Development of a Self-Calibrated Motion Capture System by Nonlinear Trilateration of Multiple Kinects v2

by

Bowen Yang

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements

For the M.A.Sc. degree in

Electrical and Computer Engineering

School of Electrical Engineering and Computer Science

Faculty of Engineering

University of Ottawa

Abstract

In this paper, a Kinect-based distributed and real-time motion capture system is developed. A trigonometric method is applied to calculate the relative positions of Kinect v2 sensors with a calibration wand and register the sensors' positions automatically. By combining results from multiple sensors with a nonlinear least square method, the accuracy of motion capture is optimized. Moreover, to exclude inaccurate results from sensors, a computational geometry is applied in the occlusion approach to discover occluded joint data. The synchronization approach is based on the NTP protocol, which synchronizes the time between the clocks of a server and of clients dynamically, leading to the proposed system being real time. Experiments to validate the proposed system are conducted from the perspective of calibration, occlusion, and accuracy. More specifically, the mean absolute error of the calibration results is 0.73 cm, the proposed occlusion method is tested on upper and lower limbs, and the synchronization component guarantees the clock synchronization and real-time performance for more than 99% of the measurement process. Furthermore, to demonstrate the practical performance of our system, a comparison with previously developed motion capture systems (the linear trilateration approach [52] and the geometric trilateration approach [51]) with the benchmark OptiTrack system is performed for the tracked joints of the head, shoulder, elbow, and wrist, therein showing that the accuracy of our proposed system is 38.3% and 24.1% better than the aforementioned two trilateration systems. Quantitative analysis is also conducted on our proposed system with the commercial inertial motion capture system Delsys smart sensor system by comparing the measurements of lower limbs (i.e., hips, knees, and ankles), and the standard deviation of our proposed system's measurement results is 4.92 cm.

Keywords. Motion capture, Kinect v2, occlusion compensation, trilateration approach comparison

Acknowledgements

I would like to give my sincerest gratitude and appreciation to my supervisor, Prof. Abdulmotaleb El Saddik, for his continuous guidance and support not only in the academic domain but also in my personal life.

Unique and sincere thanks go to Dr. Haiwei Dong for the precious assistance, invaluable guidance, and feedback that he supplied during my research, as well as his review and revision of this thesis. I also need to thank to Mr. Lin Yang for the basis of the work built by him.

I would also like to thank all my colleagues in the MCRLab for their suggestions and contributions throughout the research and all my friends for their help on my campus life. I list some of them here: Shiai Zhu, Longyu Zhang, Zhongli Li, Bote Han, Liu Yang, Basim Hafidh, Majed Alowaidi, Faisal Arafsha, and Saeed Alharthi.

Finally, I am grateful to my family. Their love, understanding and support made me grow over these years.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

The motion capture technique was first proposed in the late 1970s [21]. It helps people track a human body's movement and apply the captured motion for further purposes. Nowadays, it has been widely used in multiple fields: In the film industry, it is utilized to track an actor's posture and facial expression to build a virtual character with animation or computer graphics, as shown in Fig. 1.1 [55]. In sports medicine, it is applied to analyze athletes' motion to promote their sports performance [10]. In entertainment, particularly in somatosensory game, the motion captured data are used to synchronize the motion of the virtual character with that of the video game player, as shown in Fig. 1.2 [34].

The Kinect (Fig. 1.3) series is a series of motion sensing input devices developed by Microsoft for their XBox series of game consoles and Windows PCs. The Kinect recognizes and tracks the human body through depth measurements to provide motion capturing. The

Figure 1.1: Actors with a motion capture suit during the shooting of the movie Pirates of the Caribbean. [1]

Kinect applies random decision forest [37, 41] to build its human body tracking algorithm, which maps a depth image to the body parts and calculates the joints' positions to simulate the skeleton. It was developed on over 1 million training examples and can quickly track 3 people in a single depth image at 200 frames per second. Kinect v1 applies a light coding[5] principle using an infrared (IR) sensor to obtain depth data. It projects IR pulses consisting of numerous dots spaced at equal distances to the surface of the environment and measures the distance and the shape of the surface based on the perceived distance of the spacing of the dots (Fig. 1.4). Kinect v2 applies the time-of-flight principle, which combines the IR projector with a monochrome CMOS sensor. It resolves distances based on the known speed, providing a better performance compared with Kinect v1. The accuracy of the

