of-Gaussian (DoG) function in the scale space. At each location, an orientation is selected at a peak of a histogram of local image gradient orientations. Secondly, a keypoint is formed by measuring the local image gradients in a region around each keypoint's location according to the location, scale and orientation of the keypoint. The details are described in Section 5.2.1 and 5.2.2.

## 5.2.1 Scale-invariant keypoint localization

The first stage of computation searches over all scales and image locations. It is implemented by using a difference-of-Gaussian function to identify potential interest points that are invariant to scale and orientation changes.

Given a Gaussian-blurred image,  $G(x, y, \sigma)$  with an input image,  $I(x, y)$ , the scale space of an image is defined as a function  $L(x, y, \sigma)$  computed from the convolution of  $G(x, y, \sigma)$  and  $I(x, y)$ :

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \quad (5.1)$$

where  $*$  is the convolution operation in  $x$  and  $y$ , and

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}. \quad (5.2)$$

To efficiently detect stable keypoint locations in the scale space, Lowe makes use of the scale-space extrema in the difference-of-Gaussian function convolved with the image,  $D(x, y, \sigma)$ , which can be computed from the difference of two nearby scales separated by a constant multiplicative factor  $k$ :

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned} \quad (5.3)$$

Applying the equations above repeatedly, the input image is incrementally convolved with the Gaussian to produce images separated by  $k$  in the scale space, as shown stacked in the left of Figure 4.7. And each octave of the scale space (i.e., doubling of  $\sigma$ ) is divided into an integer number  $s$  of intervals, so  $k = 2^{1/s}$ . Then, the



difference-of-Gaussian images are produced by subtracting the adjacent Gaussian images, as shown in the right of Figure 4.7. After each octave is constructed, the Gaussian image is down-sampled by a factor of 2, and the process repeats.

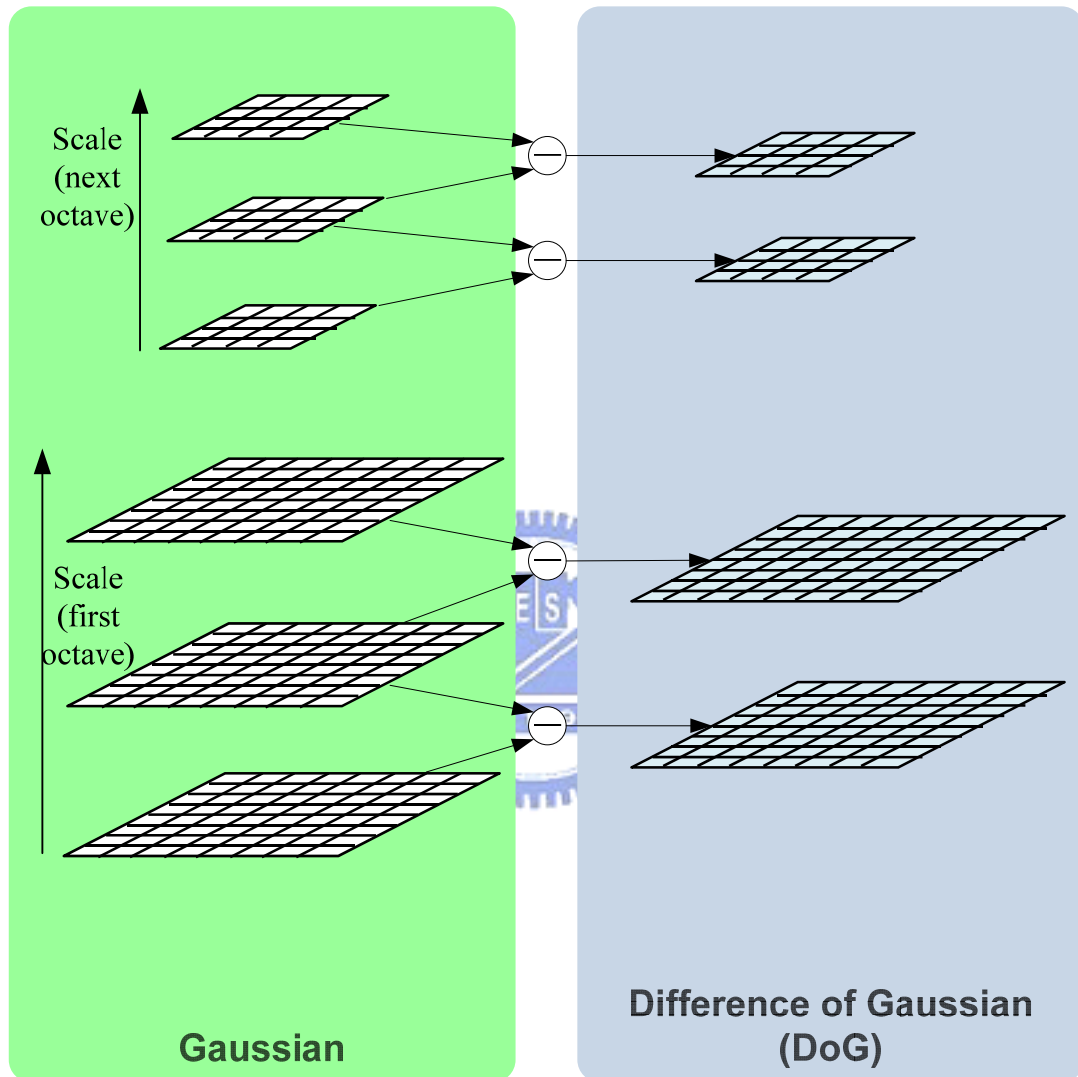


Figure 5.1 For each octave of scale space, the set of scale space images shown at the left and the computation of the difference-of-Gaussian images at the right.

The keypoint are identified as local maxima or minima of the DoG images across scales. Each sample point in the DoG images is compared with its 8 neighbors in the same scale image, and the 9 corresponding neighbors in neighboring scale images, as shown in Figure 5.2. If the sample point is a local maximum or minimum, it is

selected as a candidate keypoint.

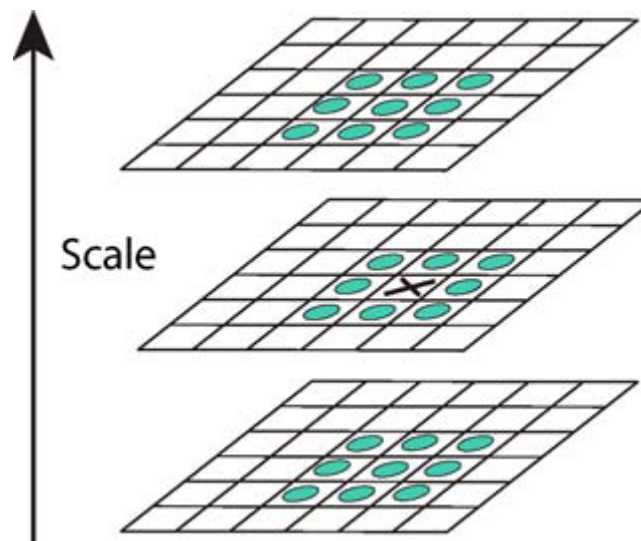


Figure 5.2 Maxima and minima of the difference-of-Gaussian images are detected by comparing a pixel, as marked with X, with its 26 neighbors, as marked with the blue circles, in 3x3 regions of the current and adjacent scales (from [8]).

Once a keypoint candidate has been found by comparing a pixel with its neighbors, the final keypoints are selected based on measures of their stability by performing a detailed modeling to fit the nearby data for location, scale, and ratio of principal curvatures. This information allows points having low contrast or being localized along an edge to be rejected.

## 5.2.2 Feature descriptor generation

One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed according to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations. To determine the keypoint orientation, a gradient orientation histogram is computed in the

neighborhood of the keypoint, using the Gaussian image at the closest scale to the keypoint's scale. For each Gaussian image,  $L(x, y)$ , at such a scale, the gradient magnitude,  $m(x, y)$ , and orientation,  $\theta(x, y)$ , are computed using pixel differences, as described by the following equations:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}, \quad (5.4)$$

$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y))). \quad (5.5)$$

The contribution of each neighboring pixel is weighted by the gradient magnitude and a Gaussian window with a  $\sigma$  that is 1.5 times the scale of the keypoint. And the orientation histogram is formed by 36 bins covering the 360 degree range of orientations. Then, peaks in the histogram will correspond to dominant orientations. If any other orientations that is within 80% of the highest peak is found, a separate keypoint is also created for that orientation.

Once a keypoint orientation has been selected, the feature descriptor is computed as a set of orientation histograms over  $4 \times 4$  subregions around the keypoint. The image gradient magnitudes and orientations are firstly sampled around the keypoint location, using the Gaussian image at the closest scale to the keypoint's scale. The coordinates of the descriptor and the gradient orientations are rotated according to the keypoint orientation for orientation invariance. Then, the contribution of each sample is weighted by the gradient magnitude and a Gaussian weighting function with  $\sigma$  equal to 1.5 times the width of the descriptor window, as the circular window indicated in the left side of Figure 5.3. These samples are then accumulated into orientation histograms summarizing the contents over  $4 \times 4$  subregions. Each orientation histogram contains 8 bins, and each descriptor contains an array of 4

histograms around the keypoint, as shown in the left side of Figure 5.3. Hence, an SIFT feature vector contains  $4 \times 4 \times 8 = 128$  elements. This vector is then normalized to enhance invariance to changes in illumination.

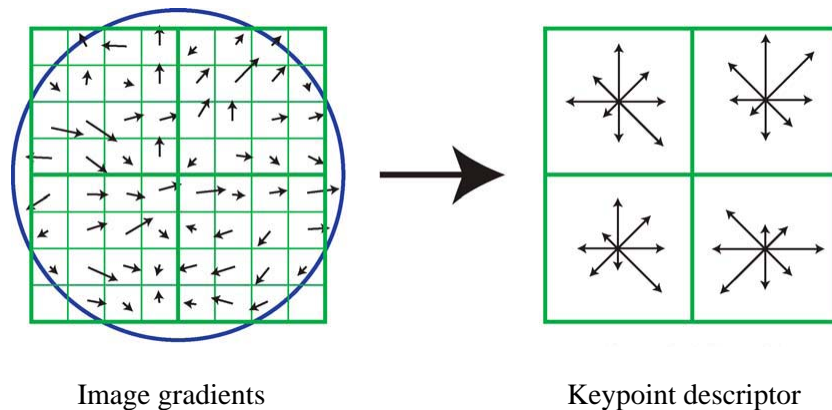


Figure 5.3 For each image, the gradient magnitude and orientation are computed in a region around the keypoint location, and weighted by a Gaussian window, indicated by the overlaid circle, as shown on the left. Four orientation histograms summarize their contents into 8 bins, as shown on the right. (from [8]).

## 5.3 Proposed Simplified SIFT Features for Detection of Monitored Objects

### 5.3.1 Necessity of simplification of original concept

The time consumption of the process of the SIFT algorithm can be divided into two parts: the processing time for feature localization and the processing time for feature descriptor generation. The first part is bounded by the size of the input image and the process layers specified by the number of intervals and octaves, and the second part is bound by the number of features and the dimensions of each feature

descriptor. In this study, the image captured by the camera of the proposed system is of a fixed resolution of  $320 \times 240$  pixels. Hence, in the first part, we can only control the number of intervals and octaves to reduce the processing time. In the second part, because the number of feature is uncontrollable, and the low dimension may result in unstable matching results, so we do not simplify the process of the feature descriptor generation.

For security monitoring, while the vehicle navigates to the monitoring node which is learned in the learning phase, the position of the monitored object will be close to the one found in the learning phase. Hence, the scale of the monitored-object image will not change too much.

Therefore, in order to speed up the process, we conducted experiments on reducing the number of octaves. And the experimental results show that the influence of reducing the number of octaves on the matching results is insignificant.

### 5.3.2 Detailed process of simplified SIFT feature generation

In this section, we describe the detailed process of the simplified SIFT feature generation. The inputs of the generation process are a color image  $I$ , a number  $o$  of octaves, a number  $s$  of intervals, a contrast threshold  $c$ , and a curvature threshold  $r$ . We can divide the entire process into 5 parts: the color-to-grayscale conversion, the Gaussian and DoG pyramids construction, the feature localization, the orientation assignments, and the feature descriptor generation. A flowchart is illustrated in Figure 5.4. After the process is finished, a set of simplified SIFT features of the given image  $I$  is generated. The detailed process is described as an algorithm in the following.

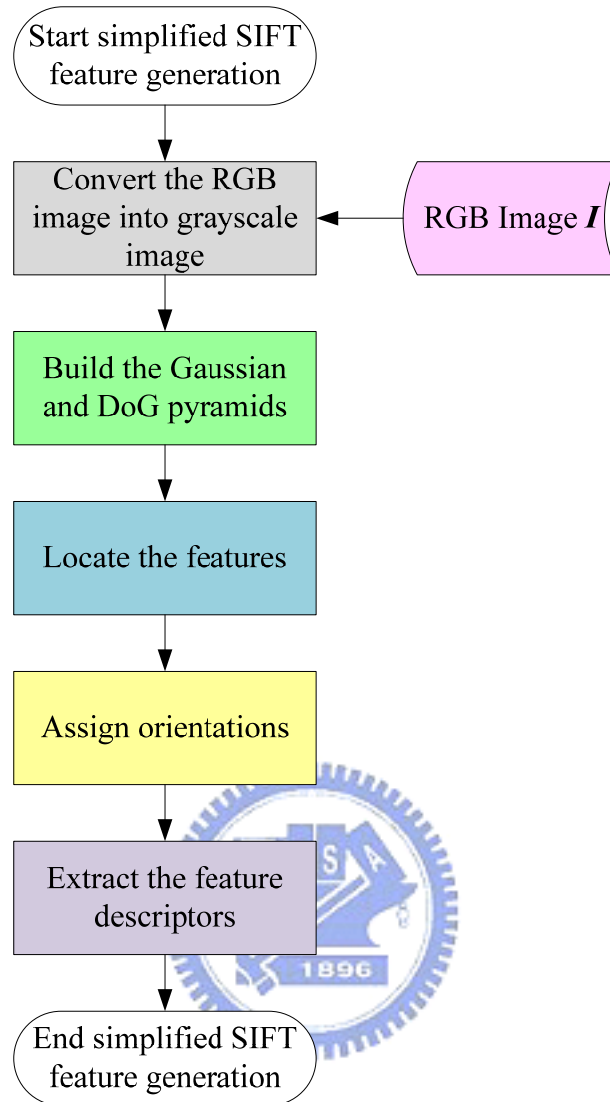


Figure 5.4 A flowchart of the simplified SIFT generation.

**Algorithm 5.1.** *Process of the simplified SIFT feature generation.*

*Input:* A color image  $I$ , a number  $o$  of octaves, a number  $s$  of intervals, a contrast threshold  $c$ , and a curvature threshold  $r$ .

*Output:* The simplified SIFT feature set of the image  $I$ .

*Steps:*

Step 1. Convert the color image  $I$  into a grayscale image  $Y_I$  using the following transformation equation:

$$Y_I = \begin{bmatrix} -0.299 & 0.587 & 0.114 \end{bmatrix} \times \begin{bmatrix} R_I \\ G_I \\ B_I \end{bmatrix}. \quad (5.6)$$

Step 2. Blur the image  $Y_I$  with  $\sigma = 0.5$  and upsample the image  $Y_I$  by a factor of 2 using linear interpolation.

Step 3. Generate the Gaussian and DoG pyramid images using Steps 3.1 through 3.2 in the following.

Step 3.1. Generate the first image of the first octave of the Gaussian pyramid images by blurring the image  $Y_I$  using a Gaussian function with the initial value of  $\sigma_1$  being 1.6.

Step 3.2. For each octave, save the initial  $\sigma$  and perform Steps 3.2.1 through 3.2.2 in the following.

