國立交通大學

多媒體工程研究所

碩士論文

利用多部KINECT建構環車3D行車紀錄器及其應用

Construction and Applications of a 3D Event Data Recorder Using Multiple KINECT Devices around a Car

研究生：張揚

指導教授：蔡文祥　教授

中華民國一百零二年六月

利用多部 KINECT 建構環車 3D 行車紀錄器及其應用

Construction and Applications of a 3D Event Data Recorder Using Multiple KINECT Devices around a Car

研 究 生：張揚　　Student：Yang Chang

指導教授：蔡文祥　　Advisor：Wen-Hsiang Tsai

國 立 交 通 大 學

多 媒 體 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Multimedia Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2013

Hsinchu, Taiwan, Republic of China

中華民國一百零二年六月

Construction and Applications of a 3D Event Data Recorder Using Multiple KINECT Devices around a Car

Student: Yang Chang Advisor: Wen-Hsiang Tsai

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

ABSTRACT

In this study, a 3D imaging system is constructed for use as an event data recorder by affixing multiple KINECT devices to the body of a vehicle. The position and orientation of each KINECT device is different, and totally the views of the 14 KINECT devices in the system cover the 360^{o} car surround. To speed up the data recording work, two computers are used, each controlling seven KINECT devices.

To implement the proposed system, at first a method for creating a 3D model of around-car objects is proposed. This method is based on the pinhole camera model. The constructed around-car object model can be rendered in the 3D space and seen from specific views.

Secondly, a method for calibration of the relationship between neighboring KINECT devices using any calibration target is proposed, which is based on matching the target in a pair of acquired KINECT images using the iterative closest point (ICP) algorithm. Moreover, the calibration process is speeded up by the use of some learned information.

In addition, a method is proposed for merging the images acquired from multiple KINECT devices around the car. With the relationship parameters acquired during the

calibration process as inputs, the method transforms the coordinates between each pair of neighboring KINECT devices to accomplish the merge work.

Also, a method is proposed for panoramic image creation by stitching multiple color images into a single image. The stitching work is carried out by an automatic stitching algorithm. The created panoramic image can be used as a background in the displayed scene because the 3D image data constructed by the system sometimes are not enough.

Finally, a method for combining 3D images with the 2D panoramic image is proposed. Specifically, the panoramic image is projected onto a spherical shape which, together with the 3D around-car object model, is rendered for inspection of the geometric relation between the nearby object and the background scene.

Good experimental results are also shown, which prove the feasibility of the proposed methods for real applications.

利用多部KINECT建構環車3D行車紀錄器及其應用

研究生：張揚　　指導教授：蔡文祥 博士

國立交通大學多媒體工程研究所

摘要

本研究利用架設在車輛上的多台 KINECT 感測器，來建立三維取像系統，做行車紀錄器之應用。每台 KINECT 感測器的位置都不盡相同，因此所有使用的 14 台 KINECT 感測器的視野加起來可以涵蓋 360°的車子週遭範圍。為了在紀錄時加快速度，本研究使用了兩台電腦，每台控制 7 台 KINECT 感測器。

為達到建構 3D 行車紀錄器的目的，首先，本研究提出一個建立三維環車近景模型的方法，該法是建立在針孔成像的原理上。所建三維還車近景模型可以借由三維空間顯像技術，由各個不同的角度觀看。

第二步，本研究提出了一個校正兩台相鄰 KINECT 感測器關係的方法，該法是利用最近點迭代演算法來進行校正工作，並利用一些預先學習的資訊，來進行加速。

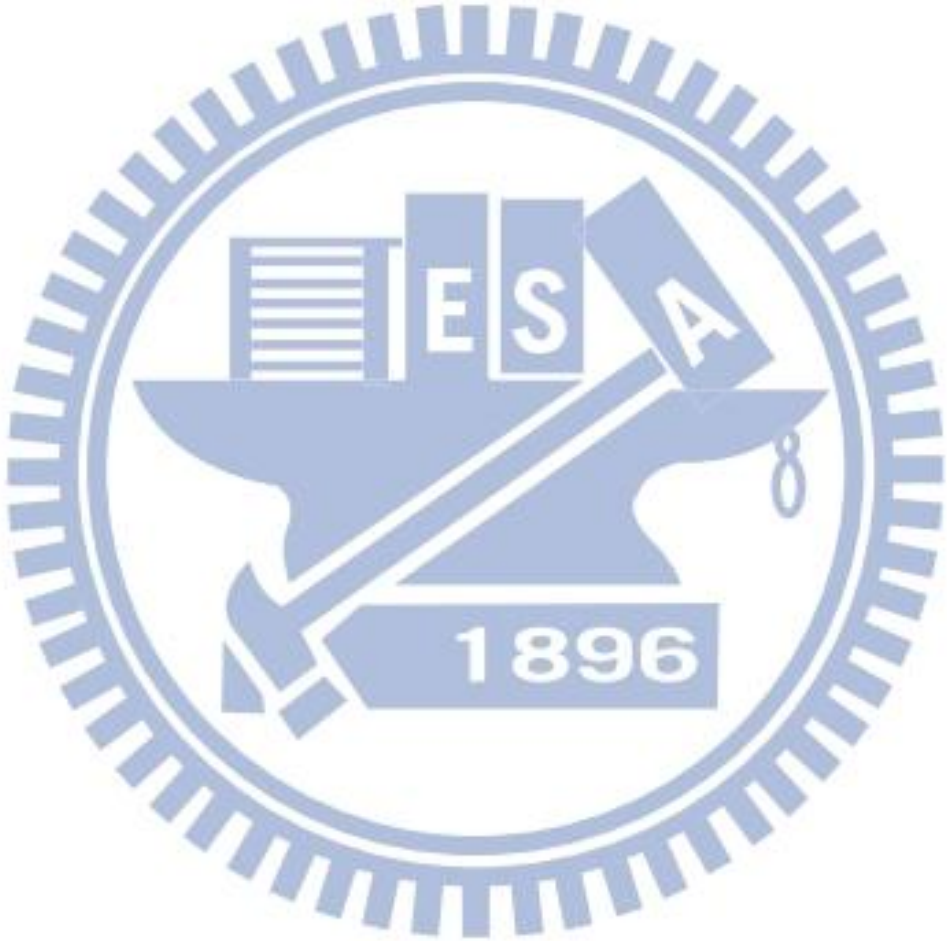
第三步，本研究提出了一個整合車輛上多台 KINECT 感測器所得資料的方法，該法是利用校正所得兩台 KIMECT 感測器的相對關係參數，對每對相鄰 KINECT 感測器所得之三維近景模型作座標轉換。

第四步，本研究提出了一個將多張由環車 KINECT 感測器所擷取之彩色影像連接成一張全景圖的方法，該法是基於一自動接圖演算法。因為室外所得到的深度資訊是不夠多的，所以本研究將環車彩色影像串接出來的全景圖，作為 3D 行車紀錄器的背景。

第五步，本研究提出了一個將三維環車近景模型與背景圖整合的方法。該法

是將背景圖投影到球體上，與三維環車近景模型做幾何的對應，因此在近的地方有三維環車近景模型，而在遠的地方則顯示出背景的部分。

最後，上述方法的實驗結果皆甚為良好，顯示本研究所提出的系統確實可行。



CONTENTS

LIST OF FIGURES

Chapter 1
Introduction

1.1 Background and Motivation

The event data recorder (EDR) of a car, which is similar to an aircraft's black box, is used by people to record the around-car information "seen" during car driving in a trip, which includes the traffic conditions and possibly car accidents along the way. Such records of "driving history" can be used for judgments of car encountering or collision conditions when legal cases arise. For example, according to a news report, a boy who was nineteen years old drove a jip bump to hit a Rover sport utility vehicle, causing an accident in which only a paralyzed baby survived. The data recorded by the EDR was used to re-build the situation at the car collision instance, finding out that a force equivalent to 42mph was applied in one fifth of a second in the crash. This helped the police to put the defendant's speed at around 72mph, and it is the first time such technology plays a role in the British court.

The EDR has become more and more popular in recent years. According to the evaluation of National Highway Traffic Safety Administration (NHTSA) of the USA, it is predicted that more than 85 percents of the cars are equipped with EDRs whose capabilities become more and more diversified and powerful due to the market growing trend and the technology advance. Some kinds of high-end EDRs are equipped with wide-angle cameras and provide high-resolution videos, offering capabilities of "seeing wider and clearer." Moreover, some even better high-end EDRs are designed to include the GPS and dual cameras, the latter being used at the front

and rear of the car. This helps people to get a more complete view around the car when one drives to a place for the first time. Furthermore, such intelligent EDRs also help saving power and storage because they start recording only if the view is changing.

In addition, with the advance of 3D vision technology, 3D cameras, 3D phones, 3D movies, 3D TVs, etc. appear one after another in fast speeds. These goods enrich our life in different ways, helping us to "show" the world on various types of displays from 2D ways to 3D ones. These 3D goods inspired us to create a 3D EDR. In this study, we try to build a 3D EDR system with multiple KINECT devices for use on a car to record around-car scenes from different views, so that complete 3D information around the car can be constructed.

Furthermore, when a car accident occurs, the proposed 3D EDR system can help re-building the scene at the accident moment as a 3D version, called a *3D scene*. Afterwards, we can browse the 3D scenes in sequence. In this way, we can see the scene just like you were there and holding a camera inside a car. For example, to see how a car bumps onto another, we can turn the 3D scene to the lateral side to check the situation between the two cars at critical moments, or to see the entire course of the event. This helps us to clarify the car-accident responsibility easier than using the record of a traditional "2D" EDR.

Furthermore, the 3D EDR is a tool useful not only for protecting people's right in car accidents, but also for recording journeys by car driving in the outdoor environment. By the use of the recorded information, people may look back at beautiful scenes, check the shapes of luxury cars driven around, view gorgeous ladies passing by, and so on, all in 3D ways, to enrich our "life of technology!" In this study, we try to design a 3D EDR using around-car KINECT devices and develop techniques to deliver 3D information for this goal.

1.2 Review of Related Works

About researches related to 3D device systems, Wilson and Benko, et al [1] proposed an idea of LightSpace created by the research team of Microsoft, and proposed accordingly a smart image display system for use in the office, for which they used three depth sensors and three projector sets installed on the ceiling (as shown in Fig.1.1). Special touch screens or monitors were not used. The depth image which is acquired by the depth camera is transformed into the real-world coordinate system after calibrating each depth camera of the system, so that they could interact with the user in ways such as moving an image from the table to the white board with a user's hand sign. That means he/she can use a hand to control many complex instructions implemented by integration of the projector and depth cameras in a usual way.

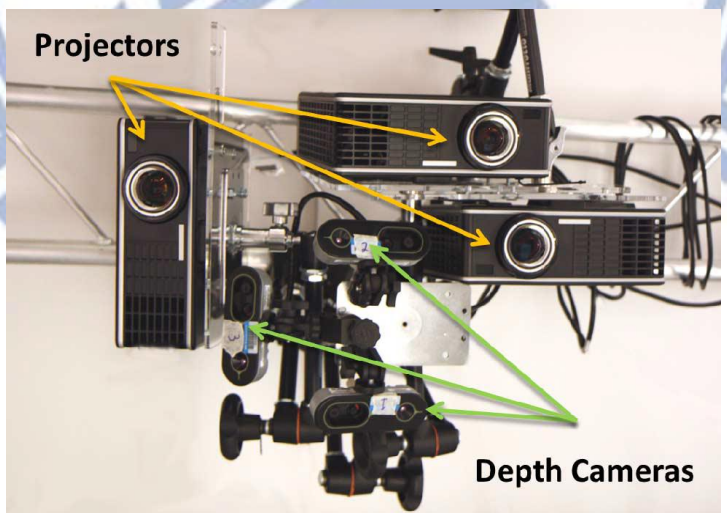


Fig. 1.1 The system used in LightSpace.

Some other systems using different devices for building 3D models have also been proposed. Biber, et al. [2] proposed a robot equipped with a laser distance meter

and an omni-camera to get depth information of the real-world environment, extract the information of walls by using the laser distance meter to solve the SLAM problem, and mix up multi-size textures to hide the seams between images. Henry et al. [3] used color and depth images acquired by the KINECT device to propose a method for building a complete 3D mapping system by combining visual features.

About other applications using the KINECT device, a team of the MIT, the University of Washington, and the Intel Lab. at Seattle [4] put KINECT devices on a light aircraft to build a multi-view integrated 3D model of an environment. An example is shown in Fig 1.2. In their method, the feature points in the images acquired by the KINECT devices were extracted and a so-called RGBD-SLAM algorithm was used to achieve their goal of environment modeling.

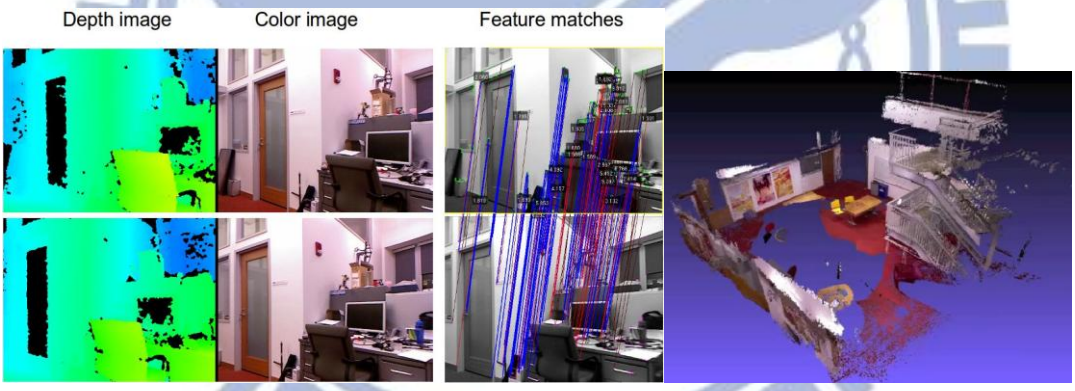


Fig. 1.2 A 3D indoor environment model constructed by [4] from KINECT images.

In seeking a method for building 3D environment models, Sharam Izadi, et al. [5] proposed the concept of KINECT fusion and designed accordingly a system that takes live data from a moving KINECT device and creates high-quality and geometrically accurate 3D models in realtime. An example of their result is shown in Fig. 1.3. They tracked the previous frame and the current frame to compute a rigid 6DOF transform that closely aligns the currently-oriented points with those of the previous frame,

using a novel GPU implementation of the ICP algorithm.

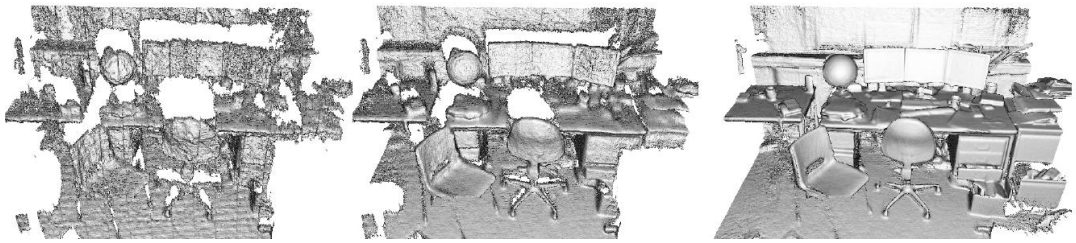


Fig. 1.3 Geometrically precise 3D models of a room constructed by [5].

1.3 Overview of Proposed Methods

In this study, we use KINECT devices around the car to build our 3D EDR system. The concept of this idea is illustrated in Fig. 1.4.

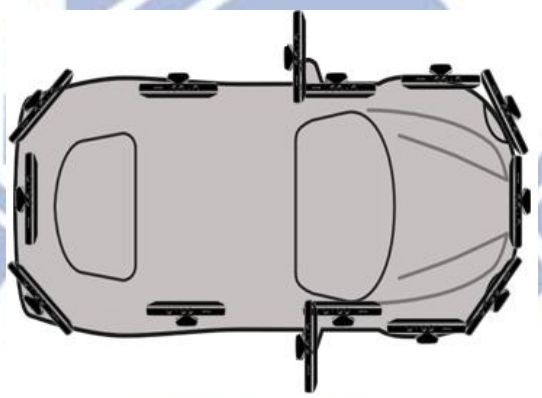


Fig. 1.4 Illustration of proposed system with multiple KINECT devices.

As shown in Fig. 1.5, before acquiring data by the KINECT devices, we measure manually roughly the geometric relation between every two neighboring KINECT devices. Afterwards, we develop a method for finding more precise inter-KINECT relations based on an ICP algorithm, using the hand-measured data as initial values

for iterations conducted by the algorithm.

Then, we use a method of coordinate conversion similar to that described in [1] to transform each pixel of the depth image into the real-world coordinate system to construct a 3D image. In the process, a cluster of points are taken to correspond to a single real point existing in the real world. Because the number of points obtained from the KINECT device is pretty large, it is important to reduce the amount of data while keeping the shape of each object good enough without deformations. For this purpose, appropriate data structures have been designed to implement the reduction work because the real-time data storing format is different from the structures we need usually.