[1] https://www.fxguide.com/featured/ilm_computers_and_that_pirate_movie/

Figure 1.2: A player with a PrioVR MoCap suit is playing a first-person-shooter game on the Wii console. [2]

Kinect v2 depth measurement is not affected by the density of the dots in the IR pulse, which decreases with increased distance, and thus, it has a larger measurement range compared to that of Kinect v1. Moreover, light coding has a more stringent requirement on setup conditions such as the light intensity and the light absorption of the surface. The Kinect v2 also includes a more powerful chipset, which supports a higher data rate of framing. The Kinect v2 can transfer the depth measurement as a 512×424 depth map stream at a frame rate of 30 Hz. The accuracy of the 3D measurement of Kinect v2 was validated in [13], where the measurement error is less than 0.07 cm in all three directions.

Several studies have been associated with motion capturing using the Kinect, including its algorithm, systems developed on the Kinect and assessments of its accuracy. First, the Kinect applies a random decision forest [37] [41] to build its human body tracking algorithm, which maps a depth image to the body parts and calculates the joints' positions to simulate the skeleton. The Kinect was developed on over 1 million training examples

[2]http://www.gamepointsnow.com/blog/category/wii/

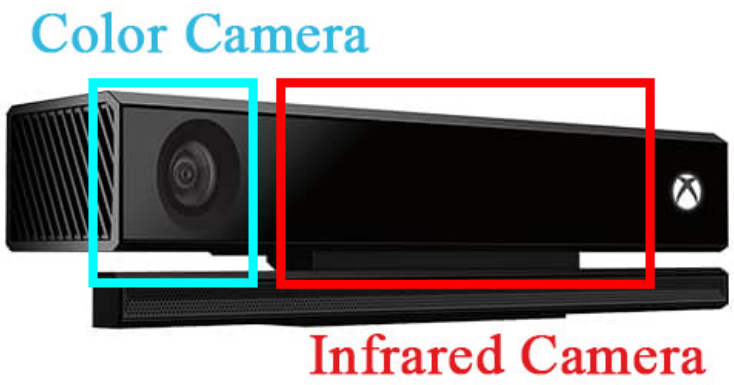


Figure 1.3: Kinect v2 motion capture sensor.[3]

and can quickly track 3 people in a single depth image at 200 frames per second. The hardware coefficient and its depth sensor accuracy, e.g., the average precision distribution and linearity, resolution, and depth spatial precision, are evaluated in [52]. The study gave the limitation of the Kinect's depth measurement at the hardware level.

The accuracy of gait tracking using a single Kinect is evaluated in [43], which certifies the validation of a single Kinect to measure lower extremity kinematics accurately. The Kinect and a digital inclinometer agree with each other to less than 0.5° for sagittal and frontal plane joints and to less than 2° for transverse plane rotation. A hand kinematic analysis system is also developed in [29] and validated to measure finger joint movement compared with a Vicon system with an average absolute error of 2.4°, which is better than clinically based alternative manual techniques for finger measurement. In [40], an advanced hand gesture tracking interface that can recognize 55 static and dynamic gestures with an accuracy of 92.4% is developed.

[3]https://www.microsoftstore.com/store/msusa/en_US/pdp/Kinect-Sensor-for-Xbox-One/productID.2267482500

Figure 1.4: Infrared image showing the laser grid Kinect uses to calculate depth [4]

However, using a single Kinect sensor to develop an accurate postural tracking system is infeasible according to [7] and [39]. Compared with the accuracy of whole-body tracking by Vicon, a single Kinect could not meet the requirements for a medical system. Since Microsoft released the SDK for the Kinect, combining multiple Kinects to improve accuracy has become feasible. In [53], a system with multiple Kinects is built to reconstruct an object in 3D coordinates, and the uncertainties of the Kinect's depth sensor and a robust way to properly align multiple Kinects are also analyzed and proposed.

