OPTICAL THREE-DIMENSIONAL COORDINATION FOR MEDICAL APPLICATIONS

TEERAPORN KUSALANUKHUN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF MASTER OF ENGINEERING
(BIOMEDICAL ENGINEERING)
FACULTY OF GRADUATE STUDIES
MAHIDOL UNIVERSITY
2006

ISBN 974-04-7015-7

Thesis
Entitled

OPTICAL THREE-DIMENSIONAL COORDINATION FOR MEDICAL APPLICATIONS

Teeraporn Kusalanukhun
Miss Teeraporn Kusalanukhun
Candidate

Mr. Songpol Ongwattanakul,
Ph.D (Computer Engineering)
Major-Advisor

B. Mahaisavariya
Prof. Banchong Mahaisavariya,
MD., Dip Thai Brd Orthop Surg, FRCS(T)
Co-Advisor

Ittichote Chuckpaiwong
Assist.Prof. Ittichote Chuckpaiwong,
Ph.D (Mechanical Engineering)
Co-Advisor

Prof. M.R.Jisnuson Svasti, Ph.D
Dean
Faculty of Graduate Studies

Theeraporn Rubcumintara
Assist.Prof. Theeraporn Rubcumintara,
Ph.D (Materials Engineering & Science)
Chair
Master of Engineering Programme in
Biomedical Engineering
Faculty of Engineering

Thesis
Entitled

OPTICAL THREE-DIMENSIONAL COORDINATION FOR MEDICAL APPLICATIONS

was submitted to the Faculty of Graduate Studies, Mahidol University
For the degree of Master of Engineering
on
March 28, 2006

Teeraporn Kusalanukhun
Miss Teeraporn Kusalanukhun
Candidate

Mr. Songpol Ongwattanakul,
Ph.D. (Computer Engineering)
Chair

B. Mahaisavariya
Prof. Banchong Mahaisavariya,
MD., Dip Thai Brd Orthop Surg,
FRCS(T)
Member

Ittichote Chuckpaiwong
Assist.Prof. Ittichote Chuckpaiwong,
Ph.D. (Mechanical Engineering)
Member

Assist.Prof. Udom Tipayamontri,
Ph.D. (Physiology)
Member

Mr. Wongwit Senavongse
Ph.D. (Biomechanics)
Member

Prof. M.R.Jisnuson Svasti, Ph.D
Dean
Faculty of Graduate Studies
Mahidol University

Assist.Prof. Piya Rattanasuwan,
M.Eng.
Dean
Faculty of Engineering
Mahidol University

ACKNOWLEDGEMENT

I would like to acknowledge and express my sincere gratitude to all those who have inspired me to complete this thesis.

First of all, I greatly appreciate the invaluable helps provided by Dr. Songpol Ongwattanakul, my major-advisor, who leads me through this research and encouraged me to proceed with my thesis. I would like to thank him for his advice, support, and valuable time spent on editing this thesis as well as his respectable friendship.

I am especially grateful to my co-advisor, Dr. Ittichote Chuckpaiwong, whose helps, motivating suggestions, repetitive explanation of important issues encourage me all the time during my thesis.

I also would like to thank Prof. Banchong Mahaisavariya for providing valuable advice for the improvement. Special thanks are given to Assist Prof Udom Tipayamontri and Dr Wongwit Senavongse for editing this thesis and participating in the thesis defense.

I am deeply indebted to the BaRT Lab, Faculty of Engineering, Mahidol University, and all staff members in this lab for giving me the opportunity to pursuit my works and a lot of guidance with extensive comments to carry out the research.

I wish to thank all the faculties of the Biomedical Engineering Programme for providing me their knowledge and suggestions. Thanks also go to the administration staffs of the BME Programme and Mechanical Engineering for their kindness and supports.

I would like to give my special thanks to my co-worker at the Bart Lab, Eakkaluck Thammacharoen for his suggestion, inspiration and numerous helps.

Many thanks go to all my friends at Biomedical Engineering Programme for their love, support and all the good time that we have been shared.

Finally, the support and encouragement from my family were greatly appreciated. Thank to them for being always entrusted me in good fate.

Teeraporn Kusalanukhun

OPTICAL THREE-DIMENSIONAL COORDINATION FOR MEDICAL APPLICATIONS

TEERAPORN KUSALANUKHUN 4536776 EGBE/M

M.Eng. (BIOMEDICAL ENGINEERING)

THESIS ADVISORS : SONGPOL ONGWATTANAKUL, Ph.D. (COMPUTER ENGINEERING), BANCHONG MAHAISAVARIYA, MD., ITTICHOTE CHUCKPAIWONG, Ph.D. (MECHANICAL ENGINEERING)

ABSTRACT

Human vision cannot precisely determine the 3D coordinates of an object. The optical tracking with marker-based system, that is capable of capturing position and orientation of an object, is implemented to correct this error. Medical applications such as an image-guided surgery will benefit from this system, which can identify the position and orientation of a surgical device relative to the markers. This thesis provides and demonstrates a design framework for combining multiple sources of information. Initially, the system was implemented using MATLAB® to test the algorithm. Then the real-time application was created via Microsoft Visual C++, which includes the capturing frames from two cameras using Microsoft DirectShow, and the manipulation of two wireless active markers that are non-regular triangle in shape via a serial communication interface. This work utilizes image processing techniques from the OpenCV Library to obtain 3D positions and orientations of markers. The system is visualized the poses of markers in a virtual environment via OpenGL. The translation errors were tested and the average errors of the relative distances were in the range of sub-millimeter in the dimensions of width and height.

KEY WORDS : 3D COORDINATION/ STEREO CAMERA SYSTEM/ INFRARED OPTICAL TRACKING/ ACTIVE-PASSIVE MARKERS

85 P. ISBN 974-04-7015-7

การหาพิกัดในสามมิติด้วยแสงเพื่อการประยุกต์ทางการแพทย์
(OPTICAL THREE-DIMENSIONAL COORDINATION FOR MEDICAL APPLICATIONS)
ธีราภรณ์ กุศลานุคุณ 4536776 EGBE/M

วศ.ม. (วิศวกรรมชีวการแพทย์)

คณะกรรมการควบคุมวิทยานิพนธ์ : ทรงพล องค์วัฒนกุล, Ph.D. (Computer Engineering), บรรจง มไหสวริยะ, MD.(Dip Thai Brd Orthop Surg, FRCS(T)), อิทธิโชติ จักรไพวงศ์, Ph.D. (Mechanical Engineering)

บทคัดย่อ

โดยทั่วไปการมองเห็นของมนุษย์ไม่สามารถบ่งบอกตำแหน่งในสามมิติของวัตถุใดๆ อย่างแม่นยำได้ ระบบวัดพิกัดวัตถุมาร์คเกอร์ด้วยภาพจึงได้ถูกพัฒนาขึ้นเพื่อเพิ่มความแม่นยำในการระบุตำแหน่งดังกล่าว การหาพิกัดและการวางตัวใน 3 มิติของมาร์คเกอร์ สามารถนำมาประยุกต์เป็นต้นแบบเพื่อใช้ในระบบนำทางเพื่อการผ่าตัดได้ โดยการสร้างความสัมพันธ์ระหว่างตำแหน่งมาร์คเกอร์ และตำแหน่งพิกัดของอุปกรณ์ผ่าตัด ระบบวัดพิกัดวัตถุมาร์คเกอร์ด้วยภาพจะนำเสนอระบบ อุปกรณ์ และวิธีการ เพื่อให้สามารถระบุตำแหน่งพิกัดและทิศทางของมาร์คเกอร์ได้ โดยในขั้นแรกได้ทดลองระบบเพื่อทดสอบอัลกอริธึมด้วย MATLAB® หลังจากนั้นได้สร้างระบบเพื่อให้ใช้งานได้ในเวลาจริงด้วย Microsoft Visual C++ ซึ่งจะทำการจับภาพจากกล้อง 2 ตัวผ่าน Microsoft DirectShow วัตถุมาร์คเกอร์ที่ใช้งานนั้นได้ออกแบบให้เป็นแอคทีฟมาร์คเกอร์ซึ่งเปล่งแสงอินฟราเรดจำนวน 2 ชุดที่สามารถควบคุมการปิด-เปิดผ่านพอร์ทอนุกรมในแบบไร้สาย รูปร่างของมาร์คเกอร์ได้ถูกออกแบบให้เป็นลักษณะของรูปสามเหลี่ยมด้านไม่เท่า ซึ่งจะสามารถบอกการวางตัวของมาร์คเกอร์ได้อย่างชัดเจน หลังจากนั้นภาพที่ได้จะถูกนำไปประมวลผลด้วยไลบรารี OpenCV เพื่อคำนวณหาตำแหน่งและทิศทางของมาร์คเกอร์ในสามมิติ ค่าพิกัดที่ได้จะนำมาแสดงผลทางกราฟิกสามมิติ โดยจะเป็นการสร้างภาพจำลองของวัตถุสมมติหรือมาร์คเกอร์ โดยใช้ OpenGL นอกจากนี้ได้ทำการตรวจสอบค่าความผิดพลาดของระบบ โดยการเปรียบเทียบตำแหน่งของมาร์คเกอร์ที่คำนวณได้ กับตำแหน่งของมาร์คเกอร์ที่ตั้งไว้ โดยค่าเฉลี่ยค่าความผิดพลาดจากระยะสัมพัทธ์ของตำแหน่งมาร์คเกอร์เทียบกับตำแหน่งอ้างอิงคงที่ในระนาบซ้ายขวาและในแนวสูง อยู่ในเกณฑ์ไม่เกิน 1 มิลลิเมตร

85 หน้า. ISBN 974-04-7015-7

CONTENTS

CONTENTS (Continued)

CONTENTS (Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

CHAPTER I
INTRODUCTION

1.1 Problem Statement

Orthopedic surgeries, such as a closed intramedullary nailing for femoral shaft fractures, are common and are difficult operations which require high expertise surgeons. A stable fixation of the nail is a crucial process resulting in the appropriate alignment of the femur and reducing further complications. The most difficult procedure is to place a locking screw through the nail which requires the location of the distal holes. The current practice for this type surgery usually relies on the fluoroscopic imaging, which requires multiple exposures of X-ray imaging to reveal the patient's internal anatomy and the surgical instruments during surgery. To make small incisions and to stabilize the fixation, several exposures of fluoroscopic imaging are required to ensure the proper alignment [1]. For this reason, it may have an adverse effect to the health of both surgeon and patient. Any novel computer aided surgery systems that allow high precision operation environment and minimum X-ray exposure are highly preferable. Therefore, the image-guided surgery is a promising technique for the orthopedic surgeon. The image-guided surgery is the system that can track positions of the surgical instruments cooperated with the position of the patient by employing optical tracking system. In general, the image-guided surgery system consists of an optical tracking unit, the tracked objects, the imaging device, the processing and display unit [2] as illustrated in Figure 1.1. The surgical navigation by image-guiding can reveal the position and orientation of the patient's internal organs and unobservable surgical instruments during the operation. The surgical planning data may also be included to the surgical guiding system for the comparison purposes. Once the transformation of the coordinates is established, the registration of 3D coordinates of markers, representing the surgical tools during the surgical operation, can be compared with the surgical plan. Therefore, the surgical procedure can be performed accurately as plan. The visualization system, which portraits the pose

estimation of the objects in three-dimension, becomes a major challenge. During surgery, surgeon mentally correlates the two-dimensional intra-operative images to three-dimensional patient anatomy and surgical tools. The offered visualization system can show the superimposed coordination of the available information, which assists the surgeon to operate more precise with the correct 3D locations.

In this thesis, the focus is on the optical tracking system that detects markers. The system includes an imaging system and a few markers. This imaging system employs stereoscopic vision which composes of two cameras that are highly sensitive in infrared region. The markers may be attached or positioned at a fix location on the surgical instrument and/or the patient's organ. The goal of the system is to accurately identify the 3D position of the markers from two views of images that are simultaneously captured. Marker detection is based on infrared optical tracking, which can perfectly distinguish the markers and background in the surgical environment.

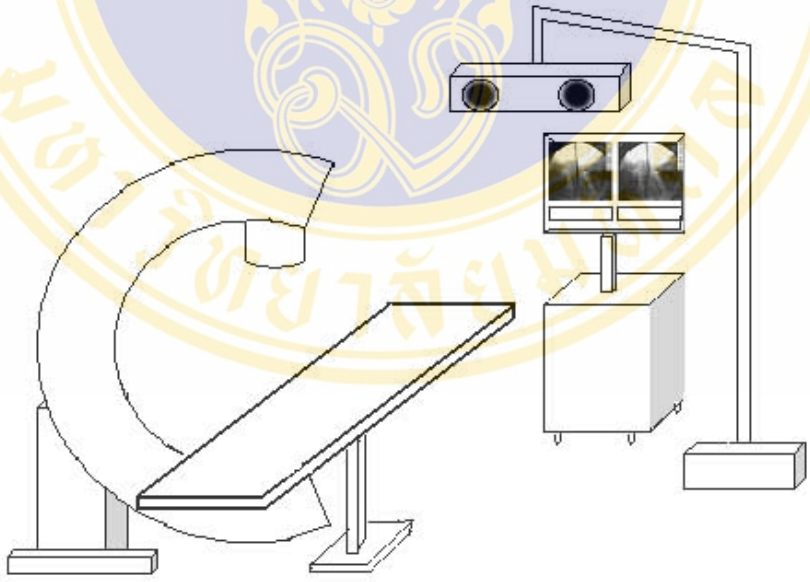


Figure 1.1 Image-guided surgery from virtual fluoroscopic system

1.2 Objectives

The main objective of this research is to find appropriate procedures and algorithms that are suitable for the implementation of surgical navigation applications. The mission is to develop a high precision 3D coordinate acquisition experimental platform that allows the measurement of position and orientation errors. An extensive error study and analysis will be conducted to ensure the robustness of the system functionality, integrity, and accuracy.

1.3 Scope

The infrared optical tracking by stereoscopic cameras will be crated as a prototype. The camera system will be installed at a static location. This tracking system is capable of tracking specific markers and identifies their 3D positions and orientations. These coordinate and orientation estimations can be correlated to the position of the defined tool.

1.4 Expected results

The expected results include:

- A prototype of a surgical guiding system that employs stereo imaging to acquire 3D coordinate of a marker, and appropriate procedures and algorithms in the computation of the extracted coordinate.
- An experimental platform for measuring accuracy of the computed 3D coordinates that allow comparison of the actual and computed coordinates.

CHAPTER II
LITERATURE REVIEW

This chapter presents the fundamental theories of components used for the implementation of the 3D coordinate acquisition system such as camera model, camera calibration, and stereo camera system. The tracking system based on marker is also discussed. The image processing and the visualization system that are involved are explained as well.

2.1 Camera Model

Camera is usually illustrated as a pinhole model as shown in Figure 2.1.

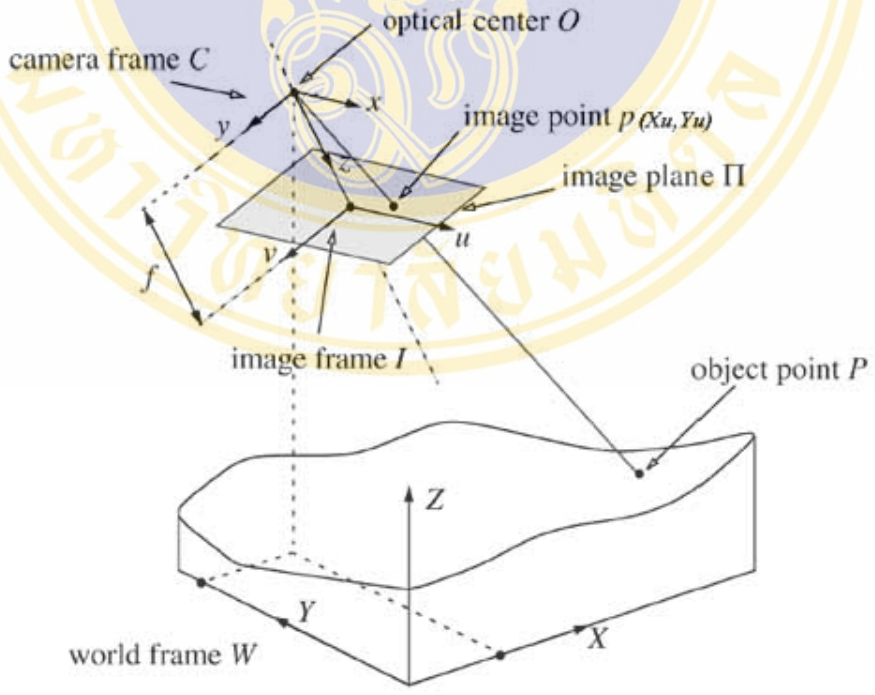


Figure 2.1 A pinhole camera model (adapted from [3])

The camera model is illustrated for the transformation from 3D world coordinate to camera coordinate system. The point from each plane can be defined as follows [4]:

- (X_W, Y_W, Z_W) is the 3D coordinate of object point P in the world coordinate system.
- (x, y, z) is the 3D coordinate of object point P in 3D camera coordinate system, which has a center at point O, called optical center.
- (u, v) is the image coordinate system. The distance between front image plane and optical center is focal length, f.
- (x_u , y_u) is the undistorted image coordinate of (x, y, z).
- Furthermore, there are (x_d , y_d), which is the actual image coordinate due to lens distortion and
- (x_f , y_f), the coordinate used in computer, which is the number of pixels in frame memory.

Many parameters require calculation and calibration to correct the deformation between the ideal projection and the image coordinate in the image plane. The entire transformation from (X_W, Y_W, Z_W) to (x_f , y_f) can be demonstrated in Figure 2.2.

First, the 3D world coordinate is transformed to 3D camera coordinate system by a rigid transformation,

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \vec{\mathbf{R}} \begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix} + \vec{\mathbf{T}} \qquad (2.1)$$

where $\vec{\mathbf{R}}$ is the rotation matrix and $\vec{\mathbf{T}}$ is the translation matrix. These are the calibrated parameters.

Next, the 3D camera coordinate is transformed to undistorted or ideal image coordinate using perspective projection with pinhole camera geometry.

$$x_u = f \frac{x}{z} \qquad (2.2a)$$

$$y_u = f\frac{y}{z} \tag{2.2b}$$

The effective focal length f is the calibrated parameter.

Then, the radial lens distortion D_r is calculated from

$$x_d = x_u + D_{rx} \tag{2.3a}$$

$$y_d = y_u + D_{ry} \tag{2.3b}$$

where

$$D_{rx} = x_d\left(\mathrm{k}_1 r^2 + \mathrm{k}_2 r^4 + \mathrm{k}_3 r^6\right)$$

$$D_{ry} = y_d\left(\mathrm{k}_1 r^2 + \mathrm{k}_2 r^4 + \mathrm{k}_3 r^6\right)$$

$$r = \sqrt{x_d^2 + y_d^2}\,.$$

The distortion coefficients k_i are calibrated parameters, which one or two coefficients are enough to compensate the distortion.

However, the centers of lens curvature are not always collinear, which present another distortion called tangential distortion D_t [5]. It can be expressed as

$$D_{tx} = 2\mathrm{k}_4 x_d y_d + \mathrm{k}_5(r^2 + 2x_d^2)$$

$$D_{ty} = \mathrm{k}_4(r^2 + 2y^2) + 2\mathrm{k}_5 x_d y_d$$

So the image coordinate can be approximated as the following form:

$$x_d = x_u + D_{rx} + D_{tx} \tag{2.3c}$$

$$y_d = y_u + D_{ry} + D_{ty} \tag{2.3d}$$

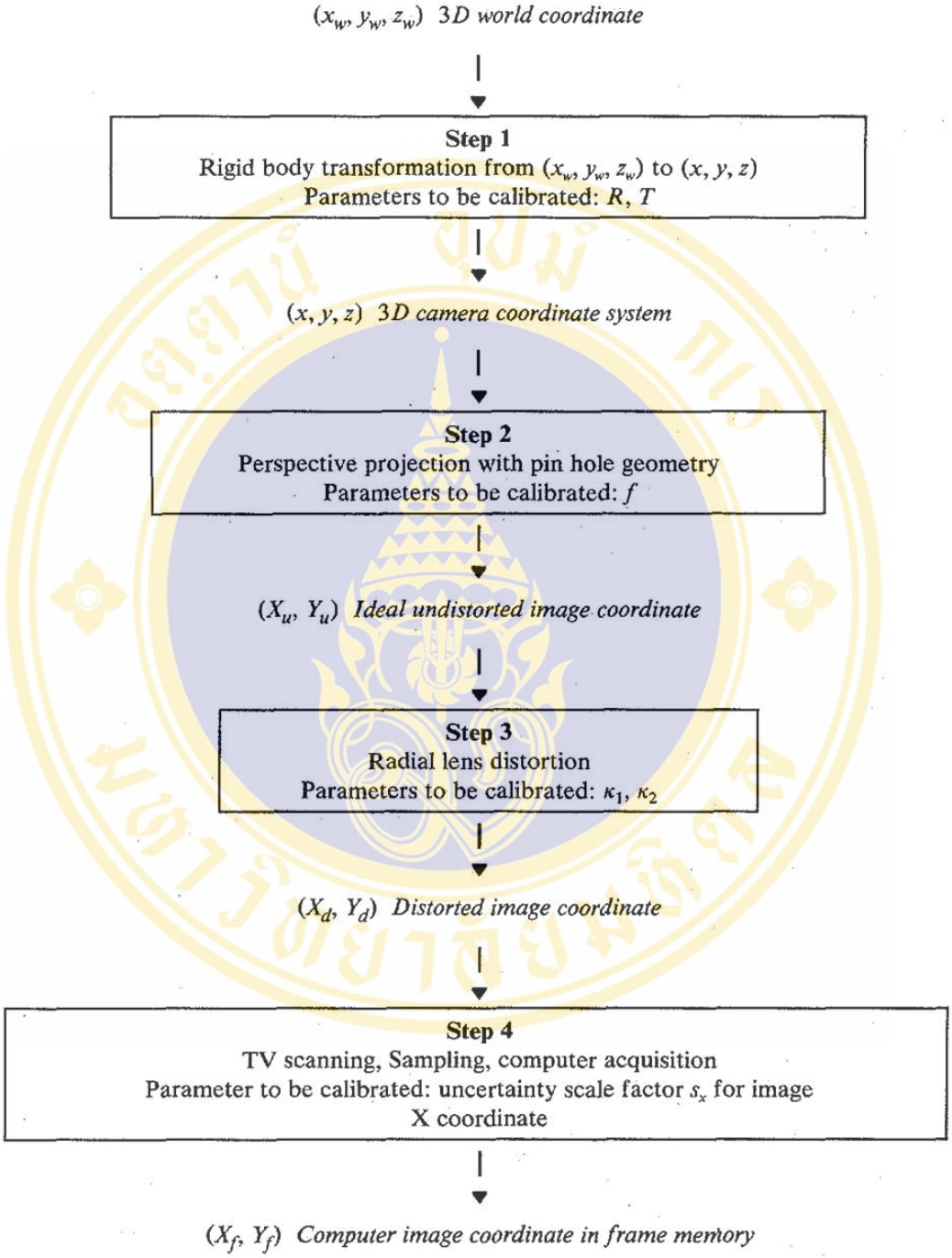


Figure 2.2 Four steps of transformation from 3D world coordinate to computer image coordinate [4].

The final step is the transformation from image coordinate (x_d, y_d) to computer image coordinate (x_f, y_f).

$$x_f = s_x d'^{-1}_x x_d + C_x \quad (2.4a)$$

$$y_f = d_y^{-1} y_d + C_y \quad (2.4b)$$

where

(x_f , y_f) are row and column numbers of the image pixel in computer frame memory,

(C_x, C_y) are row and column numbers of the center of computer frame memory,

$$d'_x = d_x \frac{N_{cx}}{N_{fx}}$$

d_x is the distance between each pixel in X direction,

d_y is the distance between each pixel in Y direction,

N_{cx} is the number of sensor elements in the X direction (from datasheet of the camera), and

N_{fx} is the number of pixels in a line as sampled by the computer.

The uncertainty image scale factor s_x is the calibrated parameter.

Consequently, the parameters that need to be found and calibrated can be classified into two categories, the extrinsic parameters and the intrinsic parameters. The extrinsic parameters are the rotation parameters in **X-, Y-,** and **Z-axes** and the translation parameters in **X-, Y-,** and **Z-axes** for the transformation from 3D world coordinate system to camera coordinate system. The intrinsic parameters are the parameters for the transformation from 3D camera coordinate system to the computer image coordinate system. There are six parameters, which are the effective focal length (f), the lens distortion coefficient (k_i), the uncertainty scale factor from camera scanning (S_x), and the center of the computer image coordinate (C_x, C_y).

