國立交通大學

資訊科學與工程研究所

碩士論文

以俯視式環場攝影機作多人擴增實境式室內商品導覽

Indoor AR-based Multi-user Navigation for Merchandise Shopping Using Down-looking Omni-cameras

研究生：楊舒琳

指導教授：蔡文祥　教授

中華民國一百零二年六月

以俯視式環場攝影機作多人擴增實境式室內商品導覽

Indoor AR-based Multi-user Navigation for Merchandise Shopping Using Down-looking Omni-cameras

研 究 生：楊舒琳　　Student：Shu-Lin Yang

指導教授：蔡文祥　　Advisor：Wen-Hsiang Tsai

國 立 交 通 大 學

資 訊 科 學 與工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2013

Hsinchu, Taiwan, Republic of China

中華民國一百零二年六月

Indoor AR-based Multi-user Navigation for Merchandise Shopping Using Down-looking Omni-cameras

Student: Shu-Lin Yang Advisor: Wen-Hsiang Tsai

Institute of Computer Science and Engineering

National Chiao Tung University

ABSTRACT

When people enter unfamiliar indoor environments, like shopping malls, supermarkets, grocery stores, etc., they generally have to rely on staff members to guide them to the locations of desired merchandise items. In this study, an indoor multi-user navigation system based on augmented reality (AR) and computer vision techniques by the use of a mobile device like an HTC Flyer is proposed.

At first, an indoor vision infra-structure is set up by attaching fisheye cameras on the ceiling of the navigation environment. The locations and orientations of multiple users are detected from the acquired images using the fisheye cameras by a remote server-side system, and the analysis results are sent to the client-side system on each user's mobile device. Meanwhile, the server-side system also analyzes the acquired images to recognize merchandise items and sends the information of the surrounding environment and the merchandises, as well as the navigation path to the client-side system. The client-side system then displays the information in an AR way on the mobile device, which provides clear information for each user to conduct the navigation.

For multi-user identification, a method is proposed to attach a multicolor-edge mark on top of each user's mobile device and the server-side system analyzes them in each consecutive image frame captured by the closest fisheye camera and classifies the edge mark according to its color pattern to obtain the identification number of each user.

For multi-user localization, a method is proposed to analyze the omni-image captured from the fisheye cameras and detect human activities in the environment. The server-side system separates the foreground from the background in the image and detects the location of each user. Furthermore, three techniques are proposed and integrated together to conduct user orientation detection effectively. The first technique is analysis of user motions in consecutive images. The second is utilization of the orientation sensor on the user's mobile device. The last is estimation for the direction of the multi-color edge mark attached on the top of the mobile device using the omni-image.

For AR-based merchandise guidance, the client-side system sends the image captured from client device camera to the server-side system. Then, the server-side system analyzes it by the SURF algorithm, matches the resulting features against a pre-constructed merchandise image database, and transmits the corresponding information to the client-side system for display in an AR way. Also, a path planning technique is used for generating a collision-free path from the current user's position to a selected merchandise item via the use of an environment map. Finally, the navigation and merchandise information is overlaid onto the images shown on the mobile devices. In this way, the system can accomplish the AR functions and provide a convenient guidance for merchandise shopping or other similar activities. Good experimental results show the feasibility of the proposed system and methods.

以俯視式環場攝影機作多人擴增實境式室內商品導覽

研究生：楊舒琳　　指導教授：蔡文祥 博士

國立交通大學資訊科學與工程研究所

摘要

本研究提出了一個室內商品導覽系統，結合了電腦視覺及擴增實境技術，利用事先在室內天花板上安裝的魚眼攝影機來建立基礎硬體設備，並採用主從式系統架構及平板電腦裝置，提供更直覺的擴增實境介面，供使用者應用。

首先，本研究提出了一個以電腦視覺為基礎的方法，可用於進行多使用者定位及辨識。該方法分析魚眼攝影機影像來偵測使用者的活動資訊，利用空間映射的方式來進行影像中人物真實空間位置的轉換，並採用三項技術來偵測使用者的方向。該三項技術分別為：1)分析使用者的移動路徑；2)利用行動裝置上的方向感測器；以及 3)在行動裝置上方貼一條多色彩長條標記，並在魚眼影像中分析該標記來進行方向偵測。接著，藉由分析多色彩長條標記上不同色彩的排列組合，來進行多使用者身分辨識。

本研究也提出了一個利用加速型穩健特徵(speeded-up robust feature, SURF)演算法來進行商品辨識導覽的方法，該方法是從使用者端傳送行動裝置上的影像至伺服器端，再由伺服器端利用事先建立的商品資訊資料庫進行比對與辨識。最後，伺服器端將導覽資訊傳送到行動裝置上的使用者端，此資訊包括商品資訊、定位資訊、周遭環境地點及搜尋商品路徑，讓使用者端能將導覽資訊覆蓋在行動裝置影像中對應的真實物件上，來提供擴增實境導覽介面。

最後，上述方法的實驗結果良好，顯示出本研究所提系統確實可行。

ACKNOWLEDGEMENTS

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

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

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

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1
Introduction

1.1 Background and Motivation

When people are entering unfamiliar indoor environments, like shopping malls, supermarkets, grocery stores, etc., they generally have to rely on store staff members to guide them to the locations of desired merchandise items. If the price label of a desired item is found missing by a customer during the checkout process, he/she has to request a store staff member to find the answer. In addition, when shopping a merchandise item, a customer usually will take it off the shelf and check the detailed information, such as the brand, ingredients, manufacturer, etc., of the merchandise. If the merchandise is putted on a higher shelf, he/she will have to pay more effort to acquire it for such an information-checking action. People often find these basic needs an annoying problem in a market place with very few salespersons roaming among the aisles to help merchandise search works. Thus, people need a more intuitive navigation system to guide them to get to desired merchandise items.

Commonly-seen systems for outdoor navigation purposes use the Global Positioning System (GPS) to retrieve the user's position, but the GPS is generally not precise enough to provide the indoor location of the user since its signals will be attenuated and scattered by roofs, walls, and other objects in indoor environments. Specifications for many GPS receivers indicate their accuracy will, 95% of the time, be within about 3 to 15 meters.

In this study, it is desired to design an indoor navigation system using a different

localization technique for purposes like merchandise shopping in various types of market places. Specifically, we propose an image-based localization technique to analyze the images captured from omni-cameras installed on the ceiling in indoor environments and detect human activities in the environments by a mixture of computer vision and signal processing techniques.

With the advances of mobile network technologies, the applicability of the mobile device is getting wider and wider daily. Thus, in this study we try as well to use the mobile device as the user-end device. Mobile devices have evolved in recent years into powerful image and video processing equipments with built-in high-resolution cameras, color displays, and hardware-accelerated graphics. They are also connected wirelessly to broadband networks. Therefore, we can use them to develop more interesting and useful applications by combining real-world images captured from cameras with virtual augmentations of various information created by computers. In other words, the real-world environment can be *augmented* by computer-generated objects to enhance the perception of the real-world object like merchandise, and this is the so-called *augmented reality (AR)* technique. In this study, we try to design an indoor AR-based navigation system using mobile devices which are connected to a cloud server via 4G/LTE networks. With the 4G/LTE network, we can reduce the computation load of the mobile device and provide a more efficient and convenient navigation system for the above-mentioned purpose. The concept of the proposed system is illustrated in Figure 1.1.

In summary, the goal of this study is to develop an indoor AR-based navigation system for merchandise shopping and other alike activities with the following capabilities:

1. working in indoor environments, and being able to detect the positions and orientations of multiple users;

2. integrating real images with virtual augmentations, such as the nearby place information, the specification of a merchandise, …, etc., to provide the users convenient and clear navigation interfaces for merchandise shopping and similar activities;

3. planning a proper path from the user's location to the destination of a desired merchandise item, and updating the path dynamically when the user moves to a location not in the path.

Figure 1.1 Concept of proposed indoor navigation system using augmented reality technique

1.2 General Review of Related Works

In this section, we conduct a survey of works related to this study, such as indoor navigation, AR techniques, indoor positioning, visual search, etc.

1.2.1 Related Indoor Navigation Works

In recent years, there are more and more researches about indoor navigation systems using various techniques. These systems adopted different types of

positioning and search techniques to meet the requirements for indoor navigation, such as image-based techniques, sensor-based techniques, etc.

About sensor-based techniques, Lukianto, et al. [1] proposed an indoor navigation system based on an inertial navigation system (INS) for use on the smart phone. Ozdenizci, et al. [2] proposed a near field communication (NFC) based system using a smart phone, which detects the user's position by touching NFC tags. And about image-based techniques, the most common one used in image-based systems is *marker-based* navigation using camera phones. Mulloni, et al. [3] proposed an indoor navigation system which uses off-the-shelf camera phones to determine the user location in realtime by detecting unobtrusive fiduciary markers.

Our system uses image-based positioning and AR techniques. There are also systems using these techniques, such as Hile and Borriello [4] which developed an indoor navigation system that can find the camera pose by detecting the landmark in an image acquired with a phone camera, matching it with previously-cached landmarks, and then overlaying relevant information onto the image.

1.2.2 Related Augmented Reality Works

With augmented reality (AR) techniques, artificial information about an environment and objects in it can be overlaid on an image of the real world. This enhances the real world with virtual objects or digital information. AR was initially used in this way for military, industrial, and medical applications, but was soon applied to commercial and entertainment areas as well. For example, it can be used to aid in visualizing building projects, to help mechanics to perform maintenance and repair tasks, to conduct treatment of psychological disorders, etc.

In this study, the proposed system uses AR techniques for navigation. Miyashita, et al. [6] designed a museum guide system which uses a markerless tracking technique

and an AR platform — Unifeye SDK. Jongbae and Heesung [7] proposed a vision-based indoor navigation system which recognizes the location of a user by applying marker detection and image sequence matching on images captured from a wearable camera, and displaying relevant navigation information in the AR way.

1.2.3 Related Indoor Positioning Works

Indoor positioning techniques for navigation systems mentioned in the previous sections can be roughly classified into two types, sensor-based and image-based. Below is a brief description of these positioning techniques that have emerged in the past few years to aid in navigation.

In sensor-based positioning techniques, the positioning systems usually need infrastructures with infrared, RFID, NFC tags [1], or other customer-designed hardware [2]. For example, in the use of the RFID technology, the obtained position accuracy depends on the type of the tag, which is either active or passive, as well as on the amount of these tags. A weakness is that the installation of RFID sensors can be costly.

In image-based localization techniques, the positioning systems usually obtain the user's location by analysis of the images captured by cameras. The most common technique is to identify the markers which are attached on the environments [8]. The other techniques try mainly to find the image features from the images and match with the data which are cached previously by learning [4, 5].

1.2.4 Related Visual Search Works and Applications

With technology advances, using the camera phone to initiate search queries about objects is a new trend. Many visual search applications now enable mobile devices to retrieve information about products simply by snapping a photo. Such

applications can be used, e.g., for identifying products, finding information about movies, comparing prices of shopping items. Such systems include Google Goggles [9], Nokia Point and Find [10], Kooaba [11], and Amazon Snaptell [12], which are used for identifying products. For example, taking a picture of a product's barcode will enable the search for information about the product. Typically, robust image-based features like SIFT [13], SURF [14], or CHoG [15] are extracted from the photo and matched against an online database, yielding accurate retrieval results.

In this study, we try to develop an AR system for merchandise shopping and similar activities. There exist other systems using AR techniques for the same purpose. David Chen1, et al. [16] designed a mobile AR system for retrieving the information about a book by recognizing the book spine without taking it off the bookshelf.

1.3 Overview of Proposed Methods

In a navigation system, a core function is *user positioning*, called alternatively *user localization* in this study, which finds the location and orientation of the user. As discussed previously, an outdoor navigation system usually uses a GPS to retrieve the position data, but the GPS is not suitable for indoor environments due to its less precise positioning capability, as mentioned before. Therefore, we propose a new method for indoor user localization in this study.

In most indoor environments, many omni-cameras (like fish-eye cameras) are installed on the ceiling. Thus, we construct at first a top-down vision infrastructure in the indoor environment using fish-eye cameras. Then, we create an *indoor environment map*, including the specifications and places of the merchandises for uses in AR-based guidance of the user. Furthermore, we analyze the images captured from the cameras to detect the positions of *multiple* users via the use of a server-side

system.

To identify multiple users in an indoor environment, we propose a new user localization method which detects a *multi-color edge mark* attached on top of the mobile device held by each user via the use of the client site, namely, the computing unit in the mobile device which is a smart phone or pad. With the use of the multiple colors of the edge mark, we have more choices than a single color. We can detect the users holding client devices with differently-colored edge marks from acquired images, and identify them by a color classification scheme proposed in this study.

Figure 1.2 Two users holding devices with multi-color edge marks.

After getting the users' locations, the server system sends the environment information to the client system, which then displays the information on the device screen at the client site. For merchandise shopping, we analyze, by the use of the server-side system, the images captured with the camera on the client system. The server extracts, with an SURF algorithm [14], the features existing in the images transmitted by the client, and then matches the result with a pre-constructed database of merchandises. The server sends subsequently the corresponding information of the merchandise to the client system. Besides, in order to guide a user to get the desired merchandise item, a path planning technique is used for indoor navigation. The path

planning technique is based on a layout plan of the indoor environment in the form of a graphic picture.

Furthermore, the server system sends relevant information of the environment and the merchandise to the client system. The client system maps next the information from the real world to the screen of the hand-held mobile device. And then the merchandise information and the navigation path are overlaid onto the real places or objects shown in the current image taken of the environment by the built-in camera of the mobile device. In other words, the device displays the guidance information in an AR way. As such, it is easy for the user to know about the surrounding environment and merchandise information.

1.4 Contributions

The major contributions of this study are listed in the following.

1. An AR-based indoor multi-user navigation system for merchandise shopping or similar activities using computer vision techniques is proposed to satisfy the demands of path guidance or merchandise browsing in indoor environments.
2. An image-based multi-user localization method by analyzing the images captured from the omni-cameras affixed on the ceiling is proposed to compensate for the insufficiency of the GPS positioning technique in the indoor environment.
3. An AR interface is proposed to provide users with augmented environment and merchandise information on acquired images.

1.5 Thesis Organization

The remainder of this thesis is organized as follows. In Chapter 2, the

configuration of the proposed system and the system process are introduced in detail. In Chapter 3, the proposed process for learning of an indoor environment, which includes the data used in the proposed system, is described in detail. In Chapter 4, the proposed user identification method by color image analysis using multi-color edge marks attached on top of the users' mobile devices is described. In Chapter 5, the proposed multi-user localization method for indoor environments and the proposed method for detecting the user's viewing orientation are described. In Chapter 6, the proposed AR process for guidance of merchandise shopping and merchandise recognition is presented. In Chapter 7, the proposed AR technique, a method to conduct the perspective transformation for information displays on each user's mobile device, and the adopted technique for rendering augmentations on real images are described. In Chapter 8, some experimental results to show the feasibility of the proposed techniques for indoor navigation for merchandise shopping or similar activities are presented. At last, conclusions and some suggestions for future works are included in Chapter 9.

Chapter 2
System Design and Processes

2.1 Ideas of Proposed System

2.1.1 Application Environment and Possible Scenarios

People usually need a navigation system to help guiding them when they visit an unfamiliar place. In this study, we propose an indoor AR-based multi-user navigation system for merchandise shopping or other similar activities. It can be applied in indoor environments like shopping malls, supermarkets, grocery stores, etc. It is hoped that a more intuitive mall or market navigation system can be designed to guide customers to get to the locations of desired merchandise items.

When applying the system proposed in this study to real world applications, a customer uses the mobile device to navigate a shopping mall in the following way. First, the customer browses the nearby place information by the AR function on mobile device screen. Next, if he/she cannot find the place desired to visit, he/she can search it in a database as well using the proposed system. Then, the path will be displayed on the screen to guide the customer to the destination. Also, the customer can search the merchandise he/she desires, and the system will also guide him/her to the shelf on which the item is placed. Finally, when the customer is in the front of the shelf, he/she can view the information, such as the brand and the price, of the merchandise on the mobile device screen. In short, the proposed AR-based navigation

system provides a more convenient and easy-to-use interface for merchandise shopping and similar activities.

2.1.2 Main Techniques for Indoor Localization and Visual Search

By the indoor localization technique, the proposed system analyzes the omni-images captured from the fish-eye cameras affixed on the ceiling, and then finds the users' foot points in each omni-image. When the users' foot points are obtained, their coordinates in the *fisheye-camera image coordinate system* (FICS) are transformed into those in the *global coordinate system* (GCS), to get the actual positions of the users in the indoor environment.

Next, the users' orientations must be detected. Based on the Hsieh and Tsai method [17], three techniques may be used to compute the users' orientations. The first technique is to track each user's locations in consecutively acquired images, and to use the resulting motion vectors of the user's foot points to compute the user's orientation. But when the user is not walking, this method will not work because there is then no more moving vector for use. Therefore, the second technique is to utilize the orientation sensor installed in the user's device. The orientation sensor measures the azimuth angle of the device by detecting changes and disturbances in the magnetic field in the surrounding environment. However, according to experimental experiences, the azimuth values detected are found not stable enough for the application due to indoor magnetic interferences from various sources. Hence, the third technique is to improve the stability of detected orientations is to attach a *color edge mark* on the top edge of the user device, and detect this linear mark appearing in the omni-image to compute a more accurate orientation of the user at each visited spot.

In this study, for the purpose of identifying the multiple users in the environment, we propose a new user localization method which detects a *multi-color edge mark* attached on top of the mobile device. With the different combinations of the multiple colors, we have more choices than a single color. That is, we detect the users holding different-colored client devices from acquired images, and identify them by a color classification scheme.

In addition, in order to provide a more convenient interface for merchandise shopping, the proposed system analyzes the images captured from the mobile-device camera, and matches them with the merchandise patterns using SURFs. Also, the field-of-view of the mobile-device camera must be estimated to get a perspective projection matrix for use later. Furthermore, to improving the performance, the server-side system is designed to match the images with the merchandise patterns at detected heights rather than all the patterns at all heights. If the matching succeeds, the server-side system will then calculate the position of the matched merchandise, and sends the corresponding information to the client-side system.

When the client-side system receives the navigation information sent from the server, the system will display the information on the device screen. With the perspective projection matrix described above, the 3D points of the navigation information can be transformed into the 2D screen plane. Then, the navigation information can be overlaid onto the real places or objects in the image taken of the current scene, and the user can so understand the surrounding environment and the merchandise information easily, achieving the major goal of AR-based indoor environment guidance of this study.

2.2 System Configuration

In this section, we will introduce the configuration of the proposed system. In Section 2.2.1, we will introduce the hardware configuration. The hardware of the proposed system includes omni-cameras which we use for localization detection of the multiple users, and the smart device which we use as the client-side device. In Section 2.2.2, we will introduce how to connect the hardware over a 4G/LTE network, and how it operates. Finally, we will describe in Section 2.2.3 the software development environment and the operating system we use both in the server-side system and in the client-side system.

2.2.1 Hardware Configuration

The hardware configuration for this study can be decomposed into three parts: the fish-eye cameras, mobile devices, and LTE router customized specially. The fish-eye camera we use is of the model of Axis 207MW, which is made by Axis Communications, and the original lens is replaced with a fisheye lens in this study to expand its field-of-view. The Axis 207MW camera has a dimension of 85×55×40mm (3.3”×2.2”×1.6”, not including the antenna), and a weight of 190g (0.42 lb., not including the power supply). Its appearance is shown in Figure 2.1. The maximum resolution of the images captured with it is up to 1280×1024 pixels. For performance efficiency, we use the resolution of 640×480 pixels in our system, and the frame rate is up to 15 fps. The cameras can be accessed through wireless networks (IEEE 802.11g/b), but for speed improvement, we access the cameras through the Ethernet.

We build an experimental environment in the Computer Vision Research Center at National Chiao Tung University by installing several fisheye cameras on the ceiling of the center (see Figure 2.2). The images captured from the cameras are analyzed by

a virtual machine (VM) in the cloud server to detect the user's location and orientation. The server sends the navigation information to the users' mobile devices so that the users can begin the navigation.

Figure 2.1 The camera used in the proposed system.

The mobile device we use in the experiment is an HTC Flyer tablet made by HTC Corporation. Its appearance is shown in Figure 2.3. The HTC Flyer has a dimension of 195×122×13.2mm (7.7"×4.8"×0.5") and a weight of 420g (0.93 lb). It has a screen size of 7 inches, a camera acquiring 5-megapixel images, and an e-compass that can detect the device orientation in a magnetic field, etc. For performance efficiency, we use the image resolution of 533×426 pixels in our system .The user uses the HTC Flyer as the client device, and connects it to the cloud server.

Because there is no commercial 4G/LTE network services in Taiwan now, in this study the 4G/LTE experimental network is provided by a project in cooperation with Broadband Mobile Lab (BML) at National Chiao Tung University. Mobile devices applicable in the 4G/LTE network are not available in the current market, so we use the mobile devices as described above and an LTE router customized specially as a

bridge between the 4G/LTE and Wi-Fi networks. The LTE router is a notebook with a 4G/LTE USB Dongle LU221 made by Quanta, Inc. Its appearance is shown in Figure 2.4. The notebook applies the technique of Internet Connection Sharing (ICS) of Windows to share the LTE connection with the WiFi local area network (LAN) by means of Network Address Translation (NAT). The notebook can be connected to the Evolved Packet Core (EPC) via the 4G/LTE network. Therefore, the LTE router can transmit data to a cloud server. On the other hand, the LTE router plays the role of a WiFi access point as well. As a result, a mobile device such as a tablet can access the LTE connection through the WiFi LAN.

Figure 2.2 The camera installed on the ceiling in the indoor environment.

2.2.2 Network Configuration

The configuration of the network used on this study is as shown in Figure 2.5, where the fish-eye cameras and the cloud server are connected through an Ethernet. The cloud server can access the images captured by the fish-eye cameras in a more reliable way through the Ethernet, and so one can make sure that the system always accesses correct and immediate images and messages.

Figure 2.3 The HTC flyer used as the client device in this study.

Figure 2.4 The Quanta 4G/LTE USB Dongle used in this study.

For the purpose of user mobility, we propose the use of the network configuration with 4G/ LTE in this study. LTE is a standard for wireless communication of high-speed data for mobile phones and data terminals. The client device we use is a mobile device, but there is no such mobile device which can connect through the 4G/LTE in Taiwan market now, as mentioned previously. So we use the Customer

Premise Equipment (CPE) --- an LTE router connected via the Wi-Fi network to access the service via the E-UTRAN in the 4G/LTE network. And a cloud server of a service provider is connected to an Evolved Packet Core (EPC) through the Internet. In this work, we apply a private network instead of the Internet to separate possible interfering traffic from the public network.

In short, the client device used in this study can access the cloud server and receive the navigation information through the 4G/LTE network in the environment. Figure 2.5 illustrates the network configuration described previously and used in this study.

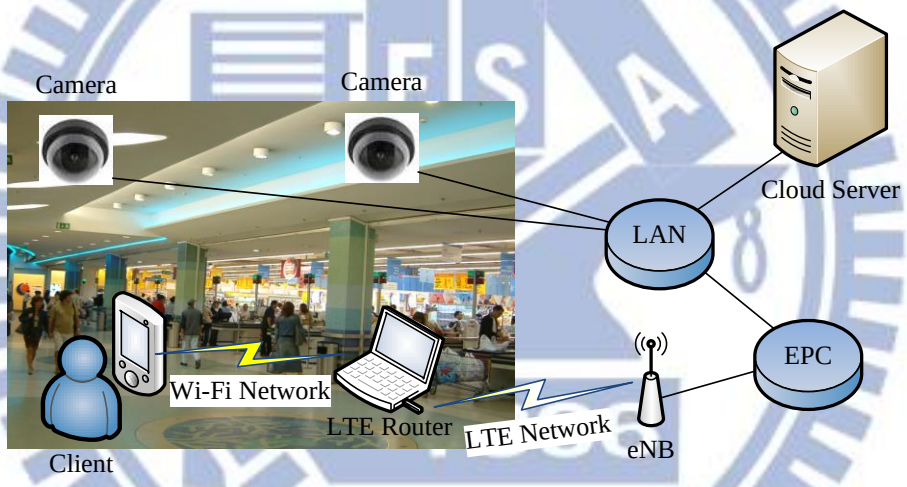


Figure 2.5 The network architecture of the proposed system.

2.2.3 Software Configuration

The server-side system operates on the Windows 7 Operating System in the VM of the cloud server, and the system is written using the C# programming language in the Microsoft Visual Studio 2010 development environment. The server-side system accesses the cameras by the *AXIS Media Control SDK (AMC SDK)*, which provides the application programming interface (API) for developers to access the camera images or control the cameras using C# and C++ programming languages. It is

provided by the manufacturer of the cameras, Axis Communications, Inc.

As to the client-side system, it is written in the Java programming language and operates on the Android 2.3.4 Operating System. The client-side system uses the *Qualcomm's Augmented Reality (QCAR)* platform, which provides many useful functions for AR developments on mobile devices. But in our system, we only use the QCAR platform to handle capturing of camera images. The work of rendering 3D augmented objects is conducted by the Android OpenGL API.

