OPTICAL TRACKING FOR ARTHROSCOPIC SURGICAL TRAINING

A LOW-COST OPTICAL TRACKING SYSTEM FOR ARTHROSCOPIC SURGICAL TRAINING

By YUERU WANG, B.Eng.

A Thesis Submitted to the School of Graduate Studies in Partial Fulfilment of the Requirements for the Degree of Master of Applied Science in Engineering

McMaster University Master of Applied Science in Engineering (2020) Hamilton, Ontario

TITLE: A Low-Cost Optical Tracking System for Arthroscopic Surgical Training

AUTHOR: Yueru Wang, B.Eng.

SUPERVISOR: Dr. Gregory R. Wohl

NUMBER OF PAGES: xiii, 107

Lay Abstract

Arthroscopy is an orthopedic minimally invasive surgical procedure on the joint. Surgeons observe the interior of the joint through a miniature camera attached to the arthroscope, and a live X-ray image called fluoroscopy assists surgeons' manipulations simultaneously. However, arthroscopy is technically demanding and requires intensive practice, and fluoroscopy exposes surgeons under X-ray radiation. Therefore, the objective of the overall project is to develop a cost-effective surgical training station with no special safety procedures required for trainees.

The present work is to build a virtual fluoroscopy system without X-rays and track the surgical tool in realtime. Some passive markers attached to the surgical tool are tracked by Raspberry Pi cameras, then the tool is superimposed on the virtual fluoroscopic images reconstructed by a set of CT images of an artificial hip joint.

The results demonstrate the possibility of using Raspberry Pi cameras in a low-cost optical tracking system for surgical training purposes.

Abstract

The minimally invasive surgical procedure is more technically demanding than normal open-joint surgeries because of the limited vision. Thus, preoperative training for surgeons is essential. Current training for arthroscopy uses a fluoroscopy system, but that is costly, and the trainees will be at high risk under X-ray radiation exposure.

The purpose of the overall project is to design an affordable arthroscopic surgical training station and no special safety procedures for trainees. Our system combines a virtual imaging system (to replace fluoroscopy) with a physical synthetic model of a hip joint. The purpose of the current project is to develop a 3D visual tracking system using low-cost Raspberry Pi cameras and to test the resolution and accuracy.

Two Pi cameras were used to track markers on a surgical tool. The tracked data are intended to be used with a synthetic hip and superimposed on a CT dataset of the hip that can mimic surgery with real-time fluoroscopy. The reconstructed surgical tool can be overlaid on the virtual fluoroscopy to mimic the display in the real arthroscopic surgery. Pi cameras tracked passive coloured markers on a tool from different angles. The markers were tracked independently by colour segmentation, and positions were sent to a central computer simultaneously for 3D reconstruction.

The optical tracking system supports 25fps, 1080p live video streaming. The largest errors in the X, Y and Z-axis are 12.46±0.14, 8.55±0.3, 10.09±0.42 mm respectively while the repeatability is in a range from 0.61 to 5.17 mm.

These results demonstrate the possibility of using Raspberry Pi camera modules in a low-cost optical tracking system for surgical training purposes. Currently, the frame rate is low (25fps) and the error is still too large (up to 12.46mm) for use in surgical tracking. The resolution of the camera could improve when a better camera module is available.

Acknowledgement

I would like to express my deepest appreciation and heartfelt gratitude to my supervisor Dr. Gregory R. Wohl. I will never forget his support, guidance, and help throughout my degree. It would have been impossible to complete this project without his encouragement and suggestions. I want to thank Dr. Shahram Shirani for his kind support and help in this project. It has been a great pleasure to work with them.

Beyond my project, I am also thankful to everyone in the biomechanical engineering lab. They made the process of learning much easier. I want to thank Dr. Cheryl Quenneville for her kind support and encouragement. I also want to thank my lab mates Akiv, Jared, Hamad, Taylor and Elyse, and my colleagues Marisa, Julia, Fatemeh, Cooper, and Noah. It has been wonderful to work with them.

I am also grateful to my exceptional family. My parents are always supporting me and standing with me even though we are separated by the Pacific. A special thanks to my great friends Xiaotong Ding and Yixing Sun. Thank you for being such amazing friends and for giving me countless supports and encouragement. A deep thanks to my friend Sungha Kim, it was so nice to meet you and be a friend with you. You made my study and life in Canada much easier and I am forever grateful for our friendship.

Table of Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

$\boldsymbol{a_x, a_y, a_z}$: Scaling Factor in the X-axis, Y-axis, and Z-axis

$\boldsymbol{d_x}$, $\boldsymbol{d_y}$: Pixel Dimension Along X-axis and Y-axis

f: Focal Length

$\boldsymbol{K_n}$: N^{th} Radial Distortion Coefficient

Pc: A Point on the Image Plane Projected by A Point in the World Coordinate

$\boldsymbol{P_n}$: N^{th} Tangential Distortion Coefficient

$(\boldsymbol{t_x, t_y})$: Translation Values in the X-axis and Y-axis

$(\boldsymbol{u_0, v_0})$: The Origin of Pixel Coordinate

$\mathbf{U_L V_L}$: Image Plane of the Left Camera

$\mathbf{U_R V_R}$: Image Plane of the Right Camera

$(\boldsymbol{x_c, y_c})$: Centre of Distortion

$(\boldsymbol{x_c, y_c, z_c})$: A Point in the Camera Coordinate

$(\boldsymbol{x_d, y_d})$: Distorted Image Point

$(\boldsymbol{x_u, y_u})$: Undistorted Image Point

$(\boldsymbol{x_w, y_w, z_w})$: A Point in the World Coordinate

θ: Rotation Angle

2D: Two Dimensional

3D: Three Dimensional

CAD: Canadian Dollar

CPU: Central Processing Unit

CT: Computed Tomography

DICOM: Digital Imaging and Communications in Medicine

DLT: Direct Linear Transformation

FAI: Femoroacetabular Impingement

FDA: Food and Drug Administration

FOV: Field of View

FPS: Frame Per Second

HD: High Definition

HSV: Hue, Saturation, Value Colour Space

Hz: Hertz

IR: Infrared

LAN: Local Area Network

LED: Light-Emitting Diode

mGy: Milligray

min: Minute

mm: Millimeter

MRI: Magnetic Resonance Imaging

nm: Nanometer

rad: Radian

RGB: Red, Green, Blue Colour Space

SVD: Singular Value Decomposition

USB: Universal Serial Bus

UV: 2D Cartesian Image Coordinate

XYZ: 3D Cartesian World Coordinate

YUV: A Color Space Composed by One Luma Component and Two Chrominance Components

Declaration of Academic Achievement

I, Yueru Wang, completed the research in this thesis and performed the system design, programming by Python for optical tracking, testing, and data collection, process and analysis. Dr. Gregory Wohl supervised and guided this work. The virtual fluoroscopic system, described in Chapter 2, was based on the original concept by Zahra Hosseini (Hosseini 2013), and some updates were made to reconstruct the CT dataset using plugins in ImageJ.

Chapter 1. Introduction

1.1. General Background

1.1.1. Femoroacetabular Impingement

Femoroacetabular impingement (FAI) is a structural abnormality associated with the hip joint. A hip joint with irregular structure may cause more contact stresses at some specific points, and this increases the potential for joint damage around the hip (Leunig, Beaulé et al. 2009). This is because the hip joint is like a pair of precisely meshed gears, and any abnormalities of the bony acetabulum or severe hip motion may cause repetitive collisions and injure the soft tissue at the acetabular rim, leading to the hip failure gradually. A non-spherical proximal femur head or an excessive acetabular cover may cause the limitations of hip motion and result in FAI, and patients often reveal limitations on internal rotation and adduction in flexion (Ganz, Parvizi et al. 2003). However, FAI could also happen on patients who have excessive demands on the hip, even though their hip anatomic structures are normal.

FAI is one of the main causes of early hip osteoarthritis (Ganz, Parvizi et al. 2003, Leunig, Ranawat et al. 2011). Osteoarthritis is the most common form of arthritis, which is mainly associated with the breakdown of cartilage (Sodeman 1999, Centers for Disease Control and Prevention 2020). When the cartilage in the joint gradually deteriorates, there is less cushioning buffer, and more friction occurs between bones. Patients suffer from pain, tenderness, stiffness and inflammation with osteoarthritis, which severely affects their movements and daily lives.

Ganz et al. (Ganz, Parvizi et al. 2003) proposed two distinct patterns of femoroacetabular impingement. The first one is called cam impingement (Figure 1.1), which is caused by an abnormal femoral head adjacent to the acetabular rim during vigorous motions like flexion. This type of FAI results in chondral abrasion and detachment of the acetabular labrum. The other pattern is the pincer impingement (Figure

1.2). In this case, there is a linear contact between the acetabular rim and femoral head-neck junction. Unlike cam impingement, this impingement is the result of an abnormal acetabulum while the femoral head may have normal morphology and it is one of the origins of chondral injury in the posteroinferior acetabulum.

Ganz et al. also described the treatment of femoroacetabular impingement, which is to modify the abnormal morphology of the hip joint by surgery. The main surgical treatment for the cam impingement is to adjust the non-spherical femoral head and enhance the neck offset and subsequent clearance. While in pincer impingement, as acetabular overcover the femoral head, surgery helps remove the bony prominence at the rim. After the excision osteoplasty, surgeons need to check the impingement-free range of motion of the hip joint to evaluate if any residual impingement exists and any further osteoplasty is needed. This surgery is normally performed by arthroscopic surgery, a minimally invasive surgical procedure.

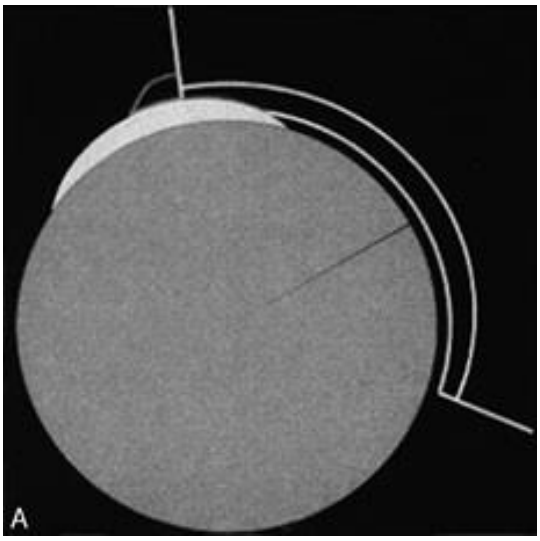


Figure 1.1 Cam Impingement

A schematic of the mechanism of cam impingement. The femoral head is nonspherical and abuts against the acetabular rim during hip flexion. (Ganz, Parvizi et al. 2003)

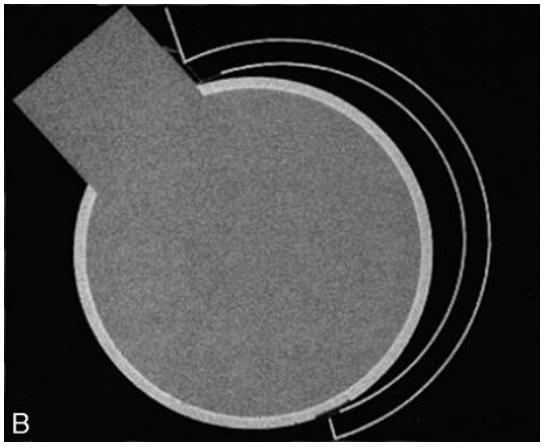


Figure 1.2 Pincer Impingement

A schematic shows pincer impingement. There is a linear contact between the acetabular rim and the femoral head-neck junction. (Ganz, Parvizi et al. 2003)

1.1.2. Arthroscopic Surgery

Arthroscopic surgeries have become more common in orthopedic treatment in recent years. Like all other minimally invasive surgical techniques, arthroscopic surgery has features such as less trauma, reduced pain, and quicker recovery. The arthroscopic surgery was first introduced in hip treatments by Burman in 1931 (Burman 1931). However, because of anatomic constraints and greater surrounding soft tissue, the development of hip arthroscopic techniques was relatively slow comparing to arthroscopic surgery on the shoulder or knee. The hip joint is a ball-and-socket joint (Figure 1.3), and the ball-shape femur head fits deeply into the acetabulum, which makes sturdy conjunction between the leg and pelvis. This stable anatomic structure causes more difficulties to visualize with an arthroscope compared to other joints such as shoulder and knee. Also, as the second-largest joint in the human body, the weight that hip joints bearing during a walk can be several times the body weight and even higher during running (Van Den Bogert, Read et al. 1999, Giarmatzis, Jonkers et al. 2015). Thus, there are strong and large muscles as well as thick fibrocaspule that envelopes the hip joint to support the hip motions and functions (Kelly,

Williams et al. 2003). This morphologic feature reduces the allowed amount of hip distension and makes it difficult to pull the hip joint apart. Another limitation comes from the complicated and adjacent nerves and neurovascular structures, such as the sciatic nerve and cutaneous nerve, which brings more challenges to portal placement.

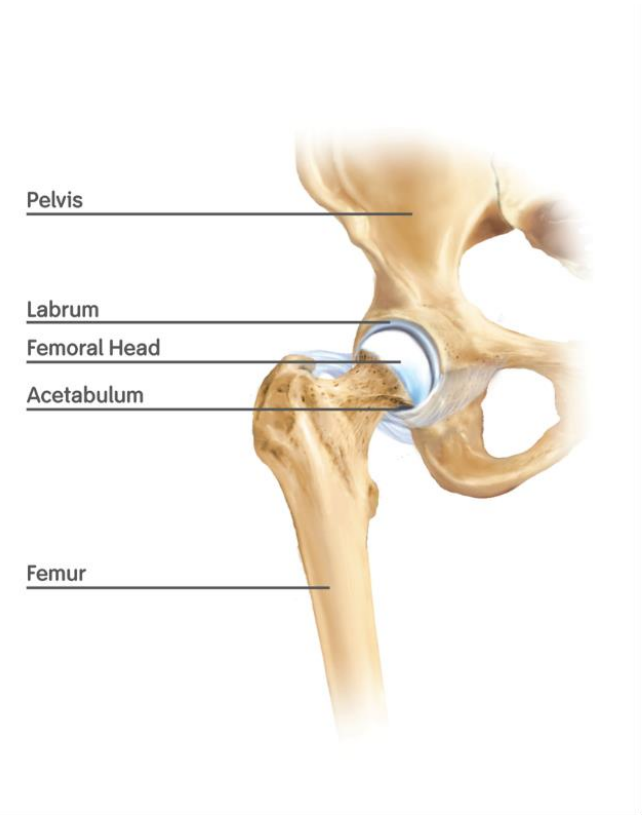


Figure 1.3 The Ball-Socket Structure of Hip Joint

(Nephew 2011)

Hip arthroscopy is an effective treatment for some athletic injuries including labral tears, lateral impact injury, loose bodies, etc. (Kelly, Williams et al. 2003). Problems such as management of osteonecrosis of the femoral head and different types of joint abnormalities can also be treated by hip arthroscopy. With the advent of specifically designed hip arthroscopic instruments and adaptations of more flexible scopes

on arthroscopy equipment, surgeons can visualize the joint and articular surface clearly in real-time with fewer constraints caused by anatomic structures of the hip. The long cannula or flexible probe helps penetrate the thick soft tissues around the hip joint and reach to the joint (Figure 1.4). There are three fundamental positions (anterior, anterolateral, and posterolateral) to place the surgical instrument in hip arthroscopy, and they are treated as optimal positions to visualize and penetrate in the hip joint safely. The patient's positioning can be either supine or lateral, and the operative leg is supposed to be in traction and distracted and under the direct fluoroscopy in the meantime (Kinnaman and Mabrey 2006).

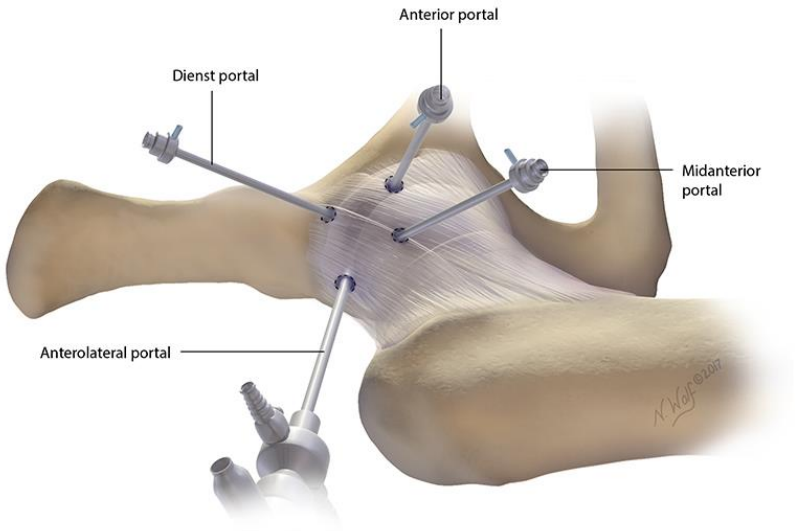


Figure 1.4 Puncture Capsulotomy

Showing anterolateral (viewing), anterior, mid-anterior and dienst portal [15]

For the surgical treatment of femoroacetabular impingement, arthroscopy helps inspect the severity of disease preoperatively and the interior of joint during the surgery. The portal for placing cannulated hip arthroscopy cannulas locate in anterior, anterolateral or posterolateral around the hip joint, which assists

the surgeon to check the articular cartilage of the femoral head, the acetabulum and the acetabular labrum (Clohisy and McClure 2005).

Researchers compared the arthroscopic surgery with the traditional open incisive surgery to evaluate its performance. Sussmann et al. (Sussmann, Ranawat et al. 2007) compared the accuracy and precision of arthroscopy to open osteoplasty and confirmed that for the head-neck junction for isolated cam impingement, the precision and accuracy of arthroscopic compression showed no difference with the open surgical technique. After a follow-up of 88 patients taking surgeries for the symptomatic cam or mixed femoroacetabular impingement between 2004 and 2007, Horisberger et al. (Horisberger, Brunner et al. 2010) concluded that the outcome of arthroscopic treatment for femoroacetabular impingement is comparable to open-incisive treatments and found postoperative improvements from arthroscopic therapy.

1.1.3. Minimally Invasive Surgeries

The minimally invasive procedure is a promising and rapidly developing surgical procedure. Instead of cutting and detaching the skin and musculature around the joint, a small incision can help the patient recovery in a shorter time, and the postoperative pain and complications will be reduced during recovery (Horisberger, Brunner et al. 2010). On the other hand, small incisions also cause vision restriction, less mobility of surgical instruments and counter-intuitive hand-eye coordination, which may increase the risk of unexpected injuries to patients.

The surgical instruments are specially designed for its high technical demand as the surgeons can only see the surgical region of interest by using X-ray fluoroscopy and images from the endoscope (a small camera) inside the body. This requires the system to provide unique visual feedback from an endoscope view and muted haptic feedback. The endoscope offers a 2D view of the interior of the body within the field of view. With the assistance of fluoroscopy, surgeons can conceptualize the surgical tool position

relative to the joint, while it is different from what the surgeons see from a direct view with the open incision.

Minimally invasive surgery is more technically demanding than normal open-incision surgeries because it is difficult for surgeons to build up the sensation of the operative region. Also, surgeons are required to manipulate the arthroscopic surgical instruments accurately and proficiently, which may be different from what they practiced with the traditional open-incision surgery training (Kühnapfel, Cakmak et al. 2000). This added skill for arthroscopic surgery needs additional training.

1.1.4. Fluoroscopy in Arthroscopy

Fluoroscopy is a type of medical imaging that displays successive (real-time) X-ray images on a monitor. In arthroscopic surgeries, surgeons usually track the position of the surgical instrument coordinating with the fluoroscopy in real-time. X-ray fluoroscopy can also provide a patient's anatomical information without causing an incision on their skin.

The X-ray beam passes through or is absorbed by tissue in the human body depending on the tissue it encounters. Because different tissue has a different density, their ability to absorb the X-ray beam varies. In this case, the level of X-ray absorption can reflect the corresponding tissue, which is the 2D X-ray images. X-ray fluoroscopy has advantages of fast speed and good performance on demonstrating rigid bones, which is ideal for arthroscopic surgeries.

1.1.5. Surgical Simulation and Training

Practice is essential for surgeons' training. In traditional surgical education, trainees are instructed by experienced surgeons through clinical involvement and build up their operative skills gradually

(Ballantyne, Marescaux et al. 2004). However, this training methodology is challenged by growing concerns about patients' safety, work-hours restriction, increasing cost on health care and operating room, and more surgical complications from both ethical and legal perspectives (Roberts, Bell et al. 2006). Surgical simulations can be an alternative method to train surgeons regardless of the restrictions from cost, time and ethical concerns. Therefore, it can create less expensive opportunities for surgeons to practice before they operate on real humans, which can help build up confidence both for surgeons and patients.

With the development of technologies, more concepts such as minimally invasive surgeries and arthroscopy are introduced to the surgical community and make an evolution on surgical simulations. These novel techniques require fully different skills comparing to traditional open-incisive surgeries (Figert, Park et al. 2001, Peters, Fried et al. 2004) and high demand for eye-hand cooperation and the ability to manipulate precise operations in a 3-dimensional space under the guidance of a two-dimension screen (Wanzel, Hamstra et al. 2003).

A surgical simulation system needs to be consistent with real surgery. To duplicate a real orthopedic surgery environment, an orthopedic simulator usually consists of a visual system to navigate the surgical tool, a synthetic bone or a cadaver model to provide haptic feedback and software to coordinate the system and evaluate users' manipulations (Vankipuram, Kahol et al. 2010).

Surgical simulators help trainees develop their cognitive, clinical, and technical skills before they practice on living patients. Trainees can practice on synthetic bones or cadaver models under the assistance of a simulator in a virtual environment (Escobar-Castillejos, Noguez et al. 2016). Haptic simulators can provide visual and tactile feedback or even the sensation of shape and texture to users to mimic the real surgical operations (Tsai, Hsieh et al. 2007). Synthetic bones have advantages including consistent size, shape and density and are also possible to be modified into almost any form (Hausmann 2006). In some cases, simulators can generate scenarios of various complexity based on the applications (Lateef 2010).

For novel surgeons, the simulator can provide a more complex scenario when the prior level of standard is satisfied (Agha and Fowler 2015). Moreover, there is no requirement for ethics committee approval or storage techniques before using them (Hausmann 2006). Surgical simulations in orthopedic training can nearly merge all tasks such as the skin incision, placement of the sliding hip screw, positioning of the C-arm, etc.(Atesok, Mabrey et al. 2012). Besides, surgical simulation also plays a vital role in preoperative planning. With the assistance of simulation systems, surgeons can reconstruct the digital model of targeting surgical site. For example, new trainees or orthopedic surgeons who have few experiences can conceptualize acetabulum fractures and get familiar with the operations procedures preoperatively (Cimerman and Kristan 2007). Computer software can simulate all steps in real surgery under a virtual three-dimensional environment, for example, rendering CT or MRI images to reconstruct a virtual three-dimensional model of the region of interest.

Research also proved the significance of surgical simulations in training non-experienced orthopedic surgeons and helping them acquire necessary skills before they operate on patients. For example, Chetan et al. (Modi, Morris et al. 2010) concluded that the arthroscopy computer simulators can help improve the skill levels for inexperienced participants. Howells et al. (Howells, Gill et al. 2008) also found the simulator-trained group had a considerable improvement in operating performance compared to the untrained group.

1.1.6. Tracking of Surgical Tools

According to the tracking method, tracking systems are commonly classified into mechanical tracking systems, electromagnetic tracking systems and optical tracking systems (Hosseini 2013).

One of the patterns of mechanical tracking systems is a mechanical arm, which consists of joints and links. Joints connect the separate links so that the mechanical arm can move freely. An encoder placed at

each joint measures the relative motions between the connected links, and the movement of the endpoint of the arm can be calculated by the cumulated movements of each link. The mechanical tracking systems are reliable and accurate, but they are usually bulky which limits the applications in most surgery scenarios (Hosseini 2013).

Electromagnetic tracking systems use electromagnetic signals generated from the field generator, and the sensors on the target can detect these signals (Glossop 2009). Although it detects the target directly and the sensors are small, metal in the workspace is a problem as it can distort measurements and cause errors.

In orthopedic surgeries, optical tracking is much dominant (Glossop 2009). The optical tracking system applies localization technology to monitor a defined measurement space. Cameras are always used to capture visual information and complete tracking tasks in real-time. Single camera in the system can only track in a two-dimensional view while 3D localization requires at least two cameras to provide visual information from different perspectives. Cameras are usually fixed, and computer vision technologies would utilize their positions, optical parameters and object 2D positions on the image coordinates to triangulate the 3D poses of the target.

An optical tracking system can mainly be classified into two types based on the target. In passive optical tracking systems, special markers coated with retroreflective material can reflect lights from the surroundings. Usually, the light reflected by the markers is generated near the camera lens. Depending on the type of light, the camera threshold filters all lights except lights reflected by markers. For example, OptiTrack Flex 13 camera (NaturalPoint, Inc. DBA OptiTrack, Corvallis, OR) emits 850 nm infrared lights from its LED ring around the lens, while the 800 nm IR long pass filter on the lens only allows light for which wavelength is larger than 800 nm to pass. In this case, only the surface can reflect lights in this range that can be detected, and all other surroundings are treated as non-reflective and be shown as black. The centroid of the marker is extracted as a representation of its position on the camera's 2D view. The passive optical tracking system helps the user to get rid of the restrictions from cables or power supply.

The other type of optical tracking is active tracking. Rather than reflecting lights, the active markers transmit light (e.g. LED lights) by itself and can be detected by the sensor (e.g. camera). Even though the active marker requires cable to generate light, it has advantages including increasing the capture volume and reducing the marker jitter.

Current commercial optical tracking systems are expensive. A set of OptiTrack optical tracking system (NaturalPoint, Inc. DBA OptiTrack, Corvallis, OR) costs approximately CAD 6, 500 while the costs of NDI optical system which is widely used in medical fields vary from CAD 11, 500 (passive optical tracking system) to CAD 20, 000 (active and passive optical tracking system)(Hosseini 2013). In comparison, USB cameras are relatively inexpensive, and camera and lens technology have improved significantly in the last decade. High precision is critical in a surgical suite during tracking of surgical tools and for surgical planning of real surgeries, but lower precision cameras could be useful for training purposes if they have suitable precision. The purpose of my project was to develop a 3D tracking system using an inexpensive camera system (Raspberry Pi USB cameras) and to assess the precision and repeatability of the tracking system to see if it can be used in the proposed surgical training system.