2.2 Camera Calibration

Camera calibration is the necessary step to acquire the relationship between object in world coordinate and image coordinate for 3D computer vision. The transformation of the geometry includes intrinsic parameters or the optical camera characteristics and extrinsic parameters or the position and orientation of the camera. There are various methods to determine these parameters. Generally, they are classified into 3 approaches, which are linear, nonlinear and hybrid methods. Linear techniques determine the transformation parameters by solving simple linear equations [6]. The methods are fast but inaccurate because lens distortions are not included in the calculation. So the nonlinear techniques are presented by using more unknowns and nonlinear iterative optimization. They are more accurate but take more computational time. Then the hybrid methods appear with several steps of calibration procedure. Tsai [4] proposes the two-step method. The first step is to determine the extrinsic parameters using a linear method. The second step is to solve the rest of parameters by a non-linear method. Furthermore, there are additional three- [7] and four-step [5] methods. The four-step procedure is an expansion to the two-step method. The first two stages are similar to the original two-step. The third step offers the compensation of the distortion from circular features in the perspective projection. And the fourth step is presented for correcting the distorted image coordinates. The image correction is done by using the inverse model based on the calculated camera parameters from previous steps and a linear method can solve to correct arbitrary image coordinates.

2.3 Stereo Camera System and Stereo Triangulation

Stereo camera system is the use of two cameras to find 3D position of an object from acquired images. Stereo imaging offers more benefits over the images from one camera. For examples, it provides the depth component of estimating 3D positions and it can help to solve the scale problem when an object looks smaller in the image as it moves away. The problems of this system are correspondence and reconstruction. The correspondence issue is to determine which element in the left image corresponds to which element in the right image. The reconstruction issue is to determine the conversion from 2D images into a 3D map of the scene based on the

geometry of the stereo system and on the disparity map. Disparity is the computed difference between corresponding objects. Not only the intrinsic parameters of each camera but also the extrinsic parameters are needed to find for describing information of relative position of the two cameras.

After we obtain the information from calibrated stereo cameras, the next step is to compute the 3D position of a point in space from the known disparity map and the geometry of the stereo cameras. This process called Triangulation is to find the intersection of two projection lines in two images of a point in space. The simple way is to use the mid-point method [8]. The mid-point of the common perpendicular to the two lines is chosen by dividing the common perpendicular in proportion to the distance from the two camera centers. The illustration of this method can be modeled as Figure 2.3.

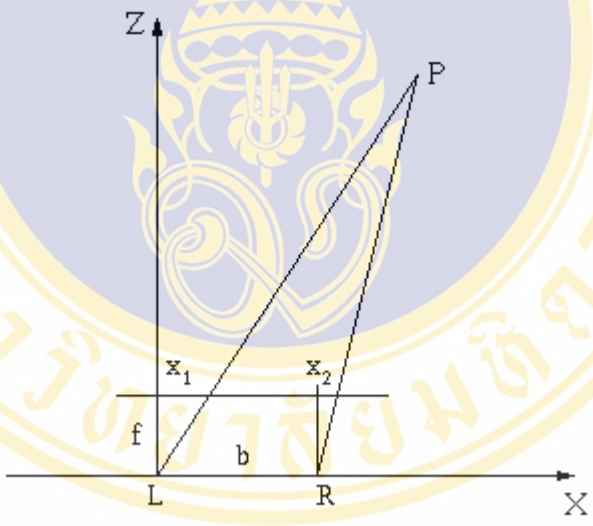


Figure 2.3 Standard models of stereo cameras [9]

Let left camera reference system be the 3D world reference frame. The right camera is translated and rotated with respect to the left camera frame. In this case the two optical axes are parallel, which the translation of the right camera is only on the **X-axis**. The optical axes lie on the **XZ** plane. Also, let *f* be the focal length of both camera and *b* is the distance between the two lens centers. The stereo triangulation can be expressed as:

$$Z = \frac{(b \times f)}{(x_1 - x_2)}$$

$$X = x_1 \times \frac{Z}{f} \tag{2.5}$$

$$Y = y_1 \times \frac{Z}{f}$$

2.4 Infrared Optical Tracking

The main purpose of a tracking system is to determine the three-dimensional coordinate of an object. There are many types of 3D tracking, for example, acoustic tracking, mechanical tracking, magnetic tracking, and optical tracking. Each method has different advantages and disadvantages in aspects of performance and price as discriminating in Table 2.1.

Table 2.1 Comparison of tracking techniques [10].

Detection Techniques	Description	Advantages	Disadvantages
Acoustic Tracking	Based on measurement of time-of-flight of a sound signal from an emitter to a receiver.	- Inexpensive	- Low accuracy - Suffer from acoustic reflections in surrounded conditions
Mechanical Tracking	Object is connected to a mechanical armature. When the object moves, its position is known from related device movement.	- Fast - High accuracy	- Encumbered movement - Limited working area

Table 2.1 Comparison of tracking techniques (Continued)

Electromagnetic Tracking	Use the distortion of electromagnetic signals to determine position and orientation relative to source coils.	- Require no contact to the sensor - Not require direct line of sight between the source and sensor - High accuracy	- High cost - Accuracy affected by electromagnetic generator devices such as motor - Accuracy affected by Earth's magnetism
Optical Tracking	Use 2 or more cameras to detect visual signals and involve camera information and epipolar geometry to calculate positions	- Require no contact to the sensor - Easy to develop - High accuracy - inexpensive	- Limited by light source intensity - Require clear line of sight

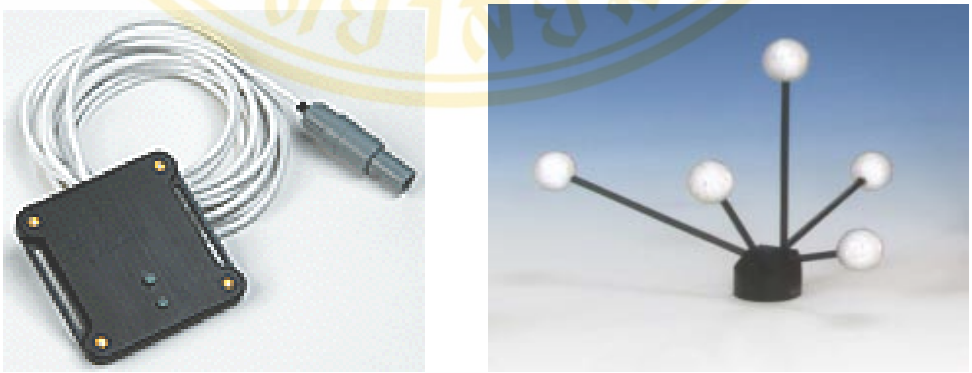
The system that we are interested is the optical tracking, which involves imaging with cameras mounted at a fixed location and image processing. Optical tracking has many benefits over other tracking. For instance, it offers a wireless interaction, presents accuracy measurements even with low cost, and requires no additional sensors when there are additional tracking objects. However, there are some disadvantages from applying the optical tracking system. For example, it may have an occlusion problem, needs computational time, and may require specific set-up on the environment such as the illumination on the scene [11]. The optical tracking is widely used in many applications for example; augmented reality, computer animation, human motion analysis, and image guided-surgery. There are various methods to detect objects such as color-based, shape-based, feature-based method.

Our system employs the infrared optical tracking system to extract the differences between markers and background. The infrared tracking is widely used in

both research institutes and commercial systems like Polaris [12] or Trackd® from A.R.T. tracking system [13].

2.5 Markers

The infrared optical tracking system can use either active or passive tracked objects. Active optical tracking is the use of light-emitting diodes such as Infrared-Light Emitting Diodes (IR-LEDs) as markers or tracked objects. (See Figure 2.4(a)) Passive optical tracking can be retroreflective materials in forms of bead, paint, or tape attached to markers (See Figure 2.4(b)). The wavelength used for activating the retro-reflective marker is normally 880 nm. There are some differences for using passive or active markers. Choosing markers depends on applications and constraints of each system. Passive marker presents several advantages for example, it is simple to use, it is convenient to locate on a patient, and it requires no power. On the other hand, passive marker may have the precision problem due to the partial occlusion and small distortion from infrared reflection, which lead to miscalculation of marker centroids. Alternatively, an active marker works even though there is an obstruction at the line of sight between marker and cameras and it can be controlled on-off state to help in tracking algorithm. However, the active marker still requires power cable.



(a) **(b)**

Figure 2.4 (a) Active markers [12], (b) Passive markers [13]

Moreover, the tracked markers have some features needed to be considered such as the field of emission, size and shape of the marker, the inter-marker distance constraints and the occlusion problem.

Therefore, wireless active IR-LEDs markers are used to solve the problem of the inconveniences and offered alternating control if there are more than one set of markers. To design the wireless active markers, there are some conditions needed to be considered as follows [14]:

1. Markers should be small enough to attach to any medical tools.
2. Wireless communication must have some error detection.
3. Markers should be compatible to any personal computer and Microsoft Windows.
4. Markers can communicate at least 10 meters and consume low power.

2.5.1 Wireless active marker system

From the requirements mentioned above, the power from battery is used to make the system small and low power consumption. We chose a tiny PIC12F509 Microcontroller from Microchip Technology Inc. [15] to control and communicate the module via wireless serial port (TLP-434 and RLP-434 from Laipac Technology Inc. [16]) at 434 MHz frequency. The wireless marker system consists of two parts: master transmitter and slave receiver or active marker.

2.5.2 Transmitter

PIC12F509 Microcontroller communicates to personal computer via serial port (RS232) to control the On/Off of the markers. Furthermore, the microcontroller arranges the sending data for transmission through module TLP-434 by Amplitude Shifted Keying (ASK) Modulation. The additional data is added to the preamble or header, which are the 20 bits of synchronization between the transmitter and receiver, 1 byte of active marker address, 2 bytes of the On/Off data of IR-LEDs, and 1 byte of CRC checksum for error detection of all data. All data is encoded by Manchester Encoding. (See Figure 2.5)

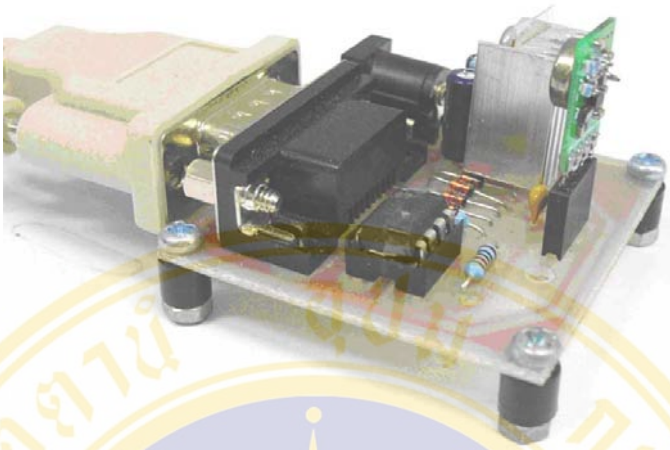


Figure 2.5 Transmitter circuit

2.5.3 Receiver

Data packet transmitted in the air is received by module RLP-434. This module transforms the ASK modulated data signal to digital signal for the microcontroller PIC12F509. The microcontroller will adjust itself to synchronize to the transmitted data by checking 20-bit preamble. When they are synchronized, the microcontroller will save the next 4 bytes to the internal memory for error checking. The first checking is to test whether the data in the first byte is the same as the identification number of the receiver or not. If they are different, all of this data packet will be ignored and the receiver will be reset. If they are alike, the microcontroller will check the correctness of data by CRC checksum in the last byte. If the data is wrong, the receiver will be reset. If the data is correct, the microcontroller will serially shift the data to the IC 74LS595, which translates the data to the On/Off state of the IR-LEDs that connected to this IC. (See Figure 2.6)

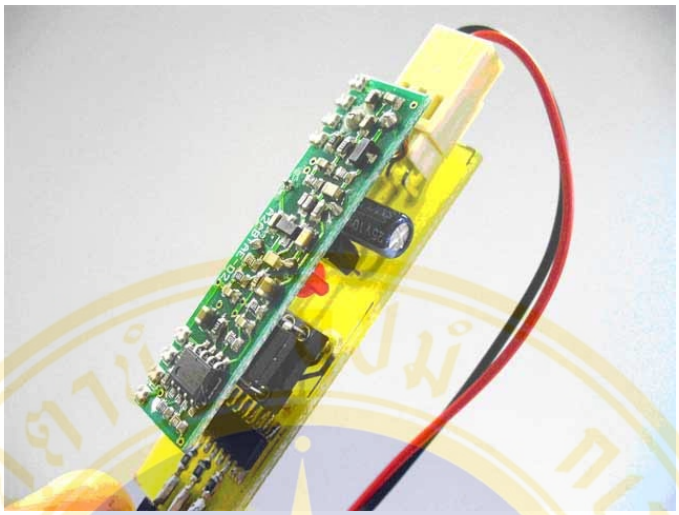


Figure 2.6 Receiver circuit

2.6 Related Researches

For the implementation in the image-guided surgery, the medical instrument and the markers are calibrated into the same coordinate system. During tracking, the physical positions of the markers in world coordinate system are mapped to the calibrated positions in the instrument coordinate systems using registration techniques. The error from the instrument tracking is a crucial factor of the overall system accuracy [17]. There are some researches that have tried to detect and identify markers with greater accuracy. From Schwald and Figueiredo [18], they set up the system using stereo cameras with both passive markers (7 retroreflective spheres), and active markers (IR-diodes). The algorithm for detect the markers is to locate the correspondences between the markers and the tracked points. The correspondences are the comparing distances between each point of tracked points and the stored distances of each point of the model. They also develop a learning algorithm to recognize the markers in 3 cases. First is to learn markers in different poses without moving. Second is to learn the markers with different moves. In this case, markers may have the occlusion. And third is to add more markers to the system. The learning algorithm is used as a reference to compare the measured distance incorporated with the positions of markers from the previous frame. Their algorithms showed good results within 1 millimeter error. However, there is one constraint that markers should be seen at least 3 points to attain the accuracy. Another research for detecting infrared markers is from

Klaus Dorfmuller [11]. He developed the optical tracking system to track human motion for application in augmented reality. He presented two approaches, which are tracking head and hand with the rigid body from active and passive markers through infrared light and tracking non-rigid objects like hands. For the rigid-body tracking, he tested the algorithm with the passive-spherical markers form in non-regular triangle. He identified each marker by calculating distance between each 3D point and calculating differences between each measured distance and pre-known distances of triangle lengths. Then, the differences or error distances are compared to the error threshold to consider which possible triangle length is in a candidate set. The set that offers smallest error value is a set of chosen cyclic path. The results showed that maximum error is 2.5 cm in a range of 4 m, which provided quite accurate outcome.

2.7 Image Processing and Implementations

When the two cameras are connected to the computer, there must be some interfaces for manipulating the camera settings, acquiring images and image processing to determine the coordinates of markers in three-dimension. First, we use MATLAB® to test the feasibility of the system. After that, we choose Microsoft DirectShow and OpenCV library to implement the system in Microsoft Visual C++ for real-time application.

2.7.1 Image Processing Algorithms

2.7.1.1 Thresholding

Thresholding is a method to distinguish the interesting objects in an image from background by creating a binary partitioning of the image intensities. The procedure is accomplished by comparing the desired intensity value called threshold to intensities of all pixels. The pixels that have intensity value greater than the threshold are assigned to one class. Otherwise, they are assigned to another class.

The threshold value can be set manually or automatically to a percentage of the maximum intensity within the image. The simple thresholding can partition the image into only two classes. However, some drawbacks could occur. For

example, noise and intensity variation may be sensitive to the threshold segmentation, which can affect to the connectivity of desired component.

2.7.1.2 Connected Components Labeling

Connected components labeling scans an image and groups its pixels into components based on pixel connectivity [19]. All pixels in a connected component have the same intensity values and connect to each other through its neighbors. Each pixel in the same group is labeled with a gray level or a color (color labeling) that is assigned as shown in Figure 2.7.

Connected components labeling can be used in both binary and grayscale image with different measures of connectivity such as 4-, or 8-connectivity. The algorithm scans through the image pixel-by-pixel to find the connected regions. For a binary image, it searches for connected pixels that have intensity value V = {1}. For a grayscale image, the same component may have the intensity in a range of values such as V= {51, 52, 53, ... , 77, 78, 79, 80}. For the following, the image is assumed as the binary image to ease the explanation. Connected component labeling scans across each row until finding the pixel P that has intensity value V= {1} and examines the neighbors as follows:

- If all neighbors have intensity values V= {0}, then create a new label to P.
- If there is only one neighbor that has V= {1}, then assign the label of that neighbor to P.
- If there are more than one neighbor that have V= {1} and have different labels, then assign one of the labels to P and make a note of equivalences.

When the image is all scanned, equivalent label pairs are sorted into equivalence classes and the labels are reassigned to unique labels. Then scan the image again and replace the labels with the distinctive labels from equivalence classes.

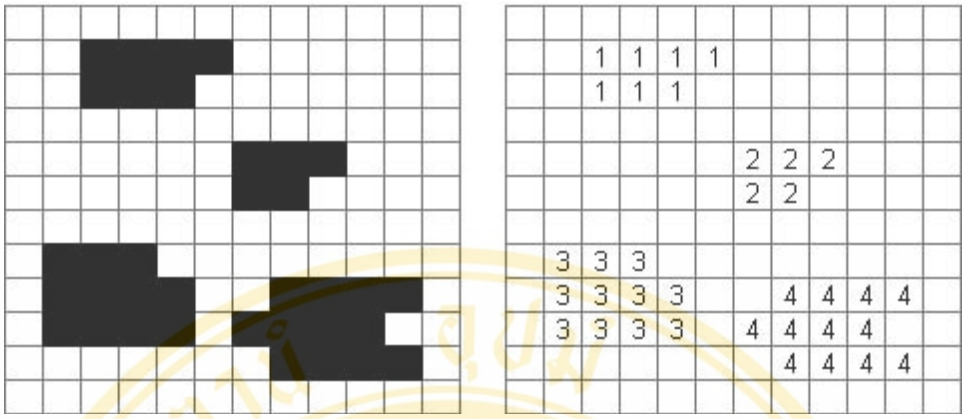


Figure 2.7 An example of connected component labeling

2.7.1.3 Finding centroids of markers

After the labeling is performed to locate the area of markers, centroids of markers are determined to represent the position of the marker. The calculation of centroids has the same formula as the computation of center of mass, which are

$$\bar{x} = \frac{\sum x\ m}{\sum m} \tag{2.6a}$$

$$\bar{y} = \frac{\sum y\ m}{\sum m} \tag{2.6b}$$

where $\bar{x}$ is a centroid position of a marker in x axis.

$\bar{y}$ is a centroid position of a marker in y axis.

x is a pixel position of marker area in x axis.

y is a pixel position of marker area in y axis.

m is the intensity value of marker area in each pixel.

2.7.2 Image Acquisition Toolbox and Image Processing Toolbox in MATLAB®

For the implementation of this project, we initially used MATLAB® to test the algorithms. MATLAB® is a high-performance language for technical computing, which contains many toolboxes for computation, visualization, and programming. The

major toolboxes, which are helpful to the project, are Image Acquisition Toolbox and Image processing Toolbox.

Image Acquisition Toolbox supports a variety of image acquisition operations as follows [20]:

- Acquiring images from image acquisition devices such as frame grabbers, USB-based Webcams, and FireWire Cameras
- Viewing a preview of the live video stream
- Triggering acquisitions
- Configuring callback functions that execute when certain events occur
- Bringing the image data into the MATLAB® workspace

We employed this toolbox to capture frames from two cameras. The basic steps to accomplish the operation must be performed for the following:

- Install and configure the image acquisition device.
- Retrieve hardware information that uniquely identifies the image acquisition device to the Image Acquisition Toolbox. To get the information, '*imaqhwinfo*' function is used, which will tell us about the adaptor name, device ID, and supported video format.
- Create a video input object to establish the connection between MATLAB® and an image acquisition device. The '*videoinput*' function is used with the retrieved information about adaptor name, device ID, and video format. The video input object can be used to control the image acquisition process.
- Preview the video stream (Optional). After creating the video input object, one may want to see the preview output of the video stream for further adjustments i.e., the camera position, lights, focus, or other image acquisition setup. The '*preview*' function opens a window and displays the live video stream.
- Configure image acquisition object properties (Optional). Some characteristics may be changed by using '*get*' and '*set*' function. Using properties of the object, some acquisition process can be controlled, such as the amount of video data to capture.

- Acquire image data. There are some steps to process the capturing. First, start the video input object to prepare data by using '*start*' function. Next, trigger the acquisition. Triggers depend on the TriggerType property that is previously configured. If the property is set to 'immediate', the object executes the trigger and acquires image immediately after the '*start*' function is called. After that, to bring data into the MATLAB® workspace, the 'getdata' function is called. The data is brought in multiple frames and then it can be manipulated as you want.
- Clean up. Once the image capturing is finished, the image acquisition device should be removed from the memory and MATLAB® variables should be cleared using 'delete' and 'clear' functions.

When images are brought to the MATLAB® workspace, next step is to performed image processing using Image Processing Toolbox. This toolbox provides a wide range of image processing operations which are explained in the morphological operations. Morphology is a technique of image processing based on shapes. The value of each pixel in the output image is based on a comparison of the corresponding pixel in the input image with its neighbors. The toolbox has the function '*bwlabel*', which can be used to determine the areas of markers existed in the frame.

2.7.3 Camera Interface with Microsoft DirectShow

To implement the system in real time application, we chose C++ Programming with Microsoft DirectShow. Microsoft DirectShow has the capabilities in capturing multimedia data from a wide variety of input devices.

Microsoft DirectShow is a multimedia architecture that is used to interact and control streaming data from media input devices such as digital video camcorders, webcams, DVD drives, TV tuner cards and analogue video capture cards based on the Windows Driver Model (WDM) or Video for Windows (VFW). It supports many types of media formats such as Advanced Systems Format (ASF), Motion Picture Experts Group (MPEG), Audio-Video Interleaved (AVI), MPEG Audio Layer-3 (MP3), and WAV sound files. DirectShow provides media playback, format

conversion, and capture tasks for both video and audio devices. DirectShow is based on the Component Object Model (COM). It is the reusable code object that has a globally unique identifier (GUID), which is a string of hexadecimal numbers. The COM objects that are used in DirectShow applications are represented as the class ID, which is easy to understand. The class ID provides the symbolic name that is used to instantiate a COM object [21].

2.7.3.1 DirectShow concepts

In DirectShow Application Programming, there are two types of classes of objects, which are filters and filter graphs. Filters are software components that perform some operation on a multimedia stream. DirectShow filters can read files, get video from a video capture device, decode various stream formats, such as MPEG-1 video and pass data to the graphics or sound card. Filters receive input and produce output and connect to each other via pins. The connecting chain of filters to produce any application is called the filter graph. The example of the filters and filter graphs can be illustrated in the diagram as shown in Figure 2.8.

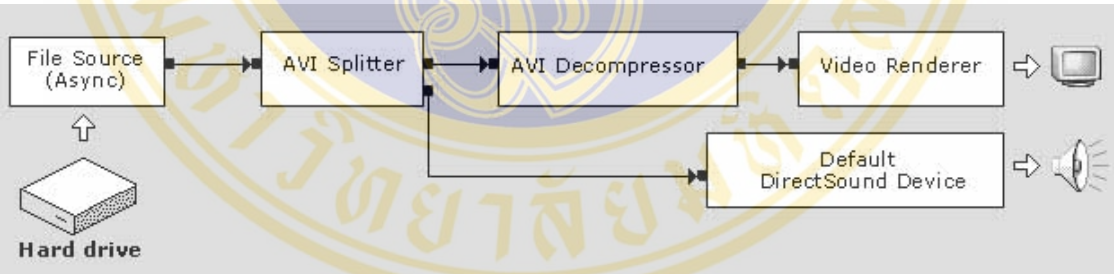


Figure 2.8 Diagram shows a filter graph for playing an AVI file [21]

Figure 2.8 shows a filter graph for playing an AVI file from the hard disk The AVI Splitter filter parses the file into a compressed video stream and an audio stream. The AVI Decompressor filter decodes the video frames. The Video Renderer filter plays the output display. The Default DirectSound Device filter plays the audio stream, using DirectSound.[21] On the other hand, this concept can explain in the programming language by thinking the filters as function calls and filter graphs as a program that called these functions like in C++ pseudo code as follows:

```
FilterGraph()
{
  SourceFilter();              // A source filter
  TransformFilter();           // Usually, a transform filter
  :                            // As many other filters as needed
  RendererFilter();            // A renderer filter
}
```

Like other programming languages, DirectShow filter graphs execute sequentially from the first filter to the last filter. However, there is a difference from common C++ program that the filter graph also executes continuously as the stream of data. The filter graph can have multiple streams that can process simultaneously such as video stream and audio stream from a webcam and a microphone, respectively.

DirectShow filters, which are the basic units of the program, comprise of three classes; source filters, transform filters and renderer filters. Source filter is the filter that produces a stream from either a file on hard disk or a multimedia device. Transform filter receives an input stream from a source filter or other filters and processes the stream and then passes it to the next filter. Transform filter can interpret a data stream, encode and decode, for example. A renderer filter translates a stream to a form of output such as a written file on the hard disk or a display window on the desktop. The flow of data can be controlled by a high-level component called the Filter Graph Manager such as calling “Run” (to move data through the graph) or "Stop" (to stop the flow of data). Moreover, the filter can be accessed directly through COM interfaces to control the stream operations. The Filter Graph Manager also passes event notifications to the application. It also provides methods to connect the filters to build the filter graph.