2.3 System Design

In this section, we describe in detail the design of the proposed network system. It is a client-server architecture as mentioned, which is composed of a server side and a client side. In Section 2.3.1, the server-side system used for conducting complicated works with heavy computations running on a VM in the Cloud server will be introduced. In Section 2.3.2, the client-side system running on a user's smart device, which obtains navigation information from the server-side system and displays it on the screen of the device, will be introduced. Finally, in Section 2.3.3, we will introduce the cooperation between the client and server sides.

2.3.1 Server-side System

The server-side system runs on a VM of the cloud server as mentioned, and it is connected to the fish-eye cameras on the ceiling through the Ethernet. In the learning stage, we build an environment map, which includes environment and merchandise information such as the locations and titles of the target places, the camera locations, the feature, price, and brand of each merchandise item, and so on. In the navigation stage, the server accesses the omni-images captured by the fish-eye cameras and the

image acquired from the mobile-device camera. The server analyzes the omni-images to detect the users' locations and orientations at each visited spot and identify them by a color classification scheme. After the server detects the multiple users via the images acquired by the cameras, it sends the corresponding users' locations, orientations, and the information of nearby places to the users' client-side systems. Meanwhile, the server extracts the features of the images acquired by each mobile-device camera and matches them with the features of the merchandise patterns by an SURF algorithm [14]. If matching with the merchandise pattern seen by a user is successful, the server will send the information of the merchandise to the user's client-side system. All of such information will be updated when the user moves. When the user wants to reach a certain destination, the server will receive a request from the client, and then plan a path from the user's location to the destination, and send a set of intermediate points of the path to the user's client-side system to display.

As a whole, the server is designed mainly for conducting multiple users' localizations, path planning, and matching with the merchandise items, and these tasks are heavy computational works. Because the client-side system runs on the user's mobile device, which has lower power and inferior computational capabilities than the cloud server, conducting these heavy computational works on the server, as done in this study, can increase the computational performance and reduce the battery power usage of the client-side system.

2.3.2 Client-side System

The client-side system runs on each user's mobile device. It transmits the images captured by the mobile-device camera to the server-side system. Because the mobile device has lower power and inferior computational capabilities than a laptop or desktop computer, the client-side system set up on it must be assigned as few works

as possible to reduce the power consumption and increase the computational performance. Therefore, most tasks of the client-side system are just information displays, such as view projection, display rendering, and creation of the navigation path's geometric shape (arrows, thick line segments, etc.).

When a user enters the environment, the user's client-side system is connected to the cloud server through a 4G/LTE network, and sends the images captured by the camera to the cloud server; meanwhile, it receives relevant information from the server. Then, the client-side system displays the information on the screen of the user's mobile device.

2.3.3 Cooperation between Client and Server Sides

The server and client side systems are described in Sections 2.3.1 and 2.3.2. Here we describe the cooperation between the client-side and server-side systems in more detail. An illustration of the cooperation between the two systems is shown in Figure 2.6.

When the client is connected to the server, the server detects the multiple users' locations and orientations, analyzes the images received from the client, and sends the messages, such as location coordinates, the orientation vectors, the nearby environment information, and the matched merchandise information, to the user. The information is updated continuously to make sure that the user can receive correct and immediate messages.

When the user wants to reach a certain destination, the client-side system will send a request, which includes the name of a desired destination or the name of a desired product, to the server. After server receives the request, it plans a path starting from the user's location and ending at the destination, and sends a set of intermediate points of the path to the client.

Finally, the client displays all of information described above in an AR way.

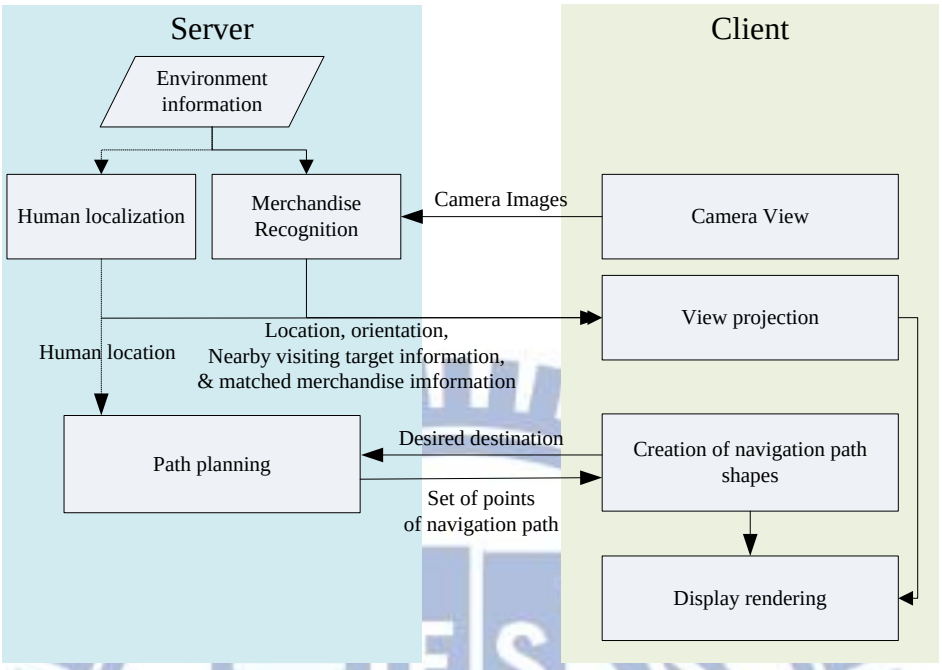


Figure 2.6 Cooperation between client and server sides.

2.4 System Processes

2.4.1 Learning Process

In the learning stage of the proposed system, the goal is to establish an environment map, which includes information about the places available for visits, fish-eye cameras, merchandise patterns, magnetic fields, and obstacle orientations. The entire learning process is shown in Figure 2.7. Only a brief description of the process is given here. More details of it will be described in Chapter 3.

At first, we establish an environment map in the form of a floor plan drawing. The floor plan is drawn at a specific ratio relative to the actual size of the environment. After specifying the ratio, we compute the corresponding size in the unit of pixel. The use of this scaling ratio is necessary for the transformation between the FICS

(fisheye-camera image coordinate system) and the GCS (global coordinate system). Next, the target places for visits in the environment are specified on the environment map. Furthermore, we specify as well the installation information of the fisheye cameras, which includes the locations and heights of the cameras for use in computing the transformations between the FICS and the map coordinate system (MCS).

After the environment map is established, the learning processes is conducted, which can be decomposed into three phases: learning for path planning, learning for user localization and learning for merchandise recognition. The two phases of learning for path planning and learning for user localization are based on Hsieh and Tsai method [17]. The goal of learning for path planning is to "understand" the obstacles information. And in the user localization phase, the camera calibration process and magnetic field learning process are conducted. The camera calibration process is to map the points between different coordinate systems; and the magnetic field learning process is to establish an azimuth map, which keeps a record of four direction azimuth values for every sample location in the environment map. A more detailed description of the learning works for path planning and for user localization will be given in Section 3.3.

In the learning of merchandise recognition, we construct the merchandise patterns, including feature extraction and merchandise information. A more detailed description of the learning works for merchandise recognition will be given in Section 3.3.4.

After the above learning steps, the preparation works needed in the navigation stage of the proposed system process are completed. We will describe the works conducted in the navigation stage in the next section.

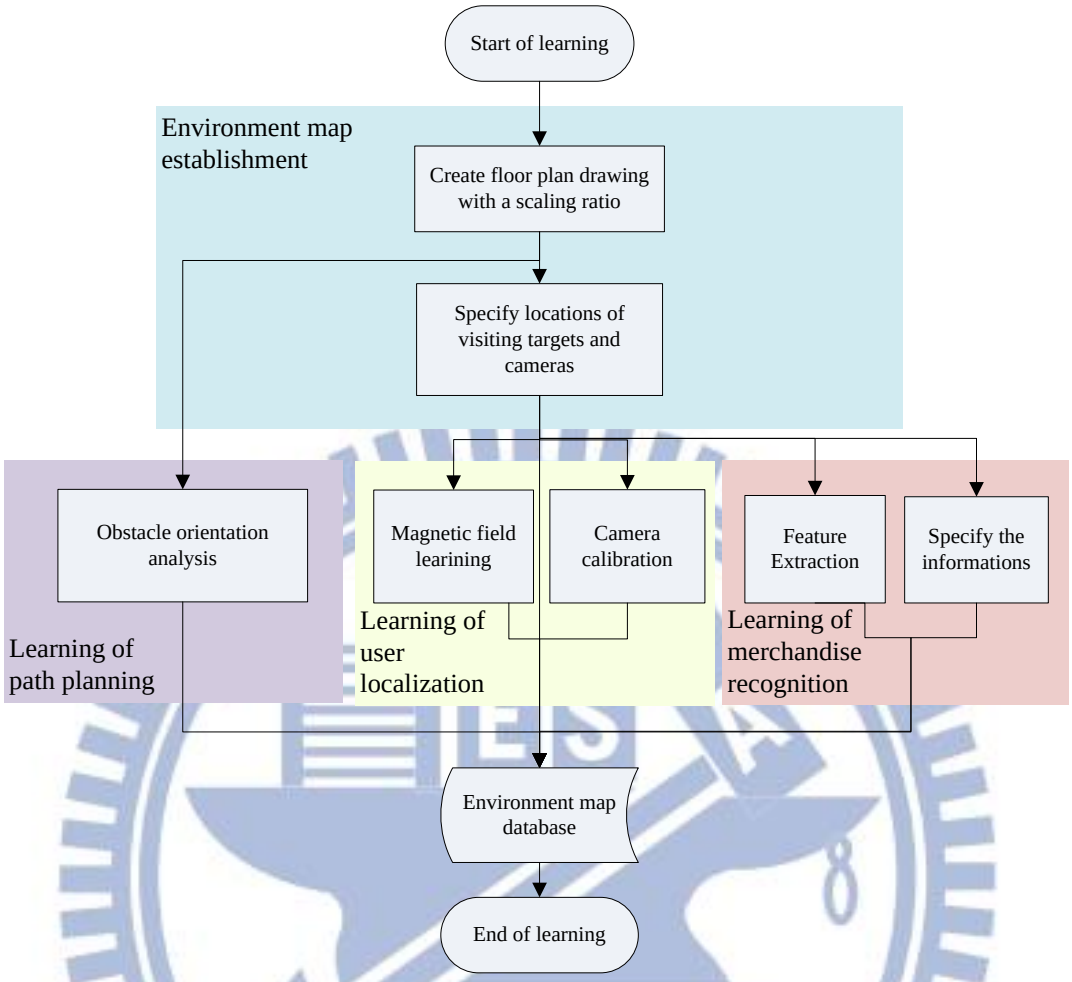


Figure 2.7 Learning Process of proposed system.

2.4.2 Navigation Process

In the navigation stage, the server analyzes the omni-images captured with the cameras continuously, and sends the environment information to the client. Meanwhile, the server analyzes the images received from the client, and sends merchandise information to the client. The client-side system displays the information on the screen of the user's mobile device. When the user wants to reach a certain destination or product, the server plans a path, and sends a set of intermediate points of the path to the client. The entire navigation process proposed in this study is shown in Figure 2.8.

At the server side, the first step is user location detection. The proposed user localization process transforms the detected user location from the FICS into the GCS using the camera information acquired in the learning stage. Then, the user's location is used in the steps of user tracking, user orientation detection, and multi-user identification. The step of user tracking aims to identify identical persons in consecutive video frames. In the user orientation detection step, the system detects every user's orientation by three techniques mentioned above. And in the multi-user identification step, the system detects and classifies the multi-color edges, which are on the tops of the client devices, to identify different users. Meanwhile, the system analyzes the images in the merchandise recognition stage, conducting the works of feature extraction and pattern matching. Finally, the server sends the information of the user's location and orientation, merchandise information, and the nearby target places to the user's mobile device (the client). Next, if the server receives a request of a client showing his/her desire to reach a certain destination or merchandise item, the server begins the path planning process; if not, the server continues to conduct user localization repetitively.

At the client side, when a client receives the navigation information mentioned above from the server, it begins to conduct the work of display rendering by "drawing" information of the target place, the desired merchandise item, and the navigation path on the device screen for the user to inspect. In order to map real-world objects onto the mobile device screen, the first step of the client is to set up a perspective projection by use of the location and orientation of each user detected by the server and the pitch angle of the user view obtained from the orientation sensor of the client device.

If the client receives a navigation path, it creates a geometric shape of the path; specifically, the client will transform the set of intermediate points of the path into an

arrow shape pointing to the destination. Finally, the client begins to draw the information and overlays the generated virtual objects onto the real image taken of the current scene to accomplish the display rendering task.

The above processes of both the server and client sides are run repeatedly until the client terminates the navigation system.

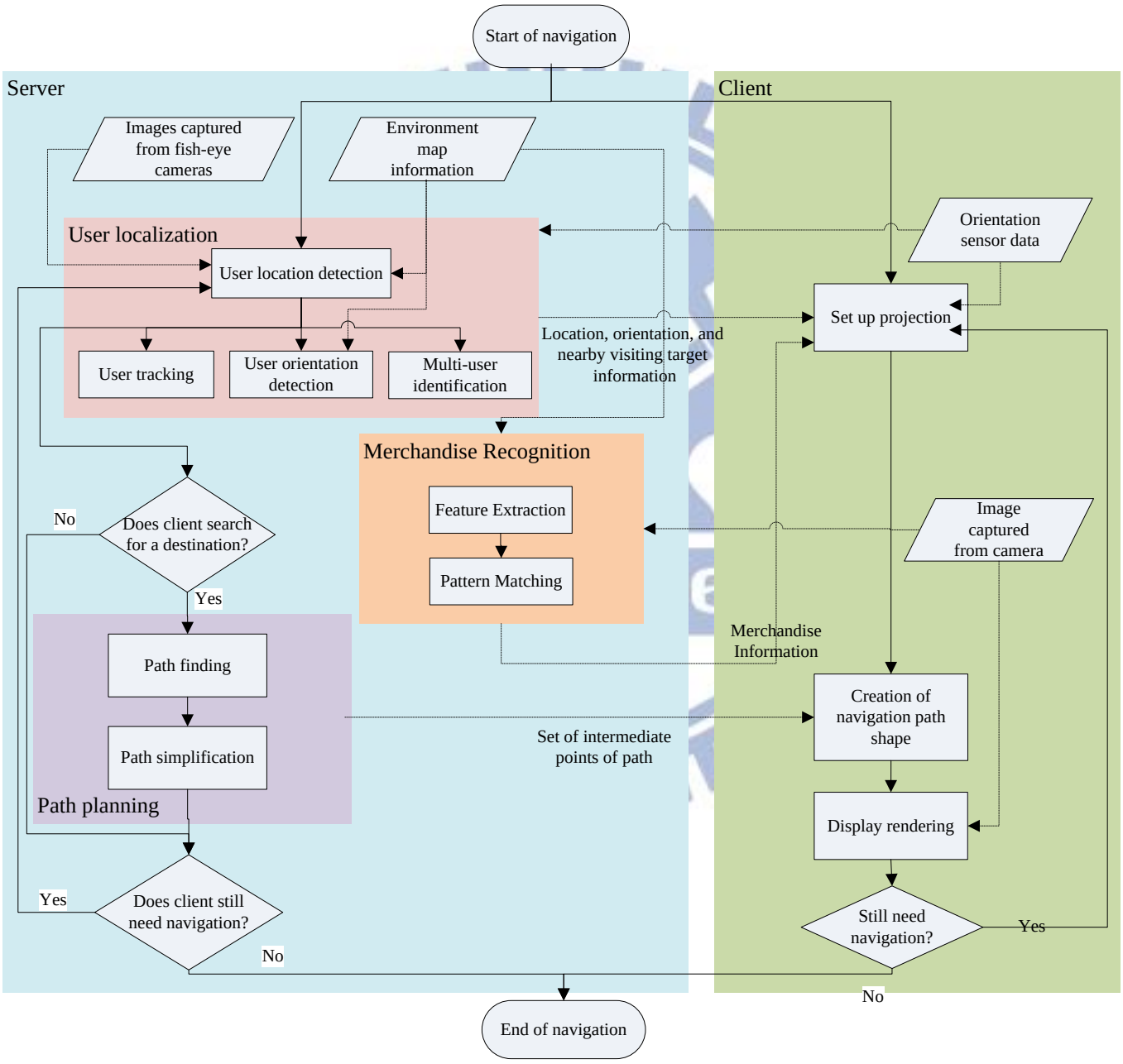


Figure 2.8 Navigation process of proposed system.

Chapter 3
Learning of Environments

3.1 Ideas of Proposed Environment Learning Techniques

In the learning stage of the proposed system, we must construct an *environment map*, which includes information about the cameras, target places for visits, merchandise items, magnetic fields, and obstacles. Then we can use such information in the processes conducted in the navigation stage, such as user localization, path planning, and merchandise recognition. A digital floor plan of the environment is used to create the environment map and to specify the locations of the cameras as well as the target places for visits in the map. A more detailed description of environment map construction will be introduced in Section 3.3.

After environment map construction, we conduct the learning processes for path planning and user localization as described in Hsieh and Tsai [17]. Specifically, a floor plan drawing of the environment is analyzed to detect obstacles. The resulting obstacle information is used for collision avoidance in the path planning process. Meanwhile, in the process for user localization, the magnetic field at every spot in the environment is "learned" and recorded, and the cameras at both the server and client sides are calibrated. The magnetic fields are used for user orientation detection by the orientation sensor installed on the client device, and the output of the orientation sensor is an azimuth value specifying the orientation of the hand-held client device. The calibration of the server-side fish-eye cameras is to obtain a space mapping between the GCS and the FICS. And the calibration of the client-side camera on the

mobile devices is to obtain a mapping from the locations of real-world objects onto the device screen. A more detailed description of these tasks will be introduced in Section 3.3.

In the learning process for merchandise recognition, we extract the feature points of the merchandise images and specify the positions, brands, and prices of the merchandise items. A more detailed description of the learning process for merchandise recognition will be introduced in Section 3.3.3.

3.2 Coordinate Systems Used in This Study

In this section, we introduce the coordinate systems adopted from the previous study of [17]. Then, we modify them to be described more clearly, fitting the application of this study. These coordinate systems may be used to describe the relations between the used mobile devices and the environment map. The four coordinate systems used in this study are as follows, and shown in Fig. 3.1.

(1) Map coordinate system (MCS): denoted as (M_x, M_y). The MCS is used to represent the environment map. The M_x-M_y plane coincides with the image plane of the floor plan. The origin is at the left-top position of the image plane.

(2) Global coordinate system (GCS): denoted as (G_x, G_y, G_z). The G_x-G_y plane of this system coincides with the ground and the G_z coordinates "grow to the top of the building" in the real space. The origin is at the left-top point in the MCS.

(3) Fisheye-camera image coordinate system (FICS): denoted as (*Fu*, *Fv*). The *Fu*-*Fv* plane of this system coincides with the image plane of each fish-eye camera and the origin is at the left-top of the image plane.

(4) Mobile-device-camera image coordinate system (MICS): denoted as (*Mu*, *Mv*). The *Mu*-*Mv* plane of this system coincides with the image plane of the mobile device camera and the origin is at the left-top of the image plane.

(5) Camera coordinate system (CCS): denoted as (*x*, *y*, *z*). The CCS is used to represent the real-world space with respect to the mobile-device camera. The *x* coordinates "grow to the right of the camera," the *y* coordinates "grow to the top of the camera," and the *z* coordinates "grow to the back of the camera." The origin is at the lens center of the camera.

We use a floor plan drawing of the environment to establish the environment map, and the MCS is used for describing the geometry of the map, as mentioned previously. The relationship between the MCS and the GCS is illustrated in Figure 3.1. As shown in the figure, the origin of the MCS is mapped to a corresponding point in the real-world space. However, for the MCS the unit of pixel is used, so the global coordinates should be computed as the result of multiplying the MCS coordinates by a scaling factor of the floor plan in the following way:

$$G_x = sM_x; \quad G_y = sM_y \tag{3.1}$$

where *s* is the scaling factor found by experiments.

The transformations between the FICS and GCS, MICS and GCS will be introduced in Section 3.3.3.

3.3 Proposed Method for Construction of Environment Map

In this section, we introduce the method proposed to construct the environment map. The environment map is like a database, which contains the information that we

use in the navigation stage. In Section 3.3.1, the information included in the environment map will be introduced briefly. And the other learning processes for the proposed system will be described in more detail in the subsequent sections.

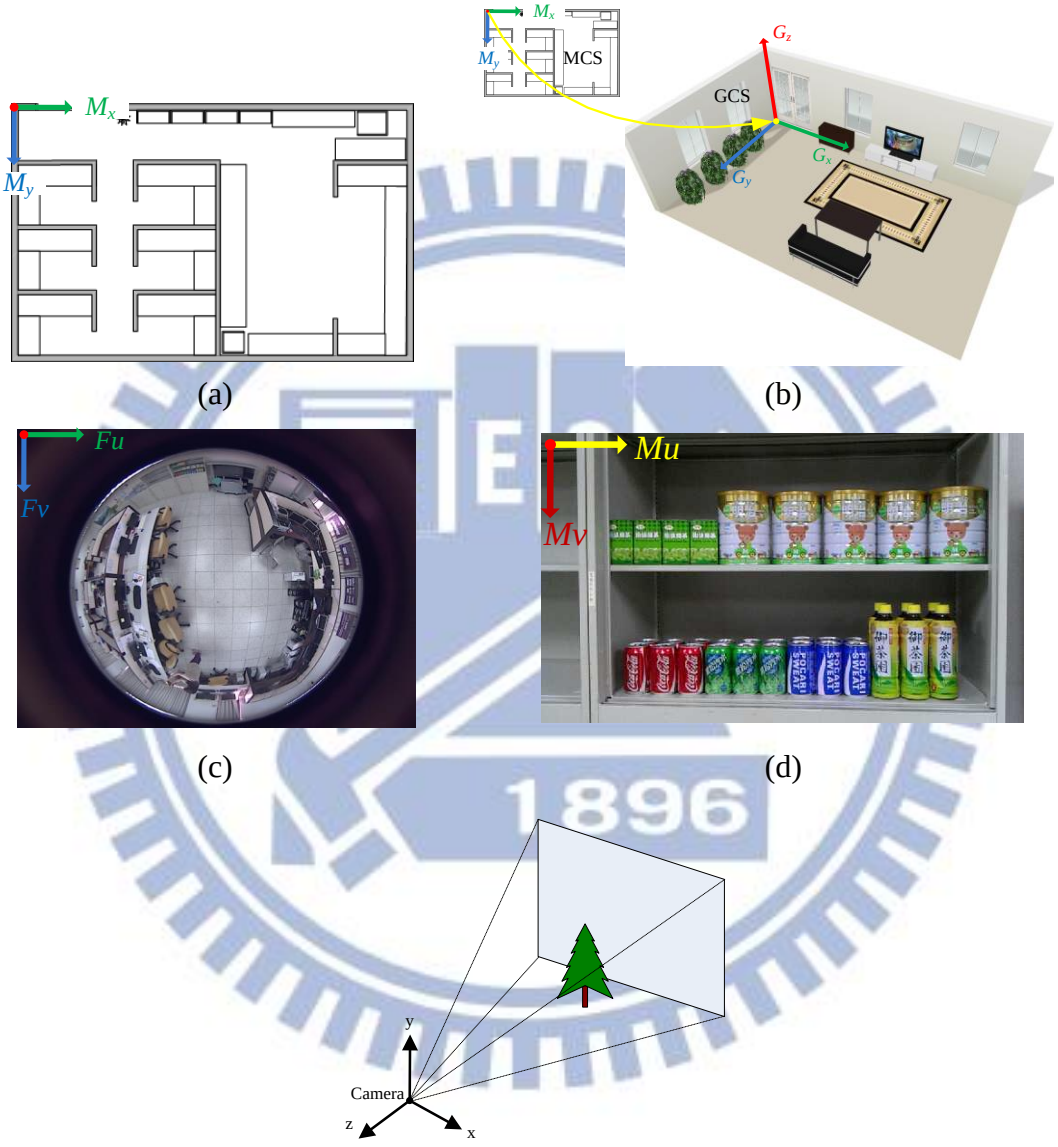


Figure 3.1 Four coordinate systems used in this study. (a) The MCS. (b) The relation between the MCS and the GCS. (c) The FICS. (d) The MICS. (e) The CCS.

3.3.1 Information of Environment Map

The information we use in the environment map includes the camera locations, information of target places for visits, obstacle information, magnetic field information, and merchandise information. We use a floor plan as the environment map, which is two-dimensional. But the information included in the map can be three-dimensional, that is, it includes height information.

Each camera location specifies the position of a fisheye camera and its height. The height of the camera is used for the transformation between the FICS and the GCS. Furthermore, we can place a lot of target places for visits and merchandise items in the environment. A target place in the real world is a spot of interest to the user in the environment. The information of a target place includes its name, its coordinates in the GCS, and its height and width. Also, the information of a merchandise item includes its name, brand, location, and price. The names of target places and merchandise items can be used for searching and displays. The location of a target place is specified by 3D coordinates in the environment map, including its height. The height and width of a target place represents the visible region of the target place in the real world.