1.2. Objectives of Project

The overall goal of my project is to develop a low-cost optical system to track surgical instruments for arthroscopic surgery training. Current arthroscopy training sometimes uses a fluoroscopy system, but that is costly and less efficient because only one trainee can use the system at one time (Hosseini 2013). Moreover, the trainees will be at high risk under X-ray radiation exposure, which could harm their health (Silverman, Tuncali et al. 1999, Hafez, Smith et al. 2005).

The purpose of the overall project is to design an arthroscopic surgical training station with an affordable price and no special safety procedures for the trainees. The system combines a virtual imaging system

with a physical synthetic model of the hip joint. The position of the surgical tool needs to be tracked by the system in a 3-dimensional space, which is a cube with approximately 50 cm sides. The physical synthetic model of a hip joint (including artificial skin and muscle tissue) will be placed into this space and associated with the tracking system. A 3D CT dataset of the artificial hip joint has already been acquired and was used to set up views that mimic a fluoroscopic view of the joint. The trainee can visualize a projected 2D image of the artificial bone on a monitor like a fluoroscopy system.

The main objective of my project was to track the position of the surgical tools used by the surgeon and overlay the position of the tools in real-time on the 2D projected artificial-fluoroscopy image on the monitor to mimic the real surgery process. In the end, the training station is meant to create a similar environment with arthroscopic surgery to visualize and project images of the joint from any angle. Simultaneously, the physical artificial joint model can provide real tactile feedback when the surgeon is cutting or drilling it. This set-up requires that the physical artificial joint is co-registered in space with the tool tracking system so that the surgical tool can be reliably and precisely overlaid on the projected artificial fluoroscopy images.

I designed the tracking system to track the position of the arthroscopic surgical tool in the determined 3-dimensional workspace with the synthetic hip joint model. The task of the tracking system is to obtain the real-time position and orientation of the surgical tool based on a fixed coordinate system with high accuracy. While we could use a commercially available motion tracking system, they are very expensive (> $10K) relative to the scope of the project. With the attached passive reflective markers on the surgical instrument, I captured the motions of the instrument in the workspace using two low-cost Raspberry Pi cameras. Each surgical tool will be mounted with at least three markers, and the software then generates a rigid body based on the positions of these markers to represent the surgical tool in 3D space. By obtaining these real-time position and orientation information, an interface was designed to show the surgical tool in the real-time integration with a two-dimensional virtual fluoroscopic image shown on the monitor for trainees to use.

1.3. Thesis Overviews

This thesis includes the following contents:

Chapter 2 introduces the theory and process of reconstructing the virtual fluoroscopy system;

Chapter 3 explains the general concepts of this optical tracking system, covering the hardware design and camera selection. The use of OptiTrack motion capture system is also described in this chapter;

Chapter 4 discusses the workflow of this system and explains the working principle of each part;

Chapter 5 demonstrates the tracking result collected by this system;

Chapter 6 analyzes the performance of this system and provides conclusions and future work for this project;

Chapter 7 is the conclusion of the thesis.

Chapter 2. Virtual Fluoroscopy System

2.1. Types of Surgical Simulation

Orthopedic surgical training meets more challenges because of more restrictions on the work hour (Frank, Erickson et al. 2014). According to the regulation released in 2003, residents should work no more than 80 hours a week by considering resident fatigue. Other restrictions from ethical issues and expense concerns also reduce the chance of practice and training for residents. Simulator-based surgical training is an alternative for trainees to acquire the necessary skills. Currently, there are three main types of model can be used in surgical simulations: cadavers, physical models and virtual models.

2.1.1. Cadavers

Cadavers or fresh tissue-based surgical training offer maximal surgical simulation because of its high fidelity to live patients. Weber et al. (Weber, Leland et al. 2017) confirmed the use of fresh tissue in preoperative surgical rehearsal improved resident confidence and perception of technique. However, the applications of cadavers for training purposes are restricted by its high cost, shortage of availability and ethical concerns. Besides, it requires special storage conditions and can only be used once. Moreover, cadavers do not always exhibit the pathology under consideration, and some diseases (e.g., osteoarthritis) can significantly alter the tissue structure. Therefore, an artificial model such as physical models or virtual models is more commonly used in residents' education.

2.1.2. Physical Models

Compared to the cadaver model, physical models and virtual models have benefits including repetitive use, training can be performed everywhere even outside of the operating room and regardless of the limitation by space, and easy assessment of trainees' surgical technique (Hammond and Karthigasu 2006). Physical models are supposed to keep consistency with live patients on their physical properties. For instance, it should tightly approximate the behaviour of tissues when being cut or stretched, and soft tissue can be deformed while bones should behave relatively rigidly under force. Furthermore, the model should fail or respond in the same way as the real tissue under the cutting or drilling process. Although there is still much room for improvement in fidelity, physical models can meet most of the needs for training novel surgical residents.

2.1.3. Virtual Models

Virtual models using virtual reality can have better adaptation than physical models. They can be more flexible on customizations for different patient anatomical structures and pathologies. They also allow users to interrupt at any time for correction and repetition. However, it is still a challenge for virtual models to incorporate realistic haptic feedback. Moreover, the high initial setup cost causes a barrier to some programs (Sharma and Horgan 2012). To provide real haptic feedback to users, a virtual-patient model should generate the haptic feedback that is consistent with live patients. Thus, the virtual soft tissue should have the same texture with live tissue and should deform with contact pressure.

Also, simulators should represent the region of interest equally with intraoperative views. For example, in arthroscopic surgery, orthopedic surgeons use fluoroscopy to track the position of the surgical instrument in the patient's joint, so what they can see intraoperatively is the surgical tool projecting on the bone from

a certain perspective by a real-time X-ray image (Figure 2.1). Thus, the simulator for arthroscopic surgery should also mimic the performance of fluoroscopy.

Figure 2.1 Fluoroscopic Image of a Cannula Inserted into the Hip Joint

(Abramsa, Harrisb et al. 2013)

2.2. Justification for An Inexpensive Physical System with Virtual Fluoroscopy

Fluoroscopy is an essential component in arthroscopy. It assists in the assessment of the patient preoperatively, and it also guides surgeons to determine portal placement and bony anatomy during the surgery (Larson and Wulf 2009). Additionally, it is useful intraoperatively for determining the amount of traction for appropriate hip distraction and cam and pincer correction. A fluoroscopic C-arm is an X-ray image intensifier generating continuous 3D real-time X-ray images (Figure 2.2), and it can move along the longitudinal axis and rotate with the transverse axis when the patient laying on the C-arm table. It can be used to track a surgical procedure, inspect anatomic structures of patients, and visualize the position of

implants, etc. (Bott, Dresing et al. 2011). A C-arm provides CT-like images in various inspection planes, as it generates an intact volumetric dataset of the patient by a single rotation of the X-ray source and detector (Orth, Wallace et al. 2008). It is beneficial for acquiring a large anatomic structure of interest. As the surgeons can only see the 2D projections of the region of interest from the screen, it is especially helpful for acquiring X-ray images from different perspectives. Taking femoroacetabular impingement (FAI) as an example, a 180-degree C-arm assessment of femoral head-neck junction can help with a better conceptualization of the cam lesion in a dynamic pattern. Furthermore, fluoroscopy also supports optimal decision making on the positioning of cam decompression. During the cam-type femoral osteoplasty, visualization by fluoroscopy plays an essential role to guarantee restoration of the sphericity to femoral head-neck junction or sufficient decompression of the cam lesion (Larson and Wulf 2009, Lee and Clark 2011).

Figure 2.2 Siemens Cios Fusion C-arm

(Siemens Healthineers, Erlangen, Germany)

However, the use of X-ray in fluoroscopy exposes surgeons and patients under continuing radiation. The Center for Devices and Radiological Health of the Food and Drug Administration (FDA) released an advisory in 1994 due to the potential for serious radiation-induced skin injury caused by constant fluoroscopy use (US Food Drug Administration 1994). The typical fluoroscopic entrance exposure rate for a medium-sized adult patient is approximately 30 mGy/min (3 rad/min), and it can be higher in image-recording modes. According to FAD, too much radiation could lead to skin disease including cancer, burns and teratogenic effect after long time use. Even worse, safety measures such as keeping the surgeons and patients away from the fluoroscopy or wearing protective equipment make it cumbersome for surgeons to manipulate in the technically demanding surgeries like hip arthroscopy. Moreover, unlike patients who just need to undergo fluoroscopy during the treatment, surgeons have a much greater chance to be exposed to radiation due to multiple exposures with subsequent surgeries. For training purposes, it is risky and unnecessary to expose surgeons to such radiation, so we need a replacement for the fluoroscopy system that does not require X-rays or other radiographic technologies to track the position of the surgical instrument during the surgical practice.

Unlike comprehensive surgery simulators, the application scenario for our training system only focuses on arthroscopic surgery, so the orthopedic trainers are looking for a cost-efficient arthroscopic training system without radiation effect on the trainees.

2.3. Artificial Hip

The intended simulator requires a physical patient model for training so it can be repeatedly used, and some parts of the model are replaceable to mimic different pathologies. Sawbones (Pacific Research Laboratories Inc.) or other composite bone models have similar physical properties and geometry to real bones and provide haptic feedback when the trainees drill or cut them. Therefore, an artificial synthetic hip joint can be employed for a realistic representation of the region of interest in the surgery and surgical

trainees can practice their arthroscopic skills on it. This simulation model also helps increase the chance of rehearsals for the novel surgical residents as the medical education institutes can afford multiple systems at once rather than making the trainees wait in a long line for the availability of relatively expensive virtual simulators. A hip joint model manufactured by DePuy Synthes Mitek Sports Medicine Inc. (Hip in a Box) (Figure 2.3) was provided as a practice model. The current tracking system was developed and tested in accordance with this model. The purpose of my project was to test the performance of our system and calibrate the tracked tool with the global coordinate based on the artificial hip model. However, the design idea fits general situations, and it is adjustable if the physical model is replaced by another model in future work.

Figure 2.3 Hip in a Box

(DePuy Synthes Mitek Sports Medicin Inc.)

2.4. CT Images Reconstruction

As we are not using X-ray imaging in the training, we need to generate a virtual fluoroscopy image of the synthetic hip joint. A fluoroscopy system can be mimicked by using CT data of the physical artificial

joint. The CT dataset of the DePuy Synthes Mitek artificial hip joint was obtained by Zahra Hosseini (Hosseini 2013) by using a Philips scanner (0.24-by-0.24 mm pixel spacing at a slice thickness of 1 mm).

When imaging with X-rays, the body tissues absorb parts of the energy of the X-ray beam; namely, the X-ray beam is attenuated. In the meantime, there is a detector or a film located on the other side of the tissue to capture the attenuated X-rays, which generates a medical image (Wehling 2015). Based on the density of different tissues, their sensitivity to radiation varies. Bones absorb more photons than soft tissues, which leads to the image demonstrating bones in lighter colours such as white, while tissue like lungs and fat are relatively darker on the film. The denser tissue is represented in brighter colours on X-ray images (Yang 2017). Unlike the conventional radiography that only generates a 2D image each time, in Computed Tomography (CT) both the X-ray generator and the detector rotate around the patient during the procedure and produce a stack of images to construct a 3D visualization.

Fluoroscopy is a real-time X-ray image that is capable of projecting images in realtime. The only difference between fluoroscopy and X-ray image is that the image is displayed on a fluorescent screen. The modern fluoroscopy allows the images to be played on a monitor after a series of digital intensifications and surgeons can examine the patient in live mode. Compared to standard X-ray images, fluoroscopy represents soft tissue in lighter intensity values while bones are represented by darker pixel intensity, so a fluoroscopic image looks like it is inverted (by intensity) compared to an X-ray image (Figure 2.4).

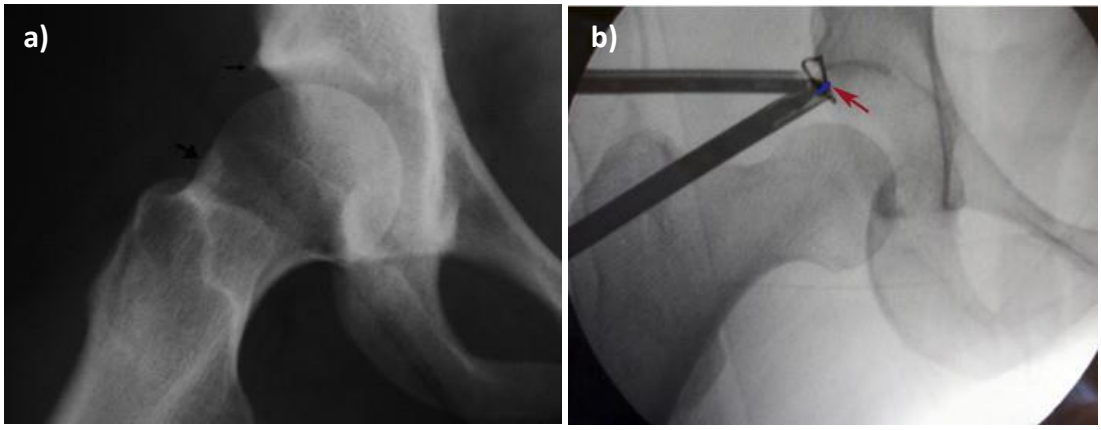


Figure 2.4 X-ray and Fluoroscopy

a) X-ray Image of Lateral View of Right Hip: Notice that bone is represented by bright intensity, and soft tissue and air are represented by darker shades (Matsuda 2009)

b) Fluoroscopy of Distracted Right Hip: Bone is represented by dark shades, while air and soft tissues appear as the brighter intensity (Matsuda 2009)

As we only have the CT dataset of the artificial hip joint, we need to convert the CT images to fluoroscopy-like images to replicate the real arthroscopic surgery scenario.

Standard 2D images can be represented by two means. The first one is called vector graphics while the second is raster graphics. Vector graphics use mathematical equations to express lines, curves and polygons, and they constitute all the different shapes in the vector graphics. Vector graphics can be scaled to any size and its quality can be consistent regardless of the scale variations. Raster graphics are also known as bitmaps because they are assembled by many tiny squares called pixels, and each pixel is in certain colour value. When the scale of bitmap changes, the quality of the image will be sacrificed. Usually, the more pixels in the image, the more fine and smoother design can be expressed, so more details can be stored and result in larger file sizes.

Similar to pixels in 2D raster graphics, a voxel is the unit of 3D digital imaging. In the 3D CT dataset, the elements of the image are voxels. Voxels that have all equal dimensions (e.g., cubic) are considered as 'isotropic'. However, typically the shape of voxels for clinical CT data is not cubic, and they are considered as 'anisotropic'.

Digital Imaging and Communications in Medicine (DICOM) are an international standard of medical imaging and our CT dataset of the artificial hip joint is composed of a sequence of 2D DICOM images. The length and width of the voxel are the same as the pixel size in each 2D image, while the height of the voxel is equal to the distance between two slices. An anisotropic voxel has identical dimensions within the plane of the slice, but the dimension between slices is longer.

Usually, a spiral CT scanner can produce an isotropic dataset. However, sometimes to fully render the region of interest of the scanned item (e.g. bone) with high resolution, we need to adjust the field of view of the scanner. As the relations between the pixel size of each slide and the field of view can be described as pixel length = FOV (mm) / slice length (px) and pixel width = FOV (mm) / slice width (px) respectively (Goldman 2007), if the FOV is adjusted, the voxel height will be affected and differ from its length and width (Liu, Li et al. 2018).

To simulate the real fluoroscopy, our system is intended to allow users to view the artificial hip joint they practice on from almost any angle they desire, like what the C-arm does in fluoroscopy. To realize that, we need to re-project the CT dataset from different angles. Currently, we just need to design some specific angles that are most commonly used (e.g. 0˚, 45˚, 90˚, etc.) while it will be populated into random angles in the future. Nevertheless, anisotropic voxels cause some barriers to reconstruction. Unlike the isotropic voxels, which can keep the shape of the image consistency when it is projected from different perspectives, anisotropic voxels deform the image when it is reconstructed (Figure 2.5 and Figure 2.6).

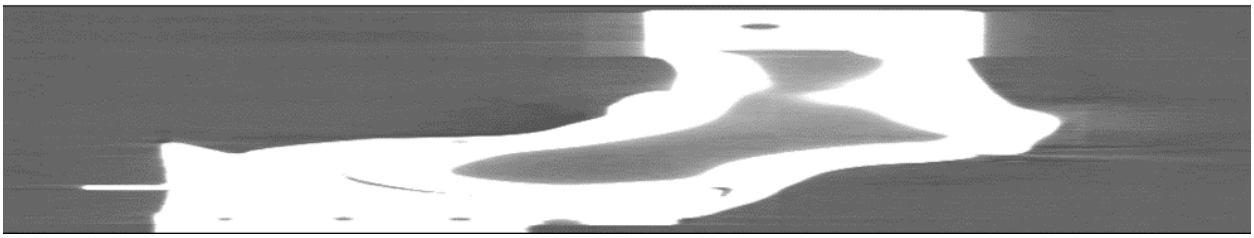


Figure 2.5 Reconstructed Anisotropic CT Dataset

(anteroposterior view)

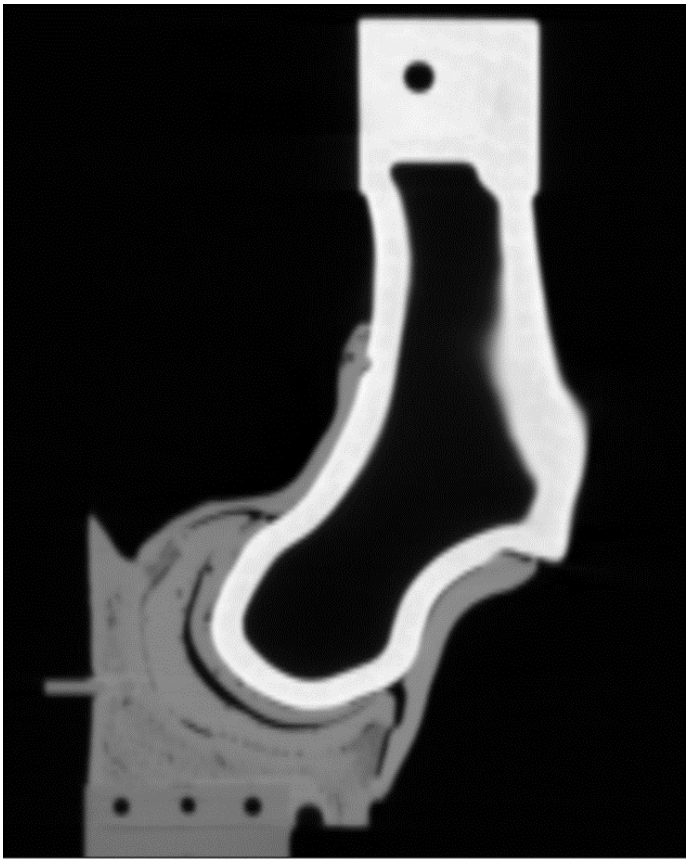


Figure 2.6 Reconstructed Isotropic CT Dataset

(anteroposterior view)

Therefore, to reconstruct the CT dataset, the first step was to convert the anisotropic voxels into isotropic. Fiji (Schindelin, Arganda-Carreras et al. 2012) is an open-source distribution of ImageJ focusing on

biological-image analysis. The original voxel size of the artificial hip CT scans is 0.15 mm x 0.15 mm x 1 mm (by Fiji) and the joint was scanned from its superior-inferior view. After importing the DICOM image sequence into Fiji, a plugin called *'Make Isotropic'* (Cooper and Julian.Cooper[at]uhb.nhs.uk 2009/05/22 First version) converts the anisotropic dataset into isotropic automatically. This plugin re-slices the stack of images along the Z-axis by linear interpolation to generate the voxel in which depth and width are the same, and the default Z-axis is along with the voxel depth. The original voxel depth should be larger than the width, otherwise, it may cause data loss.

After obtaining the isotropic CT dataset, we can project the object from any angle we want. Here, the anteroposterior view and the lateral view of the artificial bone were generated as an example. As fluoroscopy is a 2D image, the second step is to project the 3D CT scans from its vertical axis to construct a 2D fluoroscopy-like image. The function *Z-project* in Fiji can calculate the average intensity of each voxel along its vertical axis. The scanned object is expressed in the new average intensity voxels into a 2D plane (Figure 2.7). However, the pixel intensity in the CT images is opposite to the fluoroscopy, so we needed to invert the pixel values (e.g. rigid bones are represented by darker pixels and the air is white) (Figure 2.8).

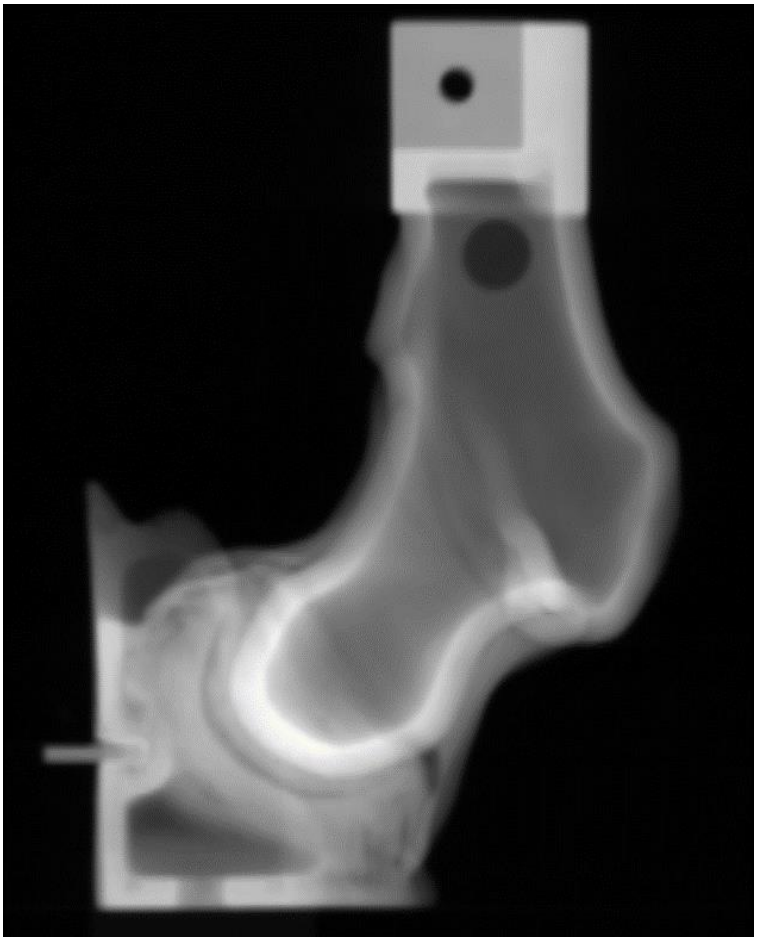


Figure 2.7 Projected Image Calculated by Average Intensity

(anteroposterior view)

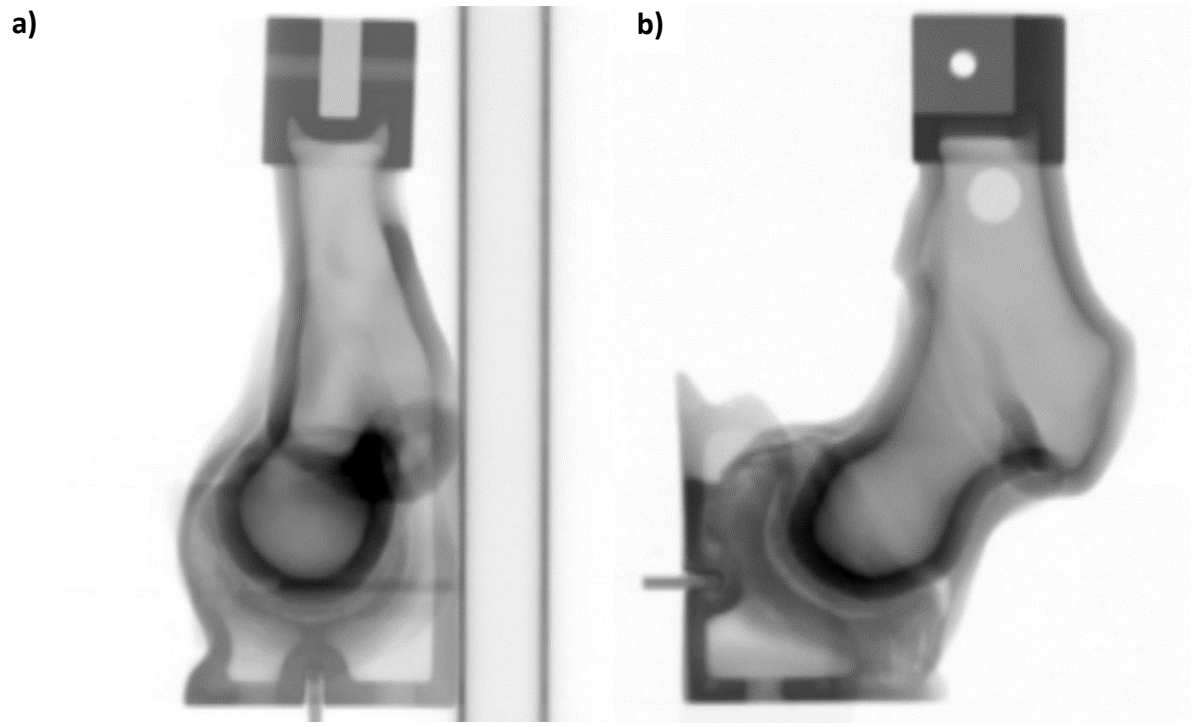


Figure 2.8 Reconstructed Virtual Fluoroscopy

(a. lateral view. b. anteroposterior view)

With the virtual fluoroscopic images, we are able to emulate the fluoroscopic C-arm in real orthopedic surgeries. By re-projecting the virtual 3D fluoroscopic images, we can 'observe' the artificial hip joint from different view angles, which is similar to what C-arm does. The intended tracking system tracks the surgical tool, which is a cannula, in 3D space in real-time by optical 3D motion tracking. There are some passive markers attached to the surgical tool and multiple cameras placed at different positions in the system to detect and track the 3D positions of these markers in the world coordinate. As a rigid body can be represented by at least three markers in a 3D space, we can obtain any point on the surgical tool by knowing the positions of each marker. The tool is then represented by a line and overlaid on the virtual fluoroscopy. This line is a projection of the tracked tool from the viewing angle. When the viewing angle

changes, the surgical tool should also be projected from the new angle, and then overlaid on the re-projected fluoroscopic image. The registration of fluoroscopic images with the global coordinate is necessary in order to cooperate with the tracked tool as its position is presented in the world coordinate.