[4]https://en.wikipedia.org/wiki/Kinect

1.2 Objective and Contribution

In this thesis, a multiple-Kinect motion capture system is developed. Four key issues are addressed in this thesis: calibration, occlusion compensation, accuracy improvement, and clock synchronization. Specifically, a calibration approach is designed to be easily set up and quickly configured. An occlusion compensation method is proposed to avoid incorrect tracking due to occlusion occurring in the body movement. A nonlinear trilateration approach is proposed to optimize the tracking accuracy based on the redundant measurements of multiple Kinect v2 sensors located in different positions and oriented in different angles. Finally, a synchronization method is proposed to synchronize clocks of clients and the server to organize measurements from sensors. The experiments are designed and conducted to evaluate the aforementioned four approaches, including examining the calibration approach by analyzing the factors affecting the Kinect's depth measurement; evaluating the occlusion compensation approach by comparing the tracking result with and without occlusion compensation; validating the trilateration approach by comparing the accuracy with state-of-the-art methods, i.e., the linear trilateration approach [52], the geometric trilateration approach [51], and the Delsys smart sensor system [3]; and testing the synchronization method by measuring the processing time of the frame and the clock error of the proposed system. Compared with [52], the proposed nonlinear trilateration method achieves a higher accuracy. Furthermore, the accuracy of our proposed trilateration approach can be improved by adding more sensors, whereas the geometric trilateration approach [51] is restricted, as it only supports a 3-Kinect setup.

Based on the thesis, I involved 2 journal papers which are [51] and [50].

1.3 Thesis Organization

This thesis is organized as follows:

- **Chapter 2:** This chapter reviews the categorization of current optical motion capture technologies and then proposes the approaches of different schemes in motion capture systems, including how to resister camera positions, occlusion compensation and clock synchronization.

- **Chapter 3:** This chapter gives the overall design of the proposed system, proposes the approaches and implements each component, including the synchronization component, calibration component, trilateration component, occlusion compensation component, and visualization component.

- **Chapter 4:** This chapter gives the setups and results of experiments used to evaluate and show the performance of each component and the whole system, including the calibration evaluation, occlusion compensation verification, trilateration method comparison, accuracy comparison between trilateration systems (the linear trilateration method, the geometric trilateration method, and the Delsys smart sensor system), and efficiency analysis. This chapter also presents a visualization of the results by applying OpenSim kinematics analysis.

- **Chapter 5:** This chapter concludes the study and discusses future work.

Chapter 2

Related Work

2.1 Motion Capture Categorization

Usually, optical motion capture systems can be classified as marker-based or markerless-based systems. Optical marker-based motion capture systems track markers placed on the human body, where most commonly used commercial systems (e.g., Vicon and Optitrack) apply IR reflective markers (Fig. 2.2), although LED markers (Fig. 2.1) are also utilized in certain marker-based systems [38]. On the other hand, the Kinect can be considered as a markerless-based optical motion capture device. The Kinect applies the human body structure principle [9] to recognize the human body against the background and estimates the 3D coordinates of the body joints.