The environment map includes the information of obstacles and magnetic fields in the environment as well. The obstacle information is used for path planning. It includes the regions and orientations of the obstacles in the environment. The obstacle region is used for collision detection, and the orientation is used for collision avoidance direction analysis. And the result of the magnetic-field learning process is an azimuth map, which can be used for user orientation detection.

3.3.2 Review of Learning Processes of Previous Study

In this study, we adopt the learning processes for path planning and user localization from the previous study of Hsieh and Tsai [17]. In the learning process for path planning, we should find the walkable region in the environment floor plan at first. For the experimental environment, the floor plan image we use is shown in Figure 3.2, which is created using the Microsoft Visio 2007 and exported as a bitmap.

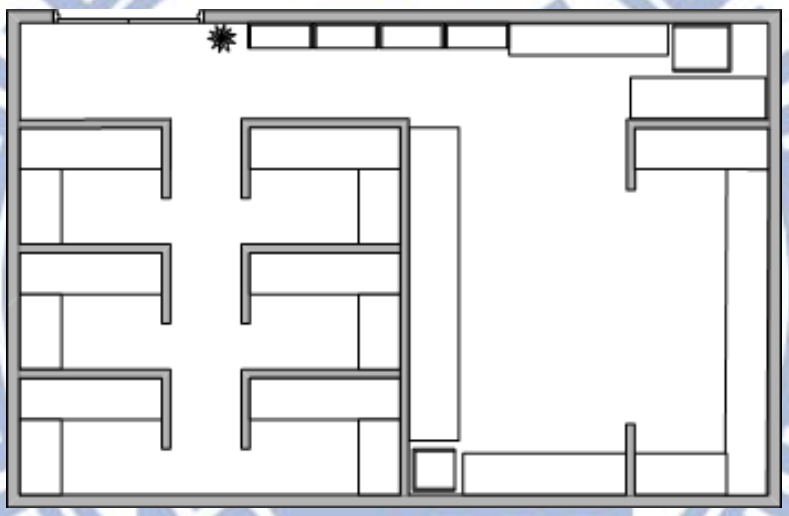


Figure 3.2 Floor plan image of the experimental environment map.

The technique of finding the walkable region is to find the largest connected component in the floor plan image I_f by an algorithm, namely, Algorithm 3.1 described in the following. Next, we have to analyze the walkable region image I_w to obtain an obstacle image I_{obs}. We can find the collision avoidance directions from the obstacle image I_{obs} and obtain an obstacle avoidance map by an algorithm, namely, Algorithm 3.2 described in the following as well. The avoidance directions are those vectors perpendicular to the obstacle orientation vector. Then, the obstacle avoidance

map may be used in the path planning process in the navigation stage, as described in Chapter 6.

Algorithm 3.1 Finding the walkable regions in the floor plan image.

Input: A floor plan image I_f, where the walkable region is the largest connected component and drawn with white pixels, and the obstacle regions and the walkable region are separated by non-white pixels.

Output: A binary walkable region image I_w, where the pixel values of the walkable region are specified by 1 and those of the obstacle regions by 0.

Steps

Step 1. Apply a threshold value t on I_f to get a temporary binary image I_{tmp}: if $I_f(x, y) > t$, then regard the pixel at (x, y) as white, and set $I_{tmp}(x, y)$ to 1; else, set $I_{tmp}(x, y)$ to 0.

Step 2. Find connected components in I_{tmp} using a conneted component labeling algorithm.

Step 3. Select the maximum connected component C_{max} from the result of the last step as the walkable region.

Step 4. For all (x, y) in I_w, set $I_w(x, y)$ to 1 if C_{max} contains the pixel at (x, y); else, set $I_w(x, y)$ to 0.

Algorithm 3.2 Finding collision avoidance directions.

Input: A walkable region image I_w, where the pixels of the walkable region are specified by 1 and those of the obstacle regions by 0.

Output: An obstacle avoidance map A.

Steps

Step 1. Get an obstacle image I_{obs} from the negative image of I_w.

Step 2. Dilate the obstacle image I_{obs} to expand the obstacle regions.

Step 3. Calculate the x and y derivatives of I_{obs} using the Sobel operator, resulting in two derivative maps D_x and D_y.

Step 4. Calculate the edge orientations and create an orientation map O by the following steps :

(1) set $O(x, y)$ to the angle between the vector $(D_x(x, y), D_y(x, y))$ and the vector (1, 0) if $D_y(x, y) \neq 0$ or $D_x(x, y) \neq 0$.

(2) set $O(x, y) = -1$, otherwise.

Step 5. Split O into small blocks, and for each block B_{ij} in O, construct a block orientation map O_b by the following steps:

(1) set $O_b(i, j) = m_{ij}$ if B_{ij} contains any non-negative value, where m_{ij} is the mean value of all non-negative values in B_{ij};

(2) set $O_b(i, j) = m'_{ij}$ if B_{ij} contains all negative values and the region of B_{ij} in O is all walkable, where m'_{ij} is the mean value of all non-negative values whose distances to the center of B_{ij} are smaller than a threshold d;

(3) set $O_b(i, j) = -1$, otherwise.

Step 6. Add $\pi/2$ to each element in O_b in the following way to get the obstacle avoidance map A:

$$A(i, j) = O_b(i, j) + \frac{\pi}{2}.$$

The learning process for user localization can be decomposed into two phases: magnetic-field learning and camera calibration. In the magnetic-field learning phase, we measure the azimuth values at several sample points in the environment and construct an azimuth map by repeating Algorithm 3.3, which then can be used for user

orientation detection. A more detailed description of user orientation detection using the azimuth map will be described in Chapter 5. And the camera calibration will be introduced in the next section.

Algorithm 3.3 Construction of an azimuth map for the experimental environment.

Input: A set *S* of sample point in the environment whose four major directions are set to be (0, 1), (1, 0), (–1, 0), and (0, –1).

Output: An azimuth map *A*.

Steps

Step 1. Take the client device, and go to the first sample point *S*0 at coordinates (*x*, *y*) in the environment map *A*.

Step 2. Face toward the direction (1, 0) in *A*, and measure the azimuth value *a*0.

Step 3. Face toward the direction (0, 1) in *A*, and measure the azimuth value *a*1.

Step 4. Face toward the direction (–1, 0) in *A*, and measure the azimuth value *a*2.

Step 5. Face toward the direction (0, –1) in *A*, and measure the azimuth value *a*3.

Step 6. Store the value set (x, y, a_0, a_1, a_2, a_3) in A.

Step 7. Go to the next sample point and repeat Steps 2 through 6 until reaching the last sample point.

3.3.3 Review of Camera Calibration of Previous Study

The camera calibration process includes two parts: fish-eye camera calibration and mobile-device camera calibration. The process of fish-eye camera calibration is based on a space-mapping scheme for obtaining the transformation between the GCS and the FICS. First, we construct a calibratttgion box with a *calibration coordinate*

system (CACS), which contains three coordinates (*x*, *y*, *z*) in the unit of cm. The calibration box and the CACS are shown in Figure 3.3. Accordingly, we can get a mapping table between the calibration points in the FICS and the real points on the *x*-*y* plane in the CACS by projecting the calibration points of the vertical calibration boards onto the *x*-*y* plane.

In order to perform the ground-point location mapping, at first, the mapped calibration points of the vertical boards of the calibration box are projected onto the x-y plane of the CACS as shown in Figure 3.4, where C is a calibration point on the calibration board with CACS coordinates (C_x, C_y, C_z), and C' is the projection point with CACS coordinates (C'_x, C'_y, 0). The relation between C and C' can be expressed by the following expression according to the principle of similar triangles:

$$\frac{H_c}{C_z} = \frac{C'_y}{C'_y - C_y} = \frac{C'_x}{C'_x - C_x}. \tag{3.1}$$

Rearranging the above expression, we can get the coordinates of C' as:

$$C'_x = \frac{H_c \times C_x}{H_c - C_z}; \quad C'_y = \frac{H_c \times C_y}{H_c - C_z}. \tag{3.2}$$

For the calibration points on the bottom calibration board, C' is identical to C.

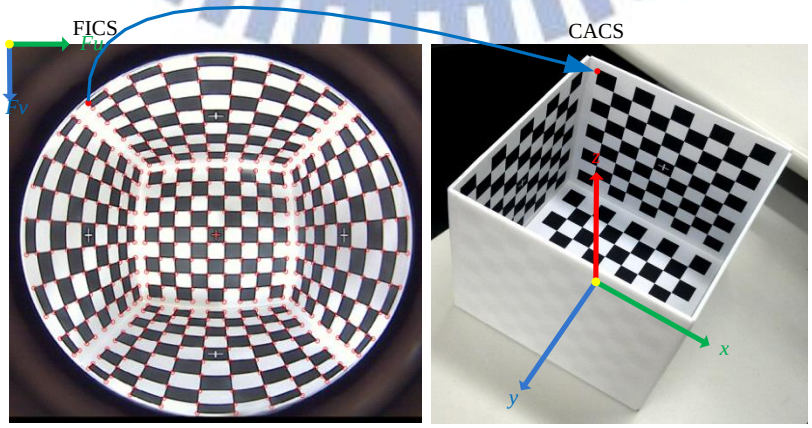


Figure 3.3 Mapping between the FICS and the CACS of a calibration point.

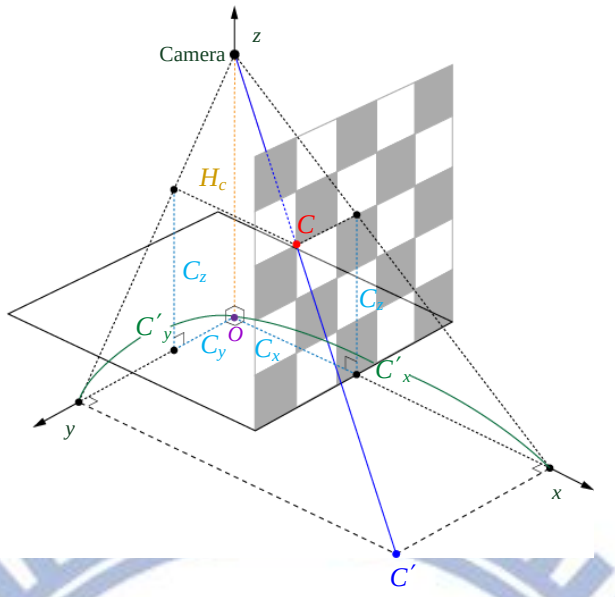


Figure 3.4 Projection of a point in CACS on the (x, y) plane, where H_c is the camera height, C is the calibration point on the calibration board, and C' is the projection point.

Then, we can use the mapping table between the CACS and the FICS to compute global coordinates in the GCS. As shown in Figure 3.5, a camera is affixed on the ceiling at a height of H, and G indicates the ground point of C'. By the principle of similar triangles, the distance d in the GCS can be computed by Equation 3.3. Accordingly, the coordinates (G_x, G_y) of G can be computed by the following equations:

$$d = \frac{H \times H_c}{d_c}; \tag{3.3}$$

$$\begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \frac{H \times H_c}{C'_x} \\ \frac{H \times H_c}{C'_y} \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \end{bmatrix} \tag{3.4}$$

where p_x and p_y are the coordinates specifying the location of the camera in the GCS, and θ is the orientation angle of the calibration points (see Figure 3.6).

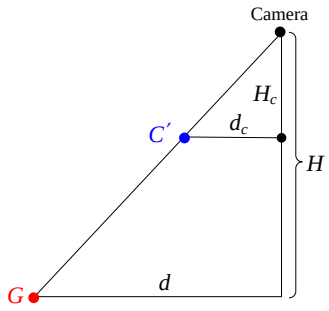


Figure 3.5 The projection of calibration point on the ground, where G is the projection point of C', and H is the height of the camera affixed on the ceiling.

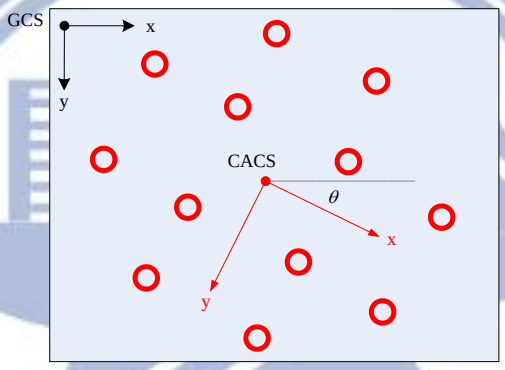


Figure 3.6 Angle between the GCS axis and the CACS axis. The red circles indicate the positions of calibration points.

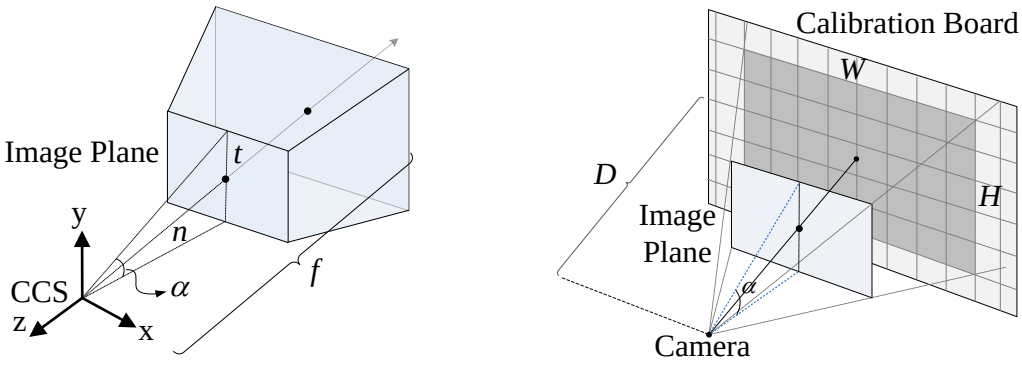
Besides fish-eye camera calibration, we calibrate the camera on the mobile device as well. A perspective camera model is used to represent the camera [18] by which a 3D space point with coordinates (p_x, p_y, p_z) in a symmetric view frustum as shown in Figure 3.7(a) in the camera coordinate system (CCS) can be transformed into an image point with coordinates (M_u, M_v) in the MICS by the following way:

$$\begin{pmatrix} p'_x \\ p'_y \\ p'_z \\ p'_w \end{pmatrix} = \begin{pmatrix} \frac{h}{w}\cot\frac{\alpha}{2} & 0 & 0 & 0 \\ 0 & \cot\frac{\alpha}{2} & 0 & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} \tag{3.5}$$

$$\begin{pmatrix} M_u \\ M_v \\ M_z \end{pmatrix} = \begin{pmatrix} w & 0 & 0 \\ 0 & h & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.5 & 0 & 0 & 0.5 \\ 0 & -0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \end{pmatrix} \begin{pmatrix} p'_x/p'_w \\ p'_y/p'_w \\ p'_z/p'_w \\ 1 \end{pmatrix} \tag{3.6}$$

where α is the angle of the field of view of the frustum in the *y* direction; *w* and *h* are the image width and height, respectively; M_z is the distance of the image plane with respect to the CCS; and *n* and *f* are respectively the nearest and farthest visible distances of the view frustum, which do not affect the mapped image coordinates in the MICS in Equation 3.6. Therefore, only one unknown variable α need be calibrated. For this, a simple method is proposed here. As shown in Figure 3.7(b), the camera is directed to face a calibration board with a grid-pattern region, and adjusted to "observe" the board in such a way that the region fills up the entire image scope. Accordingly, the maximum visible height *H* can be measured by counting the grids drawn on the calibration board. In addition, the distance *D* of the calibration board to the camera is measured manually in advance. Consequently, the angle α may be seen to satisfy the equality: $\tan(\alpha/2) = (H/2)/D$, leading to the following solution of α:

$$\alpha = 2(\tan^{-1}(H/2D)). \tag{3.7}$$



(a) (b)

Figure 3.7 Mobile-device camera calibration. (a) A view frustum. (b) Calibration of the field-of-view angle.

3.3.4 Learning of Merchandise Information

We construct the merchandise information for the purpose of providing AR-based guidance for merchandise shopping or other similar activities. The merchandise information includes the feature points of the merchandise image, and the name, location, brand, and price of the merchandise item. The name, brand, and price are used in displays. They are augmented on the screen of the mobile devices. And the location is used for searching for the merchandise item. A costumer can use the name of the merchandise item as a keyword for the search. The locations of the merchandise items are used in path planning.

In the feature extraction procedure, we detect the feature points from the merchandise image by an SURF algorithm [14]. The algorithm detects interesting points from the image and records the set of them as a descriptor. Then, we can specify the descriptor and other information for each merchandise item used in AR-based guidance for merchandise shopping or other similar activities. More details of the proposed merchandise recognition process will be described in Chapter 6.

Algorithm 3.4 Construction of merchandise information for the experimental

environment.

Input: An image *I* of a set *S* of selected merchandise items.

Output: An environment map *M*.

Steps

Step 1. Extract feature points from the merchandise image *I* using the SURF extraction algorithm described in [14] and get a descriptor *d* of *I*.

Step 2. Pick a merchandise item *s* and specify the name of it.

Step 3. Specify the descriptor *d* of *s*.

Step 4. Specify the name *n* of the location where *s* is placed.

Step 5. Specify the brand *b* and the price *p* of *s*.

Step 6. Specify the height *h* of the shelf on which *s* is placed.

Step 7. Store the value set (s, n, l, b, p, h) $_{\text{into}}$ M.

Step 8. Pick the next merchandise item in *S* and repeat Steps 2 through 7 until the last merchandise item in *S* has been processed.

3.4 Proposed Learning Algorithm for Environment Construction

In this section, we summarize the processes described in the previous sections as a single total process — the process of environment construction, as described in Algorithm 3.5 below.

Algorithm 3.5 Construction of the environment map.

Input: A floor plan image *I*.

Output: An environment map *M*.

Steps

Step 1. Affix fisheye cameras onto the ceiling at proper locations in the environment.

Step 2. Create an environment map *M* by use of the floor plan image *I*, and specify a scaling factor.

Step 3. Specify the locations of the cameras on *M*.

Step 4. Specify the locations and the names of selected target places on *M*.

Step 5. Find the walkable region in *I* by Algorithm 3.1.

Step 6. Find the obstacle regions and analyze avoidance directions in *I* by Algorithm 3.2.

Step 7. Construct an azimuth map by Algorithm 3.3.

Step 8. Specify the information of all the merchandise items on *M* by Algorithm 3.4.

3.5 Experimental Results

An environment map of our experimental environment obtained by applying Algorithm 3.5 is shown in Figure 3.8. The scaling factor of the map is taken to be 40 pixels/m. The environment map includes six target places for visits (shown as green regions), eight merchandise items (shown as blue labels), and two fisheye cameras (shown as blue circles). Two examples of images captured with the two fisheye cameras are shown in Figure 3.9.

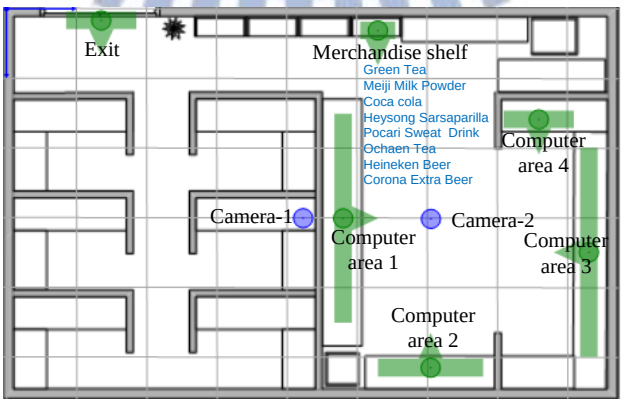
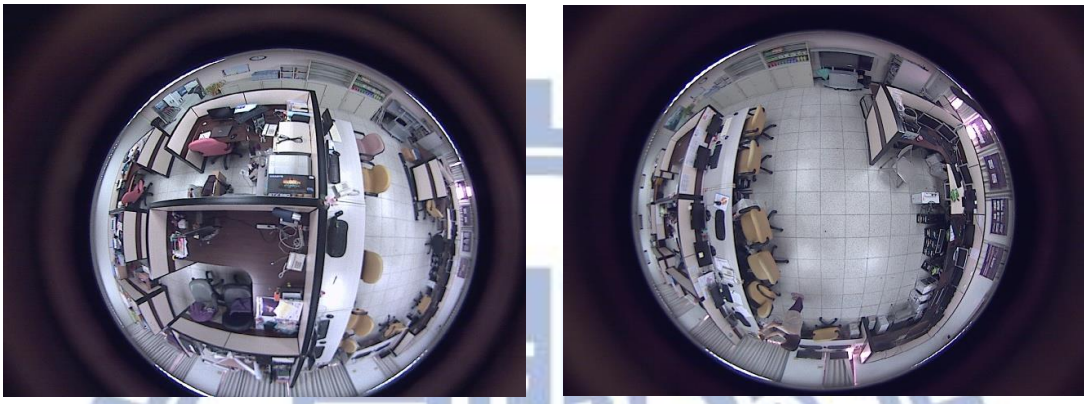


Figure 3.8 Environment map of the experimental environment, where target places are shown as green regions, and cameras as blue circles.

An obstacle avoidance map of the experimental map obtained by applying Algorithm is shown in Figure 3.10, in which the avoidance directions are shown as green arrows. Blocks without avoidance directions means that the blocks are obstacle regions or regions which are away enough from obstacles.



(a) (b)

Figure 3.9 Images captured from the two fisheye cameras of the experimental environment. (a) An image captured from the Camera-1 of the map shown in Figure 3.8 (b) An image captured from the Camera-2.

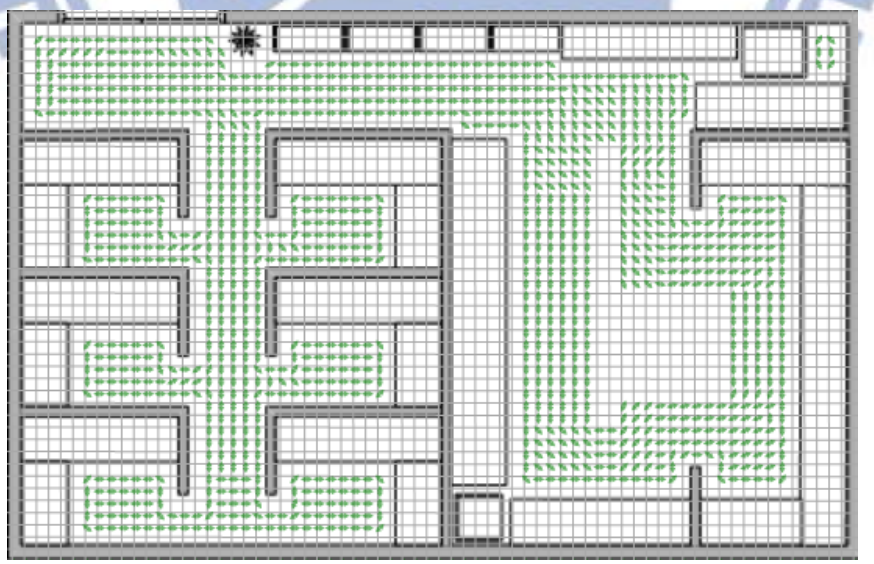


Figure 3.10 Obstacle avoidance map of the experimental environment.

Chapter 4
User Identification by Color Image Analysis Using Multicolor Edge Marks on Top of Client Devices

4.1 Ideas of Proposed User Identification Method

For the purpose of providing a *multi-user* AR-based navigation system, we must identify each of the multiple users in the environment. In this study, we propose a user identification method by color image analysis for the indoor environment. The method is described in this chapter.

We have built a vision-based infrastructure with fisheye cameras affixed on the ceiling. The server-side system can access the omni-images captured with the cameras, and conduct detections of multiple users the in indoor environment. The method we propose for user identification is to attach a *multicolor edge mark* on top of each client device (the mobile device) and detect it in each consecutive video frame which is an omni-image. We segment the multicolor edge mark from the omni-image and classify the edge mark according to its color pattern.

Sometimes we cannot segment out all the color segments of the mark because the multicolor edge mark may be blocked by the user's body. Therefore, we propose also a technique for classification error reduction. In this technique, three schemes are used to detect and track multicolor edge marks. A more detailed description of these

schemes will be described in Section 4.2.3.

4.2 Proposed User Identification Method by Color Image Analysis

4.2.1 Multicolor Edge Mark Detection

Here we describe the proposed method to identify multiple users by the multicolor edge mark on the top of each client device. The colors of the multicolor edge mark have high saturation and high lightness, so they appear to be very prominent in the acquired omni-image, and can be segmented out easily from the image. For example, as shown in Figure 4.1, we can see the yellow-green-pink edge mark very clearly in the omni-image.

Figure 4.1 The clear multicolor edge mark (The yellow-green-pink strip) in the acquired omni-image.

In order to separate the multicolor edge mark from the rest part of an omni-image, at first we convert the color space of the omni-image from the RGB space to the HSV one. The HSV color model assigns three color components into a pixel, which

respectively are *hue*, *saturation*, and *value*. The *hue* component may be described with the words we normally think of colors: red, blue, green, etc. The *saturation* refers to the dominance of hue in the color. The *value* indicates the lightness of the color. By the use of the HSV color model, we can separate the multicolor edge mark from the omni-image more easily, because the colors of the multicolor edge mark have high saturation and high lightness. In addition, we assume that there are three colors on multicolor edge marks. We detect and classify them to obtain the identification number for each user. The classification scheme will be introduced in the next section. In addition, the multicolor edge mark becomes a strip shape in the omni-image, so we can use it to detect as well the orientation of the users (i.e., the direction to which the user is facing). The proposed scheme for detection of the user's orientation will be described in Chapter 5.