2.7.3.2 Writing a DirectShow Application

There are 3 basic steps to perform DirectShow application as shown in the following diagram:

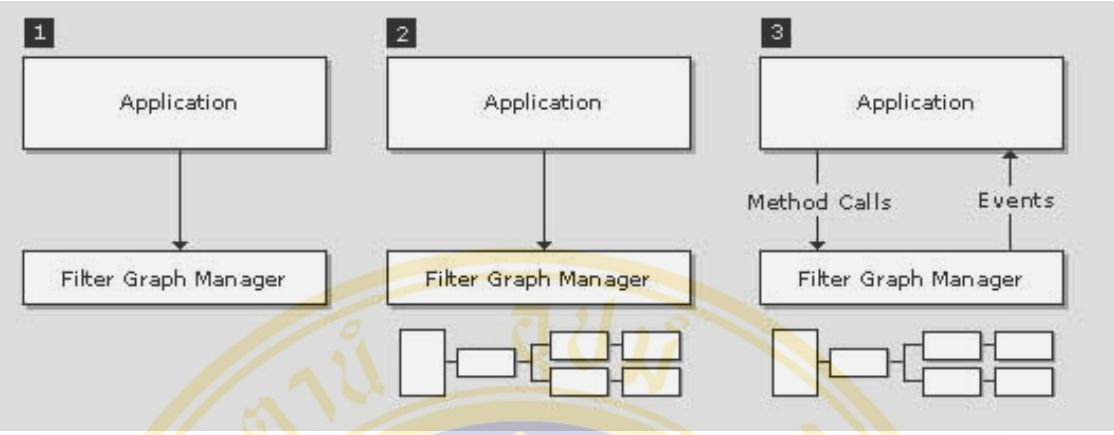


Figure 2.9 Basic steps for writing DirectShow application [21]

1. Create the Filter Graph Manager by calling function *CoCreateInstance*.
2. Uses the Filter Graph Manager to build a filter graph.
3. Use the Filter Graph Manager to control the filter graph and stream data through the filters.

2.7.3.3 Video Capture in DirectShow

Video capture means video that is received from hardware devices such as cameras, TV tuner cards, and video tape recorders (VTRs). The captured video can be saved to file or previewed live.

A filter graph that performs video or audio capture is called a capture graph. DirectShow provides a helper object to build capture graphs called the Capture Graph Builder by creating the *ICaptureGraphBuilder2* interface, which contains methods for building and controlling a capture graph.

To create the new instances of the Capture Graph Builder and Filter Graph Manager, *CoCreateInstance* is called. Then the Capture Graph Builder is initialized by calling *ICaptureGraphBuilder2::SetFiltergraph* with a pointer to the Filter Graph Manager's *IGraphBuilder* interface. The diagram of the process is shown in Figure 2.10.

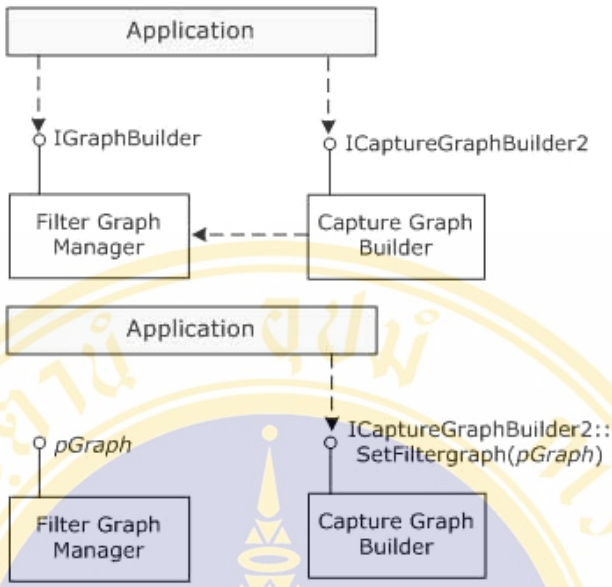


Figure 2.10 Capture Graph Builder Diagram [21]

There are some points that capture filters are different to other DirectShow filters. For example, the capture filter has two or more output pins that deliver the same kind of data such as a preview pin and a capture pin. The preview pin is used to render video to the screen, while the capture pin is used to write video to a file. However, the Capture Graph Builder automatically locates the correct pin to ICaptureGraphBuilder2 methods to avoid mistakes from query pins directly.

2.7.3.4 Selecting Capture Device

To select the capture device, system device enumerator is called by specifying the name of the device for instance, Audio capture, Video capture or Video compression. The procedures are as follows: [See Figure 2.11]

1. Create System device enumerator by calling *CoCreateInstance* through Class named *CLSID_SystemDeviceEnum*.
2. Specify the type of device by calling *ICreateDevEnum:: CreateClassEnumerator* with device's CLSID
3. Use *IEnumMoniker::Next* to identify name of each Moniker that is a COM object that collect information from other objects.
4. Get the property interface from the moniker by calling *IMoniker::*

BindToStorage

5. Create DirectShow Filter by calling *IMoniker::BindToObject* and call *IfilterGraph::AddFilter* to add the actual filter

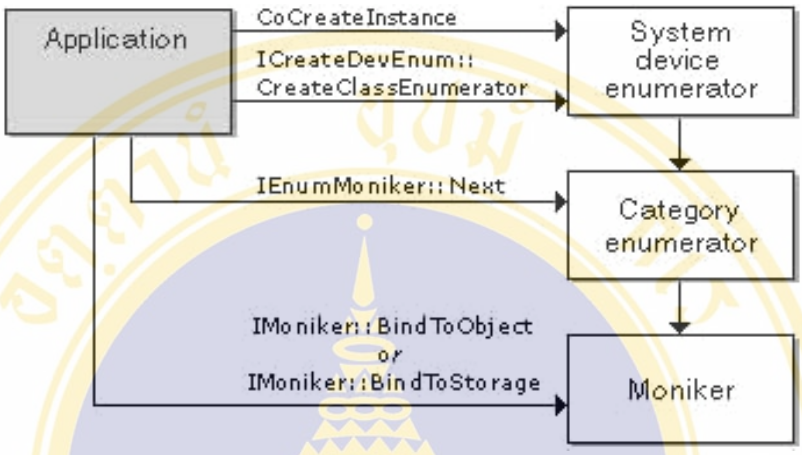


Figure 2.11 System device enumerator diagram [21]

2.7.3.5 Previewing Video

To create Video Preview Graph, *ICaptureGraphBuilder2::RenderStream* is called as shown in source code below:

```
// Capture Graph Builder
// Initialize pBuild
ICaptureGraphBuilder2 *pBuild;
// Video capture filter.
// Initialize pCap and add it to the Filter Graph
IBaseFilter *pCap;
hr = pBuild->RenderStream(  &PIN_CATEGORY_PREVIEW,
                            &MEDIATYPE_Video, pCap,
                            NULL, NULL    );
```

The first parameter of *RenderStream* function specifies which output pin of the source filter to connect. In this case is the *PIN_CATEGORY_PREVIEW*. Next is the parameter that specifies the media type. Then, the third parameter is the pointer to Capture Filter's *IBaseFilter* Interface. After that is the pointer to

compression filter's *IBaseFilter* Interface. And the last parameter is the pointer to renderer of the *IBaseFilter* Interface.

2.7.4 Image Processing Using OpenCV

As MATLAB® implementation, after acquiring images from cameras to the computer, the next step is the image processing part. We select OpenCV Library to perform this computation. OpenCV derives from Open Source Computer Vision Library used for real time computer vision, which includes in the Intel Performance Libraries as well as IPL (Intel Image Processing Library) and IPP (Intel Performance Primitives). They are C++ libraries that offer algorithms for the development of scientific and mathematical applications as expressed in Table 2.2 [22].

Table 2.2 OpenCV capabilities

Operations	Contents
Image functions	Creation, allocation, destruction of images. Fast pixel access macros.
Data Structures	Static types and dynamic storage.
Contour Processing	Finding, displaying, manipulation, and simplification of image contours.
Geometry	Line and ellipse fitting. Convex hull. Contour analysis.
Image Statistics	In region of interest: Count, Mean, STD, Min, Max, Norm, Moments, Hu Moments.
Morphology	Erode, dilate, open, close. Gradient, top-hat, black-hat.
Thresholding	Binary, inverse binary, truncated, to zero, to zero inverse.
Flood Fill	4 and 8 connected
Matrix	Matrix Math: SVD, inverse, cross-product, Mahalanobis, eigen values and vectors. Perspective projection.
Drawing Primitives	Line, rectangle, circle, ellipse, polygon. Text on images.

There are some common patterns for naming the OpenCV functions. Function names usually form in: cv<ActionName><Object>[<Modifiers>], e.g. *cvCalibrateCamera*, *cvCreateImage*. Sometimes function is called following its algorithm, e.g. *cvSobel*, *cvCanny*. OpenCV has some fundamental data types that are

preferred to use as the argument types used inside function for example, IplImage for raster images, *CvMat* for matrices, *CvSeq* for contours.

To execute the OpenCV program, the frames of images must be brought to the OpenCV environment by encapsulating into IplImage structure. Then the OpenCV functions can be performed. The basic functions of OpenCV that are used in the algorithms for image processing are such as *cvCvtColor*, *cvThreshold*, and *cvFloodFill*.

cvCvtColor is used to converts input image from one color space to another for example, RGB to Gray, RGB to YCC, RGB to HSV. In our algorithm, we used it for convert RGB images to grayscale images. Then *cvThreshold* is used to apply fixed-level threshold to get bi-level image from grayscale image. After that, *cvFloodFill* is applied for finding connected components of marker areas. Subsequently, the centroids of marker areas are determined. Moreover, we used the capabilities of OpenCV to manage the calculation of matrices for the stereo triangulation of marker positions.

2.8 Visualization system

2.8.1 OpenGL

All marker positions can be illustrated in graphical view in virtual reality using OpenGL C++ library. OpenGL is short for “Open Graphics Library”, which is a software interface for graphics hardware [23]. OpenGL consists of a set of commands that provide the graphics operations to render an image in three-dimensional scene on screen. An image is constructed shape from a set of geometric primitives - points, lines, polygons, images, and bitmaps. These primitives together with commands can be produced high-quality graphical images in two or three dimensions. OpenGL allows user for appropriate drawing with translation, rotation, shear and scaling commands to select the viewing scene on screen. Furthermore, there are coloring and lighting commands to add illusion of solidity and realism to the scene with different type of identified views.

OpenGL is a hardware- and system-independent interface. An OpenGL application can work on any platform that have it installed. Because it is system

independent, there are no functions to create windows, perform windowing tasks, or obtain user-input etc., but there are helper functions for each platform. OpenGL contains two basic libraries, GL and GLU. Some basic geometric primitives such as points, lines, and polygons are implemented in GL. The OpenGL Utility Library (GLU) provides more complex features, such as quadric rectangular surfaces and NURBS curves and surfaces. However, we do not use GLU directly in this implementation but employ a very useful library, GLUT (OpenGL Utility Toolkit), which includes GLU in the library. GLUT helps to create windows, handle the messages and support creating event driven functions. It exists for several platforms, that means that a program which uses GLUT can be compiled on many platforms without (or at least with very few) changes in the code.

2.8.1.1 OpenGL Syntax

To be hardware independent, OpenGL provides its own data types. They all begin with "GL". (See Figure 2.12)

- Constants begin with ‘GL_’ and use underscore to separate words, for example, GL_COLOR_BUFFER_BIT.
- OpenGL commands begin with ‘gl’, for example, glClearColor().
- OpenGL commands may end with a number following by a character, for example, glVertex3f().
- Those commands may have a character ‘v’ at the end to identify that they are vectors, for example, glVertex3fv().
- To mention those commands, they can be called such as glVertex*(), glVertex*v()

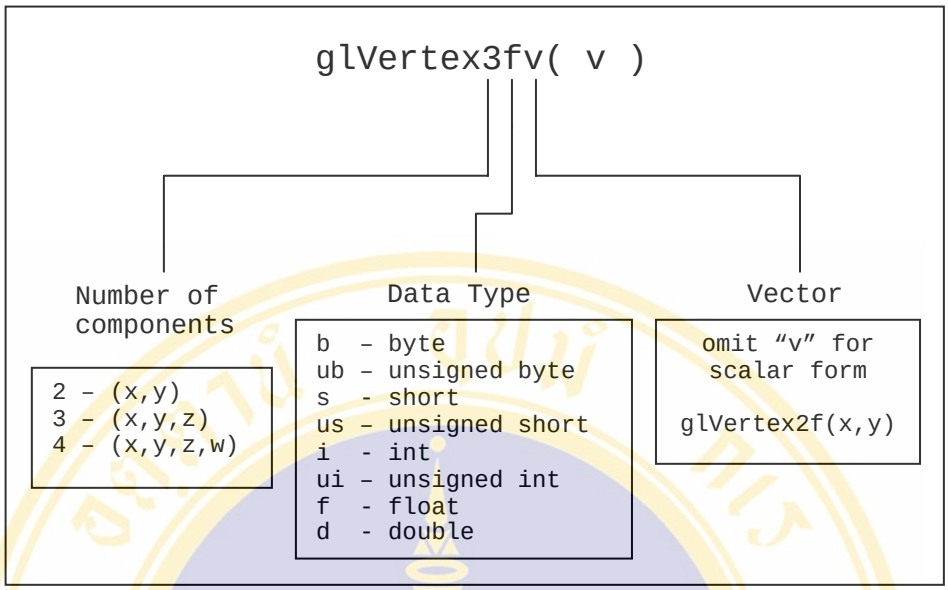


Figure 2.12 The components of an OpenGL command

2.8.1.2 Drawing primitives

OpenGL draws primitives- points, line segments, or polygons, which are described inside *glBegin()...glEnd()* pairs. For example, to draw a triangle the following statements would be used: [24]

```
glBegin(GL_TRIANGLES);
        glColor3f(...);
        glVertex3f(...);
        glVertex3f(...);
        glVertex3f(...);
glEnd();
```

The argument to '*glBegin*' describes the type of primitive, which can be as follows:

```
GL_POINTS
GL_LINES
GL_LINE_STRIP
GL_LINE_LOOP
GL_TRIANGLES
GL_TRIANGLE_STRIP
GL_TRIANGLE_FAN
GL_QUADS
GL_QUAD_STRIP
GL_POLYGON
```

The *'glEnd()'* indicates when to stop drawing that type of primitive. Moreover, color can be specified for each vertex that is followed the command *'glColor()'*. Besides, drawing 3D geometry can perform using GLUT, for example drawing cube, sphere, torus, cone, and teapot. These objects are also defined in solid or wire shape. The followings are the example of drawing 3D object commands.

- `void glutWireCube(GLdouble size);`
- `void glutSolidCube(GLdouble size);`
- `void glutWireSphere(GLdouble radius, GLint slices, GLint stacks);`
- `void glutSolidSphere(GLdouble radius, GLint slices, GLint stacks);`

2.8.1.3 Viewing Systems

The viewing systems of the geometric model from real world 3D coordinates to the 2D window coordinates have the transformation as shown in Figure 2.13.

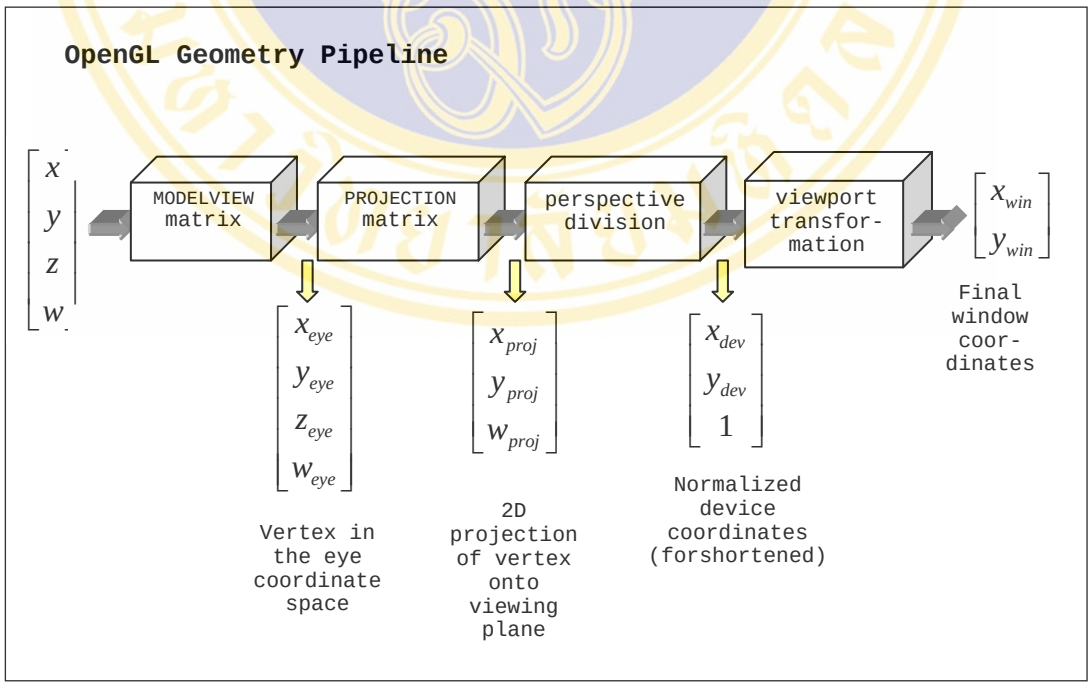


Figure 2.13 Stages of vertex transformation [25]

First, the original world coordinates are transformed to eye coordinates by the ModelView matrix, whose coordinates are relative to the viewer eye or camera. Next, the eye coordinates are transformed by the Projection matrix to produce clip coordinates that define the field of view. There is a defined clip coordinate W value that OpenGL clips all coordinates outside this range. Then the normalized device coordinates are produced from dividing the clip coordinates X, Y, and Z by clip coordinates W. The normalized device coordinates range from -1 to 1 in all three axes. The last transformation is the Viewport transformation, which maps the normalized device coordinates by scaling and translating to produce 2D window coordinates.

To create a view of scene, a description of the scene geometry and a camera or view definition are needed. OpenGL camera is initially located at the origin looking down the negative Z-Axis. The camera definition allows projection of the 3D scene geometry onto a 2D surface for display. The projection could be orthographic projection, perspective projection, or skewed orthographic projection. The following source code [26] shows an example how to define the transformation to create a view.

```
void mainReshape(int w, int h){

  // VIEWPORT TRANSFORMATION
  glViewport(0, 0, w, h);
  // PROJECTION TRANSFORMATION
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glFrustum(wnLEFT, wnRIGHT, wnBOT, wnTOP, wnNEAR, wnFAR);
  // MODELVIEW TRANSFORMATION
  glMatrixMode(GL_MODELVIEW);

  ....
```

The function glViewport(x, y, width, height) specifies the Viewport transformation, where x, y are the coordinates of the lower left of the window viewing rectangle, and width, height are the dimension of the viewport. They are all given in pixels. Then glMatrixmode() is called to select current matrix and specify projection

transformation. The glLoadIdentity() is called to set unity matrix before any matrix transformation. Then glFrustum(left, right, bottom, top, near, far) declares the 3D perspective, which specifies the clipping planes at positions left, right, bottom, top, near, and far. These parameters are specified in eye coordinates and their magnitudes determine the volume of space that is mapped to the viewport. Finally, the current matrix is changed for the model view matrix from calling glMatrixMode with GL_MODELVIEW parameter.

Because OpenGL has a capability to create view in 3D, it can be assembled to our program by passing the calculated 3D positions and orientations to output the program in virtual reality with different views. However, the OpenGL out put frame locates as Figure 2.14a and the frame of calculated positions is orientated as in Figure 2.14b, which they are in different frames. Therefore, there must be some transformations to relate information in each frame together. This can be explained in the next issue.

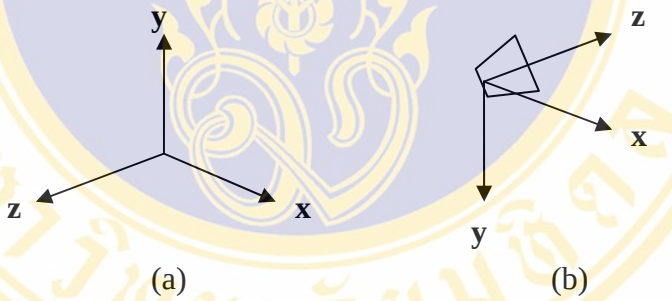


Figure 2.14 (a) OpenGL coordinate frame (b) Camera coordinate frame

2.8.2 Spatial Descriptions and Transformations [27]

To describe the position and orientation of an object in space, it must be rigidly attached to a coordinate system, or frame. The coordinate system is described its own position and orientation relative to some reference coordinate systems as shown in Figure 2.15.

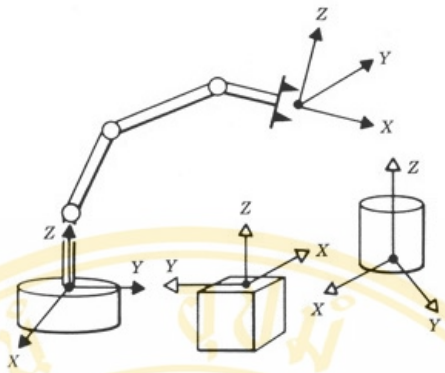


Figure 2.15 Coordinate systems or frames are attached to the manipulator and to objects in the environment.

Any frame can perform as a reference frame by describing the position and orientation through the transformation or change of the description of the attributes from one frame to another.

To describe the position of any point in a coordinate system, a 3x1 position vector is represented the location in the frame, where shown below.

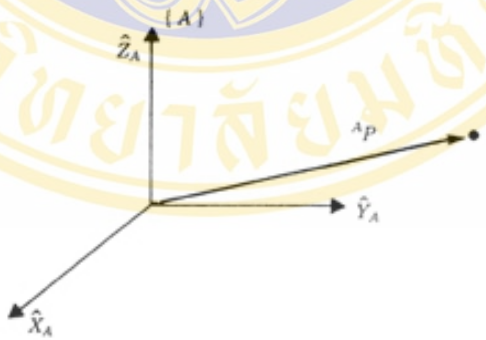


Figure 2.16 Position vector relative to frame [27]

$$^{A}\mathbf{P} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \quad (2.7)$$

where $^{A}\mathbf{P}$ is the point in the coordinate system {**A**}

p_x is the displacement in X direction

p_y is the displacement in Y direction

p_z is the displacement in Z direction .

Moreover, the description of the orientation is necessary to represent a point in space. In order to perform the orientation, a coordinate system is attached to the object and then defined a description of this coordinate system relative to the reference system [27]. The example is shown in Figure 2.17, where coordinate system {**B**} is attached to the object in a known pose. A description of {**B**} relative to {**A**} is determined to identify the orientation of the object from a rotation matrix as follows [27].

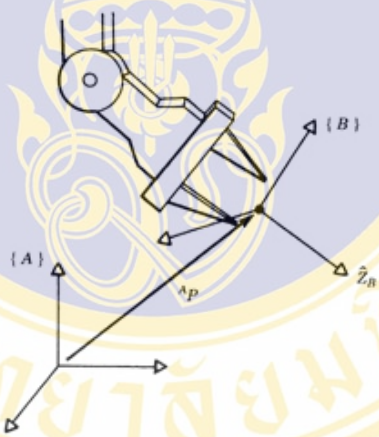


Figure 2.17 Orientation of the object relative to frame [27]

$$ {}^{A}_{B}\mathbf{R} = \left[{}^{A}_{B}\vec{\mathbf{x}} \quad {}^{A}_{B}\vec{\mathbf{y}} \quad {}^{A}_{B}\vec{\mathbf{z}} \right] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.8} $$

where $^{A}_{B}\mathbf{R}$ is Rotation matrix of frame {**B**} relative to frame {**A**}

r_{11}, r_{12}, r_{13}, ….., r_{33} are the scalar components that computed from the projection of the vectors onto the unit directions of its reference frame.

Besides, there are additional representations to describe the orientation of a frame {**B**}, for example, X-Y-Z fixed angles, Z-Y-X Euler angles, Z-X-Z Euler angles.