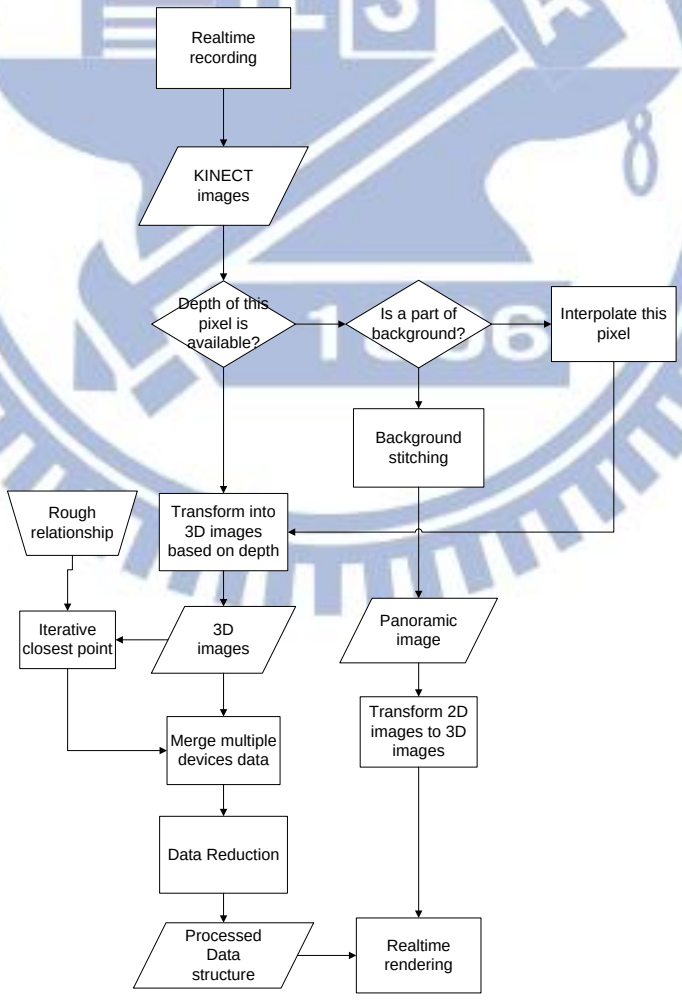


Fig. 1.5 Major tasks of proposed KINECT-based around-car EDR system.

Because of the restriction of the range of KINECT devices, depth data out of 6m are not available, but we still want to see some beautiful scene at far distances when we are driving. To satisfy this desire, the long-range views of the surrounding scene provided by the color images are stitched in this study to produce a panoramic image automatically. Furthermore, this panoramic image is attached to a part of a sphere for the purpose of fitting the nearby data obtained from the KINECT devices.

Finally, we can browse the generated 3D images and panoramic views in a form like video from both near and far views, just like browsing a Google Street View in a 3D version. At every instant of the generated video sequence, we can see the 3D objects around us and the scenes at far distance just as if we are standing on the street (but in fact we are standing inside a sphere).

1.4 Contributions

Some contributions of this study are listed in the following.

1. Constructing 3D images by color and depth images acquired with KINECT devices
2. Constructing 3D images for a subset of the KINECT devices and combining them into a single 3D image for each of some specific views (front, lateral, rear, etc.).
3. Constructing a complete around-car 3D image without blind spots from the 3D images corresponding to all the KINECT devices respectively on the car.
4. Combining the long-range color-image views into a mosaic panoramic image for use as the far-view background image.
5. Providing a realtime browsing system with a series of offline processes for users' browsing of the panoramic background image.

6. Combining near objects and far views around the car to construct 3D images like street views for each time instant for the user to browse.

1.5 Thesis Organization

The remainder of this thesis is organized as follows. In Chapter 2, we introduce the configuration of the proposed system and the system processes in detail. In Chapter 3, we present the design of the proposed 3D around-car imaging system. In Chapter 4, a method for constructing 3D images is described. In Chapter 5, we introduce the proposed method for modeling around-car objects, including calibration and merging multiple KINECT devices. In Chapter 6, the proposed method for browsing a 3D image including integrated long-range views and nearby objects is described. In Chapter 7, experimental results and discussions are presented. Finally, conclusions and some suggestions for future works are given in Chapter 8.

Chapter 2
System Design and Processes

2.1 Ideas of Proposed System

In order to build a complete panoramic view around the car, we affix KINECT devices with different orientations to the car body at different locations. We try to consider the use of the minimal number of KINECT devices to cover the entire car body with a 360-degree surround. With the horizontal angle of a KINECT device being 57 degrees, the minimal number of KINECT devices required to accomplish a complete covering of 360^{o} would be 7, but our final design of the 3D around-car system uses 14 KINECT devices as mentioned in Chapter 1. The reason and more details of the design will be described in Chapter 3.

With the 14 KINECT devices, we can acquire depth and color images through universal serial buses (USBs). Since each KINECT device needs one distinct USB controller, we need 14 USB controllers for the proposed system. In turn, we need a special computer with at least 14 USB controllers. But such a computer is not available commercially. Therefore, we insert a USB extension card onto the mother board of a common desk-top computer for use in this study. But the computing speed becomes so slow that we decided to use two computers to build our system, with each computer containing seven USB controllers. Analysis of the resulting processing speed will be described in Chapter 3.

In Section 2.2, we will describe the configuration of the proposed system, including the hardware and the software. The processes conducted by the proposed

system, which include a learning process and a data recording and analysis process, will be described in Section 2.3.

2.2 System Configuration

To build the proposed 3D imaging system on the car, we affix 14 KINECT devices on the car as mentioned before. Color and depth images are acquired with the KINECT devices via the OpenNI which is an open source software for use as the device driver. In Section 2.2.1 we will review the functions of the KINECT device, and describe how and where we affix the 14 KINECT devices on the car. In Section 2.2.2, we will introduce system development environment for this study, and the functionality of each component in the system.

2.2.1 Hardware Configuration

At first, we review the structure of the KINECT device as shown in Fig. 2.1. Inside a sensor case, a KINECT device contains an infrared (IR) emitter, an RGB camera, an IR depth sensor, and a tilt motor to control the tilting angle of the KINECT device.

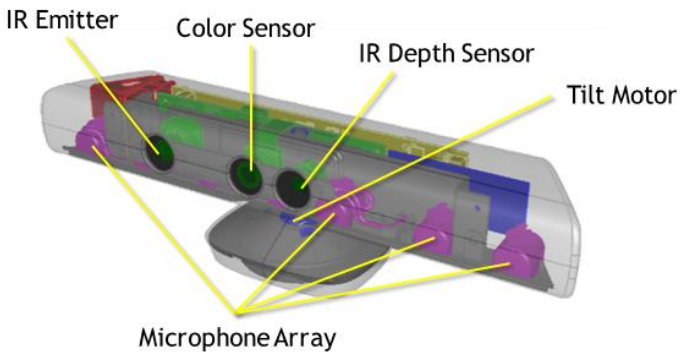


Fig. 2.1 The structure of a KINECT device.

The functions of the components of the KINECT device are explained below.

1. An infrared (IR) emitter and an IR depth sensor —The IR emitter emits light to objects in the environment, and the IR depth sensor reads the reflected light from the objects to compute the distances between the objects and the KINECT device by converting the times of flight of the reflected light rays into depth values.
2. An RGB camera — The camera in the KINECT device senses image data of three color channels R, G, and B.
3. An accelerometer — The accelerometer is conFig.d for sensing a 2G range (G is a unit of acceleration due to gravity), by which, we can know the current operational condition of the tilter (i.e., the tilt motor).
4. A multi-array microphone — This device contains four microphones which may be used to record the sound and know the direction of the sound.

Some more specifications of the KINECT device are listed in Table 2.1. Especially, the view angle, the tilt range of the device, and the range of the sensed depth values are important for our design of the proposed system. We will discuss these parameters in more detail later in Chapter 3. Furthermore, some specifications of the desktop computer used in this study are listed in Table 2.2.

Table 2.1 Specifications of the KINECT device.

Horizontal viewing angle	57 degrees
Vertical viewing angle	43 degrees
Tilt range of the device	±27 degrees
Range of the depth sensor	1.2-3.5 meters
Resolution of color images	640 x 480
Resolution of depth images	320 x 240

Table 2.1 Specifications of the desktop computer used in this study.

Processor	Intel Core i7-3770
Memory	16 GB
Mother board	GA-Z77X-UD4H-1
graphics card	GV-R7750C 2GI
PCIE USB 3.0 extension card (AISYS Vision)	Aguila SU16T base x 1 Aguila SU16T expansion x 1

Fig. 2.2 The USB extension card (the upper is the expansion and the lower is the base)

The Aguila SU16T base mentioned in Table 2.1 and shown in Fig. 2.2 contains four USB controllers, and the Aguila SU16T expansion also contains four USB controller as an expansion of the base. We use two computers with each holding an Aguila SU16T base and an Aguila SU16T expansion. Thus each computer can allow connections to eight USB controllers, but in our study, only seven USB controllers in a computer are used. In addition, we use a ferrous box and affix it on the car to hold each of the KINECT devices. Two examples are shown in Fig. 2.3.

(a)

(b)

Fig. 2.3 Ferrous boxes for holding KINECT devices. (a) Without a KINECT device. (b) With a KINECT device.

2.2.2 Software Configuration

About the software configuration, firstly we review the OpenNI which is an open source software for development of 3D sensing middleware libraries and applications as shown in Fig.2.4. It provides a tool for getting depth and color images from KINECT devices, or in other words, it provides an interface to communicate with the hardware and the computer.

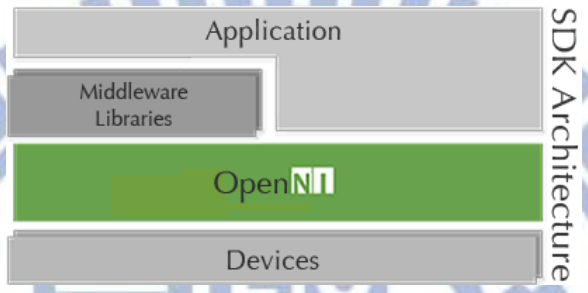


Fig. 2.4 illustration of the role OpenNI plays.

Secondly, the OpenGL is an application programming interface (API) for generating 2D and 3D images. This API is typically used to interact with a GPU to achieve hardware-accelerated rendering. In our case, the OpenGL is a tool for rendering 3D data which are obtained from transforming the color and depth images. The details will be described later.

Finally, we use the Visual Studio 2010 as a development environment for integrating multiple kinds of libraries, such as OpenNI and OpenCV. Therefore, we can write programs using the C and C++ languages and the libraries to conduct works of image processing, rendering, …, etc. on this software platform.

2.3 System Processes

2.3.1 Learning Process

Before the data recording process, a learning process is necessary. We divide the learning process into two parts as follows.

The first part is the process for system calibration, including the calibrations of the KINECT devices and some system parameters. The first task to be done in this process is to find out the height of each KINECT device with respect to the road in the 3D image. For this, we have to transform the depth and color images acquired by a KINECT device into a 3D image at first. Then, we find the desired height by a try-and-error manner using the 3D image; when an appropriate height parameter is found, we can use it for next steps.

Secondly, we have to calibrate the geometric relation between every two neighboring KINECT devices. For this, we use a box with a simple shape as the calibration target, and take the previous result, the height of each KINECT device, to find the calibration target out in precise. The result of this process is the relative angle between every two KINECT devices, which can be used in the data recording and analysis process. A flowchart of this part of the learning process is shown in Fig. 2.5, and the detail of calibration is described in Chapter 5.

The second part is the process for stitching of multiple images to construct a long-range panoramic view. In this processing, at first we want to learn a threshold value to separate far-view contents from near-view ones in each color image which is taken by the KINECT device. The work is completed by try-and-errors. The resulting set of far views can then be combined together to get a panoramic view by a stitching process which will mentioned in more detail in Chapter 6. A flow chart of this part of

the learning process is shown in Fig. 2.6.

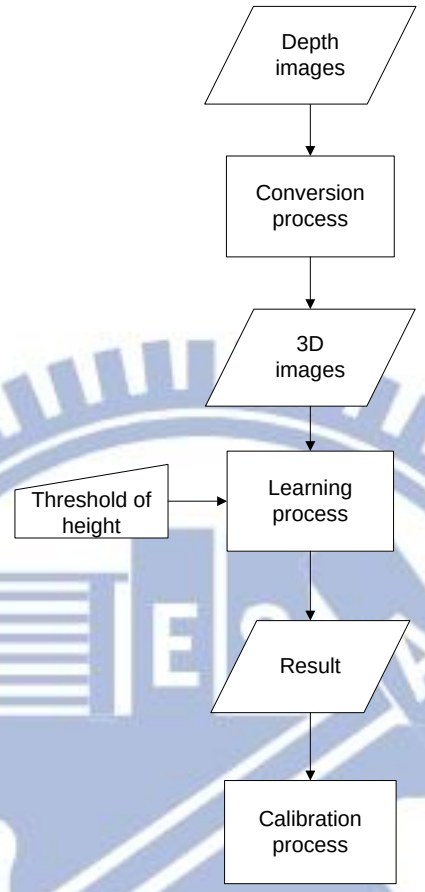


Fig. 2.5 The learning process for system calibration.

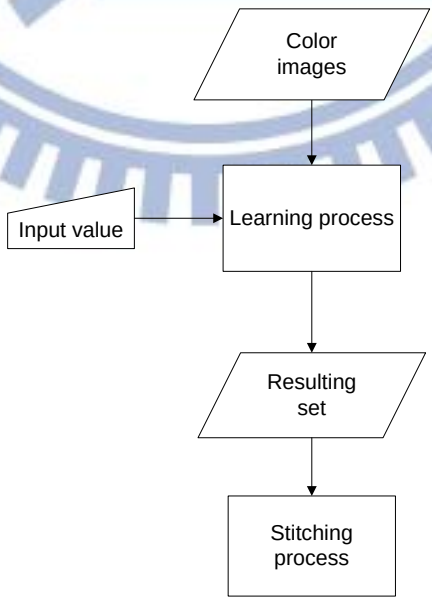


Fig. 2.6 The learning process for image stitching.

2.3.2 Data Recording and Analysis Process

In the data recording and analysis process, two computers are in use as the controllers of the 14 KINECT devices and are connected by a cable. They are of a master-slave structure, as shown in Fig. 2.7. The software implementation is based on a client-server architecture which we use a windows socket to conduct the inter-computer communication.

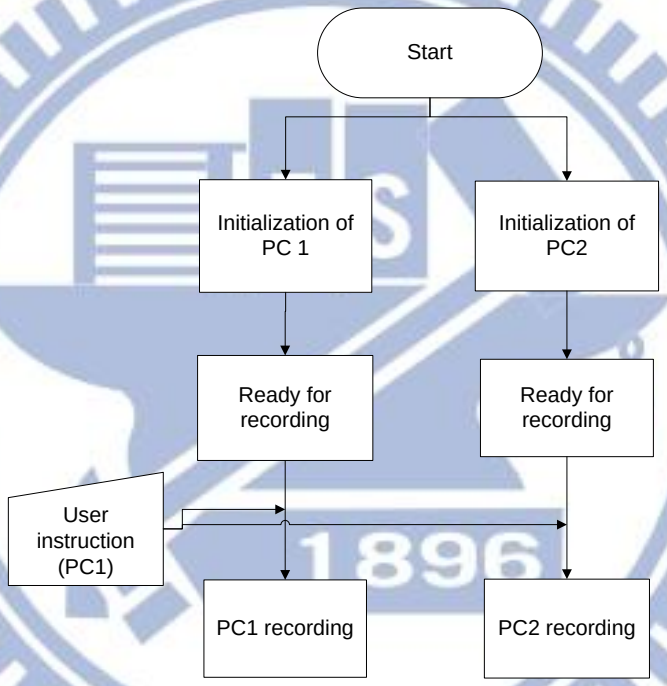


Fig. 2.7 Master-slave structure of proposed data recording and analysis process where PC1 is the master computer and PC2 is the slave computer.

At the beginning of the data recording and analysis process, as shown in Fig. 2.8, at the client side, the master receives an instruction from the user, and then sends a request to the slave at the server side for recording. The slave starts the recording process and returns a reply to the master after getting the request. While receiving a reply from the slave at the server side, the master starts to run the recording process,

too. All the communications are implemented by multi-thread, because we have to communicate two devices and record at the same time. The ending of the data recording and analysis process is shown in Fig. 2.9.

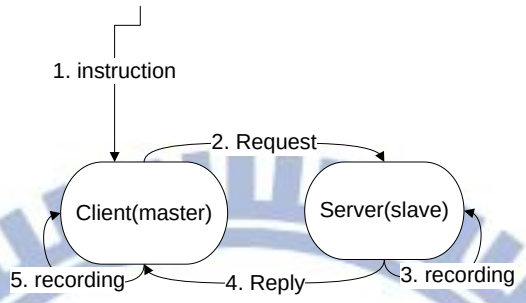


Fig. 2.8 Starting the data recording and analysis process.