The following algorithm describes the process to detect the color regions of multicolor edge marks.

Algorithm 4.1. Detection of the multicolor edge mark on top of the client device.

Input: an omni-image I and the foreground region R of a user.

Output: the three middle points P_1, P_2, and P_3 of the color regions on the multicolor edge mark, and an approximating line L for the edge mark.

Steps.

Step 1. Create an image I_{hsv} by converting the color space of I from the RGB model to the HSV one by the following equations:

$$V = \max(R, G, B);$$

$$S = \begin{cases} \dfrac{V - \min(R, G, B)}{V} & \text{if } V \neq 0; \\ 0 & \text{otherwise;} \end{cases}$$

$$H = \begin{cases} \dfrac{(G-B)\times 60}{S} & \text{if } V = R; \\ 180 + \dfrac{(B-R)\times 60}{S} & \text{if } V = G; \\ 240 + \dfrac{(R-G)\times 60}{S} & \text{if } V = B. \end{cases}$$

If $H < 0$, then increment H by 360° to make H positive.

Step 2. Create a binary image $I_{c(i)}$ and for all (u, v) in $I_{c(i)}$, and assign values to $I_{c(i)}(u,v)$ for i = 1, 2, 3 (meaning the three colors on the mark) by the following steps:

(1) set $I_{c(i)}(u,v) = 1$ if the value of $I_{hsv}(u,v)$, (h, s, v), is between two threshold value sets $(H_{min(i)}, S_{min(i)}, V_{min(i)})$ and $(H_{max(i)}, S_{max(i)}, V_{max(i)})$;

(2) set $I_{c(i)}(u,v) = 0$, otherwise.

Step 3. Find the connected components C_i in region R of $I_{c(i)}$: if there is no connected component in region R, set the middle points P_i to be *non-usable*.

Step 4. Find the bounding box B_i of C_i.

Step 5. Find the middle points P_i of B_i.

Step 6. Create a binary image I_c, and for all (u, v) in I_c, assign values to $I_c(u,v)$ by the following steps:

(1) set $I_c(u,v) = 1$ if $I_{c(1)}(u,v) = 1$ and $I_{c(2)}(u,v) = 1$ and $I_{c(3)}(u,v) = 1$;

(2) set $I_c(u,v) = 0$, otherwise.

Step 7. Find the connected components C in region R of I_c.

Step 8. Find the bounding box B of C.

Step 9. Apply a line approximation algorithm to the pixels in C resulting in an approximating line L.

Step 10. Take P_1, P_2, P_3 and L as output.

In the above algorithm, we apply six threshold values to extract the three different color regions of the multicolor edge mark in Step 2, and extract the entire mark in Step 6. An example of the result is shown in Figure 4.2. After we detect the middle points of the extracted color regions, we can classify them to obtain the identification numbers for identification of multiple users, as described next.

(a) (b)

Figure 4.2 Detection of the color regions on the color edge mark on top of the mobile device. (a) The three color regions segmented from the omni-image. (b) The three bounding boxes (in red) detected in (a).

4.2.2 Multicolor Edge Mark Classification

We detect the color regions by applying Algorithm 4.1 to each omni-image taken by a fisheye camera on the ceiling and obtain the three middle points of the color regions. The combination of the three colors is then mapped to different user identification numbers by a pattern classification scheme proposed in this study, as described now. First, we construct a mapping table between the detected color regions and the user identification numbers. To guarantee a correct performance of the

proposed classifier, we have to omit some combinations of the three colors. In more detail, in theory we have $3^3 = 27$ user identification numbers. But we have to consider two "ambiguous" cases on the edge mark here.

(1) The three colors on an edge mark are *left-right symmetric* to those on another (e.g., yellow-green-pink and pink-green-yellow) — If the colors of two edge marks are of this case, then when they appear in the image *simultaneously* and *face symmetrically* to each other, then they cannot be differentiated as two different marks.

(2) More than one identical color is *neighboring* in the mark (e.g., yellow-yellow-pink) — Neighboring colors on the edge mark are extracted by image processing to be just a single region in this study, so the edge mark will be regarded to consist of less than three colors. For example, the combination of seemingly three colors, yellow-yellow-pink, will be considered to be just a combination of two colors, yellow-pink, only.

Therefore, only nine *effective* combinations of colors are left, instead of 27, for use in user identification when three colors are used on the edge mark, as can be figured out and seen from the mapping table shown in Table. 4.1. Of course, we may use a larger number of colors on the edge mark to increase the number of effective combinations for identifying more persons. For example, when four distinct colors are used, then it can be figured out that twenty-two effective combinations of them can be used for user identification.

The following algorithm describes the process to classify the color regions of the multicolor edge mark.

Algorithm 4.2. Classification of the multicolor edge mark on top of the client device.

Table 4.1 Mapping table of combinations of three colors and identification numbers.

ID	1	2	3	4	5	6	7	8	9	-
	GGG	YYY	PPP	YYG	PPG	YYP	PGY	PYG	YPG	PYP
				YGG	PGG	YPP	YGP	GYP	GPY	PGP
				GYY	GPP	PPY				GYG
				GGY	GGP	PYY				GPG
										YGY
										YPY

Input: three middle points P_1, P_2, P_3 of the color regions of the multicolor edge mark, an approximating line L of the mark, and the identification number table T shown in Table 4.1.

Output: a user identification number n.

Steps

Step 1. Find the normal $\overrightarrow{d_l}$ of the approximating line L.

Step 2. Find the direction θ of $\overrightarrow{d_l}$ if L is not vertical.

Step 3. If L is not vertical, then conduct the following two steps:

(1) for i = 1, 2, 3, perform the following operation:

if P_i is not non-usable, then rotate P_i to P_i' through the angle of θ by the following equation:

$$\begin{bmatrix} P_{ix}' \\ P_{iy}' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} p_{ix} \\ p_{iy} \end{bmatrix};$$

else, do nothing;

(2) sort all of those P_{ix}' whose corresponding original P_i are usable; else, sort P_{iy} directly for i = 1, 2, 3.

Step 4. Concatenate the pixels in an order corresponding to the sorting result to obtain a pixel sequence *r*.

Step 5. Map the sequence of colors of the pixels in *r* to a user identification number *n* according to Table 4.1 and take it as the output.

4.2.3 Technique for Classification Error Reduction

We introduced the proposed user identification technique in the previous sections. However, it has stability and precision problems which cause failures in identifying users. Therefore, we propose further a technique to reduce the errors of classification. We construct a record to remember the last five results of yielded identification numbers. If the last five identification numbers are all the same, meaning that the person in question has been stably identified already, then the newly-yielded identification number will be discarded.

In addition, sometimes we cannot detect all the three colors on a multicolor edge mark successfully because the multicolor edge mark may undesirably be blocked by the user's body. So, we give a priority sequence to multicolor edge marks as: the three-color edge mark, two-color edge mark, and one-color edge mark. This priority sequence will be used in correcting erroneous classification results. For example, if we detect three colors on a multi-color edge mark, which was classified as a two-color edge mark in previous image frames, and then it will be corrected to a three-color edge mark in the current frame according to the priority. The following algorithm describes the proposed process to reduce the error of classification.

Algorithm 4.3. Reduction of errors in classification of multicolor edge marks on

top of the client device.

Input: a sequence of user identification numbers n_1, n_2, n_3, …, n_n.

Output: an sequence of corrected user identification numbers n_1', n_2', n_3', …, n_n'.

Steps

Step 1. Put n_1, n_2, n_3, n_4, n_5 in a record R and set $n_i' = n_i$ for i = 1, 2, …, 5.

Step 2. For $i > 5$, if the following two qualities are satisfied:

(1) $n_{i-1} = n_{i-2} = \ldots = n_{i-5}$;

(2) $priority(n_{i-1}) > priority(n_i)$,

then, set $n_i' = n_{i-1}$ to keeping the detected identification number;

else, set $n_i' = n_{i-1}$ to correct the identification number.

Step 3. Put n_i' in R.

Step 4. Delete n_{i-5} from R if the number of elements in R is larger than five.

Step 5. Repeat Step 2 through Step 4 until the last user identification number has already processed.

4.3 Proposed Algorithm of User Identification

We introduced three techniques for user identification in the previous sections. In this section, we will describe how we integrate the three techniques to perform the user identification work more reliably.

Algorithm 4.4. Integrated determination of the user identification number.

Input: an omni-image I and the foreground region R of a user.

Output: the user's identification number n.

Steps

Step 1. Detect the middle points P_1, P_2, and P_3 of the color regions and the approximating line L of each multicolor edge mark M_i in omni-image I by Algorithm 4.1.

Step 2. Use the normal direction $\overrightarrow{d_l}$ of L, the middle points P_1, P_2, and P_3, and the identification number table T to classify the edge mark M_i to obtain a user identification number n by Algorithm 4.2.

Step 3. Reduce the error classifications and correct the identification number n into n' by Algorithm 4.3.

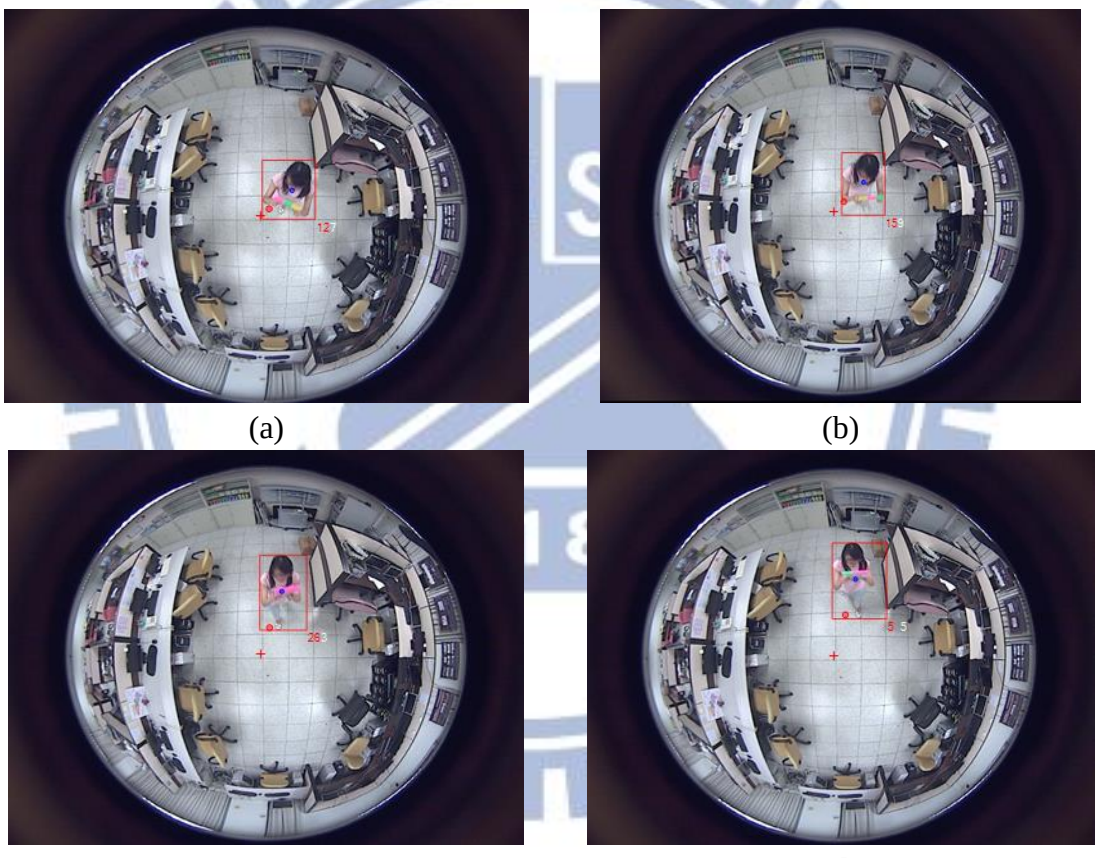
Step 4. Take n'as the output.

4.4 Experimental Results

In this section, we show experimental results of user identification by the use of the previously-proposed algorithms. Figure 4.3 shows some examples of the successful results of recognizing multicolor edge marks with four different identification numbers. In each case, the obtained user identification number is indicated at the right-bottom corner of the red rectangle enclosing the detected region of the user.

Figure 4.4 shows the results of classification error reduction, where Figures 4.4(a) through 4.4(d) are an image sequence obtained before the proposed classification error reduction technique is applied, from which we can see that the identification numbers obtained in the sequence are not stable with the non-identical results of 5-5-7-5. Figures 4.4(e) through 4.4(h) are an image sequence obtained after classification error reduction is carried out, which shows that the obtained numbers 7-7-7-7 become identical now.

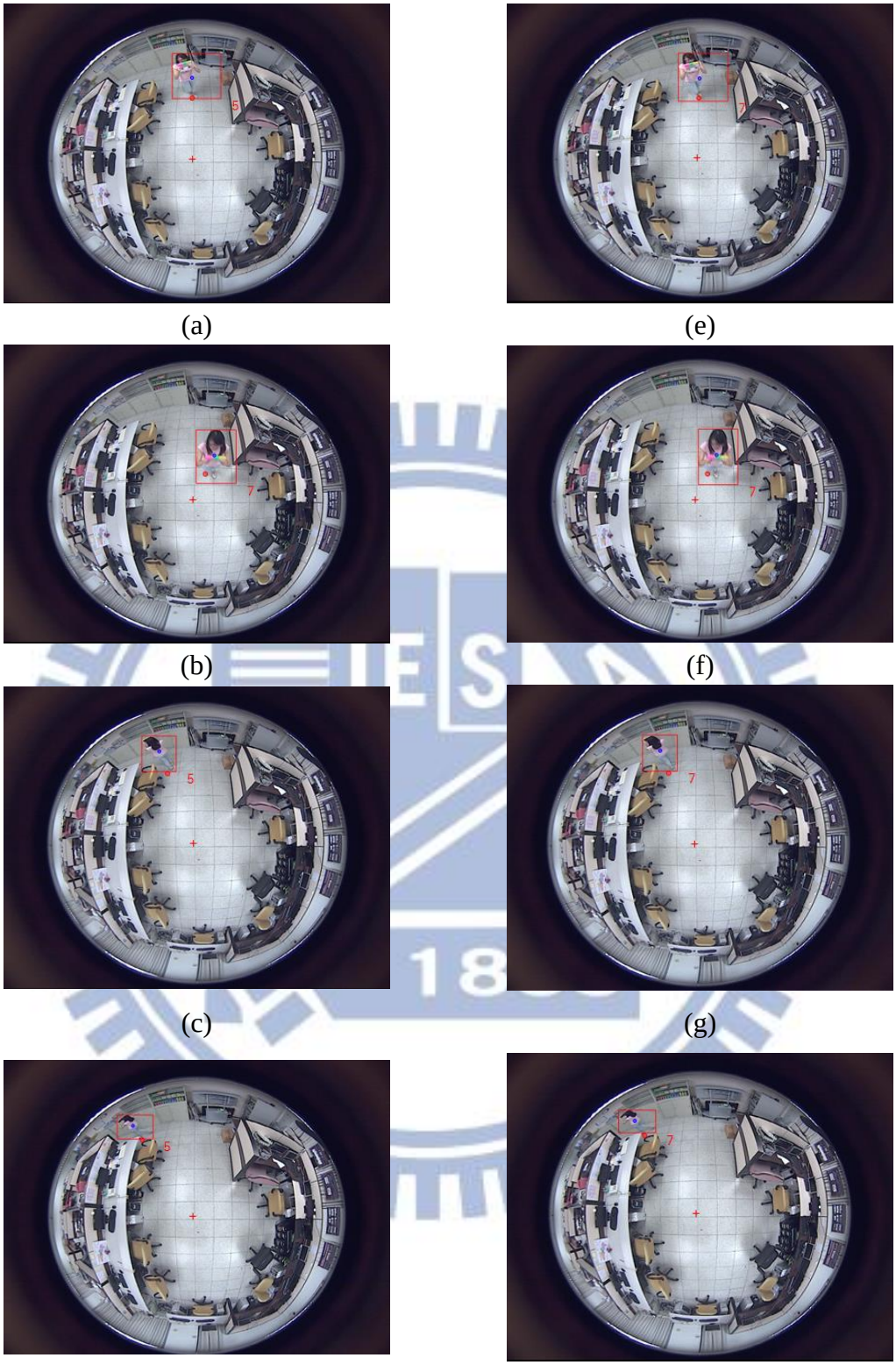
Figure 4.5 shows the results of applying classification error reduction to the case of two users being in the environment, where Figures 4.5(a) through 4.5(d) are an image sequence obtained before classification error reduction, from which we can see that the identification numbers obtained in the sequence are not stable with the results of 5-7-7-5 and 5-3-3. Figures 4.5(e) through 4.5(h) are an image sequence after classification error reduction, which shows that the obtained numbers 7-7-7-7 and 5-5-5 become identical now.



(a) (b)

(c) (d)

Figure 4.3 User identifications at different locations with different colors on edge marks. (a) Identification number 7. (b) Identification number 9. (c) Identification number 3. (d) Identification number 5.



(a) (e)

(b) (f)

(c) (g)

(d) (h)

Figure 4.4 Effect of user identification using classification error reduction for a single-user case. (a) to (d) An image sequence obtained before classification error reduction. (e) to (f) An image sequence obtained after classification error reduction.

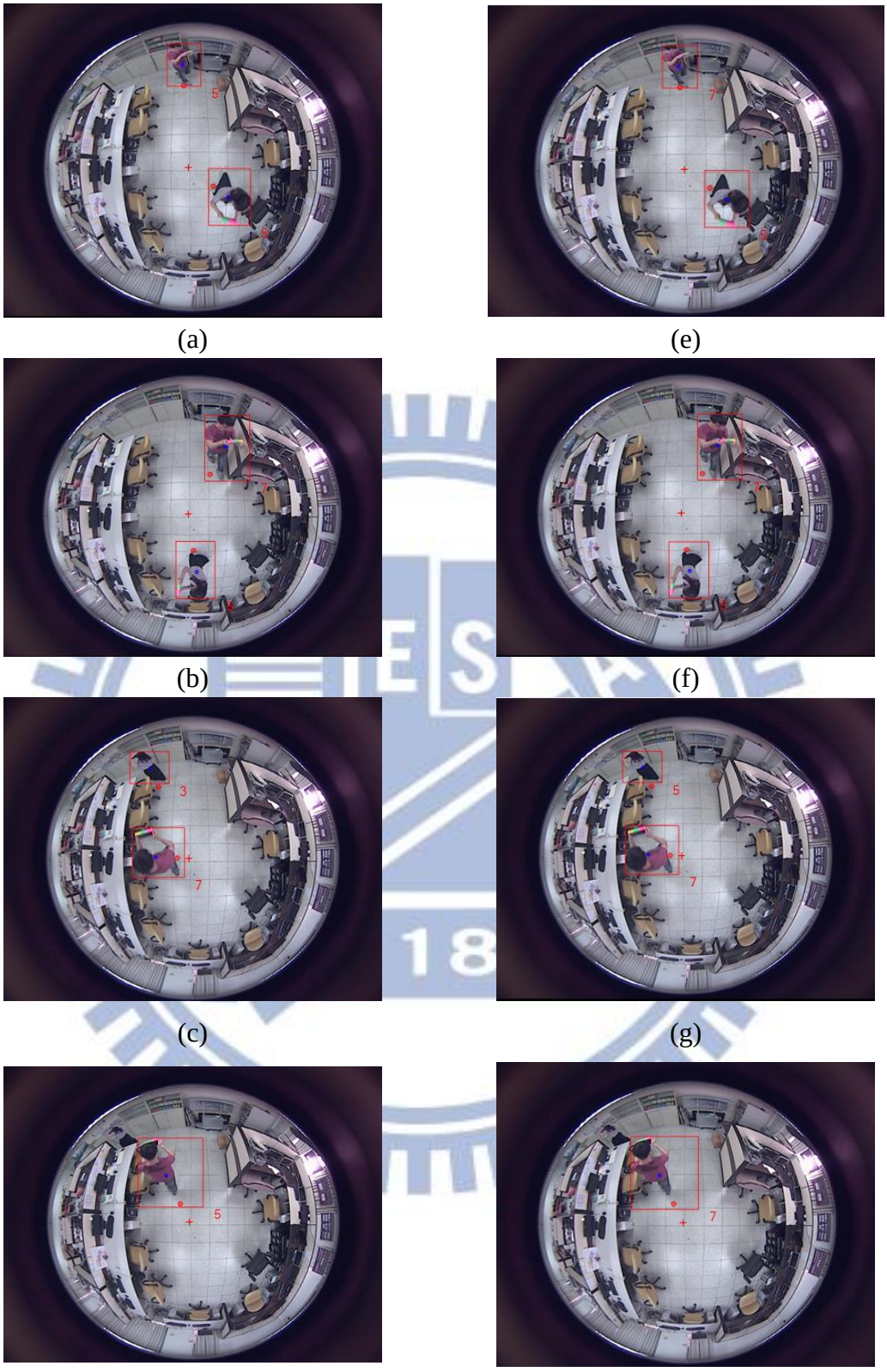


Figure 4.5 Effect of user identification results using classification error reduction for a two-user case. (a) to (d) An image sequence obtained before classification error reduction. (e) to (f) An image sequence obtained after classification error reduction.

Chapter 5
Multi-user Localization in Indoor Environments by Computer Vision Techniques

5.1 Review of a Previous Work and Idea of Proposed Method

In this study, we propose a multi-user localization method using image-based analysis techniques for AR-based guidance in indoor environments. We have built a vision-based infrastructure with fisheye cameras affixed on the ceiling. The server-side system can access the omni-images captured with the cameras, and conduct detections of both the users' locations and orientations. We integrate single-user localization techniques as described in Hsieh and Tsai [17] and the proposed multi-user identification technique to localize multiple users in indoor environments.

For multi-user location detection, we perform background/foreground separation to detect foreground images, and then apply connected component analysis to find the users' activity regions. Then, the users' foot points in the regions are analyzed and transformed into the GCS. A more detailed description of the proposed multi-user location detection scheme will be described in Section 5.2.

For user reviewing orientation detection, we use three different techniques integrally to obtain the viewing orientation of users. The first is the simplest way, which is to calculate user motions by use of the users' locations detected from

consecutive video frames. The second is to use the orientation sensor on the client mobile device to detect the users' orientation. The last is to attach multicolor edge marks on the mobile devices held by users, and then analyze acquired omni-images to detect the multicolor edge marks which are used to determine the orientation and identification number of the users. In addition, we use the frustum for user viewing detection to detect the height of viewing orientation. A more detailed description of the proposed scheme for user viewing orientation detection will be described in Section 5.3.

5.2 Multi-user Location Detection

5.2.1 Review of an Algorithm for Single-user Location Detection

In this study, we adopt a single-user location detection scheme from the previous study of Hsieh and Tsai [17]. To detect a user's position, at first the user's body part is extracted from the input image. For this, a *background image* is captured in the learning stage as a reference. Then, when the user enters the environment in the navigation stage, his/her body is found from the acquired fisheye-camera image by a process of *foreground region detection*, including background subtraction, thresholding, and region growing. And the user's foot point in the found body region is detected according to an optical property of the fisheye camera — in an image acquired with a downward-looking fisheye camera, a space line perpendicular to the ground appears as a radial line going through the image center. Accordingly, because the user is standing on the ground, the axis of his/her body will go through the image center. And so the user's foot point may be found to be the image point in the detected

body region *nearest* to the image center. Finally, the user's foot point in this region is transformed into the GCS. In this way, we can obtain the user location in the environment.

5.2.2 Proposed Technique for Multi-user Location Detection

Based on the single-user location detection scheme as described previously, we propose a technique for multi-user location detection in this section. The first step is background/foreground separation. As shown in Figure 5.1, we capture a background image before running the server-side system. When users enter the environment, they will be considered as parts of the foreground regions. Therefore, we can obtain the users' regions by finding the connected components in the foreground image. Algorithm 5.1 below illustrates the steps to obtain such connected components in an omni-image.

Algorithm 5.1 Finding foreground regions in an omni-image.

Input: An omni-image I captured from a fisheye camera, a background image B captured beforehand, and a pre-selected threshold value T_D.

Output: Foreground regions $R_1, R_2, \ldots, R_n$ in I.

Steps

Step 1. Subtract B from I to get a difference image D.