The challenges of this project include:

1) Since this system is for surgical training, accuracy is a priority. We proposed that the system must track the surgical tool in a 3D workspace with reasonable precision (less than 1mm) in real-time. The purpose of this project is to test this with low-cost cameras (e.g. USB cameras or camera modules).

2) To reduce the systematic errors, the physical hip joint must be placed with precision into a set location relative to the global coordinate.

3) The 3D position of the tracked tool must be overlaid on the projected virtual fluoroscopic image of the hip joint from a required angle.

4) The CT dataset and the image projections that have been created to make the virtual fluoroscopic image must be aligned and oriented so that they match the position of the physical hip within the global tracking frame of reference.

The purpose of my project is to test the accuracy of a 3D optical tracking system based on inexpensive USB camera modules and to perform preliminary testing to overlay a tracked tool on the virtual fluoroscopy image (based on the CT dataset). This objective primarily addresses challenge No. 1 in the above list.

Chapter 3. Optical Tracking

3.1. General Concept of the Tracking System

3.1.1. Pinhole Camera Model

Camera is the core part of optical tracking. As a general rule of thumb, a camera with one lens can only produce a 2D image, which only contains the information on one plane. To track an object in the 3D space, it is vital to use more than one camera.

The most fundamental camera model is the pinhole camera model (Figure 3.1). It describes the relationships between a point in the 3D world and its projection on the 2D image plane of an ideal pinhole camera. A pinhole camera has no lens attached and its aperture is treated as a point. As well as projection distortions, blurring of unfocused objects and finite-sized apertures are all not considered in the ideal model. As the pinhole is extremely small, only one or just a few light rays emitted from the point on the 3D object can pass through, and then project on the camera film. Thus, there is a one-on-one linear relationship between the 3D object and its projection on the film, and the 2D object on the film is the mapping of the 3D object. The focal length is defined as the distance between the image plane and the pinhole.

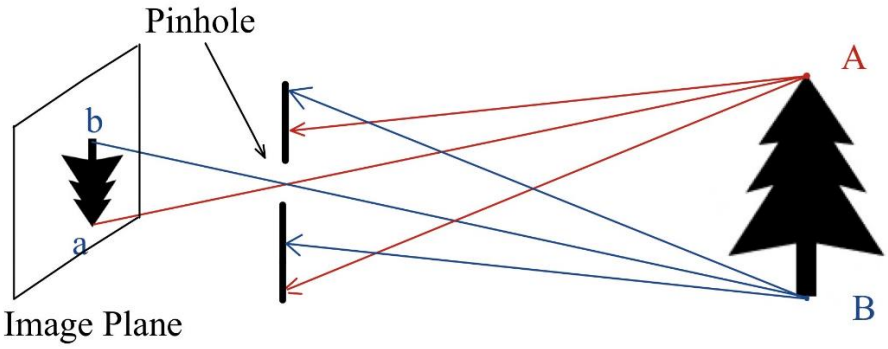


Figure 3.1 Pinhole Camera Model

To simplify the mathematical modelling, the virtual image plane is introduced to demonstrate the geometry of the pinhole camera model (Figure 3.2). The virtual image does not physically exist, and it is usually located in front of the aperture instead of standing behind, which is geometrically equivalent to the projection on the real image plane. Besides, the object is projected on the virtual image plane without being reversed. For example, a point P (x, y, z) is in a 3D coordinate XYZ, and it is mapping to the virtual image plane UV at the position (u, v). The origin of the 3D coordinate is defined at the centre of projection (pinhole in Figure 3.1) and its Z-axis is perpendicular to the image plane while the X and Y-axes are parallel to the image plane. The intersection of the Z-axis and the virtual image plane is defined as the origin of the UV coordinate. The UV coordinate is the camera reference system and the Z-axis is the optical axis of the camera system. The distance between the virtual image plane and the project centre is equal to the focal length defined previously. As there is no lens in the pinhole camera model, the mapping relationship between the points P and P_c is linear and can be described by the mathematical equations

$$(X, Y, Z)^T \rightarrow (u, v)^T = (\frac{fX}{Z}, \frac{fY}{Z})^T$$

where f is the focal length of the camera.

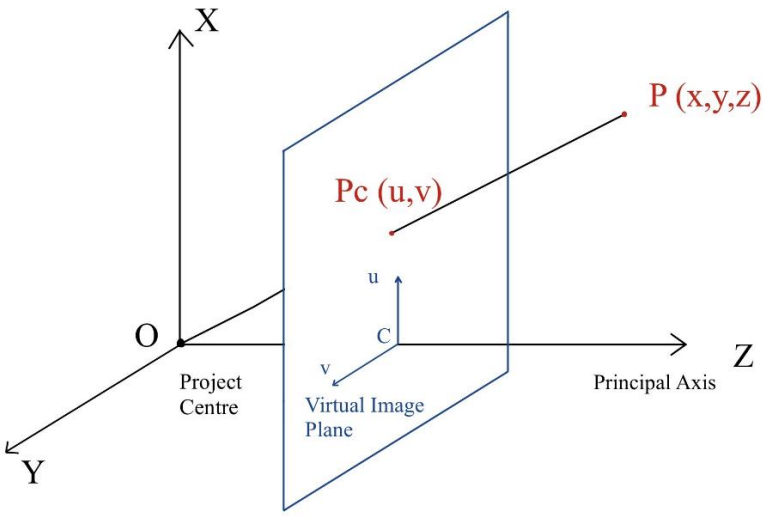


Figure 3.2 Geometry of the Pinhole Camera

With the use of a lens, the situation will be more complicated. As the lens refracts lights, the relations of mapping are not linear anymore and may cause distortions of the image. The most common patterns of image distortion are barrel and pincushion distortion (Figure 3.3). This distortion will introduce errors to tracking and it will be discussed in detail later.

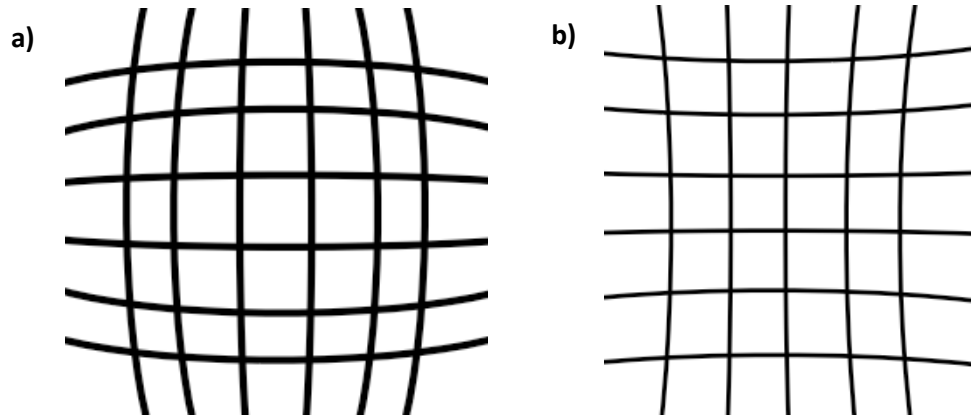


Figure 3.3 Image Distortion

a) Barrel Distortion b) Pincushion Distortion

3.1.2. 3D Reconstruction by Multiple Cameras

A single image is showing the 2D projection of the scene and the depth is lost. To reconstruct the 3D information, we need at least two images captured from different positions. This process is also referred to as triangulation. In a multi-camera system, if a 3D point is in the field of view of one camera, it can be captured on the image plane of the camera. By acquiring multiple 2D images captured by different cameras, we can calculate the interior parameters of the cameras such as image distortions, and the relative positions as external parameters by these 2D images. This process is called camera calibration.

Therefore, to track the surgical tool in a 3D space, at least two cameras must be set at different positions and their positions must be calibrated.

3.1.3. General Workflow of Artificial Fluoroscopy System

Passive markers are used in this system to track the surgical instrument during the arthroscopy training in real-time. Some reflective markers are attached to the surgical tool and cameras can detect and track these markers. After both cameras capture the markers on their images, we can reconstruct their 3D information. Then, the computer software generates the position and orientation of the surgical tool based on the realtime 3D positions of these markers. Using the position and orientation data, a virtual representation of the surgical tool (seen as a black line in Figure 3.4) will be overlaid on the virtual fluoroscopy to mimic the real arthroscopic surgery.

The general schematics of this system is described in Figure 3.4:

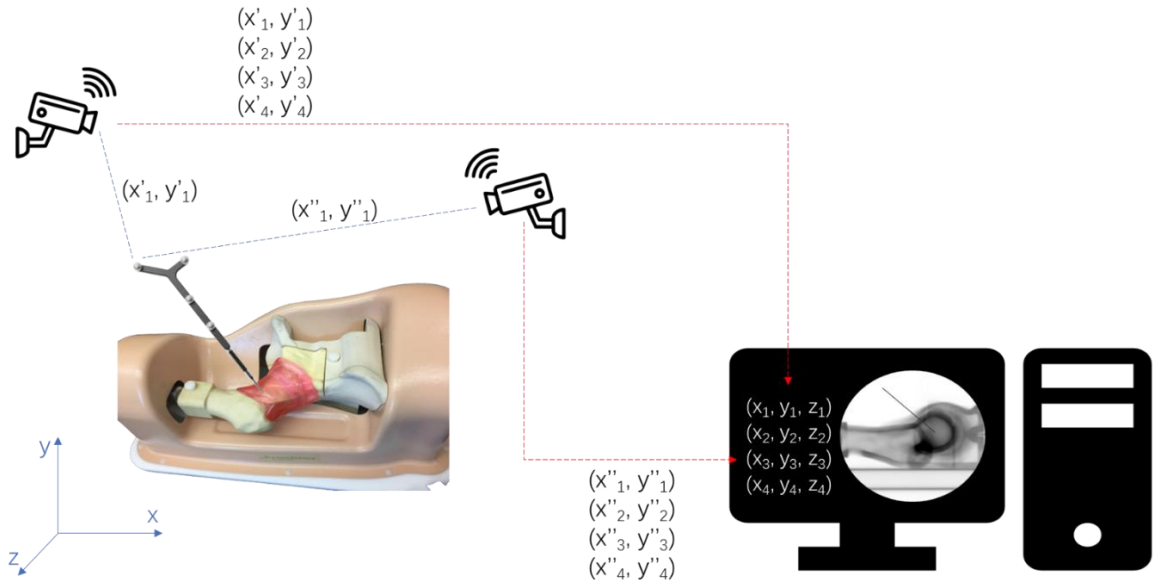


Figure 3.4 Schematics of the System

3.2. Camera Selection

The camera is the core part of this system, and the objective of this project is to attempt to build a surgical training system with a low budget. Therefore, it is essential to find a camera that can provide high-quality images but is also cost-effective. Ideally, the accuracy of this system can reach to sub-millimetre.

The workspace of the training system is intended to be a cube-shaped volume with approximately 50 cm sides, so the field of view of cameras should cover this volume to ensure any motions of the surgical tool are captured. Resolution is a parameter of the camera representing how many pixels a single frame can fit. An image with a higher resolution means its pixel size is smaller and it can provide more details. For instance, if the camera covers a 960 mm x 540 mm rectangle in the real world, and this area is projected on an image with a resolution of 1920 x 1080 pixels, then each pixel demonstrates a 0.25 mm^2 area (0.5 mm x 0.5 mm) in the real world. However, if the resolution is 480 x 270, each pixel covers a 4 mm^2 area (2 mm x 2 mm). Based on the definition, an image with higher resolution captures more photographic details than a low-resolution image. Accuracy defines the distance between the real geographic position of an object and its position on the image. Several factors affect accuracy. For 3D triangulation in this project, accuracy is dependent on the image distortion introduced by the lens, camera calibration, algorithm for 3D reconstruction, etc., so it is not directly affected by the image resolution. However, high-resolution images can provide more details in the field, which is beneficial to accuracy. In this case, to achieve the intended accuracy requirements of less than 1 mm, it is necessary to use cameras that are capable of relatively higher resolution.

Another concern is the frame rate of the camera. Frame rate describes how many frames the camera can record each second. As the system is to track the surgical tool in real-time, the frame rate should be high enough to capture the movements of the surgical tool. In most systems, there is a trade-off between resolution and frame rate. A higher frame rate means the computer needs to calculate a larger amount of

data per second, which is a great challenge to the capacity of the computer hardware, so typically the system reduces the resolution of the camera at a higher frame rate. This is also a major reason why the current high-resolution tracking systems or the optical systems are relatively expensive – they are capable of capturing high-resolution images at high frame rates. There are also expenses due to better lens quality with fewer optical aberrations.

Unlike tracking systems used in sports like tennis or baseball, the movements in surgeries are not considered as high speed. For most of the video on the websites or films, the standard frame rate is 24 fps as it supports most of the continuous motions (Engstrom 1935). 24 fps is also the lowest frame rate to avoid objectionable flickers to the audience. 30 fps videos are also common in our daily life and it plays video in a much smoother way compared to 24 fps. Though a higher frame rate has a positive effect on accuracy, it also increases the cost of the hardware. Since motions in surgeries are usually slow and a general frame rate for the endoscope is between 20-30 fps (McKenna, Charif et al. 2005, García-Peraza-Herrera, Li et al. 2016), it is reasonable to set the frame rate as 30 fps in our system, which is also the frame rate in the standard full high-definition video (1080p: resolution is 1920 pixels displayed horizontally across the screen and 1080 pixels vertically, the frame rate is 30 fps).

Lots of USB webcams can provide Full HD video. For example, Logitech C920 Webcam HD Pro (Logitech International S.A., Lausanne, Switzerland) can generate 1080p /30 fps video streaming and costs only CAD 99.00. Logitech BRIO – 4K Ultra HD (Logitech International S.A., Lausanne, Switzerland) provides professional-quality video including 4K (3840 x 2160 pixels) video or 1080p with the frame rates at 60 fps at the price of CAD 249.99. However, the functions are highly constrained in these commercial USB cameras, and we are not able to change or adjust their functions as we need for the custom tracking system. The physical structure of a camera is another consideration in using USB webcams, as it is difficult to supplement a lens if any complementary requirements appear in the future scenario. Thus, programmable cameras or camera modules are a better choice in this system. The

programmable cameras are flexible in any applications and users can adjust the parameters or write customized image processing programs based on the demands.

The Raspberry Pi camera module is one of the most common programmable cameras in numerous systems. The Raspberry Pi is a small single-board computer that does not contain peripherals such as keyboards or mice in customary computers. Instead, users may add peripherals according to their requirements.

This project focussed on using a Raspberry Pi 3 Model B+ and Camera Module V2 (Raspberry Pi Foundation, Cambridge, United Kingdom) for the tracking system and to test out the error and repeatability. Raspberry Pi 3 Model B+ is the third generation Raspberry Pi single-board computer, which holds a 1.4GHz 64-bit quad-core processor. The Camera Module V2 supports 8-megapixels (3264 x 2448 pixels) still captures and 1080p /30 fps video outputs. Independent processing capacity is another merit of Raspberry Pi. A challenge of live tracking is that as the resolution and frame rate increase, the amount of data waiting to be processed each second grows. When multiple cameras are working in one system, there is an exponential growth in the amount of data, which is a great challenge to the Central Processing Unit (CPU) in the computer. The independent CPU on the Raspberry Pi can share the data processing pressure with the central computer. So, instead of outputting the whole image, a Raspberry Pi module can output only the 3D positions of the markers as a numerical matrix, which greatly reduces the amount of data transferred from the Raspberry Pi to the central computer.

There are two different types of Raspberry Pi camera modules, the normal visible light cameras (Camera Module V2) and NoIR cameras (Pi NoIR Camera V2). The NoIR camera modules do not have infrared filters, so the colour of images it captures in the daytime is a bit different from normal cameras. However, it supports night vision with the use of an IR illuminator. Since the application scenario of this optical tracking system is indoors and under good light conditions, it is unnecessary to use NoIR cameras and normal cameras can provide better quality images.

3.3. OptiTrack System: Cameras and Software (Motive)

A commercial optical tracking system (OptiTrack, NaturalPoint, Inc. DBA OptiTrack, Corvallis, OR) was used to set up the calibration system for the custom USB camera system and as a comparison for the measurements obtained from the USB camera system. As explained in section 3.4, a custom calibration frame was built to calibrate the positions of the USB cameras relative to the global coordinate system. The commercial OptiTrack system was used to measure the 3D positions of the markers in the custom calibration frame.

3.3.1. Camera Setup and Calibrations

OptiTrack motion capture system (NaturalPoint, Inc. DBA OptiTrack, Corvallis, OR) is an optical tracking system that consists of multiple cameras and a processing software (Motive). Currently, four Flex 13 cameras are available in the lab. Flex 13 offers images with 1.3 megapixels (1280 x 1024 pixels) and the pixel size of the image sensor is 4.8 µm x 4.8 µm. The frame rate is adjustable from 30 to 120 fps. There is a LED ring around the lens and illuminates 850 nm infrared light. A default 800 nm IR long-pass filter is attached in front of the lens, which allows wavelengths longer than 800 nm to pass, and it blocks wavelengths shorter than 800 nm. Therefore, the camera can detect all objects reflecting lights from the 850 nm LED IR ring.

The Flex cameras detect and track the movements of the reflective spherical markers attached to the tracking target. Theoretically, cameras can detect all reflective surfaces satisfying the brightness threshold set in the camera properties, so it is necessary to minimize the ambient light either from surroundings or non-targeted surfaces. All reflective surfaces including the floors need to be covered with non-reflective materials like rubber mats and all unnecessary reflective obstacles need to be removed. This helps avoid

optical noise and unintended light interference from the environment. During the motion capture, multiple synchronized cameras are installed around the intended tracking space, and each camera captures 2D images simultaneously. Then the Motive software reconstructs the 3D positions based on the 2D positions collected by the cameras.

In general, cameras should be placed around the periphery of the capture volume at various heights and angles to capture from a distinctive view of the volume. In the meantime, their views should overlay as much as possible in the region where most of the tracking will take place, and the cameras should be closer to the target for precise tracking. Each marker must be captured by at least two cameras in order to triangulate its 3D position. The diverse perspectives help the system minimize the blind zone and occlusion, which improves the accuracy of tracking. If the capture volume size and camera configurations are optimal, the tracking accuracy of this system can be submillimeter (NaturalPoint 2020). It is also essential to mounting the cameras securely onto the stable structures or frames, as any slight vibrations or tiny movements of cameras may cause errors on the motion captures. To gain optimal tracking data, all the cameras should be accurately focused on the target. By adjusting the camera properties, the camera can capture markers with the best clarity.

Before capturing the movement, the most vital process is camera calibration. The software Motive supports a convenient calibration operation for users. By waving the calibration wand manufactured by OptiTrack in the capture volume, Motive tracks the positions of markers on the wand on each frame. When enough samples are collected, Motive computes the positions and orientations of cameras as their external parameters. Meanwhile, it calculates the image distortion caused by the lens as a part of the camera interior parameters. There are two models of calibration wands currently available in the lab, one is CW-500 Calibration Wand Kit and the other is CWM-125 Calibration Wand (NaturalPoint, Inc. DBA OptiTrack, Corvallis, OR) (Figure 3.5.b). The maximum space between the markers with compatible posts on the CW-500 wand is 500 mm and it is 125 mm on the CW-125 wand. CW-125 is a micro-series wand and designed for precision tracking scenarios, as any slight distortions in the wand length can

greatly affect the result in capturing precision measurement. If the capture volume is large, CW-500 (Figure 3.5.a) is more suitable for calibration while the CWM-125 provides precise calibration when the capture volume is small. It is important to match the calibration tool you select with the tool type chosen on Motive, otherwise, the calibration results would be wrong.

The waving wand only calculates the relative camera positions while the calibration square helps set the ground plane and origin of the world coordinate. Place the CS-200 Calibration Square (NaturalPoint, Inc. DBA OptiTrack, Corvallis, OR) (Figure 3.5.c) on the plane which is defined as the ground plane and position the right angle of the calibration square at the origin of the world coordinate defined by the user, then Motive will capture the markers on the square and determine the world coordinate in the capture volume. Every position reconstructed in the motion capture process will be expressed based on this world coordinate.

a)

b)

c)

Figure 3.5 Calibration Tools

a) CW-500 Calibration Wand Kit. b) CWM-125 Calibration Wand. c) CS-200 Calibration Square

The positions of markers can be streamed out simultaneously while tracking and it supports third-party software including Autodesk Motion Builder, 3ds Max or customized clients to receive the data of

motions. The clients are coded by Matlab, C#, C++, etc. so that users can utilize these live data streaming according to their applications.

3.4. Custom Calibration Frame

3.4.1. Design and Fabrication

A multilayer wooden structure was designed and assembled for camera calibrations by the Direct Linear Transformation (DLT) method (Abdel-Aziz, Karara et al. 2015). The DLT calibration process requires at least 6 markers placed at different planes, so a multilayer calibration structure is ideal to set the markers for camera calibration.

As the width, length and height of the artificial hip joint are approximately 25.4 cm x 48.3 cm x 19.0 cm respectively, the calibration structure is supposed to be similar dimensions to keep the capture volume consistent with the workspace. The structure consists of four plywood boards (21-inch-long, 10.5-inch-wide and 1-inch-high (53.3 cm x 26.7 cm x 2.5 cm respectively)). These boards were held together by screws, so the integral dimensions of the structure were 21 in x 15 in x 4 in (53.3 cm x 38.1 cm x 10.2 cm) where each step was around 1.5-inch-wide (Figure 3.6). As the capture volume was only about 50 cm x 50 cm, it was better to use the CWM-125 Calibration Wand to calibrate instead of CW-500 Calibration Wand for better calibration results. Also, three cameras worked better than four cameras because if too many cameras are set in a small space, the LED rings on the cameras cause interference to each other which even deteriorates accuracy.

Twelve spherical markers were set on the structure, with each of the four levels containing 3 markers (Figure 3.6). The marker bases were fixed on the structure by double-sided tape so that they were fixed but could still be taken off if needed. These markers were used both in the calibration process and testing the accuracy and repeatability of our system.

We defined the origin of the world coordinate at the left bottom corner of the top board (point O on Figure 3.6), the positions of these markers were measured by the OptiTrack system. One challenge was that the software for the OptiTrack system can only capture the moving markers, but the markers on the calibration frame were static. Thus, we needed to use an OptiTrack measurement tool to measure the positions of markers.

The Digitizing Probe (NaturalPoint, Inc. DBA OptiTrack, Corvallis, OR) is designed for 3D digitalization of the real-world object or identifying virtual points. By detecting the markers attached to the probe (Figure 3.7), the tracking software (Motive) outputs the real-time position of the probe tip. Therefore, the probe can be used to measure the positions of these markers when the probe is pointing at the top of the marker bases (Figure 3.8).

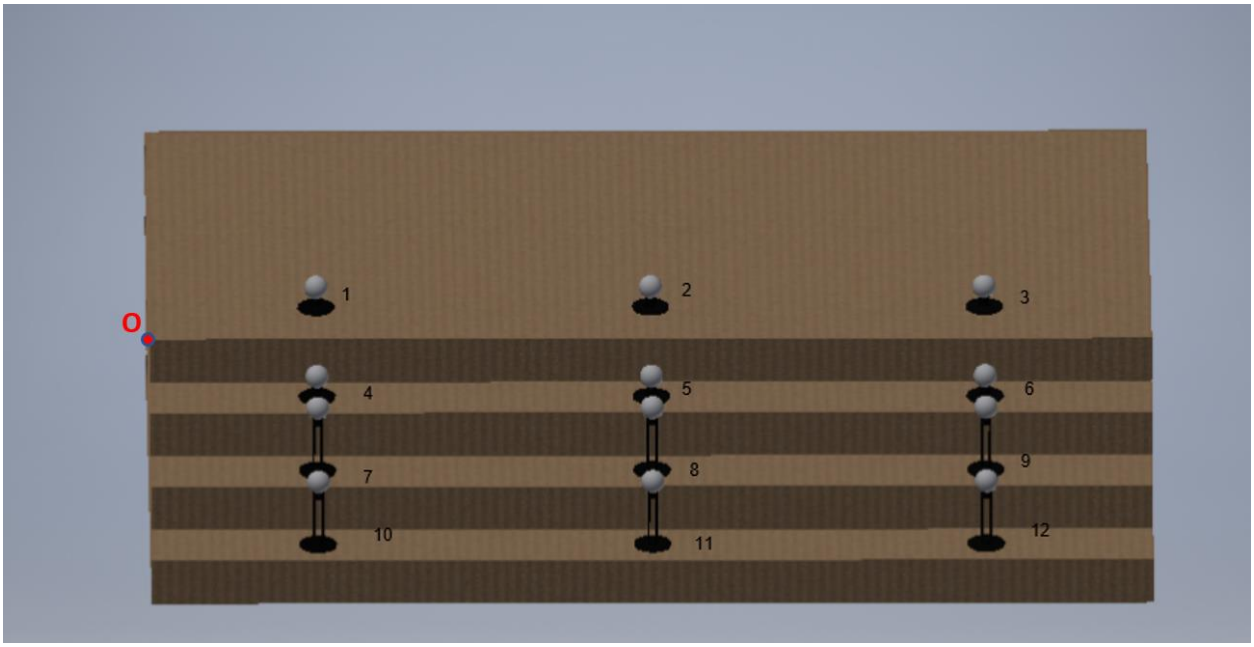


Figure 3.6 Labeled Markers

Standing on the layers with the origin 'O' of the world coordinate

Figure 3.7 OptiTrack Digitizing Probe

Figure 3.8 OptiTrack Marker Bases

To calibrate the probe before measurement, the four markers on the probe had to be in the capture volume before calibration. These four markers were defined as a "measurement probe" under the Measurement Pane in Motive. During calibration, the tip of the probe was pointed at one of the tiny grooves on the calibration bar and then the probe was rotated with the fixed tip for the software to collect sample data.

After calibration, Motive shows the live 3D position of the tip, which was then used to measure the positions of the marker bases (Figure 3.9 and Figure 3.10).