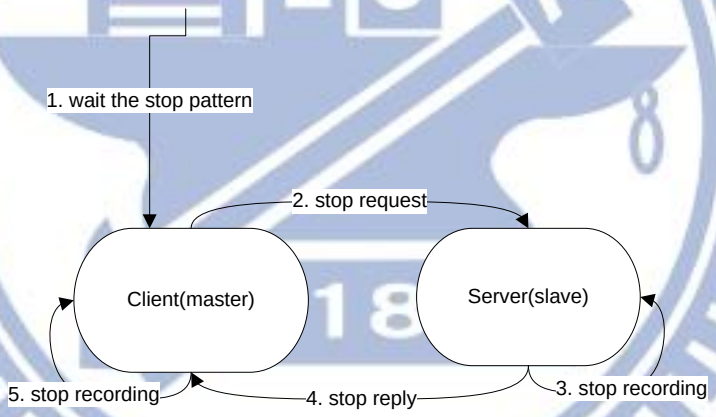
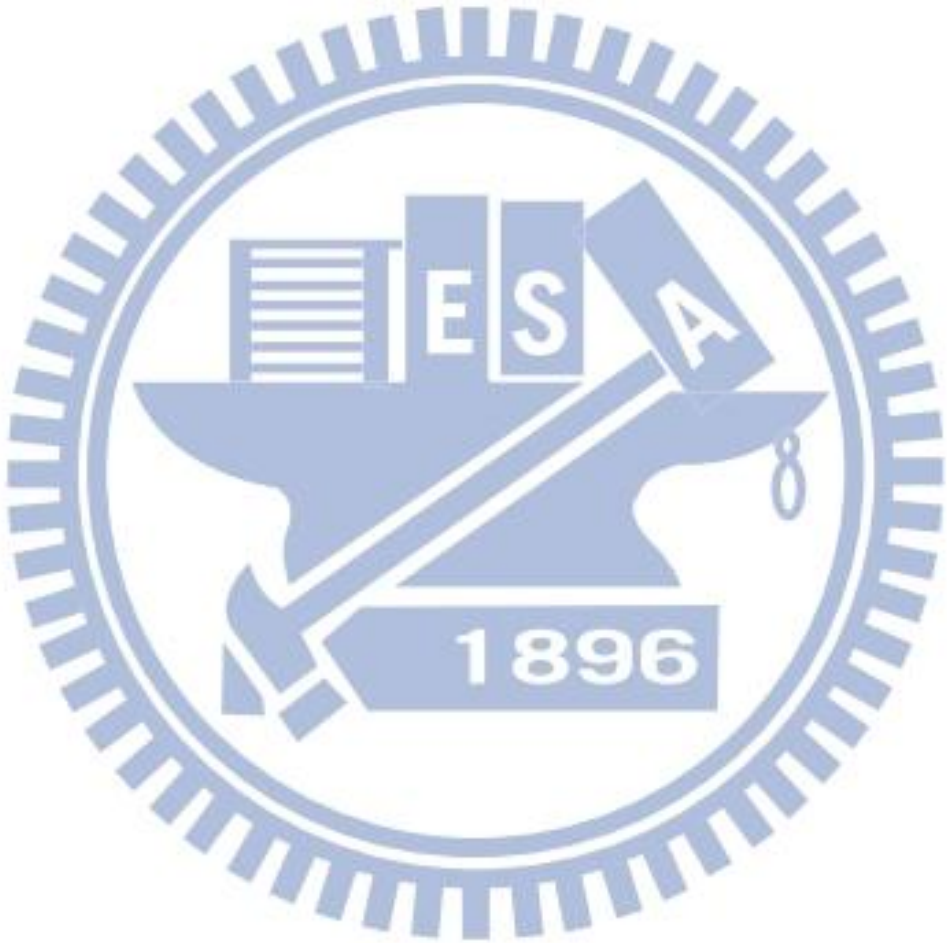


Fig. 2.9 Ending the data recording and analysis process.

Furthermore, a series of tasks are conducted in the data recording and analysis process as described in Section 1.3, including (1) transforming color and depth images acquired by each KINECT device at each time instant into a 3D image; (2) stitching all the color images into a panoramic color background image; (3) extracting nearby 3D objects from the 3D image corresponding to each KINECT device; (4) merging the extracted 3D objects into the panoramic color background image; and (5) allowing

the user to browse the merging result from any viewpoint and display the corresponding partial view.



Chapter 3
Design of Proposed 3D Around-car Imaging System

3.1 Idea of Proposed 3D Around-car Imaging System

When constructing a 3D EDR, it is important to let the EDR “see” the view around the car with no blind spot. It is obviously not enough to use only one KINECT device whose horizontal angle range is 57 degrees only. Instead, we have to affix multiple KINECT devices around the car. In addition, the way of design for this system is different for each distinct part of a car. Before starting the description of the proposed system design, we give a brief review of the design of a car model with multiple cameras produced by Luxgen Motor Co. Ltd.

Luxgen has released a new car equipped with six RGB cameras around the car body. They are called “eagle views.” A camera is affixed to the front of the car, and another to the rear. The remaining four cameras are affixed below the side mirrors, with each side equipped with two cameras, one camera facing to the rear of the car, and the other is facing askew to the rear as shown in Fig. 3.1.

Fig. 3.1 The cameras affixed on the body of the car (a) (b) front part of the car (c) (d) side part of the car (e) (f) rear part of the car (g) (h) the recorder on the mirror.

This design inspired us, because we can affix cameras on the side mirror. With these cameras, we can see the around-car view like through the window of the car. If someone gets close to the car near the window, it would be found by the nearby KINECT device.

About the coverage of the front view of the car, Luxgen uses only one camera to cover the front view, because the camera is of the fisheye type which yields a wider view than from a normal projective camera. The overlapping portion of the front view and the side view is narrow, and so there exist four bind spots on the corner. To improve it, we use four additional KINECT devices to cover the corner views in our design. Specifically, to cover the front-left corner, a KINECT device with its view covering the portion of (a) as shown in Fig. 3.2 is deployed. A second KINECT device is used to cover the right symmetric portion in the front. For the rear-left corner, a third KINECT device with its view covering the portion of (c) is deployed. And the fourth KINECT device is used to cover the right symmetric portion in the rear.

About the other deployed KINECT devices, we affix three KINECT devices for the front view, four for each side view, and three for the rear view. In addition, we affix a KINECT device on each of the two side mirrors which looks backward to cover the view portion of (B) as shown Fig. 3.2. More details will be described in Section 3.3.

3.2 Details of System design

3.2.1 Front Part

In this section we will focus the part of our design which is related to the driver's views. Specifically, we will affix KINECT devices to proper car body parts to cover "blind spots" that the driver cannot notice during driving, for example, the part of the

front view which is lower than the height of the engine hood.

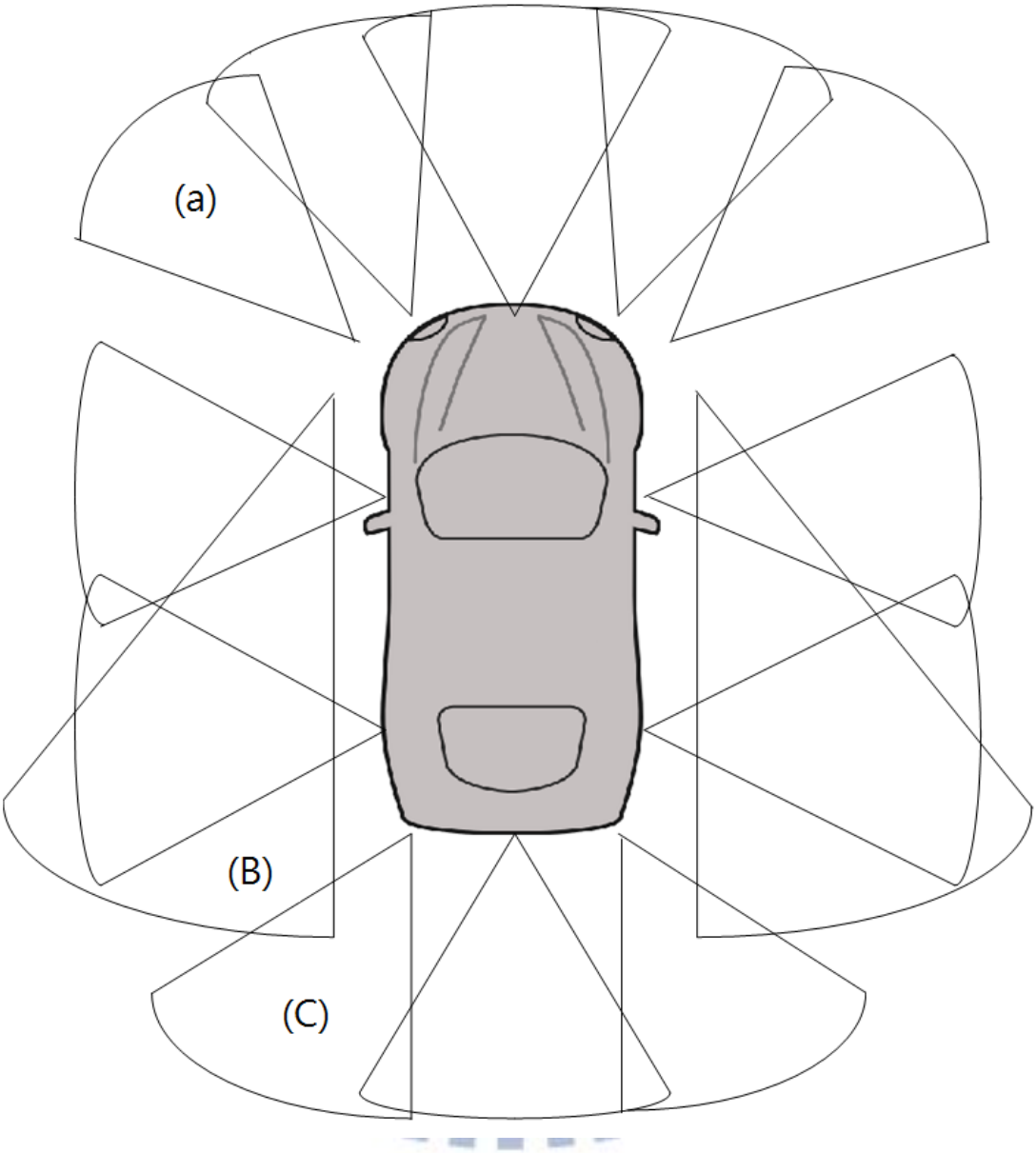


Fig. 3.2 Proposed design of the KINECT-device system affixed on the vehicle and the views of the KINECT devices.

When one drives a normal car in the street, the car might accidentally run over a dog or some animal. In this situation, the driver won't know what is happening because what is going on is within the blind spots of the car. Using the car with our

design of the multiple KINECT device system, the blind spots can be eliminated and such accidents can be avoided. Moreover, blind spots seen from a driver on the truck are much larger than those of a usual vehicle, so proposing a design to cover completely the surround of a car is really important. And this is done in this study.

Also, we affix three ferrous boxes on the bumper (as shown in Fig. 3.6(a)) for the maximal utilization of the KINECT devices. This allows us to see the region under the engine hood, as shown in Fig.3.3. The yellow object is used to tag the limit of driver's view (below or closer than this object would not be seen in the driver's view).

(a) (b)

Fig.3.3 A test for driver's view. (a) Side view. (b) Front view.

3.2.2 Right- and Left-side Parts

Originally, a KINECT device was put on the iron stand which is stitched on the car as shown in Fig. 3.4. This design was considered the maximal utilization of the depth information, which is available in the range from 0.5m to 6m according to our experiment. However, this design violates the law of car modification. Other by-passing cars will possibly be scratched by the iron stand while driving on the road, so this wasn't an appropriate design.

(a) (b)

Fig. 3.4. A car-side iron stand for holding a KINECT device. (a) With a KINECT device. (b) Without a KINECT device.

After this experience, a new design was developed with the KINECT devices affixed at the higher side rack on the car roof, as shown in Fig. 3.5. This design is considered to be safer and more convenient. The ferrous box preserves a position for each KINECT, so the position of each KINECT device will not change whenever we put KINECT devices back on the car.

(a) (b)

Fig. 3.5. Ferrous boxes for holding KINECT devices. (a) With a KINECT device. (b) Without a KINECT device.

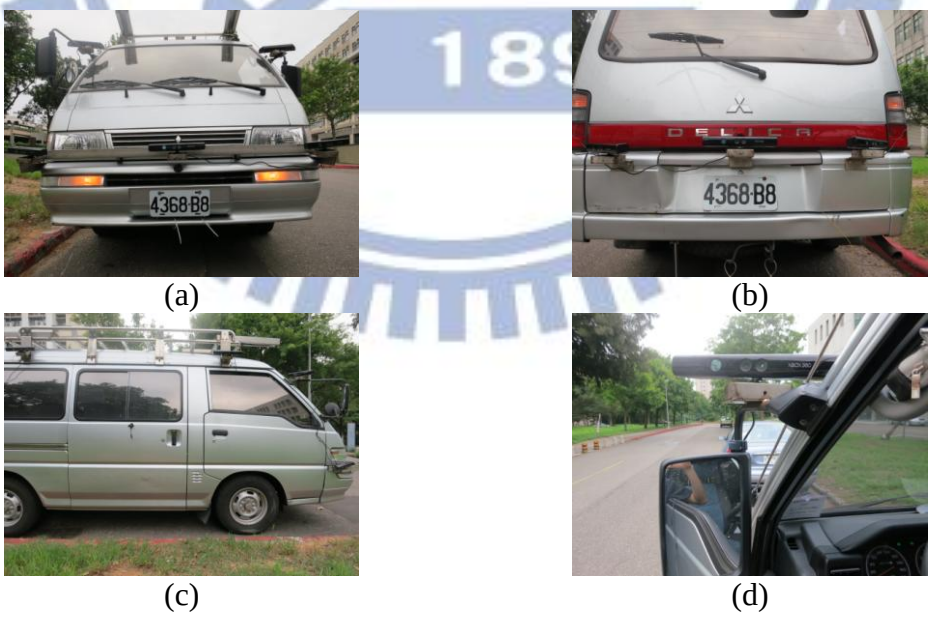
With this new design, the KINECT device is unmovable and safer when we are driving. Though the tilter of the KINECT device is movable, these two KINECT

devices are too high. To solve this problem, we affix two KINECT devices on the side mirrors to cover the lower view ranges as shown in Figs. 3.6(c) and 3.6(d)

3.2.3 Rear Part

In the previous part we used a ferrous bar and affixed the boxes on that. Contrastive to the previous part, because we don't have any space to put the ferrous bar, or there is some ferrous stand originally, we drilled a hole for fastening the screw and affixed the ferrous box as shown in Fig. 3.6(b). We use this method only as a last resort.

The rear part is similar to the part of the front view, but we have to consider the case when the car is driven backward. From common drivers' experience, the back view is known to be as a serious bind spot of the car, so we extend the view by using three KINECT devices (originally it was only one which is not enough). Furthermore, we use two KINECT devices on the side mirrors to cover more of the back view.



(a) (b)

(c) (d)

Fig. 3.6 Around-car KINECT devices (a) A front view. (b) A back view. (c) A lateral view. (d) A rear-view mirror.

3.3 System Performance Analysis

3.3.1 Ranges of Camera Views

With the proposed 3D images system, driving a car is like to carry a box. This box is used to collect 3D data. At each instance every view is dropped into this box which is a parallelepiped shape. The size of this box is an extension of the car plus 6m outward. After recording KINECT images and processing them in this study, we can see a 3D image around the car for every instance. However, the depth data of KINECT devices are partially available in outdoor environments due to the interference from the sun to the emitted infrared light of the device. Note that the sun has a full spectrum of light. But according to our experiments, we have found that the data acquired by the KINECT device still works during sunset time. Our experimental results about this aspect are shown in Fig. 3.7 which were collected in August. We know from this result that the size of the parallelepiped box mentioned above is floating, and the 6m range is just an ideal case.

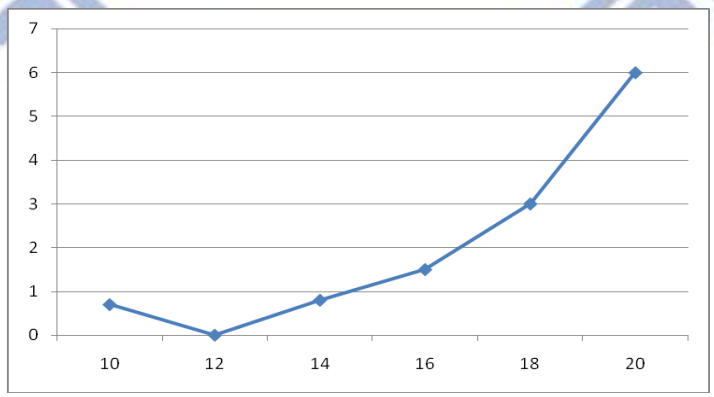


Fig. 3.7 The relationship between the depth image quality and the sun intensity (the *x*-axis specifies time, the *y*-axis specifies the available depth range).

Though this would be a bad news to our application, but we can still utilize the color images as the views in day time, and use the depth image to construct 3D models for night time, since the quality of depth images is pretty good in night. We improve the vision in the night by using depth images, and the night time is when traditional 2D EDRs do not work well.

3.3.2 Imaging Sequence and Speed

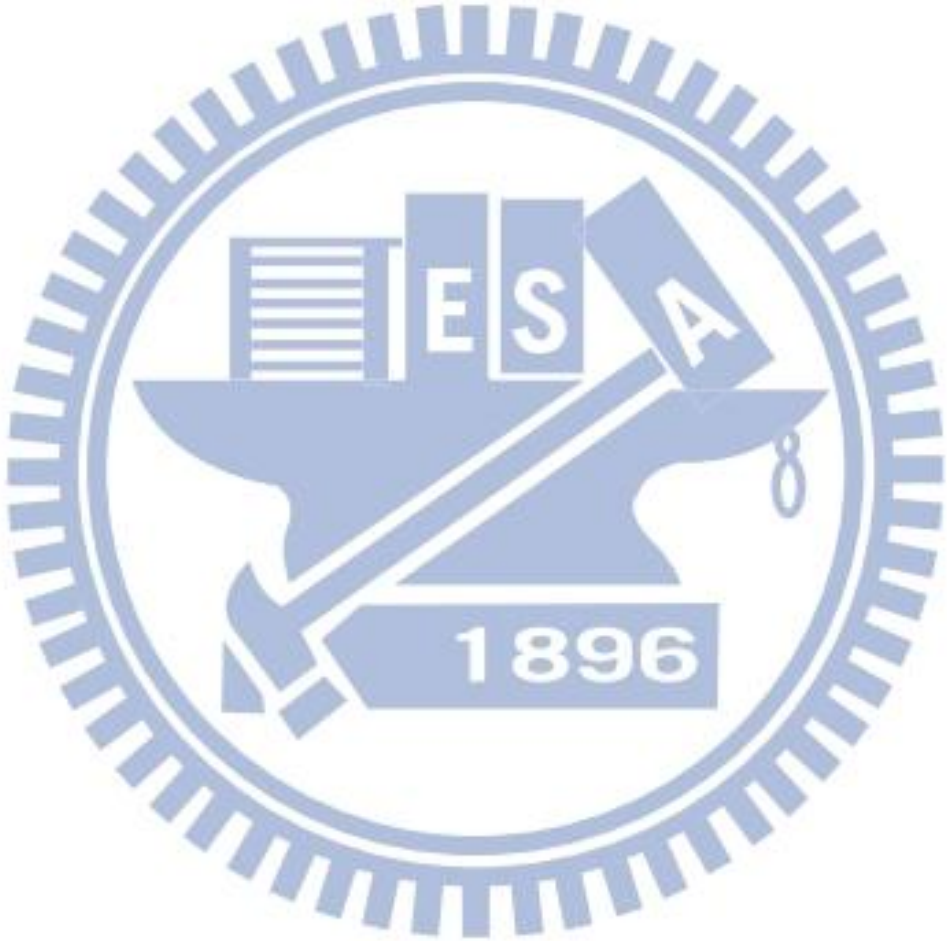
We use two computers to speed up our imaging speed. Two problems so arises. The first is the communication time, and the second is the synchronization of the rates of FPS (frames per second).