Step 3.2.1. For each interval, compute  $\sigma_{f,i}$  needed to produce the next level of the Gaussian pyramid images, from  $\sigma_i$  using the following equations:

$$\sigma_{f,i} = \sqrt{2^{\frac{2}{s}} - 1} \times \sigma_i, \quad (5.7)$$

where  $s$  is the number of intervals which span the octave, and then compute the convolution of the Gaussian function  $G(\sigma_{f,i})$  and the image  $Y_I$ , as  $L_{i+1}$ .

Step 3.2.2. Subtract the adjacent Gaussian images  $L_{i+1}$  and  $L_i$  to produce the DoG image  $D_i$ .

Step 4. Scan the DoG pyramid to find keypoints using Step 4.1 through 4.4 in the following.

Step 4.1. For each sample point in the DoG pyramid images, compare it with its 8 neighbors in the same scale image, and the 9 corresponding neighbors in neighboring scale images.

Step 4.2. If the sample point is a local maximum or minimum, compute the contrast value  $|D(x, y)|$ , where  $D(x, y)$  is the corresponding DoG image, and test if the value is greater than a contrast threshold  $c$ .

Step 4.3. If the sample point is above the contrast threshold, check if the ratio of the principal curvatures is below the curvature threshold  $r$ , by the following equation:

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} < \frac{(r+1)^2}{r}, \quad (5.8)$$

where  $\mathbf{H}$  is a  $2 \times 2$  Hessian matrix.

Step 4.4. If the sample point's ratio of the principal curvatures is below the curvature threshold  $r$ , add the sample point into the keypoint list.

Step 5. Compute the gradient orientation and magnitude of the Gaussian pyramid images, as the gradient orientation pyramid and the gradient magnitude pyramid, using Equations (5.4) and (5.5).

Step 6. Assign orientations to the keypoints using Steps 6.1 through 6.4 in the following.

Step 6.1. Set up an orientation histogram including 36 bins.

Step 6.2. For each keypoint, create a Gaussian weighting mask with  $\sigma_{wo} = 1.5 \times \sigma_i$ , where  $i$  is the level where the keypoint at, in the Gaussian pyramid, Also, accumulate the gradient orientation weighted by the gradient magnitude and the Gaussian weighting mask into the orientation histogram.

Step 6.3. Find the largest peak in the orientation histogram.



Step 6.4. Iterate over all peaks within 80% of the largest peak and copy this keypoint as a new keypoint with its corresponding orientation into the keypoints list.

Step 7. Extract the feature descriptors for the keypoints using Steps 7.1 through 7.2 in the following.

Step 7.1. Set up an orientation histogram including 8 bins.

Step 7.2. For each keypoint:

Step 7.2.1. create a  $4 \times 4$  array of  $4 \times 4$  sample cells as the sampling grid for the orientation histogram;

Step 7.2.2. rotate the sampling grid according to the orientation assigned in Step 6, for this keypoint;

Step 7.2.3. create a Gaussian weighting mask with  $\sigma_{wd} = 8$  (1/2 times of the sampling grid's width);

Step 7.2.4. accumulate each sample's orientation weighted by the gradient magnitude and the Gaussian weighting mask into the neighboring bins of the orientation histogram;

Step 7.2.5. add the orientation histogram bins as the descriptor into the descriptor list.

Step 8. Store the descriptors as a simplified SIFT feature set.

## 5.4 Proposed Matching Technique Using Simplified SIFT Features

### 5.4.1 Concept of proposed matching algorithm

For indoor security monitoring, the vehicle will navigate to the front of each monitored object. Then, the image taken by the camera is transformed into a set of SIFT features. In order to achieve the security monitoring mission, an efficient matching technique to match the feature set extracted from the current image with the one learned in the learning phase, is indispensable. Hence, the problem is that, given two sets of features, how we can match them efficiently.

According to the paper proposed by Lowe [8, 16], we adopted a matching algorithm by the Hough transform. For the given feature set, the best candidate match for each feature is firstly found by identifying its nearest neighbor in the other feature set. The nearest neighbor is defined as the feature with the closest Euclidean distance for the feature descriptor described in Section 5.2.2. After discarding the outliers, the Hough transform is used to identify the best subsets of matches. Let the given feature set which is found in the navigation phase be denoted as  $F_{navi}$  and the one which is learned in the learning phase be denoted as  $F_{learn}$ . Each SIFT feature specifies 4 parameters: feature locations in the image, scales, and orientations. By applying the affine transform model, as shown in the following equation:

$$\begin{bmatrix} x & -y & 1 & 0 \\ y & x & 0 & 1 \\ & & \dots & \end{bmatrix} \times \begin{bmatrix} m \\ n \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \\ \dots \end{bmatrix}, \quad (5.9)$$

where  $m = s \cos\theta$ ,  $n = s \sin\theta$ , and  $(x, y)$  and  $(u, v)$  specify the locations of  $F_{navi}$  and

$F_{learn}$ , respectively, the unknown similarity transform parameters between each match pair are collected as  $t_x$ ,  $t_y$ ,  $s$ , and  $\theta$  by the following equations:

$$\theta = \tan^{-1}\left(\frac{n}{m}\right) \quad \text{and} \quad s = \frac{m}{\cos \theta}. \quad (5.10)$$

A Hough transform entry is then created to predict the model location, orientation, and scale from the match hypothesis, and each feature votes for all poses that are consistent with the feature. Then, a peak cluster found in the Hough space is regarded to specify the best subsets of matches.

## 5.4.2 Detailed algorithm

In this section, we describe the detailed matching algorithm. The input of the algorithm are a SIFT feature set  $F_{navi} = \{F_{navi,0}, F_{navi,1}, F_{navi,2}, \dots, F_{navi,n}\}$  found in the navigation phase and a learned SIFT feature set  $F_{learn} = \{F_{learn,0}, F_{learn,1}, F_{learn,2}, \dots, F_{learn,n}\}$ . Each feature  $F_i(x, y, s, o, \Phi)$  consists of 5 parameters where  $x$  and  $y$  specify the feature locations, and  $s$ ,  $o$ , and  $\Phi$  are the scale, the orientation, and a set of feature descriptors of the feature described in Section 5.2.2, respectively. While finding the nearest neighbors, a threshold  $r_{dis}$  for the maximum ratio between the distances of the closest and the second closest neighbors for a match to be allowed, is needed. And for the Hough transform, it also needs a threshold  $r_{ht}$  for the minimum number of matches before a model is selected. The detailed process is described in the following.

**Algorithm 5.2.** *Process of the matching of the SIFT features.*

*Input:* Two SIFT feature sets  $F_{navi} = \{F_{navi,0}, F_{navi,1}, F_{navi,2}, \dots, F_{navi,n}\}$  with each being of the form  $F_{navi}(x_{navi}, y_{navi}, s_{navi}, o_{navi}, \Phi_{navi})$  and  $F_{learn} = \{F_{learn,0}, F_{learn,1}, F_{learn,2}, \dots,$

$F_{learn,n}$  with each being of the form  $F_{learn}(x_{learn}, y_{learn}, s_{learn}, o_{learn}, \Phi_{learn})$ , a threshold  $r_{ht}$ , and a threshold  $r_{dis}$ .

*Output:* The set of matches or the message ‘false’ if no match is found.

*Steps:*

Step 1. Find the nearest neighbors of  $\Phi_{navi}$  from  $\Phi_{learn}$  using the following Steps.

Step 1.1. Compute the closest Euclidean distance  $D_1$  between each feature descriptor in  $\Phi_{navi}$  and  $\Phi_{learn}$ .

Step 1.2. Compute the second closest Euclidean distance  $D_2$  between each feature descriptor in  $\Phi_{navi}$  and  $\Phi_{learn}$ .

Step 1.3. If the ratio of  $D_1$  and  $D_2$  is smaller than  $r_{dis}$ , add the match into a nearest neighbor list.

Step 2. Construct a 4D Hough space as a Hough histogram with 4 dimensions: two dimensions for translation  $(x, y)$ , one dimension for orientation  $o$ , and one dimension for scale  $s$ .

Step 3. Set up  $21 \times 21$  bins with sizes of 0.25 times the maximum image dimension for the dimensions of translation, 12 bins with sizes of 30 degrees for the orientation, and 17 bins with sizes of a factor of 2 for the scale.

Step 4. For each match, perform the following steps to vote for the same pose of the model.

Step 4.1. Apply the following equations with the  $F_{navi}$  and  $F_{learn}$ :

$$\begin{bmatrix} x_{navi} & -y_{navi} & 1 & 0 \\ y_{navi} & x_{navi} & 0 & 1 \\ x_{navi} + t_{navi} & -(y_{navi} + t_{navi}) & 1 & 0 \\ y_{navi} + t_{navi} & x_{navi} + t_{navi} & 0 & 1 \end{bmatrix} \times \begin{bmatrix} m \\ n \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_{learn} \\ y_{learn} \\ x_{learn} + t_{learn} \\ y_{learn} + t_{learn} \end{bmatrix}, \quad (5.11)$$

where  $t_{navi}$  and  $t_{learn}$  are a slight shift in the direction of each orientation of  $F_{navi}$  and  $F_{learn}$ , respectively.

Step 4.2. Compute the  $s$  and  $\theta$  using Equation (5.10) for this match.

Step 4.3. Accumulate the translation, scale, and rotation weighted by the distances between  $(t_x, t_y)$  and the nearest translation bins, the difference angle between  $\theta$  and the nearest orientation bins, and the difference scale between  $s$  and the nearest scale bins, into the nearest  $2 \times 2 \times 2 \times 2 = 16$  bins, and add the match into the corresponding bins respectively.

Step 5. Find the peak of the Hough histogram as the set of matches if the number of matches of the peak is greater than  $r_{ht}$ ; otherwise, return false.

## 5.5 Experimental Results

In the section, some experimental results are shown in the following. In Figure 5.5, we firstly control the vehicle to learn a valuable painting, as shown in 5.5(a), as an example of the concerned objects. The features located by the simplified SIFT algorithm are shown in 5.5(b). Then, we start the security patrolling by the vehicle to monitor this painting with three cases: the original painting exists, the painting is missing, and the painting is replaced by another painting. In the first case, a successful matching is performed as shown in (c). In the second and the third cases, the matching fails as shown in (d) and (e).

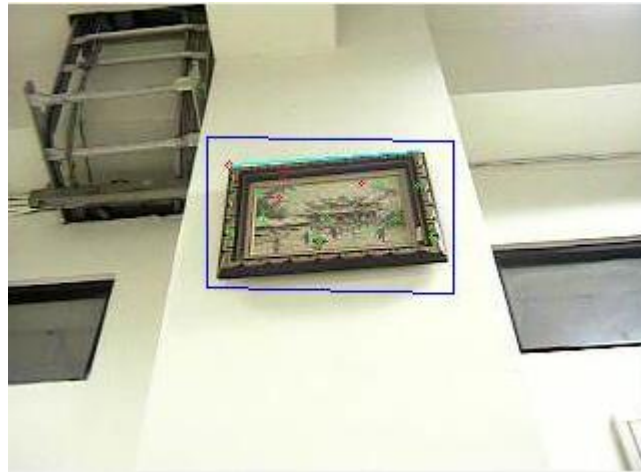
For the necessity of simplification discussed in the previous section, some experimental results are shown in the following. In Figure 5.6 and Figure 5.7, we compare the original SIFT algorithm and the simplified SIFT algorithm. The experimental results are shown that the influences of reducing the number of octaves on the matching results are insignificant.



(a)



(b)



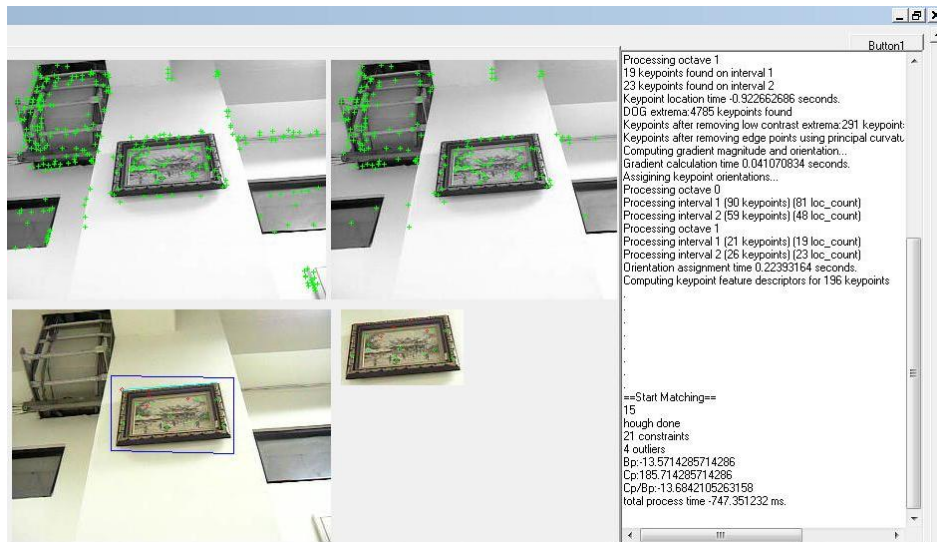
(c)



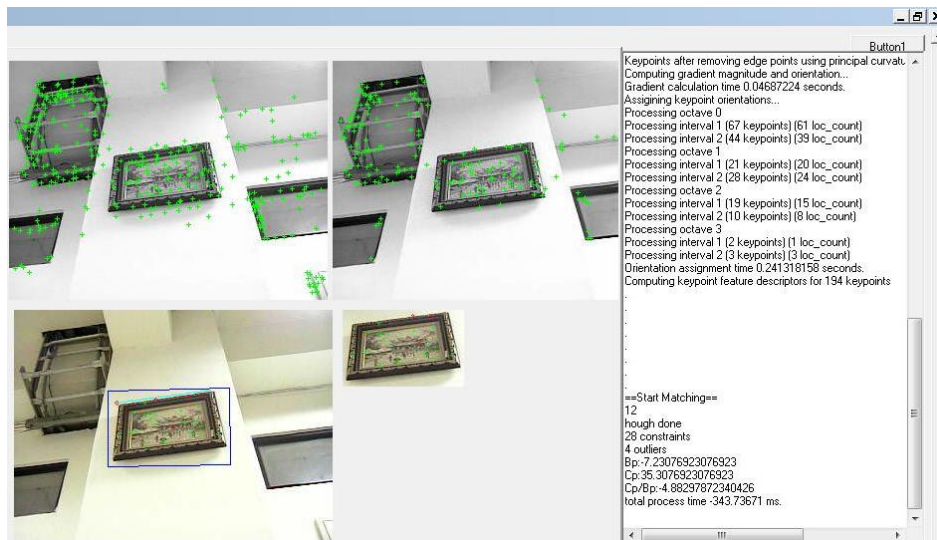
(d)

(e)

Figure 5.5 The experimental results of the monitored object matching process. (a) is the monitored object learned in the learning phase. The location of features are marked as green crosses in (b). (c) is the successful matched result which is specified by the blue region. (d) and (e) are the matching results which fail to match with the learned object.



(a)



(b)

Figure 5.6 The experimental results of the monitored object matching process with the comparison between the successful matching results by the use of the simplified SIFT algorithm and the original one. (a) is the matching result conducted by the use of simplified SIFT algorithm with reducing the number of octaves and (b) is the one conducted by the use of original SIFT algorithm.