- X-Y-Z fixed angles (See Figure 2.18)

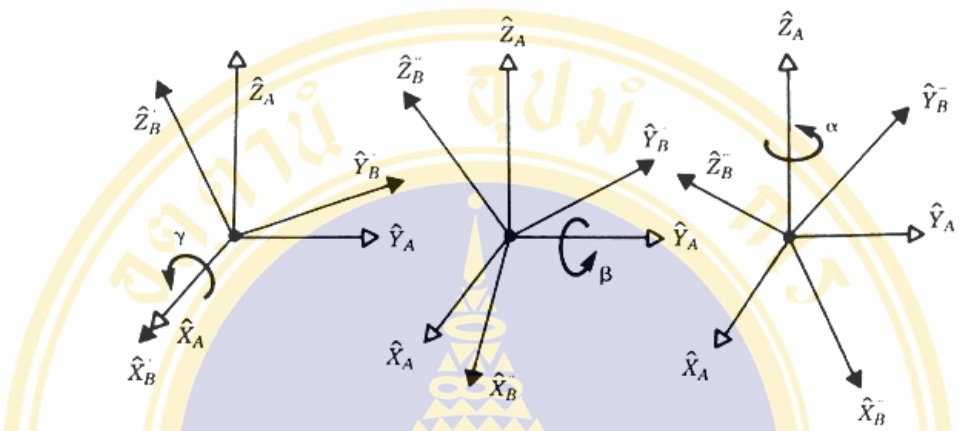


Figure 2.18 X-Y-Z fixed angles, Rotations are performed in the order $\mathbf{R}_X(\gamma), \mathbf{R}_Y(\beta), \mathbf{R}_Z(\alpha)$ [27]

The orientation starts with the frame {**B**} coincident with a known reference frame {**A**}. Rotate {**B**} first about $\vec{\mathbf{X}}_\mathbf{A}$ by an angle γ, then about $\vec{\mathbf{Y}}_\mathbf{A}$ by an angle β, and, finally, about $\vec{\mathbf{Z}}_\mathbf{A}$ by an angle α. The equivalent rotation matrix can be expressed as follows,

$$
\begin{aligned}
{}^{\mathbf{A}}_{\mathbf{B}}\mathbf{R}_{\mathbf{XYZ}}(\gamma,\beta,\alpha) &= \mathbf{R}_\mathbf{Z}(\alpha)\mathbf{R}_\mathbf{Y}(\beta)\mathbf{R}_\mathbf{X}(\gamma) \\
&= \begin{bmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & 0 & c_\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\gamma & -s_\gamma \\ 0 & s_\gamma & c_\gamma \end{bmatrix} \qquad (2.9)
\end{aligned}
$$

$$
{}^{\mathbf{A}}_{\mathbf{B}}\mathbf{R}_{\mathbf{XYZ}}(\gamma,\beta,\alpha) = \begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{bmatrix} \qquad (2.10)
$$

However, the definition given in equation (2.11) is correct for rotations in the order: about $\vec{\mathbf{X}}_{\mathbf{A}}$ by γ, about $\vec{\mathbf{Y}}_{\mathbf{A}}$ by β, and about $\vec{\mathbf{Z}}_{\mathbf{A}}$ by α only.

The inverse problem of extracting the equivalent X-Y-Z fixed angles from a rotation matrix can be performed by solving a set of transcendental equations between equation (2.8) and (2.10) as the following.

$$ {}^{\mathbf{A}}_{\mathbf{B}}\mathbf{R}_{\mathbf{XYZ}}(\gamma,\beta,\alpha) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.11} $$

$$ \begin{aligned} \beta &= \text{atan2}\left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right), \\ \alpha &= \text{atan2}\left(r_{21}/c_\beta, r_{11}/c_\beta\right), \\ \gamma &= \text{atan2}\left(r_{32}/c_\beta, r_{33}/c_\beta\right), \end{aligned} \tag{2.12} $$

where $\text{atan2}(y,x)$ is a two-argument arc tangent function.

- Z-Y-Z Euler angles (See Figure 2.19)

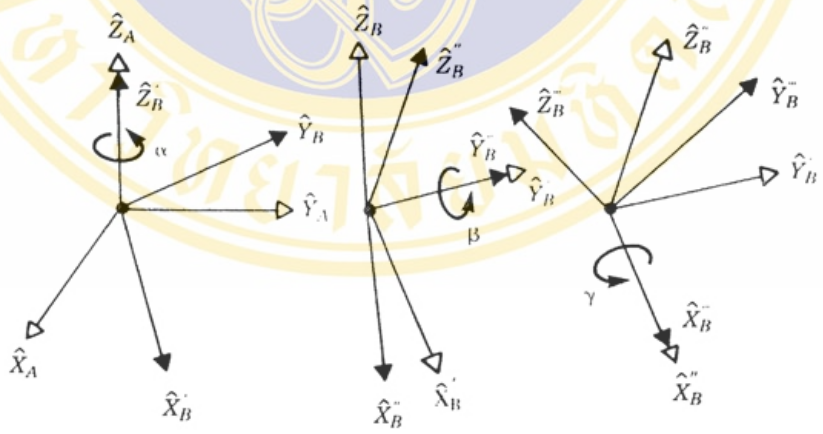


Figure 2.19 Z-Y-X Euler angles [27].

The orientation starts with the frame {**B**} coincident with a known reference frame {**A**}. Rotate {**B**} first about $\vec{\mathbf{Z}}_{\mathbf{B}}$ by an angle α, then about $\vec{\mathbf{Y}}'_{\mathbf{B}}$ by

an angle β, and, finally, about $\vec{\mathbf{X}}''_{\mathbf{B}}$ by an angle γ. The equivalent rotation matrix can be expressed as follows,

$$
\begin{aligned}
{}^{\mathbf{A}}_{\mathbf{B}}\mathbf{R}_{\mathbf{Z'Y'X'}}(\alpha,\beta,\gamma) &= \mathbf{R}_{\mathbf{Z}}(\alpha)\mathbf{R}_{\mathbf{Y}}(\beta)\mathbf{R}_{\mathbf{X}}(\gamma) \\
&= \begin{bmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & 0 & c_\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\gamma & -s_\gamma \\ 0 & s_\gamma & c_\gamma \end{bmatrix} \qquad (2.13)
\end{aligned}
$$

$$
{}^{\mathbf{A}}_{\mathbf{B}}\mathbf{R}_{\mathbf{Z'Y'X'}}(\alpha,\beta,\gamma) = \begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{bmatrix} \qquad (2.14)
$$

Because equation (2.14) is equivalent to (2.10), the solutions of extracting Z-Y-X Euler angles from a rotation matrix are the same as (2.12).

- Z-X-Z Euler angles

Another possible description of a frame {**B**} is that the orientation starts with the frame {**B**} coincident with a known reference frame {**A**}. Rotate {**B**} first about $\vec{\mathbf{Z}}_{\mathbf{B}}$ by an angle α, then about $\vec{\mathbf{X}}'_{\mathbf{B}}$ by an angle β, and, finally, about $\vec{\mathbf{Z}}''_{\mathbf{B}}$ by an angle γ. The equivalent rotation matrix can be expressed as follows,

$$
\begin{aligned}
{}^{\mathbf{A}}_{\mathbf{B}}\mathbf{R}_{\mathbf{Z'X'Z'}}(\alpha,\beta,\gamma) &= \mathbf{R}_{\mathbf{Z}}(\alpha)\mathbf{R}_{\mathbf{X}}(\beta)\mathbf{R}_{\mathbf{Z}}(\gamma) \\
&= \begin{bmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\beta & -s_\beta \\ 0 & s_\beta & c_\beta \end{bmatrix} \begin{bmatrix} c_\gamma & -s_\gamma & 0 \\ s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (2.15)
\end{aligned}
$$

$$
{}^{\mathbf{A}}_{\mathbf{B}}\mathbf{R}_{\mathbf{Z'X'Z'}}(\alpha,\beta,\gamma) = \begin{bmatrix} -s_\alpha c_\beta s_\gamma + c_\alpha c_\gamma & -s_\alpha c_\beta c_\gamma - c_\alpha s_\gamma & s_\alpha s_\beta \\ c_\alpha c_\beta s_\gamma + s_\alpha c_\gamma & c_\alpha c_\beta c_\gamma - s_\alpha s_\gamma & -c_\alpha s_\beta \\ s_\beta s_\gamma & s_\beta c_\gamma & c_\beta \end{bmatrix} \qquad (2.16)
$$

The solution for extracting Z-X-Z Euler angles from a rotation matrix is expressed in equation (2.17).

$$\begin{aligned}\gamma &= \text{atan2}\left(r_{31},\, r_{32}\right), \\ \alpha &= \text{atan2}\left(r_{13},\, -r_{23}\right), \\ \beta &= \text{atan2}\left(r_{31}\text{s}_{\gamma} + r_{32}\text{c}_{\gamma},\, r_{33}\right)\end{aligned} \tag{2.17}$$

To complete the description of a frame, the information should have both position and orientation relative to reference frame (See Figure 2.20), which is expressed in a form of 4x4 matrix called homogeneous transform or transformation matrix; that is

$$ {}^{\mathbf{A}}_{\mathbf{B}}\mathbf{T} = \left[\begin{array}{ccc:c} & {}^{\mathbf{A}}_{\mathbf{B}}\mathbf{R} & & {}^{\mathbf{A}}\mathbf{P}_{\mathbf{BORG}} \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.18}$$

where ${}^{\mathbf{A}}_{\mathbf{B}}\mathbf{T}$ is homogeneous transform of Frame {**B**} relative to frame {**A**}

${}^{\mathbf{A}}_{\mathbf{B}}\mathbf{R}$ is the rotation matrix of frame {**B**} relative to frame {**A**}

${}^{\mathbf{A}}\mathbf{P}_{\mathbf{BORG}}$ is the position vector that locates the origin of frame {**B**} relative to frame {**A**}.

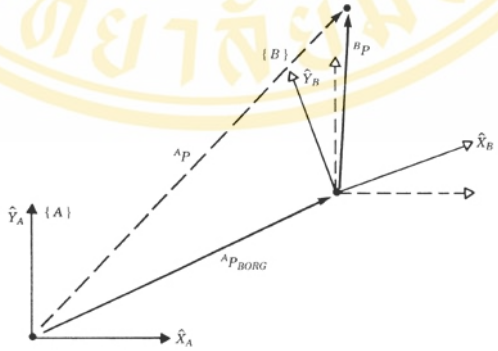


Figure 2.20 Frame {B} rotated and translated [27]

From above description, we determine the homogeneous transform of frame {**B**} relative to frame {**A**}. On the other hand, we can determine the homogeneous

transform of frame {**A**} relative to frame {**B**} from computing the inverse of homogeneous transform as the following.

$$ {}^{B}_{A}\mathbf{T} = {}^{A}_{B}\mathbf{T}^{-1} \tag{2.19} $$

$$ {}^{A}_{B}\mathbf{T}^{-1} = \left[\begin{array}{ccc|c} & {}^{A}_{B}\mathbf{R}^{T} & & -{}^{A}_{B}\mathbf{R}^{T}\,{}^{A}\mathbf{P}_{BORG} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.20} $$

2.9 Summary

In this chapter, multiple sources of information needed for the implementation of the system are explained and discussed. They compose of the camera system, markers, software and principles that are involved in interface, image processing and 3D visualization.

CHAPTER III
METHODOLOGY

3.1 System Overview

The system is setup as a prototype for tracking 3D coordinates of markers from stereo camera system. Our 3D coordinate acquisition system consists of a pair of digital cameras from Unibrain Fire-i ™ connected to the computer via FireWire (IEEE 1394) interface. The cameras are positioned precisely 20 cm apart on similar image plane (Figure 3.1) and they are arranged in a working area of 1m x 1m x 1m. To filter out the visible light, each camera is equipped with IR pass filters over the camera lenses.

Figure 3.1 Stereo camera setup

At first, we setup the system to test the algorithms using MATLAB® as shown in Figure 3.2 with both passive markers and active markers. However, the processing time of image capturing and visualizing in MATLAB® is not fast enough. Therefore, Microsoft Visual C++ is presented instead of MATLAB® to provide fast calculation with real-time applications. Moreover, we employ 2 sets of active markers to represent the relationship of two tools. Other procedures are the same as the procedures in MATLAB® as shown in Figure 3.3.

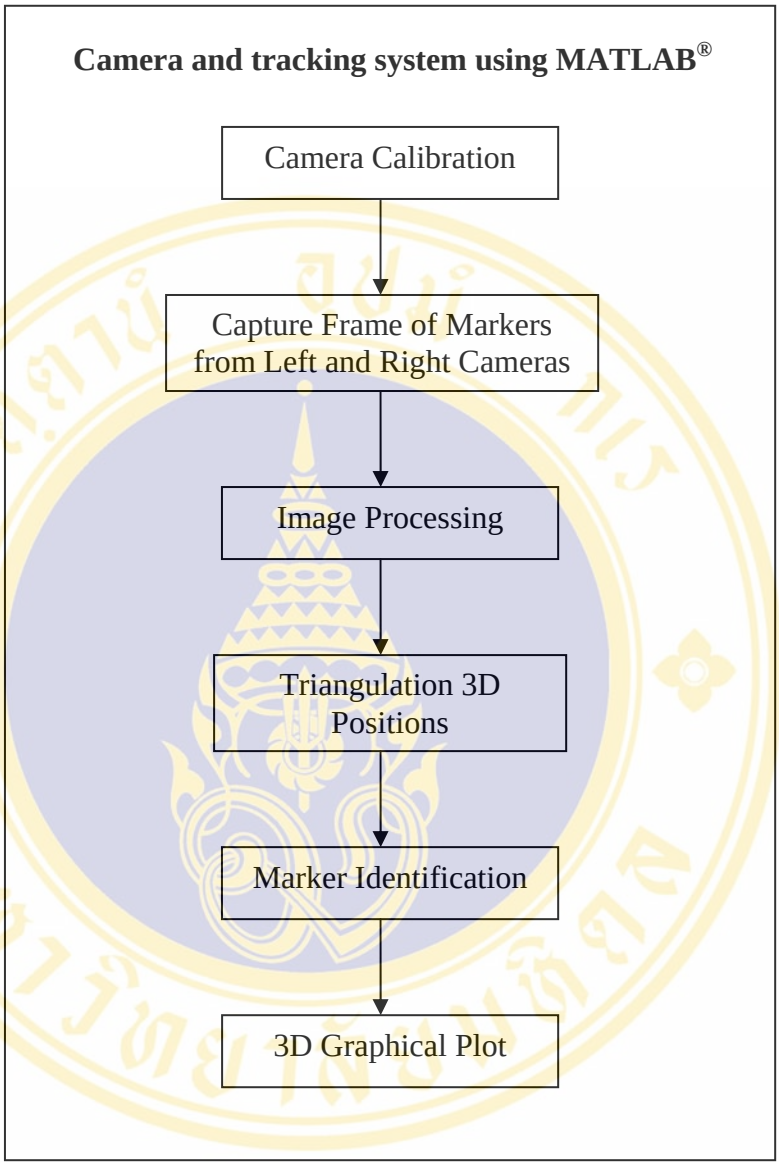


Figure 3.2 Diagrammatic representation of research system using MATLAB®

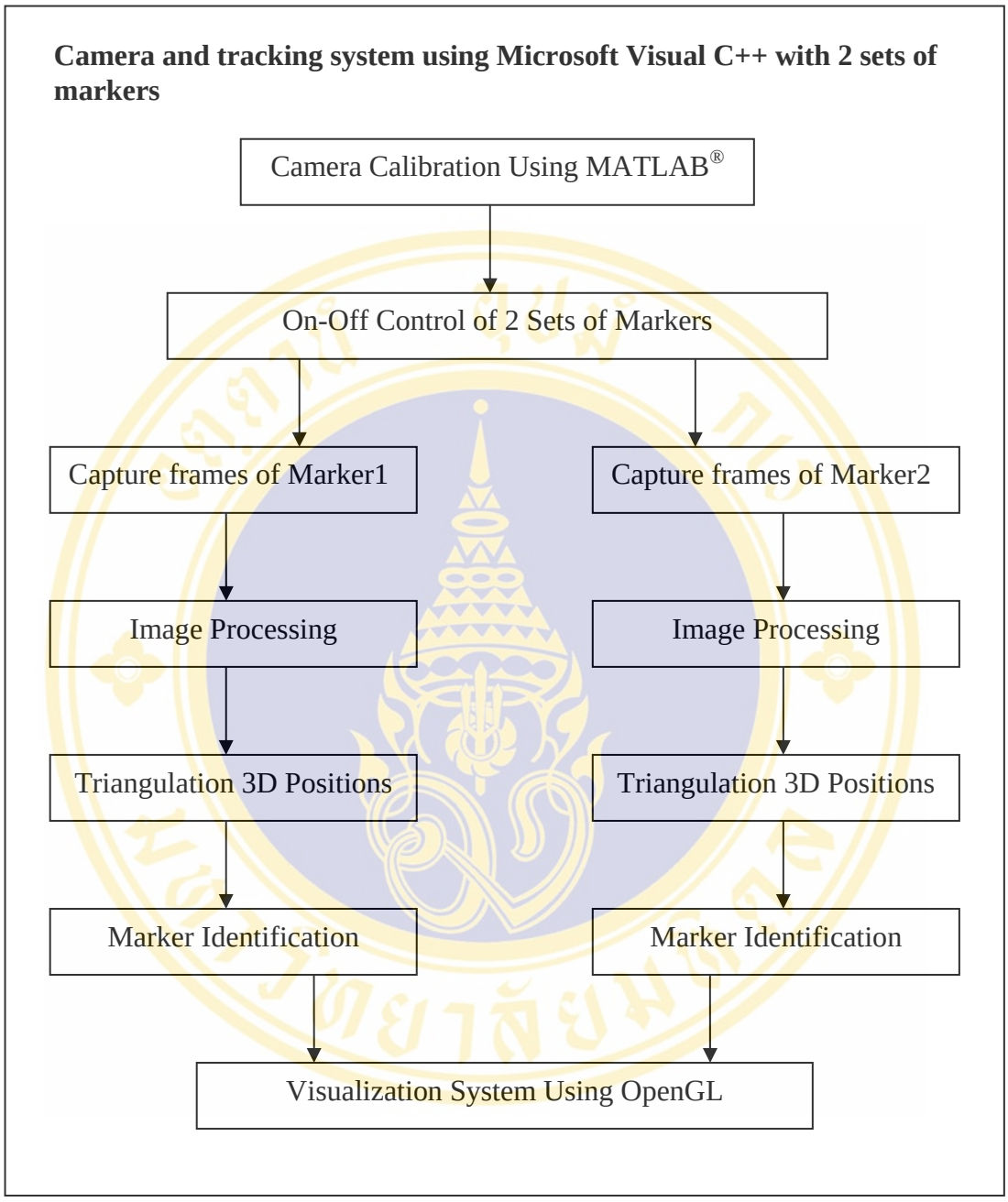


Figure 3.3 Diagrammatic representation of research system using Microsoft Visual C++

From the diagram in Figure 3.2 and Figure 3.3, mostly, both approaches share the same algorithms except the control of 2 sets of markers that is added on the implementation in Microsoft Visual C++**.** To start the system of both approaches, stereo cameras must be calibrated to find camera intrinsic and extrinsic parameters in

order to suppress the deformation of imaging sensor and distortion of lenses to the minimum level. The extracted 2D positions from both cameras from the image processing can be used to triangulate for a 3D position. Then they are arranged by the marker identification to specify the position and orientation of each marker and will be illustrated in a graphical view.

For the following, the author will describe the processes of the system using MATLAB® first, and then explain the system implemented in Microsoft Visual C++.

3.2 System Using MATLAB®

There are two formats to test the tracking system. The first format is to use infrared-light emitting diodes (IR-LEDs) as active markers (Figure 3.4: Left). The other format is to use passive markers (Figure 3.4: Right), whose tips are coated with retroreflective material. The latter format requires infrared light sources positioned at the camera and pointed in the same direction as the cameras.

Figure 3.4 Left: active markers and Right: passive markers

3.2.1 Calibration Procedure

The camera calibration is performed using the Camera Calibration Toolbox for MATLAB® developed by Jean-Yves Bouguet, MRL - Intel Corp [28]. It is mainly based on Zhang [29], Heikkila and SilvCn [5], and Tsai camera calibration method [4], which is used to fit the camera model parameters to the captured images. The

calibration process requires a calibration target. A checkerboard pattern, with the size of 20cm x 20cm containing 10 x 10 squares, is employed for this purpose. The target is placed in front of the cameras in various positions/orientations, and the images are captured (Figure 3.5). The corners of the checkerboard pattern are determined. The positions of corners are known from relative coordinate system of the target. The process of corner detections uses edge detection, straight line fitting to detect linked edges and intersecting the lines to obtain the image corners. After that image corners are matched to 3D target checkerboard corners by counting if number of squares is all visible in the image. The number of squares is predefined for comparing. As a result, the coordinate pairs of images frame and world frame are found which can be employed for calculation to find the intrinsic parameters and also extrinsic parameters [30]. The cameras are calibrated once before the tracking process.

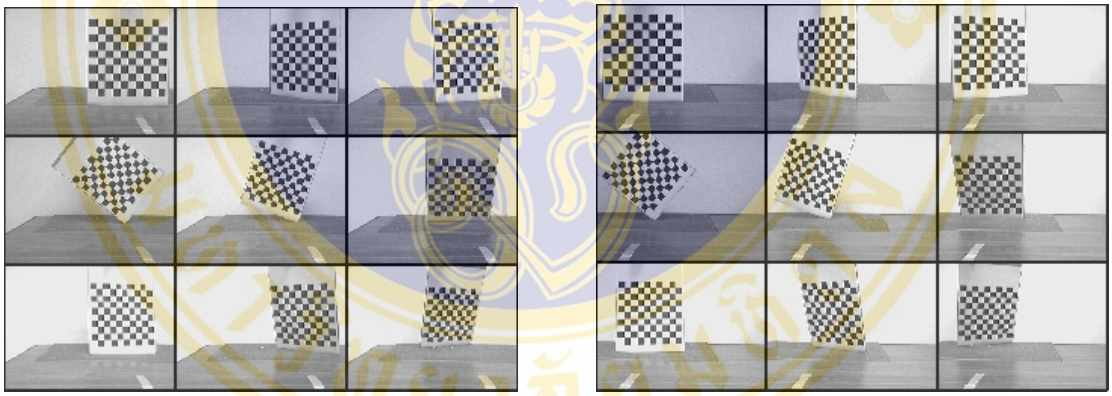


Figure 3.5 Examples of the target images for calibration from left and right camera.

3.2.2 Capturing Image Frames

With the convenience of Image Acquisition Toolbox in MATLAB®, capturing images from stereo camera uses only a few sets of commands as the following.

```
vid1 = videoinput('winvideo', 1,'UYVY_640x480');
vid2 = videoinput('winvideo', 2,'UYVY_640x480');
pl   = getsnapshot(vid1);
pr   = getsnapshot(vid2);
```

The command '*videoinput*' is used to create the video input devices with adaptor name called '*winvideo*' on device ID '*1*' (left camera) and '*2*' (right camera) using video format '*UYVY_640x480*'. Then the image frames are captured into the image parameters pl and pr respectively.

3.2.3 Image Processing

In this procedure, captured images are processed to obtain 2D positions of centroids of markers. The steps are as follows. First, images from both left and right cameras are defined the threshold to make binary images for eliminating the background. The white areas of the black and white images are labeled the connected components to calculate marker centroids. According to our markers that composed of three spheres, the centroids of markers that we are determined are a set of three pixel coordinates.

3.2.4 Triangulation 3D Positions

The centroids of markers from left and right cameras are brought to the function '*stereo_triangulation*' in Camera Calibration Toolbox as shown below, which also takes both extrinsic and intrinsic calibration parameters as inputs for correcting deformity from the imaging hardware.

```
[pos_L, pos_R] = stereo_triangulation( centroidL, centroidR, om, T,
                    fc_left, cc_left, kc_left, alpha_c_left,
                    fc_right, cc_right, kc_right, alpha_c_right);
```

where pos_L is the calculated 3D positions of the markers relative to left camera

pos_R is the calculated 3D positions of the markers relative to right camera

centroidL is the extracted marker centroids in image of left camera

centroidR is the extracted marker centroids in image of right camera

om is the rotation matrix that identifies the orientation of 2 cameras obtained from the calibration

T is the translational matrix that identifies the position of 2 cameras obtained from the calibration

fc_left, cc_left, kc_left, alpha_c_left are the intrinsic parameters of the left camera obtained from the calibration

fc_ right, cc_ right, kc_ right, alpha_c_ right are the intrinsic parameters of the right camera obtained from the calibration.

This function computes the 3D coordinates of the markers relative to the position of the camera. However, with no marker identification, the centroids from left and right cameras can be paired in any combination. This situation can lead to incorrect matching of centroids pairs and incorrect 3D positions. Therefore, marker identification for tracking algorithms is proposed to solve this matter, which has details in the next issue.

3.2.5 Marker Identification

The proposed method for tracking markers in this system is to match all possible combination pairs of marker centroids between left camera and right camera. The maximum number of the arrangements will be $n!$ where n = number of balls on a marker. Then, the distances of all possible coordinate pairs are calculated and compared with the predefined distances of the markers. The correct pairs should give the closed value to the real distance within a defined clearance. Moreover, the candidate set is evaluated with a distance from a cyclic order path as shown in Figure 3.6. As a result, for each capture, the system can determine the right pair of balls on a marker that exist in the frame without using information from the previous frame, and can give the accurate 3D position and orientation of the markers. The valid pairs can be reconstructed in computer as a 3D plot. Consequently, a real world coordinate can be precisely illustrated in a computerized virtual environment.

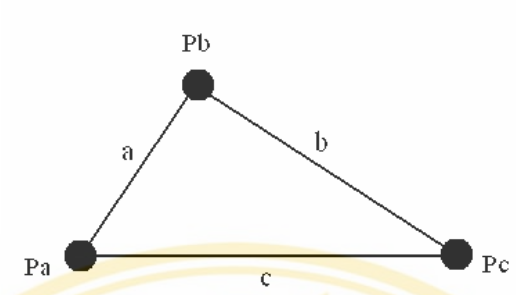


Figure 3.6 The distance between each marker in a cyclic path form

3.3 System Using Microsoft Visual C++

Visual C++ is the programming language that provides the Object-Oriented Programming (OOP) concept. The OOP concept gathers the details of problem and interprets to a new data type. Data types are included together to form an object. The C++ programming uses class to declare the details of the object, which consists of data members and member functions. Therefore, to manage the solution of the system, classes are created to support the different components of the system as shown in Figure 3.7, which is analogous to Figure 3.3.