Firstly, we have to know the imaging speed of a single KINECT device. By reading the specification of the KINECT device, we know that the imaging speed is 30 FPS. In other words, we take a picture in 33ms. A single request from the master computer to the slave takes about 1 ms according to our experimental experience. Since the communication steps will not affect our imaging speed too much, we can conduct sequential processes with this speed for applications.

Secondly, we want to use a clock to synchronization the FPS rates of the two computers. When a signal is received in both the master computer and the slave, they start to count their time. In that way, each side may be controlled to take a picture at the same time (or we can say in a nearly identical time).

Finally, the last problem is the synchronization of the FPSs when car is moving. We allow 7 KINECT devices to be controlled by each computer. Take the sequential processing nature of the CPU, the imaging speed is so 33ms × 7 = 231ms, so the FPS is 1/231 ≈ 4.32. In other words, our car box mentioned above acquires a pair of KINECT device images in 231ms for *each instance*. Suppose that this car is driven slowly, just like a person working (a normal driving speed is 4km/hr = 111.11 cm/sec)

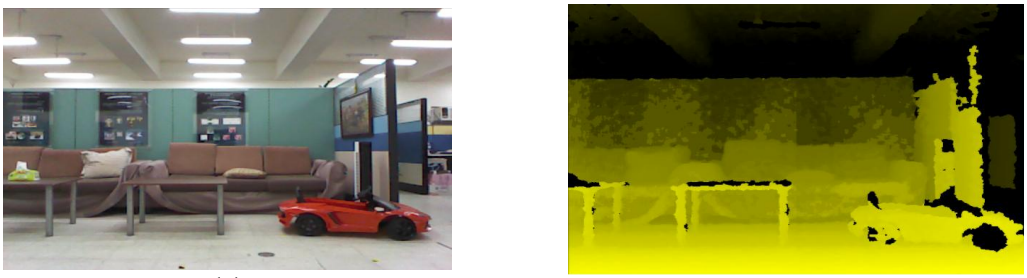
The *delay length* for each KINECT device coming from the delay of image acquisition time (i.e., 231ms) with respect to a proceeding KINECT device will be at most 111.11 × 0.231 = 25cm; and the delay length of the neighboring KINECT device will be 111.11×0.033= 3.67cm. By these parameters, we know that the car speed will not be a problem to our processing work.



Chapter 4
Construction of 3D Images from KINECT Images

4.1 Review of Structures of Depth and Color Images Taken by KINECT Devices

The data acquired with KINECT devices are of two types. One is the traditional color image, and we compress this type of image into the JPEG format. In this format, we can get a reasonable image quality and a good compression rate. The other is the depth image. It stores the distance between the objects in front and the KINECT device in the unit of pixel. Unlike the color image, the depth image can't be seen straightly. It is composed of many object distances, and we can see this image more properly by quantizing its values to be in the range of 0 to 255. After that, we can see a gray-level image that shows the distance to every object point from the camera view, as shown by the example in Fig. 4.1.



(a) (b)

Fig. 4.1 Images acquired with the KINECT device. (a) Color image. (b) Depth image.

4.2 Construction of 3D Images from KINECT Images

4.2.1 Review of Pinhole Camera Model

The pinhole camera model describes the relationship between the coordinates of a 3D point and its projection onto the image plane of a pinhole camera, where the camera aperture is a point as illustrated in Fig. 4.2.

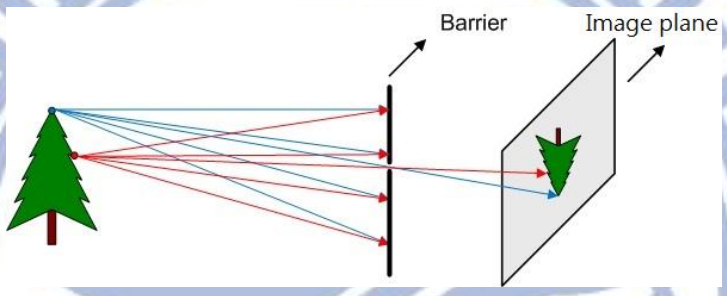


Fig. 4.2 A tree is projected onto the image plane through a pinhole model.

The geometry of the pinhole camera model may be illustrated by Fig. 4.3, which includes the following components.

1. A 3D orthogonal coordinate system with its origin at O. The three axes of the coordinate system are X_1, X_2, and X_3. A point P somewhere in the world is specified by coordinates (x_1, x_2, x_3) with respect to the X_1-, X_2-, and X_3-axes.
2. The image plane is parallel to the X_1- and X_2-axes. The image center is denoted as R.
3. The projection of a space point P onto the image plane is denoted as Q. This point Q is just the intersection of the projection line (green) "emitted" by P and the image plane.

4. There is also a 2D coordinate system in the image plane, with its origin at R and its Y_1- and Y_2-axes parallel to the X_1- and X_2-axes, respectively. The coordinates of point Q in this coordinate system are denoted as (y_1, y_2).

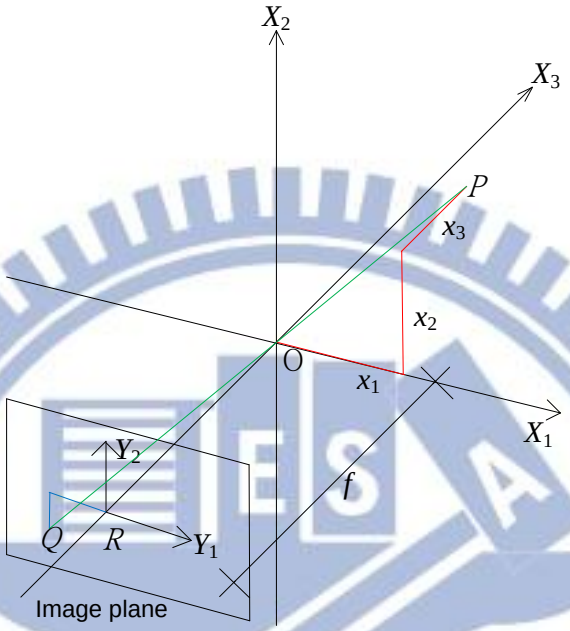


Fig. 4.3 The geometry of a pinhole camera.

Next, we want to derive transformations between the coordinates (y_1, y_2) of point Q and the coordinates (x_1, x_2, x_3) of point P. In Fig. 4.4, we see two similar triangles from which the following two equations can be derived:

$$\frac{-y_1}{f} = \frac{x_1}{x_3}; \tag{4.1}$$

$$\frac{-y_2}{f} = \frac{x_2}{x_3}. \tag{4.2}$$

Summarizing Equations 4.1 and 4.2, we get a vector equality as follows:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = -\frac{f}{x_3}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}. \tag{4.3}$$

With the above equation, we can construct 3D images. The proposed method for this purpose will be explained in the next section.

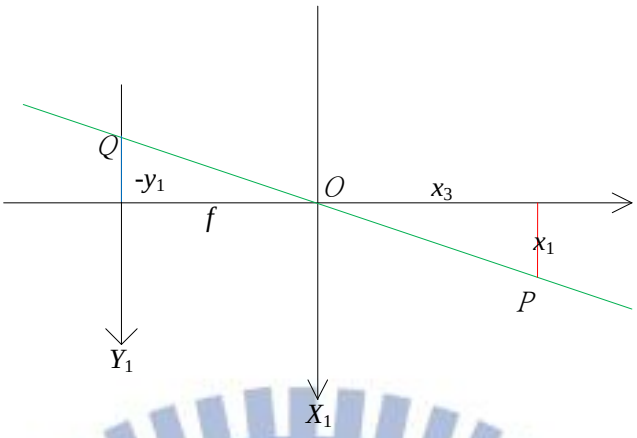


Fig. 4.4 The geometry of a pinhole camera as seen from the X2 axis

4.2.2 Ideas of 3D Image Construction and Coordinate Conversion

With the pinhole camera model reviewed in the last section, we can now describe how we construct a 3D image by mixing color and depth images according to the pinhole camera model.

In more detail, from Equation 4.3, we can get:

$$x_1 = -\frac{x_3}{f} \times y_1 ; \tag{4.4}$$

$$x_2 = -\frac{x_3}{f} \times y_2 ; \tag{4.5}$$

$$x_3 = \frac{x_3}{f} \times f \ . \tag{4.6}$$

And from Fig. 4.3, based on the similar-triangle principle again, we have the following equation:

$$\frac{x_3}{f} = \frac{\sqrt{{x_1}^2 + {x_2}^2 + {x_3}^2}}{\sqrt{(-y_1)^2 + (-y_2)^2 + f^2}} \tag{4.7}$$

where $\sqrt{(-y_1)^2+(-y_2)^2+f^2}$ is the length of line segment *OQ*, and $\sqrt{x_1^2+x_2^2+x_3^2}$ is the length of line segment *OP*. In the same way, we can get:

$$\frac{x_1}{y_1}=\frac{\sqrt{x_1^2+x_2^2+x_3^2}}{\sqrt{(-y_1)^2+(-y_2)^2+f^2}}; \tag{4.8}$$

$$\frac{x_2}{y_2}=\frac{\sqrt{x_1^2+x_2^2+x_3^2}}{\sqrt{(-y_1)^2+(-y_2)^2+f^2}}. \tag{4.9}$$

For our applications here, from the above equations we can derive more detailed facts in the following which are useful for the purpose of 3D image construction:

1. $\sqrt{x_1^2+x_2^2+x_3^2}$ is the distance between KINECT device and a point on the object, and we denote it as *d*;
2. the image center is *R* whose location is (X_{mid},Y_{mid}) in our captured depth image by the KINECT device;
3. denote the focal length of the depth cameras by f_d;
4. we can get Equations (4.10), (4.11), and (4.12) below from Equations (4.7), (4.8), and (4.9) for each point (x_p, y_p) in the depth image:

$$x_1=\frac{d}{\sqrt{(x_p-X_{mid})^2+(y_p-Y_{mid})^2+f_d^2}}\times(x_p-X_{mid}); \tag{4.10}$$

$$x_2=\frac{d}{\sqrt{(x_p-X_{mid})^2+(y_p-Y_{mid})^2+f_d^2}}\times(y_p-Y_{mid}); \tag{4.11}$$

$$x_3=\frac{d}{\sqrt{(x_p-X_{mid})^2+(y_p-Y_{mid})^2+f_d^2}}\times f_d. \tag{4.12}$$

By (4.10), (4.11), and (4.12) , we can convert a 2D point with coordinates (x_p, y_p) in a depth image into a 3D point with coordinates (x_1, x_2, x_3) and construct the 3D image by associating the corresponding color.

For the purpose of mapping 3D points to 2D pixels in the *color* image, we can derive, in a similar but inverse way, the following equations according to Equations (4.7), (4.8), and (4.9) (note that the focal length of the color camera is $f_C = 525$):

$$(x_c - X_{mic}) = -x_1 \times \frac{f_c}{x_3} \quad => \quad x_c = -x_1 \times \frac{f_c}{x_3} + X_{mic}; \tag{4.13}$$

$$(y_c - Y_{mic}) = -x_2 \times \frac{f_c}{x_3} \quad => \quad y_c = -x_2 \times \frac{f_c}{x_3} + Y_{mic}, \tag{4.14}$$

where (X_{mic}, Y_{mic}) are the image center of the color image.

Finally, we can construct a colorful 3D image from a depth image I_d and a color image I_c acquired by the KINECT device, and the details are described in the next section.

4.2.3 Construction Algorithm and Experimental Results

We have described each component of the proposed 3D image construction algorithm, and the full vision of this algorithm is presented below now.

Algorithm 4.1: construction of a 3D image.

Input: a color image I_c and a depth image I_d acquired with a KINECT device.

Output: a 3D image I_{3D} constructed by the use of I_c and I_d.

Steps:

Step 1. Transform the coordinates (x_p, y_p) and the depth value d of each pixel P_d in the depth image I_d into the 3D coordinates (x_1, x_2, x_3) of P_d in the 3D space by Equations (4.10), (4.11), and (4.12).

Step 2. Transform the 3D coordinates (x_1, x_2, x_3) into 2D coordinates (x_c, y_c) in the color image coordinate system by Equation (4.13) and (4.14).

Step 3. Use (x_c, y_c) as indices to find the color values (R, G, B) of the pixel P_c at coordinates (x_c, y_c) in the color image I_c.

Step 4. Take (R, G, B) as the color values of pixel P_d with coordinates (x_1, x_2, x_3) in the 3D space, and use these data (color values and 3D coordinates) to render a 3D color image I_{3D} using the OpenGL as output.

The tool of OpenGL mentioned in Step 4 above can draw 3D points in the 3D space. It so can be used to draw the 3D image from different views so that we can see a constructed model or screen in the 3D image from a specific view by the projecting the points of the model or scene onto the chosen-view plane.

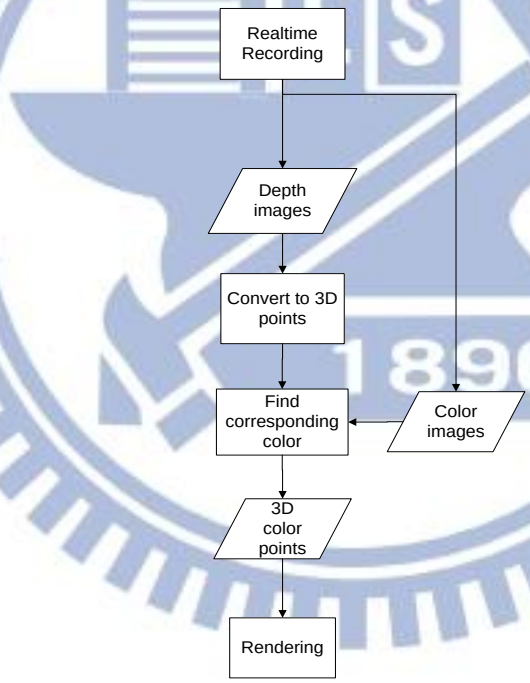
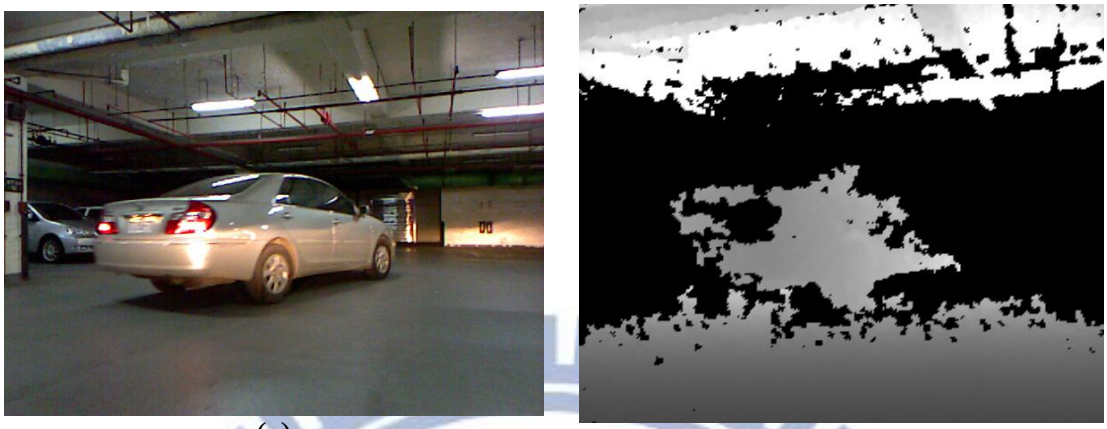


Fig. 4.5 A flowchart of 3D image construction algorithm.

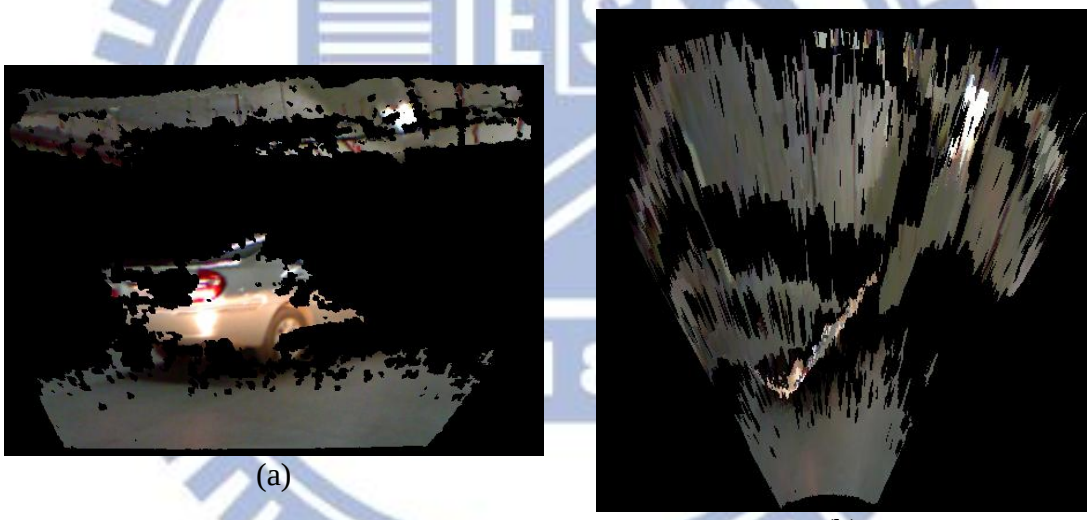
A result of applying Algorithm 4.1 is shown in Fig. 4.7, where the raw data (the original color image and depth image) are shown in Fig. 4.6. By Algorithm 4.1, the data can be converted into 3D format, and can be drawn with its corresponding color. Note that the black region represents no value of depth being available, so there is no

mapping to corresponding colors there.