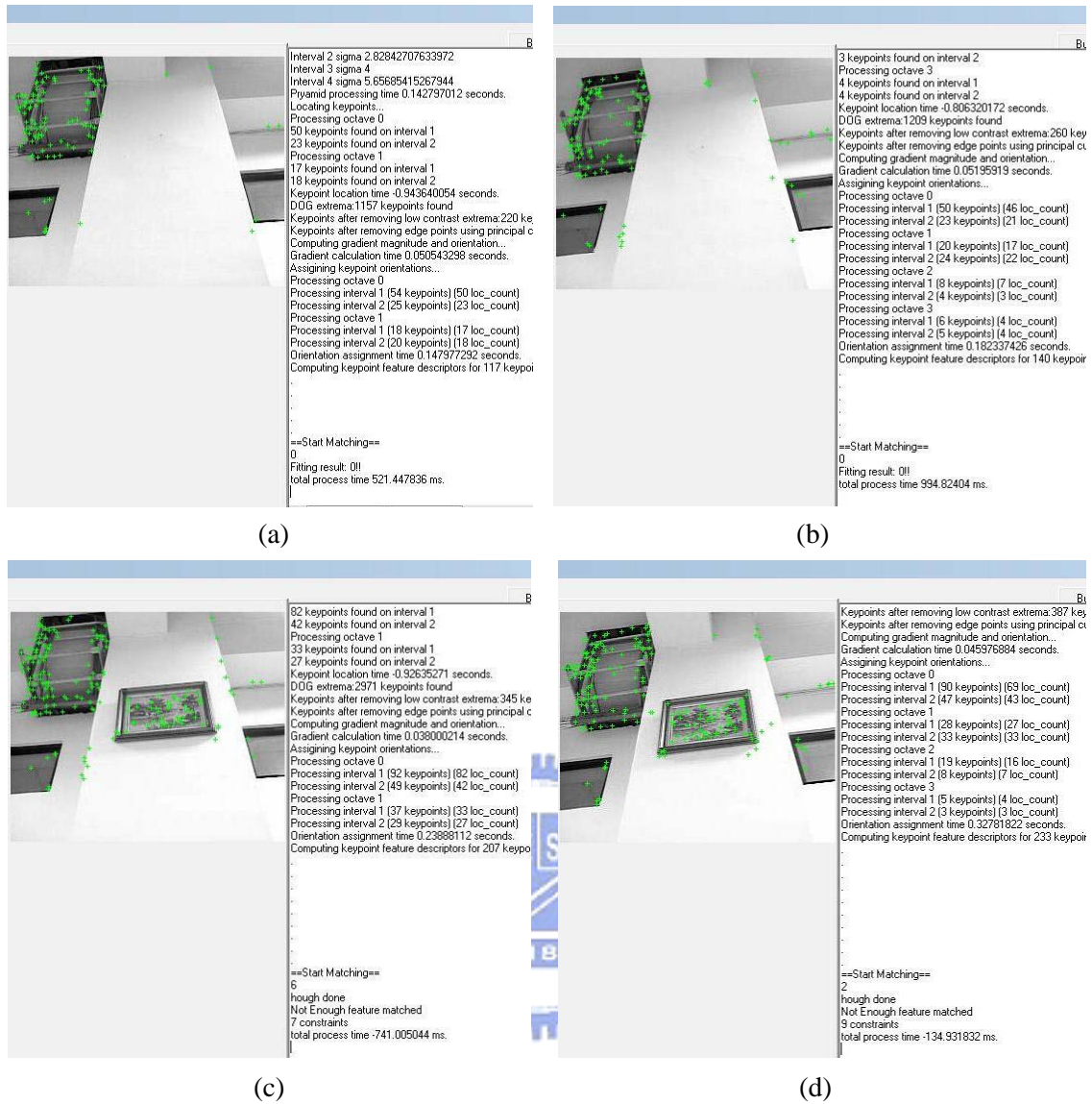


Figure 5.7 The experimental results of the monitored object matching process with the comparison between the failed matching results by the use of simplified SIFT and original SIFT algorithm. (a) and (c) are matching result conducted by the use of simplified SIFT algorithm with reducing the number of octaves, and (b) and (d) are the one conducted by the use of original SIFT algorithm.



# Chapter 6

## Vehicle Guidance by Location Estimation Based on 2D Object Image Matching Results

### 6.1 Introduction

For vehicle navigation in indoor environments, the vehicle location is the most important information to guide the vehicle in track. Though the location information including the position and the direction, which are provided by the odometer of the vehicle, are precise enough for most applications, it cannot be used solely for the navigation process because the incremental mechanical errors might result in imprecise odometer data and so the deviation of the navigation path. Hence, in order to keep the navigation in track, vision-based vehicle location estimation is helpful to eliminate the errors.

In this study, we focus on the use of the artificial landmarks or specific scene features for vehicle localization. A number of methods are reviewed in Section 6.2. Among these methods, a method proposed by Chou and Tsai [12] utilized a house corner to estimate the vehicle location. The proposed vehicle location estimation method in this study is based on the use of the monitored-object matching result and a simplified version derived from Chou and Tsai. The idea and the detailed process of the method are described in Section 6.3.

With successively estimated vehicle locations, we can correct the navigation path

by comparing the estimated location and the learned one, and transform the adjustment into the global coordinate system. The detailed process of path correction is described in Section 6.4.

## 6.2 Review of Vehicle Location Estimation Techniques

In a known indoor environment, an ordinary approach to vehicle location estimation is to use some special landmarks and to analyze monocular images captured by a camera. A the standard landmark proposed by Fukui [9], as shown in Figure 4.7 (a), is a diamond shape whose boundary consists of four identical thick line segments all with a known length. The boundary of the diamond images taken by the camera is firstly extracted, and the lengths of the two diagonals are computed. Then, by the use of the two diagonals in the image, the location of the camera is derived. Another landmark proposed by Magee and Aggarwal [10] for vehicle location estimation is a sphere on which two perpendicular great circles are drawn, as shown in Figure 4.7 (b). According to the size of the projected circle in the image of the sphere, the distance from the camera to the sphere center and the direction of the camera are computed. Then, the vehicle location is computed accordingly in terms of the sphere coordinates. Huang et al. [11] also proposed a colored rectangle signboard, as shown in Figure 4.7 (c) which is placed in a known position, as an artificial landmark. The signboard area is firstly extracted and the vertex points of the signboard are detected. Then, by computing the vanishing points in the image of the signboard, the position of the signboard is identified and therefore the vehicle position is identified reversely.

In the previous works of our laboratory, Chou and Tsai [12] proposed a set of house corner edges which exists naturally in a house as a landmark. Such a landmark is visible from a house floor, and show as an identical geometric structure of a “Y” shape in the image like Figure 4.7 (d). The projections of the three edges on the image plane are then extracted. Then, under a reasonable assumption that the distance from the camera to the ceiling is known, the vehicle location is estimated by the use of these edges. Chiang and Tsai [13] also proposed a method using a simplified version derived from Chou and Tsai’s formula and the image taken by the PTZ camera. Two edges are firstly extracted by line detection, as shown in Figure 4.7 (e), and then the vehicle location is estimated by applying the simplified formula.

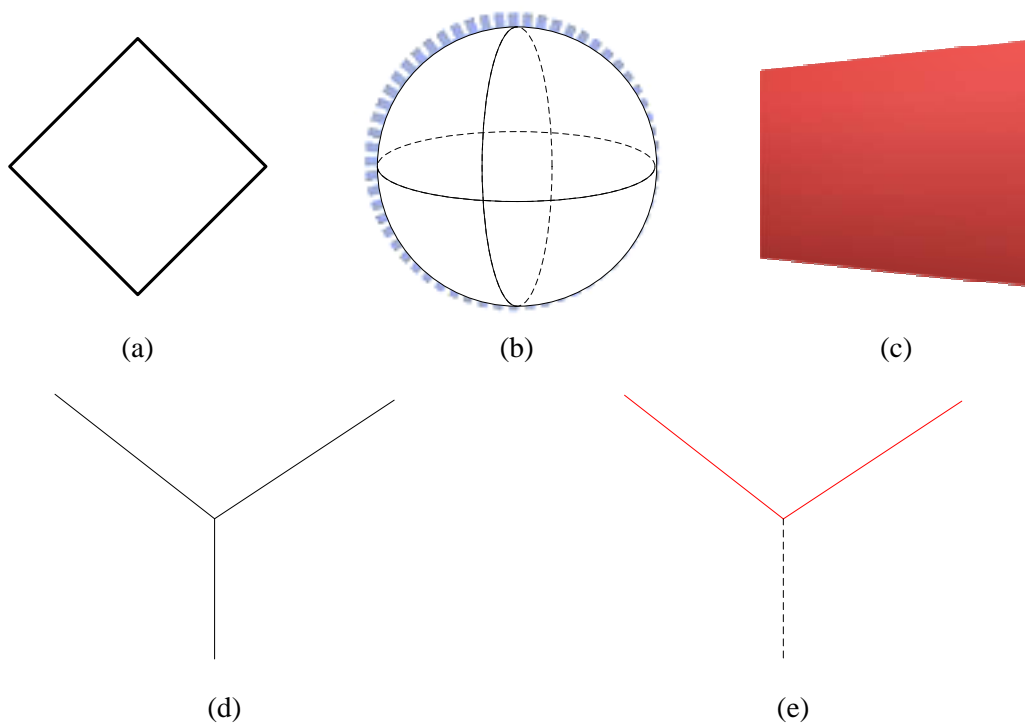


Figure 6.1 Some landmarks used in previous approaches. (a) A diamond-shaped standard mark used in [9]. (b) A sphere used for robot location in [10]. (c) The perspective projection of a colored rectangle signboard used in [11]. (d) The perspective projection of a house corner used in [12]. (e) The two red edges of the perspective projection of a house corner used in [13].

## 6.3 Vehicle Location Estimation by Object Image Matching Results

In the previous sections, we have described the learning of a given horizontal line and path correction by the given one found in the navigation phase. We use only one edge for the horizontal line, and the corner point for the start point of the horizontal line, instead of the three edges of a house corner to estimate the vehicle location. The linear equation's coefficients and the start point are computed in the learning and the navigation phases respectively, as described in Section 3.3.4 and 4.2.3.

In order to describe the proposed method conveniently, we introduce the coordinate systems used in Section 6.3.1. The idea of the proposed method is to use the coefficients of the equation of the edge and the location of the corner point in the image coordinate system to estimate the vehicle location. The detailed idea is described in Section 6.3.2, and the detailed process of the proposed vehicle location estimation is described in Section 6.3.3.

### 6.3.1 Coordinate systems

Here, we use three kinds of the coordinate systems: the reference coordinate system, the camera coordinate system, and the image coordinate system. With these coordinate systems, it will be clear and convenient to describe a vehicle location. The definitions of the three coordinate systems are described in the following.

- (1) The reference coordinate system (RCS, denoted as  $X$ - $Y$ - $Z$ ): In the reference coordinate system, we assume the  $X$ - $Y$  plane is parallel to the floor where the vehicle navigates on, and the  $Z$ -axis is perpendicular to the  $X$ - $Y$  plane.

The corner point is the origin  $R_o$  of the reference coordinate system.

- (2) The camera coordinate system (CCS, denoted as  $U$ - $V$ - $W$ ): In the camera coordinate system, we also establish the  $U$ -,  $V$ -, and  $W$ -axes. The  $U$ - $V$  plane is parallel to the image plane, and the  $U$ -axis is parallel to the  $X$ - $Y$  plane of the reference coordinate system. The origin  $C_o$  is located at the camera lens center and the  $W$ -axis is aligned to be parallel to the camera optical axis.
- (3) The image coordinate system (ICS, denoted as  $u_p$ - $v_p$ ): The image plane is located at  $W = f$ , where  $f$  is the focus length of the camera. The  $u_p$ - $v_p$  plane is coincident with the image plane of the image coordinate system and the origin  $I_o$  is the center of the image plane.

The relations among the three coordinate systems are illustrated in Figure 6.2.

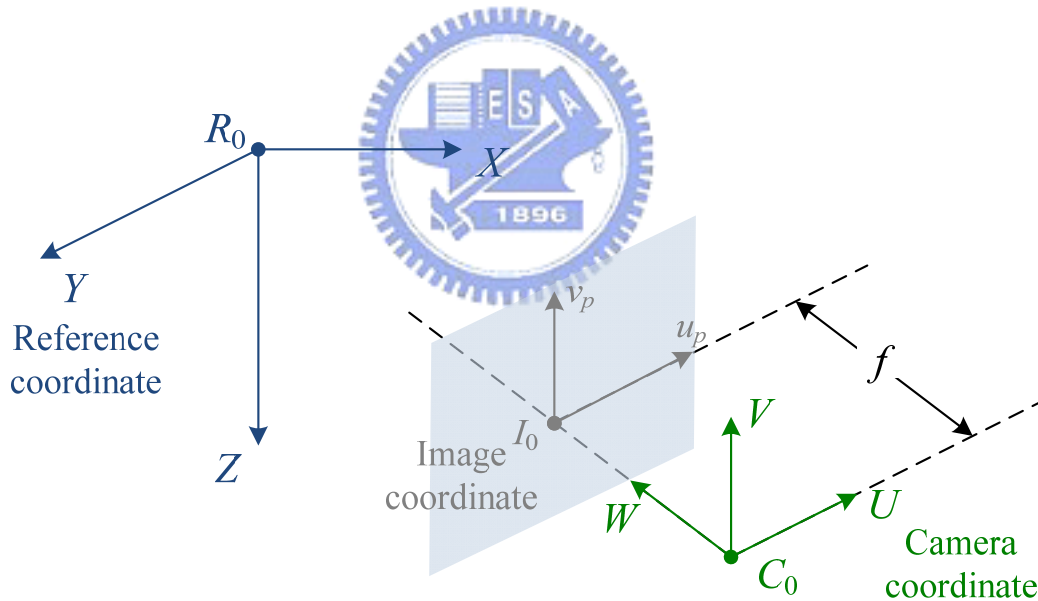


Figure 6.2 The relations of the reference coordinate system, the camera coordinate system, and the image coordinate system.

### 6.3.2 Idea of proposed method

As described in Section 3.3.4, a horizontal line is given in the learning phase to specify  $X$ -axis of the RCS. A start point of the given horizontal line also specifies the

origin  $R_0$  of the RCS. Because the  $X$ - $Y$  plane is parallel to the floor, we can treat the RCS as a virtual house corner. The  $X$ - and  $Y$ -axes specify the two perpendicular lines on the ceiling of the virtual house corner, as shown in the left-top of Figure 6.3, and the  $Z$ -axis specifies the virtual line of the virtual house corner.

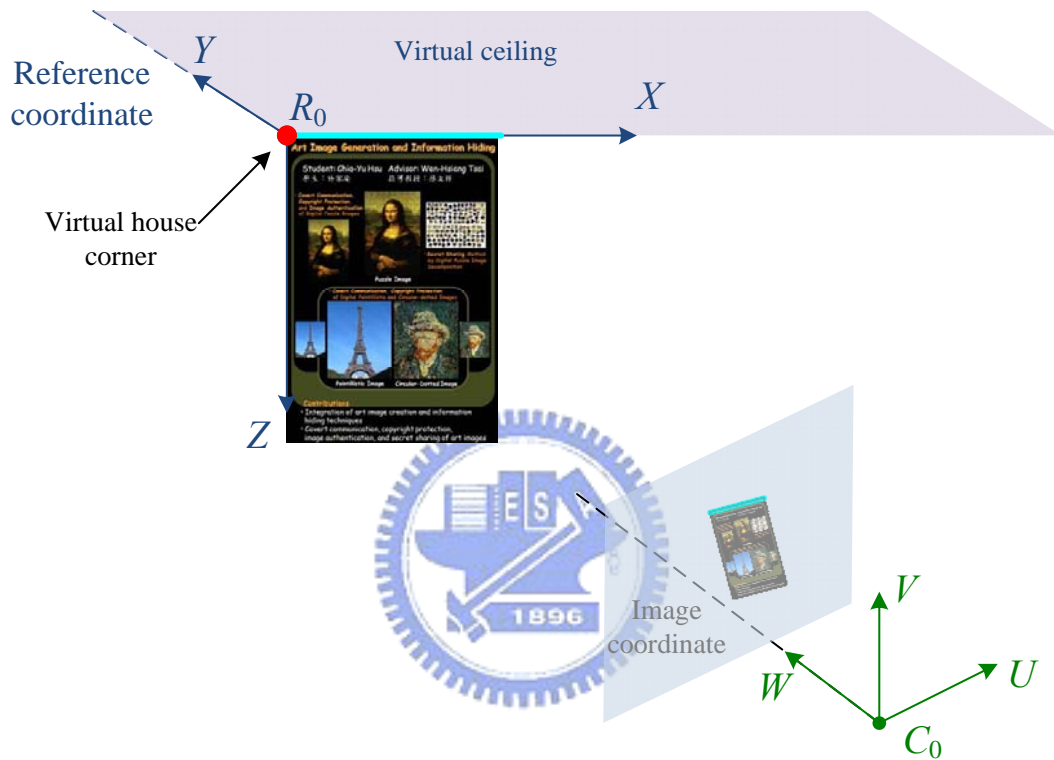


Figure 6.3 A diagram of the virtual house corner which is specified by the given horizontal line (the cyan line on the top of the poster), and the start point (the red point on the left-top of the poster).