3.4.2. Ground Plane Refinement

After calibrating the system by waving the wand and setting the ground plane and the origin of the system, there were still errors in the vertical direction. Concretely, when we put the tip of the probe at random places on the floor (in the range of the calibration frame), the value difference in the vertical direction was as large as 2 cm. To correct this vertical offset, we set several OptiTrack reflective markers on the floor, then selected these markers in the software Motive to refine the ground plane. After levelling the ground plane, the heights of the floor were consistent, and the final step was to set the origin of the system again by using the calibration square.

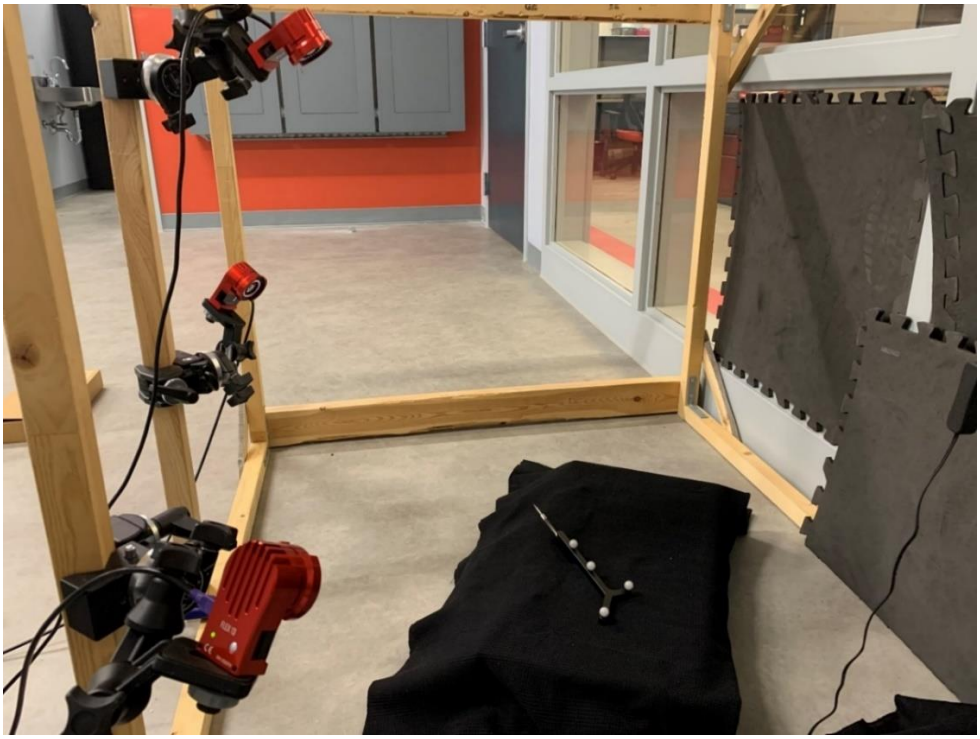


Figure 3.9 Motion Capture with 3 OptiTrack Cameras

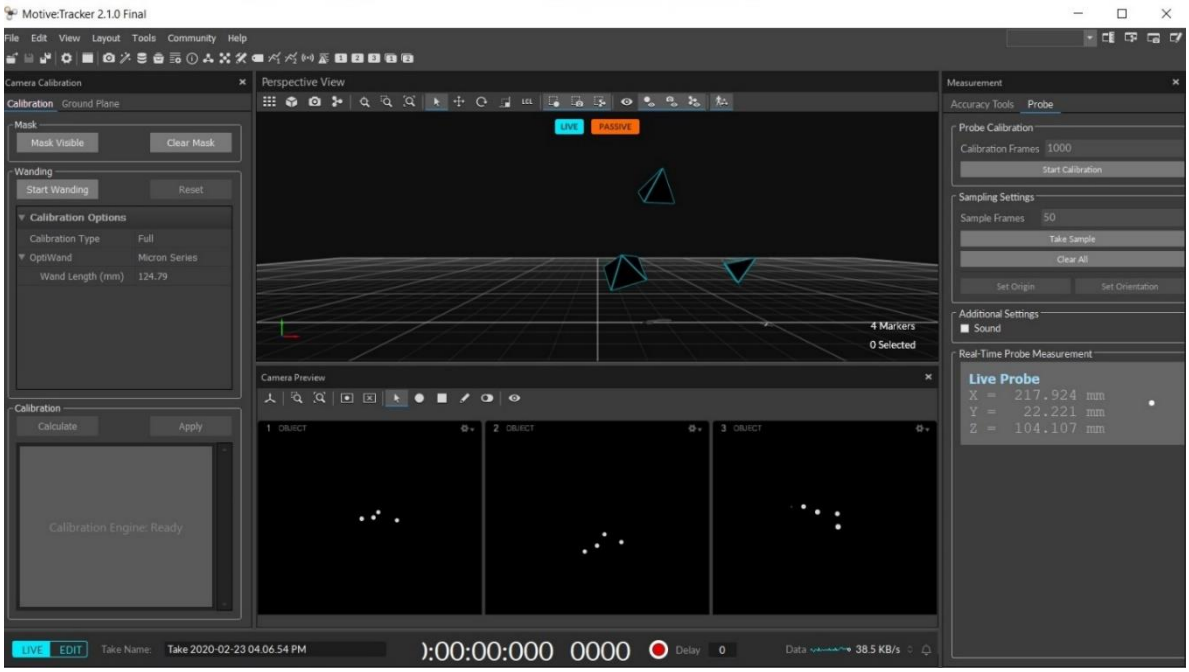


Figure 3.10 Motive Interface with Live Positions of the Probe

3.4.3. Calibration Frame Marker Position (Measured by OptiTrack)

To measure the 3D position of each marker based on the global coordinate we defined on the calibration frame, we used the digitizing probe to point at the upper centre of each marker base five times (Figure 3.11), and the average value was considered as the position of this marker. As we defined the top left corner of the calibration frame as the origin of the world coordinate, the final position of each marker was corrected with the measurement position of the origin. The final positions of markers are shown in Table 1. The largest standard deviation of the measurement was from the x value of the No. 1 marker (1.13 mm), and most of the standard deviations were in the range of 0.11 mm to 0.6 mm.

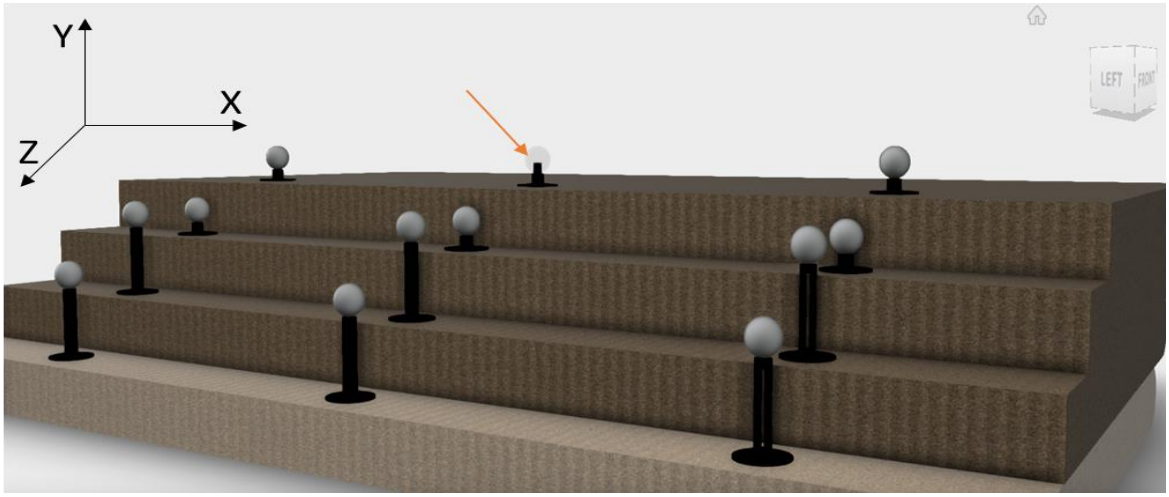


Figure 3.11 Upper Center of a Marker Base

(Where the digitizing probe pointing)

Table 1 Final Positions of Markers Measured by OptiTrack

	Measured Position (mean ± SD) [mm]		
Point	X	Y	Z
1	114.2 ± 1.13	11.4 ± 0.37	-36.0 ± 0.60
2	267.4 ± 0.47	10.6 ± 0.12	-37.3 ± 0.46
3	421.2 ± 0.39	11.4 ± 0.11	-35.8 ± 0.12
4	121.9 ± 0.31	-16.2 ± 0.28	23.4 ± 0.54
5	274.3 ± 0.09	-15.7 ± 0.08	22.8 ± 0.43
6	418.2 ± 0.14	-16.0 ± 0.18	24.0 ± 0.22
7	123.2 ± 0.45	-18.9 ± 0.32	60.4 ± 1.05
8	272.9 ± 0.59	-20.3 ± 0.13	58.4 ± 0.15
9	419.3 ± 0.19	-20.4 ± 0.34	61.5 ± 0.18
10	119.1 ± 0.14	-49.0 ± 0.39	98.37 ± 0.37
11	270. 2 ± 0.48	-48.7 ± 0.58	98.3 ± 0.14
12	423.4 ± 0.32	-48.6 ± 0.16	99.1 ± 0.44

As it is impossible to put the tip of the probe at the same point on the base top each time, the measurement process generated errors in the X and Z directions. This explains why the standard deviations of the positions are in the range of 0.08 mm to 1.13 mm, and the standard deviations in the Y-axis are less than 1 mm. The errors generated in the measurement process caused errors in the camera matrix calculation in the calibration process, and the reconstructed surgical tool in the virtual fluoroscopy

will not be in the real place during the tracking. However, the repeatability should not be greatly affected by measurement errors.

In future work, we can try other physical measurement methods to obtain the positions of markers more accurately. For example, high-precise coordinate measurement machines (CMM) like FaroArm (FARO Technologies, Inc., Florida, USA) may provide precise position information to the submillimeter level. In other words, the more accurate the positions of the markers, the better results we can receive from our tracking system.

Chapter 4. Raspberry Pi with Camera Modules

4.1. Camera Distortion Elimination

Image distortion is universal in real camera imaging. It is caused by light refractions when the lights pass through a lens. Distortion makes a supposed-to-be-straight line to be curved on the image, which does not perfectly map the real scene. Furthermore, the effect is more obvious on the edge compared to the centre of the image. The distortion leads to errors in optical tracking as the positions of objects showing on the image may be different from their real positions.

There are two major types of distortion: radial distortion and tangential distortion. Tangential distortion usually happens when the lens is not perfectly parallel to the image plane, which causes some area to look closer than expected. The classical lens distortion model was proposed by Brown (Brown 1966, Duane 1971), and the equations of the model are (De Villiers, Leuschner et al. 2008):

$$x_u = x_d + (x_d - x_c)(K_1 r^2 + K_2 r^4 + \cdots) +$$

$$\left(P_1(r^2 + 2(x_d - x_c)^2) + 2P_2(x_d - x_c)(y_d - y_c)\right)(1 + P_3 r^2 + \cdots)$$

$$y_u = y_d + (y_d - y_c)(K_1 r^2 + K_2 r^4 + \cdots) +$$

$$(2P_1(x_d - x_c)(y_d - y_c) + P_2(r^2 + 2(y_d - y_c)^2))(1 + P_3 r^2 + \cdots)$$

Where (x_d, y_d) = distorted image point,

(x_u, y_u) = undistorted image point,

(x_c, y_c) = centre of distortion,

$K_n = N^{th}$ radial distortion coefficient,

$P_n = N^{th}$ tangential distortion coefficient,

$$r = \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2}$$

This formula transforms a distorted image point into an undistorted image point. By obtaining the parameters K_n and P_n, we can correct the image without distortion. In most of the cases, at least four points on a straight line are required to determine the mapping parameters. In practice, lots of algorithms take a rectangular grid of straight lines, like a chessboard, to calculate the mapping parameters. These parameters correct the distorted images and improve the result of camera calibration, which is beneficial to precise tracking.

Distortion corrections are generally based on that the mapped straight line on the image should still be straight. By extracting the points on a line, the distortion error should be the vertical distance between the point to the corrected line. Points on the line help us to determine what the straight line should be on the image and get the lens distortion parameters by the nonlinear least-squares fit or other mathematic models (Figure 4.1).

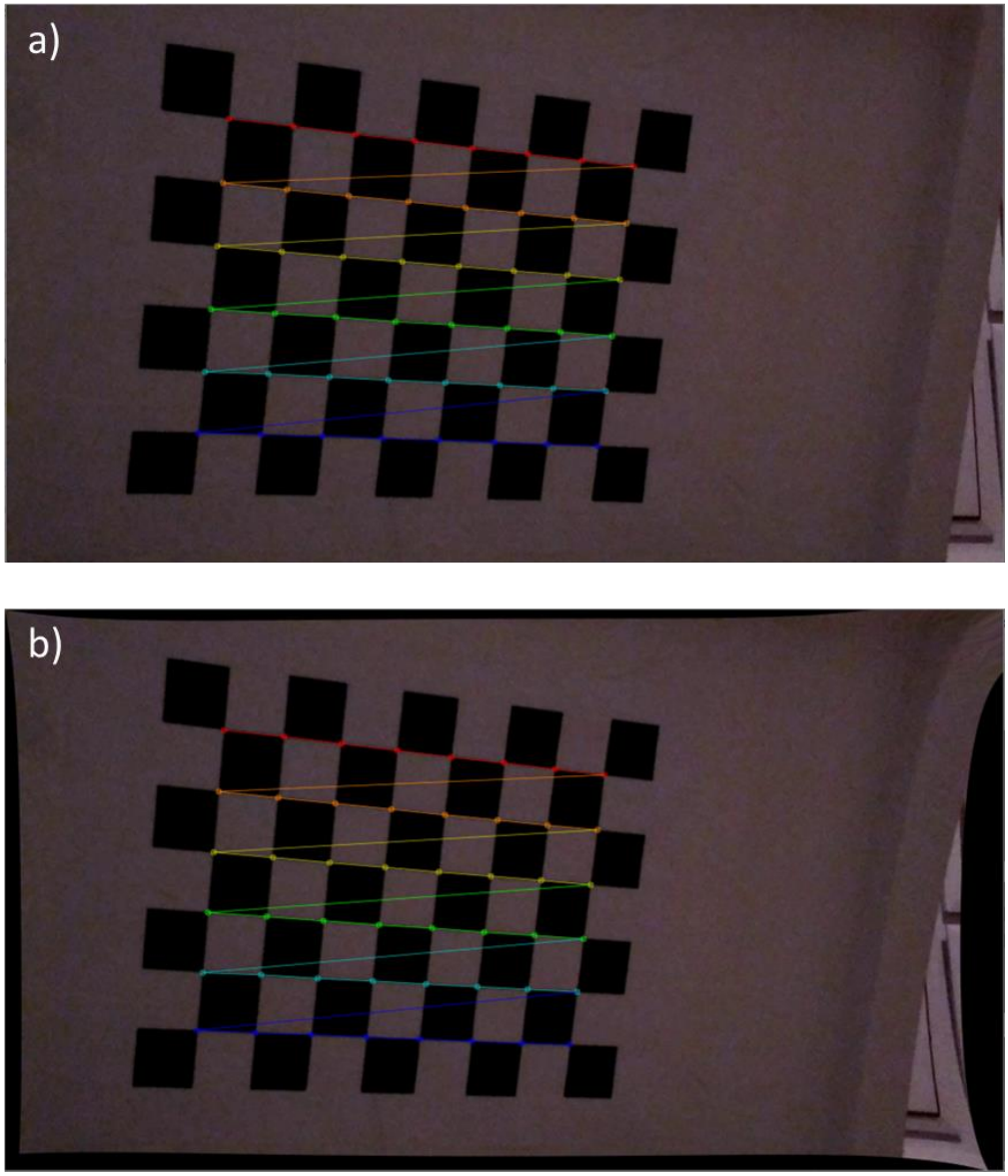


Figure 4.1 Original Image vs. Undistorted Image

a) Original Image Captured by Raspberry Pi (resolution: 1920 × 1080).

b) Undistorted Image Corrected by OpenCV

Image distortion can bring errors in the tracking. To increase the accuracy of our system, I implemented the image distortion algorithm to correct the images captured by a printed chessboard and Raspberry Pi

cameras. As shown in Figure 4.1, the edges of the corrected image were warped. Since the original image was distorted, its edges were pulled away after correction. The average error of the distorted image in Figure 4.1 is 5.30 pixels. The scale of the pixel and millimetre is approximate 0.22 mm /pixel, so the error converted to mm is around 1.16 mm. Regrettably, I didn't have a chance to apply the distortion correction to our system because the University was shut down during the COVID-19 pandemic. Therefore, the results in this thesis were based on the uncorrected images I collected before the pandemic.

4.2. Marker Detection, Filters and Noise Cancellation

There are several techniques to track a moving object, and they are generally based on the special features the object holds. For example, we could track an object by using the current frame to subtract the previous frame, and the difference between these two frames would be the moving object. However, as we are going to track the surgical tool held by surgeons, the moving components will also include the surgeon's hand, which will add more processes to differentiate the hand from the tool. Alternatively, we can detect markers attached to the tracking object and reconstruct the object by these markers.

Markers should be unique to distinguish them from their surroundings. Passive markers and active markers are two types of markers in the optical tracking. Active markers usually show ideal performance in large tracking space. Since the active marker emits LEDs by itself, it relies on cables to provide electricity, which confines its application scenarios and is barely used in the optical tracking system. Passive markers reflect lights in the designated threshold and are wireless, so they are handier and more cost-effective. Therefore, passive markers are used in this optical tracking system.

As the OptiTrack tracking system is employing IR reflective markers, it is a good try to detect markers based on their colour. Furthermore, if we detect markers with IR reflections, we need the IR light source and IR pass filter attached to the camera lens, which makes the structure more complicated. Another

reason is that it would be difficult for us to differentiate markers as they appear the same on the filtered images. Colour detection does not need additional accessories such as LED lights or IR filter and helps reduce the cost and external errors introduced by the optical components to some extent.

4.3. Colour Segmentation

Colour segmentation simplifies the detection problem as it assumes the objects are coloured distinctively. As it discards extra information including surface texture or shape which can be influenced by colour or brightness variations, the speed of image processing can be rapidly improved so that it is a benefit for real-time tracking. If we have markers in an unmatched colour with the background, only the pixels in the particular range can be retained and all the rest of the data can be discarded. In this way, we only need to treat pixels representing the markers, and it greatly reduces the processing time, which is critical in real-time tracking. Another reason we chose the colour segmentation is that the Raspberry Pi processes complicated calculations very slow. Therefore, other more robust tracking methods like object trackers are not suitable in our application. Since object trackers use a more complicated algorithm, the speed of tracking can only reach approximately one frame per second, which is useless in real-time tracking.

Any colour consists of the primary colour of pigment which defines a colour space. Colour space is a mathematical model to represent the colour by a tuple of numbers. For example, RGB (R = red, G = green, B = blue) is one of the most popular colour spaces, which uses red, green and blue as the fundamental components of an arbitrary colour. In this colour space, each colour can be expressed by these three colours with a specific ratio, and the intensity value of each component is in the range of (0, 255) (Figure 4.2). Other than RGB colour space, HSV (Hue, Saturation, Value) colour space is more common in object detection and tracking purposes, and it is transformable with RGB colour space. This is because HSV separates a colour from the intensity, which is more robust to lightness variations or shadows. Shadows or lightness changes may alter the colour, for example, red does not look like red.

However, the hue value of colour keeps similar when the light conditions are different. In this way, it is convenient for us to extract all red-like colours in the tracking space (Figure 4.3).

YUV is another colour space composes of a luma component (Y) and two chrominance components (U and V), where U represents blue projection and V represents red projection. When the U value is larger, the colour is close to blue while the colour is approaching red when V increases. Theoretically, YUV colour space behaves well when we need to segment the blue and red objects. It also has advantages such as high-speed processing and not sensitive to lightness variance (Figure 4.4).

All of these colour space conversions can be realized by OpenCV with the function cv2.cvtColour() (Bradski 2000).

Figure 4.2 Image in RGB Colour Space

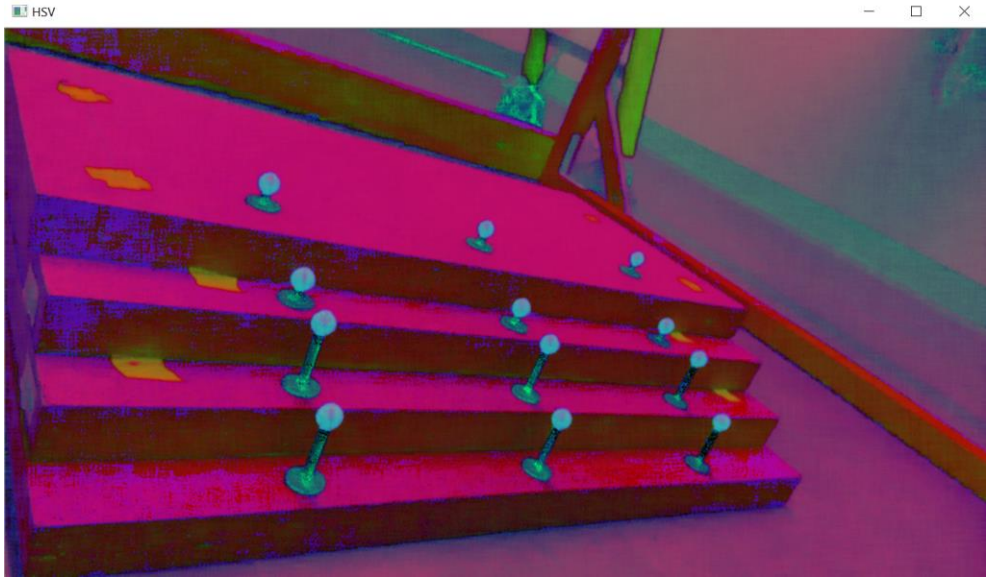


Figure 4.3 Image in HSV Colour Space

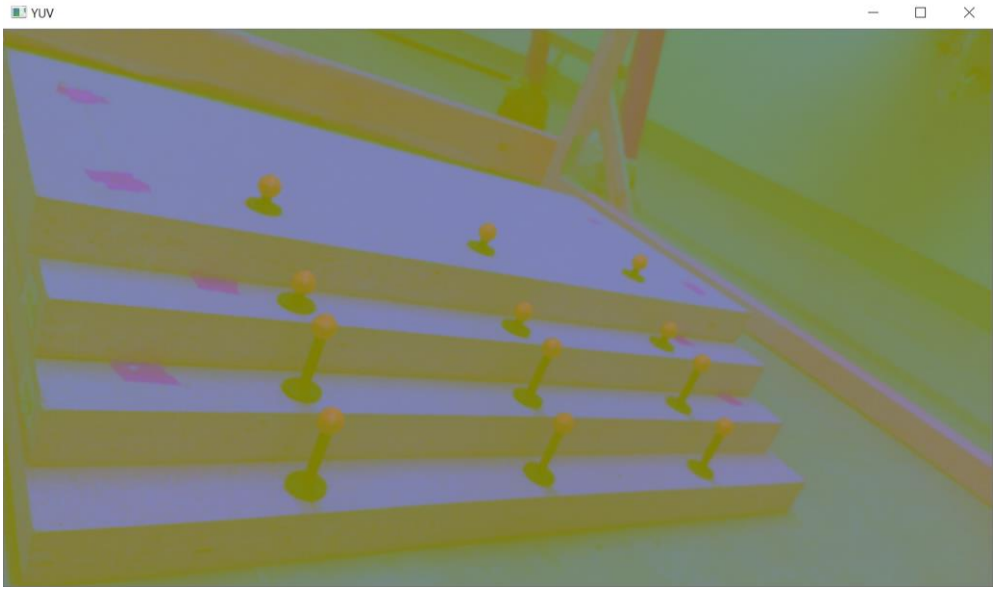


Figure 4.4 Image in YUV Colour Space

According to the tests I performed, HSV has the most robust and stable performance in the segmentation of the markers between three colour spaces.

We 3D printed 16 spherical red markers with a diameter of 12.7 mm (Figure 4.5). The size of marker influences the visibility of the tracking system (NaturalPoint 2018), and it is related to the marker distance to the camera (Pentenrieder, Meier et al. 2006). Large markers are obvious for tracking when the tracking field is large, but it is not sensitive enough for fine movements or small objects. On the contrary, smaller markers work better in precision tracking but cause difficulties at long-distance tracking because the area of pixels representing the marker is too small to detect. Since the tracking field of our system can be considered as small (50 cm x 50 cm x 50 cm), and we need to track the markers precisely, we chose the size of the 3D printed markers based on the OptiTrack Digitalizing Probe (Figure 3.7), which also been used as the tracker in the testing. The diameter of the reflective markers attached to the Digitalizing Probe is 12.7 mm (1/2 inch). We decided to adopt spherical markers rather than the planar calibration tool because the spherical markers can be detected at any angle and greatly reduce the blind zone, which helps obtain a better calibration result. The initial design was to use three red markers and one green marker to track. The green marker is the fundamental marker, and other markers are labelled based on their distance with the green marker. The tracking algorithm only needs to detect and label the markers once and only if all four markers are detected and labelled, the tracker from OpenCV starts to track these markers independently. Theoretically, the tracker is a more robust algorithm for tracking than simply employing colour segmentation, as the tracker follows the object moving until the object is out of the image. However, two issues prevented us from using this method. The first one is the relatively large computations. As the limitation of Raspberry Pi, the tracker algorithm is too complex for it to process in realtime. In reality, the Raspberry Pi can only process approximately one frame each second using the tracking algorithm based on generic markers. In this case, we have to be moving the object extremely slow, otherwise, the tracker would lose tracking because the motion between the two adjacent frames the algorithm processes is too large for the tracker to follow. This is not what we want in our application. The other problem is the distance calculation for labelling markers is not always reliable. As the cameras are located at the right and left of the frame respectively, the distance between a marker and the fundamental marker is not always constant, which may lead to confusion for the system to label the markers. For

example, a marker may be labelled as No. 3 in the left camera while it is the No.4 marker in the right. This could give the wrong reconstruction in the later steps. Therefore, we need to find a simpler and more stable way to detect markers.