(a) (b)

Fig. 4.6 Images acquired by a KINECT device. (a) The depth image. (b) The color image.



(a) (b)

Fig. 4.7 A constructed 3D image. (a) A perspective view of the 3D image. (b) A top view.

4.3 Review of a Method for Geometric Correction of 3D Images

4.3.1 Idea of Geometric Correction

Once after we took a picture of a flat wall, which included both a depth image and a color one, and then conducted Algorithm 4.1, we found that the flat wall was a curved surface instead a plane when the resulting 3D image was displayed for inspection. The reason is that the infrared light rays emitted by the KINECT device are not all parallel to the X_3-axis shown in Fig. 4.3, so that the we won't get accurate data.

A method has been proposed to solve this problem and is reviewed here. This curved surface was supposed to be of the shape of a paraboloid. Then, a paraboloid equation was derived for correcting this error. Specifically, after the paraboloid equation was found, the coordinates of x and y in the 3D image were substituted into this equation to get a corrected z value, as illustrated in Fig.4.8.

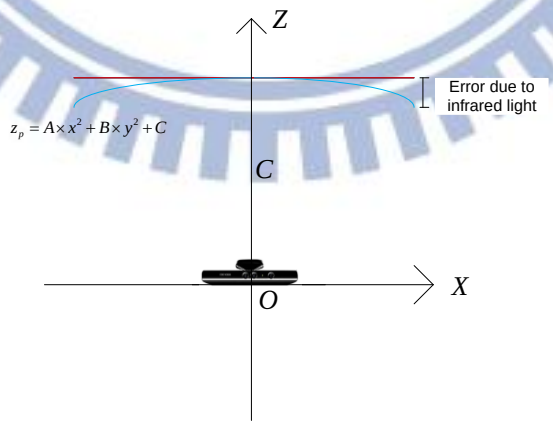


Fig. 4.8 The paraboloid seen from the direction of the Y-axis (i.e., from the top view).

4.3.2 Correction Algorithm and Experimental Results

Based on the above idea, an approximating paraboloid equation may be derived according to the criterion of minimum sum of squared errors (MSSE) in the following way.

(1) Let the equation of the paraboloid be written as:

$$z_p = Ax^2 + By^2 + C. \tag{4.15}$$

where A and B are the quadratic coefficients and C is an intercepted length from the KINECT device to the apex of the paraboloid, as shown in Fig. 4.8.

(2) The equation for computing the value *SSE* of the SSE is:

$$SSE = \sum_{i=0}^{i=h\times w} \left[z_i - \left(Ax_i^2 + By_i^2 + C \right) \right]^2. \tag{4.16}$$

where (x_i, y_i, z_i) are the values of a 3D image pixel computed by Algorithm 4.1 and h and w are the height and width of this depth image, respectively.

(3) To find the coefficients A, B and C, according to the minimum SSE criterion, the partial derivatives of Equation (4.16) with respect to variables A, B and C, respectively are derived, yielding the following equations:

$$\sum_{i=0}^{i=h\times w} \left[z_i - \left(Ax_i^2 + By_i^2 + C \right) \right] \times \left(x_i^2 \right) = 0; \tag{4.17}$$

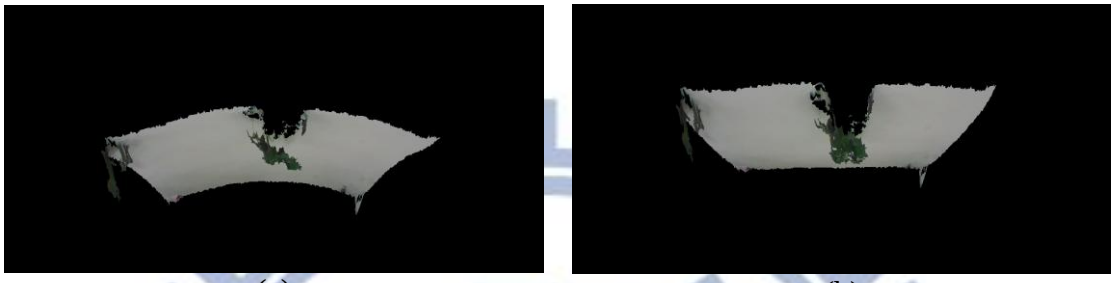
$$\sum_{i=0}^{i=h\times w} \left[z_i - \left(Ax_i^2 + By_i^2 + C \right) \right] \times \left(y_i^2 \right) = 0; \tag{4.18}$$

$$\sum_{i=0}^{i=h\times w} \left[z_i - \left(Ax_i^2 + By_i^2 + C \right) \right] = 0. \tag{4.19}$$

(4) The above simultaneous equations may be solved to obtain analytic solutions for the values of the coefficients A, B, and C.

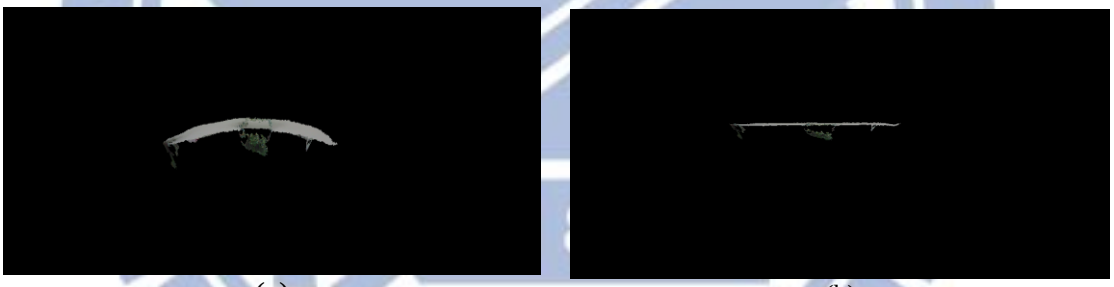
(5) The paraboloid is obtained by substituting *A*, *B*, and *C* back to (4.16).

Finally, we show two examples of the experimental results of applying the above scheme of geometric correction to 3D images in Figs. 4.9 and 4.10.



(a) (b)

Fig. 4.9 The 3D image of a wall seen from above. (a) Before correction. (b) After correction.



(a) (b)

Fig. 4.10 The 3D image of another wall seen from the top view. (a) Before correction. (b) After correction.

Chapter 5
Modeling of Around-car Objects

5.1 Introduction

In this chapter, we describe how we combine data acquired with multiple KINECT devices, and show the results in the same screen as a single view, like as a front view, a side view, or a rear view. Before this, we have to do calibration for the purpose of associating the parameters of the real world and those of the virtual world which is constructed in this study.

The calibration work in 3D space costs processing time, so we use a k-d tree to speed up our calibration steps. After this process, we know the relationship between each pair of neighboring KINECT devices. Then, we merge all the data to see a single view not only as a sparse point cloud, but also as a complete model which is constructed by many polygons. Since the 3D rendering bottleneck for model construction is the number of polygons, we try to reduce the number of polygons in our data without breaking its geometric property. The method of reduction is based on the use of the quadric error matrix (QEM). Before using the QEM, we have to construct polygons in the acquired depth images first. As a consequence, we can see a 3D image which is constructed by multiple KINECT data via real-time rendering.

Moreover, the object in the screen which we want to tag such as cars or people is the kinds of important information in a video surveillance. After that, we can add an index into each frame which includes the object information, so that the search of interesting objects or scenes can be made automatic without human involvement.

5.2 KINECT Camera Calibration

5.2.1 Review of Calibration of a Single KINECT

In this section, we will introduce how we get the focal length of the RGB-camera in the KINECT device, which we mentioned in the previous chapter and will be used in the calibration process. At first, we measure an object in the screen to know the object's height p_y in the unit of pixel. Next, by the pinhole camera model again as shown in Fig. 5.1, we can measure the distance d_z from the object to the KINECT device, as well as the object's height h in the real world in the unit of mm. Then, we can apply the similar-triangle principle again to derive the following equation:

$$\frac{p_y}{f} = \frac{h}{d_z}.$$

Finally, the focal length f can be obtained by solving the above equation to be

$$f = p_y \times d_z / h. \tag{5.1}$$

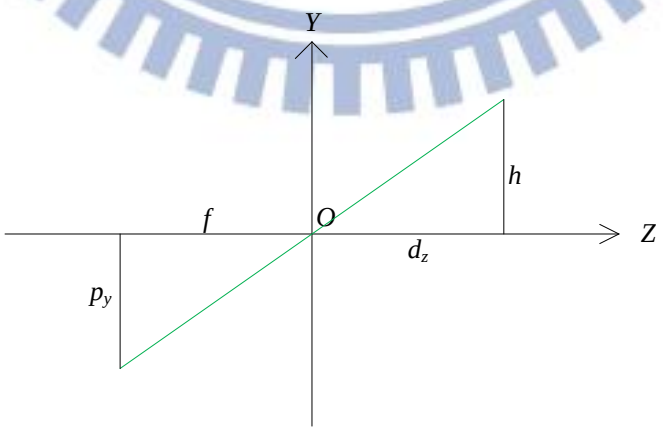


Fig. 5.1 The pinhole camera model for calibration of the focal length of the camera.

5.2.2 Transformation of Coordinates

In this section, we will introduce the method of coordinate transformation with6 degrees of freedom, which can be derived by two parts — one is translation, and the other is rotation.

First, we define a vector used in the transformation to be:

$$T = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{5.2}$$

where a is the distance of translation in the X direction, b and c are interpreted similarly. Next, we define matrices for three ways of rotations, namely, pan, tilt, and swing, respectively, below:

$$R_P = \begin{bmatrix} \cos\theta_P & 0 & \sin\theta_P \\ 0 & 1 & 0 \\ -\sin\theta_P & 0 & \cos\theta_P \end{bmatrix}; \tag{5.3}$$

$$R_T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_T & -\sin\theta_T \\ 0 & \sin\theta_T & \cos\theta_T \end{bmatrix}; \tag{5.4}$$

$$R_S = \begin{bmatrix} \cos\theta_S & -\sin\theta_S & 0 \\ \sin\theta_S & \cos\theta_S & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.5}$$

where θ_P is the pan angle, θ_T is the tilt angle, and θ_S is the swing angle. Then, a total rotation R can be derived to be $R = R_S \times R_T \times R_P$, leading to:

$$R=\begin{bmatrix}\cos\theta_S & -\sin\theta_S & 0\\ \sin\theta_S & \cos\theta_S & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta_T & -\sin\theta_T\\ 0 & \sin\theta_T & \cos\theta_T\end{bmatrix}\begin{bmatrix}\cos\theta_P & 0 & \sin\theta_P\\ 0 & 1 & 0\\ -\sin\theta_P & 0 & \cos\theta_P\end{bmatrix}, \qquad (5.6)$$

or equivalently, to

$$R=\begin{bmatrix}\cos\theta_S\cos\theta_P-\sin\theta_S\sin\theta_T\sin\theta_P & -\sin\theta_S\cos\theta_T & \cos\theta_S\sin\theta_P+\sin\theta_S\sin\theta_T\cos\theta_P\\ \sin\theta_S\cos\theta_P+\cos\theta_S\sin\theta_T\sin\theta_P & \cos\theta_S\cos\theta_T & \sin\theta_S\sin\theta_P-\cos\theta_S\sin\theta_T\cos\theta_P\\ -\cos\theta_T\sin\theta_P & \sin\theta_P & \cos\theta_T\cos\theta_P\end{bmatrix}.$$

As a result, if a point P_1 become P_1' after the transformation, we can express P_1' as

$$P_1' = R\times P_1 + T \qquad (5.7)$$

for some θ_P, θ_T, θ_S, a, b, and c. This concludes our review of the coordinate transformation process, and this process will be used in the next section for our application of 3D scene modeling.

5.2.3 Review of Iterative Closest Point (ICP) Algorithm

The ICP algorithm is a method useful for 3D model alignment [6-8]. In this section, we will introduce the operation of the ICP step by step, and describe the bottleneck of the algorithm.

First, the original ICP algorithm is basically a brute-force method for aligning two clusters of points which are used to construct a scene or object model. The core idea of this algorithm is that “each point in one group finds a closest point in the other group.” More points this rule is satisfied by, more possibly the corresponding transformation (including the translation and rotation) is true. The detail of this

algorithm is shown below.

Algorithm 5.1: aligning two groups of points in a 3D space.

Input: Two groups of points, P_a and P_b.

Output: A translation matrix T and a rotation matrix R for aligning P_a and P_b.

Steps:

Step 5. Make a little change of T and R on P_b, and compute $P_b' = P_b \times R + T$. where R and T are as described in Section 5.2.2 (leading to Equation 5.7).

Step 6. For each point b in P_b', find the closest point s to it in P_a, and calculate the Euclidean distance d_{bs} between b and s.

Step 7. After all the points in P_b' have been processed, sum up all the distances to get

$$D_{min} = \sum_{b \in P_b} [d_{bs}]$$

Step 8. If D_{min} is the minimal in all the possible changes of transformation and rotation, then the corresponding T and R are the transformation and rotation matrices for aligning the two groups of points, P_a and P_b.

From the algorithm described above, we know that the speed bottleneck is Step 2 since for each point in one group, we have to check all the points in the other group by brute force. An improved method is proposed in the next section.

5.2.4 Calibration by the ICP algorithm Using Speeded-up k-d Tree

In this section we introduce how k-d tree can be utilized to speed up Algorithm 5.1. First, it is noted that the k-d tree is a data structure for "partitioning the 3D space." After such partitioning, we can speed up Algorithm 5.1. In Step 2 of

Algorithm 5.1, it is necessary that for every point b in P_b', the closest point s to it in P_a should be found exhaustively. And this requires that all the Euclidean distances d_{bs} between b and s for all s in P_b' be computed. Instead of that way, we can maintain a better data structure using the k-d tree to know the neighboring points so that we can speed up by avoiding an exhausted search. The method for construction of the k-d tree is described as an algorithm below:

Algorithm 5.2: Construction of a k-d tree.

Input: A model with M points, and a pre-defined value S which imposes a limit on the number of elements in each node of the k-d tree.

Output: A k-d tree T_{kd} which stores M.

Steps:

Step 1. Create a node for storing M, and label its level as 0.

Step 2. If the current level mod 3 is 0, then load the X-axis value of M;

If the current level mod 3 is 1, then load the Y-axis value of M;

If the current level mod 3 is 2, then load the Z-axis value of M.

Denote each of these values of M as M_c.

Step 3. Calculate the median M_m of all M_c obtained so far.

Step 4. Create two nodes as the left child N_L and the right child N_R of the current node N, and assign their levels as L+1 where L is the level of N.

Step 5. Assign M to N_L if M is smaller than M_m; or assign M to N_R if M is larger than or equal to M_m

Step 6. If the numbers of points of the left child is larger than S, then back to Step 2.

Step 7. If the numbers of points of the right child is larger than S, then back to Step 2.

Step 8. If the numbers of points of both the right and the left nodes are smaller than S, then construction of the desired tree T_{kd} is completed.

Note that in the above algorithm, if S = the number of M, the space isn't partitioned. By the above algorithm, the search will become a binary one in the k-d tree T_{kd}, which will then speed up the search. The original algorithm conducts an exhaustive search with a time complexity of $O(N^2)$, where N is the number of points in M, while the new algorithm by the k-d tree search instead has a time complexity of $O(N+N(\log N)+S)$ where S is a constant and $N\log N$ is greater than N, so that the final time complexity is just $O(N \log N)$.

5.2.5 Calibration of Relation between Neighboring KINECT Devices

The ICP algorithm is described in the previous section. We can "calibrate" the relation between neighboring KINECT devices by the use of the ICP algorithm as well as some of our knowledge which have been mentioned in Chapter 2. The details of the algorithm to implement the above idea are shown below. Note that we use a box as the calibrate target in the algorithm.

Algorithm 5.4: Calibration of the geometric relation between two neighboring KINECT devices

Input: two depth image I_{d1} and I_{d2} which are acquired with two neighboring KINECT devices.

Output: the relationship (T, R) between these KINECT devices where T denotes a translation and R denotes a rotation.

Steps:

Step 1. Put a chair on the ground to elevate a box which is used the calibration target.

Step 2. Take depth images from the two neighboring KINECT devices, and conduct

Algorithm 4.1 without the color associating steps, i.e., transform only the 2D depth images into non-colored 3D images.

Step 3. Segment out the calibration target in the acquired images by some pre-learned knowledge like the position of the calibration target and the height of the KINECT devices.

Step 4. Using two 3D models resulting from the last step, Step 3, to conduct Algorithm 5.1 to get the precise relationship (T, R) where T denotes a translation and R denotes a rotation, both from a model to the other.

Step 5. Repeat Step 1 through Step 4 to find the relationship for each pair of neighboring KINECT devices until done.

After the above process of calibration is completed, a series of rotation and translation parameters are obtained, which may be used to merge the data from all the KINECT devices. More details will be described in Section 5.3 next, and the result is shown in Section 5.4.

5.3 Merge 3D Images from Multiple KINECT Devices

5.3.1 Coordinate Mapping between Local and global

The coordinate mapping between the local and the global coordinate systems is a serial process which can be written as a matrix as described in Section 5.2.2. In our system, to merge 3D images, the global coordinate system (indicated by the green lines in Fig. 5.2) is taken to be on the front KINECT device which faces straightly ahead of the car. The other coordinate system built on another KINECT device (indicated by the red lines in Fig. 5.2) are mapped into the global coordinate system

with their relationship computed by Algorithm 5.4. By this mapping, the 3D images corresponding to the two KINECT devices can be merged into a single one.