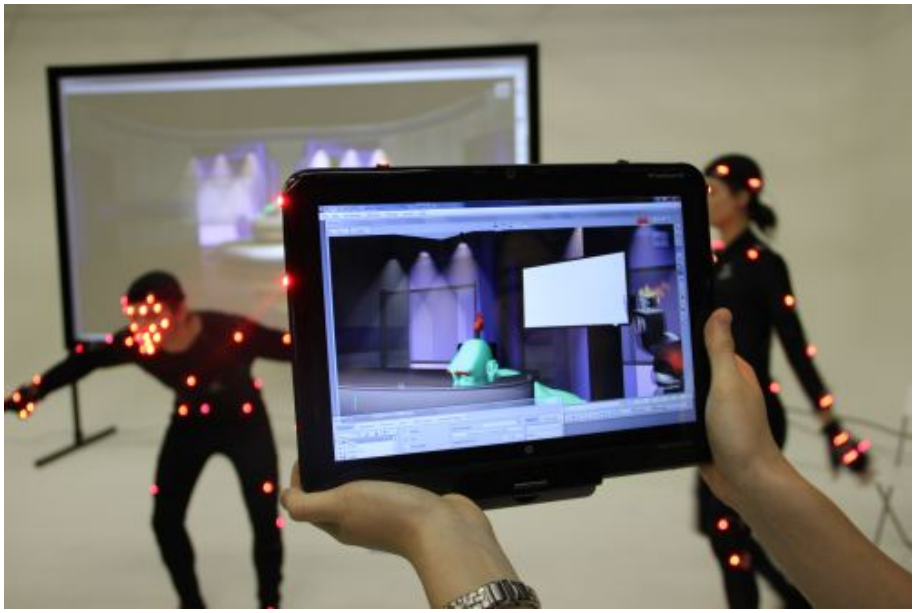


Figure 2.1: The Phasespace motion capture system applies LED markers. [1]

2.2 Calibration

To calibrate sensors, motion capture systems are required to measure the relative positions of calibration points by transferring image points to the corresponding world coordinates [6]. Marker-based and markerless-based systems are supposed to focus different fields during calibration because the markers can be accurately recognized from the background, but the positions of the markers cannot be easily measured. Thus, marker-based systems always set up multiple markers (Fig. 2.3) [42, 49] for calibration by comparing the difference in the relative positions of the markers from the sensors or comparing the relative speeds of moving markers [47]. The advantage of the marker-based systems' calibration is the stable accuracy; IR sensors are robust to the influence of the environment of the measurement area. However, the disadvantage is that the calibration setup of the markers is always

[1] http://www.phasespace.com/virtualcamera.html

Figure 2.2: The Microsoft Kinect captures actor movements to visualize the in-game character's actions. [2]

complex and time consuming.

On the other hand, markerless-based systems apply light coding or time-of-flight methods to generate the depth map of their field of view and use RGB or depth data to recognize the calibration objects from the environment. Thus, the calibration object is required to be easily recognizable due to its special shape or optical property. Currently, a matrix barcode on a planar surface, i.e., quick response codes (QR Codes), is the most wildly used technology in the optical identification field. Open-source libraries for QR codes has been released by OpenCV and can be applied in various types of optical cameras such as Optitrack [36], Kinect [53], and Wii remotes [38]. In addition, calibration objects can be designed with special shapes to be easily recognizable by depth sensors [23][19]. Compared with marker-based calibration, the markerless-based calibration with a QR Code

[2] http://www.digitalmediaworld.tv/games/120-343-industries-builds-optitrack-mocap-system-for-halo-5-guardians

Figure 2.3: Using the 3-marker wand to calibrate the Optitrack system with 12 Flex 13 cameras in a Green Room. [3]

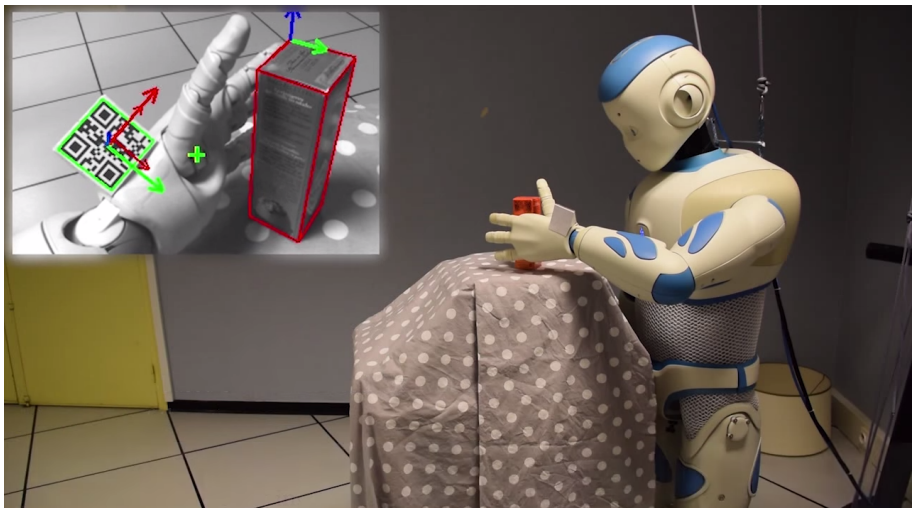


Figure 2.4: The robot Romeo detects his hand pose using a QR Code and control its eyes to focus on his hand to attempt to catch the box. [4]