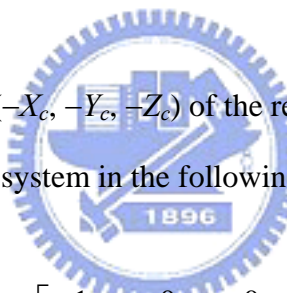
The equations of the edge line through the corner point in terms of image coordinates  $(u, v)$  are described by  $u_p + bv_p + c = 0$ . The desired vehicle location will be described by three position parameters  $X_c$ ,  $Y_c$ , and  $Z_c$  and two direction parameters  $\theta$  and  $\psi$ , where  $Z_c$  is the distance from the camera to the ceiling and is assumed to be known;  $\theta$  is the pan angle between the optical direction of the camera and the  $Y$ -axis of the RCS; and  $\psi$  is the tilt angle of the optical direction of the camera with respect

to the RCS and is also assumed to be known by solving the equation  $\psi = 90^\circ - \phi$ , where  $\phi$  is the tilt angle provided by the PTZ camera. The five vehicle location parameters can be derived in terms of the two coefficients  $b$  and  $c$  of the edge line equation and the start point  $(u_1, v_1)$  in the image taken by the camera. Finally the vehicle location could be estimated by the computation of these parameters.

### 6.3.3 Detailed process of location estimation

In this section, we derive the relation between the reference coordinates and the coefficients of the edge line equation in the image coordinate system. At first, we transform the reference coordinates into the camera coordinate. The transformation consists of four steps.

Step 1. Translate the origin  $(-X_c, -Y_c, -Z_c)$  of the reference coordinate system to the origin of the camera system in the following way:



$$T(X_c, Y_c, Z_c) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -X_c & -Y_c & -Z_c & 1 \end{bmatrix} \quad (6.1)$$

Step 2. Rotate the  $X$ - $Y$  plane about the  $Z$ -axis through the pan angle  $\theta$  such that the  $X$ - $Y$  plane is parallel to the  $U$ - $V$  plane using the following equation:

$$R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.2)$$

Step 3. Rotate the  $Y$ - $Z$  plane about the  $X$ -axis through the tilt angle  $\psi$  such that the

$X$ - $Y$  plane is parallel to the  $U$ - $V$  plane using the following equation:

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \psi & -\sin \psi & 0 \\ 0 & \sin \psi & \cos \psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (6.3)$$

Step 4. Reverse the  $Z$ -axis such that the positive direction of the  $Z$ -axis is identical to the negative direction of the  $W$ -axis using the following equation:

$$F_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (6.4)$$

Let  $P$  be any point in the 3-D space with reference coordinates  $(x, y, z)$  and camera coordinates  $(u, v, w)$ . Then the above coordinate transformation can be described as follows:

$$\begin{aligned} (u, v, w, 1) &= (x, y, z, 1) \cdot T(X_c, Y_c, Z_c) \cdot R_z(\theta) \cdot R_x(\psi) \cdot F_z \\ &= (x, y, z, 1) T_r \end{aligned} \quad (6.5)$$

where

$$\begin{aligned} T_r &= T(X_c, Y_c, Z_c) \cdot R_z(\theta) \cdot R_x(\psi) \cdot F_z \\ &= \begin{bmatrix} \cos \theta & -\sin \theta \cos \psi & -\sin \theta \sin \psi & 0 \\ \sin \theta & \cos \theta \cos \psi & \cos \theta \sin \psi & 0 \\ 0 & \sin \psi & -\cos \psi & 0 \\ x_0 & y_0 & z_0 & 1 \end{bmatrix} \end{aligned} \quad (6.6)$$

with



$$\begin{aligned}
x_0 &= -X_c \cos \theta - Y_c \sin \theta, \\
y_0 &= (X_c \sin \theta - Y_c \cos \theta) \cos \psi - Z_c \sin \psi, \\
z_0 &= (X_c \sin \theta - Y_c \cos \theta) \sin \psi + Z_c \cos \psi.
\end{aligned} \tag{6.7}$$

Let  $P$  be any point on the  $X$ -axis with reference coordinates  $(x, 0, 0)$  and the camera coordinates  $(u_x, v_x, w_x)$ . And let  $(u_p, v_p)$  be the image coordinates of the projection of  $P$ . Then, according to the triangulation principle, we have the following two equations:

$$u_p = \frac{f \cdot u_x}{w_x}, \tag{6.8}$$

$$v_p = \frac{f \cdot v_x}{w_x},$$

where  $f$  is the camera focus length. And Equation (6.5) can be rewritten as:

$$\begin{aligned}
(u_x, v_x, w_x, 1) &= (x, 0, 0, 1) \cdot T_r \\
&= (x \cos \theta + x_0, -x \sin \theta \cos \psi + y_0, -x \sin \theta \sin \psi + z_0, 1).
\end{aligned} \tag{6.9}$$

Substituting the values of  $u_x$  and  $v_x$  above into the two equalities in Equation (6.8), we get:

$$u_p = \frac{f \cdot (x \cos \theta + x_0)}{-x \sin \theta \sin \psi + z_0}, \tag{6.10}$$

$$v_p = \frac{f \cdot (-x \sin \theta \sin \psi + y_0)}{-x \sin \theta \sin \psi + z_0}. \tag{6.11}$$

Eliminating the variable  $x$ , we can get the equation for the projection of the

X-axis in the image plane in the following:

$$u_p = \frac{v_p \cdot (-z_0 \cos \theta - x_0 \sin \theta \sin \psi) + f \cdot (y_0 \cos \theta + x_0 \sin \theta \cos \psi)}{-y_0 \sin \theta \sin \psi + z_0 \sin \theta \cos \psi}. \quad (6.12)$$

After substituting Equation (6.12) into  $u_p + bv_p + c = 0$  and Equation (6.7) for  $x_0$ ,  $y_0$ , and  $z_0$ , we obtain the following two equalities:

$$b = \frac{Y_c \sin \psi - Z_c \cos \theta \cos \psi}{-Z_c \sin \theta}, \quad (6.13)$$

$$c = \frac{f \cdot (-Y_c \cos \psi - Z_c \cos \theta \sin \psi)}{-Z_c \sin \theta}. \quad (6.14)$$

Then, we can use Equations (6.13) and (6.14) to derive the variable  $\theta$ . Equations (6.13) and (6.14) can be transformed into Equations (6.15) and (6.16), respectively, described in the following:

$$Z_c \cdot (-b \sin \theta + \cos \theta \cos \psi) = Y_c \sin \psi, \quad (6.15)$$

$$Z_c \cdot (-c \sin \theta + f \cos \theta \cos \psi) = -f Y_c \cos \psi. \quad (6.16)$$

By eliminating  $Z_c$  and  $Y_c$  from Equations (6.15) and (6.16), we can get:

$$\tan \theta = \frac{f}{fb \cos \psi + c \sin \psi}. \quad (6.17)$$

Because the value  $\psi$  can obtain from the use of the tilt angle  $\phi$  provided by the PTZ camera and the following equation:

$$\psi = 90^\circ - \phi, \quad (6.18)$$

we can apply Equation (6.17) with  $\psi$  to get the value  $\theta$ , and Equation (6.15) with known values  $\psi$ ,  $\theta$ , and  $Z_c$ , to obtain the value  $Y_c$ .

With successively obtained values of  $\psi$ ,  $\theta$ ,  $Y_c$ , and  $Z_c$ , we can substitute these values and the start point  $(u_1, v_1)$  into Equations (6.12) and (6.7) to obtain the values of  $X_c$ .

## **6.4 Path Correction by Vehicle Location Estimation Results**

After we compute the estimated vehicle location  $(X_c, Y_c)$  and the camera direction angle  $\theta$ , the next step is path correction by the use of the estimated results. Before describing the process of the proposed path correction, some coordinate systems and the definition of the direction angle of the vehicle and the PTZ camera are firstly introduced in Section 6.4.1. Then, the detailed process of the proposed path correction is described in Section 6.4.2.

### **6.4.1 Direction angle of vehicle and coordinates of path nodes**

In this section, another two coordinate systems are utilized to describe the relation among the learned path, the vehicle, and the monitored object. The coordinate systems are shown in Figure 6.4 and the definitions of them are described in the following.

- (1) The global coordinate system (GCS, denoted as  $G_x-G_y$ ): The floor of environments is defined as the  $G_x-G_y$  plane, and the origin  $G_0$  of the global coordinate system is a pre-defined point on the floor. We define  $G_0$  as the starting position of the navigation in this study.
- (2) The vehicle coordinate system (VCS, denoted as  $V_x-V_y$ ): The  $V_x-V_y$  plane is parallel to the  $G_x-G_y$  plane, and the origin  $V_0$  is taken to be the rotation center of the vehicle, which is at the middle of the line segment connecting the two driving wheels. The  $V_y$ -axis is parallel to the line segment of the two driving wheels and through the origin  $V_0$ .

Besides, the direction angle of the vehicle and the PTZ camera are defined for the convenience of describing the coordination transformation and computing the turn angle for the correction, as illustrated in Figure 6.5.

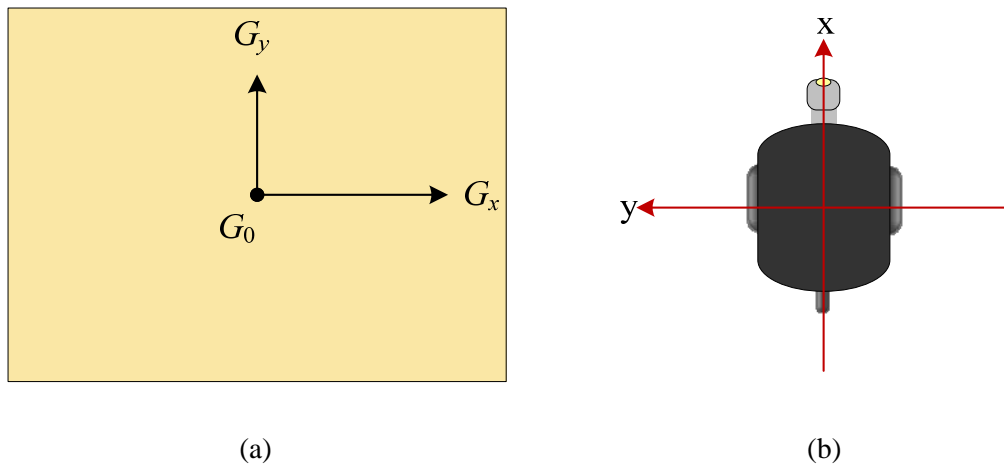


Figure 6.4 Another two coordinate systems used in this study. (a) The global coordinate system. (b) The vehicle coordinate system.

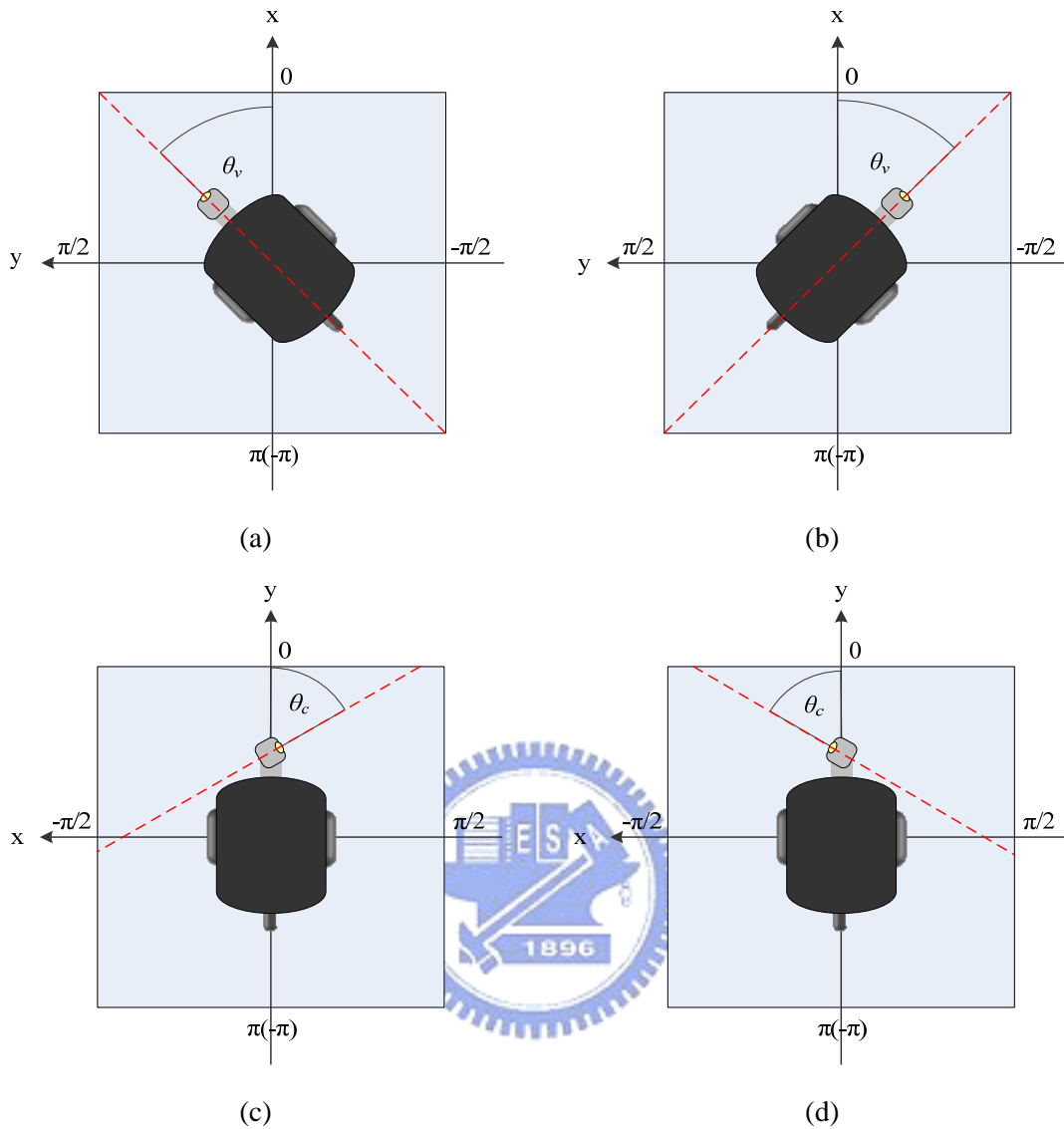


Figure 6.5 The rotate angle  $\theta_v$  of the vehicle and the pan angle  $\theta_c$  of the PTZ camera. (a)

$0 \leq \theta_v \leq \pi$ . (b)  $-\pi \leq \theta_v \leq 0$ . (c)  $0 \leq \theta_c \leq \pi$ . (d)  $-\pi \leq \theta_c \leq 0$ .

## 6.4.2 Method of proposed path correction

In the previous section, we have obtained the estimated location between the vehicle and the origin  $R_0$  of the reference coordinate. The estimated location includes three parameters: the camera location in the RCS,  $(X_c, Y_c)$ , and the direction angle of the camera,  $\theta$ . The relation among the vehicle, the camera, and the RCS is illustrated in Figure 6.6.