Figure 4.5 3D Printed Red Marker

(ø 12.7 mm)

To detect markers and distinguish them separately, each marker should have at least one unique feature for the system to identify it. Meanwhile, the identification process should be simple enough for the Raspberry Pi to process in realtime. Thus, we painted each marker with a different colour so that each colour corresponds to a label. Even though the program needs to recognize the markers on each frame, it is still fast enough to realize real-time tracking with a 25 fps frame rate.

To make the markers prominent relative to the surrounding, any surface with similar colour with the marker should be covered or painted. For convenience, we painted all objects including the calibration structure and the frame for placing the cameras into black and covered the floor and other surfaces that are not able to be painted with black cloths to prevent interference and reflections. In this project, the green marker is specified as No. 1, the yellow marker is No.2, the blue marker is No.3 and the red marker is No.4 (Figure 4.6).

Figure 4.6 Painted Markers on the Probe

To guarantee a better result, we need to filter out the noise and objects which are not the markers but can be detected by the software, so we set some thresholds. One threshold can filter the noise by its area. If the detected object is too small, it would be filtered as noise and only objects larger than a certain threshold would be recognized as a marker. In the beginning, we set the minimum detectable size as 500 pixels during the tests, but it can be changed according to the positions of the cameras because if the camera is further, the size of the marker will be smaller. To simplify the calibration process, we wrote a function to detect one marker each time (color_segmentation(), Appendix B). The function detects one marker based on its specific colour, and this function will be called four times within one frame. This also reduces the complexity and runtime of the program. Since only one marker should be detected once, the marker should be the largest detected area in the whole image. Therefore, the function will only output the centre with the largest detected area. With this setting, we do not need to set the size threshold each time and it makes the detection more robust and accurate. The other filter is due to the shape. The desired object should be approximately circular in shape. If the detected object is not circular or not similar to a circularity, it would be marked as noise.

There are other methods to detect spherical markers such as detecting all the circles in the image. However, it is not as flexible as to colour segmentation and we found that it is not robust enough and brings a lot of errors based on a number of tests that we ran. Also, the algorithm to detect circles such as Hough Circle Transform (Illingworth and Kittler 1987) requires numerous calculation so its processing speed is not as ideal as colour segmentation. Therefore, we choose HSV colour segmentation to detect markers in this tracking system.

4.4. Camera Calibration

4.4.1. Fundamental Coordinate Transformations

To establish the mapping relationship between a point in the real 3D world and the camera plane, we need to implement several coordinate transformations. First, here are the definitions of some coordinates:

World coordinate: defines a coordinate in the physical world to describe the 3D position of an object

Camera coordinate: defines a coordinate on the camera, the origin is the centre of projection, the Z-axis is perpendicular to the image plane, and the distance from the image plane to the origin is called focal length *f*. The XYZ coordinate in Figure 4.7 is a camera coordinate.

Image coordinate: defines a 2D coordinate on the image plane to demonstrate the projection of the object. Its origin is the projection of the principal axis. The UV coordinate in Figure 4.7 is the image coordinate.

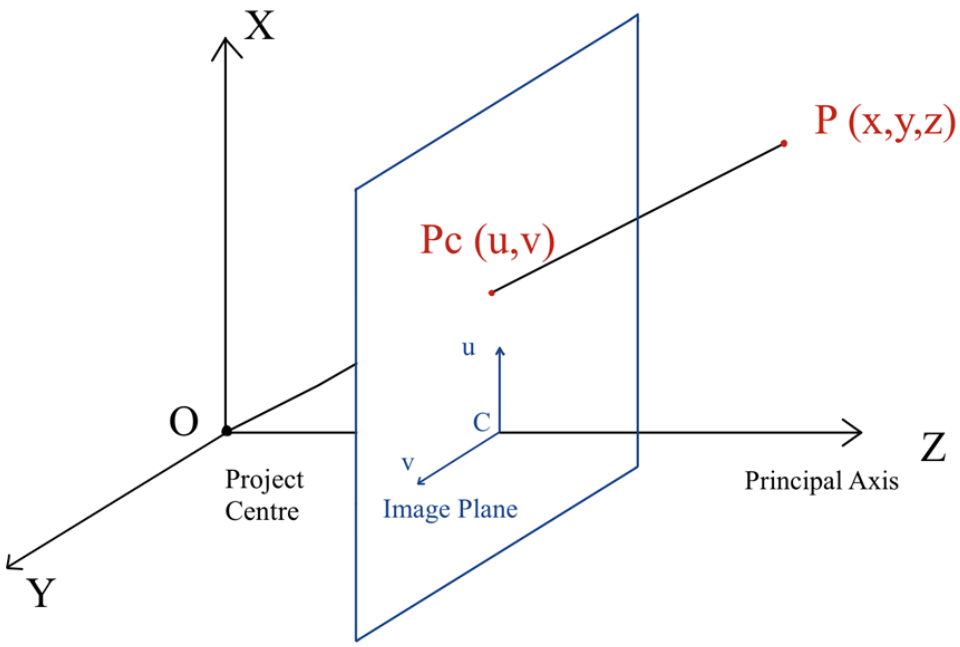


Figure 4.7 Forward Projection onto Image Plane

Pixel coordinate: the pixel coordinate builds up a relationship between the physical unit and the pixel unit. The origin is usually defined at the top left corner of the image and the pixel indices are used to describe locations. The pixel indices increase along the downwards of rows and from left to right in columns.

The derivation of coordinate transformations is explained as follows.

a) Translation

2D transformation is the simplest model of transformation and it constitutes from rotations and translations.

A translation moves a point to a different position on the 2D plane, and the new coordinate defines at the position by simply adding the translation coordinate (t_x, t_y) to the original coordinate (Figure 4.8).

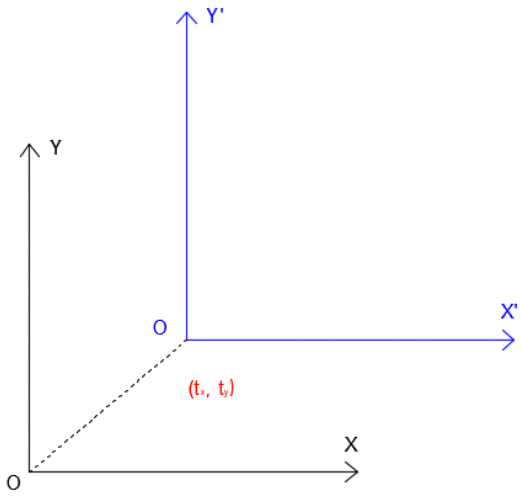


Figure 4.8 Translation of 2D Coordinate

b) Rotation

If there are two 2D coordinates XOY and X'OY', and XOY rotates at the origin with an angle θ (Figure 4.9). P is a point in XOY at (x, y) and in X'OY' at (x', y'). Assuming $x' = r\cos\theta$, $y' = r\sin\theta$ and $x = \cos(\theta + \varphi)$, $y = \sin(\theta + \varphi)$. Then, we have $x' = x\cos\theta - y\sin\theta$, $y' = x\sin\theta + y\cos\theta$. It can be written as a matrix transformation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

The matrix $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ is called the rotation matrix and represents how the coordinate rotates with the origin.

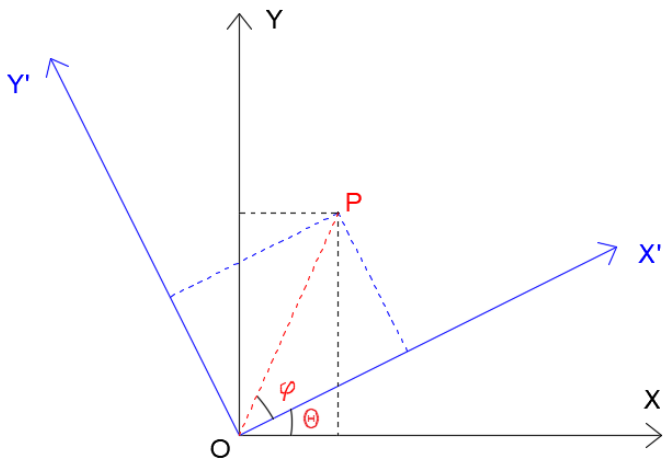


Figure 4.9 Rotation of 2D Coordinate

If we change the situation into 3D space, and the original coordinate rotates along with the Z-axis (Figure 4.10), the rotation matrix is

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

And if the coordinate rotates along the Y-axis and Z-axis respectively, the rotation matrix can be:

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

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

A general rotation is a multiplication of all types of rotations:

$$R = R_Z(\alpha)R_Y(\beta)R_X(\gamma) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}$$

$$R = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix}$$

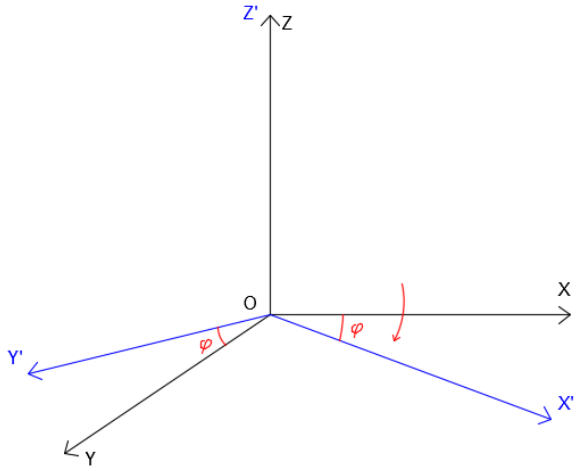


Figure 4.10 Rotation of 3D Coordinate

c) Scaling

Scaling is simply enlarging or shrinking the grid of coordinate. In a scaling process, all points on the object will be multiplied by a scale factor in every direction. If the scaling factor is 1, the object will not change. The scaling matrix is:

$$S = \begin{bmatrix} a_x & 0 & 0 \\ 0 & a_y & 0 \\ 0 & 0 & a_z \end{bmatrix}$$

4.4.2. Homogenous Coordinates

As a general of thumb, two parallel lines in a plane do not intersect forever in Euclidean space. However, things change in the projective space as the parallel lines get closer to each other when they go further away from the observer's eyes and meet at the infinite end on the project plane. Therefore, an infinite point at (∞, ∞) is meaningless in the Euclidean space.

A homogenous coordinate is introduced to solve the infinite intersection issue in the projective space. It describes a N-dimension point in a $N + 1$-dimension. By adding a non-zero scalar w at the end of a 2D point (x, y) or a 3D point (x, y, z), the point is represented by (x, y, w) or (x, y, z, w) respectively. Homogenous coordinate and Cartesian coordinate are convertible by adding or dividing by the scalar. For instance, a 3D point P is presented as $(\frac{x}{w}, \frac{y}{w}, \frac{z}{w})$ in Cartesian coordinate while it is (x, y, z, w) in homogenous coordinate. Since w is usually non-zero, when it becomes 0, the point moves to infinity as $\left(\frac{x}{0}, \frac{y}{0}, \frac{z}{0}\right) \approx (\infty, \infty, \infty)$.

An important property of the homogenous coordinate is that it is scale-invariant. This is because for any point (xa, ya, za) correspond to the same Euclidean point (x, y, z).

A homogenous coordinate has a broad application in 3D computer vision, so it is necessary to explain it in an independent part.

4.4.3. Coordinate Transformation to Digital Image Space

a) World Coordinate to Camera Coordinate

To convert a point in the 3D world coordinate into digital pixels, we need to implement various coordinate transformations. Any 3D linear transformation can be a combination of the three fundamental transformations: translation, rotation and scaling and can be represented by as:

$$T = \alpha R + d$$

Where T is the transformation matrix, α is the scaling factor, R is the rotation matrix, and d is the translation matrix.

The camera coordinate is on the camera and its position relative to the world coordinate changes when we move the camera to adjust the viewpoint, but it is consistent with the camera locally. As mentioned above, the camera coordinate defines at the camera's centre of projection and its Z-axis is the principal axis and vertical to the image plane.

A world coordinate can be converted to the camera coordinate by the linear transformation, and the point in the world coordinate (x_w, y_w, z_w) is converted to homogenous camera coordinate (x_c, y_c, z_c) as:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = sR \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} + T$$

b) Camera Coordinate to Image Coordinate

When we convert the matrix into the local coordinate (camera coordinate) from the world coordinate, the next conversion happens on the projection process, which projects a 3D point onto a 2D image plane. Assuming the origin of the image coordinate locates at the perpendicular foot of the principal axis on the image plane, and two mutually perpendicular axes are defined on the image plane.

Supposing we have a rectangle $P_1P_2P_3P_4$ in the 3D space and it has been converted into the camera coordinate from the world coordinate. It projects on the image plane as $p_1p_2p_3p_4$. The focal length of the camera is f and the camera coordinate defines as OXYZ (Figure 4.11).

If no image distortion exists, the mapping should be linear, and lines in the 3D space are still lines on the image plane. Therefore, the line P_2P_3, line p_2p_3 and their projection lines combine a pair of similar triangles. As the Z-axis of the camera coordinate is perpendicular to the image plane (principal axis), the distance between the origin of the camera coordinate and the projected point p_3 is the focal length f. Also, the distance between the P_3 and the origin O is Z_3, and the length of P_2P_3 is Y_3. Thus:

$$\frac{l_{p_2p_3}}{l_{P_2P_3}} = \frac{y_2}{Y_2} = \frac{f}{Z_3}$$

If we know where the 3D points are, it is easy to calculate their positions on the 2D image plane. The mathematical expression is:

$$x = \frac{fX}{Z}$$

$$y = \frac{fY}{Z}$$

$$z = f$$

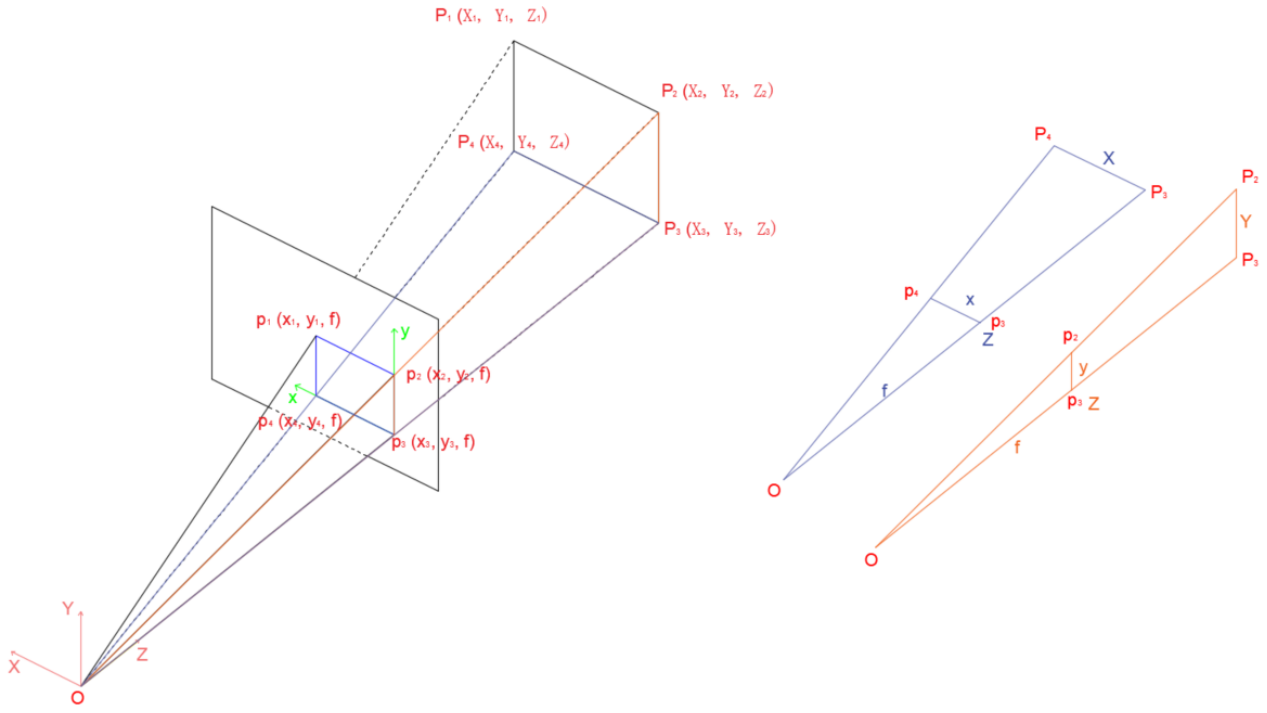


Figure 4.11 Projection Relationship.

World coordinate and the image plane

This mapping from 3D to 2D is projective transformation and can be expressed into the projective transformation matrix form as:

$$z\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

c) Image Coordinate to Pixel Coordinate

Thousands of pixels constitute a digital image. If we choose a pixel as the unit of the coordinate on the image plane, a dot on a digital image is a pixel, and its coordinate is expressed by integers because pixels are discrete. In general, the origin of this coordinate locates at the upper-left corner and its X-axis is towards the right while Y-axis points downwards. Since an image size is not infinite, the 3D scene demonstrated on the image plane is cut off according to the image shape and size. The point converted from projective transformation is not what we see in the real digital image. Besides, optical sensors introduce non-linear mapping like distortion into the imaging process, which needs to be considered into integral coordinate transformation.

As the origin of the image plane is usually at the image centre where the principal axis intersects with the image plane, while the origin of the pixel coordinate is the upper-left corner, the two origins are offset by a translation transformation. Assuming the origin of the image plane is O and the origin of pixel coordinate is O_1 and it can be expressed as (u_0, v_0) in the image coordinate (Figure 4.12), so the mapping is:

$$u = \frac{x}{d_x} + u_0,$$

$$v = \frac{y}{d_y} + v_0$$

where d_x and d_y are pixel dimensions along the X-axis and Y-axis respectively.

If we convert it into homogenous formation:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{d_x} & 0 & u_0 \\ 0 & \frac{1}{d_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

If we multiply the projection matrix with the image-to-pixel matrix and define

$$M = \begin{bmatrix} \frac{f}{d_x} & 0 & u_0 \\ 0 & \frac{f}{d_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

M is an internal matrix as it only relates to the internal parameters of the camera. Conversely, the external matrix describes the camera position including rotation and translation and it is written as $\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$.

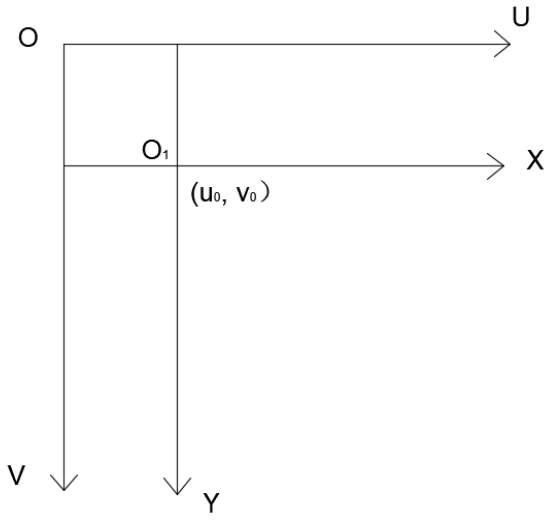


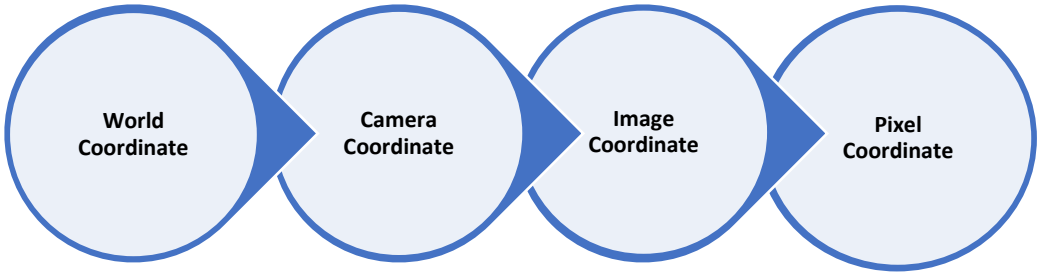
Figure 4.12 Relationship Between Image Coordinate and Pixel Coordinate

The model of distortion mentioned previously corrects the image to some extent, so if we correct the pixel positions by considering the distortion, the coordinate transformation can be more accurate.

Since the distortions are usually introduced by a lens or other optical components, it should be calculated when the camera coordinate converts to the image coordinate.

The overall process of transformation can be depicted as:

$$\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1/d_x & 0 & u_0 \\ 0 & 1/d_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$



The process to estimate these parameters is camera calibration. The camera intrinsic matrix does not change with the positions of the camera as long as the focal length is consistent.

4.5. DLT

Direct Linear Transformation (DLT) is a classical method to determine parameters of 3D reconstruction from two different views. If there are two cameras in one space, we define the image plane of the left camera is $U_L V_L$ while the right image plane is $U_R V_R$ and the 3D coordinate is XYZ (Figure 4.13). One point in the XYZ coordinate is at (x, y, z) and its projections in $U_L V_L$ and $U_R V_R$ are (u_L, v_L) and (u_R, v_R) respectively. If we know the values of x, y, z, u_L, v_L, u_R and v_R and do not consider optical distortion

errors at this moment, according to the previous derivations, we define the transformation from XYZ to $U_L V_L$ as:

$$\begin{bmatrix} u_L \\ v_L \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Then

$$u_L = \frac{L_1 x + L_2 y + L_3 z + L_4}{L_9 x + L_{10} y + L_{11} z + 1}$$

$$v_L = \frac{L_5 x + L_6 y + L_7 z + L_8}{L_9 x + L_{10} y + L_{11} z + 1}$$

$$u_R = \frac{R_1 x + R_2 y + R_3 z + R_4}{R_9 x + R_{10} y + R_{11} z + 1}$$

$$v_R = \frac{R_5 x + R_6 y + R_7 z + R_8}{R_9 x + R_{10} y + R_{11} z + 1}$$

Since we have 7 knowns and 22 unknowns from these four equations, we need at least 22 equations to solve the unknowns as the number of equations should be larger than the number of unknowns. If we have at least 6 different object points and they will introduce 24 equations with the same unknowns, then we can determine the Rs and Ls. More points can provide better calibration results. If we stack these 2n equations together and rewrite them into matrix form, we get:

$$\begin{bmatrix} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -u_{L1}x_1 & -u_{L1}y_1 & -u_{L1}z_1 \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -v_{L1}x_1 & -v_{L1}y_1 & -v_{L1}z_1 \\ x_2 & y_2 & z_2 & 1 & 0 & 0 & 0 & 0 & -u_{L2}x_2 & -u_{L2}y_2 & -u_{L3}z_3 \\ 0 & 0 & 0 & 0 & x_2 & y_2 & z_2 & 1 & -v_{L2}x_2 & -v_{L2}y_2 & -v_{L2}z_2 \\ & & & & & & \vdots & & & & \\ x_n & y_n & z_n & 1 & 0 & 0 & 0 & 0 & -u_{Ln}x_n & -u_{Ln}y_n & -u_{Ln}z_n \\ 0 & 0 & 0 & 0 & x_n & y_n & z_n & 1 & -v_{Ln}x_n & -v_{Ln}y_n & -v_{Ln}z_n \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \\ L_5 \\ L_6 \\ L_7 \\ L_8 \\ L_9 \\ L_{10} \\ L_{11} \end{bmatrix} = \begin{bmatrix} u_{L1} \\ v_{L1} \\ u_{L2} \\ v_{L2} \\ \vdots \\ u_{Ln} \\ v_{Ln} \end{bmatrix}$$

The matrix of points in the right image plane is in a similar formation.

A constraint of these points is that they must be non-planar, and this is the reason we designed a wooden calibration structure to set the calibration markers. In this system, we use 12 markers to define the points in the workspace. To solve the unknowns in these equations, common ways are using the least square method or Singular Value Decomposition (SVD), and here we chose SVD to calculate the matrix L.