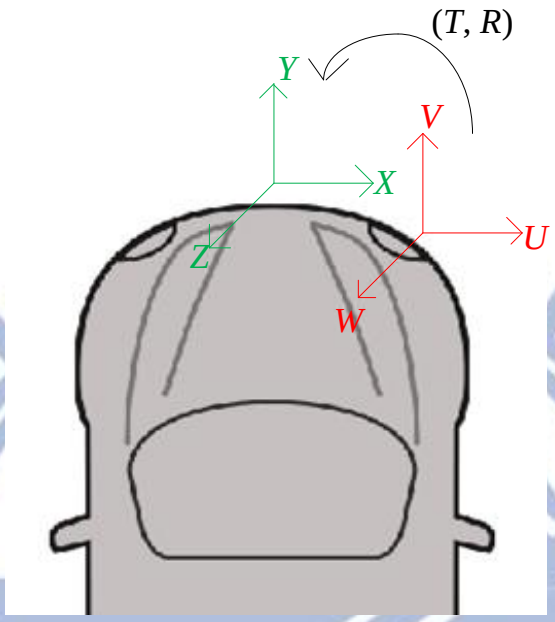


Fig. 5.2 Transformation between two coordinate systems built on KINECT devices affixed on the car.

Furthermore, to merge the 3D images of a cluster of KINECT devices (more than two KINECT devices), the matrices (T, R), which are acquired by executing Algorithm 5.4 and appear in Equation (5.7), should be applied in series. A tip for this series of transformations is “backward” since multiplications of matrices have the property that the last multiplied matrix affects the result of all the previously multiplied ones.

For instance, when we want to map the 3D image of Device 3 into the global coordinate system which is built on Device 1 as shown in Fig. 5.3, at first, we map each point P_3 in the 3D image of Device 3 to a point P_2 in the 3D image of Device 2 by the following equation according to Equation (5.7):

$$P_2 = R_2P_3 + T_2. \quad (5.8)$$

In the same way, we map a point P_2 in the 3D image of Device 2 to a point P_1 in the 3D image of Device 1 by the following equation:

$$P_1 = R_1P_2 + T_1. \quad (5.9)$$

Merging the two equations of (5.8) and (5.9), we get:

$$P_1 = R_1R_2P_3 + R_1T_2 + T_1. \quad (5.10)$$

That is, every point P_3 in the 3D image of Device 3 can be mapped *directly* into a point P_1 in the 3D image of Device 1 by Equation (5.10).

Equation (5.10) may be extended easily to cases involving more than three KINECT devices. The derivations are similar and omitted.

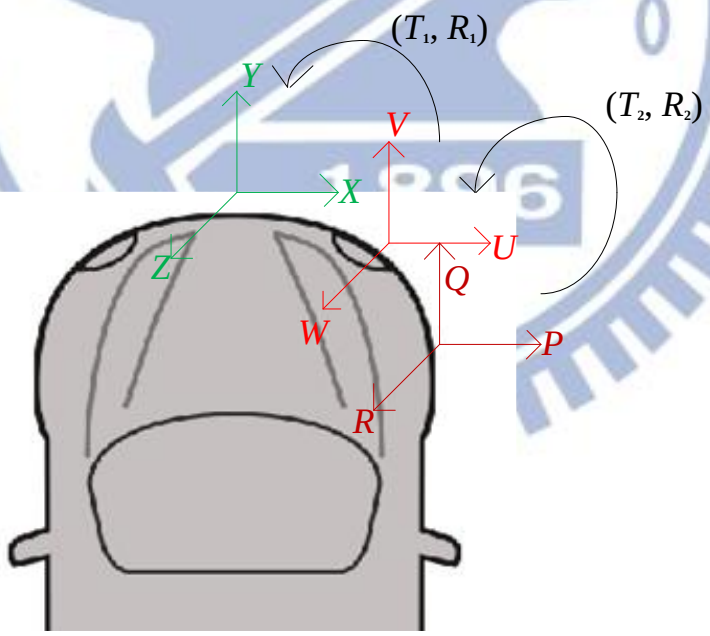


Fig. 5.3 The transformations involved in merging three 3D images taken by three neighboring KINECT devices.

5.3.2 Reduction of Merged Data Using Mesh Structure

The data structure "mesh" may be used to reduce large amounts of image data. This data structure illustrates how points are connected, and this is important information for reducing data. In this section, we will introduce the construction of mesh on the depth image.

The mesh on a depth image may be drawn in a way like that shown in Fig. 5.4. Each time we construct two triangles with three points per triangle. One is the triangle with corners (i, j), $(i, j+1)$, and $(i+1, j)$; and the other is the triangle with corners $(i+1, j)$, $(i, j+1)$, and $(i+1, j+1)$, where $1 \le i \le I$ and $1 \le j \le J$ with I and J, as shown in Fig. 5.4, being the width and height of the depth image, respectively.

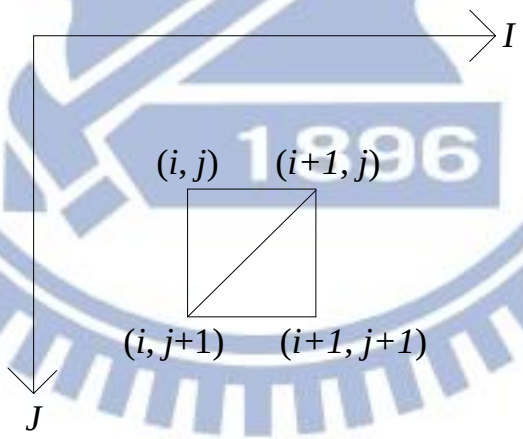


Fig. 5.4 Constructing a mesh on the depth image for each pixel (i, j).

When the depth image is with size $w \times h$, the desired mesh can be constructed to be composed of $(w-1) \times (h-1) \times 2$ triangles (or polygons), as can be Fig.d out.

5.3.3 Review of Quadric Error Matrix (QEM)

In this section, we review a surface simplification method using quadric error metrics [9]. By this method, we can remove some polygons in a 3D model, and keep the shape in the model as much as possible, as we have done in this study.

In this method, the error of removing each edge is calculated, and each time only the edge which has the minimal error is removed. In other words, removing the *minimal-error edge* will not change greatly the shape property of the model. The detail of an algorithm implementing the method is as follows.

Algorithm 5.5: simplification of 3D model shape using quadric error metrics.

Input: A model with points P.

Output: A simplify model.

Steps:

Step 1. Construct a mesh as a multiple of triangles in the following way.

1.1 For each triangle p, find its plane equation as $ax + by + cz + d = 0$ where $a^2 + b^2 + c^2 = 1$.

1.2 Construct a 4×4 matrix K for this triangle p as follows:

$$K_p = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}.$$

Step 2. Treat each vertex v of the triangle as a solution of multiple planes by calculating a 4×4 matrix Q_v as follows:

$$Q_v = \sum_{v \in P} K_p .$$

Step 3. Start edge contraction and use the notation w to express the union of v_1 and v_2, so that the new Q_w for the new point w is simply $Q_{v1}+Q_{v2}$ with w being:

$$w = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

where q_{ij} is the factor of Q_w.

Step 4. For each edge (v_1, v_2) on the model, calculate the error:

$$\text{Error } (v_1, v_2) = w^{-1}(Q_{v1} + Q_{v2})w$$

where w is obtained from Step 3.

Step 5. Select the edge with the minimal error and remove it, and then do edge contraction which is described in Step 3 and update all the errors of edges which involve points.

Step 6. Repeat the above steps until simplification is enough.

This algorithm creates a simplification effect on the mesh each time, as illustrated in Fig. 5.5. Each time, the contraction removes two polygons on the average.

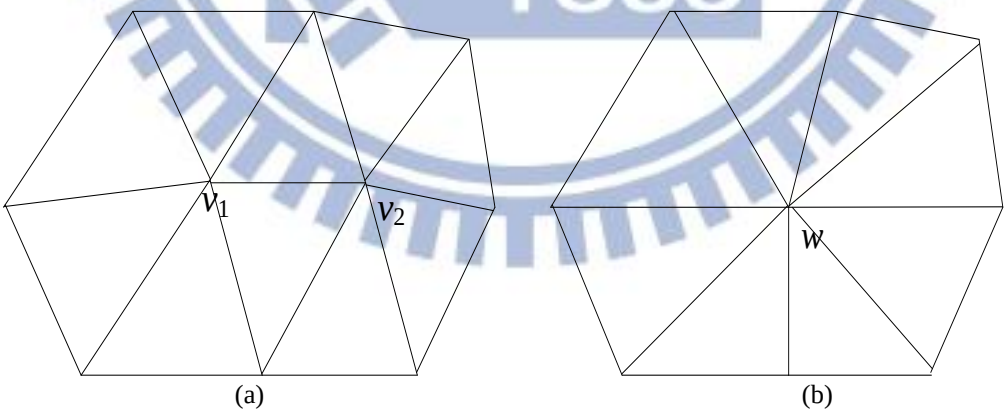


Fig. 5.5 The contraction of two vertices. (a) Before contraction. (b) After contraction.

By using this algorithm, we can remove some polygons and keep the shape as much as possible since we remove an edge with the minimal error each time.

5.3.4 Merge Algorithm

In this section, we summarize the above-described algorithms to present an algorithm for merging all the 3D images constructed from the color and depth images acquired by the multiple KINECT devices. In the algorithm, at first, we acquire depth and color images with the KINECT devices in realtime. Then, a series of processing works are done in an offline fashion. Finally, the processed data are used for rendering the shape of the object model in realtime. The detail of the algorithm is shown in below. A corresponding flow chart is shown in Chapter 2.

Algorithm 5.6: Merge of multiple 3D images.

Input: Multiple color image I_C and depth image I_d acquired with by the KINECT devices, respectively.

Output: An around-car model M for display on the screen.

Steps:

Step 1. For each KINECT device, convert their images I_d and I_C into a 3D image I_{3D} using Algorithm 4.1.

Step 2. Transform each 3D image I_{3D} into the global coordinate system using the method described in Section 5.3.1 where the relationship (R, T) is acquired by Algorithm 5.4.

Step 3. Construct a mesh on the depth image I_d as described in Section 5.3.2

Step 4. Reduce the polygons on the mesh with Algorithm 5.5 after choosing the edge with the global minimal error each time.

Step 5. Save the processed data into a file (including the coordinates of each point and the corresponding color).

Step 6. Rendering the file with the OpenGL.

As can be seen from Step 6 of the above algorithm, the data processing and rendering works are separated.

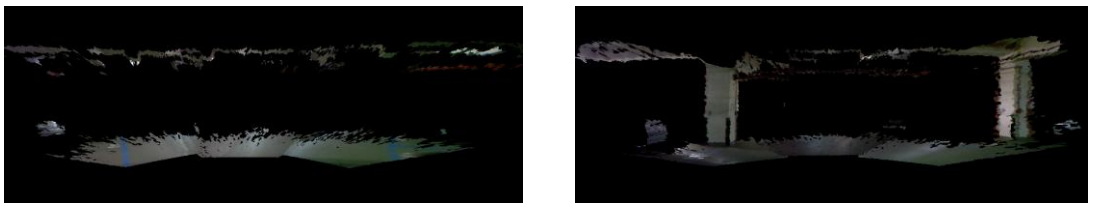
5.4 Experimental Results

First, a result of calibration is shown in Fig. 5.6. We get the relationship by aligning the two boxes which appear in two images acquired by two KINECT devices. We show the calibration target by using a method of "lighting" depth data, since the calibration uses the depth image only.

Next, a result of merging multiple 3D images to construct an around-car image is shown in Fig. 5.6.

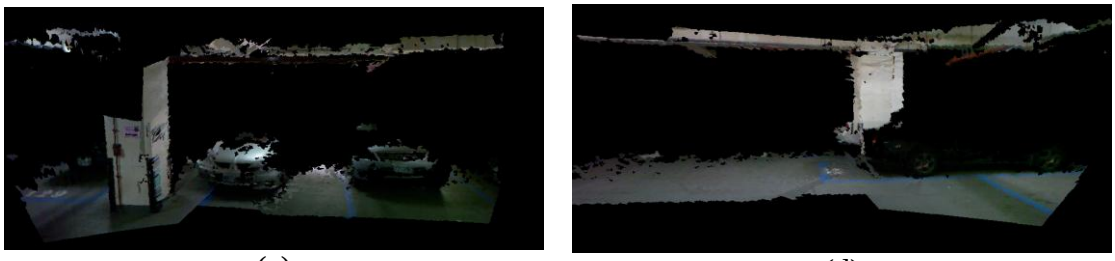
(a) (b)

Fig. 5.6 The calibration of two neighboring KINECT devices. (a) Before alignment (b) After alignment.



(a) (b)

Fig. 5.7 The constructed around-car model. (a) Front view. (b) Rear view. (c) Right side view. (d) Left Side view.



(c) (d)

Fig. 5.8 The constructed around-car model. (a) Front view. (b) Rear view. (c) Right side view. (d) Left Side view. (Cont'd).

5.5 Object Detection

5.5.1 Object Detection in Depth Images

Finding an object in a depth map is an important thing which has been mentioned at the beginning of this chapter, and it's also important for constructing our long range view since the nearby object will affect our algorithm which will be described in the next chapter.

Because we have not only depth images, but also color images, the work of object detection may be conducted differently from the traditional object detection work using color images only. Our method of object detection is proposed by using depth images.

Since the method of transforming the depth image to a 3D image was described in the previous section, our method may be regarded as a cross-layer method. Because we can't know more detailed information other than the depth image, the 3D image is also needed in the proposed method. The detail of an algorithm implementing the method will be described in the next section.

5.5.2 Component Labeling by Region Growing on Depth Images for Object Detection

In this section, we will describe our method proposed for object detection. First, we apply region growing to the input depth image, where the growing condition is the depth difference between two pixels. In addition, the object segmentation operation is controlled by a threshold. And the 3D information is used to infer the object size or the ground height in the real world.

By using this method, we can label each component (or object) in the depth image. The details implementing the proposed method are described as an algorithm in the following.

Algorithm 5.7: Object detection.

Input: A depth images I_d and the related 3D images I_{3D}.

Output: The label of each component C.

Steps:

Step 1. Filter out the ground part in I_d according to the pre-learned height of the KINECT devices.

Step 2. For each pixel in the remaining part of I_d, take out its 3D position in I_{3D} and grow accordingly this pixel in depth image I_d with a threshold τ used in determining how far a pixel should be to be included in an already-grown object in the growing process.

Step 3. Label each resulting component C_i and calculate its size by using the 3D information in I_{3D}.

Step 4. Find the corresponding part of each component C_i in the color image for display uses.

Note that the use of a very small threshold τ will lead to growing broken objects, while the use of a very large threshold will lead to yielding just one object, i.e., all objects being merged into a single one. On the average, the value of τ is considered to be 300mm for segmenting normal-sized objects. An example of nearby object detection by Algorithm 5.7 is shown in Fig. 5.8.

(a) (b)

Fig. 5.9 The object detection result. (a) Original color image. (b) The detected object part in the color image.

Chapter 6
Long-Range View Construction and Display

6.1 Ideas of Proposed Techniques

In this chapter, we will describe the proposed method for long-range view construction and display. We have introduced the method of constructing 3D images around a car in the previous chapters. In addition, we found that the far views of scenes which we are also interested in have no depth information, but we still need them to enrich our 3D images and to facilitate inspection of nearby objects against far backgrounds.

Therefore, we propose a method for stitching respective 2D color images acquired by the KINECT device to compose a *long-range view* of the background scene by putting them in an identical frame. The resulting long-rang view will appear to be a single *panoramic image* which is also a color image. We want to make this 2D panoramic image to "sit" at the back of nearby objects which have been modeled by techniques described in the previous chapter. The model is a 3D image, but our panoramic image is 2D, so the relation must to be constructed properly so that we can render this 2D image in the 3D space to yield a pleasing integrated view of the entire scene.

In the following sections, we will introduce first a method of automatically stitching multiple images in Sections 6.2, and then a method for composing a panoramic image by stitching in Section 6.3. Finally, we describe a method we

propose to merge 3D images into panoramic images in Section 6.4.

6.2 Automatic Panoramic Image Stitching from Multiple KINECT Images

In this chapter, we will describe the method used for automatic image stitching [10-14] with two images as input.

Firstly, relevant features in the acquired images using the KINECT device are found out, when then are matched to see the overlap parts between the two input images before conducting image stitching. For this, the scale-invariant feature transform (SIFT) is used. Next, whether the features extracted are enough in number to yield a correct and stable stitching result is checked. Also, the images are checked to see if they are good enough for stitching after projecting onto a cylinder. Finally, the two images are registered into one, and the area of overlapping is blended to reduce the seam effect. An algorithm implementing the above ideas to stitch two color images is presented in the following.

Algorithm 6.1: stitching two color images into one.

Input: Two color images I_1 and I_2.

Output: An image I_S resulting from stitching I_1 and I_2.

Steps:

Step 1 Detect sets of SIFT features, F_1 and F_2 in I_1 and I_2, respectively; and match I_1 and I_2 using F_1 and F_2.

Step 2 Warp images I_1 and I_2, i.e., project I_1 and I_2 onto a cylinder.

Step 3 Use the RANSAC measure to check if the number of similar features in I_1 and I_2 are enough according to the following inequality:

$$n_i > \alpha + \beta n_f \tag{6.1}$$

where n_f is the total number of features in the overlapping area of I_1 and I_2; n_i is the number of inliers; and α = 8.0 and β = 0.3 which are set for the purpose of yielding a correct image match according to experimental experiences.

Step 4 If the answer of the last step is yes, then align the two images I_1 and I_2 and blend the result to get the desired stitching result I_S; otherwise, quit the algorithm.

By using the above algorithm, we can stitch two images into one as shown by the examples of results in Figs. 6.1 and 6.2.