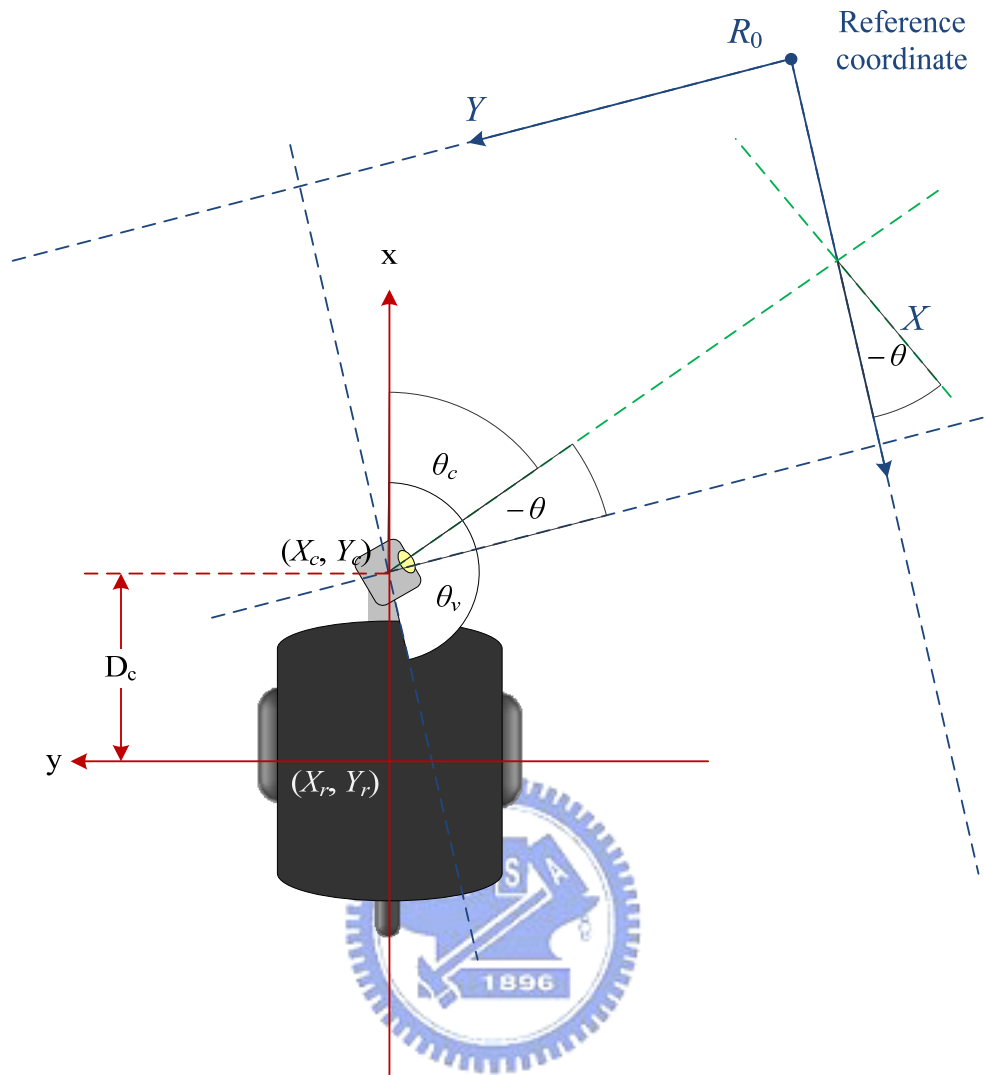


Figure 6.6 The relation among the vehicle, the camera, and the reference coordinate system.

In Figure 6.6, the direction angle of the vehicle can be derived by substituting the  $\theta$  into the following equation:

$$\theta_v = 90^\circ - (-\theta) + \theta_c, \quad (6.19)$$

where the angle  $\theta$  is negative because the angle of the clockwise rotation is positive and the  $X$ - $Y$  plane is rotated through a pan angle  $-\theta$  to be parallel to the image plane, as described in Section 6.3.3, and  $\theta_c$  is the pan angle of the PTZ camera. As soon as the direction angle  $\theta_v$  of the vehicle is obtained, we can compute the vehicle location in the RCS by substituting the angle  $\theta_v$  and the distance between the camera and the

center of the vehicle into the following equations:

$$\begin{aligned} X_v &= X_c - D_c \cdot \cos \theta_v, \\ Y_v &= Y_c + D_c \cdot \sin \theta_v. \end{aligned} \tag{6.20}$$

Finally, the location  $(X_v, Y_v)$  and the direction angle  $\theta_v$  of the vehicle are acquired. If the vehicle is in the learning phase, these parameters are saved as the calibration information data, as described in Section 3.3.4. If the vehicle is in the navigation phase, we can utilize the parameters obtained above and the learned ones to correct the navigation path.

Let the learned location parameters including the location and the direction angle of the vehicle be denoted as  $L(X_l, Y_l, \theta_l)$  in the RCS, and the estimated ones as  $V(X_v, Y_v, \theta_v)$ . Utilizing these parameters above and the corresponding learned path node  $(L_x, L_y)$  and the direction angle  $\Theta_l$  of the vehicle at this path node in the GCS, we can compute the corrected location  $(N_x, N_y)$  and the adjustment angle  $\theta_{adj}$  of the vehicle by transforming the relative location between  $(X_v, Y_v)$  and  $(X_l, Y_l)$  in the RCS into the GCS and computing the adjusting angle between  $\theta_v$  and  $\theta_l$ . The relation among the RCS, the VCS, and the GCS, and the corresponding angle is illustrated in Figure 6.7 and the detailed transformation is described as an algorithm in the following.

**Algorithm 6.1.** *Computation of the corrected location and the corrected direction angle of the vehicle.*

*Input:* The estimated location parameters  $V(X_v, Y_v, \theta_v)$ , the learned location parameters  $L(X_l, Y_l, \theta_l)$ , the learned path node  $(L_x, L_y)$ , and the direction angle  $\Theta_l$ .

*Output:* The corrected location  $(N_x, N_y)$  and the adjustment angle  $\theta_{adj}$ .

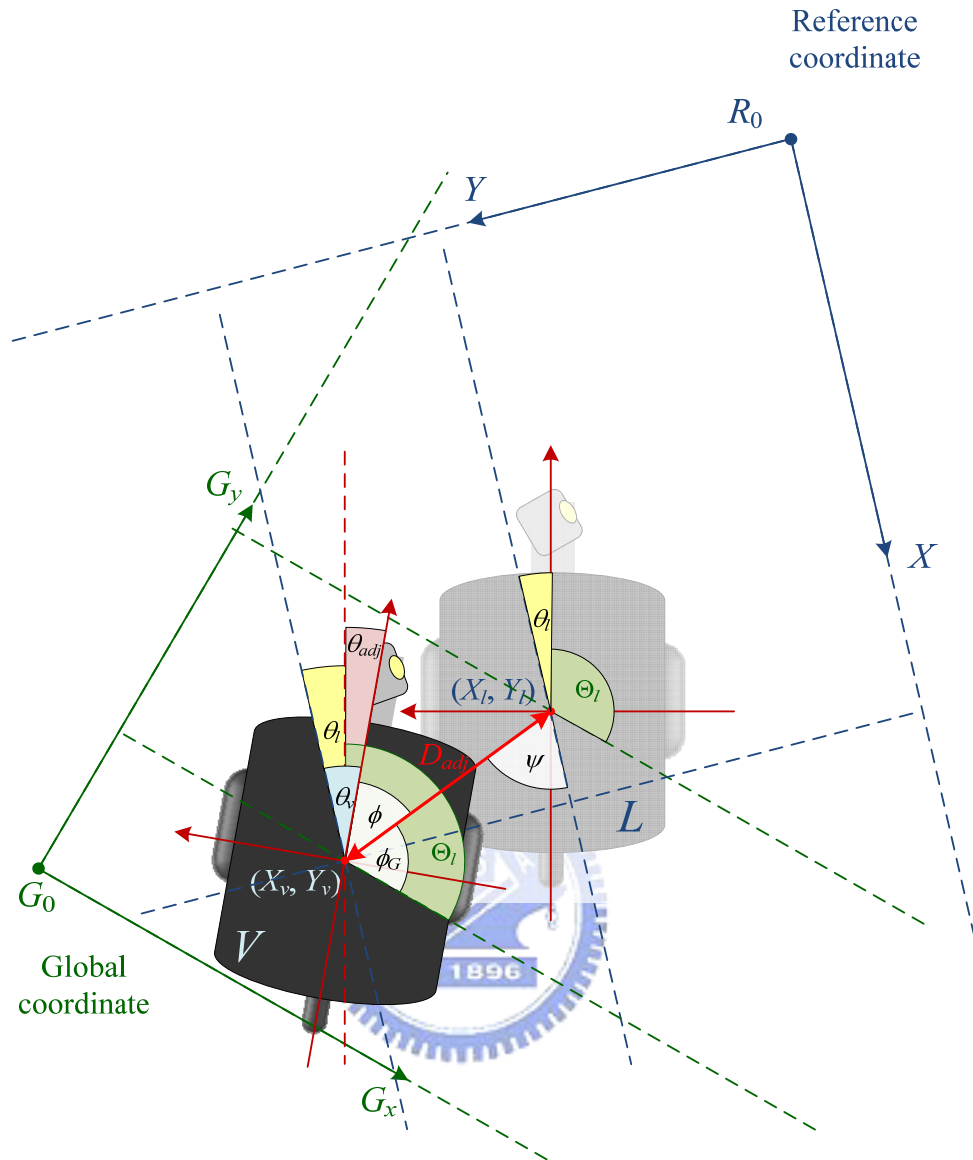


Figure 6.7 The relation among the RCS, the VCS, and the GCS, and the corresponding angle the vehicle. The learned location of the vehicle is denoted as a pastel vehicle with the location  $(X_l, Y_l)$  in the RCS, and the current location of the vehicle is denoted as the colored vehicle with the location  $(X_v, Y_v)$  in the RCS.

*Steps:*

- Step 1. Compute the adjustment angle  $\theta_{adj}$  between the direction angles of  $V$  and  $L$  using the following equation:



$$\theta_{adj} = \theta_v - \theta_l. \quad (6.21)$$

Step 2. Compute the angle  $\phi$  using the following equation:

$$\phi = \psi - \theta_v, \text{ where } \psi = \tan^{-1}\left(\frac{Y_v - Y_l}{X_v - X_l}\right). \quad (6.22)$$

Step 3. Compute the angle  $\phi_G$  of the line segment from the vehicle location  $L$  to  $V$  in the GCS using the following equation:

$$\phi_G = \Theta_l - \phi - \theta_{adj}. \quad (6.23)$$

Step 4. Compute the adjustment location with respect to the learned node using the following equations:

$$\begin{aligned} X_{adj} &= D_{adj} \cdot \cos(\phi_G), \\ Y_{adj} &= D_{adj} \cdot \sin(\phi_G), \end{aligned} \quad (6.24)$$

where

$$D_{adj} = \sqrt{(X_v - X_l)^2 + (Y_v - Y_l)^2}. \quad (6.25)$$

Step 5. Compute the corrected location  $(N_x, N_y)$  using the following equations:

$$\begin{aligned} N_x &= L_x - X_{adj}, \\ N_y &= L_y - Y_{adj}. \end{aligned} \quad (6.26)$$

## 6.5 Experimental Results

In order to conduct experiments about the ability of path correction, we set up a navigation path including a monitoring node and a path node, as shown in Figure 6.8.

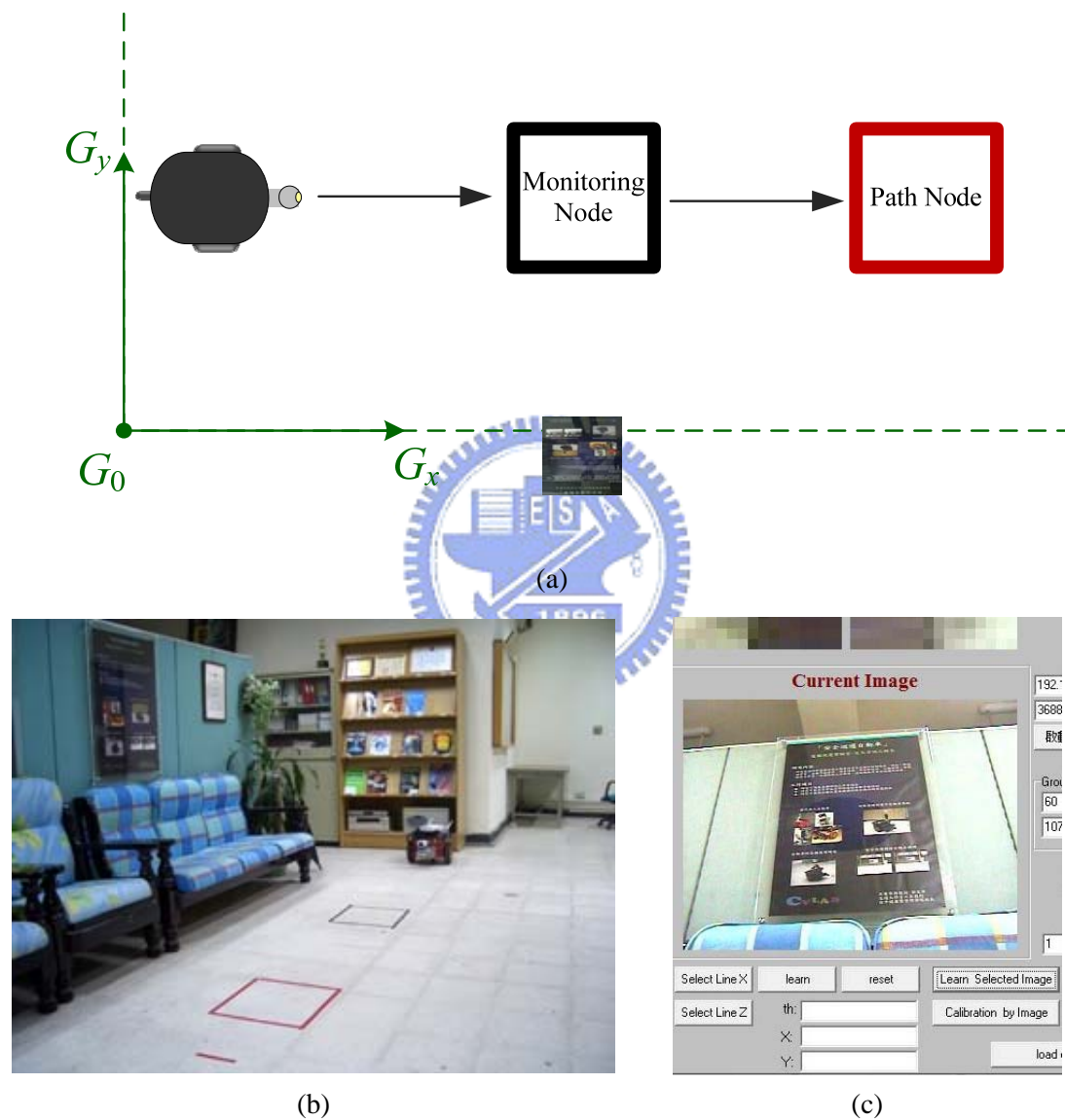


Figure 6.8 An experimental navigation path including a monitoring node and a path node. (a)

A diagram of the navigation path. (b) A photo of the navigation path. (c) The user interface of learning the monitored object and the location estimation data.

With the successively learned navigation path, we firstly put the vehicle at an identical start position to start the navigation. The experimental result shows that the vehicle navigated correctly on the navigation path, as shown in Figure 6.9.



(a)

(b)



(c)

Figure 6.9 The experimental result of path correction by the navigation starting from the original learned start position. (a) The vehicle arrived at the monitoring node and performed the matching and path correction. (b) The vehicle navigated to the next path node after correcting the navigation path. (c) The vehicle successfully matched the monitored object and estimated the location.



(a)



(b)



(c)



(d)



(e)

Figure 6.10 The experimental result of path correction by the navigation starting from a different start position. (a) The vehicle started the navigation at a position different to the start position in the learning phase. (b) The vehicle arrived to a wrong place where should be the monitoring node, and performed the matching and path correction. (c) The vehicle still successfully matched the monitored object and estimated the location. (d)-(e) The vehicle navigated to next path node after correcting the navigation path.

We also tested another case with artificial path deviations. We put the vehicle at a different position to simulate the condition that the vehicle navigates outside the learned path. The experimental result shows that the vehicle can self-correct the navigation successfully by estimating the location with respect to the monitored object, as shown in Figure 6.10.