achieves a better time efficiency because producing a QR Code on a paper or surface is easy and efficient. In this thesis, our proposed system applies the depth map to recognize the calibration object (the details are illustrated in Section 3.4).

[3]https://www.youtube.com/watch?v=d3u4btSsOJs
[4]https://github.com/lagadic/romeo_tk/wiki/Romeo-Grasping-Demo

2.3 Occlusion Compensation

Occlusion is the primary factor affecting the accuracy of optical motion capture systems. Methods to detect occlusion and furthermore compensate its influence are quite different depending on a system's structure. A single sensor can only obtain a side field of view; thus, positions of occluded joints can only be inferred. Therefore, systems with a single sensor are designed to apply a likelihood algorithm to resolve the occlusion [31], which matches the measured depth data with human body models based on its movement restrictions. In this case, the system's accuracy is related to the likelihood estimation results. On the other hand, systems with multiple sensors can utilize the advantage of the distributed sensor layout: the occluded observations would be determined and furthermore ignored prior to the sensor data measurement process. In this thesis, the proposed occlusion approach applies the computation geometry principle, which will be detailed in Section 3.6.

Another issue for markerless-based systems with multiple sensors is that it is difficult to guarantee the measured point of corresponding joints from different sensors. Most widely used markerless-based sensors, e.g., the Kinect and Wii remote device, obtain the positions of joints when they can only measure the surface of the human body. This means that all joints are covered by the surface of the body.

2.4 Synchronization

There are three types of structures for synchronization networks: the bus topology, the star topology, and the tree topology (Fig. 2.5) [26]. Motion capture systems apply these topology structures depending on the complexity of the sensor network. For instance, Op-

(a) (b) (c)

Figure 2.5: Diagrams of (a) bus topology, (b) star topology, and (c) tree topology networks.

titrack can be configured into different synchronization topologies. When a USB hub is not used, no more than 3 sensors are directly connected to the computer, and synchronization is performed by the bus through the sensors. The star topology is applied when a single USB hub, which can support no more than 6 sensors, is used. The hub is designed to synchronize the data obtained from sensors and then send the data to the computer. When there are more than 6 sensors, a tree topology is applied, meaning that more than 1 USB hub is needed. One of the hubs is configured as the chief server to control the clocks of the other hubs. In the proposed system, a star topology is applied.

Chapter 3

System Design

3.1 Overall System Architecture

The proposed system's architecture, including the components and the server-client network structure, is designed as shown in Fig. 3.1. The arrows in the figure indicate the direction of data transmission. The system essentially consists of several clients and a single server. Each client controls one Kinect v2 and continuously obtains depth images. The data will be sent to the server after being arranged and initially processed. The server controls all clients and manages the received data. Furthermore, the server acts as the interface to users that displays the results and receives commands.

Classified by the specific feature, the following five components constitute the system:

1) The transmission component establishes and maintains the connection in the system, and it is the communication interface with which other components exchange messages. As shown in Fig. 3.1, the system is distributed, meaning that a communication mechanism