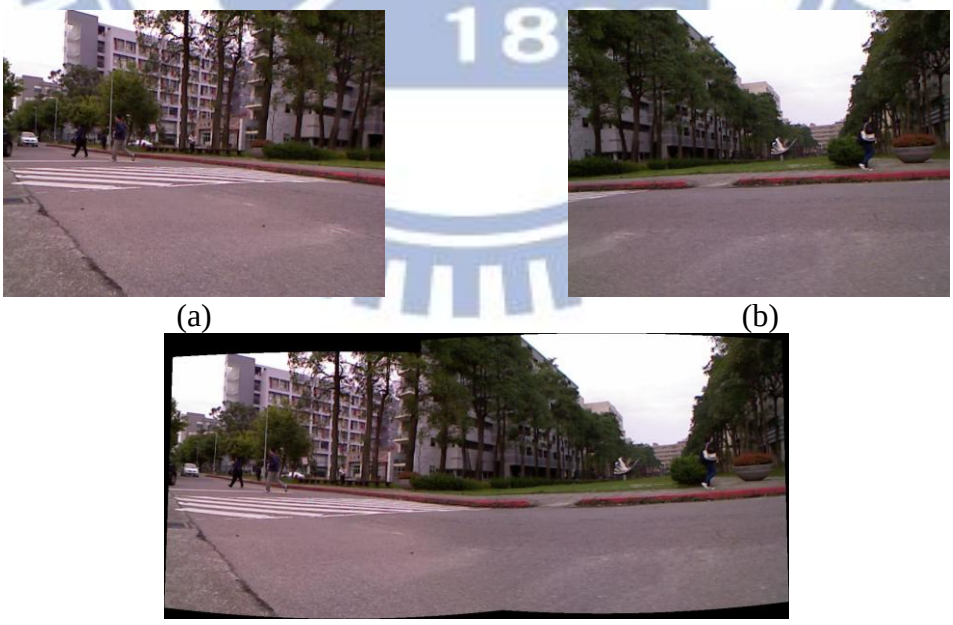
(a) (b)

Fig. 6.1 The two images before stitching, where (a) is a slightly left rotated version of (b).

Fig. 6.2 Stitching result from the two images in Fig. 6.1 using Algorithm 6.1.

6.3 Panoramic Image Stitching

We use Algorithm 6.1 to construct a panoramic image from multiple images. Sometimes, we found that the features in the overlapping area of two images are not enough; in other words, Inequality 6.1 is not satisfied. In such cases, we try the use of different values of α and β to satisfy the inequality, as shown in Figs. 6.3 and 6.4.



(a) (b)

(c)

Fig. 6.3 Stitching two images with different values of α and β — case 1. (a) and (b) are original images, and (c) is the stitching result with $\alpha = 10$, $\beta = 0.2$.

(a)

(b)

(c)

Fig. 6.4 Stitching two images with different values of α and β — case 2. (a) and (b) are original images, and (c) is the stitching result with $\alpha = 0$, $\beta = 0.13$.

Furthermore, we have found another phenomenon. If we impose a high restriction using the RANSAC measure on the involved images, the result is good and the seam is hard to find. If we impose a low restriction using the RANSAC measure on the involved images, the result is worse instead. Therefore, we propose an algorithm for stitching according to this experience. First, we try to stitch two images with a high restriction. If this leads to a failure of satisfying the above-mentioned inequality, we try a lower restriction and so, until the stitching succeeds. The detail of the algorithm for this type of dynamic image stitching is shown below.

Algorithm 6.2: dynamic mosaic image stitching.

Input: Two color images I_1 and I_2.

Output: An image I_S resulting from stitching I_1 and I_2.

Steps:

Step 1 Stitch I_1 and I_2 using Algorithm 6.1 with pre-selected initial values of α and β.

Step 2 If the algorithm fails in satisfying the inequality of (6.1), try a lower value of α first, and then a lower value of β until the inequality is satisfied.

Step 3 Take the final result yielded by Algorithm 6.1 as the desired output.

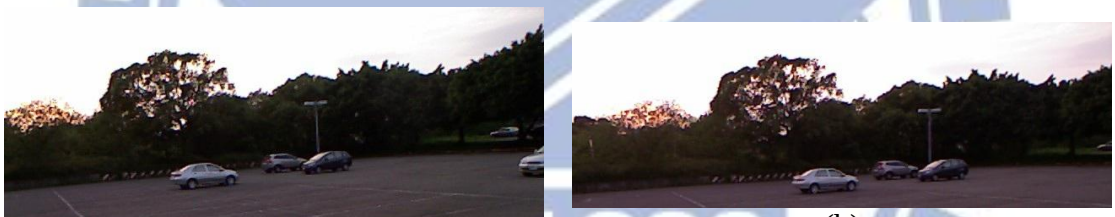
By using this "dynamic" algorithm, we can stitch images successfully while keeping the quality of the resulting image as good as possible. According to our experimental experiences, "good" images yielded by neighboring KINECT devices will pass the high restriction, and vice versa, as shown in Fig. 6.5.

Fig. 6.5 Stitching result compares with those of Figs. 6.3 and 6.4 with α = 10 and β = 0.2.

In some cases, the features in the overlapping area of the two images are too small in number, compared with those of the other parts in the images, leading to failures in stitching the two images. This case often happens when there is a big nearby region with few features in the overlap portion of the two images like the one shown in Fig. 6.6. To solve such a problem, we cut the nearby parts of the two images as shown in Fig. 6.7, and then the stitching becomes successful, as shown in Fig. 6.8.

(a) (b)

Fig. 6.6 Two color images acquired from two neighboring KINECT devices where view (a) is a left shift version of view (b), and the essential overlap part (the three cars in the middle) is too small, compared with the parking space in front which has fewer features for matching.



(a) (b)

Fig. 6.7 Two color images with nearby parts cut, where view (a) is a left shift version of view (b).

Fig. 6.8 Successful stitching result using images in Fig. 6.7 as input.

After the algorithm of mosaic image stitching is described, we can use it to stitch images taken from multiple directions around the car into a panoramic image for inspection of each surrounding view.

Specifically, we stitch each pair of images which are acquired from two neighboring KINECT devices. With such results of all possible pairs of KINECT devices as inputs, we perform Algorithm 6.2 to conduct the stitching work repeatedly until all the images are stitched into a single panoramic image, as illustrated in Fig.6.9.

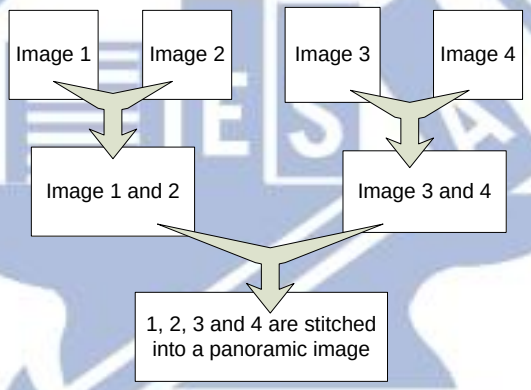


Fig. 6.9 Illustration of the stitching method. The neighboring number of images are corresponded to the neighboring position of each KINECT devices.

An example of the results obtained from applying the above process is shown in Figs. 6.10 and 6.11.

Fig. 6.10 The original panoramic image yielded by the panoramic image stitching process illustrated in Fig. 6.8.

Fig. 6.11 A well-cut version of Fig. 6.9.

6.4 Merging 3D image with background image

In this section, we introduce the proposed method for mapping the stitched image into 3D space and rendering them together with nearby object models which have been constructed as discussed in the previous chapter.

At first, we establish a bipolar coordinate system as illustrated in Fig. 6.12. Each point p on a sphere can be described by the polar coordinates (r, α, β). Then, we map the panoramic image resulting from image stitching onto the a sphere. In the coordinates (r, α, β) of each point transformed from an image point, the value r is a fake depth, and the orientation values of α and β are determined by the vertical and the horizontal viewing angles of the KINECT device, respectively.

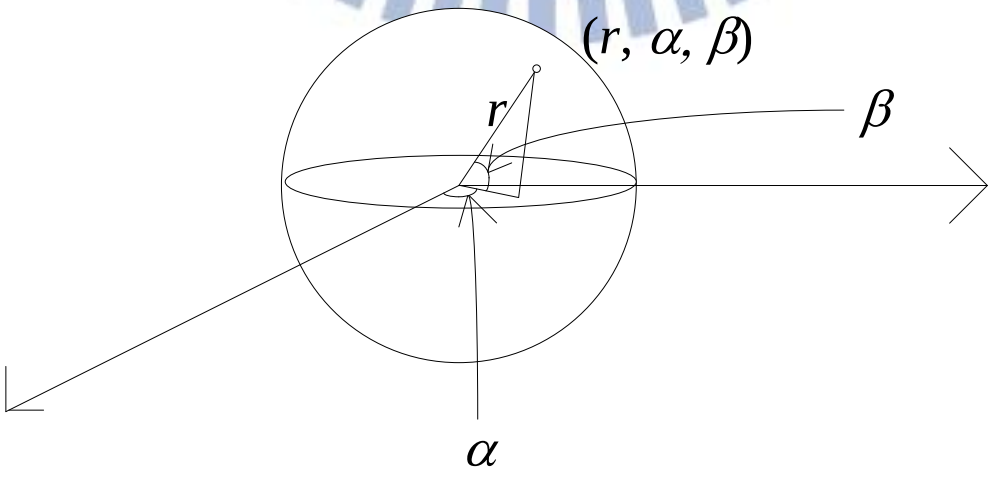


Fig. 6.12 A 3D point in a sphere expressed by polar coordinates.

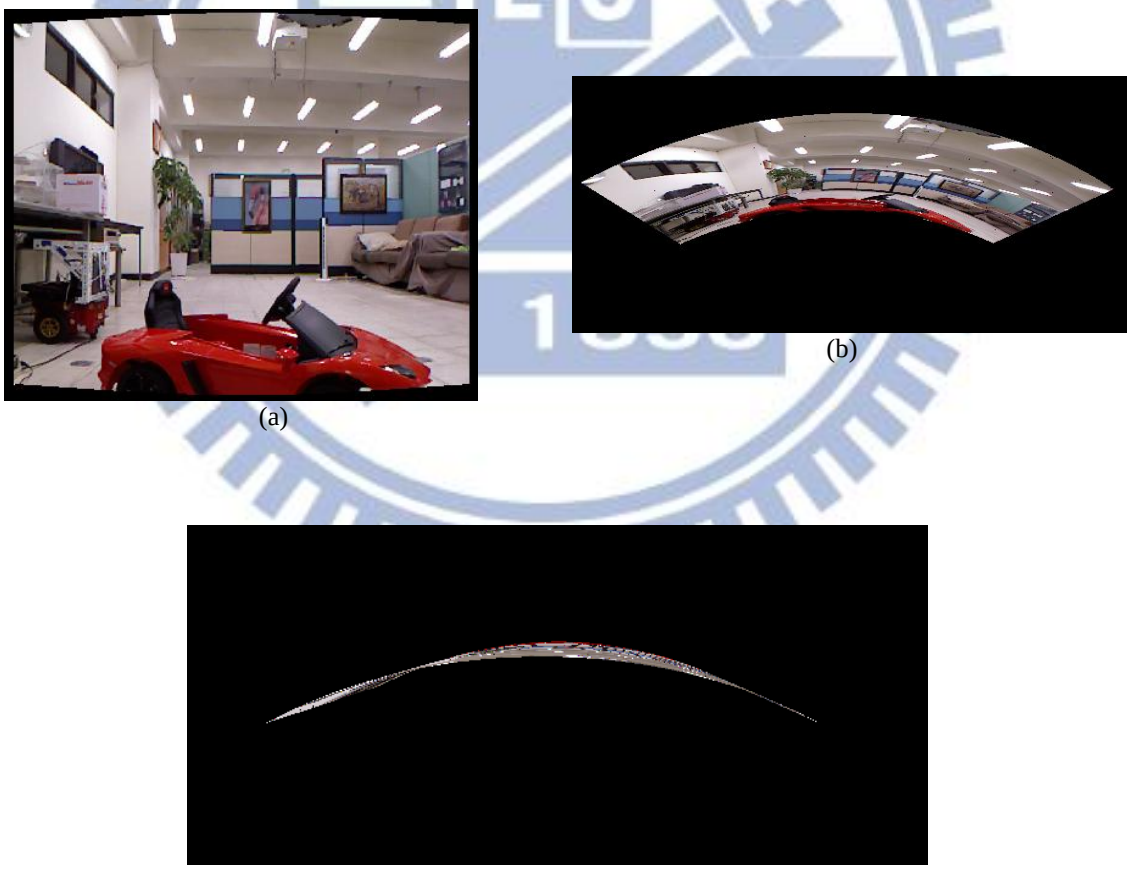
More specifically, the coordinates (x, y, z) of a 3D image point can be transformed into polar coordinates (r, α, β) by the following equations:

$$x = r\cos\alpha\sin\beta; \tag{6.2}$$

$$y = r\sin\alpha; \tag{6.3}$$

$$z = r\cos\alpha\cos\beta. \tag{6.4}$$

An example of the results of applying the above coordinate transformation to a color image acquired by a KINECT device is shown in Fig. 6.13. As can be seen from Fig. 6.13(c), the 2D image has become part of a spherical shape.



(a) (b) (c)

Fig. 6.13. A result of polar coordinate transformation applied to a color image acquired by a KINECT device. (a) Seen from the front. (b) Seen from the top with a

little slant. (c) Seen right from the top.

Now, we describe the proposed method for merging an extracted 3D image with a color background image. Firstly, each object in a given color image is removed. The removal scheme is implemented by the object detection method described by Algorithm 5.7. Specifically, if an object existing in the depth image is detected, then we remove it from the color image, so that a background image is obtained. After that, a merge of the extracted object in 3D form with the resulting background image is conducted. An algorithm implementing this idea of merging is proposed in the following.

Algorithm 6.3: merging 3D image with 2D background image.

Input: A 3D image I_{3D} and a set of background images I_B.

Output: A *2D-3D-mixed* image I_M resulting from merging I_{3D} and I_B.

Steps:

Step 1 Stitch the background images in I_B into a panoramic image I_P suing Algorithm 6.2.

Step 2 Transform the panoramic image I_P into the 3D polar coordinate system using Equations (6.2), (6.3), and (6.4) to get an image I_P' which becomes part of a spherical surface.

Step 3 The result is rendered by showing I_P' and I_{3D} in the same scene.

After the above algorithm is performed, the *2D-3D-mixed* image with objects in front a panoramic background image is obtained. Some experimental result of this algorithm is shown in the next section.

6.5 Experimental Results

The first part of the experimental results we show here is the removal of the nearby object from the color image. An example of such results are shown in Fig. 6.13, where Fig. 6.14(a) shows a color image with a nearby object (a red car) removed, and Fig. 6.14(b) shows the result of rendering the removed part as a 3D image. Also, the result of merging the 3D image and the background image is shown in Fig. 6.14. And Fig. 6.15 shows

(a) (b)

Fig. 6.14 The images before merged. (a) Nearby object in the color image being removed. (b) Rendering of the removed object part into the 3D space.

(a) (b)

Fig. 6.15 The result of merged data. (a) Seen on the front of image. (b) Seen from the top view.

Fig. 6.16 The result of merged 3D image on to background data which come from stitching by 3 background images.

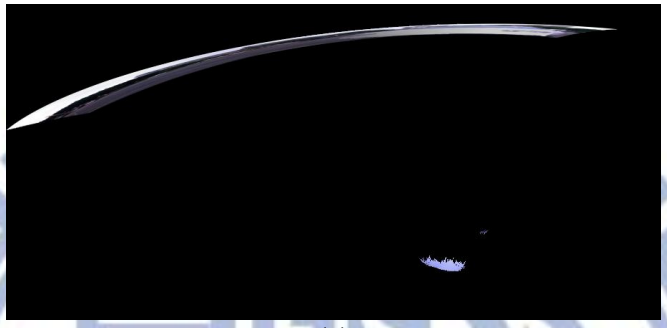
Another 2D-3D-mixed stitching result is shown in Fig. 6.17, where Fig. 6.17(a) shows the result of removing a car in front from a panoramic image constructed from stitching three background images using Algorithm 6.2. The car was not removed cleanly because the KINECT device senses no reflective signal from the glass portions of the car window and light covers and from car portion too far away. The removed part is rendered as a 3D image and shown in Fig. 16.7(b). Furthermore, we merge the 3D image of Fig. 6.17(b) into the panoramic image of Fig. 16.7(a) to yield Fig. 6.18, in which three views of the result are shown: a front view, a side view, and a top view. These views are useful for judging the relative position of the car with respect to the background scene.

(a) (b)

Fig. 6.17 Removal of a near-by car in a panoramic image. (a) Nearby car in the color image being removed. (b) Rendering of the removed car part into the 3D space.

(a) (b)

Fig. 6.18 Different views of result of merging 3D image and 2D panoramic background. (a) A front view. (b) A side view. (c) A top view.