# Chapter 7

## Obstacle Avoidance in Various Floor Environments

### 7.1 Overview of Obstacle Avoidance Methods

While the vehicle navigates, an obstacle appearing on the navigation path suddenly is an ordinary situation. In recent years, many methods for vision-based obstacle avoidance have been proposed. Ku and Tsai [17] proposed a vision-based approach by a quadratic classifier for obstacle avoidance. Obstacles including walls and objects are considered as patterns, and are used as input to the quadratic classifier. Chen and Tsai [4] proposed a fuzzy guidance technique by utilizing the result of route area extraction to compute the collision-free direction for the same purpose. Chiang and Tsai [3] also proposed a goal-directed minimum path following approach to compute collision-free paths. However, most methods are specific in their applications and unable to react in certain realistic environments. For real applications, the environment is usually complicated and the floor typically consists of textures of various colors. Hence, in order to avoid an obstacle in such environments, we propose an obstacle avoidance method for various floor environments for the proposed vehicle system.

In Section 7.2, we describe the idea of the proposed obstacle avoidance method firstly. The detailed techniques used for obstacle avoidance are described in Section

7.3. Some experimental results are shown in Section 7.4.

## **7.2 Idea of Proposed Obstacle Avoidance Method**

In this study, we propose an obstacle avoidance method in various floor environments by utilizing the colors of the floor. The first step of proposed obstacle avoidance is obstacle detection. We adopt k-means clustering for detection of an obstacle to find clusters of floor colors, and an alert line scanning technique to recognize an obstacle on the floor, as described in Sections 7.3.1 and 7.3.2, respectively. As soon as an obstacle is detected, we adopt the goal-directed minimum path following technique [3] to create a collision-free path, as described in Section 7.3.3.

For the convenience to describe the obstacle avoidance method, we define some coordinate systems and the direction angle of the vehicle, as described in Section 7.2.1. And the coordinate transformations among these coordinate systems are described in Section 7.2.2.

### **7.2.1 Coordinate system and direction angle of vehicle**

In this section, three coordinate systems: the global coordinate system, the vehicle coordinate system, and the image coordinate system, are used to describe the vehicle location and the navigation environment, as defined in Chapter 6. The learned



navigation path including several path nodes uses the GCS to represent the location of each node. While the vehicle navigates, the odometer of the vehicle provides the current location of the vehicle and the direction angle of the vehicle in the global coordinate system. The angle, denoted as  $\Theta$ , represents the rotation degree between the direction of the vehicle and the  $G_x$  axis of the global coordinate system and plays an important role in coordinate transformation.

## 7.2.2 Coordinate transformation

Among the three coordinate systems, we need the transformation between the image coordinate system and the vehicle coordinate system, and the transformation between the vehicle coordinate system and the global coordinate system. The first transformation is known by the location mapping calibration described in Section 3.2.1. The coordinate transformation between the vehicle coordinate system and the global coordinate system is illustrated in Figure 7.1.

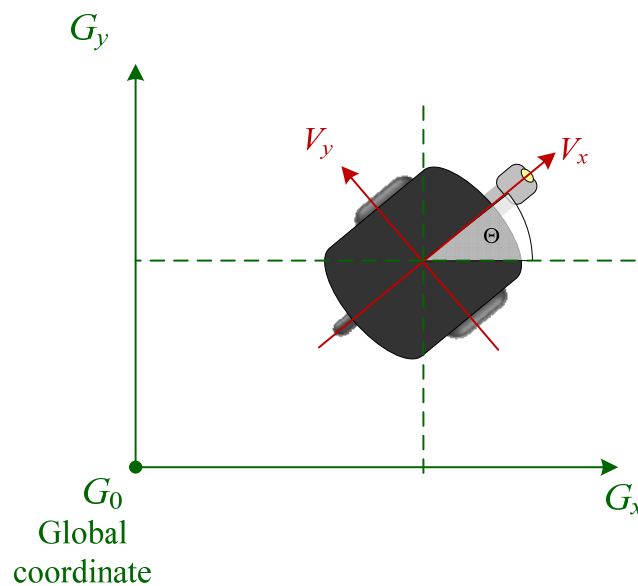


Figure 7.1 The coordinate transformation between the VCS and the GCS.



Based on this figure, the transformation functions between the vehicle coordinate system and the global coordinate system using the direction angle can be calculated by Equations (7.1) and (7.2) in the following, where  $x_v$  and  $y_v$  are the coordinates of the vehicle in the world coordinate system.

$$x = V_x \cos \Theta - V_y \sin \Theta + x_v, \quad (7.1)$$

$$y = V_x \sin \Theta + V_y \cos \Theta + y_v. \quad (7.2)$$

## 7.3 Obstacle Avoidance Techniques

### 7.3.1 Finding floor colors by k-means clustering

In order to find the most representative clusters of the floor colors, an efficient algorithm to cluster the floor colors is needed. Cluster analysis is one of the applicable techniques. It has long been used for image segmentation and other image analysis works. The clustering technique for color image segmentation normally chooses the RGB space as the feature space. However, for real applications, the color distance in the RGB space usually cannot represent the differences of the colors in the real world. Lucchese and Mitra [22] proposed an unsupervised segmentation algorithm based-on k-means clustering in the chromaticity plane. They suggested that exploiting the separability of colors in the 3D space may be projected onto a 2D chromatic subspace and onto a 1D luminance subspace. This inspires us in this study to propose an efficient unsupervised k-means clustering algorithm for clustering the floor colors.

Among the techniques developed for clustering, the *k-means clustering*

*algorithm* [18, 19] is the most widely used and studied. In order to meet real-time processing, we need an efficient k-means clustering algorithm. Kanungo et al. [20] proposed a filtering algorithm which is an efficient implementation of Lloyd's k-means clustering algorithm [21]. Hence, we adopted it as the basis of the proposed k-means clustering algorithm.

For a given color image, we firstly transform the color image from the RGB space into the  $Lu'v'$  space using the following equations:

$$L = \begin{cases} 116Y^{\frac{1}{3}} - 16 & \text{for } Y > 0.008856, \\ 903.3Y & \text{for } Y \leq 0.008856, \end{cases}$$

$$u' = \frac{4X}{(X + 15Y + 3Z)},$$

$$v' = \frac{9Y}{(X + 15Y + 3Z)},$$
(7.3)

where

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$
(7.4)

In order to achieve an unsupervised clustering, we apply a 2D k-means algorithm where  $k = 2, 3, \dots, n$ , on the  $u'v'$  chromaticity plane of the image, to find the a set of  $k$  clusters, denoted as  $C_{ch}$ , so as to minimize the mean squared distance, the so-called *distortion*, from each color to its corresponding cluster, respectively. Obviously, as the number  $k$  of the clusters increases, the distortion decreases. In realistic environments, the colors of the floor are normally limited to several kinds. Thus we set up thresholds to prevent too many clusters. There are two thresholds: a threshold  $\tau_1$  to limit the

minimum distance between the centers, and a threshold  $\tau_2$  to limit the difference between the average distance of the cluster centers and the average distortion. The detailed algorithm is described in the following.

**Algorithm 7.1.** *2D unsupervised k-means clustering in the  $u'v'$  chromaticity plane.*

*Input:* A set of sample colors,  $\mathbf{V}$ , in the  $u'v'$  chromaticity plane, two thresholds  $\tau_1$  and  $\tau_2$ .

*Output:* A set of clusters,  $\mathbf{C}_{ch}$ .

*Steps:*

- Step 1. Set  $k = 2$ .
- Step 2. Apply k-means algorithm on  $\mathbf{V}$ , where  $i = 1, 2, \dots, N$  with  $k$  being the number of the clusters.
- Step 3. Compute the average *distortion*  $r$  of the clustering result.
- Step 4. Compute average distance  $d_{avg}$  between the centers of the clusters and find the minimum distance  $d_{min}$  among them.
- Step 5. If  $d_{min}$  is smaller than  $\tau_1$ , repeat Step 1.2 with  $k = k - 1$  to get the clusters as  $\mathbf{C}_{ch}$  and return the resulting clusters and finish the algorithm.
- Step 6. If  $|r - d_{avg}|$  is smaller than  $\tau_2$ , return the clusters computed in Step 1 as  $\mathbf{C}_{ch}$  and finish the algorithm; otherwise, repeat Steps 1.2 through 1.6 with  $k = k + 1$ .

After we obtain a set of clusters,  $\mathbf{C}_{ch}$ , the next step is to find the luminance clusters for each cluster in  $\mathbf{C}_{ch}$ . We apply a 1D k-means clustering algorithm which is a simple dimensional reduction of the 2D algorithm above to obtain the clusters  $\mathbf{C}_l$  for each cluster in  $\mathbf{C}_{ch}$ . The thresholds  $\tau_3$  and  $\tau_4$  for the 1D algorithm play the corresponding roles of thresholds  $\tau_1$  and  $\tau_2$  of the 2D algorithm, respectively. By combining the clusters  $\mathbf{C}_l$  and their corresponding clusters in  $\mathbf{C}_{ch}$ , we can obtain a set

of the floor color clusters, denoted as  $C_{floor}$ . A flowchart of the entire clustering process is shown in Figure 7.2.

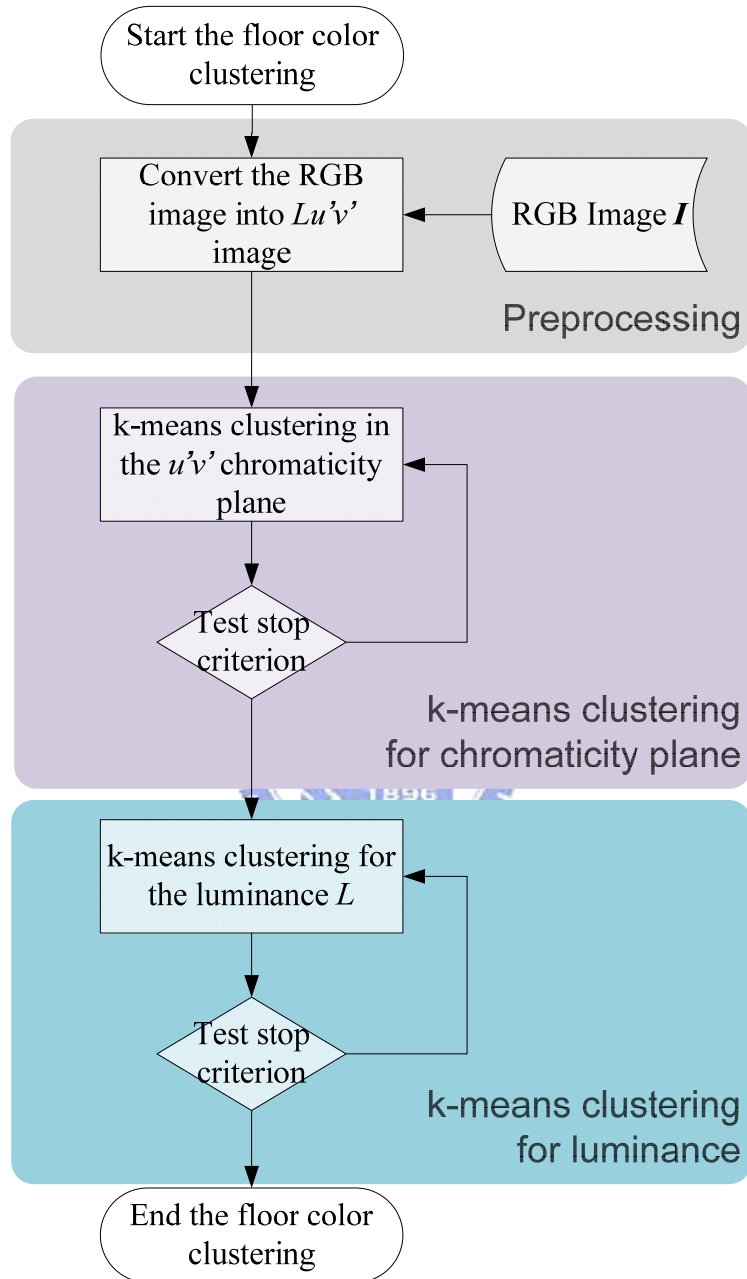


Figure 7.2 A flowchart of finding the floor color clusters by the k-means clustering.

### 7.3.2 Obstacle detection by alert line scanning

Once we obtain the floor color clusters, we can utilize the clusters to recognize

the non-floor colors by scanning an alert line. The alert line is established by three line segments in the image, as shown in Figure 7.3. If an obstacle is detected on the alert line, the vehicle should change its navigation path to avoid the obstacle.



Figure 7.3 An alert line specified by three red line segments in the image.

Each color sample of the image on the alert line is scanned respectively. We first transform the color samples, denoted as  $C_{alert}$ , into the  $Lu'v'$  space, and then compare each sample  $C_i$  in  $C_{alert}$  with the clusters  $C_{floor}$  in the  $Lu'v'$  space. If the  $C_i$  is similar to  $C_{floor}$ , it means that no obstacle exists at this sample point in the image. Otherwise, the sample is marked as a candidate obstacle seed. After scanning the alert line, we can collect a set of candidate obstacle seeds. Then we apply a region growing algorithm on these seeds to find a mask of the distribution of the obstacle, and apply the morphological operations on the mask to remove the noise and small regions. Obstacles in the mask are set as white parts, as shown in Figure 7.4(b) and (c). Finally, we rescan the same alert line in the mask to detect the distribution of obstacles on the alert line. The green and the red parts represent the floor and the obstacles respectively, as shown in Figure 7.4(a). Then we can obtain the left-most side and the right-most side points of the obstacles in the alert line. The detailed process is described as an algorithm in the following.

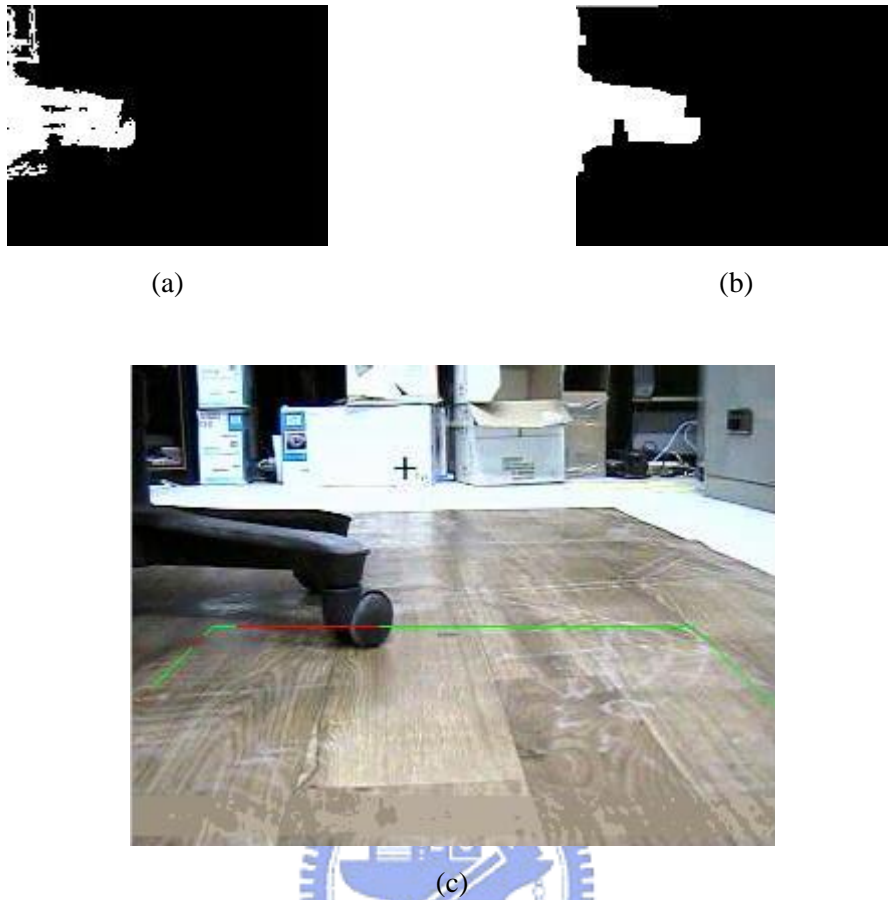


Figure 7.4 The alert scanning results. (a) The mask of the region growing result of the candidate obstacle seeds. (b) The mask generated by applying morphological operations on (a). (c) The green parts of the alert line represent the floor and the red parts of the alert line represent the obstacles.