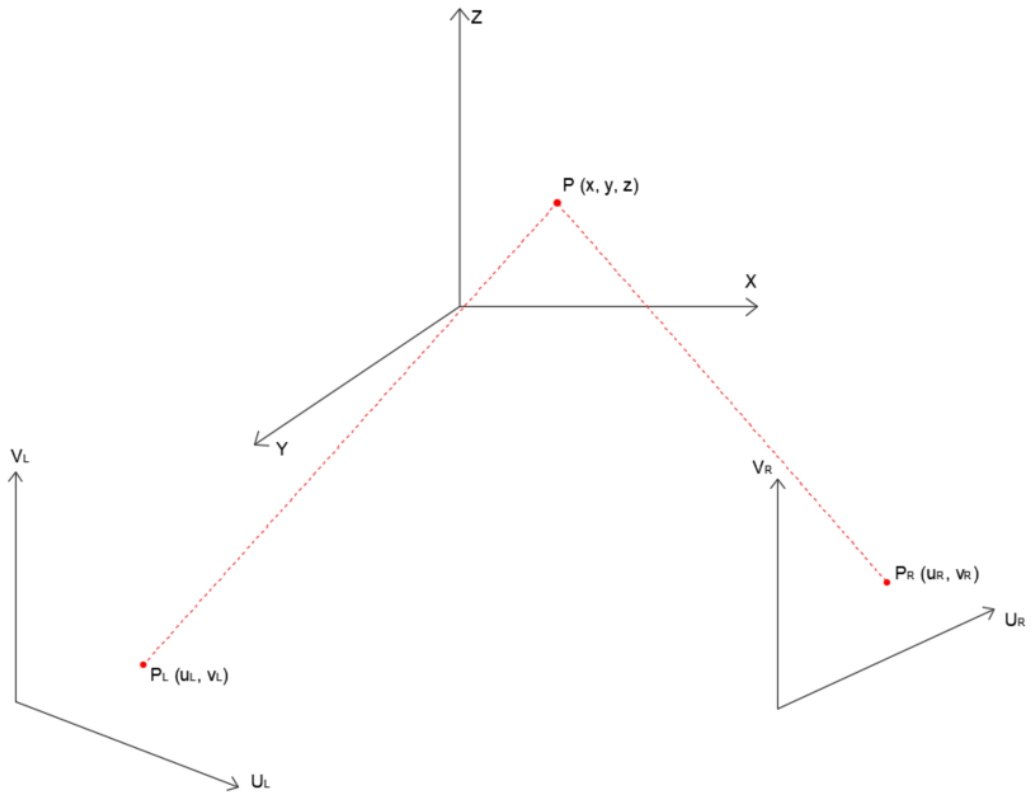


Figure 4.13 Object Space and Two Image Plane Coordinates

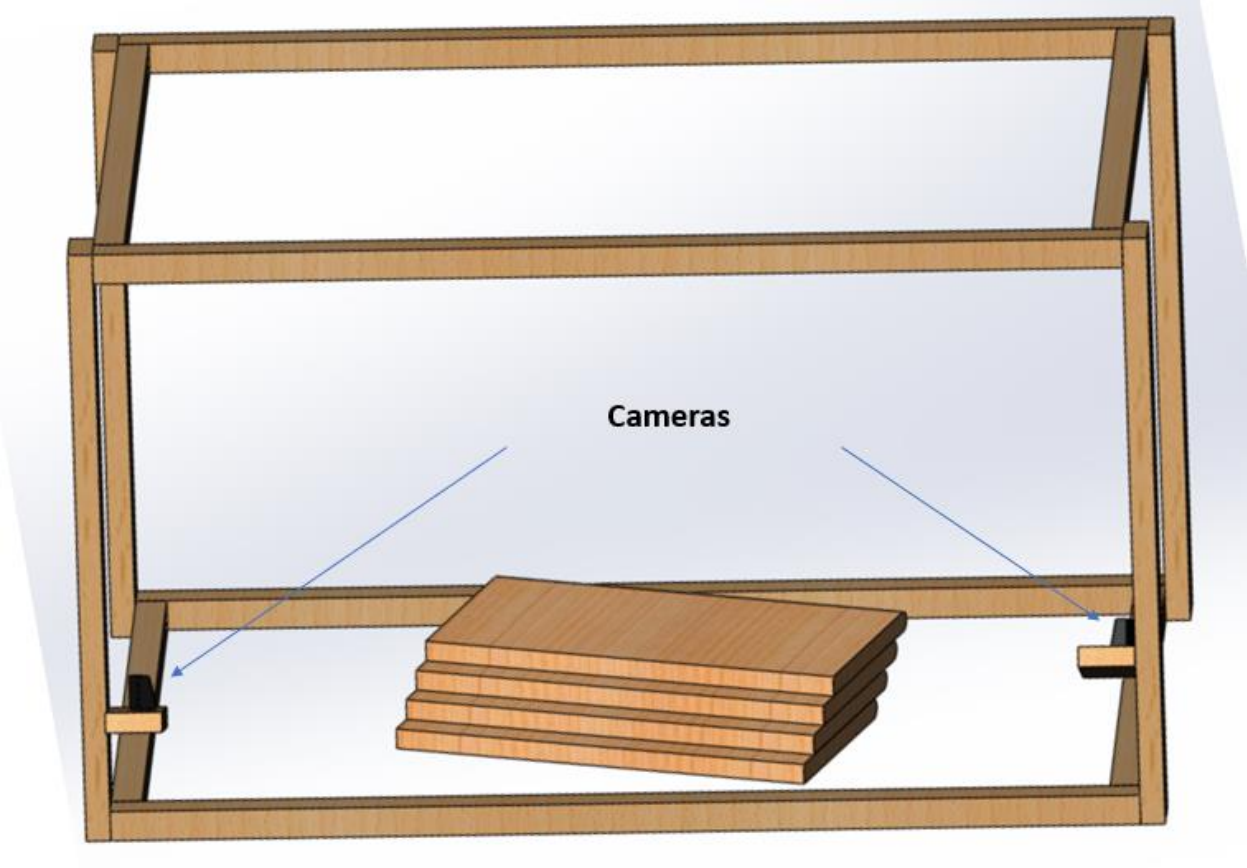


Figure 4.14 Layout of Cameras and the Calibration Frame

Camera calibration in this project used the 12 calibration markers fixed on the calibration frame and their positions were measured by the OptiTrack system. These 3D positions are known values (x_w, y_w, z_w) in the world coordinate system. The background in the field of view was filtered out by colour (Figure 4.15). The position of each marker is represented by the pixel index on the image and then we can obtain camera parameters from the DLT method with known (x_w, y_w, z_w), (u_L, v_L) and (u_R, v_R).

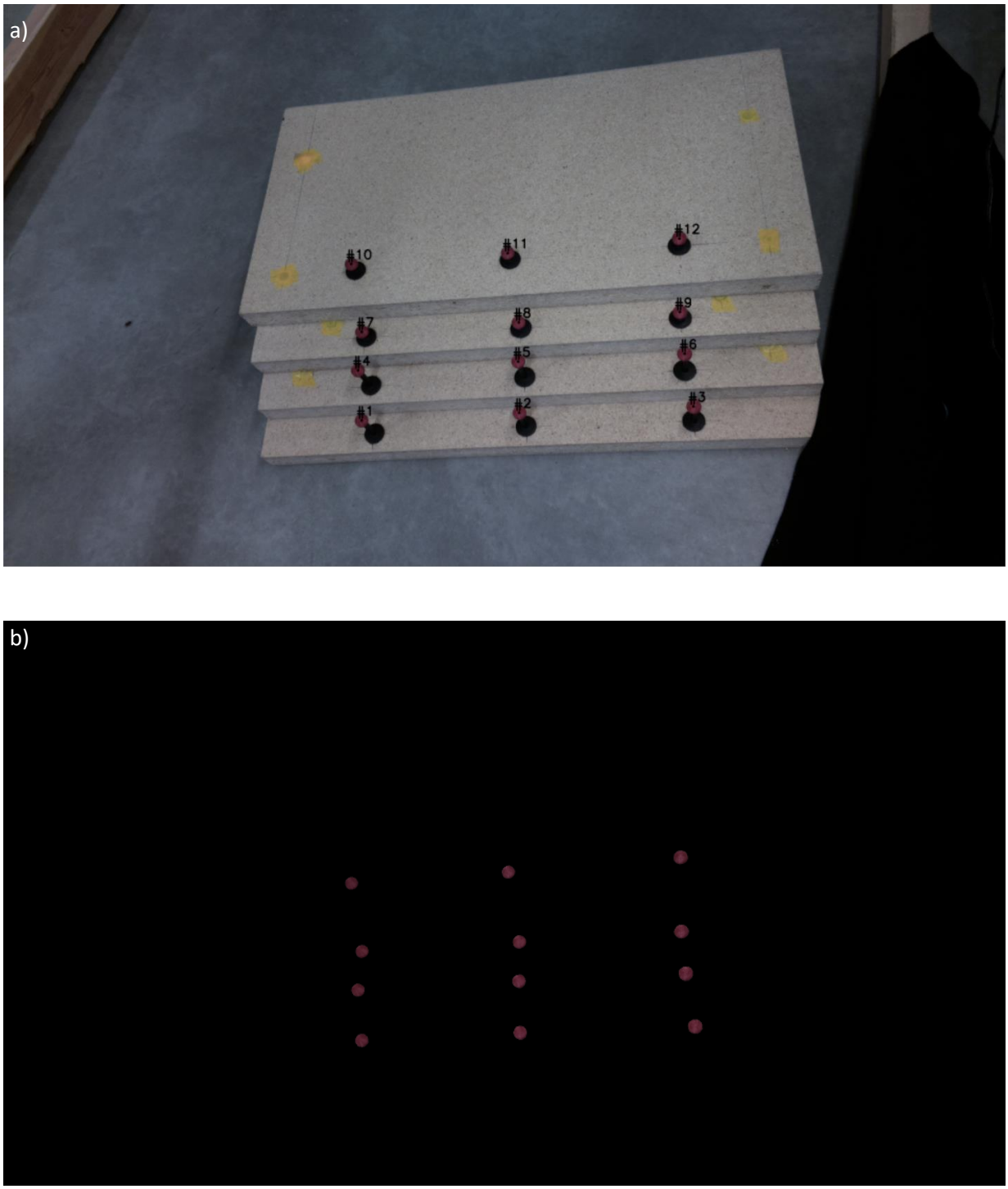


Figure 4.15 Background Removal

a) Calibration frame captured by the left camera (resolution: 1920 x 1080).

b) Filtered calibration markers.

4.6. 3D Reconstruction

Now we have all the camera parameters we need, and the next step is to reconstruct 3D points based on the images we get from cameras. Back to the equations

$$u_L = \frac{L_1 x + L_2 y + L_3 z + L_4}{L_9 x + L_{10} y + L_{11} z + 1}$$

$$v_L = \frac{L_5 x + L_6 y + L_7 z + L_8}{L_9 x + L_{10} y + L_{11} z + 1}$$

$$u_R = \frac{R_1 x + R_2 y + R_3 z + R_4}{R_9 x + R_{10} y + R_{11} z + 1}$$

$$v_R = \frac{R_5 x + R_6 y + R_7 z + R_8}{R_9 x + R_{10} y + R_{11} z + 1}$$

Now, parameters L_1 to L_{11} and u_L, v_L, u_R, v_R are known, and the object converts to calculate x, y, z. Then we have:

$$\begin{bmatrix} L_1 - L_9 u_L & L_2 - L_{10} u_L & L_3 - L_{11} u_L \\ L_5 - L_9 v_L & L_6 - L_{10} v_L & L_7 - L_{11} v_L \\ R_1 - R_9 u_R & R_2 - R_{10} u_R & R_3 - R_{11} u_R \\ R_5 - R_9 v_R & R_6 - R_{10} v_R & R_7 - R_{11} v_R \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} u_L - L_4 \\ v_L - L_8 \\ u_R - R_4 \\ v_R - R_8 \end{bmatrix}$$

and it is simple to obtain x, y, z.

In the tracking process, if we have the positions of markers on the target tool from both left and right cameras (known (u_L, v_L) and (u_R, v_R)), it is convenient to reconstruct the 3D information of markers in the world coordinate. Besides, we can also reconstruct the 3D positions of calibration markers to verify the accuracy of the DLT method.

4.7. Multi Cameras Connection

As we need to track the surgical tool in real-time, it is essential to get the tracking information from each camera with minimum delay, and two cameras in the system should be synchronized. Only when the images captured by each camera at the same moment, the markers in the image could be corresponding to each other from different camera coordinates.

In the beginning, I tried to transmit the live video streaming and searched lots of methods to realize that. However, if we received the whole video streaming (i.e. complete video data for each image frame) from each Raspberry Pi continuously, it would cause tremendous calculations and result in low-speed transmission to the central computer. Also, lots of data in the image is useless. There is always an evident lag during transmission, and we have to sacrifice the video quality to pursue a better transmission speed. This violates our target of realtime tracking. Therefore, instead of transmitting every pixel on the image, we resolved to only extract the position of each marker on the image plane and transmit it to the central computer. If we calibrate the cameras before the tracking, we can reconstruct the real-time 3D positions of the markers on the tool by simply use the local 2D positions from the right and left cameras.

Since Raspberry Pi is a single-board computer and it has an independent processor that supports its computing capability, it can abstract the data we need rather than merely send all data it collects to the central computer and wait for processing. This feature greatly lessens the burden of the central computer and is better for real-time data processing compared to USB cameras. However, the computing capability of Raspberry Pi is still limited, so its processing speed is much slower than normal computers especially when the calculation is complex. Therefore, if we assign some simple image processing to it, we can realize real-time tracking in this system.

The Raspberry Pi camera module captures movements of the surgical tool and each marker should be identified and extracted. The pixel index of the centroid of each marker represents its current location, and the method we extract the centroid of the detected area is to use a function cv2.moment() (Appendix B) in OpenCV (Bradski 2000). This function can help find the centre of arbitrary shape. Concretely, the area we detect is a blob, which is a group of connected pixels with similar properties. As we know the centre of an object is the arithmetic mean of all the points on it, which is:

$$c = \frac{1}{n}\sum_{i=1}^{n} x_i$$

If we implement it into an image, then each point is represented by a pixel and the problem converts to find the weighted average of all pixels in a blob. The image moment is a specific weighted average of pixel intensities, so we can obtain the centre of a blob by:

$$c_x = \frac{M_{10}}{M_{00}}$$

$$c_y = \frac{M_{01}}{M_{00}}$$

We use these centroids to represent each marker. Then, all the position information would be packed as a matrix and sent to the central computer. Each frame will send one matrix and the computer accepts 60 4 x 2 matrices each second, which is a relatively low-intensity work for the computer to take over.

The calibration and reconstruction results were verified by offline video recordings. As the data were collected by each camera respectively, I could only capture a video and save the data into the buffer instead of sending it immediately to the central computer to process. Thus, the next step was to transmit information instantly between the two Pi cameras and the computer to achieve the real-time tracking goal.

4.8. Socket: Data Communication Between the Laptop and Raspberry Pis

Data transmission between the Raspberry Pis with the computer is a critical step. Ideally, we need a reliable, stable and delay-minimized passageway for our devices to communicate. Since we have more than one camera, the problem becomes more complicated as this is not a one-on-one communication. Therefore, we chose Network Socket (Oracle Corporation) to handle this issue.

Socket combines with a server and clients, and it is an internal endpoint that supports receiving and sending data within a computer network. The server bounds a designated port and waits for a client to send a connection request. The server in this system supports multiple-client connections. There is a hostname of the computer where the server is running and a port number to help clients identify the server where they will send the connection request to. The client also has a local port number so that the server can identify it and vice versa. I tried to use Wi-Fi to set up the socket connections but there are lots of issues coming with the university Wi-Fi as it uses Wi-Fi Protected Access 2 (WPA2), which is a security method for wireless networks. Therefore, Pi cannot merely connect to it as normal individual-use Wi-Fi. In the end, we used a router to build up a Local Area Network (LAN) between the two Raspberry Pis and the central computer. Data from the “left camera” (the Pi set at the left side of the frame) was assigned a label “left” while data from the right was assigned a label “right”, so the data from different cameras would not be confused to the computer. The reconstruction calculation only starts when the server receives data from both cameras and the difference between the times when the server receives the data from the first and second client should be less than 0.2 seconds, which ensures the left and right position data used for reconstruction are collected at the same moment.

4.9. 3D Reconstruction of the Tip of the Probe

4.9.1. Dimensions of the Measurement Probe

OptiTrack could not provide an accurate dimension of the measurement probe for us, so we had to measure the distances between markers on the probe by ourselves. We used both ImageJ and a calliper. Before measurement, we unscrewed all markers from the threaded-shaft and defined the distances between the centres of the shafts as the distances of the corresponding markers. By using ImageJ, we placed the probe on the flat table and took pictures of it from its facade, front and flank respectively. Then, we used the measurement tool from ImageJ to measure the distance between each marker. The dimensions are in Table 2 (defined the tip of the probe as the origin, and the local coordinate is shown in Figure 4.16).

Table 2 Measured Positions of Markers on the Probe (ImageJ vs. Calliper)

	Measured Position by ImageJ [mm]			Measured Position by Calliper [mm]		
Marker	X	Y	Z	X	Y	Z
1	0.00	112.77	12.70	0.0 ± 0.00	126.6 ± 0.47	16.6 ± 0.31
2	0.00	164.18	12.70	0.0 ± 0.00	177.3 ± 0.17	16.6 ± 0.31
3	-30.02	212.83	12.70	29.9 ± 0.18	226.4 ± 0.23	16.6 ± 0.31
4	29.01	247.84	12.70	-29.4 ± 0.21	262.1 ± 0.10	16.6 ± 0.31

Next, we measured the distances again physically using a digital calliper.

As the tracking accuracy was higher by using the dimensions measured by callipers, all tracking results in the thesis are based on the dimension measured by the calliper.

4.9.2. 3D Reconstruction of the Tip of the Probe

After reconstructing the 3D positions of the markers on the calibration frame, we can get the position of the endpoint of the tool. As shown in Figure 4.16, the markers are labelled as No. 1, No. 2, No. 3 and No. 4 respectively, and if we define a local coordinate on the tool and set the origin at the tip, the position of each marker should be $(0, 126.617, 16.553)$, $(0, 177.277, 16.553)$, $(-29.88, 226.424, 16.553)$ and $(29.403, 262.097, 16.553)$ respectively (unit: mm). As the tool is a rigid body, relative positions on it are fixed. When the positions of markers in the world coordinate are known, we can get the transformation matrix between the tool coordinate and the world coordinate, then convert the tip from the tool coordinate to the world coordinate.

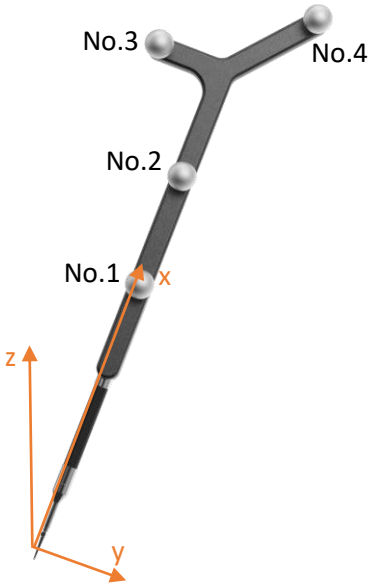


Figure 4.16 OptiTrack Digitizing Probe

4.10. Marker Projection on The Artificial Hip CT Images

The final step is projecting an image of the tool on the virtual fluoroscopy image that we created earlier. For this project, we used the tracking probe as an analogue for an arthroscopy surgical tool. We used the positions of the tip and the No. 1 marker to represent the tool, and the tool is drawn as a line on the fluoroscopy image. Here we can use the projection matrix we derived previously to convert the 3D positions into 2D. As the projection perspective alters, the tool will be overlaid at different places on the artificial hip to replicate the C-arm function during arthroscopic surgical training (Figure 4.17).

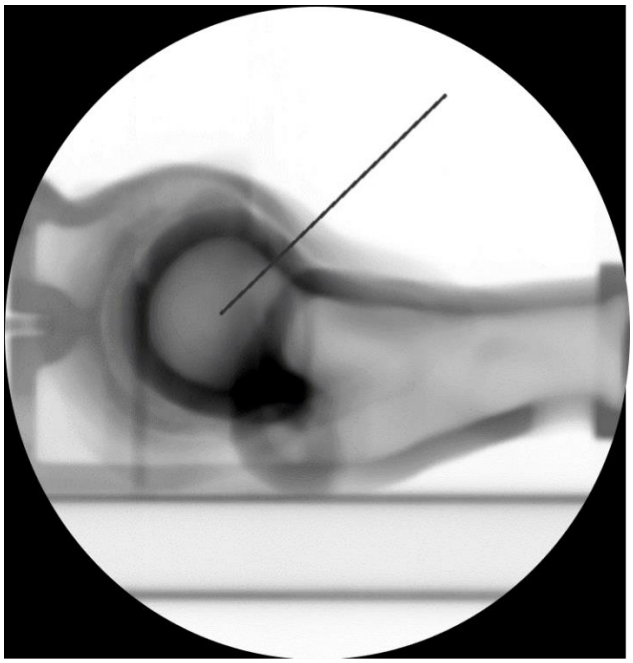


Figure 4.17 The Final Rendering of Virtual Fluoroscopy

In the current projection shown in Figure 4.17, the tool appears as a straight line. The tool is assumed to be a rigid 3D object that will maintain a straight vector from the tracked markers attached to the tool.

Ultimately, the project tool image can be made to look similar to the actual surgical tool so that it appears as though the surgeon trainee is able to see a real-time image of the arthroscopic tool.

4.11. Programming

All programs either on Raspberry Pis and the central computer were written in Python. The programs support all functions include detection, transmission, real-time tracked tool display, etc. The integral codes are attached in Appendix B.

4.11.1. Raspberry Pi Programming

First, the Raspberry Pi initiates its camera module, and the camera starts to capture live video. Then the program will detect the markers by filtering the area in a specified HSV range. An OpenCV function findcontours() will extract the contour of the detected marker. If there is no object detected, the program will give a prompt that "No Marker Detected". The centroid of the marker will be calculated based on the contour and represented by the pixel indexes on the image. If all four markers are detected, their locations will be packed as a 4 x 2 matrix and sent to the central computer (server). The order of the positions is consistent with the label of markers (green, yellow, blue, red).

The socket builds up connections between the central computer (server) and the Raspberry Pis (clients). Each client has a unique port number. The server can identify where the data come from by recognizing the port numbers.

4.11.2. Central Computer Programming

The central computer receives and reconstructs the position information from Raspberry Pis and draws the surgical tool on the virtual fluoroscopy. Before building up the socket connection, the Raspberry Pi sends a request to the central computer with its distinctive port number. The central computer will label the data from this Pi by its port number as ‘right ‘or ‘left’. Only if the central computer receives data from both Raspberry Pis, and the difference between their timestamp is less than 0.2 seconds, the data will be considered as valid and used to the future computations. To ensure the data is newly received, the timestamp of the current data should be different from the previous one. As the data from one Pi is a 4 x 2 matrix, with the known relative positions of the four markers in the tool local coordinate, the program will reconstruct the tip of the tool and markers by calling the function DLT.

The 3D positions of the reconstructed tip and the marker No. 1 (green marker) will be used as a start point and the endpoint to draw a line to represent the tool on the virtual fluoroscopy. Since the fluoroscopy is 2D, the 3D positions will be reprojected again based on the user’s angle of view. The line will move with the tool movements in the real world and be displayed on the virtual fluoroscopy in realtime.

Chapter 5. Accuracy and Repeatability Measures of Raspberry Pi Camera System

Accuracy and repeatability were analyzed by testing the system to verify the possibility of using Raspberry Pi camera modules in the optical tracking system. Static jitter and tracking latency will also be discussed in this chapter.

5.1. Accuracy

Since we used the OptiTrack optical tracking system to measure the positions of markers on the calibration frame, we believe one of the final errors came from this process. According to the system's calibration results based on the OptiTrack measurements shown in Chapter 3, the mean 3D error was approximately 0.86 mm and the mean tip error of the digitalizing probe was around 1.67 mm.

We tested the accuracy of the system in the same way we did in the measurement process. Since the dimension of the thread on the marker base is M4 (diameter of the threaded shaft is 4 mm), and any spot on the end of the bolt is possible to be pointed at and be considered as the position of the marker, it is difficult for us to determine if we pointed at the same spot every time. Another realistic problem in the testing is that the tip of the probe might slip on the contact surface as the surface is not flat and is slightly convex. To minimize the possible errors in the test, we used some nuts to screw on the marker bases because the upper surface of the nut is slightly higher than the upper surface of the base, which forms a tiny notch on the target surface and reduces movements during contact. We measured the position of the OptiTrack probe touching the tips of marker stems No.1, No.5, No.7, No.8, and No.9 respectively (Figure 5.4) with the resolution of 1920 × 1080 pixels at a frame rate of 25 fps (25 frames per second). We pointed at each marker base at least three times and there were at least 50 frames captured each time.

Then we got the average and standard deviations of these values as the result. The errors in each axis-direction are demonstrated in the graphs below:

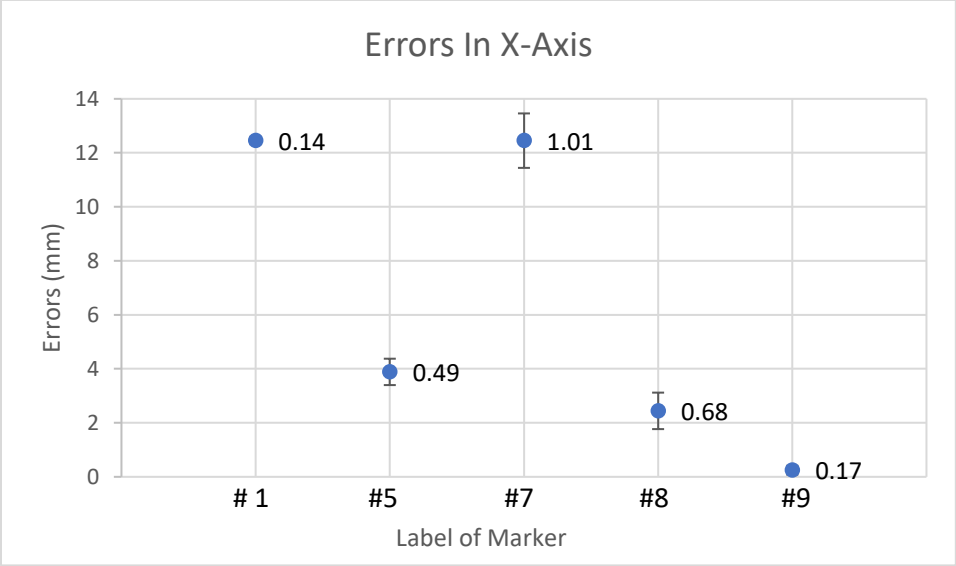


Figure 5.1 Errors in X-Axis

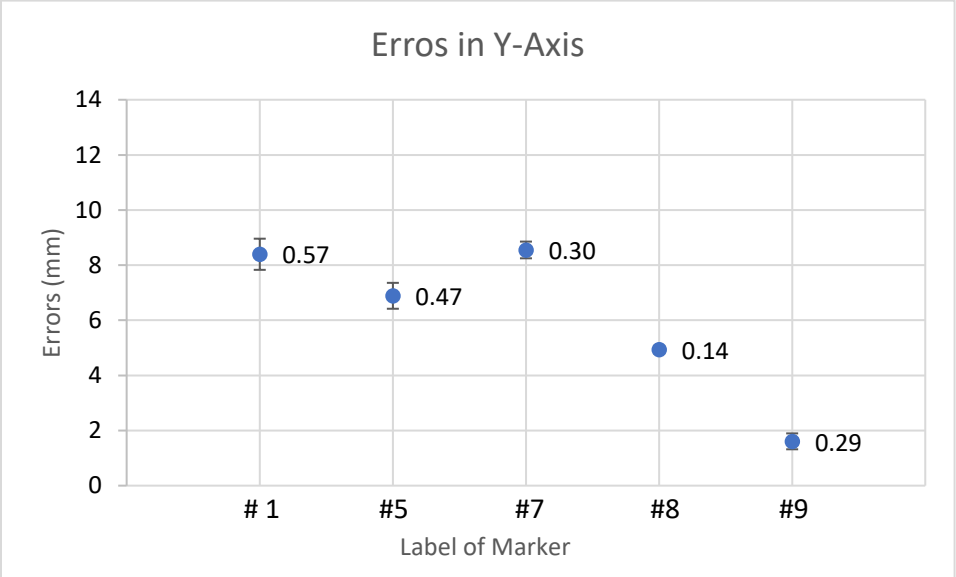


Figure 5.2 Errors in Y-Axis

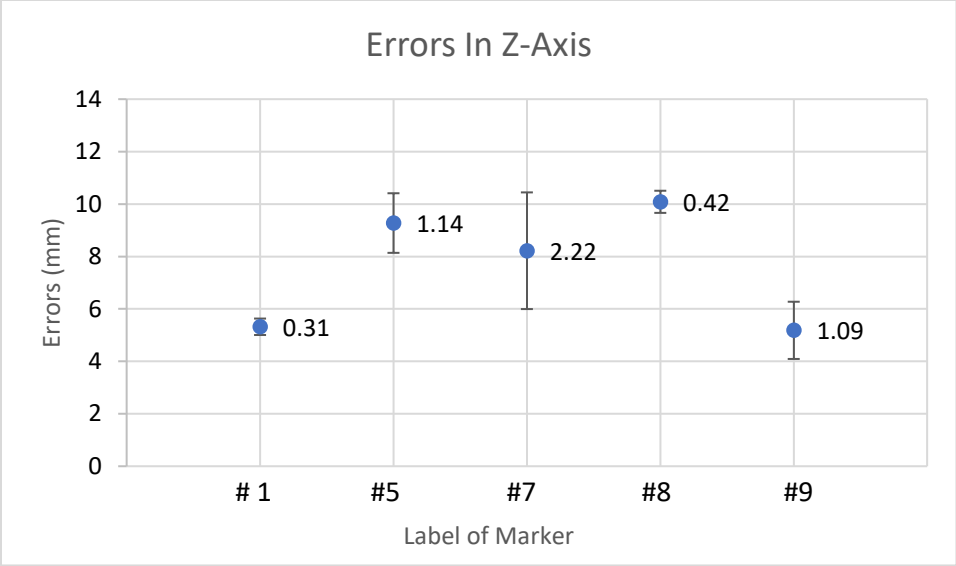


Figure 5.3 Errors in Z-Axis

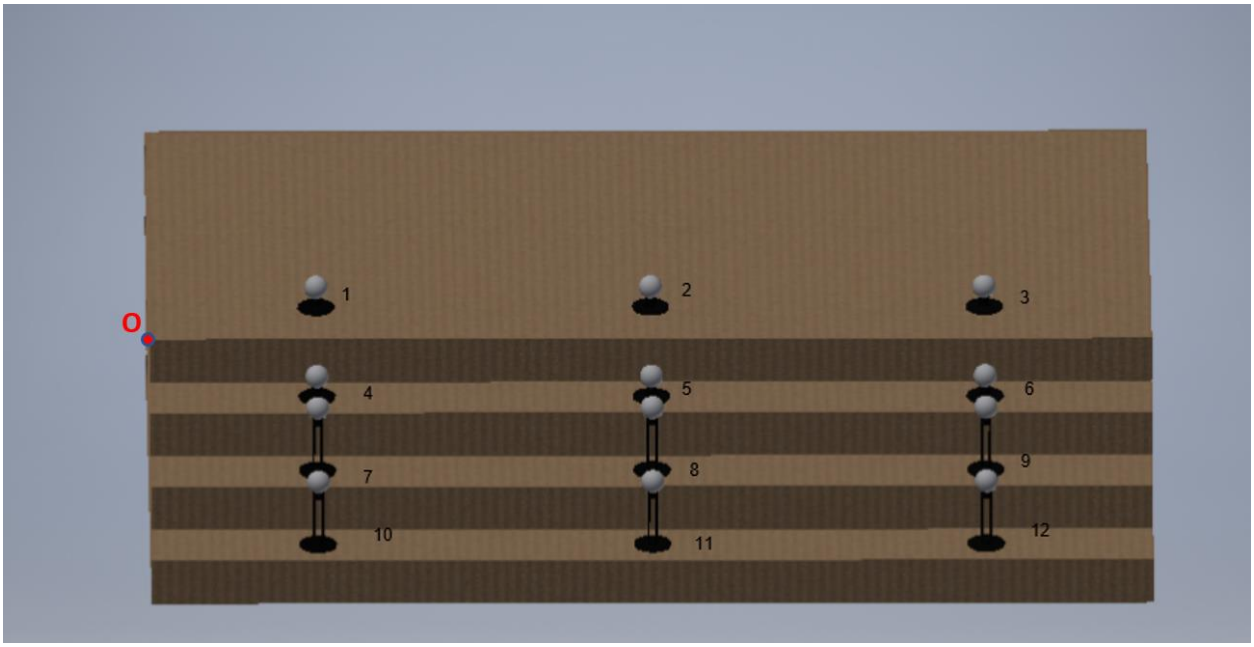


Figure 5.4 Labeled Markers

Standing on the layers with the origin 'O' of the world coordinate

The largest errors in the three directions were 12.46 mm, 8.55 mm, 10.09 mm respectively, while the standard deviations were more stable ranging from 0.14 mm to 2.22 mm. The errors in the Y-axis were relatively smaller than in other directions. This is because the height of all marker bases on the same layer was fixed, and it caused the least errors in the testing and calibration process.

The DLT camera calibration process also caused some errors because camera imaging and object projection are not perfectly linear. Since the calibration process requires known 3D positions of an object, the DLT errors were acquired by calculating the difference between the reconstructed positions of calibration markers and the known positions of these markers. The errors caused by the DLT calibration process were 1.83 mm in the left camera and 1.87 mm in the right camera.

Another accuracy test used 960 × 540 pixels with 25 fps video streaming. In this test, we checked the positions of marker No.1, No.2, and No.3 respectively in the same way as the previous tests, but we pointed at the same marker base only one time instead of multiple times. The tracked positions are No.1 (100.93 mm, 11.47 mm, -33.62 mm), No.2 (256.11 mm, 11.16 mm, -43.10 mm) and No.3 (415.63 mm, 7.68 mm, -36.89 mm), and the errors are (13.27 mm, 0.08 mm, 2.37 mm), (11.31 mm, 0.59 mm, 5.79 mm) and (5.55 mm, 3.67 mm, 1.11 mm). The largest error is still the x value of marker No. 1 in this case, and its value is similar to the previous test. The accuracy in the Y-direction reaches the submillimetre level with markers No.1 and No.2. The DLT calibration error was 0.84 mm in the left camera and 1.75 mm in the right.

5.2. Repeatability

Repeatability is another important test as it represents the stability of the tracking system. To avoid the movements and errors that occurred when pointing at the top surface of the marker base shaft, we drilled some small holes on each layer of the wooden calibration base. For testing repeatability, we placed the

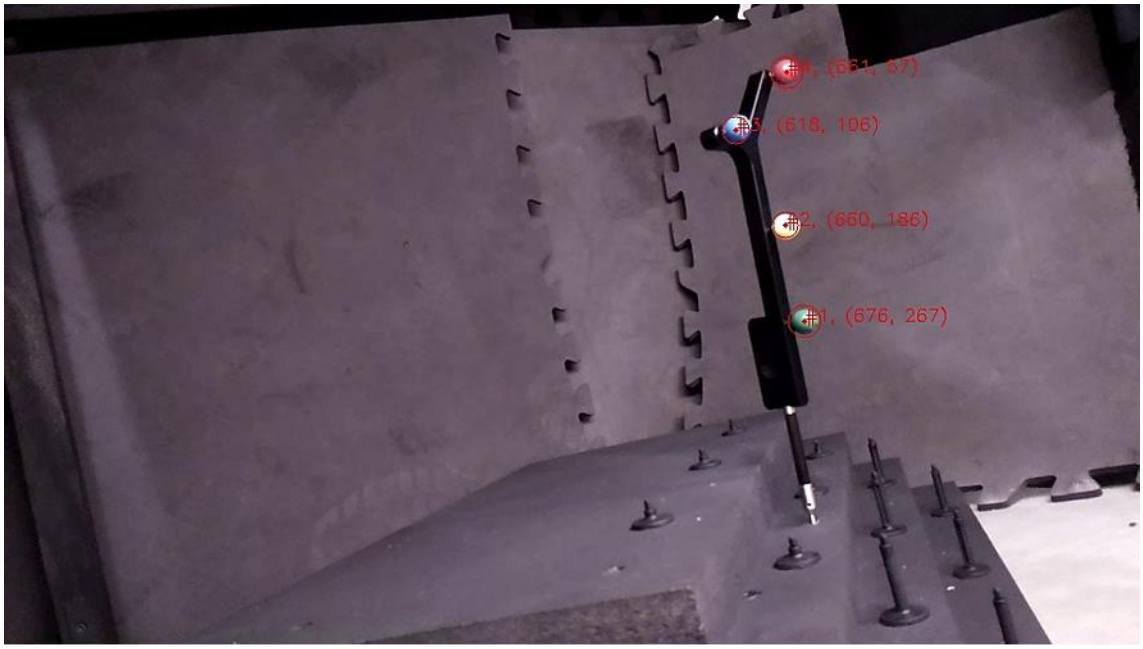
probe into different holes and tracked its tip. The probe was static after being inserted into the hole. There were 3 holes on every layer next to the markers, and they were labelled in the same order as the markers.

We checked the repeatability in two types of tests. In the first test, we inserted the probe into four different holes (No.1, No. 3, No.4, No.5) located at different layers. Then we twisted the probe and the positions of markers changed but the position of the tip was unaltered. It evaluated the ability of the system to locate the tip of the probe as the four markers on the probe changed location. We repeated the operation five times with the same hole, and each time approximately 30 frames were captured (Figure 5.5). The pixel resolution of all tests is consistent at 960 × 540 with 25 fps. The repeatability can be illustrated by the average standard deviation in each hole (Table 3).

Table 3 Measured Standard Deviations of Markers in X, Y, Z axes

Marker Number	Standard Deviations in X	Standard Deviations in Y	Standard Deviations in Z
1	± 3.12 mm	± 2.59 mm	± 1.67 mm
3	± 5.06 mm	± 1.11 mm	± 1.46 mm
4	± 2.41 mm	± 0.61 mm	± 2.85 mm
5	± 5.17 mm	± 1.09 mm	± 3.21 mm

a)



b)

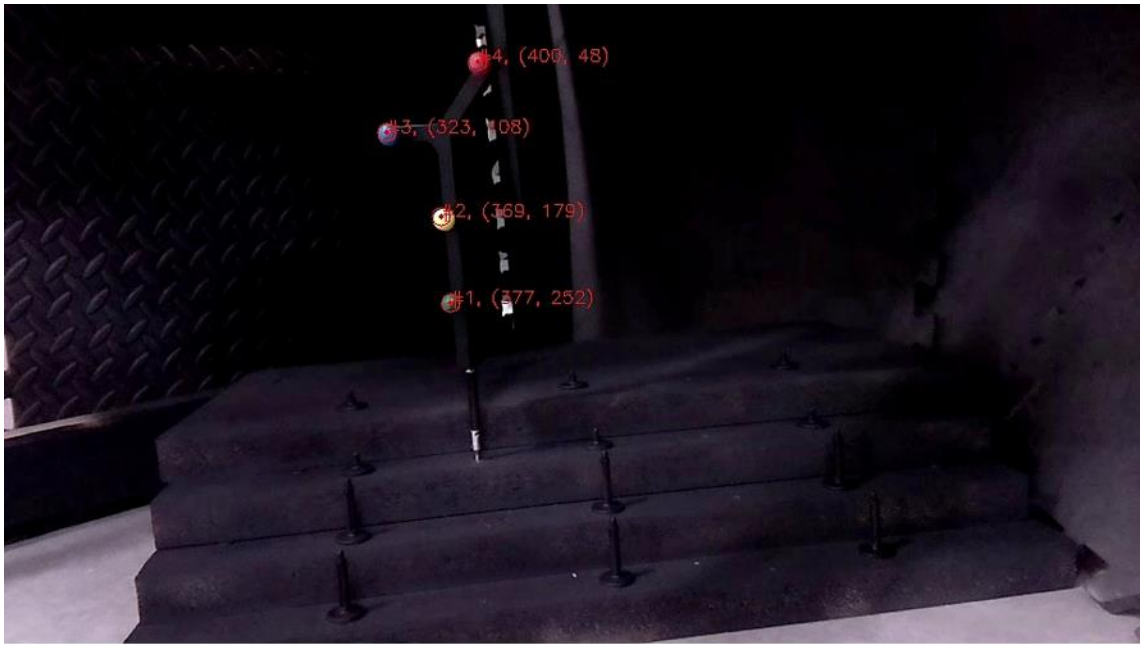


Figure 5.5 View of Camera

a) View in the left camera. b) View in the right camera

In the second test, we only tested hole No.3. The variable in this test was lightness alteration. We turned on and off an LED light installed on the frame where we set the cameras, and the brightness in the tracking space varied. The average standard deviations dropped down to 0.53 mm, 1.05 mm, 0.41 mm, which proved the lightness alternations have subtle effects on the results.

5.3. Static Jitter

There were some jitters between each frame even when the markers were static, which resulted in slight variations between the tracking results. We analyzed the values in each frame from the second repeatability test data, and the standard deviations between frames were in the range of 0.22mm to 0.73mm. Generally, there is a one-pixel difference between frames. Not every frame may be exactly the same, because even minimal lightness variations can result in the change of the marker boundaries the software detects. Also, there is always a delay in the real-time network transmission no matter what device we use, and this inconsistent delay causes jitter as well. Meanwhile, the 3D printed markers have uneven surfaces and their shapes are low in sphericity, which causes the contours of the markers are not accurate enough. Therefore, the centre of the marker the software detects varies slightly with the video streaming.

5.4. Latency

Synchronization of the cameras is crucial in an optical tracking system. If the tracking data collected by the cameras are not synchronized, the tracking result could be incorrect and cause significant errors in the results. Currently, the software of our system outputs reconstructed positions only if the position information was received from both cameras, and the time difference between when the computer

receives the new information from the left Raspberry Pi and the right Pi is less than 0.2s. However, we do not have a mechanism to measure the net latency of the cameras currently.

5.5. Frame Rate and Resolution

Frame rate in most of the real-time surgical tracking systems is in the range of 10 fps to 60 fps (Handa, Newcombe et al. 2012, Murray 2017), and it is a general rule of thumb that the accuracy improves when the resolution of the frame increases. The NDI Polaris Vega (Northern Digital, Waterloo, Ontario) is an optical tracking system widely used for medical purposes, providing an excellent frame rate at 60 to 250 Hz. However, a number of real-time surgical tracking systems are slower than 30 fps (Dockter, Sweet et al. 2014, Qiu, Li et al. 2019). Currently, our optical tracking system supports live video with 1080 x 1920 pixels at a frame rate of 25 fps.

Chapter 6. Discussion

In minimally invasive surgery, an orthopedic surgeon can only conceptualize the position of a surgical tool by auxiliary equipment such as endoscope and fluoroscopy rather than visualizing the tool directly. Therefore, the high accuracy of the auxiliaries is indispensable in surgical tracking systems. Generally, a submillimeter-accuracy is important in orthopedic image-guided surgery because it is used under a less-invasive and smaller-vision condition. Surgeons need to know the precise location of the surgical tool provided by the tracking system to avoid damaging the unintended tissue. It also requires real-time tracking capability to ensure the hand operation and the surgical tool locations synchronize with what the display shows. The popular surgical tracking system NDI Polaris Vega supports a volumetric accuracy to 0.12mm (Northern Digital Inc. 2020). A surgical optical tracking system developed by Zhou et al. (Zhou, Wu et al. 2017) can track the location with the distance RMSE 0.049 mm to 0.32 mm, and its estimated cost is $ 3,366. Dockter et al. (Dockter, Sweet et al. 2014) proposed a fast, low-cost approach to track surgical tools, and the average 3D error is 3.05 mm with webcams while the frame rate is 25.86 fps.

In our system, both the accuracy and repeatability can reach to submillimeter. However, the errors can be large to 12.46 mm in some directions. This does not satisfy our requirements. If the error is systematic as we supposed, it could be corrected by image distortion elimination to some extent. In summary, the total error of the tracking result is accumulated by errors generated by:

- Marker sphericity

- Calibration of marker positions by OptiTrack system

- DLT calibration error

- Lens distortion

- Probe marker locations

According to the first accuracy test based on the resolution of 1920 x 1080 pixels video streaming, the largest errors in the X-axis and Y-axis both occurred at markers No.1 (12.46 mm in X, 8.39 mm in Y) and No.7 (12.45 mm in X, 8.55 mm in Y) and these two markers were aligned along the Z-axis, and the values of errors were close. This can be partly explained as a systematic error caused by image distortion. As we know, the largest image distortion usually appears at the edge of the image. Since the markers No.1 and No.7 were relatively located at the edges of the image, it could lead to large distortions and errors in the results. According to the test of distortion correction in Chapter 4, the error of the distorted image is approximately 1.16 mm. Although it is not the distortion error of our tracking tests, it approves the effect on accuracy from the image distortion.

Errors that arose in the measurement process could be compensated by high-precision markers and the calibration frame with known dimensions. For example, we can determine the positions of markers by other accurate measuring instruments or replace the plywood calibration frame with much precisely gridded structures. Meanwhile, we can 3D print the markers with higher-resolution printers or substitute them with other commercial high-quality markers.

The calibration results obtained from DLT reconstruction also introduced some errors to the system as it considers the geometric relationship between the 3D object and 2D image projection as linear, and the camera parameters are ‘estimated’. Currently, the calibration error with a 1920 ×1080 pixels image is approximately 1.8 mm while it is around 1 mm in the image with 960 ×540 pixels. Since the field of view in the lower-resolution images captured by Raspberry Pi is larger than high-resolution images, but the number of pixels is less, every pixel covers a larger area in the real world. Therefore, images with lower resolution are less sensitive to errors, and the errors calculated by DLT reconstruction are smaller in low-resolution images. The DLT calibration method could also be improved by considering the optical distortions and other superior correction algorithms. Additionally, more points collected for calibration

also help increase the accuracy in DLT calibration. For example, OptiTrack Motive usually collects thousands of points in the workspace to calibrate its cameras. Therefore, it will be better to use more markers or calibration data in the future for better calibration results.

As OptiTrack could not provide the accurate dimension of the measurement probe for us, all positions of the markers on the probe were measured by us in-house. We used both ImageJ and digital callipers to measure the distances between markers respectively, and all the results were based on the dimension measured by the calliper. We believe the measurement process introduced some errors to the system, and if we can replace the probe with other known-dimension tools or we can have a better way to obtain the dimensions of the probe, the error can be reduced.

The static jitter of the tracked marker can be defined as a noise in tracking and can be reduced by a filter. Pintaric et al. (Pintaric and Kaufmann 2007) applied predictive filtering in their real-time tracking system to smooth the data. Since the static jitter is always below 2 pixels, a higher resolution image that covers a smaller area in the real world is more sensitive to the changes of the pixels and can reduce the total errors of the tracking result. The markers that we 3D printed could have also contributed to this. Commercially available passive reflective markers (e.g., OptiTrack) have relatively high sphericity with uniform reflectance, but they also produce static jitter.

To evaluate the latency of the system, we can employ either software solutions or hardware solutions. We can set timestamp when the Raspberry Pi camera captures an image and record the timestamp when the computer receives the data of the corresponding image. The difference between the two timestamps should be the latency of the system. To evaluate if the two cameras are working simultaneously, we can place equipment which can send signal regularly in the common field of view of the two cameras. The equipment could be a flashing light or a mechanical pendulum with an angle calliper behind, then the latency between the two cameras is the difference between the images the two cameras captured. For instance, if we have a mechanical pendulum and know the speed of its swinging, and its position can be

demonstrated by an angular calliper sitting in the back. By comparing the positions of the pendulum in the images captured by each camera, it is easy to get the latency between the cameras.

Infrared passive markers are the mostly used markers in surgical optical tracking. Though these markers tend to be more stable in many scenarios (and can reduce static jitter), the matching between markers in different views requires a large volume of computations and creates computational barriers for real-time tracking (especially for computers with low computational power). It requires feature detection and matching which premeditates more complicated features of the object such as shape and its surrounding environment. We chose to use colour segmentation by HSV (hue, saturation, value) to track the markers of the "surgical tool" (for our purposes this was the tracking probe). Tracking by colour segmentation requires much less computational power at each Raspberry Pi to track each marker frame-to-frame, so it is better for maintaining real-time tracking for Raspberry Pi cameras.

Since most of the cameras in 3D tracking are stereo cameras, the depth information can be generated by epipolar geometry, which may simplify the matching problem. Epipolar geometry describes the geometric relationships between the 3D points and their projections on the two 2D camera planes located at different positions, and it constrains the image points in different views. However, two cameras in a stereo camera are close to each other, and their field of view is highly coincident, so they can only observe the object from one side. This could increase errors in directions other than the direction along the principal axis of the camera, as the position information in these directions is estimated rather than directly observed. Cameras located at different perspectives can reduce the blind zone, and inspect the object with multiple angles so the reconstruction can be more accurate and reliable. However, because the stereo cameras are prevalent in surgical tracking systems, it would be worthwhile to attempt to use infrared passive markers with stereo cameras if the accuracy of the current system is limited and difficult to be further improved.

The number of cameras also affects accuracy. In general, more cameras increase the redundancy for reconstruction and reduces errors in the tracking. However, if the camera is infrared-based and can emit

infrared lights itself, we should avoid placing the two cameras facing to each other. Otherwise, there will be more interference in the field of views of the camera, and cause errors or problems in detection. Since there is no infrared-interference issue with the Raspberry Pi cameras we use in our system, more cameras in distinctive locations may reduce the tracking errors and make the system more robust. On the other hand, because Raspberry Pi has independent computational ability, and its output is just a 4 x 2 matrix, increasing the number of cameras will not increase the computational load for the central computer. Since the COVID-19 pandemic, I did not have a chance to upgrade the system with more cameras, but the concept can be tested in future work.

The model of Raspberry Pi in this system is Raspberry Pi 3 Model B+, and a new version Raspberry Pi 4 was released in June 2019. The new model has a faster processor and a larger RAM, which can be beneficial in more complicated computations. The image resolution could also be improved when a better camera module is available. The current camera module we are using is Camera Module V2, and it supports video with 1080p at 30 fps, 720p at 60 fps, and 640 x 480p at 60 or 90 fps. We believe the accuracy of the system can be ameliorated with better hardware support.

Chapter 7. Conclusion

The purpose of this project was to test if low-cost, Raspberry Pi cameras could be used to provide optical tracking for a surgical training system for arthroscopic surgeons. The overall system combines a physical model of a synthetic hip with an optical tracking system to mimic fluoroscopy that the surgeon would use in real surgery. The fluoroscopy images are based on a CT dataset of the artificial hip that is virtually aligned to the position and orientation of the physical synthetic hip joint. As the arthroscopic tool is optically tracked, it can be overlayed on the virtual fluoroscopy image to show the surgeon where the surgical tool is inside the hip. To function effectively, this system requires that the physical hip be precisely aligned and positioned relative to the global tracking position and that the optical tracking be repeatable and accurate to 1 mm or less. At present, the tracking errors of our system can be lower than 1 mm, however, the errors can also be as large as 12 mm at some points. The main errors are from the calibration process and 3D reconstruction by the DLT method, and they can be reduced by optimizing the algorithm and a better measurement method. This research also proves the possibility to use low-cost cameras such as USB cameras or Raspberry Pi camera modules to support the optical tracking for the orthopedic surgical training purpose.

References

Abdel-Aziz, Y., H. Karara and M. Hauck (2015). "Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry." Photogrammetric Engineering & Remote Sensing 81(2): 103-107.

Abramsa, G. D., J. D. Harrisb and M. R. Safranc (2013). "Hip Arthroscopy: Portal Placement."

Agha, R. A. and A. J. Fowler (2015). "The role and validity of surgical simulation." International surgery 100(2): 350-357.

Atesok, K., J. D. Mabrey, L. M. Jazrawi and K. A. Egol (2012). "Surgical simulation in orthopaedic skills training." JAAOS-Journal of the American Academy of Orthopaedic Surgeons 20(7): 410-422.

Ballantyne, G. H., J. Marescaux and P. C. Giulianotti (2004). Primer of robotic and telerobotic surgery, Lippincott Williams & Wilkins.

Bott, O. J., K. Dresing, M. Wagner, B.-W. Raab and M. Teistler (2011). "Informatics in radiology: use of a C-arm fluoroscopy simulator to support training in intraoperative radiography." Radiographics 31(3): E65-E75.

Bradski, G. (2000). "The OpenCV Library." Dr. Dobb's Journal of Software Tools.

Brown, D. C. (1966). "Decentering distortion of lenses." Photogrammetric Engineering and Remote Sensing.

Burman, M. S. (1931). "Arthroscopy or the direct visualization of joints: an experimental cadaver study." Journal of Bone and Joint Surgery 13(4): 669-695.

Centers for Disease Control and Prevention. (2020). "Osteoarthritis (OA)." Retrieved June 5, 2020, from https://www.cdc.gov/arthritis/basics/osteoarthritis.htm#:~:text=Osteoarthritis%20(OA)%20is%20the%20most,underlying%20bone%20begins%20to%20change.

Cimerman, M. and A. Kristan (2007). "Preoperative planning in pelvic and acetabular surgery: the value of advanced computerised planning modules." Injury 38(4): 442-449.

Clohisy, J. C. and J. T. McClure (2005). "Treatment of anterior femoroacetabular impingement with combined hip arthroscopy and limited anterior decompression." The Iowa Orthopaedic Journal 25: 164.

Cooper, J. and Julian.Cooper[at]uhb.nhs.uk (2009/05/22 First version). ImageJ Make_Isotropic Plugin.

De Villiers, J. P., F. W. Leuschner and R. Geldenhuys (2008). Centi-pixel accurate real-time inverse distortion correction. Optomechatronic Technologies 2008, International Society for Optics and Photonics.

Dockter, R., R. Sweet and T. Kowalewski (2014). A fast, low-cost, computer vision approach for tracking surgical tools. 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE.

Duane, C. B. (1971). "Close-range camera calibration." Photogramm. Eng 37(8): 855-866.

Engstrom, E. (1935). "A study of television image characteristics: Part Two: Determination of frame frequency for television in terms of flicker characteristics." Proceedings of the Institute of Radio Engineers 23(4): 295-310.

Escobar-Castillejos, D., J. Noguez, L. Neri, A. Magana and B. Benes (2016). "A review of simulators with haptic devices for medical training." Journal of medical systems 40(4): 104.

Figert, P. L., A. E. Park, D. B. Witzke and R. W. Schwartz (2001). "Transfer of training in acquiring laparoscopic skills." Journal of the American College of Surgeons 193(5): 533-537.

Frank, R. M., B. Erickson, J. M. Frank, C. A. Bush-Joseph, B. R. Bach Jr, B. J. Cole, A. A. Romeo, M. T. Provencher and N. N. Verma (2014). "Utility of modern arthroscopic simulator training models." Arthroscopy: The Journal of Arthroscopic & Related Surgery 30(1): 121-133.