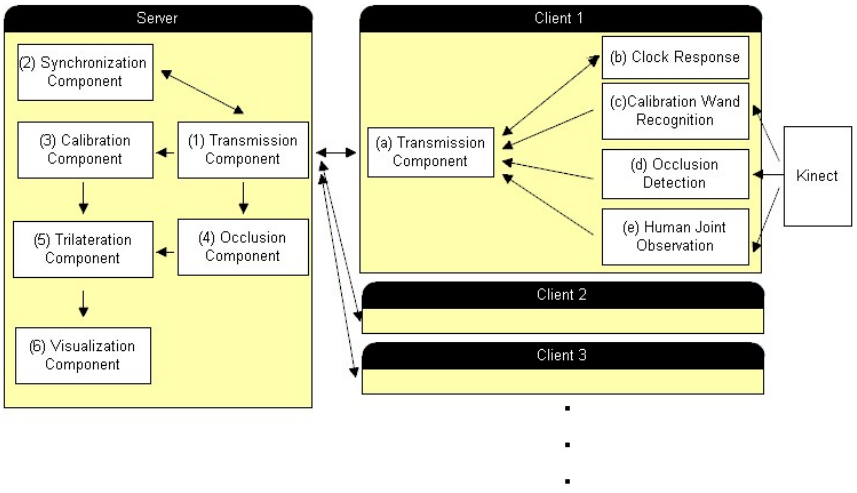


Figure 3.1: Overall system architecture of the multi-Kinect trilateration system

is needed. There are multiple components working at the same time when the system is running. Most of the components need to exchange data between the client and server to perform their functions. Because the system adopts at most 16 clients, to show results in real time as well as ensure accuracy and robustness, it is imperative to guarantee unimpeded network transmission.

The transmission component establishes the connection between the clients and server when the system starts up and undertakes all the communication tasks from other components. All data passed to the transmission component will be packaged before being sent out, therein promoting the efficiency of the network. The hardware part of the component consists of network adapters and cables, and the software part is based on the TCP/IP protocol and applies Winsocket APIs [22] to perform the functions.

2) The synchronization component guarantees the stability of a sequence of time in the system. It provides millisecond-scale clock synchronization between clients and the server.

Because the system is built on the local area network (LAN), there are many factors that can affect the data transmission between clients and the server. The delays of each frame in the network reaching their destination are different. Although the TCP/IP protocol guarantees the correct order of frames, the transmission can be obstructed in the worst case. Thus, it is necessary for the system to be able to support fluctuations of the network transmission status.

There is always a difference between the clocks in different computers. Although every computer connected to the Internet corrects their clocks with the Internet clock server multiple times a day, there are still some factors that affect the accuracy of the clock in the computer such as overclocked CPUs and unstable voltage input. Furthermore, considering that the system may disconnect from the Internet, a mechanism for correcting the clock is needed; otherwise, the accuracy of the result cannot be assured because the server might obtain a result with data measured at a different time.

The synchronization component starts working once a connection is established by the transmission component between the server and clients and will not stop until the system shuts down. This component dynamically calculates the difference between the clock of the server and those of the clients and the delay of the data transmission over the network. This component is built based on the NTP protocol and has been modified to precisely meet the requirements of the system. The component continuously exchanges messages nailed by a time stamp through the transmission component to obtain the status of the network and the computers' clocks. All such operations are controlled by the component at the server, and all data are collected by the server for further processing.

3) The self-calibration component allows the system to obtain the positions of the

clients automatically once set up. A Kinect is able to measure the object's position in three-dimensional space. However, the result is based on its local coordinate system. An integral coordinate system is needed when combining the data from multiple Kinects, for which the position of each sensor needs to be measured.

In this respect, the mark stick is designed to define the direction of the axes and the data point. The stick would be placed in the measurement area with which clients obtain the relative position and direction. Data would be sent to the server to generate the absolute position. Once the connection between one client and server is established, the component in the client would start working until the calibration procedure was completed. The hardware part of the self-calibration component consists of the mark stick and Kinects, and the software part is based on Kinect SDKs and geometry and mathematical principles.

4) At the clients, the trilateration component generates the joint position from the obtained depth image; at the server, the trilateration component identifies robust data sent by clients and calculates the corresponding joint position. Once the self-calibration procedure is completed, meaning that all sensors' positions have been obtained, the server would send commands to clients to start up the trilateration components.

In the client, the trilateration component applies the Kinect SDKs to recognize the human body from the obtained color image and calculates the relative positions of joints using the depth image. The measurement data of the joints' position is passed to the transmission component and then sent to the server.