Step 2. Apply the threshold value T_D on D to get a foreground image F by the following steps:

(1) set $F(u,v)=1$, if $|D(u,v)|>T_D$;

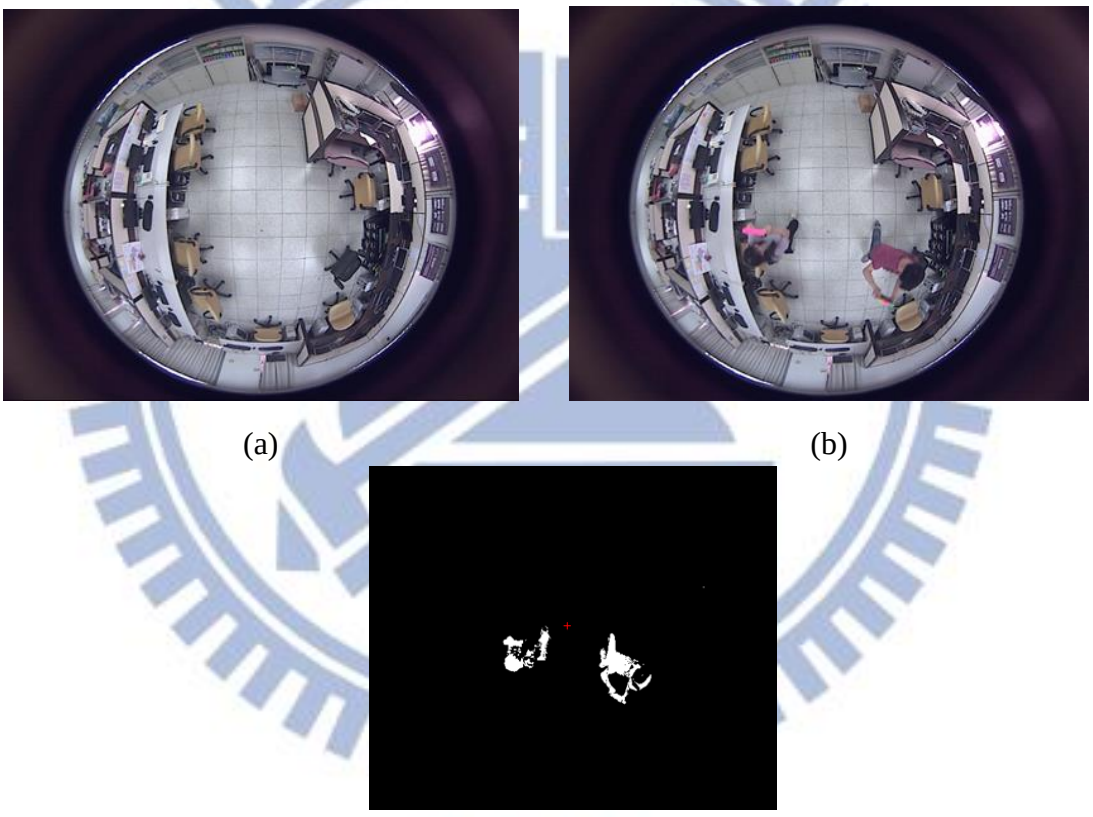
(2) set $F(u,v)=0$, otherwise,

where $D(u, v)$ denotes the value of a pixel on D.

Step 3. Apply the erosion operation to F to eliminate noise.

Step 4. Find connected components in F as the desired foreground regions R_1, R_2, …, R_n using a connected component labeling algorithm.

In Step 3, we reduce noise by applying the erosion operation on the foreground image. However, the erosion operation will also eliminate the details of the foreground image. Another way to reduce noise is to set a larger threshold value in Step 2.



(a) (b)

(c)

Figure 5.1 Background/foreground separation. (a) The background image. (b) The image of the environment with two users. (c) The foreground image resulting from by subtracting (a) from (b).

With the regions of the users extracted, we continue to find their foot points in the regions to determine the users' locations. As described in the previous section, we

assume that the users using the proposed indoor AR navigation system are standing on the ground all the time, and so the axis of each user's body are perpendicular to the ground, meaning that the axis of his/her body will go through the image center. So we can detect the users' foot points using the property, and then transform them to GCS. We can find the users' locations by the following algorithm using the output of Algorithm 5.1 as the input.

Algorithm 5.2 Computation of the multi-user locations.

Input: The foreground regions $R_1, R_2, \ldots, R_n$ of multiple users.

Output: The locations of the users in the GCS.

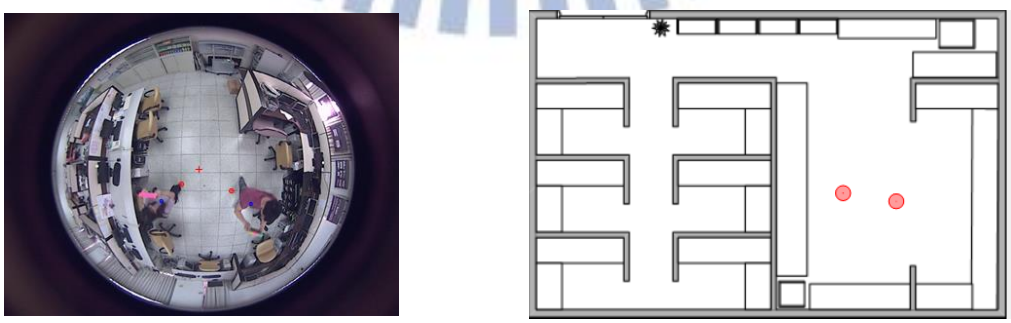
Steps

Step 1. Find the nearest point f_i to the omni-image center in R_i.

Step 2. Project f_i onto the line $\overline{CC_R}$ to obtain a projection point f_i', where C is the omni-image center and C_R is the center of the bounding box circumscribing R_i.

Step 3. Transform f_i' into the GCS as output.

The user location can be computed by the spatial transformation described in Section 3.4.3. An example of the results is shown in Figure 5.2.



(a) (b)

Figure 5.2 Detected foot points of two users (shown as red circle). (a) The original image captured from the camera (b) The foot points in MCS.

5.3 Detection of Users' Viewing Orientations

5.3.1 Review of an Algorithm for User Orientation Detection

The second stage in user localization is orientation determination. We adopt the user orientation detection scheme proposed by Hsieh and Tsai [17]. Three techniques, namely, *human motion estimation*, *magnetic-field sensing*, and *color edge mark detection*, are proposed in this study for uses in different situations.

In motion estimation, a user's position is detected in every navigation cycle as described previously in Section 5.2, resulting in a *position sequence*, which can be used to compute the user's orientation by motion estimation. However, the user's positions detected by image analysis are not always very accurate, and so the user's orientation computed by this way of motion analysis will not always be smooth. A solution to this problem is to average all the motion vectors obtained within a period of time to get a more stable result for use as the user's orientation. Yet, such an averaging operation will delay undesirably the orientation computation result when the user is turning. That is, the user's orientation changes quickly when he/she is turning, but the averaging operation will cause the computed orientation to change slower. Therefore, we use a *turning flag* to determine whether a user is turning in the proposed user orientation determination scheme. In this way, the result of motion vector averaging will be changing more quickly in time to reflect the user's turning speed.

As to magnetic-field sensing, the magnetometer on the mobile device can be used to measure the azimuth angle of the device by detecting the changes and disturbances

of magnetic fields. In the learning stage, we have established an azimuth map in which the magnetometer reading of each of four pre-selected major directions d_0 through d_3 in the environment is recorded, i.e., for each sample environment spot p, a 6-tuple (x, y, a_0, a_1, a_2, a_3) is kept in the map, where (x, y) specify the position of p and a_0 through a_3 specify the azimuth values which are obtained when the mobile device is oriented toward the four directions d_0 through d_3. To use the azimuth map in the navigation stage, with the detected user's position p as input, at first the sample spot A_p nearest to p is picked out from the map. Then, the azimuth angle value a_p at p for the current orientation of the user is measured using the magnetometer. Finally, the desired user's orientation d_p is computed by interpolation using the learned azimuth values a_0 through a_3 of A_p recorded in the azimuth map.

Finally, in color edge mark detection, the color edge mark becomes a strip shape in the image, so a line approximation scheme can be applied to detect the mark as described in the last chapter. Under the assumption that the user holds the device horizontally, the color edge mark becomes parallel to the ground. Therefore, we can determine the orientation of the color edge mark by the orientation of the line. The color edge mark has two mutually reverse directions, and we take as output the one closer to the orientation u_0 detected by the use of the magnetometer readings

5.3.2 Proposed Technique for User Viewing Orientation Detection

Based on the ideas about user orientation detection described previously, we propose a method for user viewing orientation detection in this study, which is described in this section. Recall that three techniques have been proposed for uses in different situations, namely, *human motion estimation*, *magnetic-field sensing*, and

multicolor edge mark detection.

In motion estimation, the positions of the users are detected as described previously in Section 5.2, which can be used to compute the orientations of the users by motion estimation. We adopt the same method to estimate the motions of multiple users as described in Section 5.3.1. We average the motion vectors to get a more stable result and use *turning flags* to determine whether users are turning. In this way, the result of motion-vector averaging will be changing more quickly in time to reflect the user's turning speed.

In magnetic-field sensing, we have established an azimuth in the learning stage map as well. Therefore, we can obtain the desired orientation of each user, which is computed by interpolation using the learned azimuth recorded in the azimuth map.

In multicolor edge mark detection, a multicolor edge mark, which is detected as described in Chapter 4, is used for orientation detection as well. We try to compute the direction vector of the approximating line in order to determine the device orientation. Under the assumption that the user holds the device horizontally, the multicolor edge mark becomes parallel to the ground. As shown by the example in Figure 5.3, the multicolor edge mark is represented as a yellow-green-pink line, and the red line and the yellow-green-pink line are projected onto identical image points; meanwhile, the vertical projection (shown as the dotted green line) of the multicolor edge mark is parallel to the red line. Therefore, we can determine the orientation of the color edge mark by the orientation of the red line.

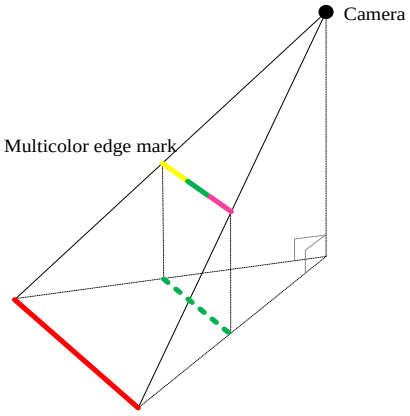


Figure 5.3 The red line and the multicolor edge mark (shown as yellow-green-pink line) are projected onto identical image points. The vertical projection (shown as dotted green line) of the multicolor edge mark is parallel to the red line.

The following algorithm describes the process to detect the user orientation by the three methods.

Algorithm 5.3 User viewing orientation detection.

Input: A users' position and body region in an image obtained by the schemes described in Section 5.2.

Output: the user's orientation *d*.

Steps

Step 1. Determine the user's orientation *using the magnetometer readings* provided by the mobile device and the azimuth map constructed in the learning stage, resulting in a direction d_0 for use as the initial user orientation

Step 2. With d_0 *as a reference*, apply *multicolor edge mark detection* to find another user orientation d_c and a corresponding reliability index r_c.

Step 3. If r_c is larger than a pre-selected threshold, then take d_c as the desired user orientation *d* and exit; otherwise, perform the next step.

Step 4. If the user is walking, then conduct *human motion estimation* to obtain a third user orientation *d* as the desired output and exit; otherwise, perform

the next step.

Step 5. Take d_0 obtained in Step 1 as the desired output and exit.

5.4 Review of User Tracking

5.4.1 User Tracking under a Single Fisheye Camera

The objective of user tracking is to identify the same person in consecutive video frames captured by fisheye cameras. For this aim, we adopt a *high-level tracking* technique proposed by Senior, et al. [19]. At first, as described in Section 5.2, the foreground regions in a foreground image are extracted, and each foreground region is enclosed with a bounding box. Then, for each successive video frame, each foreground region is *associated* with one of the existing *tracks* of objects including the user. A track here represents an identical object's movements in consecutive video frames. This process of *track association* is achieved by constructing a *tracking matrix* representing the distance between each of the foreground regions and all the existing tracks. Each row of the tracking matrix corresponds to one track, and each column corresponds to one foreground region. The distance is computed using a *bounding box distance measure* proposed in [19] as illustrated in Figure 5.4(a): the distance between two bounding boxes, *A* and *B*, is the smaller of the distance from the center of *A* to the nearest point on *B* and that from the center of *B* to the nearest point on *A*. In particular, if either box center lies within the other box as shown in Figure 5.4(b), then the distance is set to be zero.

(a) (b)

Figure 5.4 Bounding box distance measure. (a) The distance between A and B is the smaller of the distance from the center of A to the nearest point on B or from the center of B to the nearest point on A. (b) The distance is zero if either center lies within the other bounding box.

If a foreground region is close enough to only one track and only a region is close enough to the track, i.e., if there is a one-to-one correspondence between the track and the region, then the corresponding column and row will both have a "1" as shown in Figure 5.5(a), meaning that the region is associated with the track. However, two regions may be both close enough to one track, and this will produce two "1's" in a row as shown in Figure 5.5(b), meaning that both regions are associated with the track. Similarly, if a region is close enough to two tracks, the region is associated with both tracks. Finally, if two regions are both close enough to two tracks as shown in Figure 5.5(c), then the two regions are associated with both tracks. Accordingly, multiple regions may be associated with multiple existing tracks properly, by which *the same user* may be identified in consecutive video frames, achieving the goal of user tracking *under a single fisheye camera*.

The user tracking algorithm using the adopted method is described in Algorithm 5.4.

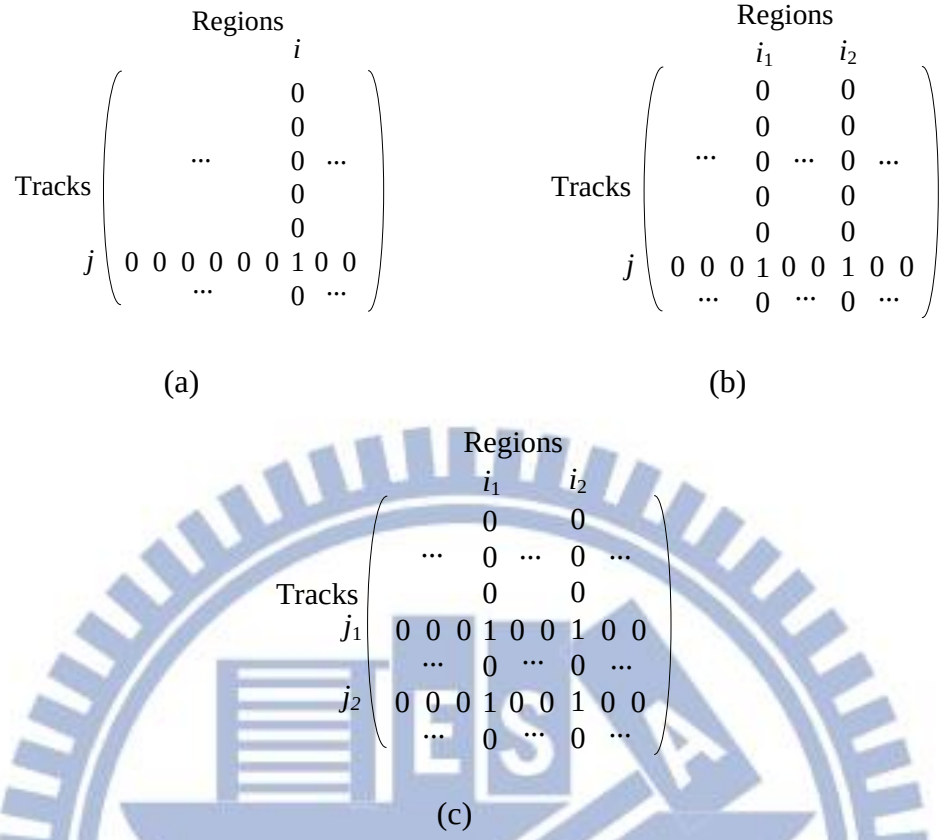


Figure 5.5 Tracking matrix at different situations. (a) A region is close enough to only a track, and only one region is close enough to the track. (b) Two regions are close enough to a track. (c) Two regions are close enough to two identical tracks.

Algorithm 5.4 User tracking.

Input: The foreground regions C in a frame and a set T of tracks with each track being associated with at least one foreground region; T_i meaning the ith track in T; and C_i meaning the ith foreground region in C.

Output: Tracks T.

Steps

Step 1. Create a tracking matrix M with all zeros with each row of M corresponding to one track of T, and each column of M corresponding to one region of C.

Step 2. Compute the bounding box distance d_{ij} between each C_i and $R(T_j)$ using the bounding box distance measure where the function $R(t)$ returns the associated region of the track t.

Step 3. For each d_{ij}, set $M(i, j)$ to be 1 if $d_{ij} < T_D$, where T_D is a pre-selected threshold value.

Step 4. Perform the following steps for M.

4.1. For each column i with only one non-zero element at row j which has only one non-zero element at column i, associate C_i with T_j.

4.2. For each column i with all zero elements, create a new track t_{new}, associate it with C_i, and add t_{new} into T.

4.3. For each row j with all zero elements, remove T_j from T.

4.4. For the columns $i_1, i_2, \ldots, i_m$ which have more than one non-zero elements at rows $j_1, j_2, \ldots, j_n$, associate $C_1, C_2, \ldots, C_m$ with $T_1, T_2, \ldots, T_n$.

In Step 3, we binarize the tracking map by the resulting distance; if two bounding boxes are close enough, the resulting value is set to be one; otherwise, it is set to be zero. In Step 4.2, if a foreground region is not associated with any track, then it is regarded as a new object to track. We remove tracks which are not associated with any object in Step 4.3.

5.4.2 Camera Hand-off for Tracking under Multi-cameras

A fisheye camera has a wider field of view, and can observe a wider range than a traditional perspective camera. However, if an object is located outside the view of a fisheye camera or far away from a fisheye camera, it will be projected onto a small region in the fisheye camera image, and the small region's pixels will be too few to perform aforementioned image analysis effectively. To avoid this problem, multiple cameras must be deployed so that the user can always move not far from at least one

fisheye camera. This also makes tracking of a user in a larger environment possible. But then, a user may "appear" in more than one image captured by the cameras. Therefore, we have to determine which camera should be used in tracking the same user, and this causes the *camera hand-off* problem — switching user tracking from one camera to another correctly. To handle such a problem, in each user-tracking cycle, we try to select a *single* fisheye-camera image, which includes the user's body region, from *multiple* ones according to a *minimum user-to-camera distance* criterion. If we have a foreground region obtained in the previous processing work, then we can obtain the foreground region representing the same user in the current processing by the following algorithm.

Algorithm 5.5 Camera hand-off.

Input: The foreground region h_u of a user, and the tracks $T(,)$, where $T(i,)$ means all tracks in omni-image I_i captured from camera C_i, and $T(i, j)$ means the jth track in $T(i,)$.

Output: The human region of the user in the current processing work.

Steps

Step 1. Find the track $T(s, u)$, which is associated with the foreground region h_u

Step 2. Set h_s to be the associated region of $T(s, u)$.

Step 3. Compute the location p_s of h_s in the GCS by

Step 4. Algorithm .

Step 5. Set $T' = \bigcup_{i \neq s} T(i,)$.

Step 6. Compute the locations in the GCS of all foreground regions associated with T', resulting in a location set P.

Step 7. Find the foreground region with its location being p_t, which is associated with the track in $T(t,)$ and satisfies the following two constraints:

(1) $d(p) = \left|\overline{pp_s}\right|$ with $p \in P$ is the minimum when $p = p_t$;

(2) $\left|\overline{p_t p_s}\right| \le \lambda$, where λ is a pre-selected threshold value.

Step 8. If p_t is found, compute the distance $d_t = \left|\overline{p_t c_t}\right|$ and $d_s = \left|\overline{p_s c_s}\right|$, where c_t is the location of the camera C_t and c_s is the location of the camera C_s;
if p_t is not found, set h_s as output and finish this algorithm.

Step 9. Set h_s as the output if $d_s < d_t$; otherwise, set h_t as the output.

The algorithm finds a foreground region representing the same object as the input foreground region. If we find a foreground region representing a user in the previous processing work, then we can use the region as input to find the corresponding region in the next processing work.

At first, we find the track which is associated with the input region. The found track is called a "*user track*." And then we find a region which is closest to the region associated with the user track in Steps 4 through 6. Then, we compare each of the locations of the two regions with each of the locations of its corresponding cameras. Finally, the one closer to its corresponding camera is chosen to be the output.

5.5 Proposed Algorithm for Multi-user Localization

In this section, we will describe the complete steps for multi-user localization. To do so, we integrate the technique of identifying multiple users in the environment described in Chapter 4 with those of multi-user localization described in the previous sections. After the localization work, the server-side system will send the location and orientation to the corresponding user's client-side system. The following algorithm

illustrates the complete steps for this task.

Algorithm 5.6 Algorithm for multi-user localization.

Input: images I_1 through I_n captured by n cameras C_1 through C_n, respectively and identification numbers of multiple users U_1 through U_n.

Output: the positions p_1 though p_n, orientations d_1 though d_n, and foreground regions $h_{u(1)}'$ though $h_{u(n)}'$of U_1 through U_n, respectively.

Steps

Step 1. Find foreground regions $R(,)$ in each of I_1, I_2, …, I_n by Algorithm 5.1, where $R(i,)$ means all foreground regions in omni-image I_i, and $R(i, j)$ means the jth foreground region in $R(i,)$.

Step 2. Track the foreground regions $R(1,)$, $R(2,)$, …, $R(n,)$ by Algorithm 5.4, resulting in the tracks $T(1,)$, $T(2,)$, …, $T(n,)$, where $T(i,)$ means all tracks in omni-image I_i, and $T(i, j)$ means the jth track in $T(i,)$.

Step 3. For i = 1, 2, …, n, do the following steps to deliver localization information to users identified and localized:

3.1. If the foreground region $h_{u(i)}$ of the user U_i in the previous navigation cycle has been found and recorded, then find $h_{u(i)}'$ by Algorithm 5.5 for the current cycle;

else, take the following steps find the foreground region for user U_i:

i. detect the multicolor edge mark by Algorithm 4.1 for $R(1,)$, $R(2,)$, …, $R(n,)$;

ii. detect the identification number N_j of $R(j,)$ by Algorithm 4.4 for j = 1, 2 , …, n;

iii. if the detected identification number N_j is equal to the identification number of U_i in any foreground region, apply Algorithm 5.5 to any of the foreground regions with identification

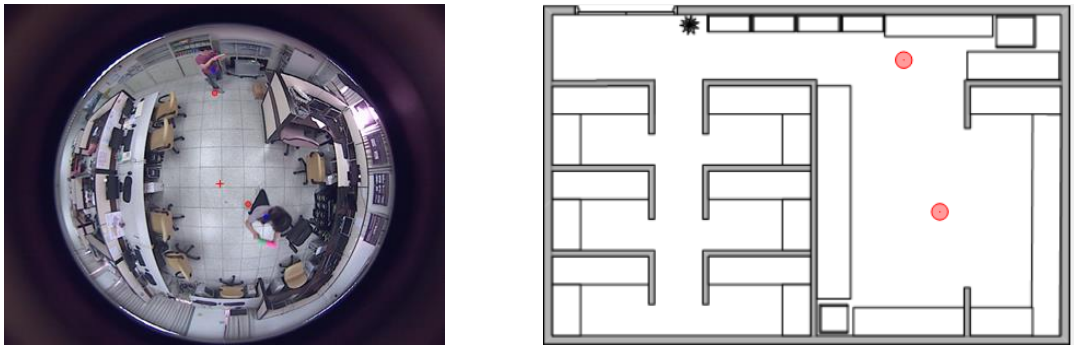
numbers detected to find the user region $h_{u(i)}'$ of U_i and go to 3.2.

3.2. Compute the location p_i of $h_{u(i)}'$ as an output for U_i by Algorithm 5.2.

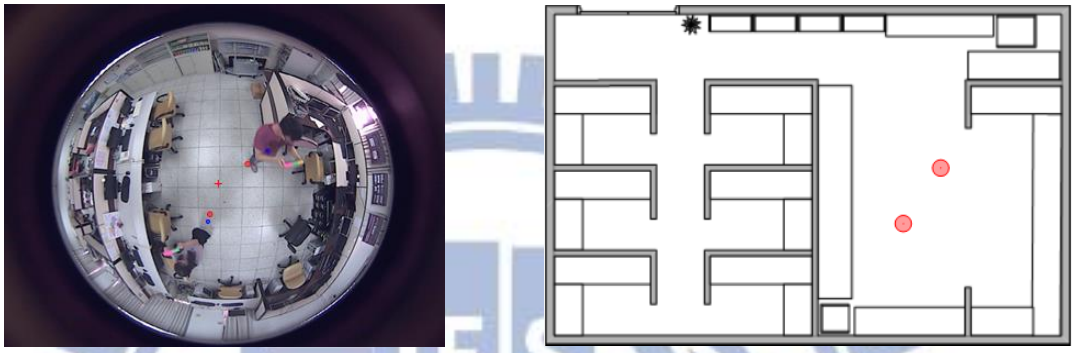
3.3. Compute the orientation d_i of $h_{u(i)}'$ as another output for U_i by Algorithm 5.3.