Ganz, R., J. Parvizi, M. Beck, M. Leunig, H. Nötzli and K. A. Siebenrock (2003). "Femoroacetabular impingement: a cause for osteoarthritis of the hip." Clinical Orthopaedics and Related Research® 417: 112-120.

García-Peraza-Herrera, L. C., W. Li, C. Gruijthuijsen, A. Devreker, G. Attilakos, J. Deprest, E. Vander Poorten, D. Stoyanov, T. Vercauteren and S. Ourselin (2016). Real-time segmentation of non-rigid surgical tools based on deep learning and tracking. International Workshop on Computer-Assisted and Robotic Endoscopy, Springer.

Giarmatzis, G., I. Jonkers, M. Wesseling, S. Van Rossom and S. Verschueren (2015). "Loading of hip measured by hip contact forces at different speeds of walking and running." Journal of Bone and Mineral Research 30(8): 1431-1440.

Glossop, N. D. (2009). "Advantages of optical compared with electromagnetic tracking." Journal of Bone and Joint Surgery 91(Supplement_1): 23-28.

Goldman, L. W. (2007). "Principles of CT and CT technology." Journal of Nuclear Medicine Technology 35(3): 115-128.

Hafez, M., R. Smith, S. Matthews, G. Kalap and K. Sherman (2005). "Radiation exposure to the hands of orthopaedic surgeons: are we underestimating the risk?" Archives of orthopaedic and trauma surgery 125(5): 330-335.

Hammond, I. and K. Karthigasu (2006). "Training, assessment and competency in gynaecologic surgery." Best Practice & Research Clinical Obstetrics & Gynaecology 20(1): 173-187.

Handa, A., R. A. Newcombe, A. Angeli and A. J. Davison (2012). Real-time camera tracking: When is high frame-rate best? European Conference on Computer Vision, Springer.

Hausmann, J.-T. (2006). "Sawbones in biomechanical settings-a review." Osteosynthesis and Trauma Care 14(04): 259-264.

Horisberger, M., A. Brunner and R. F. Herzog (2010). "Arthroscopic treatment of femoroacetabular impingement of the hip: a new technique to access the joint." Clinical Orthopaedics and Related Research® 468(1): 182-190.

Hosseini, Z. (2013). Virtual fluoroscopy system for arthroscopic surgical training.

Howells, N., H. Gill, A. Carr, A. Price and J. Rees (2008). "Transferring simulated arthroscopic skills to the operating theatre: a randomised blinded study." The Journal of bone and joint surgery. British volume 90(4): 494-499.

Illingworth, J. and J. Kittler (1987). "The adaptive Hough transform." IEEE Transactions on Pattern Analysis and Machine Intelligence(5): 690-698.

Kelly, B. T., R. J. Williams and M. J. Philippon (2003). "Hip arthroscopy: current indications, treatment options, and management issues." The American journal of sports medicine 31(6): 1020-1037.

Kinnaman, K. and J. D. Mabrey (2006). "Arthroscopy of the hip joint: an overview." Orthopaedic Nursing 25(2): 93-97.

Kühnapfel, U., H. K. Cakmak and H. Maaß (2000). "Endoscopic surgery training using virtual reality and deformable tissue simulation." Computers & Graphics 24(5): 671-682.

Larson, C. M. and C. A. Wulf (2009). "Intraoperative fluoroscopy for evaluation of bony resection during arthroscopic management of femoroacetabular impingement in the supine position." Arthroscopy: The Journal of Arthroscopic & Related Surgery 25(10): 1183-1192.

Lateef, F. (2010). "Simulation-based learning: Just like the real thing." Journal of Emergencies, Trauma and Shock 3(4): 348.

Lee, C. B. and J. Clark (2011). "Fluoroscopic demonstration of femoroacetabular impingement during hip arthroscopy." Arthroscopy: The Journal of Arthroscopic & Related Surgery 27(7): 994-1004.

Leunig, M., P. E. Beaulé and R. Ganz (2009). "The concept of femoroacetabular impingement: current status and future perspectives." Clinical Orthopaedics and Related Research 467(3): 616-622.

Leunig, M., A. S. Ranawat and R. Ganz (2011). CHAPTER 28 - Surgical Hip Dislocation for Femoroacetabular Impingement. Techniques in Hip Arthroscopy and Joint Preservation Surgery. J. K. Sekiya, M. R. Safran, A. S. Ranawat and M. Leunig. Philadelphia, W.B. Saunders: 228-239.

Liu, Y., R. Li, Y. Fan, Đ. Antonijević, P. Milenković, Z. Li, M. Djuric and Y. Fan (2018). "The influence of anisotropic voxel caused by field of view setting on the accuracy of three-dimensional reconstruction of bone geometric models." AIP Advances 8(8): 085111.

Matsuda, D. K. (2009). "Acute iatrogenic dislocation following hip impingement arthroscopic surgery." Arthroscopy: The Journal of Arthroscopic & Related Surgery 25(4): 400-404.

Matsuda, D. K. (2009). "Fluoroscopic templating technique for precision arthroscopic rim trimming." Arthroscopy: The Journal of Arthroscopic & Related Surgery 25(10): 1175-1182.

McKenna, S., H. N. Charif and T. Frank (2005). Towards video understanding of laparoscopic surgery: Instrument tracking. Proc. of Image and Vision Computing, New Zealand.

Modi, C. S., G. Morris and R. Mukherjee (2010). "Computer-simulation training for knee and shoulder arthroscopic surgery." Arthroscopy: The Journal of Arthroscopic & Related Surgery 26(6): 832-840.

Murray, S. (2017). "Real-time multiple object tracking-A study on the importance of speed." arXiv preprint arXiv:1709.03572.

NaturalPoint, I. D. O. (2018, 25 July, 2018). "Markers." 2020, from https://v22.wiki.optitrack.com/index.php?title=Markers.

NaturalPoint, I. D. O. (2020). "OptiTrack General FAQs." Retrieved March 20, 2020, from https://www.optitrack.com/support/faq/general.html.

Nephew, S. (2011). The Anatomy Structure of Hip Joint. https://commons.wikimedia.org/wiki/File:Figure_1._Basic_anatomy_of_the_hip_joint.png#filehistory, Wikimedia Commons: Hip joint.

Northern Digital Inc. (2020). "Polaris Vega Optical Tracking System." Retrieved May 13, 2020, from https://www.ndigital.com/medical/products/polaris-vega/.

Oracle Corporation. "The Java™ Tutorials - What Is a Socket?".

Orth, R. C., M. J. Wallace, M. D. Kuo and T. A. C. o. t. S. o. I. Radiology (2008). "C-arm cone-beam CT: general principles and technical considerations for use in interventional radiology." Journal of Vascular and Interventional Radiology 19(6): 814-820.

Pentenrieder, K., P. Meier and G. Klinker (2006). Analysis of tracking accuracy for single-camera square-marker-based tracking. Proc. Dritter Workshop Virtuelle und Erweiterte Realitt der GIFachgruppe VR/AR, Koblenz, Germany, Citeseer.

Peters, J. H., G. M. Fried, L. L. Swanstrom, N. J. Soper, L. F. Sillin, B. Schirmer, K. Hoffman and S. F. Committee (2004). "Development and validation of a comprehensive program of education and assessment of the basic fundamentals of laparoscopic surgery." Surgery 135(1): 21-27.

Pintaric, T. and H. Kaufmann (2007). Affordable infrared-optical pose-tracking for virtual and augmented reality. Proceedings of Trends and Issues in Tracking for Virtual Environments Workshop, IEEE VR.

Qiu, L., C. Li and H. Ren (2019). "Real-time surgical instrument tracking in robot-assisted surgery using multi-domain convolutional neural network." Healthcare Technology Letters 6(6): 159-164.

Roberts, K. E., R. L. Bell and A. J. Duffy (2006). "Evolution of surgical skills training." World Journal of Gastroenterology: WJG 12(20): 3219.

Schindelin, J., I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld and B. Schmid (2012). "Fiji: an open-source platform for biological-image analysis." Nature methods 9(7): 676-682.

Sharma, M. and A. Horgan (2012). "Comparison of fresh-frozen cadaver and high-fidelity virtual reality simulator as methods of laparoscopic training." World Journal of Surgery 36(8): 1732-1737.

Silverman, S. G., K. Tuncali, D. F. Adams, R. D. Nawfel, K. H. Zou and P. F. Judy (1999). "CT fluoroscopy-guided abdominal interventions: techniques, results, and radiation exposure." Radiology 212(3): 673-681.

Sodeman, W. A. (1999). Instructions for geriatric patients, . Elsevier Saunders.

Sussmann, P. S., A. S. Ranawat, J. Lipman, D. G. Lorich, D. E. Padgett and B. T. Kelly (2007). "Arthroscopic versus open osteoplasty of the head-neck junction: a cadaveric investigation." Arthroscopy: The Journal of Arthroscopic & Related Surgery 23(12): 1257-1264.

Tsai, M.-D., M.-S. Hsieh and C.-H. Tsai (2007). "Bone drilling haptic interaction for orthopedic surgical simulator." Computers in Biology and Medicine 37(12): 1709-1718.

US Food Drug Administration (1994). "FDA public health advisory: avoidance of serious x-ray-induced skin injuries to patients during fluoroscopically-guided procedures. Rockville, Md: Department of Health and Human Services." Center for Devices and Radiological Health, September 30.

Van Den Bogert, A. J., L. Read and B. M. Nigg (1999). "An Analysis of Hip Joint Loading During Walking, Running, and Skiing." Medicine & Science in Sports & Exercise 31(1): 131-142.

Vankipuram, M., K. Kahol, A. McLaren and S. Panchanathan (2010). "A virtual reality simulator for orthopedic basic skills: a design and validation study." Journal of Biomedical Informatics 43(5): 661-668.

Wanzel, K. R., S. J. Hamstra, M. F. Caminiti, D. J. Anastakis, E. D. Grober and R. K. Reznick (2003). "Visual-spatial ability correlates with efficiency of hand motion and successful surgical performance." Surgery 134(5): 750-757.

Weber, E. L., H. A. Leland, B. Azadgoli, M. Minneti and J. N. Carey (2017). "Preoperative surgical rehearsal using cadaveric fresh tissue surgical simulation increases resident operative confidence." Annals of Translational Medicine 5(15).

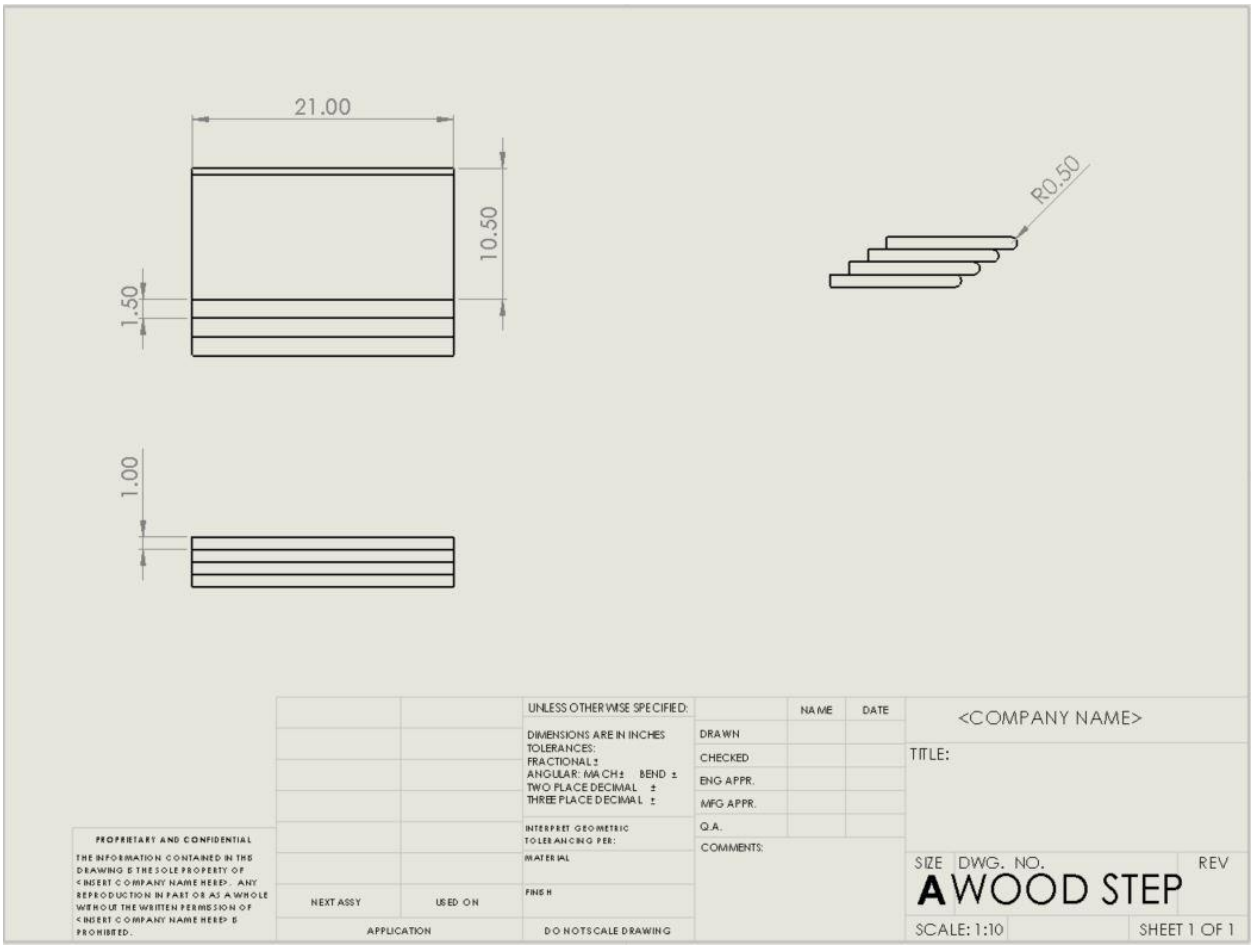
Wehling, M. (2015). Principles of translational science in medicine: From bench to bedside, Academic Press.

Yang, K.-H. (2017). Basic finite element method as applied to injury biomechanics, Academic Press.

Zhou, Z., B. Wu, J. Duan, X. Zhang, N. Zhang and Z. Liang (2017). "Optical surgical instrument tracking system based on the principle of stereo vision." Journal of biomedical optics 22(6): 065005.

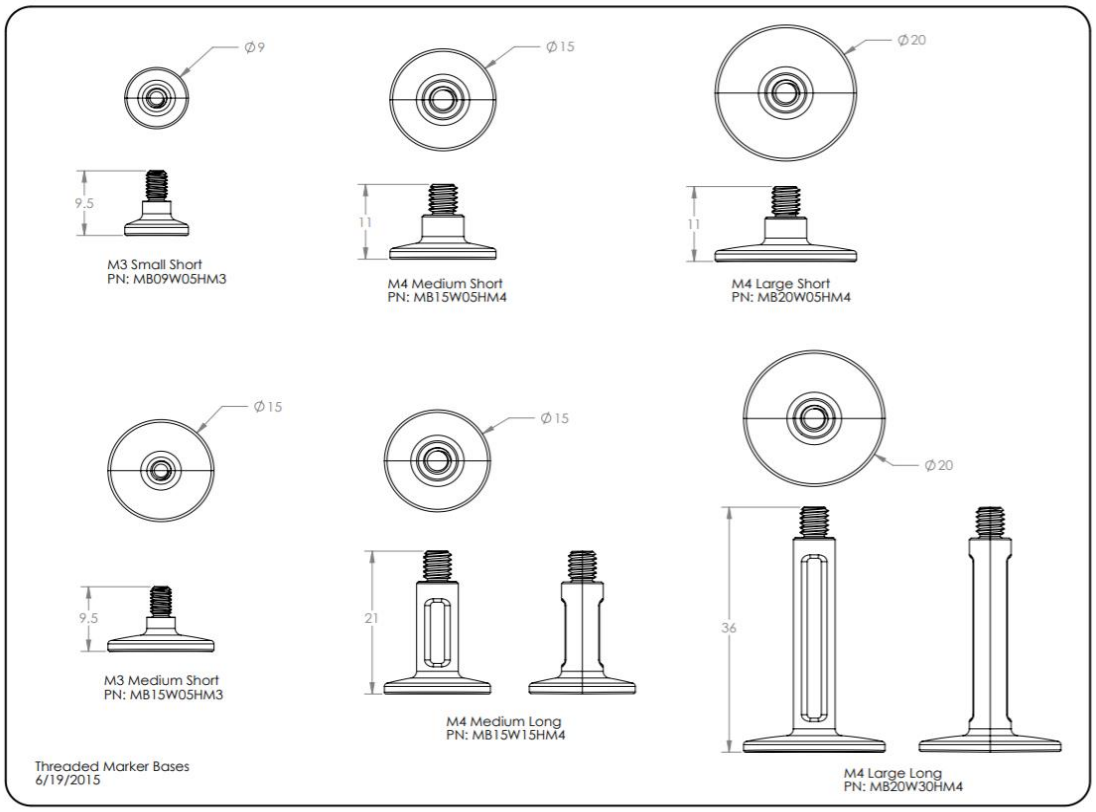
Appendix A: CAD Drawings

CAD Drawing of Wood Calibration Base



**The unit of length in this graph is inch.*

CAD Drawing of Marker Bases



**This CAD drawing is from OptiTrack (NaturalPoint, Inc. DBA OptiTrack, Corvallis, OR). The marker bases used in this project are M4 Large Short and M4 Large Long, and the diameter of the marker is Φ 12.7 mm.*

Appendix B: Python Codes

Raspberry Pi

```
import cv2
from picamera.array import PiRGBArray
import picamera
import socket
import sys
import pickle


class Client:
    SERVER_HOSTNAME = '192.168.1.97'
    HOSTNAME = '192.168.1.240'
    RECV_BUFFER_SIZE = 1024

    def __init__(self, port):
        self.port = port
        self.send_info()

    def color_segmentation(self, image, low1, high1):
        max_area = 0
        index = 0
        hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
        mask = cv2.inRange(hsv, low1, high1)
        rgb = cv2.bitwise_and(image, image, mask=mask)
        gray = cv2.cvtColor(rgb, cv2.COLOR_BGR2GRAY)
        cnts, hierarchy = cv2.findContours(gray, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
        if len(cnts) == 0:
            print('No marker detected!')
        else:
            for i, con in enumerate(cnts):
                area = cv2.contourArea(con)
                if area > max_area:
                    max_area = area
                    index = i
            M = cv2.moments(cnts[index])
            x0 = int(M['m10'] / M['m00'])
            y0 = int(M['m01'] / M['m00'])
            center = (x0, y0)
            return center, cnts[index]

    def send_info(self):
        try:
            print('connecting...')
            self.get_socket()
            print("-" * 72)

            with picamera.PiCamera() as camera:
                with picamera.array.PiRGBArray(camera) as stream:
                    camera.framerate = 30
                    # camera.resolution = (1920, 1080)
                    camera.resolution = (960, 540)
```

```
            # camera.resolution = (540, 300)
            # time.sleep(0.1)
            while True:
                camera.capture(stream, 'bgr', use_video_port=True)
                image = stream.array
                red, cnts_red = self.color_segmentation(image, (120, 80, 100), (255, 255, 255))  # red right
                blue, cnts_blue = self.color_segmentation(image, (90, 75, 100), (150, 255, 255))  # blue right
                green, cnts_green = self.color_segmentation(image, (30, 10, 30), (100, 230, 200))  # green right
                yellow, cnts_yellow = self.color_segmentation(image, (0, 0, 220), (100, 255, 255))  # yellow right
                centers = [green, yellow, blue, red]
                cnts = [cnts_green, cnts_yellow, cnts_blue, cnts_red]
                print("Right camera:\n", centers)
                self.socket.sendall(pickle.dumps(centers))
                for i, center in enumerate(centers):
                    ((x, y), radius) = cv2.minEnclosingCircle(cnts[i])
                    text = '#{}, {}'.format(i+1, center)
cv2.putText(image, text,
                            center, cv2.FONT_HERSHEY_SIMPLEX,
                            0.5, (0, 0, 255), 1)
                    cv2.circle(image, center, int(radius), (0, 0, 250), 1)
                    cv2.circle(image, center, 2, (0, 0, 255), -1)

                cv2.imshow('test', image)
                key = cv2.waitKey(50) & 0xFF == ord('q')
                if key:
                    break
                stream.truncate()
                stream.seek(0)
        except (KeyboardInterrupt, EOFError):
            print()
            print("An exception happened...")
            self.socket.close()
            sys.exit(1)

        finally:
            # executed regardless if the try block raises an error or not
            print()
            print("Closing server connection ...")
            self.socket.close()
            sys.exit(1)

    def get_socket(self):
        self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.socket.bind((Client.HOSTNAME, self.port))
        self.socket.connect((Client.SERVER_HOSTNAME, 6666))


if __name__ == '__main__':
s = Client(8080)
```

Central Computer

Server

```
import socket
import time
import threading
import pickle
import DLT as dlt
import numpy as np
import Transformation as trans
import cv2


data_left = tuple()
data_right = tuple()
matrix_left = dlt.matrix_left
matrix_right = dlt.matrix_right
tool = np.array([[0, 126.617, 16.553],
                 [0, 177.277, 16.553],
                 [-29.88, 226.424, 16.553],
                 [29.403, 262.097, 16.553]])
color = (169, 169, 169)
thickness = 2


class ClientThread(threading.Thread):
    def __init__(self, client):
        threading.Thread.__init__(self)
        self.connection, self.address_port = client
        print("-" * 72)
        print("Connection received from {} on port {}.".format(self.address_port[0], self.address_port[1]))
    def run(self):
        while True:
            try:
                recvd_bytes = self.connection.recv(Server.RECV_BUFFER_SIZE)
            except ConnectionResetError:
                print("There is an exception ... ")
                self.connection.close()
                break
            if len(recvd_bytes) == 0:
                print("Closing client connection ... ")
                self.connection.close()
                break
            msg = pickle.loads(recvd_bytes)
            time_stamp = time.perf_counter()
            global data_left, data_right
            if self.address_port[1] == 8081:
                data_left = (msg, time_stamp)
            elif self.address_port[1] == 8080:
                data_right = (msg, time_stamp)
            self.connection.sendall(recvd_bytes)


class Server:
    HOSTNAME = ""
    PORT = 6666
```

```
    RECV_BUFFER_SIZE = 1024
    MAX_CONNECTION_BACKLOG = 1
    MSG_ENCODING = "utf-8"
    SOCKET_ADDRESS = (HOSTNAME, PORT)

    def __init__(self):
        self.create_listen_socket()
        self.process_connections_forever()

    def create_listen_socket(self):
        self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.socket.bind(Server.SOCKET_ADDRESS)
        self.socket.listen(Server.MAX_CONNECTION_BACKLOG)
        print("Listening for connections on port port {} ...".format(Server.PORT) ...".format(Server.PORT))

    def process_connections_forever(self):
        count = 0
        threshold = 0.2
        window_name = 'Show'
        while count < 2:
            client = self.socket.accept()
            t = ClientThread(client)
            t.start()
            count += 1
        prev_time_stamp_left = 0
        prev_time_stamp_right = 0
       while True:
            if data_left and data_right:
                msg_left, time_stamp_left = data_left
                msg_right, time_stamp_right = data_right
                if abs(time_stamp_left - time_stamp_right) < threshold:
                    if time_stamp_left != prev_time_stamp_left and time_stamp_right != prev_time_stamp_right:
                        mtx_left = np.array(msg_left)
                        mtx_right = np.array(msg_right)
                        recon = dlt.reconstruction(matrix_left, matrix_right, mtx_left, mtx_right)
                        start_point = np.mean(recon, axis=0)
                        start_point = (int(abs(start_point[0])), int(abs(start_point[2])))
                        r, t = trans.transformation_3d(tool, recon)
                        end_point = (int(t[0]), int(t[2]))
                        print(t)
                        background = cv2.imread('Bone.PNG')
                        cv2.line(background, start_point, end_point, color, thickness)
                        cv2.imshow(window_name, background)
                        prev_time_stamp_left = time_stamp_left
                        prev_time_stamp_right = time_stamp_right
            if cv2.waitKey(50) & 0xFF == ord('q'):
                break
        cv2.destroyAllWindows()

if __name__ == '__main__':
s = Server()
```

DLT

```
def reconstruction(left, right, uv_l, uv_r):
    num_points = np.reshape(uv_l, (1, -1)).shape[1]
    matrix = np.concatenate((np.reshape(left, (1, -1)), np.reshape(right, (1, -1))), axis=0)
    uvs = np.concatenate((np.reshape(uv_l, (1, -1)), np.reshape(uv_r, (1, -1))), axis=0)
    matrix = np.asarray(matrix)
    if matrix.ndim == 1:
        print('At least two sets of camera calibration parameters are required.')
    elif matrix.ndim > 1 and matrix.shape[0] != 2:
        print('Number of view is incorrect!')
    else:
        xyz = []
        for j in range(0, num_points, 2):
            M = []
            for i in range(2):
                l = matrix[i, :]
                u, v = uvs[i][j], uvs[i][j+1]
                M.append([l[0]-u*l[8], l[1]-u*l[9], l[2]-u*l[10], l[3]-u*l[11]])
                M.append([l[4]-v*l[8], l[5]-v*l[9], l[6]-v*l[10], l[7]-v*l[11]])
            U, S, Vh = np.linalg.svd(np.asarray(M))
            xyz.append(list(Vh[-1, 0:-1] / Vh[-1, -1]))
        xyz = np.reshape(xyz, (-1, 3))
        return xyz
```

Transformation

```
def transformation_3d(a, b):
    if len(a) != len(b):
        print('Dimensions are not matching!')
    else:
        # calculate the centroid point
        n = len(a)
        a = np.mat(a)
        b = np.mat(b)
        center_tool = np.mean(a, axis=0)
        center_real = np.mean(b, axis=0)

        # decentralization
        de_tool = a - np.tile(center_tool, (n, 1))
        de_real = b - np.tile(center_real, (n, 1))
        h = np.transpose(de_tool) * de_real
        u, s, v = np.linalg.svd(h)
        r = v.T * u.T
        if np.linalg.det(r) < 0:
            v[2, :] *= -1
            r = v.T * u.T
        t = -r * center_tool.T + center_real.T
        return r, t
```