**Algorithm 7.2.** *Alert line scanning for the detection of an obstacle.*

*Input:* An image  $I$  with an obstacle and the floor.

*Output:* Two points of the left-most side and the right-most side of the obstacles or a Boolean value ‘false’ if there is no obstacle detected.

*Steps:*

Step 1. Transform the sample points on the alert line of  $I$  from the RGB space into the  $Lu'v'$  space, with the result denoted as  $C_{alert}$ .

- Step 2. For each sample point  $C_i$  in  $C_{alert}$  and each floor color cluster  $C_j$  in  $C_{floor}$  compute the difference between  $C_i$  and  $C_j$ , with the result denoted as  $D_{i,j}$ .
- Step 3. For each  $D_{i,j}$ , if  $|D_{i,j}|$  is larger than a threshold, apply region growing using the corresponding  $C_i$  as a seed to create a mask by filling the grown region with the value 255, and apply the AND operation to merge the masks created by the region growing by computing the per-element bit-wise logical conjunction of the masks.
- Step 4. Apply the morphological operations to the mask.
- Step 5. Scan the alert line in the mask, and find the first and the last samples with values equal to 255, denoted as  $O_L$  and  $O_R$ .
- Step 6. If  $O_L$  and  $O_R$  exist, return these points; otherwise, return a Boolean value 'false.'



### 7.3.3 Computation of goal-directed minimum path

In the previous section, we have detected the distribution of the obstacles, and obtain the left-most side and the right-most side points  $O_L$  and  $O_R$ . Once an obstacle is detected, a new path is planned to guide the vehicle to avoid the obstacle. In this study, we adopted the goal-directed minimum path proposed by Chiang and Tsai [3]. The first step is to transform  $O_L$  and  $O_R$  in the image coordinate system to the vehicle coordinate system. Then, the computation of a goal-directed minimum path is illustrated in Figure 7.5, where  $W_v$  is the appropriate width for the vehicle to pass and  $D_G$  is the shortest distance between the obstacle and the vehicle. With some geometric computation, we can obtain a node, the so-called *obstacle avoidance node*, to guide the vehicle to avoid an obstacle, by choosing the shortest distance of the paths. The

detailed computation is described as an algorithm in the following.

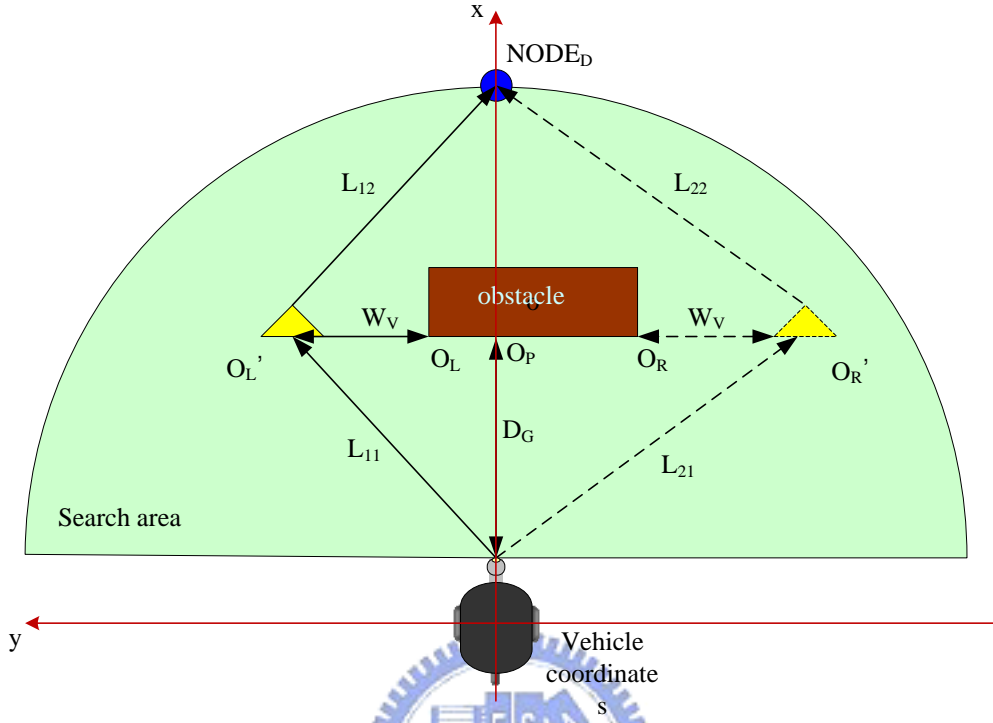


Figure 7.5 An illustration of computation of goal-direction minimum path (a modified version from [3]).

**Algorithm 7.3.** *Computation of goal-directed minimum path.*

*Input:* Two points of the left-most side and the right-most side of the obstacles, denoted as  $O_L$  and  $O_R$ , in the vehicle coordinate system.

*Output:* A node of the goal-directed minimum path.

*Steps:*

Step 1. Compute the two candidate nodes  $O_L'$  and  $O_R'$  for obstacle avoidance in the VCS using the following equations:

$$\begin{aligned} O_L' &= \bar{u} \cdot W_v + \overrightarrow{O_L O_P} + \overrightarrow{O_P V_P}, \\ O_R' &= \bar{u} \cdot W_v + \overrightarrow{O_R O_P} + \overrightarrow{O_P V_P}, \end{aligned} \quad (7.5)$$

where  $\bar{u}$  is the unit vector and  $V_P$  is the vehicle location in the vehicle



coordinate system.

- Step 2. Transform  $O_L'$  and  $O_R'$  in the vehicle coordinate system to the global coordinate system using Equations (7.1) and (7.2), with the results denoted as  $O_{GL}$  and  $O_{GR}$ , and compute the two lengths  $Length_L$  and  $Length_R$  of the path after inserting the nodes  $O_{GL}$  and  $O_{GR}$  for obstacle avoidance using the following equations:

$$\begin{aligned} Length_L &= \left| \overrightarrow{V_G O_{GL}} \right| + \left| \overrightarrow{O_{GL} N_{next}} \right|, \\ Length_R &= \left| \overrightarrow{V_G O_{GR}} \right| + \left| \overrightarrow{O_{GR} N_{next}} \right|, \end{aligned} \quad (7.6)$$

where  $V_G$  is the vehicle location in the global coordinate system.

- Step 3. If  $Length_L$  is smaller than  $Length_R$ , choose  $O_{GL}$  as the obstacle avoidance node; else, choose  $O_{GR}$ .
- Step 4. Using Equation (7.7) in the following to check whether the node is in a pre-defined search area in the image or not. If not, compute the node position using Equation (7.8) in the following:

$$\left| \overrightarrow{V_G N} \right| > r, \quad (7.7)$$

$$N = u_N \cdot r, \quad (7.8)$$

where  $N$  is the node for obstacle avoidance,  $r$  is the distance between the vehicle and the next node, and  $u_N$  is the unit vector of  $\overrightarrow{V_G N}$ .

- Step 5. Return the node position.

## 7.4 Experimental Results

In this chapter, we have proposed an unsupervised k-means clustering algorithm to find clusters of the floor colors. We have tested several kinds of floors, as shown in Figure 7.6. The experimental results are shown in Figure 7.7.

For the convenience to observe the clustering result, we constructed an  $u'v'$  chromaticity plane image. Each sample point is drawn as a point with the same color for each cluster, as shown in Figure 7.7(b). The white circles specify the clusters' average distance between the samples and their corresponding cluster centers. Each clusters' samples are specified by different colors. We collected the color samples of the alert line in the image. Each color samples of the alert line is drawn as a purple point in the  $u'v'$  chromaticity plane image, as shown in Figure 7.7(b). Then, we filtered out the samples with similar colors of the clusters, as shown in Figure 7.7(d), and apply region growing to find the distribution of the obstacle. Another experimental result for a different floor and a different obstacle is shown in Figure 7.8.



Figure 7.6 Two kinds of floors.

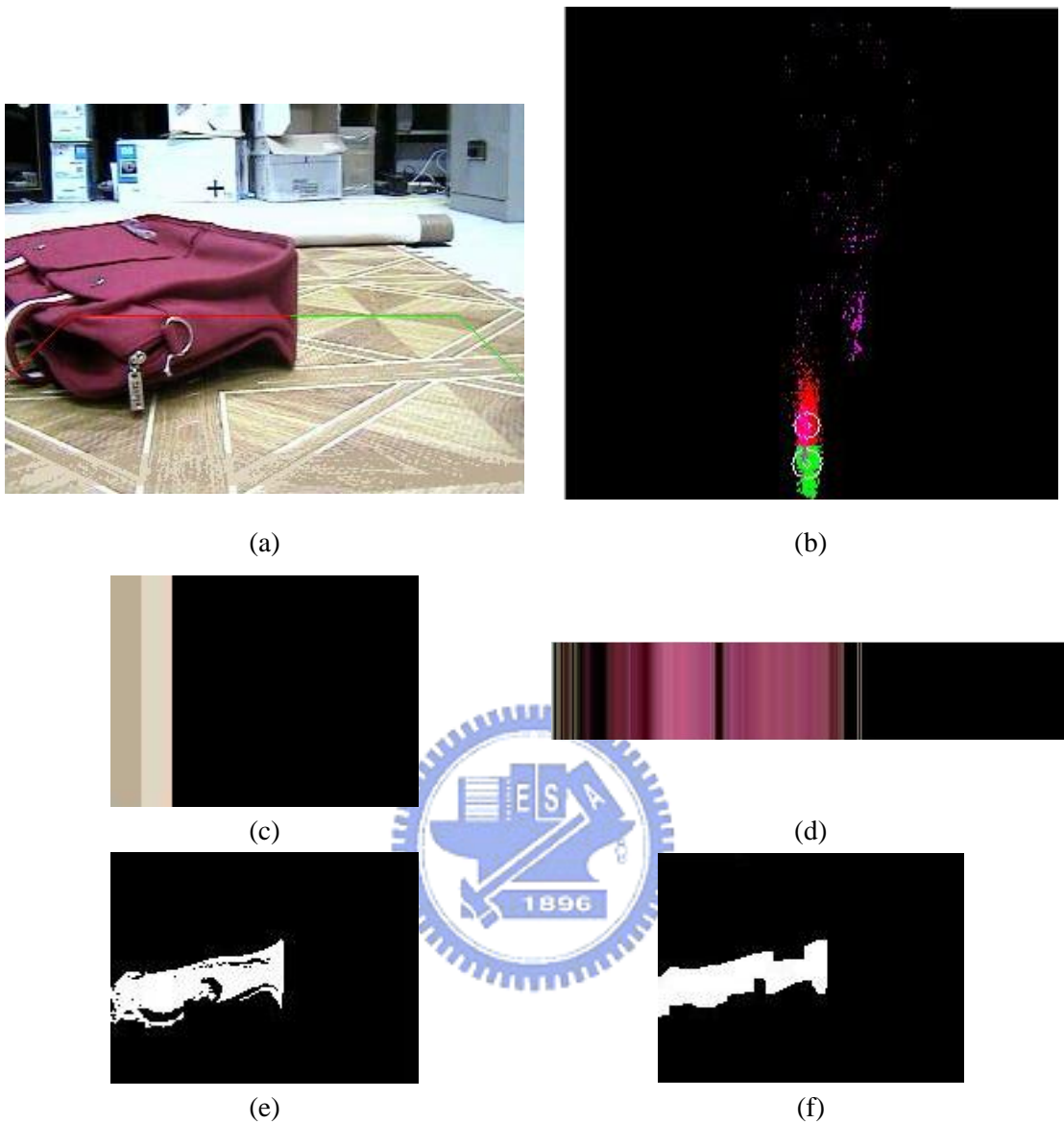


Figure 7.7 The experimental result of obstacle detection by k-means clustering. (a) The image contains the floor and an obstacle. The red parts of the alert line specify the obstacle and the green parts of the alert specify the floor. (b) The clustering result which is shown in the  $u'v'$  chromaticity plane. (c) The colors of the clusters. (d) The non-floor colors on the alert line. (e) The mask created by region growing. (f) The mask after applying the morphological operations.

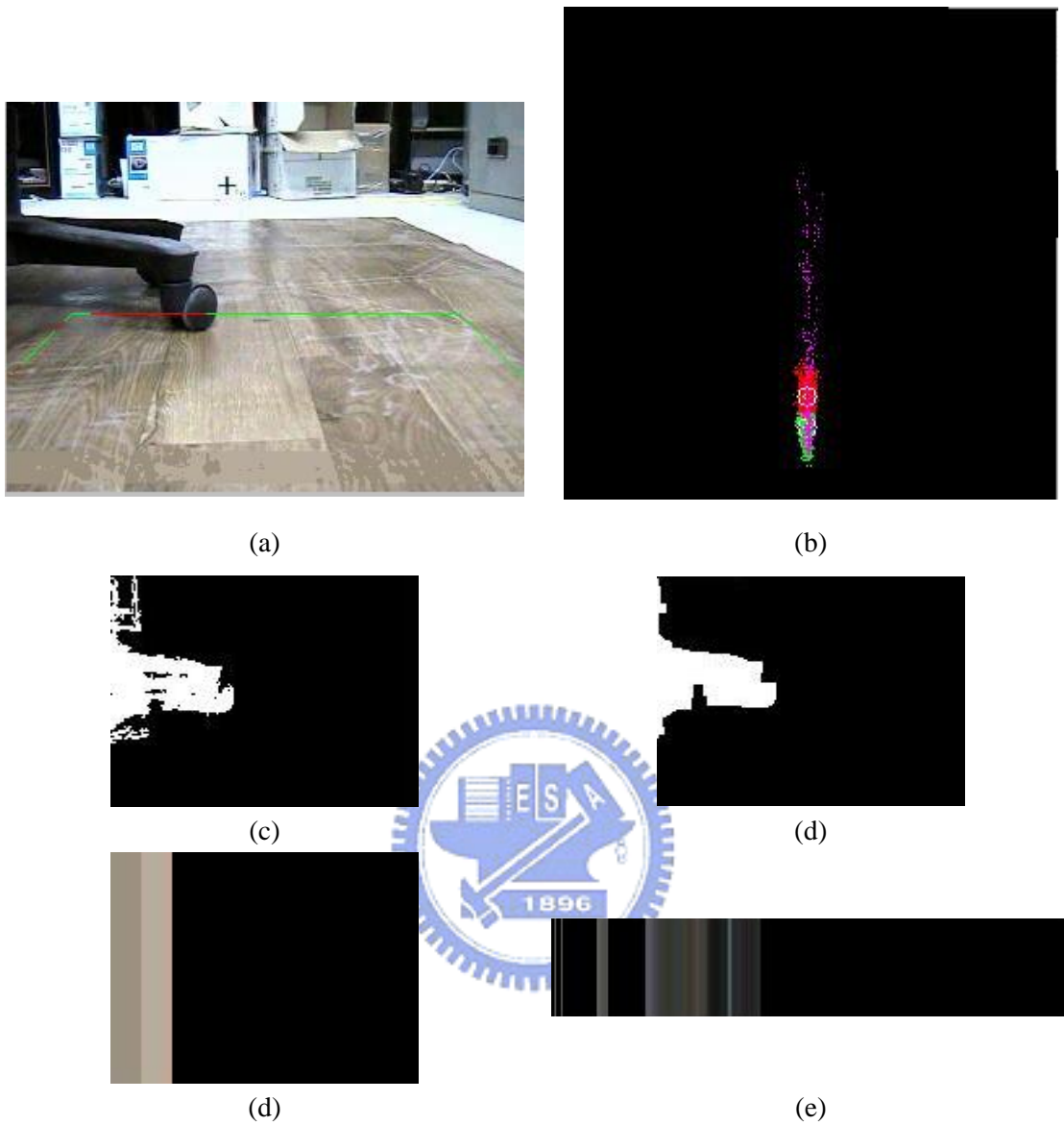


Figure 7.8 Another experimental result of detecting an obstacle by k-means clustering. (a) The image contains the floor and an obstacle. (b) The clustering result. (c) The colors of the clusters. (d) The non-floor colors on the alert line. (e) The mask created by region growing. (f) The mask after applying the morphological operations.



the vehicle to learn a path and some monitored objects on the walls. In this study, monitored objects are paintings and posters. Whenever the vehicle arrives at a spot, the user controls the system to record the monitored-object features and the calibration information. After the learning process, a navigation map is created. An illustration of the learned data, the navigation map, and the actual navigation path created in the experiment is shown in Figure 8.2.