5.6 Experimental Results

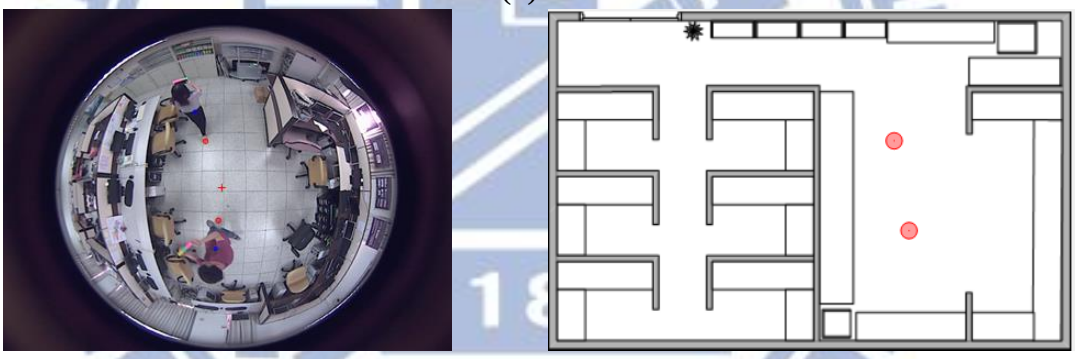
In this section, we show some experimental results of both multi-user localization, including results of location detection and orientation detection. Figure 5.6 shows the results of multi-user location detection at four different locations. It shows that the proposed method can actually find the foot points of two users and transform them from the FICS to the MCS. Figure 5.7 shows the results of user orientation detection with an integration of the three methods mentioned previously. The results are correct, showing that the proposed three methods together work effectively.



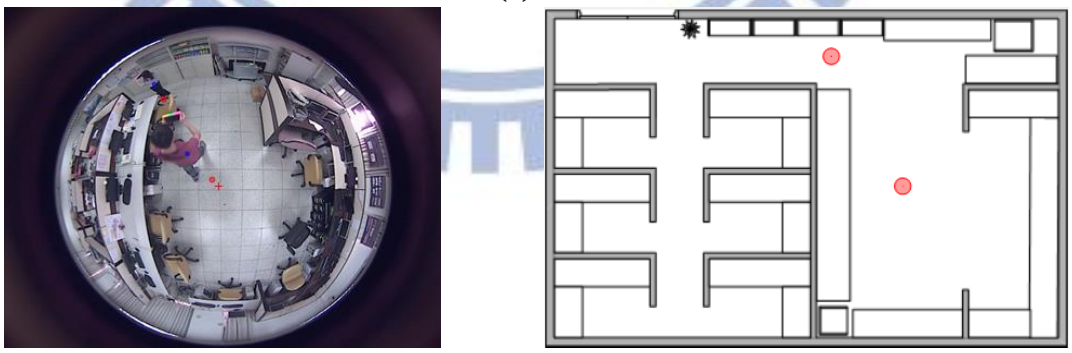
(a)



(b)

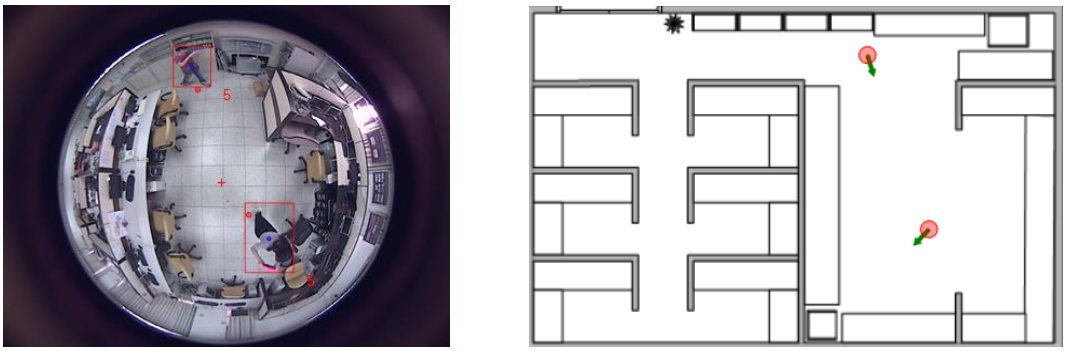


(c)

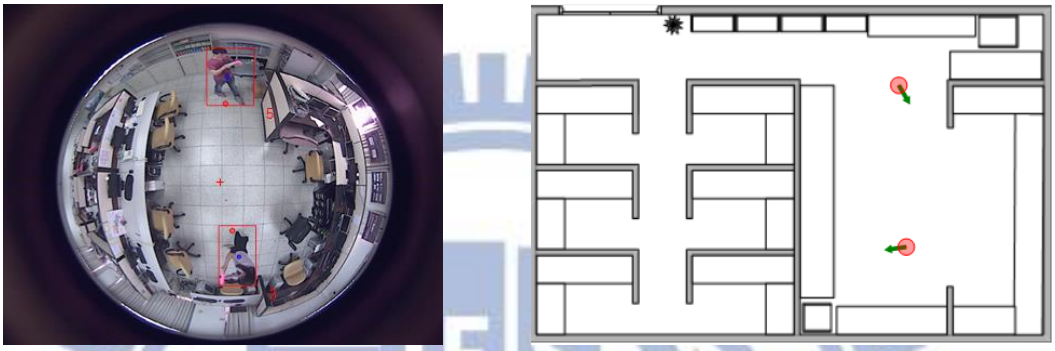


(d)

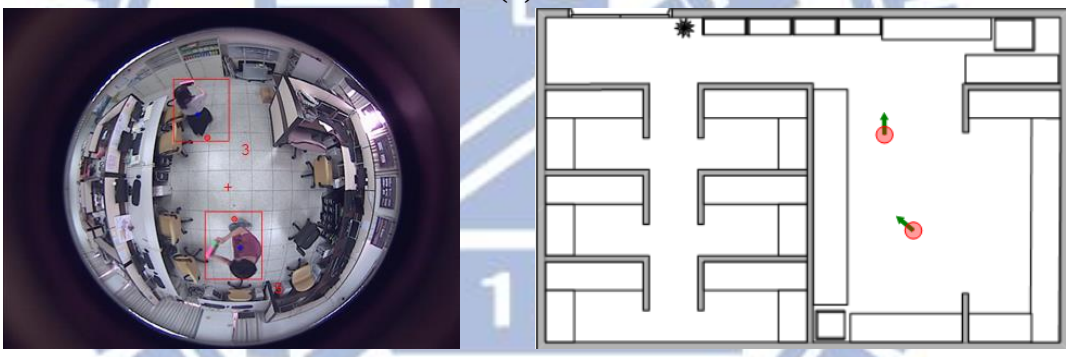
Figure 5.6 Multi-user location detection at four different locations.



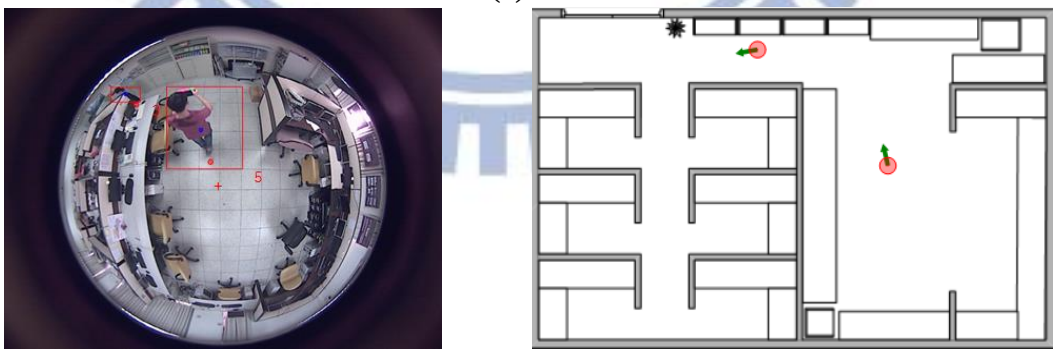
(a)



(b)



(c)



(d)

Figure 5.7 Multi-user orientation detection.

Chapter 6
AR-based Guidance for Merchandise Shopping

6.1 Ideas of AR-based Guidance for Merchandise Shopping

In this chapter, we will describe the proposed techniques of AR-based guidance for merchandise shopping used in the proposed system. In order to provide a more convenient and easy-to-use interface for merchandise shopping and other similar activities, we have to recognize merchandise items and then augment corresponding information on the client-device screen or plan paths to guide users to get the desired items.

Therefore, by the proposed techniques, at first, the server-side system analyzes the images of merchandise items and then constructs a database of merchandise items in the learning stage. Next, in the navigation stage, the server-side system analyzes the images captured from the client-device camera and matches them with the database. We adopt the SURF feature algorithm [14] to extract features from the acquired images. Finally, the server-side system sends the information of the recognized merchandise items to the client-side system. More details of the proposed processes of feature extraction both on merchandise images and on client-camera images will be introduced in Section 6.2.

For the purpose of guiding users to get the desired item, we adopt the path planning from the previous study of [17]. The server-side system will find the location of the user at first, and then find the location of the destination in the environment

map we construct in the learning stage. Next, the server-side system will begin to plan a path starting from the user and ending at the destination, and sends the result to the client-side system. More details will be introduced in Section 6.3.

6.2 Merchandise Recognition by Speeded-Up Robust Features (SURFs)

6.2.1 Review of SURF

The task of finding point correspondences between two images of the same scene or object is part of many computer vision applications. In this study, we adopt the SURF extraction [14] algorithm to extract features from the client-camera images and match with merchandise images for merchandise recognition.

In more detail, at first “interest points” with SURFs [14], which we called feature point in this study, are selected at distinctive locations in the image, such as corners, blobs, and T-junctions. The approach of [14] for interest point detection uses a very basic Hessian matrix approximation and integral images made popular by Viola and Jones [20], which reduces the computation time drastically. Next, the neighborhood of every interest point is represented by a feature vector. This descriptor has to be distinctive and at the same time robust to noise, detection displacements and geometric and photometric deformations. The descriptor describes the distribution of the intensity content within the interest point neighborhood, similar to the gradient information extracted by the SIFT [21] and its variants. Finally, the descriptor vectors are matched between different images. The matching is based on a distance between the vectors, e.g., the Mahalanobis or Euclidean distance. The dimension of the descriptor has a direct impact on the computation time, and fewer dimensions are

desirable for fast interest point matching. However, lower-dimensional feature vectors are in general less distinctive than their high-dimensional counterparts. In addition, they build on the distribution of first-order Haar wavelet responses in the *x* and *y* direction rather than on the gradient, exploit integral images for speeding up, and use only 64D. This reduces the time for feature computation and matching, and has been proven to simultaneously increase the robustness. In this study, we adopt the Euclidean distance for matching.

6.2.2 Feature Extraction from Merchandise Image

In the learning stage, we should construct a database of merchandise items. The merchandise information includes the feature points of the merchandise image, and the name, location, brand, and price of the merchandise item.

At first, we segment respective merchandise images from client-camera images. A client-device camera image is shown in Figure 6.1(a), which contains several merchandise items. We segment them to get respective merchandise images as shown in Figure 6.1(b).

Next, we extract feature points from these images using the SURF extraction algorithm [14] and record the descriptor of the feature points in the database, including the distribution of the first-order Haar wavelet responses in the *x* and *y* directions. The feature points in the merchandise images shown in Figure 6.1(b) are shown in Figure 6.1(c), where the size of the circle specifies the scale and the line in the circle is the orientation of the feature point.

In addition, if we have a large number of merchandise items, the time of matching will be long and not meet the expected performance. Therefore, we specify the height of each item when we construct the database of merchandise items. Then we can match the client-device images with the merchandise items at detected heights

rather than with all the items at all heights in the navigation stage.

Finally, in the learning stage, we can construct a database of merchandise items for merchandise recognition.

Figure 6.1 Feature extraction of merchandise image. (a) A client-device camera image (b) Merchandise images. (c) Feature points of merchandise images

6.2.3 Feature Extraction from Camera Images

In the navigation stage, the server-side system should analyze the client-camera images and match them with the database, which is constructed in the learning stage,

for merchandise recognition.

Therefore, we extract feature points from these images using the SURF extraction algorithm [14] and obtain a descriptor of the feature points. An example of client-camera images is shown on Figure 6.2(a). And the server-side system analyzes it to obtain the feature points. The extracted feature points on the client-camera image of Figure 6.2(a) are shown as Figure 6.2(b), where the size of the circle specifies the scale and the line in the circle specifies the orientation of the feature point.

Finally, with the feature points of client-camera images, we can match them with the database. The scheme of matching will be described in the next section.

(a)

(b)

Figure 6.2 Feature extraction from client-camera images. (a) A client-camera image (b) Feature points of the client- camera image of (a).

6.2.4 Matching of Merchandise Images and Camera Images by SURFs

After extracting feature points from both the merchandise image and the camera image, we match them for merchandise recognition.

The number of feature points is related to the speed of matching for merchandise recognition. And the number of feature points in an image depends on the image content and the size of the image. According to our experiments, the more feature points extracted for matching, the better performance we can obtain, but the lower the speed of matching. Therefore, we have to decide the sizes of merchandise images and those of the uploaded client-camera images. At first, the size of an originally acquired client-camera image is 800×480. According to our experimental experiences, the number of detected feature points is between 500 and 1200 and the number of detected feature points from six merchandise patterns is on the average 715. Correspondingly, the time for extraction and that for matching are both two seconds at most. They are too long to meet the expected performance. Therefore, we downsampled the size of the client-camera image to 533×320. The number of detected feature points then is reduced to be between 100 and 500 and the number of the detected feature points for each of six merchandise patterns is about 146. As a result, the time for extraction and that for matching are both one second at most, which is practical now for real-time requirements. So we decide to acquire client-camera images of the size 533×320.

After deciding the size of images to acquire, the server-side system has to detect the region of the merchandise to be recognized as well. Many merchandise items are contained in a client-camera image. We match each client-camera image with the pre-constructed database and obtain several matched feature points. We have to

specify the region of each merchandise items in an image, and then augment the corresponding information. Therefore, when we obtain the matched feature points, we try to find a bounding box to include them. As shown in Figure 6.3, the yellow boxes are the bounding boxes of the matched feature points. Finally, the server-side system obtains the region of each merchandise item, and sends the locations of the merchandise items in the image and the corresponding information to the client-side system for AR display.

Figure 6.3 The bounding boxes of matched feature points.

The following algorithm describes the process to match feature points between client-camera images and merchandise images.

Algorithm 6.1 Matching of merchandise images and camera images by SURFs.

Input: The merchandise item images M_1, M_2,… M_n obtained in advance and a client camera image I.

Output: The bounding boxes B_1, B_2, …, B_n of each merchandise items.

Steps

Step 1. Extract feature point set P from I.

Step 2. Extract feature point sets F_1, F_2, …, F_n from the merchandise item images

$M_1, M_2, \ldots, M_n$.

Step 3. Match P with $F_1, F_2, \ldots, F_n$ using the Euclidean distance to obtain matched feature point sets of $F_1', F_2', \ldots, F_n'$.

Step 4. Find bounding boxes $B_1, B_2, \ldots, B_n$ of $F_1', F_2', \ldots, F_n'$, respectively.

6.3 Path Planning for AR-based Guidance

6.3.1 Review of a Previous Work of Path Planning for Navigation

We adopt the method proposed in the previous study of [17] for path planning. The system proposed in [17] uses an obstacle avoidance map obtained in the learning stage to determine whether a planned path collides with any obstacle or not. If the path starting from the location of a user and ending at the location of the destination does not collide with any obstacle, the server-side system directly sends the two locations to the client-side system, which means that the user may now walk forward to the desired destination. However, if the path collides with an obstacle, the proposed system determines the immediate collision point to avoid the obstacle. For this purpose, the proposed system uses an obstacle avoidance map constructed in the learning stage. An obstacle avoidance map is created by splitting the MCS into small blocks. A block is a processing unit in the obstacle avoidance process; in other words, the planned path "walks" a block at a time, if the planned path walks to a block containing obstacles, it will find another block to go.

Next, the proposed system follows the avoidance directions to find the immediate points, and finally it will obtain a new path starting from the user's location to the

destination. However, the planned path may not be in the simplest form; in other words, there may exist two non-connected points on the path that can instead be connected together without any obstacle between the two points. Therefore, there is a scheme to simplify the planned plan. After a path is completely planned, it will be sent to the client-side system. A user can follow the path to reach the desired destination. However, if the user moves to a location which is not on the planned path, the planned path must be updated. Therefore, a path update scheme was also proposed in the previous study of [17]. Finally, by integrating all the schemes described above, the proposed system can conduct the path planning process completely.

6.3.2 Proposed Path Planning Techniques for AR-based guidance

Based on the path planning for navigation techniques as described previously, we propose a technique of path planning for AR-based guidance. In AR-based guidance for merchandise shopping and other similar activities, users will search the desired merchandise items or destinations. Therefore, the proposed system uses the path planning technique to guide users get to the desired items along the planned path.

At first, the proposed system uses an obstacle avoidance map constructed in the learning stage. Each element, which is a block, in the obstacle avoidance map represents two opposite *avoidance directions*. Also, the planned path "walks into" a block to reach a new spot at a time. At each spot p_i where collision avoidance is necessary, we assign three *avoidance blocks* as illustrated in Figure 6.4 for each of the two avoidance directions of p_i, including a *primary* avoidance block (shown in red) and two neighboring *secondary* avoidance blocks (shown in blue). We can find the next immediately-neighboring point for p_i from these avoidance blocks. Specifically,

for each avoidance block, we check whether it can be reached from p_i: if so, we add the center point, called an *avoidance point*, of the block into a *candidate set* S_{avoid}. The avoidance points in S_{avoid} then are sorted according to their distances to the destination point, and the sorted set are used for selecting the next immediately-neighboring point for p_i, as described next.

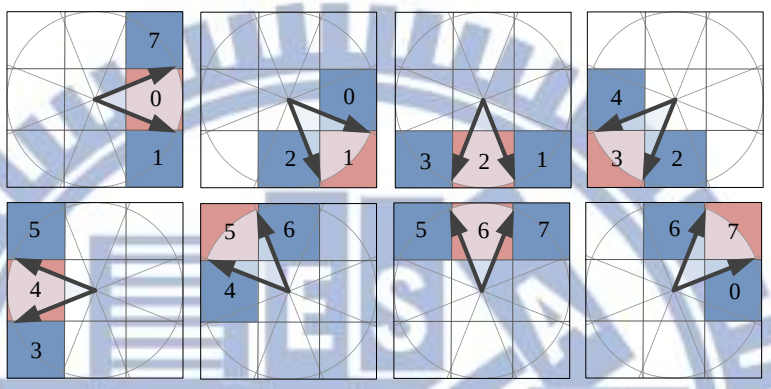


Figure 6.4 Avoidance blocks of 8 avoidance directions with each direction assigned three avoidance blocks — one primary avoidance block (red) and two secondary blocks (blue).

Next, a user searches the name of a merchandise item or a target place for visiting, and then the server-side system uses the location of the user as the start point and the location of the merchandise item or the target place as the destination to find a path by the following algorithm.

Algorithm 6.2 Path finding.

Input: A start point p in the MCS, a final destination point d in the MCS, and a set S_w which includes the points we have walked so far.

Output: A set of points of the found path S_{result}, and a flag f indicating whether a path is found or not.

Steps

Step 1. Initialize an empty set S_{result} to store the output points.

Step 2. Add p into S_{result} and S_w.

Step 3. If p can directly go to d without colliding obstacles, add d into S_{result} and then go to Step 9.

Step 4. Find the avoidance points, resulting in a set of avoidance points S_a.

Step 5. Take out the first avoidance point p_a in S_a.

Step 6. If p_a is contained in S_w, take the following steps:

6.1. if there still is any avoidance point in S_a, go to Step 5;

6.2. otherwise, set f to *fail* and then finish this algorithm.

Step 7. Apply this algorithm recursively with the inputs p_a, d, and S_w, resulting in a set of points of a found path S_{ad} and a flag f_a.

Step 8. If flag f_a is a *success*, add each point of S_{ad} into S_{result}; otherwise, go to Step 5.

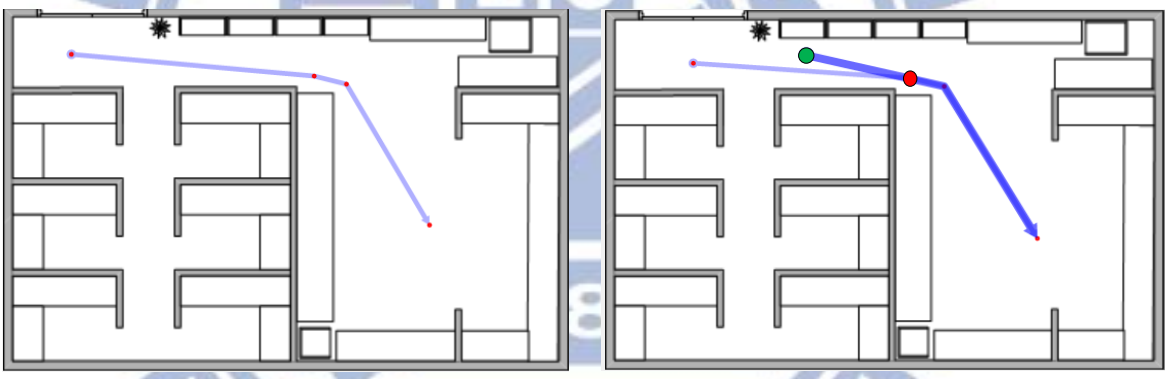
Step 9. Set S_{result} as output and set f to be a *success*.

After a path is found, we have to conduct further a *path simplification* process, which consists of two parts: *redundant point elimination* and *distance reduction*. The goal of *redundant point elimination* is to find two points that are non-connected and can instead be connected together without colliding any obstacle, and then to remove the intermediate points between them. The goal of the second part of path simplification, *distance reduction*, is to find a direct "shortcut" between *two line segments*. In short, we apply the redundant point elimination and distance reduction processes iteratively on an initially-planned path until the points of the path do not change any further.

In addition, a user might not always follow a planned path in a navigation session.

So we have to update the path when he/she walks away from the planned path. To do so (as shown in Figure 6.5(a) and Figure 6.5(b)), we just find the farthest point p_s (shown as the red circle in Figure 6.5(b)) reachable from the current point p (shown as the green circle in Figure 6.5(b)) in the planned path, and then compose the new path as p and p_s, followed by the remaining points in the old path.

After a path is updated, it contains only one new point, which is the current point p where the user is located, and the remainders in the path are the points of the originally planned path. Therefore, we have to check whether the resulting new line segments are of the simplest form or not. For this, we apply again the path simplification process described previously.



(a) (b)

Figure 6.5 Results of the path update process. (a) Original planned path. (b) Updated path.

The following algorithm describes the complete processes for path planning for AR-based guidance.

Algorithm 6.3 Path planning for AR-based guidance.

Input: A start point p in the MCS, and a final destination point d in the MCS.

Output: A set of points of the planned path.

Steps.

Step 1. If there exists a planned path P' and the destination of P' is identical to d, take the following steps.

1.1. Update P' as described previously, resulting in a set P of the immediate points of a path.

1.2. Go to Step 4.

Step 2. Find a path starting from p and ending at d by Algorithm 6.1, resulting in a set P of immediate points of the path.

Step 3. Simplify the found path on P as described previously.

Step 4. Take P as the output.

6.4 Proposed Algorithm for AR-based Guidance

In this section, we will describe the complete steps of an algorithm for proposed AR-based Guidance. To do so, we integrate the technique of identifying multiple users in the environment described in Chapter 4 with those of multi-user localization described in the previous sections. After the localization work, the server-side system will send the location and orientation information to the corresponding user's client-side system. The following algorithm illustrates the complete steps for this task.

Algorithm 6.4 AR-based guidance for merchandise shopping and other similar activities.

Input: A client-camera image I and a selected merchandise item or target place for visits.

Output: Information for augmented reality.

Steps

Step 1. When receiving a request of searching a merchandise item from the

client-side system, use each user location as a start point *p* and the location of the selected merchandise item or the target place as a destination point *d*, and take the following steps.

1.1 Find a path starting from *p* and ending at *d* by Algorithm 6.3, resulting in a set *P* of immediate points of the path.

1.2 Take *P* as the output and end the algorithm.

Step 2. Match the merchandise items at detected height and the camera image *I* by the use of SURFs by Algorithm 6.1, resulting in a set *B* of bounding boxes of the each merchandise items in *I*.

Step 3. Take *B* and corresponding information as the output.

6.5 Experimental Results

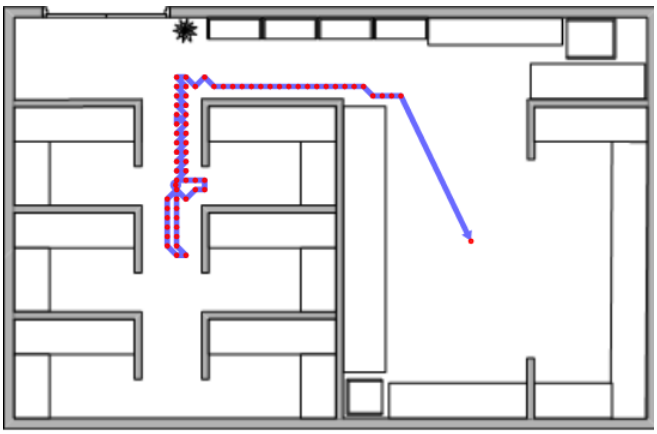
Figure 6.6 shows two examples of the results of merchandise recognition yielded by applying the previously described algorithms. Figure 6.6(a) is a figure of three merchandise items in the database. The recognition results are bounding boxes (shown as the yellow rectangles in Figure 6.6(b) and Figure6.6(c)) of recognized merchandise items yielded by Algorithm 6.1. Figure 6.7 shows an example of the results of the path planning work conducted by the above algorithm. The results contain a figure of the original planned path yielded by Algorithm 6.2 and a figure of the final simplified path.