3.3.1 Calibration procedure

The calibration procedure in this system uses the information from Camera Calibration Toolbox for MATLAB®, which is the same as the system using MATLAB®. Because the calibration process is done before running the program and the computed parameters are later used as the constants. Therefore, the system with the Visual C++ can used the parameters from MATLAB® by filling the parameters in the program.

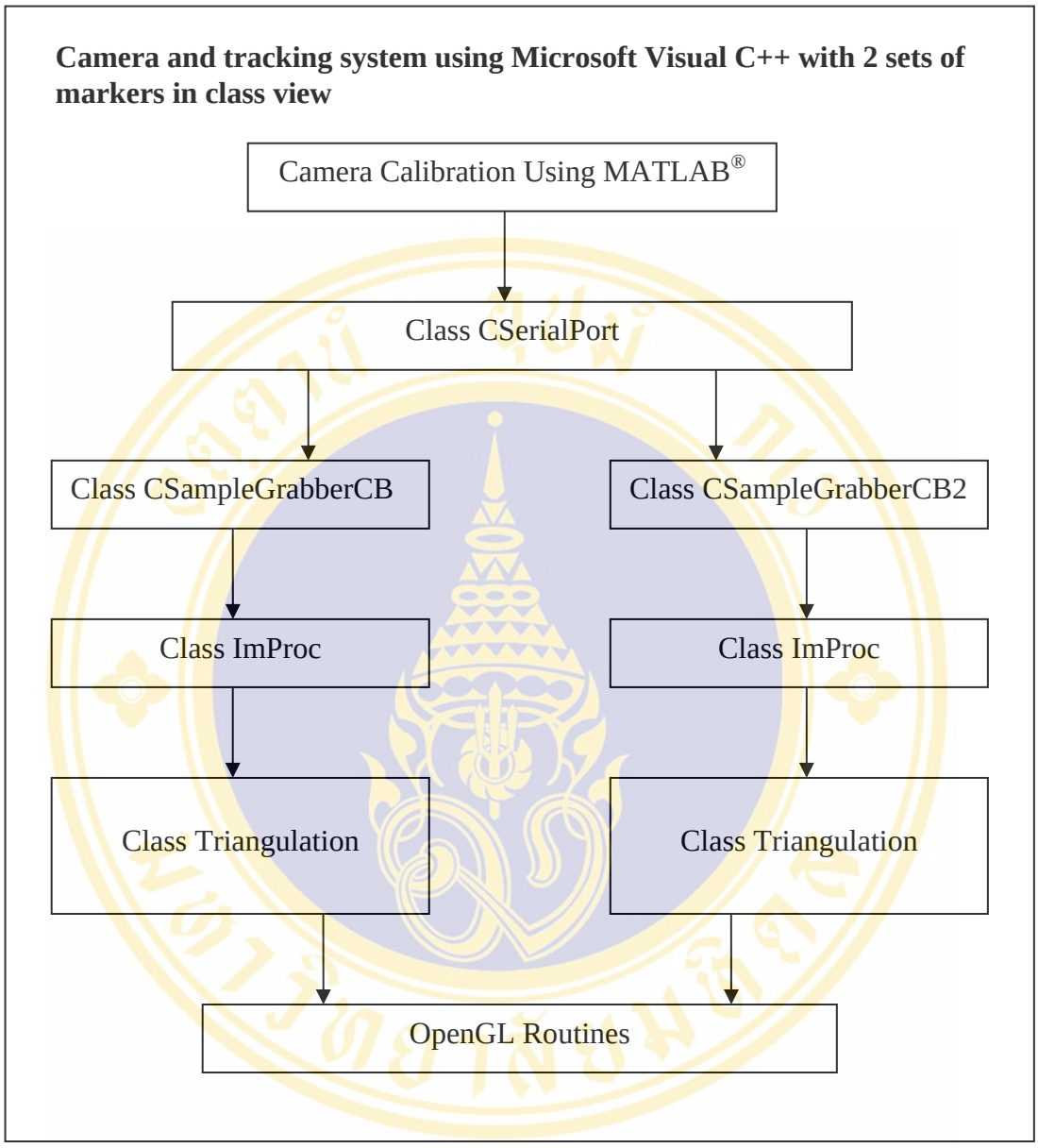


Figure 3.7 Diagrammatic representation of the system using Microsoft Visual C++ in class view

3.3.2 On -Off Control of Markers

Two sets of active markers are used to represent the relationship between 2 devices that identify the positions and orientations to each other. For this purpose, we choose the active markers that can be controlled the state by turning on/off the power via serial port control. Moreover, we selected the wireless communication to interface between the transmitter from serial port and the receiver at the active markers, which

leads to the main benefit for the mobility of markers. The electronic circuits of the interface system are built in house[1]. The markers that are used in the implementation are shown in Figure 3.8.

To implement the control of wireless markers via serial port, we use class **CSerialPort** [31] to support the communication. The class **CSerialPort** consists of member functions as the following.

```
BOOL Open(int nPort,int nBaud);  // Open serial port with
                                 // baudrate nBaud
BOOL IsOpened(void)              // Check whether the port is
                                 // open or not
int SendData(const char *,int);  // Send control data out
BOOL Close(void);                // Close serial port
```

The function '*Open*' is called to start the serial port with baud rate *nBaud* bit/second and return Boolean data type. If the serial port is opened, the return value is TRUE. Yet, if the serial port is not opened, the return value is FALSE. The function '*IsOpened*' is used to check the state of the serial port and return with TRUE (opened port) and FALSE (closed port). The command '*SendData*' is called to send the character data and the size of data in bytes to the module. When finishing the application, the function '*Close*' is called to close the port.

Although we have 2 sets of markers that switch on one after another, the processes of the calculation 3D positions are the same.

[1] The electronics circuits of the wireless active markers are designed and developed by Chatchai Neatpisarnvanit and Eakkaluck Thammacharoen (BaRT Lab, Faculty of Engineering, Mahidol University)

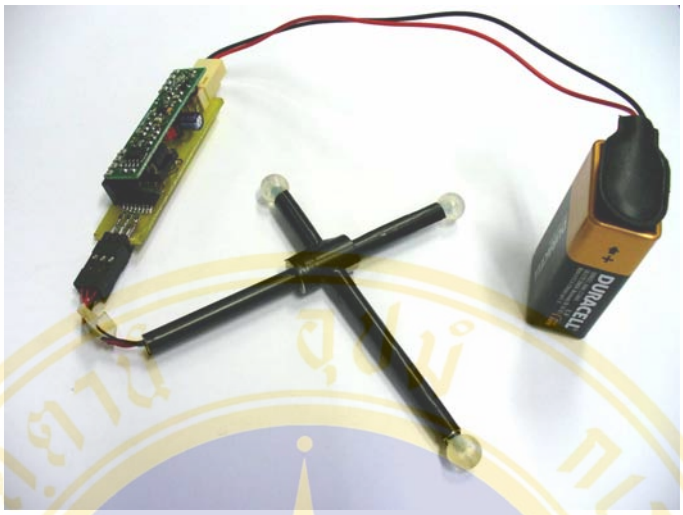


Figure 3.8 Active markers with wireless receiver

3.3.3 Capturing Image Frames

To capture image frames in Visual C++, Microsoft DirectShow is chosen. Initially, because we use 2 cameras, two filter graphs are created by calling function *'CoCreateInstance'*. Next, the enumerators are created to specify the video capture devices that exist in the system and they keep the property of the devices for the next interfaces. Then the capture filters and the grabber filters are added to the graphs and the graphs are built. In addition, the camera formats can be set to serve the applications; for example, the image size is set for 640x480 pixels, and the camera frame rate is set to 10 frames per second for visualizing the alternating lighted marker. Next, to render the video, the *renderer* filters are added to the graph. The preview windows can also be set in our dialog instead of floating popup. When these procedures are all specified, the graphs are ready to run. So far if we run the application, the dialog window will only show the live video preview of both cameras. To capture images for the processing in the next issue, a flag is created to check the status of the capturing. This flag will get set to true in order to take a picture. Then the image data is stored in a data buffer to later employ in the next procedure.

3.3.4 Image Processing

The concept of image processing is the same subject as mentioned in 3.2.3 and the details in 2.7.1. The image buffers obtained from DirectShow are processed by

using OpenCV library through our own **ImProc** class, which contains functions as follow.

```
void LabelMarkers(BYTE* pBuffer, long width, long height);
void ConnectedComponents(IplImage* image);
```

The function *'LabelMarkers'* gets image data to perform converting RGB image to gray scale image with *cvCvtColor*, thresholding with *cvThreshold*, and calling *'ConnectedComponents'* function to label marker areas through the helpful command, *cvFloodFill* and find 2D centroids of those areas. For a set of markers, there are two groups of centroids that are from left camera and right camera.

3.3.5 Triangulation 3D Positions

The 2D centroids of the markers from left camera and right camera obtained from the previous section are brought to another class called **Triangulation** class to compute 3D position of markers. To manage the 2D and 3D parameters that are generally used in the program, we create the parameters '*coor2D*' for any 2D positions and '*coor3D*' for any 3D positions. This class performs stereo triangulation with our marker identification to obtain accurate 3D positions. The member functions of **Triangulation** class are shown as follows.

```
bool arrangeMarker();
coor2D normalize(coor2D coor,double fc[],double cc[],
                 double kc[]);
coor3D stereo_triangulation(coor coxyL, coor coxyR);
coor3D findDistance(coor3D group[]);
coor3D findOrigin(coor3D positions[3]);
void findTransform(void);
```

The marker centroids from left and right cameras are paired by calling function *'arrangeMarker'*, which matches all possible combinations of marker pair and determine the right one as in 3.2.5. To accomplish this, the function *'normalize'* and *'stereo_triangulation'* are accompanied as in 3.2.4. Once we obtain the 3D positions of markers, the next step is to pass the value of marker positions to display in

virtual reality by OpenGL. This procedure involved the spatial transformation. First of all, we must generate the coordinate frame of the markers. According to the non-regular triangle shape of the marker (Figure 3.9a), we set the point $\vec{\mathbf{o}}$ as the origin of the frame. From Figure 3.9b, the point $\vec{\mathbf{o}}$ is determined in the *'findOrigin'* function and has details as the following.

Let

$$\vec{\mathbf{v}}_1 = \vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_3$$
$$\vec{\mathbf{v}}_2 = \vec{\mathbf{x}}_2 - \vec{\mathbf{x}}_3$$
$$\vec{\mathbf{v}}_3 = \vec{\mathbf{o}} - \vec{\mathbf{x}}_3$$
$$\vec{\mathbf{v}}_4 = \vec{\mathbf{x}}_2 - \vec{\mathbf{o}}$$

From Figure 3.9b

$$\vec{\mathbf{v}}_2 = \vec{\mathbf{v}}_3 + \vec{\mathbf{v}}_4$$
$$\vec{\mathbf{v}}_4 = \vec{\mathbf{v}}_2 - \vec{\mathbf{v}}_3 \quad (3.1)$$

where $\vec{\mathbf{v}}_3 = c\,\vec{\mathbf{v}}_1 \quad (3.2)$

and $\vec{\mathbf{v}}_4 \cdot \vec{\mathbf{v}}_1 = 0$

From equations (3.1) and (3.2)

$$(\vec{\mathbf{v}}_2 - c\,\vec{\mathbf{v}}_1) \cdot \vec{\mathbf{v}}_1 = 0$$
$$\vec{\mathbf{v}}_2 \cdot \vec{\mathbf{v}}_1 - c\,\vec{\mathbf{v}}_1 \cdot \vec{\mathbf{v}}_1 = 0$$
$$c = \frac{\vec{\mathbf{v}}_2 \cdot \vec{\mathbf{v}}_1}{\vec{\mathbf{v}}_1 \cdot \vec{\mathbf{v}}_1}$$

Therefore, $\vec{\mathbf{v}}_3 = \frac{\vec{\mathbf{v}}_2 \cdot \vec{\mathbf{v}}_1}{\vec{\mathbf{v}}_1 \cdot \vec{\mathbf{v}}_1}\vec{\mathbf{v}}_1$

and $\vec{\mathbf{o}} = \vec{\mathbf{x}}_3 + \vec{\mathbf{v}}_3$

Then the **x-**, **y-**, and **z-axes** are located as illustrated in Figure 3.9c. However, the created marker coordinate frame differently orientates from the camera coordinate frame as illustrated in Figure 3.10a and Figure 3.10b. Therefore, the transformation between marker coordinate frame and camera coordinate frame is determined as described in subject 2.8.2 with equation (2.8) and equation (2.18), which is processed in the *'findTransform'* function.

The **x-**, **y-**, and **z-axes** of marker frame are defined in relation to camera coordinate frame. Each axis is calculated from finding the unit vector of that axis as follows.

$$\vec{\mathbf{u}}_x = \frac{\vec{\mathbf{x}}_3 - \vec{\mathbf{o}}}{\left\|\vec{\mathbf{x}}_3 - \vec{\mathbf{o}}\right\|}$$

$$\vec{\mathbf{u}}_y = \frac{\vec{\mathbf{x}}_2 - \vec{\mathbf{o}}}{\left\|\vec{\mathbf{x}}_2 - \vec{\mathbf{o}}\right\|}$$

$$\vec{\mathbf{u}}_z = \vec{\mathbf{u}}_x \times \vec{\mathbf{u}}_y$$

Therefore, the homogeneous transform of marker coordinate frame and camera coordinate frame becomes:

$$\prescript{C}{M}{\mathbf{T}} = \begin{bmatrix} \begin{bmatrix} \vec{\mathbf{u}}_x \end{bmatrix} & \begin{bmatrix} \vec{\mathbf{u}}_y \end{bmatrix} & \begin{bmatrix} \vec{\mathbf{u}}_z \end{bmatrix} & \begin{bmatrix} \vec{\mathbf{o}} \end{bmatrix} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

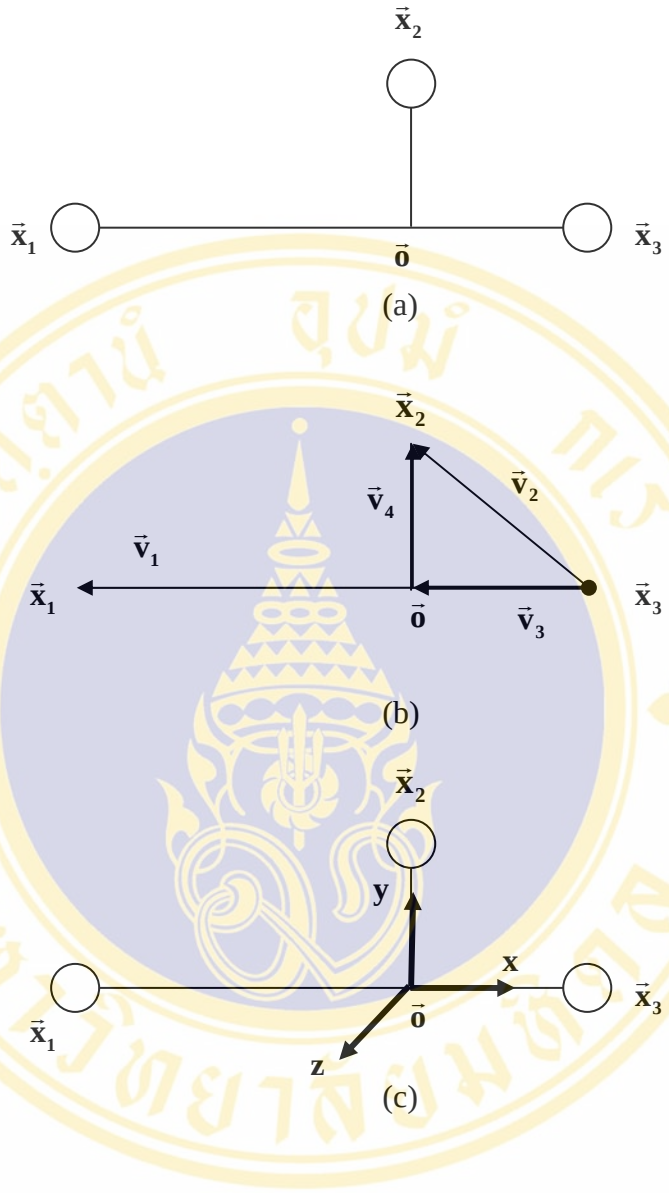


Figure 3.9 **(a) The extracted origin coordinate $\vec{o}$ from marker positions**
(b) How to calculate the origin coordinate $\vec{o}$ from vectors
(c) Marker Coordinate Frame

In addition, OpenGL also has its own coordinate frame as shown in Figure 3.10b. As a result, the transformation between frames must be established also. We must determine the transformation between marker coordinate frame and OpenGL coordinate frame to pass the correct position values for visualization. The homogeneous transform will be:

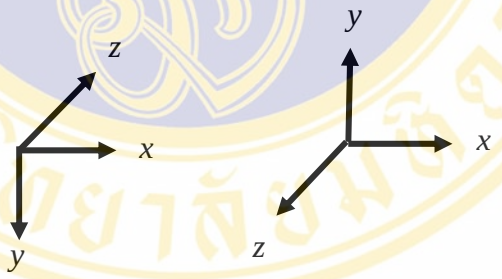
$$ {}^{GL}_{M}\mathbf{T} = {}^{GL}_{C}\mathbf{T}\,{}^{C}_{M}\mathbf{T} \tag{3.4} $$

Due to Figure 3.10 including equations (2.8), (2.18) and (3.3), equation (3.4) becomes:

$$ {}^{GL}_{M}\mathbf{T} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \begin{bmatrix} \\ \vec{\mathbf{u}}_x \\ \\ 0 \end{bmatrix} & \begin{bmatrix} \\ \vec{\mathbf{u}}_y \\ \\ 0 \end{bmatrix} & \begin{bmatrix} \\ \vec{\mathbf{u}}_z \\ \\ 0 \end{bmatrix} & \begin{bmatrix} \\ \vec{\mathbf{o}} \\ \\ 1 \end{bmatrix} \end{bmatrix} \tag{3.5} $$

The product of equation (3.5) can be represented as follows.

$$ {}^{GL}_{M}\mathbf{T} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6} $$



(a) (b)

Figure 3.10 Coordinate frames

(a) Camera coordinate frame

(b) OpenGL coordinate frame

At present, we obtain the 3D coordinates of markers that are prepared to the OpenGL domain via p_x, p_y and p_z. Next, to specify the orientation of markers to OpenGL, the rotation angles must be in the form of rotation about **x**-, **y**-, or **z**-axis. Consequently, the rotation matrix of ${}^{GL}_{M}\mathbf{T}$ from equation (3.6) is computed for rotation

angles. In this implementation, we choose the Z-X-Z Euler angles with angles γ, α and β, respectively, to order the rotation in OpenGL. The solution for extracting Z-X-Z Euler angles from a rotation matrix is expressed in equation (2.18), where

$$\gamma = \text{atan2}\left(r_{31}, r_{32}\right),$$
$$\alpha = \text{atan2}\left(r_{13}, -r_{23}\right),$$
$$\beta = \text{atan2}\left(r_{31}\text{s}_{\gamma} + r_{32}\text{c}_{\gamma}, r_{33}\right)$$

As a result, the output from the *Triangulation* class, which are the 3D positions of the origin of marker frame and the rotation angles about Z-X-Z axes indicating the orientation of the markers, are calculated to pass to the visualization in OpenGL domain.

3.3.6 Illustration of 3D Positions of 2 Set of Markers in Virtual Reality by OpenGL

The positions and orientations of 2 sets of markers from the process above are brought to the virtual environment in OpenGL every 2 captured frames. The simulated markers are drawn with those parameters to represent the poses of markers, which are continually updated for real-time application. There is a window with 4 views for the visual representation of three-dimensional objects, which are the isometric view, top view, side view and front view.

3.4 Error Study and Analysis

To analyze the error of the system, the manipulator MOTOMAN-HP6 (repeatability ±0.08 mm) from Yaskawa Electric Corporation (Figure 3.11) with NX100 robot controller [32] was used by attaching the markers at the tip of the clamp. It has a robot controller that can define and control the location of the markers, which was employed to compare to the calculated position from the stereoscopic camera system. The coordinates of markers at each location were calculated according to previous sections and are transformed to the robot coordinate frame. Then the relative

distances to the defined reference coordinate in robot coordinate system are calculated, also.

The manipulator was programmed to move in the area of 20x20x20 cm^3 with 5 centimeters spacing. Therefore, there were 125 positions to test the system. We defined an origin of this movement at one position and move the robot to the left, right, top, bottom, front, and back of this point in defined area. Then the relative distances to the origin were calculated.

The errors of the relative distances were then analyzed for the following:

- Statistic evaluation such as mean and median of the translational errors in distance units
- Specific error evaluation at the working areas by determination the error range from the variation of the displacement, for example.

Figure 3.11 The manipulator with attached marker for testing errors of the system

CHAPTER IV
EXPERIMENTAL RESULTS

The experiments were divided into 2 parts, which were the implementations of the system by MATLAB® and the system by Microsoft Visual C++ together with the results from the error analysis.

4.1 System Using MATLAB®

After performing the calibration procedure, we obtained the intrinsic parameters, which are the focal length, lens distortion coefficient, the uncertainty scale factor from camera scanning, and the center of the computer image coordinates. Also, we got the extrinsic parameters, which were the rotation parameters and the translation parameters for the transformation from 3D world coordinate system to camera coordinate system. This information was then used to calculate the 3D positions and orientations of markers from the captured images. The marker detection employed only in the infrared system. The active markers and passive markers were tested and obtained results as shown in Figure 4.1 and Figure 4.2, respectively.

The example of the program output from Figure 4.1 showed the result from capturing active marker images. Figure 4.1a and Figure 4.1b illustrate the images obtained from 2 cameras. Figure 4.1c and Figure 4.1d revealed the centroids of each marker. And Figure 4.1e was the outcome of the calculation 3D positions from the images. Figure 4.2 showed the example of detecting passive markers. Figure 4.2a to Figure 4.2e share the same explanation as Figure 4.1.

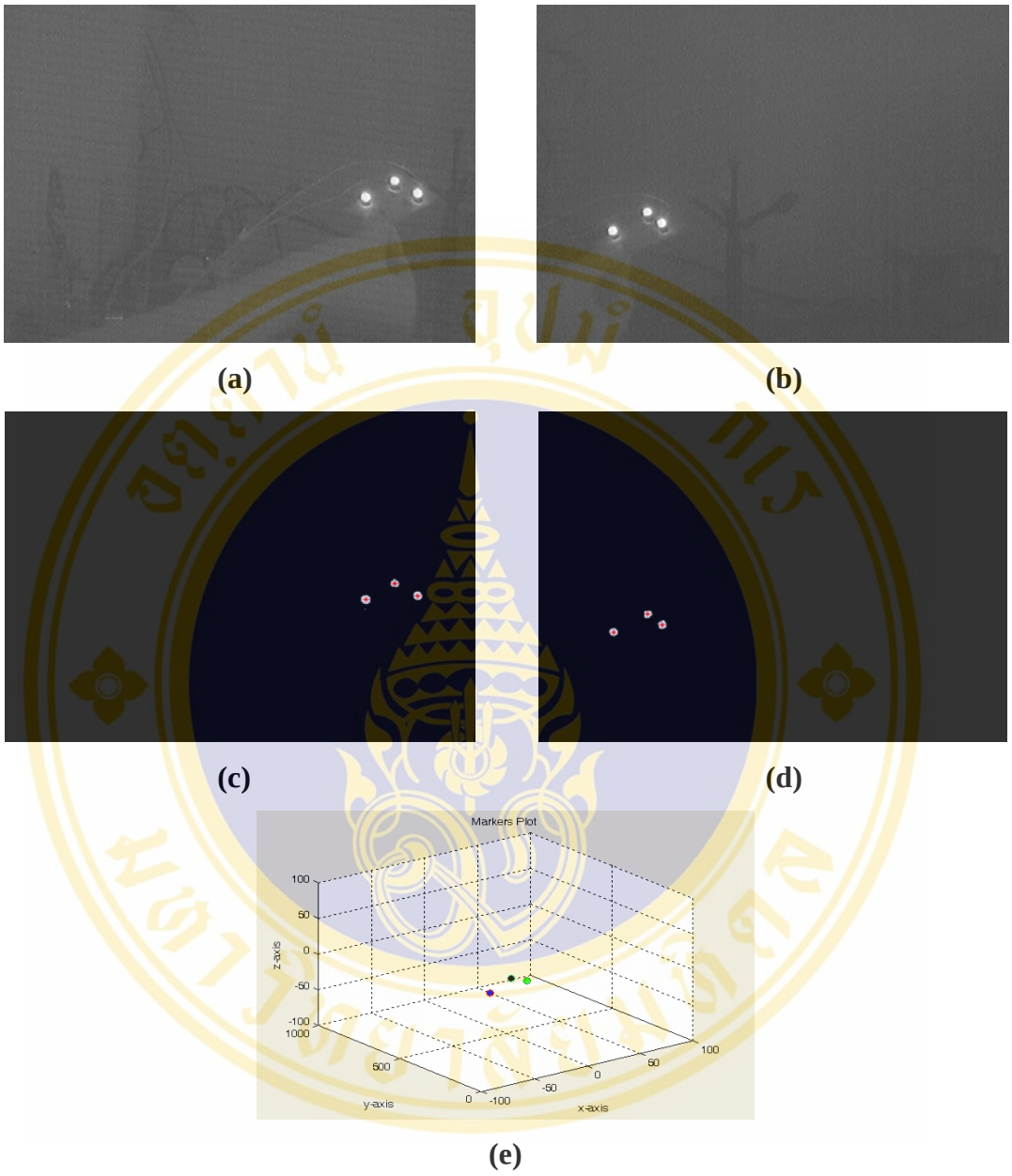


Figure 4.1 Active markers detection.