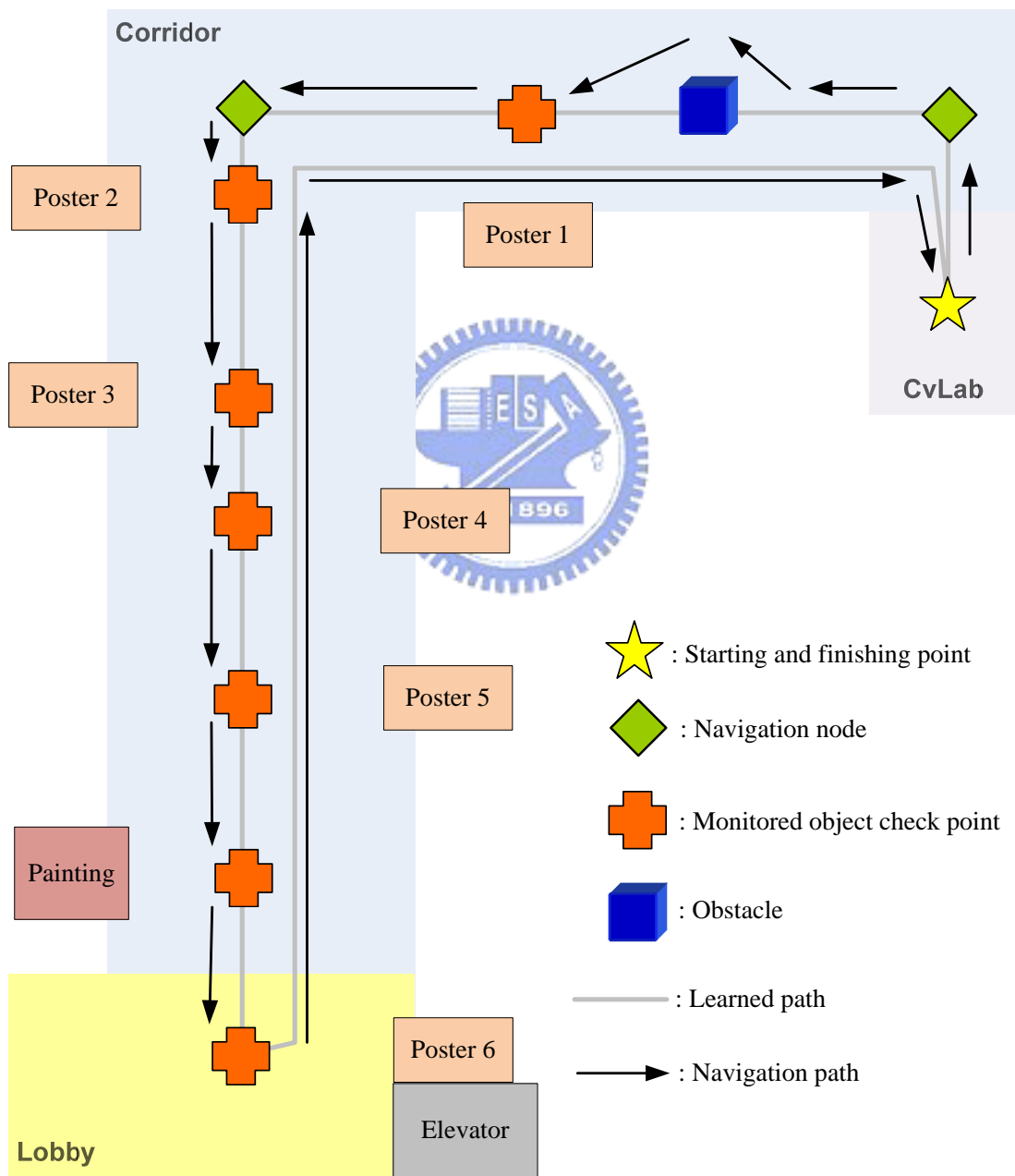


Figure 8.2 An illustration of learned data and navigation path.

The vehicle starts security patrolling according to the created map. The navigation process is shown in Figure 8.2. Whenever the vehicle arrives at a learned monitoring node, it performs the security check of the existence of the monitored object. If the check is successful, the vehicle adjusts its location to continue its navigation on the right way according to the matching result; otherwise, a message is issued. For each monitored object shown in Figure 8.2, the experimental results are shown in Figure 8.3. In Figure 8.3, the vehicle performed security monitoring to monitor 7 monitored objects. The vehicle arrived at the learned monitored node, as shown in Figure 8.3(b). Then, it extracted the features of the image and matched with the corresponding learned data. The matching results are shown in Figure 8.3(c) and the learned monitored objects are shown in Figure 8.3(d).

In addition, while the vehicle navigates, the vehicle will detect obstacles by scanning the alert line, as shown in Figure 7.8(b). If the vehicle detects an obstacle, a new path is planned to guide the vehicle to avoid the obstacle, as shown in Figure 7.8(c) through (h).



Figure 8.3 The experimental result of object monitoring and navigation path correction. (a) Monitored object labels. (b) The vehicle monitors the monitored objects. (c) The matching result and the horizontal line used for path correction. (d) The image of learned monitored objects.



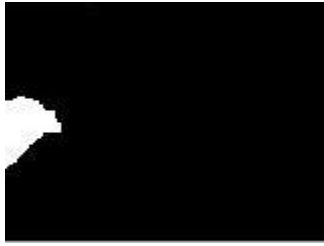


Figure 8.3 The experimental result of object monitoring and navigation path correction. (a) Monitored object labels. (b) The vehicle monitors the monitored objects. (c) The matching result and the horizontal line used for path correction. (d) The image of learned monitored objects. (continued)

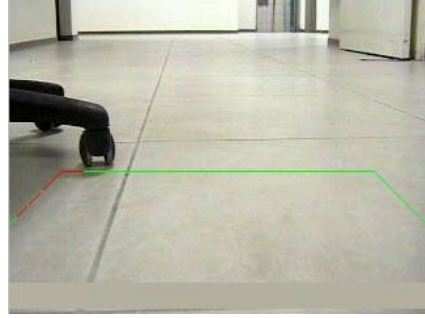


Obj5.			
Obj6.			
Obj7.			
(a)	(b)	(c)	(d)

Figure 8.3 The experimental result of object monitoring and navigation path correction. (a) Monitored object labels. (b) The vehicle monitors the monitored objects. (c) The matching result and the horizontal line used for path correction. (d) The image of learned monitored objects. (continued)



(a)



(b)



(c)



(d)



(e)



(f)

Figure 8.4 The vehicle detects an obstacle and changes the navigation path to avoid the obstacle. (a) The mask of detected obstacle region. (b) The image contains the floor, an obstacle, and the alert line specified the distribution of the obstacle. (c)~(h) show the vehicle avoiding an chair as an obstacle.

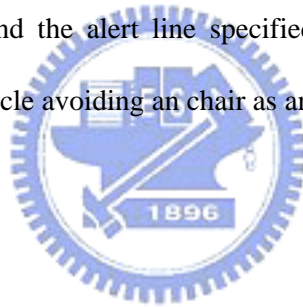


(g)



(h)

Figure 8.4 The vehicle detects an obstacle and changes the navigation path to avoid the obstacle. (a) The mask of detected obstacle region. (b) The image contains the floor, an obstacle, and the alert line specified the distribution of the obstacle. (c)~(h) show the vehicle avoiding an chair as an obstacle. (continued)



## 8.2 Discussions

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

- (1) The reflect light of the lamplight will affect the intensity of the image taken by the camera because the camera is placed at the lower position, resulting in fewer matched pairs computed by the SIFT algorithm. Thus, the horizontal line, which is computed by applying the extracted affine transform, in the image is not identical, so the computed coefficients of the linear equation will yield errors. The vehicle location estimation also produces errors.
- (2) There are two constraints of the proposed system, namely, the floor has to be flat

and the luminance has to be even. The obstacle avoidance method used in this study uses the colors of the floor which is close to the vehicle as the reference to scan the alert line. If the colors of the floor on the alert line are different to the ones of the floor which is close to the vehicle, it is difficult to separate the floor and other things on the floor. The distance between the obstacle and the vehicle is also difficult to obtain. This problem should be solved in the future.



# Chapter 9

## Conclusions and Suggestions for Future Works

### 9.1 Conclusions

In this study, several techniques and strategies have been proposed and integrated into an autonomous vehicle system for security patrolling in indoor environments with capabilities of specific-object monitoring and self-adjustment of navigation paths. Satisfactory navigation results have been obtained by this system.

At first, an easy-to-use learning technique is proposed, which has the capability of extracting specific features, including navigation path, floor color, monitored object, vehicle location with respect to monitored objects. A user can easily control the vehicle with a designed interface to navigate in the environment and specify concerned objects in the image for later security monitoring.

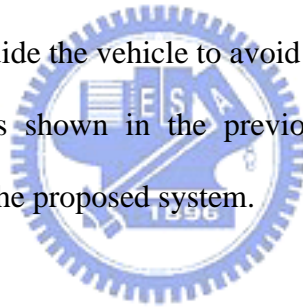
Next, a security patrolling method by vehicle navigation with obstacle avoidance and security monitoring capabilities has been proposed. The vehicle navigates according to the node data of the path map which is created in the learning phase and monitors the concerned objects by a 2D object image matching technique proposed in this study, the simplified-SIFT algorithm. Accordingly, we can extract the features of the monitored object from acquired images and match them with the learned data. The matching technique is based on the Hough transform. We construct a Hough transform histogram to predict the model location, orientation, and scale from the

match hypothesis, and find the best match by finding the peak in the Hough space.

In addition, a vehicle location estimation technique by utilizing the monitored object matching result has been proposed. The coefficients of the equation of a horizontal line and the location of the start point in the image are used to estimate the vehicle location. Also proposed is a path correction method, which compares the estimated location and the learned one to compute necessary path adjustment and transform it into the global coordinate system to correct the navigation path.

Finally, for obstacle avoidance, a k-means clustering algorithm for finding clusters of the floor colors has been proposed, by which we can detect obstacles in environments with various floor colors. We have also proposed an alert line scanning technique to detect obstacles and integrated it with a technique of goal-directed minimum path following to guide the vehicle to avoid the obstacle.

The experimental results shown in the previous chapters have revealed the feasibility and practicality of the proposed system.



## 9.2 Suggestions for Future Works

The proposed strategies and methods, as mentioned previously, have been implemented on a vehicle system. Based on our experience of the experiments, several suggestions and related interesting issues are worth further investigation in the future. We state them as follows.

- (1) Using an omni-directional camera for obstacle avoidance to take wider-view images of the environment which is close to the vehicle.
- (2) Design a learning data managing system for the incremental learning of monitored objects and learned data management.



- (3) Adding more kinds of features for vehicle location estimation.
- (4) Adding the capability of transmitting warning messages from the system to a user's cell phone or electronic mail address to warn the user immediately.
- (5) Adding the capability of voice control when users want to issue navigation orders to the vehicle.
- (6) Adding the capability of starting navigation from arbitrary start points.



# References

- [1] G.N. DeSouza and A.C. Kak, "Vision for mobile robot navigation: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No 2, pp. 237-267, February 2002.
- [2] C. C. Lai, "A study on automatic indoor navigation techniques for vision-based mini-vehicle with off-line environment learning capability," *M. S. Thesis*, Department of Computer and Information Science, National Chiao Tung University, Taiwan, June, 2003.
- [3] K. L. Chiang, "Security patrolling and danger condition monitoring in indoor environments by vision-based autonomous vehicle navigation," *M. S. Thesis*, Institute of Computer Science and Engineering, National Chiao Tung University, Taiwan, June, 2006.
- [4] Y. C. Chen and W. H. Tsai, "Vision-based autonomous vehicle navigation in complicated room environments with collision avoidance capability by simple learning and fuzzy guidance techniques," *Proceedings of 2004 Conference on Computer Vision, Graphics and Image Processing*, Hualien, Taiwan, Republic of China, Aug. 2004.
- [5] M. C. Chen and W. H. Tsai "Vision-based security patrolling in indoor environments using autonomous vehicles," *Proceedings of 2005 Conference on Computer Vision, Graphics and Image Processing*, Taipei, Taiwan, Republic of China, Aug, 2005.
- [6] C. Shmid and R. Mohr. "Local greyvalue invariants for image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 5, pp. 530-534, 1997.
- [7] K. Mikoljczyk and C. Schmid. "Indexing based on scale-invariant features," in



- Proceedings of International Conference on Computer Vision*, pp. 525-531, 2001.
- [8] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, Vol. 60, No. 2, pp. 91-110, 2004.
- [9] I. Fukui, "TV image processing to determine the position of a robot vehicle," *Pattern Recognition*, Vol. 14, pp. 101-109, 1981.
- [10] M. J. Magee and J. K. Aggarwal, "Determining the position of a robot using a single calibration object," *Proceedings of IEEE Conference on Robotics*, pp. 57-62, Atlanta, Georgia, May 1983.
- [11] J. Huang, C. Zhao, Y. Ohtake, H. Li, and Q. Zhao, "Robot position identification using specially designed landmarks," *Proceedings of 2006 IEEE Conference on Instrumentation and Measurement Technology*, April 2006.
- [12] H. L. Chou and W. H. Tsai "A new approach to robot location by house corners," *Pattern Recognition*, Vol. 19, pp. 439-451, 1986.
- [13] K. L. Chiang and W. H. Tsai, "Vision-based autonomous vehicle guidance in indoor environments using odometer and house corner location information," *Proceedings of 2006 IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 415-418, Dec. 18 - 20, 2006.
- [14] C. Shu, A. Brunton, and M. Fiala, "CAMcal: A program for camera calibration using checkerboard patterns," *Proceedings of 9th IEEE International Conference on Computer Vision (ICCV2003) demo session, CDROM proceedings*, Nice, France, October, 2003.
- [15] K. Mikolajczyk, and C. Schmid, "A performance evaluation of local descriptors," *Proceedings of International Conference on Computer Vision & Pattern Recognition*, Vol. 2, pp. 257-263, June 2003.
- [16] D. G. Lowe, "Local feature view clustering for 3D object recognition," *IEEE Conference on Computer Vision and Pattern Recognition*, Kauai, Hawaii, pp.

682-688, December 2001.

- [17] C. H. Ku and W. H. Tsai, "Obstacle avoidance in person following for vision-based autonomous land vehicle guidance using vehicle location estimation and quadratic pattern classifier," *IEEE Transactions on Industrial Electronics*, Vol. 48, No. 1, pp. 205-215, 2001.
- [18] E. Forgey, "Cluster analysis of multivariate data: Efficiency vs. interpretability of classification," *Biometrics*, Vol. 21, pp. 768, 1965.
- [19] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proceedings of Fifth Berkeley Symposium Math. Statistics and Probability*, Vol. 1, pp. 281-296, 1967.
- [20] T. Kanungo, D. M. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 7, pp. 881-892, 2002.
- [21] S. P. Lloyd, "Least Squares Quantization in PCM," *IEEE Transactions on Information Theory*, Vol. 28, pp. 129-137, 1982.
- [22] L. Lucchese, S. K. Mitra, "Unsupervised segmentation of color images based on k-means clustering in the chromaticity plane," *Proceedings of IEEE Workshop on Content-Based Access of Image and Video Libraries*, pp.74, June 22-22, 1999.