(a)

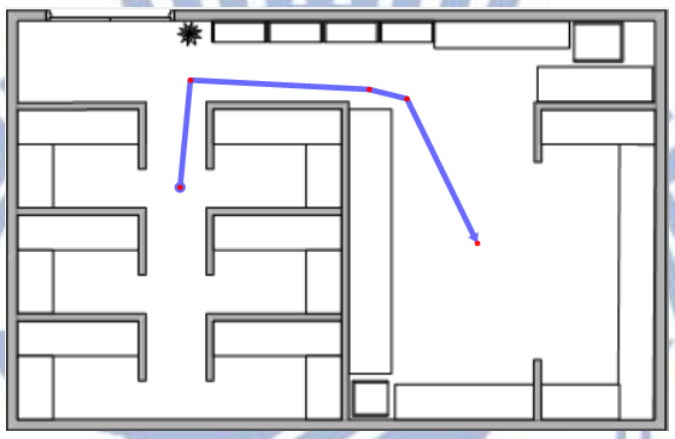
(b)

(c)

Figure 6.6 Result of merchandise recognition.(a) Three merchandise items. (b) A matching result with (a). (c) A matching result with (a).



(a)



(b)

Figure 6.7 Result of the path planning. (a) Result of the path finding. (b) Result of applying the path simplification on the path of (a).

Chapter 7
Augmented Reality for Merchandise Shopping

7.1 Ideas of Proposed Augmented Reality Techniques

We will describe the AR techniques used in the proposed system in this chapter. The AR techniques are implemented to run on the client-side system. The real-world environment can be augmented by computer-generated objects to enhance the perception of the real world. By this idea, we overlay the navigation or merchandise information onto the real image taken of the current scene seen through the client device, so that the user can just take his/her mobile device to conduct necessary navigation for merchandise shopping and other similar activities intuitively.

To implement the AR function, the server-side system sends the navigation information to the client-side devices. The navigation information includes the target place information, the merchandise information, and the navigation path. The client-side system will display these information data on the mobile device screen. The target place information and the navigation path are in the GCS. For the purpose of displaying the information in an AR way, the client-side system must transform the GCS coordinates onto the 2D screen plane of the mobile device. The calibration of the camera on the mobile device is described in Chapter 3. In Section 7.2, we will describe the process to perform the transformation between the GCS and the

mobile-device screen plane by the calibration result.

In Section 7.3 we will describe the display rendering scheme used in this study for the navigation information for merchandise shopping. We adopt the method of displaying the target place and path proposed in a previous work [17]. In Section 7.3.1, we will review how to determine the display position and how to display the information of the target place on corresponding objects in scene images. In Section 7.3.2, we will review the creation of the geometric shape of the navigation path (arrows, thick-line segments, etc.) for the navigation path to be overlaid onto the scene image to provide the AR-based guidance information. The information of each merchandise item includes the name, brand, and price of the item. We will describe how to display the information of the merchandise on corresponding items in scene images in Section 7.3.3.

7.2 View Mapping between Real World and Client Device

The client system is designed to transform the GCS coordinates of the tour sites appearing in the acquired image into the MICS on the 2D mobile-device screen plane in order to augment tour-site annotations (names and distances). This is done in two steps — from the GCS to the CCS and then from the CCS to the MICS. Transforming a point p in the CCS into a point q in the MICS can be done by following equations.

$$\begin{pmatrix} p'_x \\ p'_y \\ p'_z \\ p'_w \end{pmatrix} = \begin{pmatrix} \frac{h}{w}\cot\frac{\alpha}{2} & 0 & 0 & 0 \\ 0 & \cot\frac{\alpha}{2} & 0 & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix}; \qquad (7.1)$$

$$\begin{pmatrix} M_u \\ M_v \\ M_z \end{pmatrix} = \begin{pmatrix} w & 0 & 0 \\ 0 & h & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.5 & 0 & 0 & 0.5 \\ 0 & -0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \end{pmatrix} \cdot \begin{pmatrix} p'_x / p'_w \\ p'_y / p'_w \\ p'_z / p'_w \\ 1 \end{pmatrix}. \quad (7.2)$$

And to transform a point a in the GCS into a point p in the CCS, the following equation is used:

$$p = M_c \cdot \begin{pmatrix} a_x \\ a_y \\ a_z \\ 1 \end{pmatrix} \quad (7.3)$$

where $[a_x\ a_y\ a_z]$ are the coordinates of point a in the GCS and M_c is a matrix of the following form:

$$M_c = \begin{pmatrix} right_x & up_x & -forward_x & -c_x \\ right_y & up_y & -forward_y & -c_y \\ right_z & up_z & -forward_z & -c_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (7.4)$$

where the left three columns *up*, *right*, and *forward* of M_c are actually the three orthonormal bases of the CCS. And as shown in Figure 7.1, *up* specifies the upward direction of the mobile-device camera, *right* the rightward direction, and *forward* the frontal direction. The rightmost column includes the *x*- and *y*-coordinates (c_x, c_y) of the camera, which may be taken equivalently to be the user's position detected by the server system as described in Chapter 3. As to the *z*-coordinate c_z of the camera position, namely, its height, it is a pre-selected parameter taken to be about the average height of the eyes of an adult in the proposed system.

Accordingly, we can determine the parameters in the transformation matrix M_c by finding the three vectors *up*, *right*, and *forward*. The *up* direction of the CCS is the $+G_z$ direction of the GCS when the camera is not tilted, so the vector *up* may be taken to be (0, 0, 1) initially. Also, as mentioned, the camera location is identical to the user's position. Therefore, the 2D directional vector $d = [d_x\ d_y]^T$ of the camera on the

x-y plane can be obtained from the server system by the method described in Chapter 3. As to the *z* direction, as shown in Figure 7.2, the camera is tilted for a pitch angle β, which can be obtained from the orientation sensor of the mobile device. Therefore, we can specify the 3D direction of the camera by the vector $d' = [d_x \; d_y \; \sin\beta]^T$. Then, the vector *forward* can be obtained as the normalized version of d', i.e., as

$$forward = \frac{\vec{d}'}{|\vec{d}'|}. \tag{7.5}$$

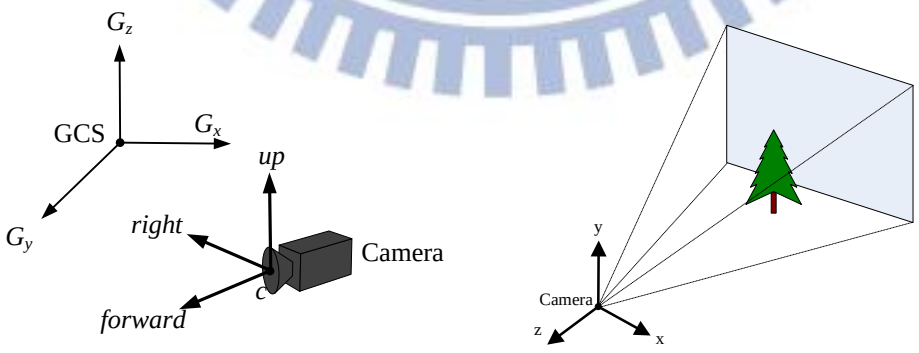
The last vector *right* can be obtained by:

$$\begin{aligned} \vec{r} &= forward \otimes up; \\ right &= \frac{\vec{r}}{|\vec{r}|} \end{aligned} \tag{7.6}$$

where $\otimes$ means the cross-product operation. Finally, we correct the vector *up* by the cross product of *right* and *forward* to make the three vectors to be orthogonal:

$$up = right \otimes forward. \tag{7.7}$$

Now, we have obtained all the parameters required to conduct the GCS to CCS transformation described by Equation 7.3 and Equation 7.4, followed by the CCS to MICS transformation described by Equation 7.1 and Equation 7.2.



(a) (b)

Figure 7.1 A camera in the GCS and the CCS. (a) A camera in the GCS with three orthonormal vectors *up*, *right*, and *forward*. (b) The CCS.

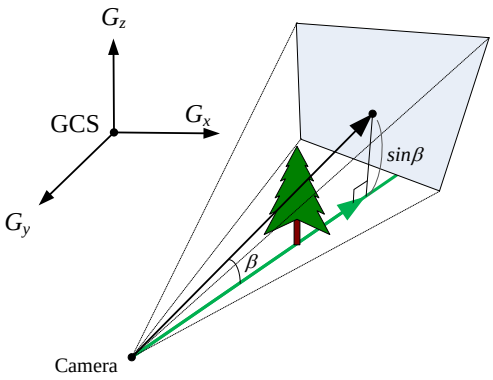


Figure 7.2 The camera is tilted for a pitch angle. The green line indicates a line on the horizontal *x*-*y* plane.

7.3 Information Augmentation and Path Rendering for Merchandise Shopping

7.3.1 Review of Rendering for Target Place Information

We adopt the method from [17] to conduct the rendering work for the target place information. To create AR effects, we overlay the names and distances of the target places onto corresponding positions in the scene image on the mobile-device screen. As illustrated in Figure 7.3(a), a target-place range *R* is defined by four parameters $\overrightarrow{f}$, *p*, *w*, and *h*, in the learning stage where vector $\overrightarrow{f}$ and point *p* specify the direction and position of *R*, respectively, and *w* and *h* specify the width and height of *R*, respectively. Accordingly, the four corner points of *R* can be computed and transformed into the MICS using the scheme described previously in Section 7.2. Also,

as shown in Figure 7.3(b), we clip the resulting coordinates of the corners to fit them into the range of the displayed image size. Finally, the position of the displayed text of the target place's name and distance as illustrated in Figure 7.3(c) can be computed to be the centroid of the four transformed corner points in the MICS. It is mentioned by the way that the distance of a target place is measured from the user's current position to the target place position, which can be computed from the target place position *p* and the user's position detected in the way described in Chapter 5.

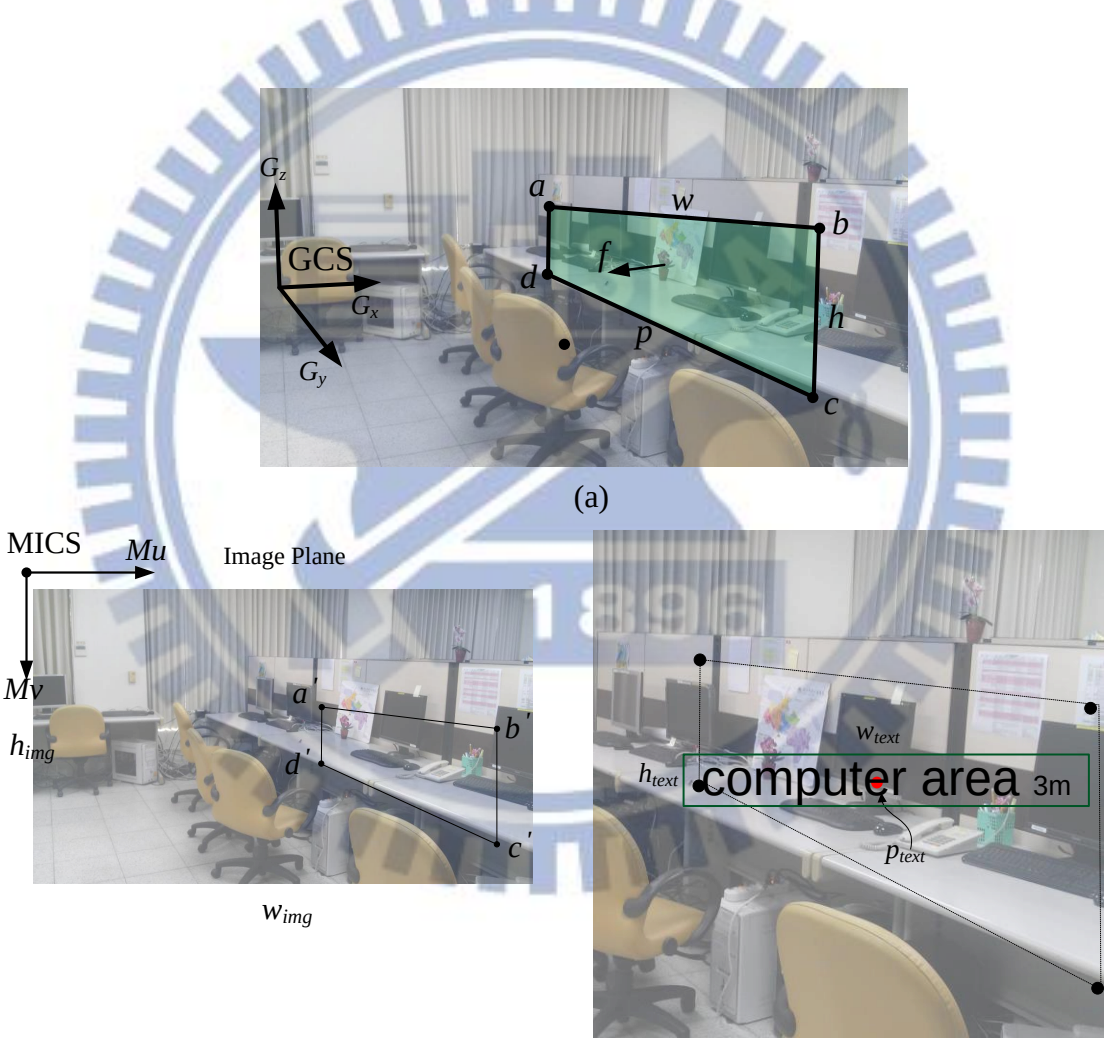


Figure 7.3 Transformation of the GCS of a target place into the MICS. (a) Parameters of the target-place range. (b) Clipping transformed results into the range of the image size. (c) Displayed text of the target place.

7.3.2 Review of Rendering for Displaying Navigation Path

We adopt the method from [17] to conduct the rendering work for displaying the navigation path. Only two line segments of a path are displayed at a time on the mobile-device screen for a clearer presentation of the planned path. The path composed of the first two line segments will be called "*display path*" in the subsequent sections. Figure 7.4Figure shows the expected result of overlaying the display path onto a scene image. The display path in Figure 7.4 is composed of thick line segments and an arrow. Therefore, we have to create the geometric shape of a path before conducting the display rendering work for them.

As mentioned in Chapter 2, the rendering of 3D augmented objects is conducted by the OpenGL API. The OpenGL API processes 3D objects composed by triangles or quadrangles, which are called "*geometric primitives*." As shown in Figure 7.5, the geometric shape of a display path is composed by 11 points, and we can get the geometric primitives of the display path by the 11 points of a path.

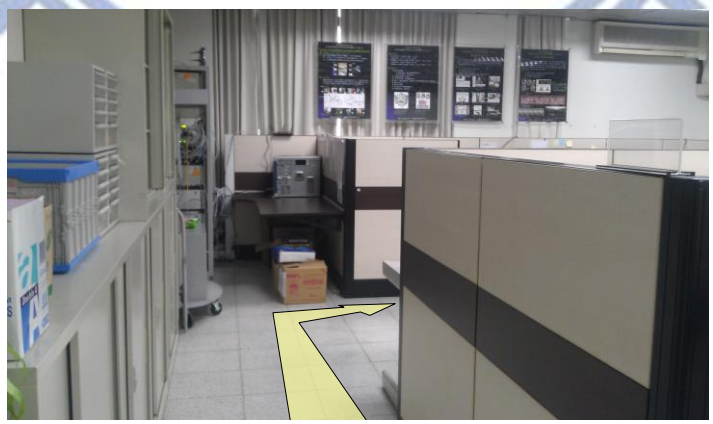


Figure 7.4 The display of the first two line segments of a path.

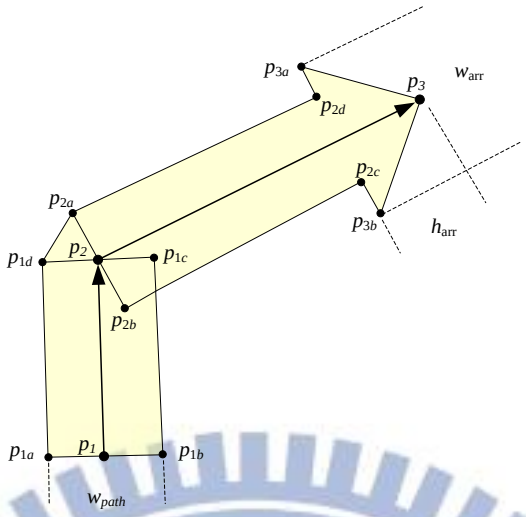


Figure 7.5 The geometry of a display path

7.3.3 Augmentation of Merchandise Information

We have to overlay the name, brand, price of each merchandise item onto the corresponding object in the real world, which appears in the image taken with the camera on the user-held mobile device. To accomplish this aim, we determine where to display the text on the device screen. For this, we compute the MICS coordinates of points *a*, *b*, *c*, and *d* shown in Figure 7.6 from the bounding box by Algorithm 6.1.

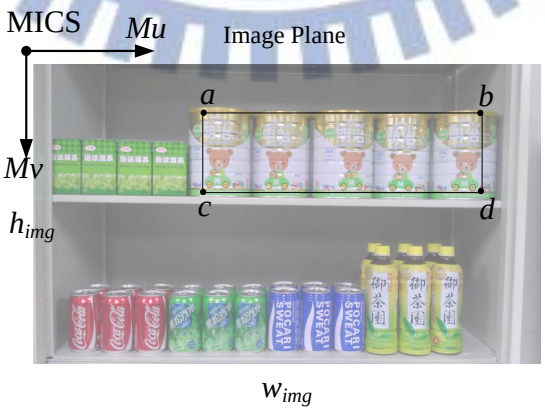


Figure 7.6 Parameters of a merchandise item.

Then, we can obtain the display position of the text of the merchandise name by the following equation:

$$p_{text} = \frac{a+b+c+d}{4}. \tag{7.8}$$

And then we can display the text on the display position p_{text} as shown in Figure 7.7Figure , where w_{text} is the width of the text, and h_{text} is the height.

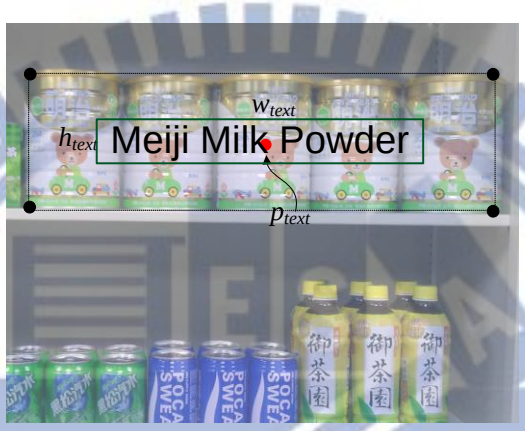


Figure 7.7 Display the name of a merchandise item on the display position p_{text}.

In addition, we have to display the text of the brand and price. It is not to figure out that we can obtain the display position of the text of the merchandise brand and price by the following equations:

$$\begin{pmatrix} p_{textb_x} \\ p_{textb_y} \end{pmatrix} = \begin{pmatrix} p_{text_x} - \frac{w_{text}}{2} + \frac{w_{textb}}{2} \\ p_{text_y} + \frac{h_{text}}{2} + \frac{h_{textb}}{2} \end{pmatrix} \tag{7.9}$$

$$\begin{pmatrix} p_{textp_x} \\ p_{textp_y} \end{pmatrix} = \begin{pmatrix} p_{textb_x} - \frac{w_{textb}}{2} + \frac{w_{textp}}{2} \\ p_{textb_y} + \frac{h_{textb}}{2} + \frac{h_{textp}}{2} \end{pmatrix} \tag{7.10}$$

by the following equations:

$$\begin{pmatrix} p_{textb_x} \\ p_{textb_y} \end{pmatrix} = \begin{pmatrix} p_{text_x} - \frac{w_{text}}{2} + \frac{w_{textb}}{2} \\ p_{text_y} + \frac{h_{text}}{2} + \frac{h_{textb}}{2} \end{pmatrix} \tag{7.9}$$

$$\begin{pmatrix} p_{textp_x} \\ p_{textp_y} \end{pmatrix} = \begin{pmatrix} p_{textb_x} - \frac{w_{textb}}{2} + \frac{w_{textp}}{2} \\ p_{textb_y} + \frac{h_{textb}}{2} + \frac{h_{textp}}{2} \end{pmatrix} \tag{7.10}$$

Accordingly, we can display the text on the display position p_{textb} and $ptextp$ as shown in Figure 7.8Figure , where w_{textb} is the width, h_{textb} is the height of the brand text, w_{textp} is the width, and h_{textp} is the height of the price text.

Figure 7.8 Displaying the brand and price of a merchandise item on the display position p_{textb} and p_{textp}.

If the range of text exceeds the range of the image plane, we correct the text position p with text width w and text height h to p' by the following equation:

$$\begin{pmatrix} p'_x \\ p'_y \end{pmatrix} = \begin{pmatrix} \max\left(\min\left(p_x, w_{img} - \frac{w}{2}\right), \frac{w}{2}\right) \\ \max\left(\min\left(p_y, h_{img} - \frac{h}{2}\right), \frac{h}{2}\right) \end{pmatrix}. \tag{7.11}$$

Now, we summarize all the processes of merchandise information augmentation discussed above by the following algorithm.

Algorithm 7.1 Augmentation of Merchandise Information

Input: An image I to be augmented with width w_{img} and height h_{img}; a merchandise item with the bounding box of corners a, b, c, d; its name t to display with

width w_{text} and height h_{text}; its brand b to display with width w_{textb} and height $h_{textb;}$ and its price m to display with width w_{textp} and height h_{textp}.

Output: An augmented image.

Steps.

Step 1. Compute the display position p_{text} by Equation 7.8.

Step 2. Correct p_{text} to be p_{text}' to make the drawn text not exceeding the range of I by Equation 7.9.

Step 3. Draw t on I at point p_{text}'.

Step 4. Compute the display position p_{textb} by Equation 7.9.

Step 5. Correct p_{textb} to be p_{textb}' to make the drawn text not exceeding the range of I by Equation 7.11.

Step 6. Draw b on I at the point p_{textb}'.

Step 7. Compute the display position p_{textp} by Equation 7.10.

Step 8. Correct p_{textp} to be p_{textp}' to make the drawn text not exceeding the range of I by Equation 7.11.

Step 9. Draw m on I at the point p_{textp}'.

Step 10. Take I as the output.

7.4 Proposed Algorithm of AR for Merchandise Shopping

In this section, we summarize the processes described in the previous sections as a total process — the process of indoor navigation for merchandise shopping and other similar activities by augmented reality, as described in Algorithm 7.2 below.

Algorithm 7.2 Indoor navigation for merchandise shopping by augmented reality.

Input: A scene image.

Output: An augmented image.

Steps

Step 1. Obtain the user's orientation and location from the server-side system.

Step 2. Obtain the pitch angle from the orientation sensor of the client device.

Step 3. Create the projection matrix by the method described in Section 7.2.

Step 4. Obtain target place information from the server-side system.

Step 5. Display all target places by the method described in Section 7.3.1.

Step 6. Obtain recognized merchandise information from the server-side system.

Step 7. Display all recognized merchandise item information by Algorithm 7.1.

Step 8. Search the desired destination or merchandise by a keyword to obtain a planned path.

Step 9. Display the planned path by the method described in Section 7.3.2.

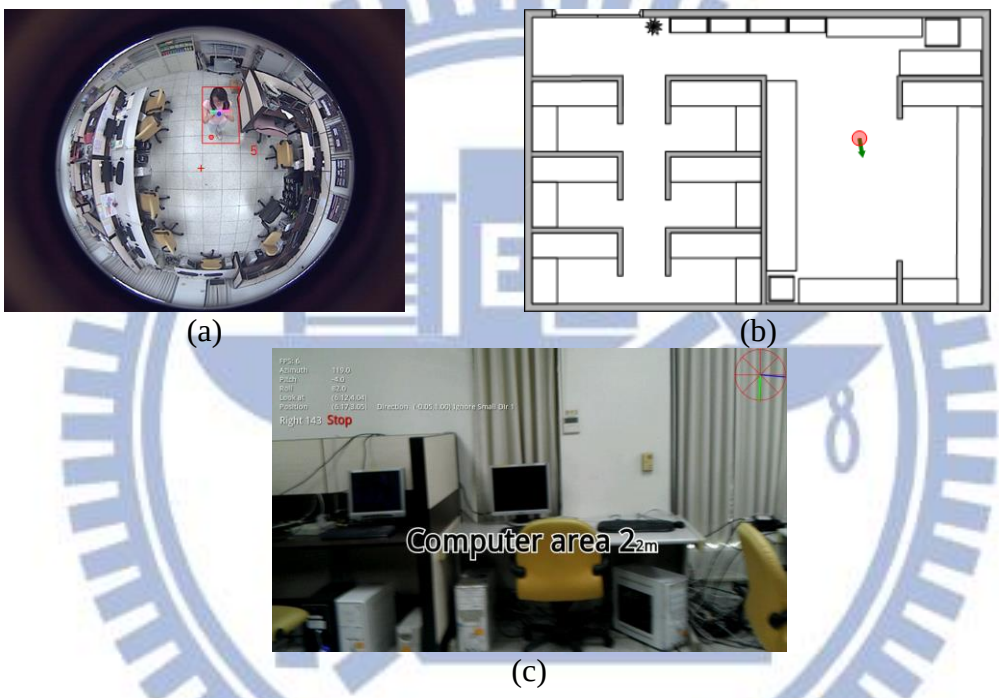
7.5 Experimental Results

Figures 7.9 and 7.10 show two results of overlaying target place information on scene images. The figure includes an omni-image captured from a fisheye camera as (a) in the figure, the detected location and orientation illustrated in (b), and the augmented image shown on the user's mobile device as seen in (c). A user can understand the surrounding environment by the target place information on the augmented image.