(a) Captured image from left camera

(b) Captured image from right camera

(c) Centroids of markers from (a) after image processing

(d) Centroids of markers from (b) after image processing

(e) Calculated 3D marker positions

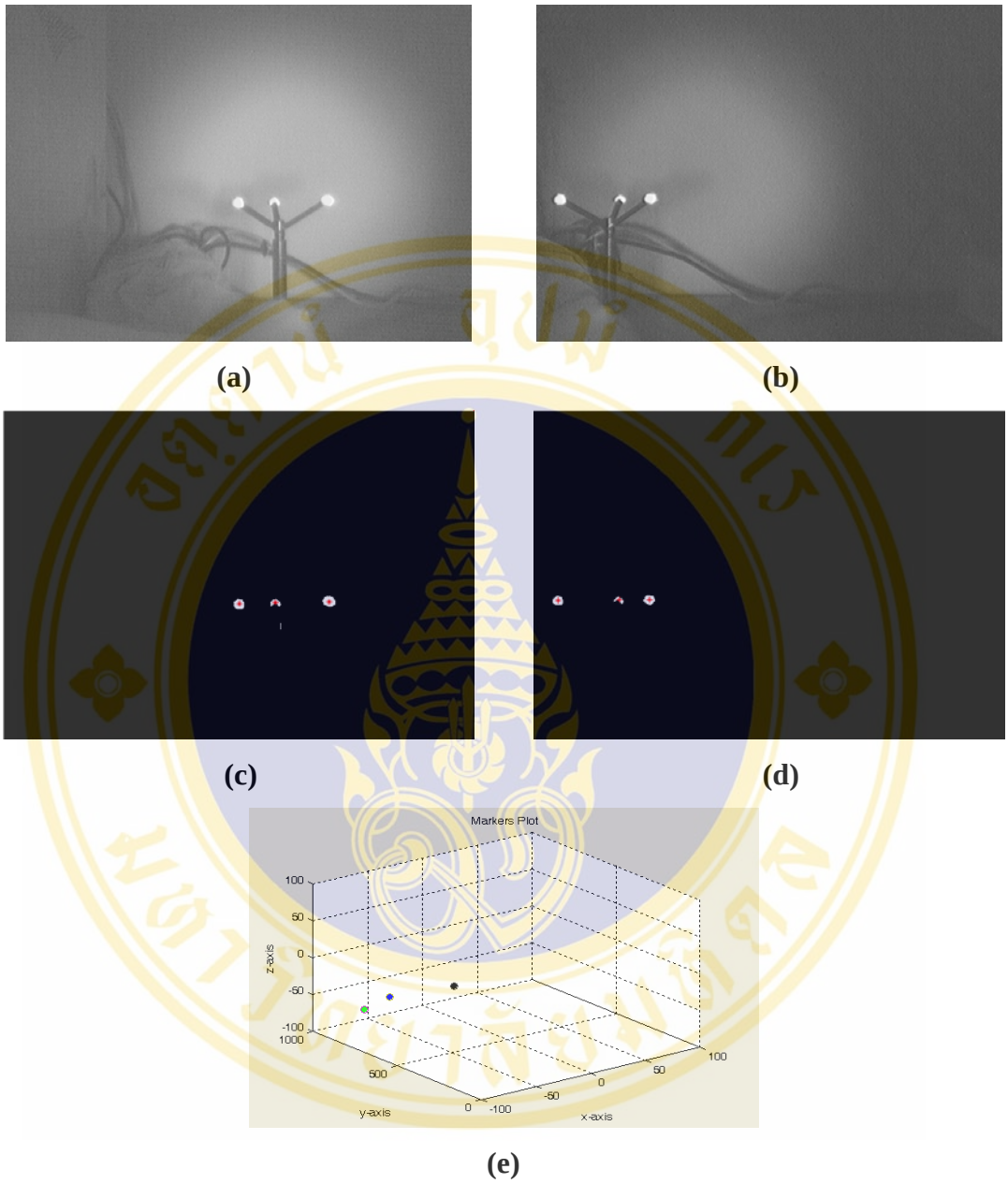


Figure 4.2 Passive markers detection.

(a) Captured image from left camera

(b) Captured image from right camera

(c) Centroids of markers from (a) after image processing

(d) Centroids of markers from (b) after image processing

(e) Calculated 3D marker positions

4.2 System Using Microsoft Visual C++

In this system we used 2 sets of active markers to represent the position and orientation of 2 devices. In the capturing process, we turned on the markers and then processed one by one and showed the 3D positions of two markers in virtual reality. In the image processing process, we used the same calibration parameters that calculated from MATLAB® implementation to calculate 3D marker positions. The example of the program that captured the marker images from left and right camera is shown in Figure 4.3 and the 3D positions and orientations of each marker were calculated and illustrated in virtual reality as shown in Figure 4.4 in isometric view, top view, side view, and front view.

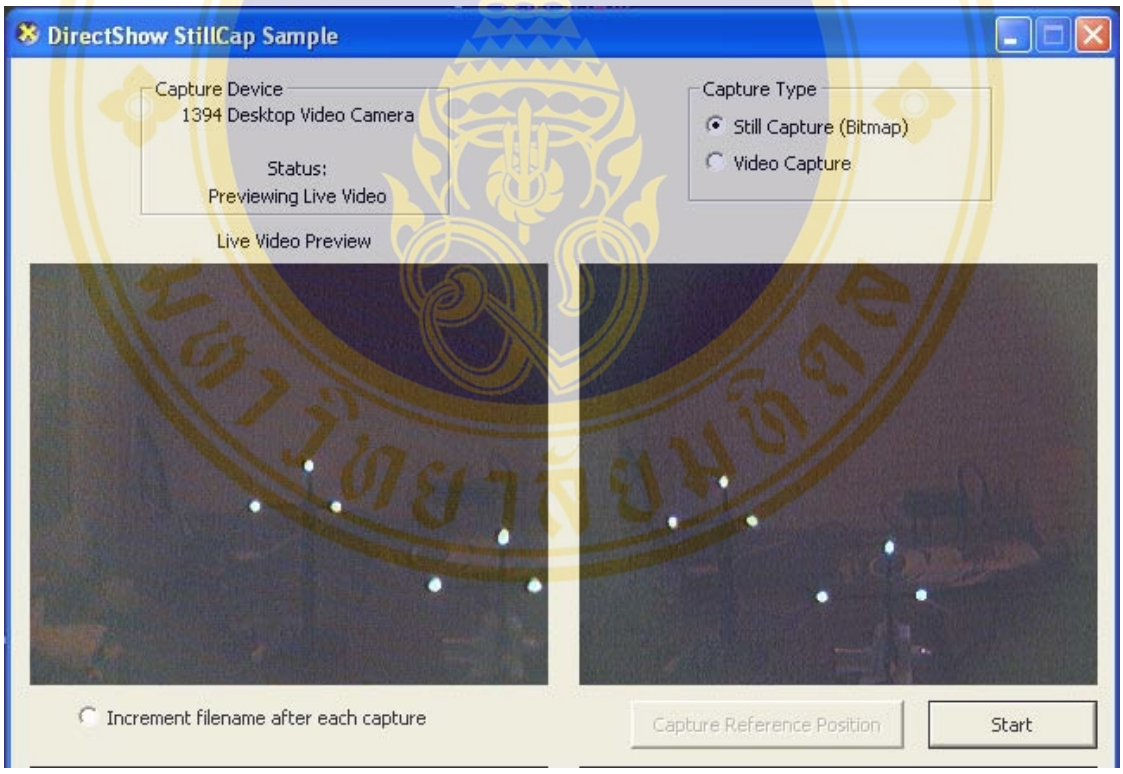


Figure 4.3 The captured window by DirectShow

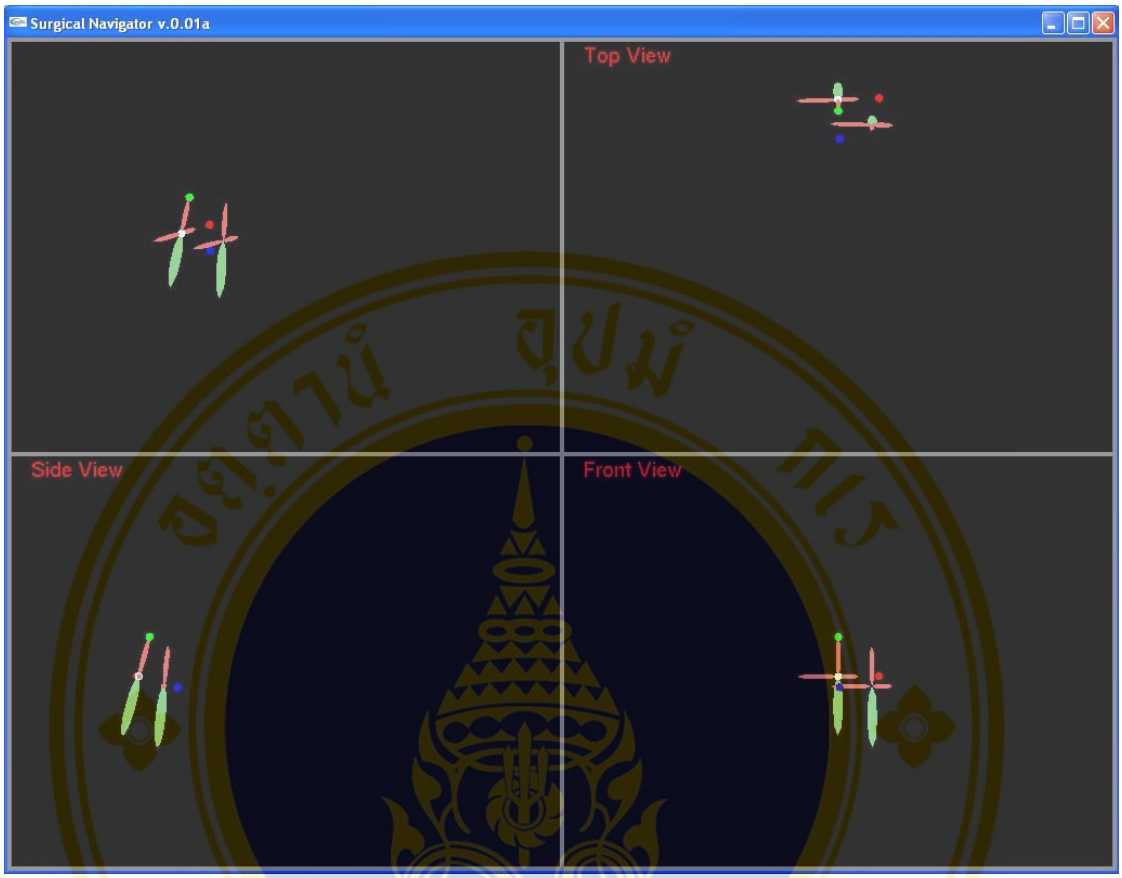


Figure 4.4 The visualization of 3D positions and orientations of marker by OpenGL.

Figure 4.3 and Figure 4.4 also show that the created virtual reality can detect the movement of the markers and can represent a three-dimensional object in two dimensions. If the markers move to the left hand side, the visualization also illustrates the objects in the same direction as well as the orientation.

4.3 Error Analysis

Markers were attached to the manipulator and moved from the defined reference origin (0, 0, 0) to the range (-100, -100, -100) to (100, 100, 100). We conducted the test for the translation errors three times. The error of the relative distances of markers were calculated for maximum error (Max), mean error (Mean), and median of error (Median) in each axis. From following table ‘X-axis’ means the direction in the left and the right. ‘Y-axis’ means the directions in the top and the

bottom. ‘Z-axis’ means the directions in the front and the back. The errors are shown in the Table 4.1, Table 4.2, Table 4.3 and analyzed errors in each axis is shown in Table 4.4.

Table 4.1 Comparison between reference positions and computed positions of Test1

Posi-tion	Reference Position x-axis	Reference Position y-axis	Reference Position z-axis	Computed Position x-axis	Computed Position y-axis	Computed Position z-axis	Error x-axis	Error y-axis	Error z-axis
1	-100.056	-100.127	-100.147	-100.317	-103.241	-98.7028	-0.26078	-3.11411	1.44419
2	-50.177	-100.117	-100.15	-50.1126	-102.24	-98.303	0.06445	-2.12263	1.84702
3	0.07	-100.12	-100.152	-0.27581	-98.2779	-99.5364	-0.34581	1.84209	0.6156
4	50.274	-100.185	-100.167	49.9496	-95.9389	-100.457	-0.3244	4.24606	-0.29028
5	100.284	-100.225	-100.163	100.397	-94.6551	-100.181	0.1133	5.56988	-0.01757
6	-100.402	-50.056	-100.206	-100.634	-55.3126	-98.4165	-0.23249	-5.25663	1.78949
7	-50.286	-50.021	-100.212	-50.6869	-54.5622	-98.4503	-0.40088	-4.54117	1.7617
8	0.113	-50.299	-100.252	-0.13286	-50.5578	-99.6683	-0.24586	-0.25884	0.58368
9	50.062	-50.31	-100.229	49.7225	-48.331	-100.022	-0.33951	1.97903	0.2071
10	100.05	-50.31	-100.219	99.7669	-47.2957	-100.717	-0.28305	3.0143	-0.49806
11	-100.223	0.179	-100.048	-101.015	-1.39289	-98.6929	-0.79151	-1.57189	1.35507
12	-50.162	0.184	-100.048	-51.3224	-1.47791	-99.1042	-1.16044	-1.66191	0.94381
13	0.149	0.196	-100.079	-0.55303	-2.84697	-99.2722	-0.70203	-3.04297	0.80685
14	50.143	0.218	-100.071	49.7114	-0.68873	-99.5833	-0.43158	-0.90673	0.48766
15	100.177	0.23	-100.111	100.178	0.5331	-99.7653	0.00135	0.3031	0.34572
16	-100.08	50.182	-100.144	-101.337	46.5894	-98.4364	-1.25716	-3.59255	1.70756
17	-50.181	50.114	-100.089	-51.4308	46.0034	-98.7812	-1.24975	-4.1106	1.30777
18	0.024	50.114	-100.114	-1.02979	49.959	-99.9631	-1.05379	-0.15501	0.15095
19	50.057	50.119	-100.133	49.6908	52.204	-100.865	-0.36621	2.08505	-0.73203
20	100.106	50.17	-100.061	99.3812	47.3395	-99.314	-0.72479	-2.83049	0.74697
21	-100.155	100.258	-100.054	-101.677	99.4138	-99.3368	-1.52191	-0.84423	0.71719
22	-50.19	100.244	-100.072	-51.5471	98.1974	-99.6064	-1.35712	-2.04656	0.4656
23	0.278	100.361	-100.082	-0.35055	95.981	-99.095	-0.62855	-4.38003	0.98703
24	50.218	100.375	-100.142	49.9433	104.574	-100.477	-0.27473	4.19902	-0.33483
25	100.233	100.379	-100.134	99.7223	98.8182	-100.023	-0.5107	-1.56077	0.11111
26	-100.066	-100.012	-50.134	-100.019	-101.9	-48.5616	0.0471	-1.88766	1.57237
27	-50.06	-100.176	-50.134	-49.8956	-101.087	-49.0435	0.16444	-0.91064	1.09053
28	0.081	-100.193	-50.117	-0.19029	-97.5046	-49.9442	-0.27129	2.68839	0.1728
29	50.053	-100.223	-50.113	50.0458	-100.194	-49.881	-0.0072	0.02866	0.232
30	100.043	-100.207	-50.124	99.9114	-95.6914	-49.9502	-0.13163	4.51556	0.17384
31	-100.106	-50.155	-50.108	-100.304	-49.7735	-49.1835	-0.19754	0.38147	0.92452
32	-50.144	-50.138	-50.111	-50.1832	-49.4331	-49.6183	-0.03917	0.70485	0.49265
33	0.033	-50.139	-50.118	-0.06983	-50.6851	-49.8107	-0.10283	-0.54613	0.30727
34	50.003	-50.142	-50.112	49.5443	-48.8869	-49.934	-0.45874	1.25511	0.17801
35	100.146	-50.151	-50.115	100.287	-48.8918	-49.7921	0.14065	1.25918	0.32295
36	-100.071	0.126	-50.112	-100.608	-1.61349	-48.954	-0.53662	-1.73949	1.15799
37	-50.104	0.127	-50.115	-50.8023	3.57919	-50.0852	-0.69828	3.45219	0.02985
38	0.078	0.126	-50.127	0.05559	1.72475	-50.2335	-0.02241	1.59875	-0.1065
39	50.097	0.125	-50.126	50.335	3.17785	-50.2837	0.238	3.05285	-0.15775

Table 4.1 Comparison between reference positions and computed positions of Test1 (Continued)

Posi-tion	Reference Position x-axis	Reference Position y-axis	Reference Position z-axis	Computed Position x-axis	Computed Position y-axis	Computed Position z-axis	Error x-axis	Error y-axis	Error z-axis
40	100.217	0.122	-50.12	99.5625	-1.90477	-49.4426	-0.6545	-2.02677	0.6774
41	-100.154	50.124	-50.118	-100.863	51.2209	-49.7493	-0.709	1.09691	0.3687
42	-50.073	50.121	-50.137	-50.8843	50.7922	-49.5571	-0.81126	0.67123	0.5799
43	0.004	50.122	-50.143	-0.40981	54.2778	-50.4514	-0.41381	4.1558	-0.30839
44	50.144	50.115	-50.148	50.313	55.834	-51.0966	0.169	5.71901	-0.94858
45	100.048	50.107	-50.148	99.9113	50.0739	-49.6024	-0.13671	-0.03307	0.54556
46	-100.119	100.116	-50.173	-101.173	97.5203	-49.0516	-1.0536	-2.59573	1.12141
47	-50.042	100.116	-50.167	-50.047	96.1463	-49.3514	-0.00503	-3.96973	0.81557
48	0.13	100.114	-50.174	0.28114	99.522	-50.2391	0.15114	-0.59201	-0.06506
49	50.031	100.111	-50.185	49.896	94.6947	-48.8696	-0.13505	-5.41633	1.31544
50	100.04	100.111	-50.172	100.248	101.282	-50.1602	0.20765	1.1714	0.01179
51	-100.107	-100.138	0.1	-99.7994	-101.122	0.87636	0.30762	-0.98435	0.77636
52	-50.164	-100.146	0.089	-49.7705	-100.562	0.96263	0.39348	-0.41552	0.87363
53	0.085	-100.124	0.028	0.78128	-97.3072	0.3313	0.69628	2.81684	0.3033
54	50.159	-100.105	0.078	50.5767	-96.2943	-0.06529	0.41775	3.81074	-0.14329
55	100.063	-100.171	0.111	100.362	-96.9732	-0.27303	0.29949	3.19782	-0.38403
56	-100.124	-50.024	0.052	-100.017	-49.8025	0.78491	0.10732	0.22147	0.73291
57	-50.149	-50.283	0.05	-50.0397	-49.7165	0.38619	0.10934	0.56648	0.33619
58	0.122	-50.256	0.021	0.428	-46.5102	-0.25795	0.306	3.74585	-0.27895
59	50	-50.259	0.043	50.3059	-50.2861	-0.24806	0.30589	-0.0271	-0.29106
60	100.002	-50.226	0.033	99.6653	-50.9781	0.03926	-0.33672	-0.75209	0.00626
61	-100.178	0.196	0.048	-100.24	-2.6759	1.03439	-0.06219	-2.8719	0.98639
62	-50.01	0.185	0.039	-49.7947	-3.09933	0.65097	0.2153	-3.28433	0.61197
63	0	0	0	0	0	0	0	0	0
64	50.198	0	0.291	50.8662	6.63207	0.19358	0.6682	6.63207	-0.09742
65	100.101	0	0.29	100.023	0.64339	0.36876	-0.07807	0.64339	0.07876
66	-100.24	50.082	0.264	-100.468	43.705	1.7905	-0.22775	-6.37698	1.5265
67	-50.013	50.062	0.243	-50.4	54.5905	0.47585	-0.38703	4.52849	0.23285
68	0.278	50.065	0.256	0.69206	51.6905	0.289	0.41406	1.62555	0.033
69	50.273	50.071	0.266	51.1024	52.7134	-0.18091	0.82941	2.64236	-0.44691
70	100.168	50.069	0.267	100.377	52.4608	0.06385	0.20867	2.39176	-0.20315
71	-100.023	100.247	0.284	-100.699	94.6574	1.4287	-0.67584	-5.5896	1.1447
72	-50.279	100.25	0.279	-50.7527	99.4501	0.5838	-0.47374	-0.7999	0.3048
73	0.214	100.249	0.31	0.18884	96.3345	0.98354	-0.02516	-3.91451	0.67354
74	50.129	100.059	0.101	50.4207	97.3484	0.49231	0.29173	-2.71062	0.39131
75	100.105	100.061	0.113	100.71	103.391	-0.35575	0.60554	3.33028	-0.46875
76	-100.177	-100.133	50.131	-99.6054	-101.407	50.1524	0.57162	-1.27439	0.02139
77	-50.17	-100.128	50.134	-49.4707	-96.9343	50.1231	0.69934	3.19365	-0.01091
78	0.17	-100.125	50.146	0.64158	-98.3504	49.8589	0.47158	1.77456	-0.2871
79	50.018	-100.136	50.15	50.2136	-97.6544	49.5889	0.19563	2.48156	-0.56112
80	100.08	-100.135	50.146	100.761	-94.3712	49.305	0.68125	5.76377	-0.84102
81	-100.059	-50.027	50.128	-99.796	-50.6674	50.4993	0.26303	-0.64042	0.37132
82	-50.078	-50.027	50.143	-49.7228	-46.1258	49.9503	0.3552	3.90121	-0.19264
83	0.151	-50.033	50.127	0.27237	-48.1468	49.6721	0.12137	1.88616	-0.45487

Table 4.1 Comparison between reference positions and computed positions of Test1 (Continued)

Posi-tion	Reference Position x-axis	Reference Position y-axis	Reference Position z-axis	Computed Position x-axis	Computed Position y-axis	Computed Position z-axis	Error x-axis	Error y-axis	Error z-axis
84	50.134	-50.03	50.12	50.7166	-47.5094	49.9023	0.58263	2.52059	-0.21765
85	100.19	-50.026	50.123	100.072	-48.4575	49.7285	-0.11787	1.56853	-0.39451
86	-100.188	0.158	50.135	-99.9766	-4.24258	51.2144	0.21141	-4.40058	1.07941
87	-50.135	0.16	50.149	-49.4497	0.30662	50.6489	0.68531	0.14662	0.49986
88	0.025	0.162	50.142	0.40123	3.03716	50.1913	0.37623	2.87516	0.04927
89	50.191	0.162	50.138	50.6749	-1.60869	50.0314	0.48394	-1.77069	-0.10659
90	100.176	0.161	50.138	100.419	2.82605	49.655	0.24299	2.66505	-0.48298
91	-100.073	50.093	50.128	-100.157	47.1145	50.8569	-0.08359	-2.97854	0.72894
92	-50.109	50.095	50.133	-50.3017	51.5604	50.2671	-0.19275	1.46541	0.13409
93	0.168	50.089	50.143	0.51586	48.3445	49.9601	0.34786	-1.74453	-0.18285
94	50.002	50.078	50.129	50.6241	49.049	49.6037	0.62208	-1.02897	-0.52574
95	100.11	50.09	50.127	100.741	53.9897	49.7696	0.63125	3.89971	-0.35735
96	-100.148	100.005	50.131	-100.336	97.783	50.6402	-0.18787	-2.22204	0.50921
97	-50.188	100.006	50.139	-50.3558	102.519	50.6286	-0.16777	2.51295	0.48957
98	0.178	100.173	50.138	0.63035	98.5952	50.3064	0.45235	-1.57781	0.16836
99	50.073	100.172	50.144	50.8479	99.1592	50.5363	0.7749	-1.01281	0.39229
100	100.183	100.186	50.161	101.093	104.914	50.0753	0.90988	4.72824	-0.08567
101	-100.031	-100.239	100.233	-98.5227	-102.169	100.192	1.50825	-1.92952	-0.04057
102	-50.108	-100.243	100.207	-49.5308	-98.3919	99.9242	0.57718	1.85109	-0.2828
103	0.182	-100.223	100.243	0.8883	-95.8729	99.7015	0.7063	4.35015	-0.54154
104	50.089	-100.218	100.242	50.4976	-95.3447	99.5291	0.40861	4.8733	-0.71288
105	100.089	-100.264	100.233	101.081	-92.2071	99.0444	0.99237	8.0569	-1.18858
106	-100.246	-50.081	100.277	-99.6249	-52.3702	100.464	0.62114	-2.2892	0.18707
107	-50.033	-50.086	100.28	-49.7461	-48.4517	100.163	0.28693	1.6343	-0.11714
108	0.112	-50.108	100.25	1.05792	-50.7464	99.7919	0.94592	-0.63844	-0.45807
109	50.06	-50.116	100.238	51.0159	-45.4238	99.7019	0.95593	4.6922	-0.53612
110	100.258	-50.097	100.252	100.365	-51.1834	99.4618	0.10751	-1.08643	-0.79022
111	-100	0.009	100.248	-99.7532	-1.50325	100.644	0.24677	-1.51225	0.39562
112	-50.044	0.008	100.254	-49.4434	-2.90223	100.183	0.60057	-2.91023	-0.07138
113	0.05	0.008	100.251	0.70791	4.88313	100.012	0.65791	4.87513	-0.23863
114	50.034	0.007	100.264	50.9846	0.06494	99.6556	0.9506	0.05794	-0.60835
115	100.222	0.006	100.281	100.7	-0.87103	99.493	0.47782	-0.87703	-0.78797
116	-100.088	50.018	100.285	-99.9208	49.8703	101.292	0.16722	-0.14771	1.00663
117	-50.188	50.034	100.291	-50.2676	47.5113	100.199	-0.07959	-2.52273	-0.09202
118	0.144	50.211	100.3	0.83491	50.2205	100.463	0.69091	0.00948	0.16277
119	50.215	50.213	100.005	50.9485	50.4827	99.6055	0.73349	0.26967	-0.3995
120	100.151	50.232	100.061	101.032	49.5774	99.409	0.88137	-0.65455	-0.65197
121	-100.185	100.241	100.051	-100.027	93.6486	101.101	0.15811	-6.59243	1.05008
122	-50.191	100.231	100.034	-50.2976	97.5265	100.698	-0.10661	-2.70445	0.66394
123	0.042	100.211	100.056	0.35947	93.3723	100.204	0.31747	-6.83867	0.14793
124	50.148	100.211	100.057	51.1864	100.423	100.041	1.03843	0.21199	-0.01632
125	100.035	100.231	100.063	100.133	99.2432	99.1807	0.09817	-0.98783	-0.88228