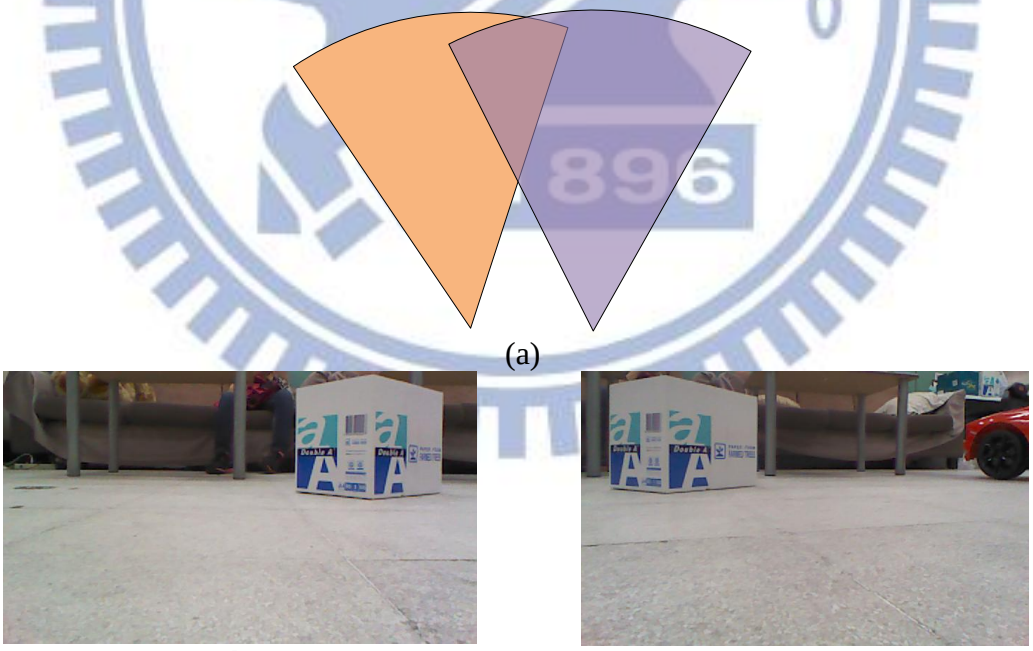
(c)

Fig. 6.19 Different views of result of merging 3D image and 2D panoramic background. (a) A front view. (b) A side view. (c) A top view. (Cont'd)

Chapter 7
Experimental Results and Discussions

7.1 Experimental Results

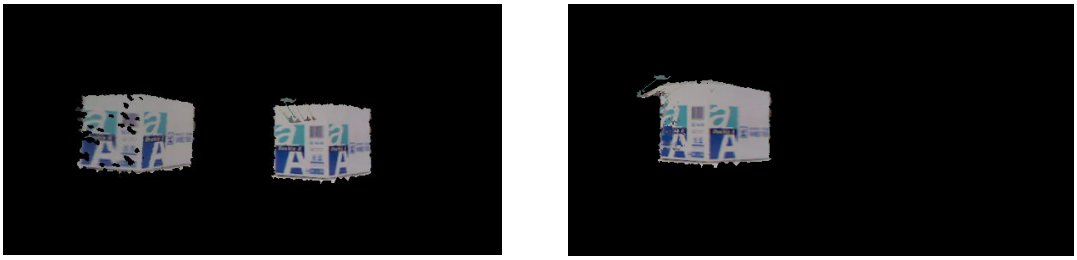
In this chapter, we present more experimental results obtained in this study. At first we describe some results of our experiment of KINECT device calibration. In the calibration steps, we set a box in the overlap region in the views of two neighboring KINECT devices, as illustrated in Fig. 7.1. result of calibration is like fig.7.2.



(a)

(b) (c)

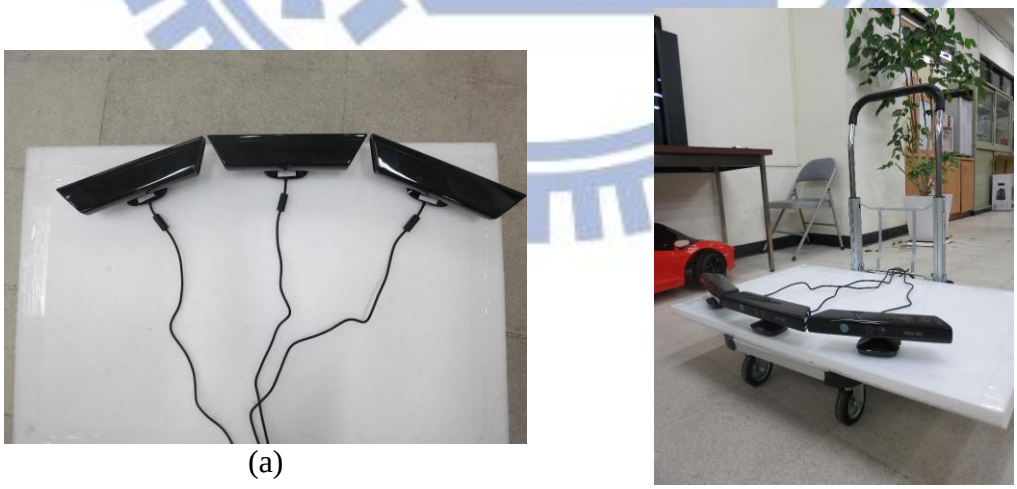
Fig. 7.1 A calibrate box placed in the overlap portion of views of two neighboring KINECT devices. (a) Illustration of the overlap area. (b) and (c) The box in the two color images acquired by the KINECT devices.



(a) (b)

Fig. 7.2 Result of calibration. (a) Two 3D images seen before calibration. (b) A single 3D image seen after calibration

Furthermore, we use a trolley to simulate a car driven on the road as shown in Fig. 7.3. Three KINECT devices are affixed on a board put on the trolley. Then we wheeled the trolley straight forward while a toy car is made to pass by. Three images acquired from the KINECT devices are shown in Fig. 7.4. The nearby objects in the images then are removed, with the result shown in Fig. 7.5. Then, we used the images in Fig. 7.5 as input to the stitching algorithm (Algorithm 6.2) to obtain a panoramic image as shown in Fig. 7.6.



(a) (b)

Fig. 7.3 Simulation using a trolley as a car and a toy car as a by-passing car. (a) three KINECT affixed on a board on the trolley. (b) The board is put on the trolley.

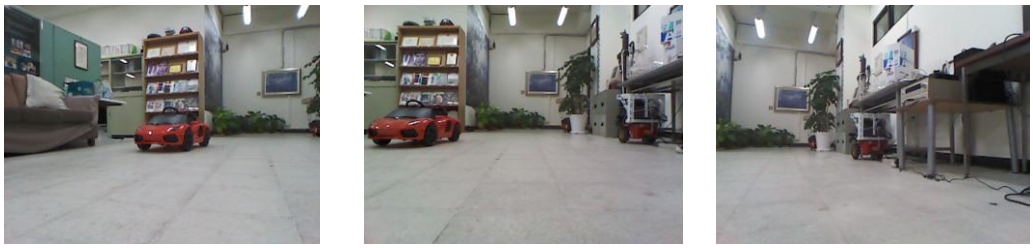


Fig.7.4 Three images acquired from the three KINECT devices.

Fig.7.5 Background images resulting from nearby object removal.

Fig. 7.6 A stitched image which will be the background.

On the other hand, the removed objects may be merged by using the obtained calibration information and proper coordinate transformations, with the result shown in Fig. 7.7. Finally, the background image of Fig. 7.6 was merged with the nearby objects of Fig. 7.7 as shown in Fig. 7.8. Another similar result of merging 3D images into panoramic background images is shown in Fig. 7.9.

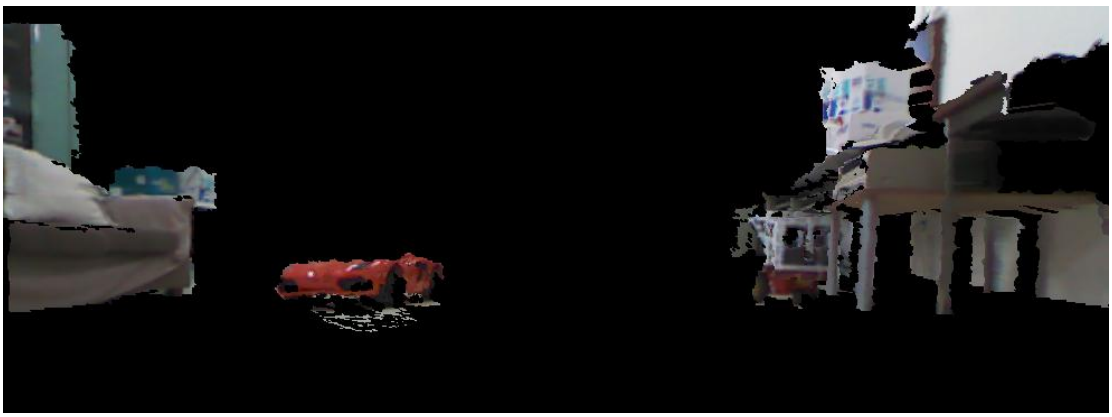


Fig.7.7 Merged objects in the scene.

Fig. 7.8 The nearby object merged with the panoramic background image.

(a)

Fig. 7.9 Another example of merging a panoramic background image with a 3D image (a car on the lateral side) (a) The panoramic background image. (b) 3D Model of detected car from images of two KINECT devices. (c) Merge result of (a) and (b). (d) A slightly-shifted view of the result of (c). (e) A top view of the result of (c).

(b)

(c)

(d)

Fig. 7.10 Another example of merging a panoramic background image with a 3D image (a car on the lateral side) (a) The panoramic background image. (b) 3D Model of detected car from images of two KINECT devices. (c) Merge result of (a) and (b). (d) A slightly-shifted view of the result of (c). (e) A top view of the result of (c).

(Cont’d).

(e)

Fig. 7.11 Another example of merging a panoramic background image with a 3D image (a car on the lateral side) (a) The panoramic background image. (b) 3D Model of detected car from images of two KINECT devices. (c) Merge result of (a) and (b). (d) A slightly-shifted view of the result of (c). (e) A top view of the result of (c). (Cont’d).

Finally, we show another result of stitching all the images acquired by the KINECT devices on the vehicle used in our experiment into a panoramic image, as shown in Figs. 7.10 and 7.11.

Fig. 7.12 A panoramic image.

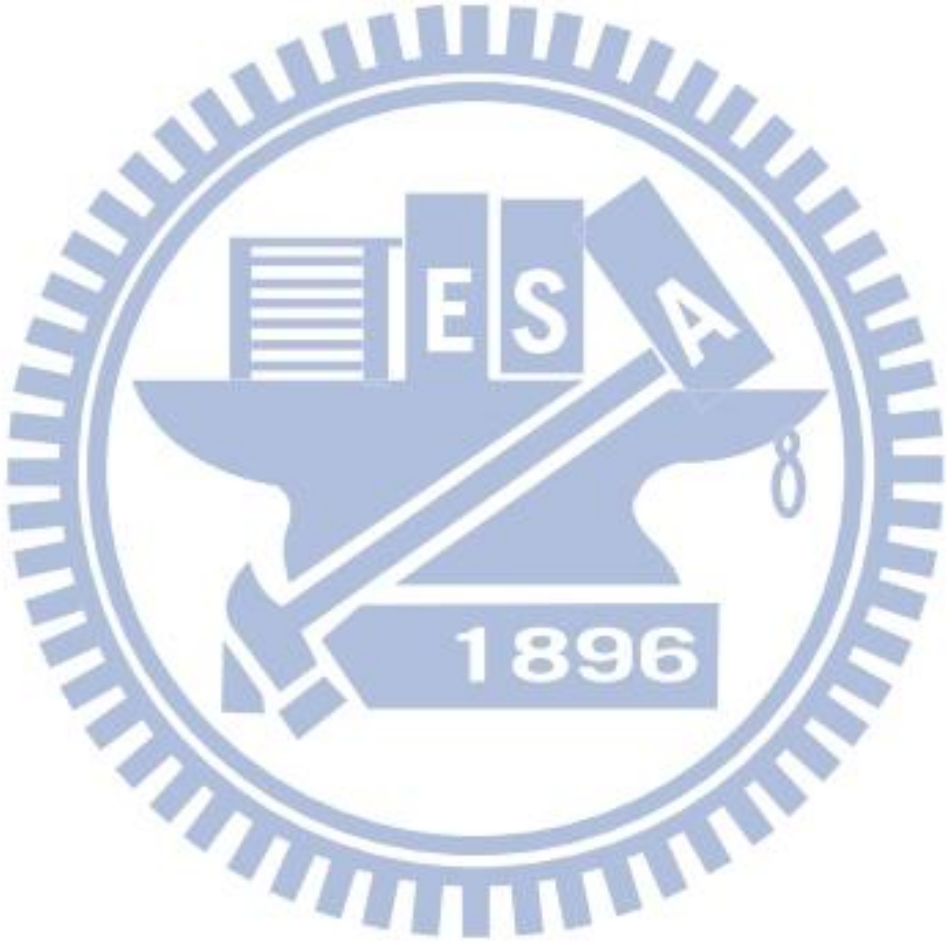
Fig. 7.13 A well-cut version of Fig. 7.9.

7.2 Discussions

In this section, we discuss some issues found in our experiments, and propose solutions to them if possible.

1. In the calibration steps, we use the ICP algorithm, and because the algorithm considers only the 3D points without color, some smooth calibration target may lead to bad results because of lacks of enough prominent features for matching. As a solution, extension of the distance-weighted correlation (DWC) to a 3D version may be used.
2. In the image recording step, the KINECT devices are dealt with by the system one by one, so the images acquired from two neighboring KINECTs have a little delay, though not much. But the totally FPS is low due to the use of 14 KINECT devices. To solve this problem, more computers are needed to speed up the image acquisition and processing speed, but then some computer communication algorithm more complex than the current used one should be designed.
3. In the outdoor environment, the infra light emitted by the KINECT device is interfered by the sun. Consequently, the time for good uses of the KINECT devices is restricted, but the proposed system can still work in a certain number of environments such as in the air port and big shopping mall, ..., etc. where the buildings are covered by huge roofs and the interior space is not interfered by the sun. Furthermore, the proposed system can still work in the afternoon in the outdoor environment, and it really help us to conduct off-line inspections of accidents occurring around the car.
4. The panoramic image constructed from images with nearby objects removed seems not good because there are “holes” created by the removed objects in the image. This can be solved by learning the background images in advance when no

car appear nearby. This is feasible if the EDR is used in local areas like school campus, large park area, fixed routes through cities or country sides, etc. Also, to place extracted 3D nearby objects like cars onto correct positions on the constructed panoramic image, we can use the navigation technique to find the current vehicle position on the map, or use the GPS.



Chapter 8
Conclusions and Suggestions for Future Works

8.1 Conclusions

In this study, we have proposed methods for construction of a 3D EDR (event data recorder) using multiple KINECT devices on the car. In this chapter, we make conclusions about our study and some suggestions for future works to improve our system or enrich the views acquired by our system. The conclusions are described as follows.

1. *A 3D imaging system* is proposed and constructed by affixing multiple KINECT devices around the car with different views to cover the 360^{o} surround.
2. *A method for transforming 2D images to 3D images* and a reverse version of the method are proposed based on the pinhole camera model.
3. *A method for calibration* of *the relationship between neighboring KINECT devices* using a calibration target is proposed, which is speeded up by the use of some learned information about the relative position of the KINECT devices.
4. *A method for 3D image merging* is proposed, which maps the images acquired by each KINECT device into the 3D space according to the relationship obtained from the calibration step.
5. *A method for speeding up rendering* of the 3D image for display is proposed, which utilizes the mesh data structure and the QEM algorithm.
6. *A method for panoramic image creation* by stitching multiple color images into a

single view by using a dynamic adjustment of the parameters.

7. *A method for combining 3D images with 2D panoramic scene images* is proposed, which can be used to render nearby objects and long-range views together.

8.2 Suggestions for Future Works

According to our experiences in the study, some suggestions for future works are made as follows.

1. It is worth studying automation of the calibrate task required for the proposed imaging system as well as adaptation of the calibration results to different environments.
2. It is interesting to improve the KINECT depth sensor for imaging in day time because so far the KINECT devices used in this study can only be used in environments with little sun light.
3. It's a challenge to embed the system into the car electronics system to improve the system processing speed.
4. A more precise transformation from the 2D coordinate system to a 3D one is needed.
5. Improvement on the proposed method of stitching images and combining nearby objects into panoramic images is needed to deal with more complicated environments.
6. The problem of removing nearby objects to create big “holes” in the resulting color images so that stitching of the images becomes impossible should be solved.

References

[1] A.D. Wilson and H. Benko, “Combining multiple depth cameras and projectors for interactions on, above, and between surfaces,” in *Proc. ACM symposium on User interface software and technology*, 2010, pp. 273-282.

[2] P. Biber, H. Andreasson, T. Duckett and A. Schilling, “3D modeling of indoor environment by a mobile robot with a laser scanner and a panoramic camera, ” *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, Sendai, Japan, vol.4, pp. 3430 - 3435, February 2004.

[3] H. Du, P. Henry, X. Ren, M. Cheng, D. B Goldman, S. M. Seitz, and D. Fox, “Interactive 3D modeling of indoor environments with a consumer depth camera,” in *Proc. 13th ACM International Conference on Ubiquitous Computing*, 2011, pp. 75-84.

[4] MIT, U. of Washington and Intel Labs. at Seattle, Visual Odometry For GPS-Denied Flight And Mapping Using A Kinect [Online], 2011, Retrieved from http://groups.csail.mit.edu/rrg/index.php?n=Main.VisualOdometryForGPS-Denied Flight.

[5] S. Izadi, R. A. Newcombe, D. Kim, O. Hilliges, D. Molyneaux, S. Hodges, P. Hokli, J. Shotton, A. J. Davison, and A. Fitzgibbon, “KinectFusion real-time dynamic 3D surface reconstruction and interaction,” in *Proc. ACM SIGGRAPH*, 2011, pp. 23-23.

[6] P. Besl and N. McKay, “A Method for Registration of 3D Shapes,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239-256, Feb. 1992.

[7] RUSINKIEWICZ, S. AND LEVOY, M. 2001. “Efficient Variants of the ICP

Algorithm," Proc. *3DIM* 2001.

[8] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy, "Geometrically Stable Sampling for the ICP Algorithm," Proc. *Int'l Conf. 3D Digital Imaging and Modeling*, Oct. 2003.

[9] Garland M and Heckbert P Surface simplification using quadric error metrics. Computer Graphics (*SIGGRAPH '97 Proceedings*) (1997), 209–216

[10] M. Brown and D. Lowe. Automatic panoramic image stitching using invariant features. *Intl. J. of Computer Vision*, 2007, pages 59–73.

[11] OpenCV library; http://code.opencv.org.

[12] Richard Szeliski. Image alignment and stitching: A tutorial. Technical Report MSR-TR-2004-92, Microsoft Research, December 2004.

[13] Heung-Yeung Shum and Richard Szeliski. Construction of panoramic mosaics with global and local alignment. *International Journal of Computer Vision*, 36(2):101-130, February 2000. Erratum published July 2002, 48(2):151-152.

[14] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk and Aaron Bobick. Graphcut Textures: Image and Video Synthesis Using Graph Cuts. To appear in Proc. *ACM Transactions on Graphics, SIGGRAPH* 2003.