Figure 7.11 shows the results of overlaying merchandise item information on scene images. The figure includes merchandise information, such as name, brand, and price, of the recognized merchandise items.

Figure 7.12 shows a result of overlaying a navigation path on the scene image.

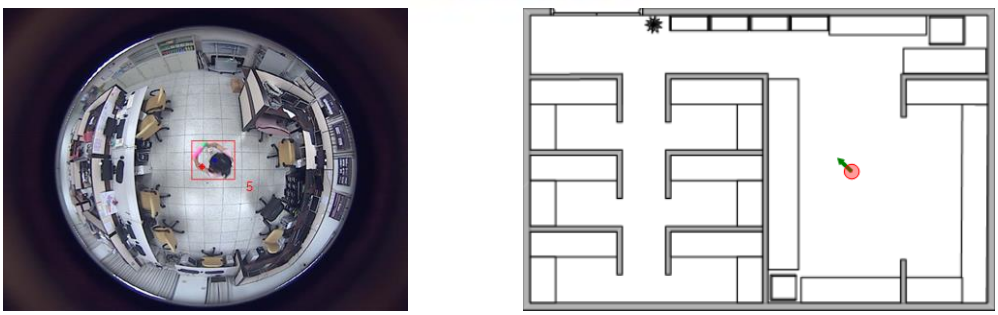
The figure includes four augmented images at different locations and in different orientations. A user can understand how to reach the desired destination by following the navigation path shown by the arrow and the line segment in the figures. When the destination is out of the screen, the navigation path may be invisible in an augmented image. At that time, the system will display the destination on the edge of the screen to indicate the correct direction.



(a) (b)

(c)

Figure 7.9 An augmented image with target place information. (a) An omni-image. (b) Detected location and orientation. (c) The augmented image shown on user's mobile device.



(a) (b)

(c)

Figure 7.10 An augmented image with target place information. (a) An omni-image. (b) Detected location and orientation. (c) The augmented image shown on user's mobile device.

(a) (b)

Figure 7.11 Augmented images with merchandise item information. (a)An augmented image shown on the user's mobile device. (b) An augmented image shown on the user's mobile device.

(a) (b)

(c) (d)

Figure 7.12 An augmented image with a navigation path. (a)(b) The augmented images at three different locations. (c)(d) When the destination is outside of the screen, the name of the destination will display on the edge of the screen (shown as the yellow stroke text); (c) The destination is on the left rear of the user; (d) The destination is on the rear of the user.

Chapter 8
Experimental Results and Discussions

8.1 Experimental Results

In this section, we will show some experimental results of the proposed indoor AR-based navigation system. The experimental environment is a laboratory of the Computer Vision Research Center in the Microelectronics and Information System Building at National Chiao Tung University. Though it is not a real market place, the objects in the room, including open shelves with merchandise items, tables with computers, etc., are sufficient to simulate the function of a supermarket. The environment map is shown in Figure 8.1, which includes six target places (shown as green regions), eight merchandise items (shown as blue texts), and two fisheye cameras (shown as blue circles).

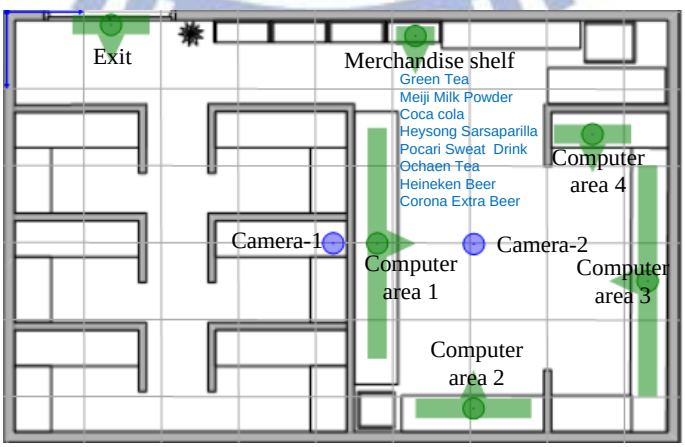


Figure 8.1 The map of the experimental environment.

8.1.1 Result of Real Multi-user Browsing

In this section, we show the results of browsing surrounding target places at certain locations. Figure 8.2 shows results of two users browsing surrounding target places in the environment. We show the omni-images captured from the fisheye cameras and the corresponding augmented images of each user's mobile device at different times. The omni-images captured from the fisheye cameras are shown on the upper side of this figure, and the augmented images shown on each user's mobile device are shown on the lower side. The lower-left image is the augmented image appearing on the female user's mobile device. The lower-right image is the augmented image appearing on the male user's mobile device. We can see that the different target places displayed on the screens depends on the different users' locations and orientations.

(a)

Figure 8.2 Two users browsing surrounding target places at certain locations. The upper side is the images captured from the fisheye cameras, the lower-left side is the augmented image shown on the female user's mobile device, and the lower-right side is the augmented image shown on the male user's mobile device.

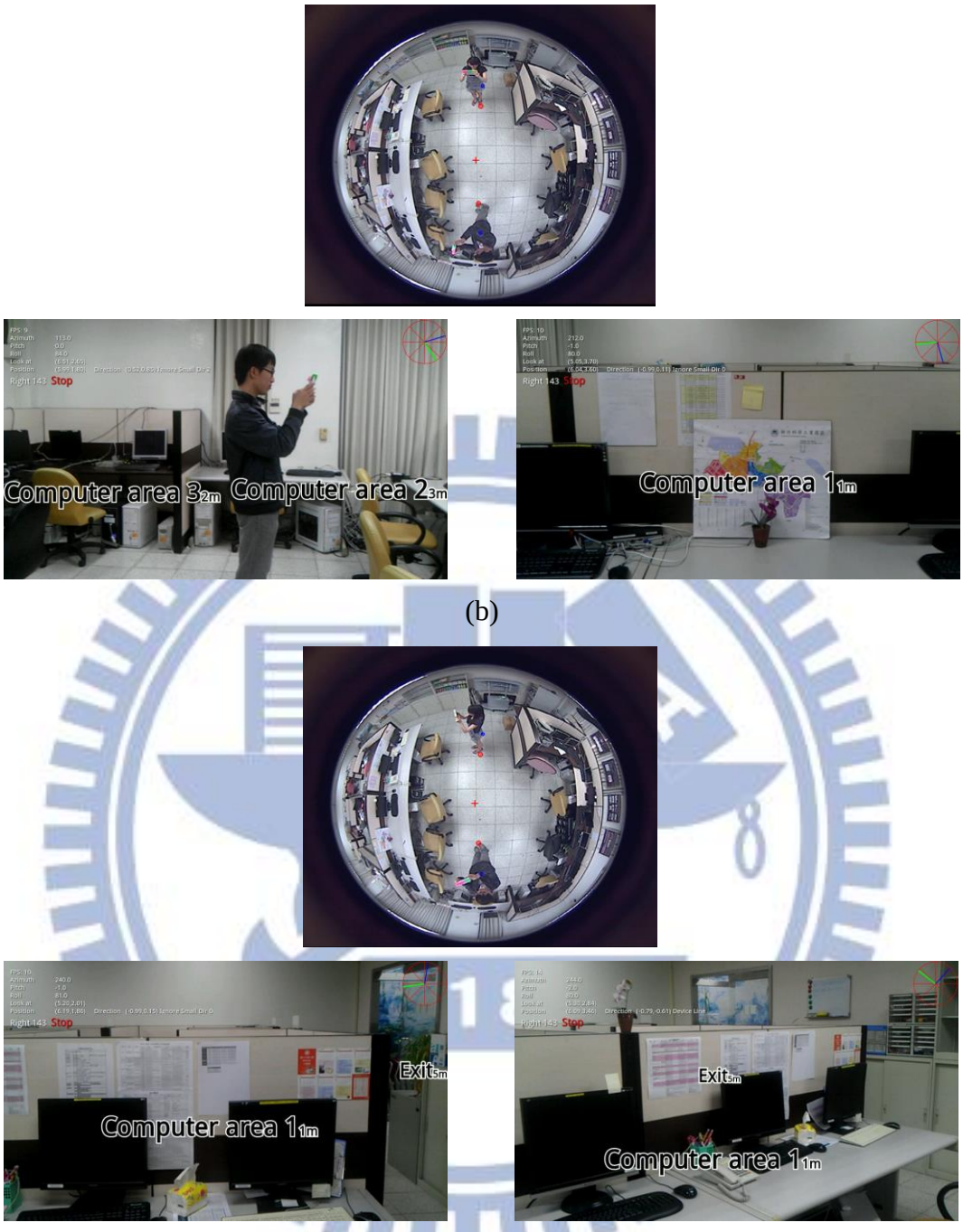


Figure 8.2 Two users browsing surrounding target places at certain locations. The upper side is the images captured from the fisheye cameras, the lower-left side is the augmented image shown on the female user's mobile device, and the lower-right side is the augmented image shown on the male user's mobile device (cont'd).

(d)

(e)

Figure 8.2 Two users browsing surrounding target places at certain locations. The upper side is the images captured from the fisheye cameras, the lower-left side is the augmented image shown on the female user's mobile device, and the lower-right side is the augmented image shown on the male user's mobile device (cont'd).

8.1.2 Result of Real AR-based Guidance for merchandise shopping

A. Browsing merchandise items

In this section, we show the results of merchandise items browsing. Figures 8.2(a)-(d) show the results of browsing single merchandise items in the augmented images. Figures 8.2(e)-(h) show the results of browsing multi-type merchandise items in the augmented images. From these results, one can see that the feasibility of the proposed methods for merchandise item recognition as well merchandise information augmentation.

(a) (b)

(c) (d)

Figure 8.3 Results of browsing merchandise items. (a)-(d) There is single-type merchandise items in the augmented image.(e)-(h) There are multi-type merchandise items in the augmented image.

(e)

(f)

(g)

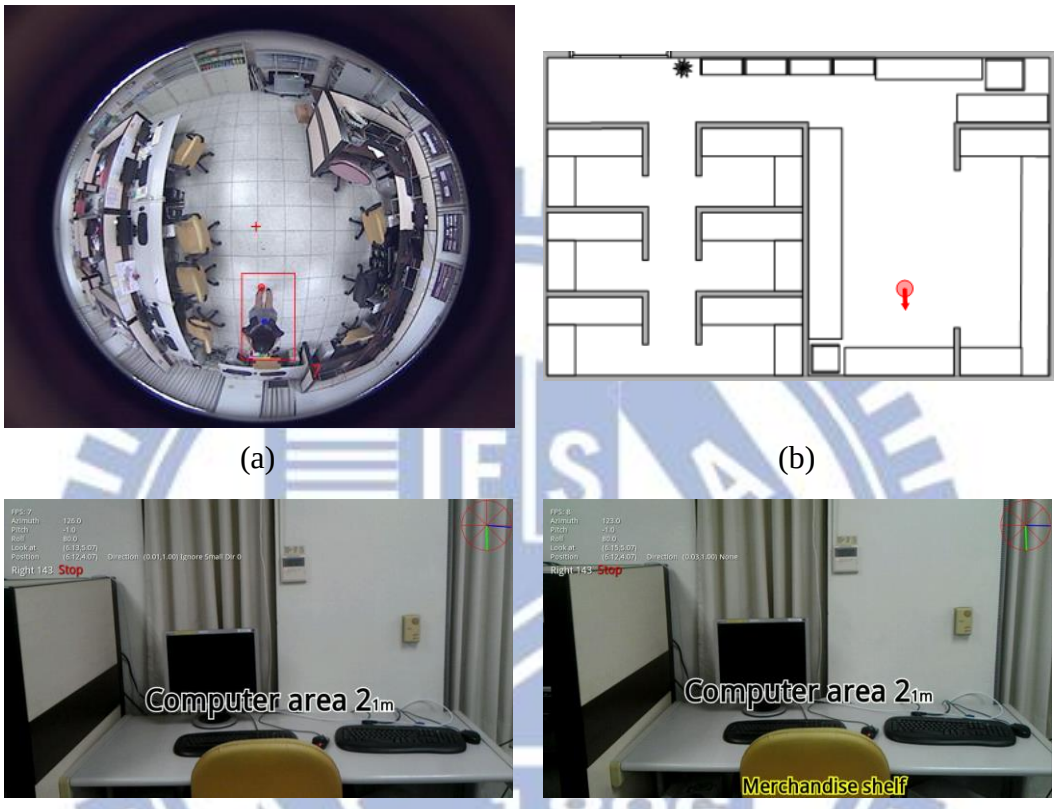
(h)

Figure 8.3 Results of browsing merchandise items. (a)-(d) There is single-type merchandise items in the augmented image.(e)-(h) There are multi-type merchandise items in the augmented image (cont'd).

B. Navigation by a planned path

In this section, we show a result of navigation according to a planned path. A user stood at a location as shown in Figure 8.4(a) and the detected location and orientation of her are shown in Figure 8.4(b). Figure 8.4(c) shows the augmented image seen by the user. Then, the user searched a merchandise item (i.e., Coca cola), and there appeared a yellow stroke text, which indicates the location of the searched merchandise item (on the merchandise shelf), on the bottom edge of the augmented image (as shown in Figure 8.4(d)). The user can then understand that the destination is on the rear, so the user began to turn to the right-hand side. As the user was turning, we can see that the destination was moving to the right-hand side of the user (as shown in Figure 8.4(e)-(f)). Then, the user saw the destination and the navigation path

when he turned to the correct direction (as shown in Figure 8.4(g)). Therefore the user moved closer to the destination. Finally, the user browsed the merchandise items and found the searched item (Coca cola).



(a) (b)

(c) (d)

Figure 8.4 A result of navigation by a planned path. (a) A user was at a certain location. (b) The detected location and orientation. (c) The augmented image seen by the user. (d) The augmented image shown when the user searched a target place, and there is a yellow stroke text shown on the bottom edge of the augmented image, which indicates the direction of the destination. (e)(f) The augmented image shown when the user is turning to the right-hand side. (g) The augmented image shown when the user is turning to the correct direction. (h) The augmented image shown when the user is moving closer to the correct direction. (i) The augmented image shown when the user is facing the searched item.

(e)

(f)

(g)

(h)

(i)

Figure 8.4 A result of navigation by a planned path. (a) A user was at a certain location. (b) The detected location and orientation. (c) The augmented image seen by the user. (d) The augmented image shown when the user searched a target place, and there is a yellow stroke text shown on the bottom edge of the augmented image, which indicates the direction of the destination. (e)(f) The augmented image shown when the user is turning to the right-hand side. (g) The augmented image shown when the user is turning to the correct direction. (h) The augmented image shown when the user is moving closer to the correct direction. (i) The augmented image shown when the user is facing the searched item (cont'd).

8.2 Discussions

The experimental results of the proposed indoor AR-based multi-user navigation system presented previously show that we can utilize fisheye cameras to detect the locations and orientations of multiple users. Meanwhile, the users can understand the surrounding environment and conduct desired navigations by the proposed AR techniques.

However, the proposed system still has some problems. If the users are too close to each other, they will be recognized as a single user. A possible way to solve this problem is to use multi-color edge marks to separate these users from detected regions of users. If we detect more than one multi-color edge marks in the detected region, we can recognize them as different users in the detected region.

Meanwhile, as users are moving far away from a fisheye camera, the detected locations will become more and more unstable. This is because the detected locations are computed by interpolation of four calibration points, but the pixels of farther objects will have higher distortions. A similar problem will occur on the multi-color edge mark detection. As the user is moving away from a fisheye camera, the region of the multi-color edge mark in the omni-image will become smaller and smaller, and eventually hard to detect. Then the identification of multiple users will become unstable. A possible way to solve these problems is to use more cameras in the environment. Therefore, when a user moves away from a camera, it can be detected by another camera which is closer to the user.

Chapter 9
Conclusions and Suggestions for Future Works

9.1 Conclusions

An indoor multi-user navigation system by augmented reality and down-looking omni-vision techniques using mobile devices has been proposed. To design such a system, several techniques have been proposed as summarized in the following.

1. *A method for multi-user identification in omni-images of indoor environments* has been proposed, by which the system can identify users by classifying the image of the multi-color edge mark attached on the mobile device to obtain the use identification number.
2. *A method for multi-user localization in indoor environments by 3D omni-image analysis* has been proposed, by which we can obtain the locations and orientations of multiple users in an indoor environment. The orientation detection algorithm integrates three different techniques to detect the orientation of a user, and each of the techniques can make up the deficiencies of the others.
3. *A method for recognition of merchandise items by SURF extraction and matching* has been proposed, by which the system can recognize merchandise items and obtain the corresponding 3D positions and information of the desired merchandise items.
4. *A method for indoor AR-based guidance for merchandise shopping by overlaying merchandise information on the real objects in scene images* has been proposed,

by which a user can understand the merchandise items in an AR way from the overlaying merchandise information on the users' mobile-device screen.

5. *A method for indoor AR-based guidance for merchandise shopping by overlaying a navigation path on the floor in the scene image* has been proposed, by which a user can follow a navigation path shown on the mobile-device screen in an AR way to reach the desired merchandise items.

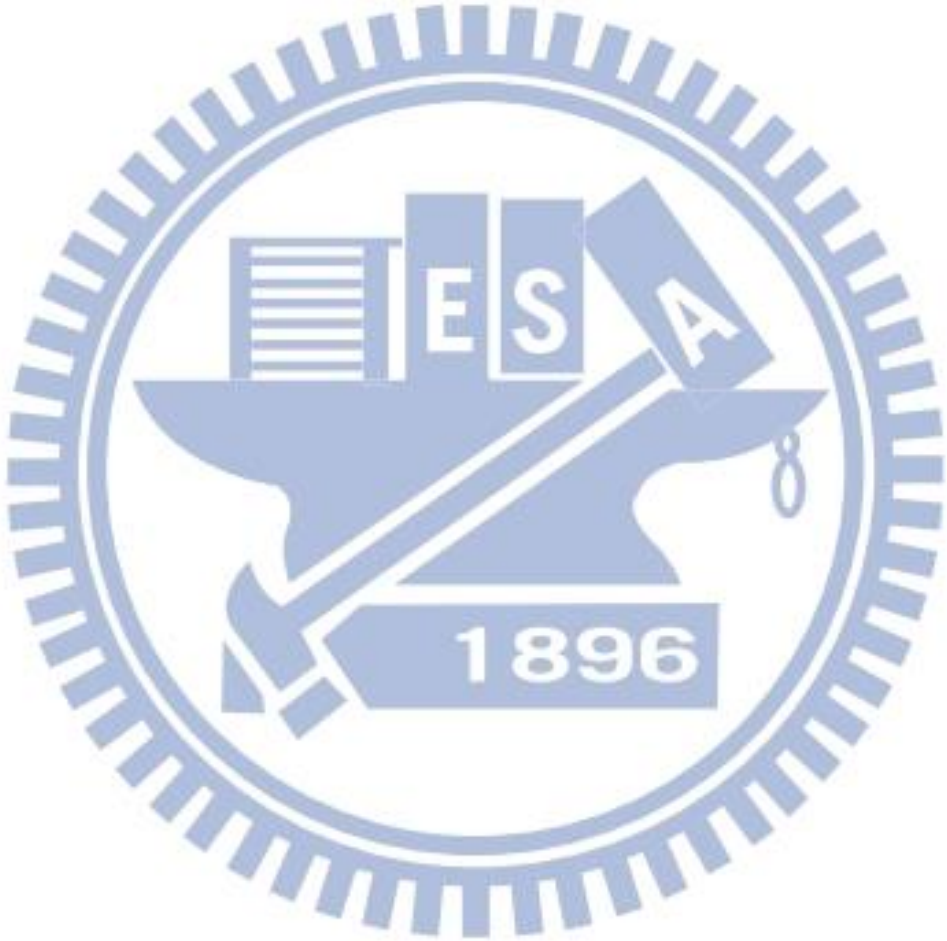
The experimental results shown in the previous chapters have revealed the feasibility of the proposed system.

9.2 Suggestions for Future Works

According to our experience obtained in this study, several issues and possible extensions of the proposed system worth further studies are listed in the following:

1. Designing a background/foreground separation algorithm which can adapt to different lighting conditions for identification of multiple users.
2. Seeking a solution to solve the problem of multi-user location detection in situations where the users' feet are occluded by obstacles or by other users.
3. Proposing a method for user orientation detection with higher precision and better stability, which can be accomplished by matching the image captured from the user's mobile device against a pre-learned database.
4. Communicating images captured the cameras on different users' mobile devices, by which the users can browse different places without going there.
5. Providing the capability for processing multiple environment maps, which can be utilized for multi-user localization on different floors or at distinct places of an indoor environment.
6. Recording the merchandise information browsed by users, which can provide the

users opportunity to compare the information between similar merchandise items.



References

[1] C. Lukianto, *et al*., "Pedestrian Smartphone-Based Indoor Navigation Using Ultra Portable Sensory Equipment," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2010, pp. 1-5.

[2] B. Ozdenizci, et al., "Development of an Indoor Navigation System Using NFC Technology " in *Fourth International Conference on Information and Computing (ICIC)*, 2011, pp. 11-14

[3] A. Mulloni, et al., "Indoor Positioning and Navigation with Camera Phones," *Pervasive Computing, IEEE*, vol. 8, pp. 22-31, 2009.

[4] H. Hile and G. Borriello, "Positioning and Orientation in Indoor Environments Using Camera Phones," *Computer Graphics and Applications, IEEE*, vol. 28, pp. 32-39, 2008.

[5] M. Werner, et al., "Indoor Positioning Using Smartphone Camera," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2011, pp. 1-6.

[6] T. Miyashita, et al., "An Augmented Reality Museum Guide," in *IEEE International Symposium on Mixed and Augmented Reality(ISMAR)*, 2008, pp. 103-106.

[7] K. Jongbae and J. Heesung, "Vision-Based Location Positioning using Augmented Reality for Indoor Navigation " *IEEE Transactions on Consumer Electronics*, vol. 54, pp. 954-962, 2008.

[8] L. C. Huey, et al., "Augmented Reality Based Indoor Positioning Navigation Tool," in *IEEE Conference on Open Systems (ICOS)*, 2011, pp. 256 - 260.

[9] Google, “Google Goggles: use pictures to search the web,” http://www.google.com/mobile/goggles.

[10] Kooaba, “Kooaba Visual Search: get instant product information,” http://www.kooaba.com.

[11] Nokia, “Nokia Point and Find: tag places and objects,” http://europe.nokia.com/services-and-apps/nokia-pointand-find.

[12] Amazon, “Amazon Remembers: create a visual list of products,” http://www.amazon.com/gp/remembers.

[13] D. G. Lowe, “Distinctive Image Features from Scale-invariant Keypoints,” *International Journal of Computer Vision*, Vol. 60, No. 2, pp. 91-110, 2004.

[14] Herbert Bay, *et al.*, "SURF: Speeded Up Robust Features," Features”, in *Computer Vision and Image Understanding (CVIU)*, vol. 110, No. 3, 2008, pp. 346-359.

[15] V. Chandrasekhar, *et al.*, “CHoG: Compressed histogram of gradients—A low bit rate feature descriptor,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Miami, FL, June 2009.

[16] David Chen,*et al.*, “Mobile Augmented Reality for Books on a Shelf”. *2011 IEEE International Conference on Multimedia and Expo*, pages 1–6, July 2011.

[17] M. Y. Hsieh and W. H. Tsai, “A Study On Indoor Navigation By Augmented Reality And Down-Looking Omni-Vision Techniques Using Mobile Devices,” in *Computer Vision,Graphics, and Image Processing*, Aug 2012.

[18] T. Akenine-Moller, E. Haines and N. Hoffman, "Perspective projection," in *Real-Time Rendering*, 3rd Edition, T. Akenine-Moller, ed., 2008, pp. 92-97.

[19] G. Klein and D. Murray, “Parallel tracking and mapping for small AR workspaces,” *Proceedings of 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, Nara, Japan, pp. 225–234, Nov. 13-16, 2007.

[20] P.A. Viola, M.J. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Computer Vision and Pattern Recognition (CVPR)*, pp. 511–518, 2001.

[21] D. Lowe, “Distinctive image features from scale-invariant keypoints,” in *International Journal of Computer Vision (IJCV)* 60 (2) (2004) 91–110.