Table 4.2 Comparison between reference positions and computed positions of Test2

Posi-tion	Reference Position x-axis	Reference Position y-axis	Reference Position z-axis	Computed Position x-axis	Computed Position y-axis	Computed Position z-axis	Error x-axis	Error y-axis	Error z-axis
1	-100.056	-100.127	-100.147	-100.383	-97.559	-100.955	-0.32654	2.568022	-0.80805
2	-50.177	-100.117	-100.15	-50.5751	-97.5508	-100.419	-0.39811	2.566232	-0.2685
3	0.07	-100.12	-100.152	0.327096	-99.2816	-100.284	0.257096	0.83837	-0.13242
4	50.274	-100.185	-100.167	50.46158	-96.8982	-100.804	0.187583	3.286836	-0.63701
5	100.284	-100.225	-100.163	100.7415	-95.7693	-100.632	0.457472	4.455744	-0.46907
6	-100.402	-50.056	-100.206	-99.4844	-52.6563	-100.319	0.917611	-2.60027	-0.11318
7	-50.286	-50.021	-100.212	-49.888	-53.4199	-100.323	0.398003	-3.39886	-0.11056
8	0.113	-50.299	-100.252	-0.01541	-49.2683	-100.412	-0.12841	1.030666	-0.16036
9	50.062	-50.31	-100.229	50.64927	-52.9303	-99.5377	0.587268	-2.62027	0.691303
10	100.05	-50.31	-100.219	100.4556	-45.6122	-100.798	0.405624	4.697802	-0.579
11	-100.223	0.179	-100.048	-99.7804	-3.84219	-99.6235	0.442623	-4.02119	0.424504
12	-50.162	0.184	-100.048	-50.2832	1.718622	-100.458	-0.12121	1.534622	-0.41007
13	0.149	0.196	-100.079	0.458011	-1.16015	-100.298	0.309011	-1.35615	-0.21935
14	50.143	0.218	-100.071	50.34008	1.518713	-100.256	0.197075	1.300713	-0.18516
15	100.177	0.23	-100.111	100.357	2.667434	-100.73	0.179958	2.437434	-0.61855
16	-100.08	50.182	-100.144	-100.038	49.26022	-100.148	0.041779	-0.92178	-0.00395
17	-50.181	50.114	-100.089	-49.6447	47.24024	-100.097	0.536338	-2.87376	-0.00769
18	0.024	50.114	-100.114	0.508045	43.94403	-99.2726	0.484045	-6.16997	0.841355
19	50.057	50.119	-100.133	50.53397	53.9967	-100.782	0.476969	3.8777	-0.64885
20	100.106	50.17	-100.061	100.4654	47.95533	-99.7411	0.359378	-2.21467	0.319945
21	-100.155	100.258	-100.054	-100.096	99.02868	-100.448	0.059172	-1.22932	-0.39373
22	-50.19	100.244	-100.072	-50.1682	104.5602	-101.321	0.02184	4.316163	-1.24858
23	0.278	100.361	-100.082	1.067535	100.6526	-100.445	0.789535	0.291573	-0.36281
24	50.218	100.375	-100.142	50.49301	95.00288	-100.136	0.275008	-5.37212	0.005686
25	100.233	100.379	-100.134	100.5714	96.65456	-100.016	0.338356	-3.72444	0.118132
26	-100.066	-100.012	-50.134	-99.342	-100.925	-50.1952	0.724004	-0.91281	-0.06119
27	-50.06	-100.176	-50.134	-49.797	-101.213	-50.298	0.263041	-1.0373	-0.16404
28	0.081	-100.193	-50.117	0.057806	-97.3804	-50.7997	-0.02319	2.812637	-0.68267
29	50.053	-100.223	-50.113	50.15237	-95.1773	-50.632	0.099371	5.045725	-0.51905
30	100.043	-100.207	-50.124	100.3963	-94.3685	-50.9686	0.353298	5.838464	-0.8446
31	-100.106	-50.155	-50.108	-99.679	-51.1586	-50.8412	0.426954	-1.00359	-0.73318
32	-50.144	-50.138	-50.111	-50.1094	-51.8589	-50.3388	0.034561	-1.72088	-0.22779
33	0.033	-50.139	-50.118	0.52396	-54.3019	-50.3374	0.49096	-4.16288	-0.21939
34	50.003	-50.142	-50.112	50.34455	-51.9243	-50.1802	0.341549	-1.78229	-0.06817
35	100.146	-50.151	-50.115	100.2672	-51.2031	-50.5404	0.121157	-1.05213	-0.42539
36	-100.071	0.126	-50.112	-100.073	3.816841	-50.9214	-0.00249	3.690841	-0.80939
37	-50.104	0.127	-50.115	-49.3324	-4.36619	-49.8322	0.771589	-4.49319	0.282815
38	0.078	0.126	-50.127	0.228873	-0.51119	-50.3852	0.150873	-0.63719	-0.25819
39	50.097	0.125	-50.126	50.02415	1.759475	-50.8607	-0.07285	1.634475	-0.73466
40	100.217	0.122	-50.12	101.2381	2.932055	-50.6398	1.021074	2.810055	-0.51983
41	-100.154	50.124	-50.118	-100.103	49.47501	-49.8467	0.051125	-0.64899	0.271291
42	-50.073	50.121	-50.137	-49.7588	47.17342	-50.5852	0.3142	-2.94758	-0.44817
43	0.004	50.122	-50.143	0.768583	51.27268	-50.5119	0.764583	1.150683	-0.36886
44	50.144	50.115	-50.148	50.75822	46.08309	-50.4387	0.614218	-4.03192	-0.29074

Table 4.2 Comparison between reference positions and computed positions of Test2 (Continued)

Posi-tion	Reference Position x-axis	Reference Position y-axis	Reference Position z-axis	Computed Position x-axis	Computed Position y-axis	Computed Position z-axis	Error x-axis	Error y-axis	Error z-axis
45	100.048	50.107	-50.148	100.028	47.34952	-50.2358	-0.01997	-2.75748	-0.08781
46	-100.119	100.116	-50.173	-100.269	106.5772	-51.3321	-0.14991	6.461194	-1.15911
47	-50.042	100.116	-50.167	-49.0106	95.75441	-50.1225	1.031408	-4.36159	0.044479
48	0.13	100.114	-50.174	0.86882	99.83452	-50.0476	0.73882	-0.27948	0.126429
49	50.031	100.111	-50.185	50.99646	102.256	-50.5872	0.965455	2.14497	-0.4022
50	100.04	100.111	-50.172	101.2757	103.6974	-50.3907	1.235733	3.586399	-0.21874
51	-100.107	-100.138	0.1	-99.605	-99.0052	-0.23692	0.502041	1.132824	-0.33692
52	-50.164	-100.146	0.089	-50.0481	-99.3807	-0.4281	0.115921	0.765327	-0.5171
53	0.085	-100.124	0.028	0.548449	-101.213	0.003857	0.463449	-1.08879	-0.02414
54	50.159	-100.105	0.078	50.3664	-99.2978	-0.31749	0.207403	0.807199	-0.39549
55	100.063	-100.171	0.111	100.2899	-98.6225	-0.036	0.226921	1.548458	-0.147
56	-100.124	-50.024	0.052	-99.898	-49.7283	-0.23312	0.226015	0.295681	-0.28512
57	-50.149	-50.283	0.05	-50.3119	-50.6499	-0.43641	-0.16295	-0.36689	-0.48641
58	0.122	-50.256	0.021	0.70152	-46.8161	-0.22126	0.57952	3.439904	-0.24226
59	50	-50.259	0.043	50.07864	-50.9604	-0.35405	0.078636	-0.70142	-0.39705
60	100.002	-50.226	0.033	99.94907	-50.3651	-0.68134	-0.05293	-0.13912	-0.71434
61	-100.178	0.196	0.048	-100.013	-2.33581	0.011227	0.165427	-2.53181	-0.03677
62	-50.01	0.185	0.039	-49.4881	-3.87457	-0.20775	0.521863	-4.05957	-0.24675
63	0	0	0	0	0	0	0	0	0
64	50.198	0	0.291	51.02344	2.391752	0.246057	0.825442	2.391752	-0.04494
65	100.101	0	0.29	100.8661	3.106012	-0.11923	0.76511	3.106012	-0.40923
66	-100.24	50.082	0.264	-100.121	49.19594	-0.38393	0.118847	-0.88606	-0.64793
67	-50.013	50.062	0.243	-49.7516	54.80007	-0.20701	0.261435	4.738074	-0.45001
68	0.278	50.065	0.256	0.559954	50.9279	-0.41122	0.281954	0.862904	-0.66722
69	50.273	50.071	0.266	50.47094	45.86003	0.073899	0.197939	-4.21097	-0.1921
70	100.168	50.069	0.267	100.9469	46.71141	-0.31927	0.778915	-3.35759	-0.58627
71	-100.023	100.247	0.284	-98.6831	97.64111	0.137171	1.339905	-2.60589	-0.14683
72	-50.279	100.25	0.279	-49.1069	94.94337	0.567008	1.172109	-5.30663	0.288008
73	0.214	100.249	0.31	0.691236	98.76049	0.087216	0.477236	-1.48851	-0.22278
74	50.129	100.059	0.101	50.69687	101.0616	-0.36961	0.567868	1.002605	-0.47061
75	100.105	100.061	0.113	100.816	102.0547	-0.80368	0.710954	1.993736	-0.91668
76	-100.177	-100.133	50.131	-99.8666	-97.6073	49.71805	0.310411	2.52566	-0.41295
77	-50.17	-100.128	50.134	-50.3033	-97.8715	50.01456	-0.13335	2.256521	-0.11944
78	0.17	-100.125	50.146	0.303701	-99.7838	50.29565	0.133701	0.341154	0.149653
79	50.018	-100.136	50.15	50.15391	-97.796	50.04157	0.135908	2.340017	-0.10843
80	100.08	-100.135	50.146	100.0517	-97.388	49.76532	-0.02833	2.747044	-0.38068
81	-100.059	-50.027	50.128	-100.074	-48.7955	49.72604	-0.01482	1.231508	-0.40196
82	-50.078	-50.027	50.143	-49.3158	-49.558	50.01969	0.762192	0.469002	-0.12331
83	0.151	-50.033	50.127	0.48932	-46.0426	49.7471	0.33832	3.990407	-0.3799
84	50.134	-50.03	50.12	50.54617	-43.8595	50.06892	0.41217	6.170481	-0.05108
85	100.19	-50.026	50.123	100.9021	-49.4018	50.30439	0.71207	0.624233	0.181386
86	-100.188	0.158	50.135	-100.325	4.988536	50.37032	-0.13743	4.830536	0.23532
87	-50.135	0.16	50.149	-49.6488	-3.68037	49.92888	0.486212	-3.84037	-0.22012
88	0.025	0.162	50.142	0.253836	7.01932	49.70591	0.228836	6.85732	-0.43609

Table 4.2 Comparison between reference positions and computed positions of Test2 (Continued)

Posi-tion	Reference Position x-axis	Reference Position y-axis	Reference Position z-axis	Computed Position x-axis	Computed Position y-axis	Computed Position z-axis	Error x-axis	Error y-axis	Error z-axis
89	50.191	0.162	50.138	50.7551	2.112359	49.31253	0.564098	1.950359	-0.82547
90	100.176	0.161	50.138	100.548	3.034875	49.61275	0.371974	2.873875	-0.52525
91	-100.073	50.093	50.128	-98.8376	48.83474	49.57429	1.235398	-1.25826	-0.55371
92	-50.109	50.095	50.133	-49.862	54.22137	49.91921	0.247037	4.126374	-0.21379
93	0.168	50.089	50.143	0.369317	50.22505	49.50289	0.201317	0.136052	-0.64011
94	50.002	50.078	50.129	50.98399	52.64385	49.8219	0.981987	2.56585	-0.3071
95	100.11	50.09	50.127	100.6018	46.19848	50.04437	0.491849	-3.89152	-0.08264
96	-100.148	100.005	50.131	-100.221	104.6128	50.16991	-0.07311	4.60783	0.038906
97	-50.188	100.006	50.139	-49.1633	93.71177	50.34801	1.024664	-6.29423	0.20901
98	0.178	100.173	50.138	0.501491	97.25717	49.97948	0.323491	-2.91583	-0.15852
99	50.073	100.172	50.144	50.42507	99.65746	49.59959	0.352068	-0.51454	-0.54441
100	100.183	100.186	50.161	100.6747	92.7236	49.82153	0.491678	-7.4624	-0.33947
101	-100.031	-100.239	100.233	-99.9284	-102.27	99.33603	0.1026	-2.03115	-0.89697
102	-50.108	-100.243	100.207	-49.4856	-102.468	99.54976	0.622433	-2.22491	-0.65724
103	0.182	-100.223	100.243	0.096606	-98.7794	100.0213	-0.08539	1.443621	-0.22168
104	50.089	-100.218	100.242	50.40636	-102.383	100.0671	0.317362	-2.1646	-0.17495
105	100.089	-100.264	100.233	100.073	-101.841	99.81115	-0.01597	-1.57729	-0.42185
106	-100.246	-50.081	100.277	-100.257	-48.2186	100.2552	-0.01056	1.862404	-0.02183
107	-50.033	-50.086	100.28	-49.487	-49.0783	99.8272	0.546044	1.007732	-0.4528
108	0.112	-50.108	100.25	0.297085	-45.598	99.66739	0.185085	4.510038	-0.58261
109	50.06	-50.116	100.238	50.82515	-49.7108	99.69599	0.765152	0.405186	-0.54201
110	100.258	-50.097	100.252	100.6673	-48.9311	100.0264	0.409342	1.165891	-0.22564
111	-100	0.009	100.248	-98.9567	-2.00041	99.84958	1.043272	-2.00941	-0.39842
112	-50.044	0.008	100.254	-49.643	3.049453	99.72214	0.400957	3.041453	-0.53186
113	0.05	0.008	100.251	0.886709	-0.03605	99.79769	0.836709	-0.04405	-0.45331
114	50.034	0.007	100.264	50.53022	2.210774	100.187	0.496223	2.203774	-0.07695
115	100.222	0.006	100.281	101.5606	3.159734	100.5289	1.338576	3.153734	0.247947
116	-100.088	50.018	100.285	-98.876	48.25612	100.052	1.212037	-1.76188	-0.23304
117	-50.188	50.034	100.291	-49.9542	53.22892	99.87957	0.233794	3.194921	-0.41143
118	0.144	50.211	100.3	0.667847	57.14577	100.3403	0.523847	6.934765	0.040305
119	50.215	50.213	100.005	50.74702	51.95597	100.3247	0.532017	1.742965	0.319694
120	100.151	50.232	100.061	101.3987	52.67482	99.31819	1.247668	2.442821	-0.74281
121	-100.185	100.241	100.051	-98.5897	94.99413	99.83969	1.595282	-5.24687	-0.21131
122	-50.191	100.231	100.034	-49.0517	99.97978	99.63429	1.139326	-0.25122	-0.39971
123	0.042	100.211	100.056	0.843117	103.8973	100.0832	0.801117	3.686346	0.027203
124	50.148	100.211	100.057	50.98505	106.3973	100.4755	0.837052	6.186267	0.418453
125	100.035	100.231	100.063	101.4736	99.42675	100.3968	1.43858	-0.80425	0.333822

Table 4.3 Comparison between reference positions and computed positions of Test3

Posi-tion	Reference Position x-axis	Reference Position y-axis	Reference Position z-axis	Computed Position x-axis	Computed Position y-axis	Computed Position z-axis	Error x-axis	Error y-axis	Error z-axis
1	-100.056	-100.127	-100.147	-98.1833	-101.593	-98.3281	1.872731	-1.46568	1.818929
2	-50.177	-100.117	-100.15	-49.2773	-96.7766	-99.7175	0.899742	3.340433	0.432524
3	0.07	-100.12	-100.152	1.05632	-98.9753	-100.05	0.98632	1.14473	0.101832
4	50.274	-100.185	-100.167	51.33181	-102.834	-100.128	1.057807	-2.64886	0.039122
5	100.284	-100.225	-100.163	102.0459	-98.4046	-100.463	1.761875	1.82038	-0.29986
6	-100.402	-50.056	-100.206	-100.113	-48.9563	-99.8225	0.28868	1.099749	0.383451
7	-50.286	-50.021	-100.212	-49.6652	-54.468	-99.0895	0.620845	-4.44701	1.122474
8	0.113	-50.299	-100.252	0.036731	-51.9546	-100.122	-0.07627	-1.65555	0.129677
9	50.062	-50.31	-100.229	50.1619	-51.1416	-100.918	0.099896	-0.83163	-0.68928
10	100.05	-50.31	-100.219	100.5572	-51.4027	-101.001	0.50722	-1.09267	-0.78208
11	-100.223	0.179	-100.048	-100.759	-1.26409	-98.7283	-0.53637	-1.44309	1.319704
12	-50.162	0.184	-100.048	-51.1736	-2.46951	-99.1909	-1.01164	-2.65351	0.857134
13	0.149	0.196	-100.079	-1.11819	0.02146	-100.194	-1.26719	-0.17454	-0.11516
14	50.143	0.218	-100.071	49.3711	1.153818	-100.423	-0.7719	0.935818	-0.35213
15	100.177	0.23	-100.111	99.12199	-5.04013	-100.209	-1.05501	-5.27013	-0.09803
16	-100.08	50.182	-100.144	-102.593	50.143	-99.8051	-2.5133	-0.039	0.338856
17	-50.181	50.114	-100.089	-51.6871	48.64615	-99.6257	-1.50607	-1.46786	0.463273
18	0.024	50.114	-100.114	-1.55397	51.15695	-100.608	-1.57797	1.042954	-0.49351
19	50.057	50.119	-100.133	48.27506	45.70848	-100.515	-1.78194	-4.41052	-0.38212
20	100.106	50.17	-100.061	98.77474	45.67009	-100.543	-1.33126	-4.49991	-0.48193
21	-100.155	100.258	-100.054	-103.129	100.281	-99.4895	-2.97427	0.023025	0.564503
22	-50.19	100.244	-100.072	-53.6283	97.73387	-99.7799	-3.43828	-2.51013	0.292143
23	0.278	100.361	-100.082	-2.48109	100.4941	-100.14	-2.75909	0.133072	-0.05801
24	50.218	100.375	-100.142	47.07721	94.33469	-100.615	-3.14079	-6.04032	-0.47294
25	100.233	100.379	-100.134	97.30402	94.59597	-99.9929	-2.92898	-5.78304	0.141083
26	-100.066	-100.012	-50.134	-98.1996	-98.7795	-49.1167	1.866419	1.232528	1.017313
27	-50.06	-100.176	-50.134	-48.2947	-99.0285	-49.63	1.765332	1.147532	0.503953
28	0.081	-100.193	-50.117	1.768073	-101.656	-49.9125	1.687073	-1.46265	0.204526
29	50.053	-100.223	-50.113	52.34486	-101.136	-49.9842	2.291856	-0.91282	0.128768
30	100.043	-100.207	-50.124	102.0443	-102.137	-50.4176	2.001296	-1.93013	-0.29364
31	-100.106	-50.155	-50.108	-100.068	-46.6907	-49.367	0.038135	3.464336	0.741027
32	-50.144	-50.138	-50.111	-49.5255	-47.6251	-49.8266	0.618493	2.512855	0.284385
33	0.033	-50.139	-50.118	1.158423	-50.7089	-50.0711	1.125423	-0.56993	0.046879
34	50.003	-50.142	-50.112	51.19251	-50.3199	-50.6449	1.189511	-0.17792	-0.53292
35	100.146	-50.151	-50.115	101.4143	-51.1415	-50.5396	1.268255	-0.99049	-0.42464
36	-100.071	0.126	-50.112	-100.597	-0.38374	-49.1533	-0.52603	-0.50974	0.958702
37	-50.104	0.127	-50.115	-50.9618	3.668302	-50.6417	-0.85784	3.541302	-0.52671
38	0.078	0.126	-50.127	-0.32187	-5.5475	-49.7476	-0.39987	-5.6735	0.379422
39	50.097	0.125	-50.126	49.71983	-5.08702	-50.3178	-0.37717	-5.21202	-0.19178
40	100.217	0.122	-50.12	99.74434	-0.00333	-50.718	-0.47266	-0.12533	-0.59801
41	-100.154	50.124	-50.118	-100.942	43.93559	-48.9453	-0.78777	-6.18841	1.172724
42	-50.073	50.121	-50.137	-51.551	48.04285	-49.832	-1.47802	-2.07815	0.304973
43	0.004	50.122	-50.143	-1.56701	50.13064	-50.5429	-1.57101	0.008637	-0.3999
44	50.144	50.115	-50.148	48.77681	50.64196	-51.1116	-1.36719	0.526958	-0.96361

Table 4.3 Comparison between reference positions and computed positions of Test3 (Continued)

Posi-tion	Reference Position x-axis	Reference Position y-axis	Reference Position z-axis	Computed Position x-axis	Computed Position y-axis	Computed Position z-axis	Error x-axis	Error y-axis	Error z-axis
45	100.048	50.107	-50.148	98.35249	43.92936	-50.4123	-1.69551	-6.17764	-0.26435
46	-100.119	100.116	-50.173	-102.78	98.78103	-49.8876	-2.66071	-1.33497	0.285401
47	-50.042	100.116	-50.167	-52.1746	95.94756	-50.1845	-2.1326	-4.16844	-0.01754
48	0.13	100.114	-50.174	-2.48296	98.04472	-50.8801	-2.61296	-2.06928	-0.70612
49	50.031	100.111	-50.185	47.55699	98.63138	-51.444	-2.47401	-1.47962	-1.25896
50	100.04	100.111	-50.172	97.83528	98.37872	-51.282	-2.20472	-1.73228	-1.10997
51	-100.107	-100.138	0.1	-98.0658	-101.039	1.39579	2.041245	-0.90081	1.29579
52	-50.164	-100.146	0.089	-48.3079	-97.0413	0.251403	1.856076	3.104747	0.162403
53	0.085	-100.124	0.028	2.103786	-95.0713	0.264669	2.018786	5.052712	0.236669
54	50.159	-100.105	0.078	52.24326	-99.7745	0.079302	2.084262	0.330479	0.001302
55	100.063	-100.171	0.111	101.8806	-100.846	0.282667	1.817566	-0.67518	0.171667
56	-100.124	-50.024	0.052	-98.7491	-50.3684	1.285723	1.374915	-0.34444	1.233723
57	-50.149	-50.283	0.05	-49.489	-46.5282	0.13632	0.660001	3.754768	0.08632
58	0.122	-50.256	0.021	1.097604	-50.003	-0.13426	0.975604	0.252954	-0.15526
59	50	-50.259	0.043	51.0489	-49.6826	0.011144	1.048895	0.57637	-0.03186
60	100.002	-50.226	0.033	101.1444	-50.8859	-0.31296	1.142428	-0.65991	-0.34596
61	-100.178	0.196	0.048	-100.44	0.091598	0.850409	-0.26235	-0.1044	0.802409
62	-50.01	0.185	0.039	-49.8493	-1.73769	0.486582	0.160721	-1.92269	0.447582
63	0	0	0	0	0	0	0	0	0
64	50.198	0	0.291	50.20913	0.423175	0.171829	0.011128	0.423175	-0.11917
65	100.101	0	0.29	100.5682	-0.67874	-0.15868	0.467204	-0.67874	-0.44868
66	-100.24	50.082	0.264	-100.931	49.88258	1.210884	-0.69057	-0.19942	0.946884
67	-50.013	50.062	0.243	-51.3578	53.70299	0.632982	-1.34476	3.640991	0.389982
68	0.278	50.065	0.256	-0.78729	42.71739	0.045431	-1.06529	-7.34761	-0.21057
69	50.273	50.071	0.266	49.78123	49.22062	-0.61354	-0.49177	-0.85038	-0.87954
70	100.168	50.069	0.267	98.92542	48.54695	-0.33829	-1.24259	-1.52205	-0.60529
71	-100.023	100.247	0.284	-102.464	97.11888	0.632454	-2.44102	-3.12812	0.348454
72	-50.279	100.25	0.279	-52.0319	94.02663	0.318431	-1.7529	-6.22338	0.039431
73	0.214	100.249	0.31	-2.10865	103.1452	0.204586	-2.32265	2.896152	-0.10541
74	50.129	100.059	0.101	48.03828	103.4506	-0.19846	-2.09072	3.391557	-0.29946
75	100.105	100.061	0.113	98.35802	102.5857	-0.53893	-1.74699	2.524692	-0.65193
76	-100.177	-100.133	50.131	-96.8934	-104.185	50.86326	3.283562	-4.0516	0.732261
77	-50.17	-100.128	50.134	-47.4259	-100.316	50.5657	2.744109	-0.18822	0.431697
78	0.17	-100.125	50.146	2.698847	-98.5536	50.29	2.528847	1.571359	0.143996
79	50.018	-100.136	50.15	52.08197	-98.7573	50.03424	2.06397	1.378742	-0.11576
80	100.08	-100.135	50.146	102.6935	-100.168	49.80296	2.613547	-0.03331	-0.34304
81	-100.059	-50.027	50.128	-98.6685	-49.7069	50.81724	1.390531	0.320102	0.689236
82	-50.078	-50.027	50.143	-48.5004	-51.1003	50.46928	1.577561	-1.07326	0.32628
83	0.151	-50.033	50.127	2.10617	-49.558	50.18265	1.95517	0.475001	0.055648
84	50.134	-50.03	50.12	51.32291	-54.6552	49.88588	1.188912	-4.62521	-0.23412
85	100.19	-50.026	50.123	101.9401	-50.9183	49.69083	1.750082	-0.89229	-0.43217
86	-100.188	0.158	50.135	-100.282	-0.39503	50.66093	-0.09359	-0.55303	0.525926
87	-50.135	0.16	50.149	-49.8192	-2.01208	50.89271	0.315829	-2.17208	0.743714
88	0.025	0.162	50.142	-0.08018	-0.4913	50.03227	-0.10518	-0.6533	-0.10973

Table 4.3 Comparison between reference positions and computed positions of Test3 (Continued)

Posi-tion	Reference Position x-axis	Reference Position y-axis	Reference Position z-axis	Computed Position x-axis	Computed Position y-axis	Computed Position z-axis	Error x-axis	Error y-axis	Error z-axis
89	50.191	0.162	50.138	50.44285	-6.26057	49.71953	0.251852	-6.42257	-0.41847
90	100.176	0.161	50.138	100.1755	-1.54006	50.11169	-0.00047	-1.70106	-0.02631
91	-100.073	50.093	50.128	-100.667	48.2836	50.44223	-0.59377	-1.8094	0.314232
92	-50.109	50.095	50.133	-51.2613	51.9326	50.14003	-1.15229	1.837603	0.007032
93	0.168	50.089	50.143	-0.48173	47.6069	50.41474	-0.64973	-2.4821	0.271737
94	50.002	50.078	50.129	49.51366	47.52789	49.53969	-0.48834	-2.55011	-0.58931
95	100.11	50.09	50.127	99.63901	46.33988	49.2948	-0.47099	-3.75012	-0.83221
96	-100.148	100.005	50.131	-102.36	101.7306	50.84246	-2.21223	1.725578	0.711455
97	-50.188	100.006	50.139	-51.7916	98.40014	49.85006	-1.60359	-1.60586	-0.28894
98	0.178	100.173	50.138	-2.19741	100.0512	49.56275	-2.37541	-0.1218	-0.57525
99	50.073	100.172	50.144	48.25149	93.2637	49.24347	-1.82151	-6.9083	-0.90053
100	100.183	100.186	50.161	98.03875	92.54902	49.63719	-2.14425	-7.63698	-0.52381
101	-100.031	-100.239	100.233	-97.0722	-98.5371	100.8881	2.95883	1.701934	0.655111
102	-50.108	-100.243	100.207	-47.4995	-99.3588	100.5962	2.608466	0.884245	0.38919
103	0.182	-100.223	100.243	2.554559	-97.8563	99.99841	2.372559	2.366742	-0.24459
104	50.089	-100.218	100.242	52.94105	-97.8882	100.4064	2.852047	2.329778	0.164396
105	100.089	-100.264	100.233	102.3937	-99.64	99.72048	2.304693	0.623971	-0.51252
106	-100.246	-50.081	100.277	-98.6085	-49.7255	100.7363	1.637517	0.355522	0.459348
107	-50.033	-50.086	100.28	-48.5371	-51.1377	100.4099	1.495939	-1.05168	0.129941
108	0.112	-50.108	100.25	1.993514	-49.3286	100.3431	1.881514	0.779394	0.093101
109	50.06	-50.116	100.238	51.71681	-49.5465	100.2074	1.656812	0.569547	-0.03056
110	100.258	-50.097	100.252	101.5788	-51.188	99.4893	1.320818	-1.09098	-0.7627
111	-100	0.009	100.248	-98.9871	-1.1779	100.2288	1.01293	-1.1869	-0.01921
112	-50.044	0.008	100.254	-49.6965	2.847698	100.8048	0.347542	2.839698	0.5508
113	0.05	0.008	100.251	0.936177	-1.36364	100.3753	0.886177	-1.37164	0.124335
114	50.034	0.007	100.264	50.84231	-1.54865	99.65515	0.808309	-1.55565	-0.60885
115	100.222	0.006	100.281	100.8602	-3.01135	99.48966	0.638197	-3.01735	-0.79134
116	-100.088	50.018	100.285	-100.437	46.39489	100.5479	-0.34939	-3.62311	0.262892
117	-50.188	50.034	100.291	-50.0897	43.77327	99.5626	0.098327	-6.26073	-0.7284
118	0.144	50.211	100.3	-0.1976	51.98583	100.4323	-0.3416	1.774832	0.132333
119	50.215	50.213	100.005	49.85602	51.5315	99.65548	-0.35898	1.318495	-0.34952
120	100.151	50.232	100.061	100.1284	50.49929	99.50507	-0.02264	0.267289	-0.55594
121	-100.185	100.241	100.051	-102.019	98.67754	100.2688	-1.83363	-1.56346	0.217813
122	-50.191	100.231	100.034	-51.6918	95.37485	99.85349	-1.50078	-4.85615	-0.18051
123	0.042	100.211	100.056	-1.05609	96.97343	99.74164	-1.09809	-3.23757	-0.31436
124	50.148	100.211	100.057	48.60918	96.98994	99.59466	-1.53882	-3.22106	-0.46234
125	100.035	100.231	100.063	98.43625	95.77283	98.78957	-1.59875	-4.45817	-1.27343

Table 4.4 Translation errors

	X-axis (mm)				Y-axis (mm)				Z-axis (mm)			
	Max	Mean	SD	Median	Max	Mean	SD	Median	Max	Mean	SD	Median
Test 1	1.5219	0.4427	0.5565	0.3479	8.0569	2.4255	3.0142	2.0466	1.8470	0.5315	0.6497	0.4469
Test 2	1.5953	0.4577	0.4044	0.3720	7.4624	2.5876	3.1188	2.3918	1.2486	0.3624	0.3492	0.3338
Test 3	3.4383	1.3780	1.6295	1.3672	7.6370	2.1644	2.6822	1.5635	1.8189	0.4473	0.5686	0.3821

CHAPTER V
DISCUSSION

The implementations of the system in this thesis are discussed as follows.

1. Camera calibration procedure

In calibration procedure, we used the calibration target which was the checkerboard pattern plate. We captured the target images in different poses to correct the deformation of the camera lenses. However, we might have not captured the calibration target to exist in every part of the captured images conducting to imperfectly correct the lenses deformation.

2. Marker detection system using MATLAB®

The system implemented in MATLAB® has some advantages. For example, it has many toolboxes to support creating the application such as Image Acquisition Toolbox, Image Processing Toolbox. It is also appropriate to test the proposed algorithm. Yet, the trouble existed when we required the real-time application.

3. Markers

There were 2 types of markers that were tested to find the appropriate one for the system, passive marker and active marker.

- Passive marker based on retro-reflective sphere shaped markers needs the infrared light source to reflect the marker. In this system, we used a set of infrared-emitting diodes positioned between 2 cameras. If the marker moves too far from the illumination of the light source, we will not see the images of the marker, which will affect the detection system. Because we used three balls of passive markers that position in equilateral shape to form one marker set, a few problems arose. First, we cannot recognize the identity of the markers when they rotate above the longitudinal axis for every 120 degree. The captured images give the same characteristics. Another point is that there could be some possibilities that two markers are in the same line of sight leading to the partial occlusion. However, it has an advantage of the mobility in a defined range that suits the application that required small and light weight marker.

- Active markers in this system were designed to get rid of the passive marker's weakness. We made the marker from a set of three IR-LEDs that are separated at different distances to each other, which create unique identity for the orientation of the markers and seldom form occlusion. Also, the active markers clearly illuminate even though they are far from camera line of sight. Active markers support the system that needs more than one set of markers, which can control the status through the attached electronics system. This state also helps to identify which group marker belongs to. Furthermore, we used 2 sets of markers that are alternated lighting to represent the relationship between 2 devices that identify the positions and orientations to each other. They were calculated with the same algorithm, which encourage the reusability of coordinating subroutine. The 2 sets of markers share the same shape that could reduce the cost for developing the prototype.

4. Marker detection system using Microsoft Visual C++

The final components of the marker coordination system implemented on Microsoft Visual C++, captured frames from two cameras using Microsoft DirectShow, controlled 2 sets of wireless markers via serial port interface, performed image processing through the OpenCV Library, attained the 3D positions and orientations of markers by our own algorithm, and visualized the poses of markers via OpenGL. The results of the programs were able to track positions and orientation of the markers with satisfied error ranges. This system can be applied for the prototype of the surgical navigation system in the future.

5. The accumulative errors

The error study shows that the average errors in the X-axis and Z-axis or the normal plane to line of sight of camera are in the range of sub-millimeter and the average errors in the Y-axis (camera depth) are within 2.6 millimeters. From the experiments, the maximum errors of any axes always occur at the edge positions of the defined area. Other positions yield the satisfied outcome.

We tried to find the trend of the errors by finding the curve fitting of the errors in each direction. The best fittings are shown in Figure 5.1, Figure 5.2, and Figure 5.3 for errors in X-, Y-, and Z-axes, where the dots showed the errors at position from -100 to 100 in each axis. From Figure 5.1, the trend of error in X-axis from left to right position of camera changed linearly in the narrow range as illustrated

in the red line. The trend of error in Y-axis that represents the depth view of the camera changed linearly in the level of polynomial order 3 like the red line shown in Figure 5.2, where the positions in range -50 to 50 almost change in constant rate. And the trend of error in Z-axis from bottom to top position of camera also changed linearly as shown in Figure 5.3.

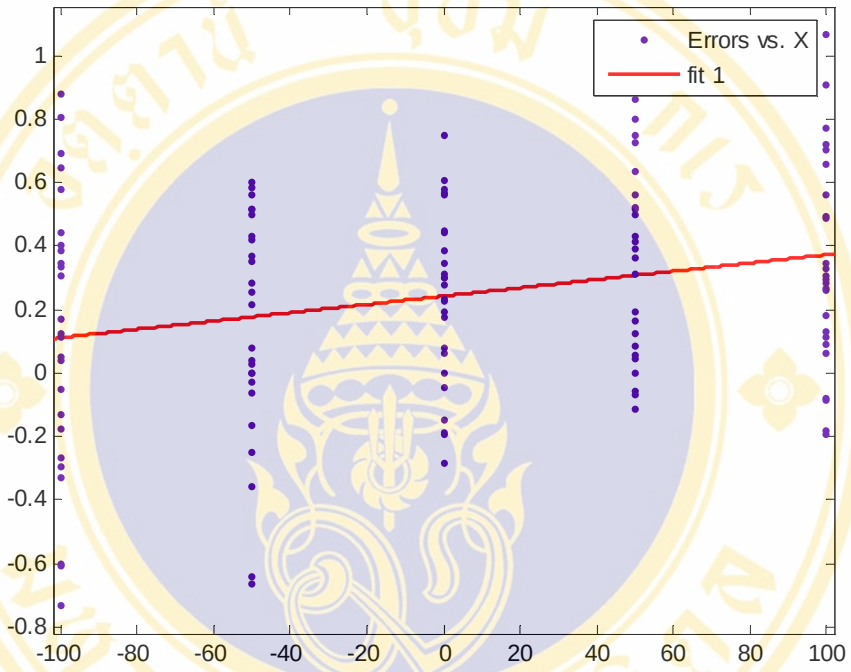


Figure 5.1 The fitting curve on the errors in X-axis

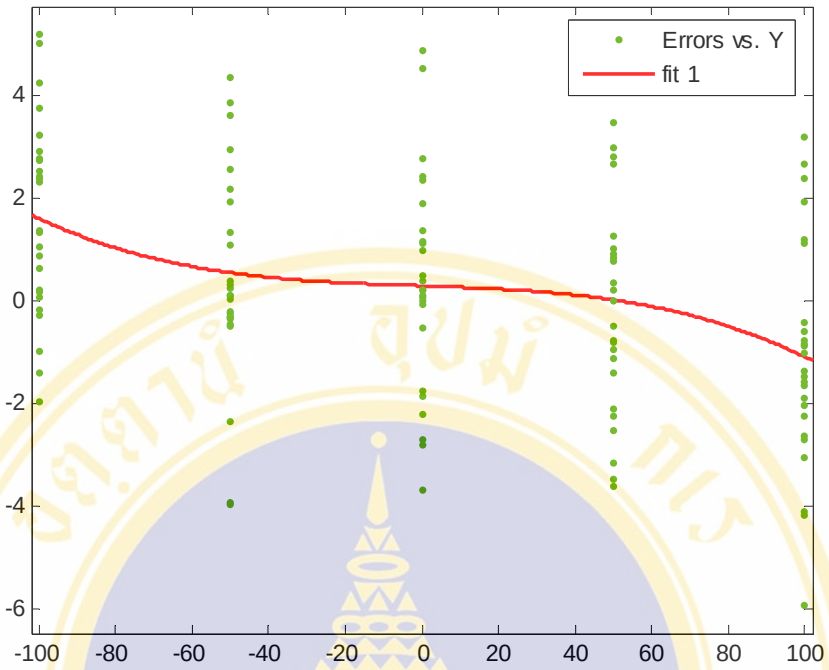


Figure 5.2 The fitting curve on the errors in Y-axis

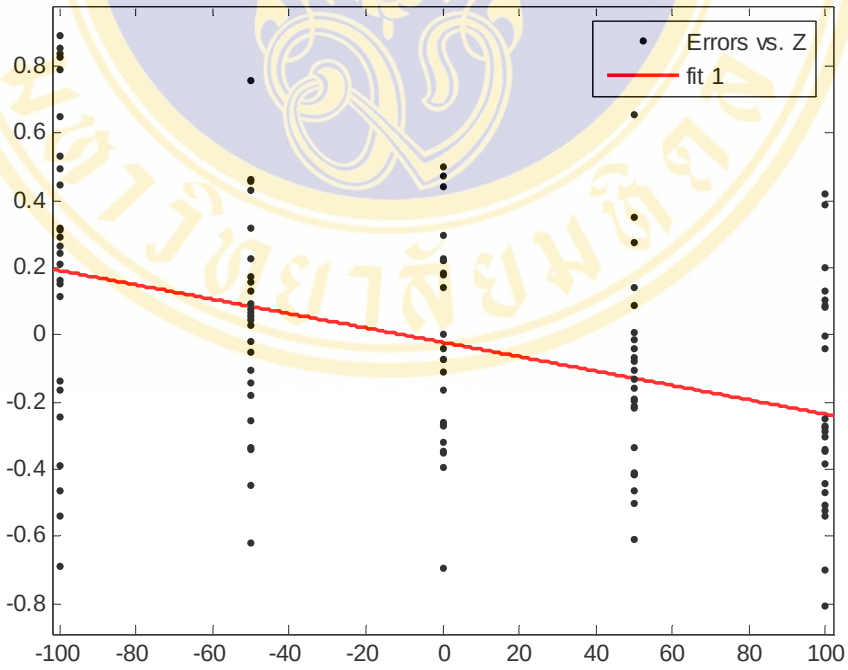


Figure 5.3 The fitting curve on the errors in X-axis

Errors of the system could be the accumulative errors from the calibration procedure conducting to incorrect triangulation for 3D positions also with the camera

configuration that can affect the line of sight of the camera, and the robot precision Moreover, marker centroids determination from the image processing algorithms is concerned as well. And for some marker orientations, the partial occlusion could occur. Marker identification may arrange the positions of markers in the wrong order because the ambiguous distances.

6. Limitations

The limitations of the system are about the setting up the cameras. Cameras after the calibration should be placed still in environments with dim light to avoid the calculation mistakes. The working area is upon the line of sight of the cameras. In our system, the working area was about 1m x 1m x 1m with 20 cm spacing between 2 cameras.

7. The marker representation

The transformation between frames leads to the positional relationship of markers and the interesting devices that they are represented through the rigid transformation. The applications can be the markers representing the distal holes of the intramedullary nail, the tip of the surgical device, or the organ of the patient with known relationship.

8. Future works

The system can be applied to be used for the surgical navigation system. Camera is one of the main factors that affect the accuracy of the coordination. Therefore, the specification of the camera with more resolution and automatic synchronization of 2 cameras would improve the system. Moreover, the algorithms should be more concerned with the various lighting conditions, the unique marker identifications despite the case of occlusion, as well as the precision.

In addition, to develop the system for the image-guided surgery, the next steps are to accurately specify the relationship of markers to the surgical devices and to assemble this information with the medical images by the registration to illustrate for 3D visualization. These procedures would be the main principles for the complete system.

CHAPTER VI
CONCLUSION

6.1 Background

Surgical navigation system helps surgeon to perform the operation accurately with more views. The important issue is to know the positions and orientations of the tools and the anatomy in three dimensions. To extract these factors, we chose the optical tracking to detect some markers that represent the medical devices.

6.2 Objectives

The main objective of this research is to find appropriate procedures and algorithms that are suitable for the implementation of surgical navigation applications. The mission is to develop a high precision 3D coordinate acquisition experimental platform that allows the measurement of position and orientation errors.

6.3 System Implementation

First, the system and algorithms were tested the feasibility using MATLAB®. Also, two types of markers, passive and active markers, were designed to suit the system. The type of markers that were later implemented, were wireless active markers. We used 2 sets of markers to represent the relationship between 2 devices. Next, we implemented the marker coordination system with Microsoft Visual C++ for real-time application. The 3D marker positions and orientations can be extracted and visualized in virtual reality.

6.4 Error Analysis

To verify the calculated 3D positions, we tested the system by attaching active markers to the manipulator that can be controlled the movement. The extracted 3D coordinates were calculated for relative distances to the reference point. The experimental results showed that our system had the accuracy in the X-axis and Z-axis, which had the average error in the range of sub-millimeter, and Y-axis had the average errors within 2.6 millimeters.

6.5 Suggested applications

Our system can be used to identify positions and orientation of some objects that cannot be easily measured if the object has a rigid relationship to the markers. The improved system with both quality of cameras and detection algorithm could be applied for the surgical navigation system.

REFERENCES

1. Hazan EJ and Joskowicz L, Computer-Assisted Image-Guided Intramedullary Nailing of Femoral Shaft Fractures, Techniques in Orthopaedics, 2003, Vol. 18(2).
2. Eyke JC et al., Computer-Assisted Virtual Fluoroscopy, The University of Pennsylvania Orthopaedic Journal, 2002, Vol. 15: 53–59.
3. Heikkila J, Geometric Camera Calibration Using Circular Control Points, IEEE Transactions on Pattern Analysis and Machine Intelligence, October 2000, Vol. 22, No. 10.
4. Tsai RY, A Versatile Camera Calibration Technique For High Accuracy 3D Machine Vision Metrology Using Off-The-Shelf TV Cameras And Lenses, IEEE J. Robotics Automat. , 1987, pp. 323-344, Vol. RA-3, No. 4.
5. Heikkila J and SilvCn O, A Four-step Camera Calibration Procedure with Implicit Image Correction
6. Abdel-Aziz YI and Karara HM, Direct linear transformation into object shape coordinates in close-range photogrammetry, In Proc. Symp. Close-Range Photogrammetry, Urbana, IL, 1971, pp 1-18.
7. Bacakoglu H and Kamel MS, A Three-Step Camera Calibration method, IEEE Transaction on Instrumentation and Measurement, 1997, Vol.46, No 5.
8. Hartley RI and Sturm P, Triangulation, in Proc. ARPA Image Understanding Workshop, 1994, pp. 957-966.
9. Iocchi L, Stereo Vision: Triangulation, Available:http://www.dis.uniroma1.it/~iocchi/stereo/triang.html
10. Chuckpaiwong I, Motion Detection of Marker, Final Report: Human Analysis for Medical Applications, 2006.
11. Dorfmuller K, Robust Tracking for Augmented Reality using Retroreflective Markers, Computers & Graphics , 1999, Vol. 23(6), pp. 795-800.
12. Polaris System, Available: http://www.ndigital.com/polaris.php
13. A.R.T. Infrared Optical Tracking System, Available: http://www.ar-tracking.de/

14. Neatpisarnvanit C, Wireless Active Marker System, Final Report: Human Analysis for Medical Applications, 2006.
15. PIC12F509 Microcontroller, Datasheet, Microship Technology Inc., Available: http://www.microship.com
16. Easy Link Wireless: TLP-434 and RLP-434 modules, Laipac Technology Inc., Available: http://www.laipac.com
17. West JB, Maurer CR, Designing optically tracked instruments for image-guided surgery, IEEE Transactions on Medical Imaging, 2004, Vol. 23, Issue 5, pp. 533 – 545.
18. Schwald B and Figueiredo PM, Learning of Rigid Point-Based Marker Models for Tracking with Stereo Camera Systems
19. Fisher R, Perkins S, Walker A, Wolfart E, Connected Components Labeling: Image Processing Learning Resources, 2004, Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/label.htm
20. Getting Started, Image Acquisition Toolbox, MATLAB® Help Document
21. Microsoft Window SDK, DirectShow, Available: http://windowssdk.msdn.microsoft.com/library/default.asp?url=/library/en-us/directshow/htm/introductiontodirectshowapplicationprogramming.asp
22. Open Source Computer Vision Library, Reference Manual, Intel Corporation, 2001.
23. Shreiner D, Woo M, Neider J,and Davis T, OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.1, Addisson Wesley Longman, Inc, 1997
24. Fernandez S, An introduction to OpenGL, Available: http://www.graphics.cornell.edu/~spf/opengl/
25. O'Sullivan C, Viewing Systems & OpenGL, Available: http://isg.cs.tcd.ie/cosulliv/4BA6/Viewing_6pp.pdf
26. Sepulveda MA, OpenGL Programming: The 3D Scene, Available: http://www.linuxfocus.org/English/May1998/article46.html
27. Craig JJ, Introduction to Robotics Mechanics and Control, Pearson Prentice Hall, 3rd Edition, 2005

28. Bouguet JY, Camera Calibration Toolbox for Matlab, Available: http://www.vision.caltech.edu/bouguetj/calib_doc/
29. Zhang Z, A Flexible New Technique for Camera Calibration Technical Report MSR-TR-98-71 Microsoft Research Microsoft Corporation One Microsoft Way Redmond, WA 98052
30. Williams AV, Computer Vision Course CMSC 828D: Fundamentals of Computer Vision 1112, Available: http://www.umiacs.umd.edu/~ramani/cmsc828.html
31. Naughter PJ, CSerialPort v1.03 - Serial Port Wrapper, A Freeware MFC Class for Win32 Serial Ports, Available: http://www.codeproject.com/system/cserialport.asp
32. MOTOMAN- Robot Controller NX 100, Available: http://www.yaskawa.co.jp/en/

BIOGRAPHY

NAME	Miss Teeraporn Kusalanukhun
DATE OF BIRTH	30 May 1981
PLACE OF BIRTH	Nakhonratchasima, Thailand
INSTUTIONS ATTENDED	Kasetsart University, 1998-2002: B.Eng. (Electrical Engineering) Mahidol University, 2002-2006: M.Eng. (Biomedical Engineering)
POSITION & OFFICE	Applied Research Laboratories of Biomedical and Robotics Technology (BaRT Lab), Faculty of Engineering, Mahidol University, Nakhonprathom, Thailand Position: Research Assistant Tel. 02-8892138
HOME ADDRESS	134 Grand Village, Ladprao Rd. Wangthonglang, Bangkok. Tel 01-3430797 E-mail: teeraku@hotmail.com