In the server, the trilateration component sets up buffers for measurement data sent from each client. To obtain result in real time, the component restricts the interval of taking out data from the buffer, and superfluous and independent data is abandoned.

The component transfers the relative position data from different clients to the integral coordinate system and combines the results by applying a nonlinear least square method to minimize the error. In addition, the competition mechanism is applied to optimize the accuracy of the result.

5) The visualization component is performed on the server and shows the results from the trilateration component to the user with a generated image and makes further analysis through other software. This component is designed to be the interface that shows the trilateration results, which consists of two parts: 1) Based on an MFC framework, the main interface of the server shows the skeleton of the participant's body in real time. 2) For further analysis, all data, including initial position data from clients and trilateration results, are exported to a hard disk. The open-source software OpenSim is chosen to handle the data. The software processes the position data with the human skeleton model and analyzes muscle flexing.

3.2 Transmission Component

Microsoft's Winsock Library (**winsock2.h**) provides functions to establish connections and receive/send data frames through the network for Windows 32 applications based on the TCP/IP protocol. The algorithms for binding the server with the port (Alg. 1) and establishing a connection between the client and the server (Alg. 2) are shown.

The Winsock API transmits data with a single-byte char stream, meaning that all types of messages (e.g., synchronization messages and calibration messages) are required to be transformed into single-byte char format when sending and receiving. The message format

Algorithm 1 Bind Server

Require: $Port$ The port through which the client is going to connect.
Ensure: $Socket_{Client}, Socket_{Server}$Sockets to transmit data with the connected client.

```
1: function BindServer(Port)
2:     Address ← InitAddr(0, Port)
3:     Socket_Server ← socket(AF_INET, SOCK_STREAM, 0)
4:     Bind(Socket_Server, Address)
5:     Listen(Socket_Server)
6:     Socket_Client ← accept(Socket_Server)
7:     returnSocket_Server, Socket_Client
8: end function
```

Algorithm 2 Connect to Server

Require: $IP, Port$ The IP address of the server and the port allocated to the client.
Ensure: $Socket_{Client}$Socket to transmit data to the server.

```
1: function ConnectServer(IP, Port)
2:     Address ← InitAddr(IP, Port)
3:     Socket_Client ← socket(AF_INET, SOCK_STREAM, 0)
4:     Socket_Client ← connect(Address)
5:     returnSocket_Client
6: end function
```

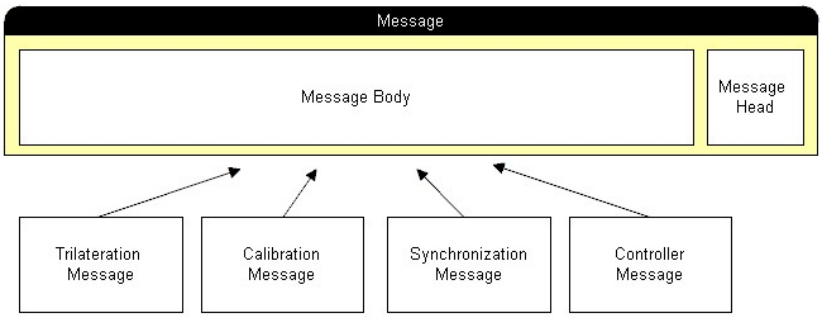


Figure 3.2: The structure of the message sent by the transmission component. The message head identifies the type of data stored in the message body. The message body is the string, consisting of single-byte chars.

Algorithm 3 Send Message through the Transmission Component

Require: $Data, Type$ The data and their type need to be sent. $Socket$ The destination of the message.

Ensure: The data will be sent in a single-byte char stream through the Winsock API.

```
1: function SendData(Data, Type, Socket)
2:     M.Data ← Data
3:     M.Type ← Type
4:     memcpy(String, M)
5:     send(Socket, String)
6: end function
```

Algorithm 4 Differentiate the Message

Require: $String$ The received message as a single-bit char string type.

Ensure: The data will be processed through a different function given its message type.

```
1: function Differentiate(String)
2:     M ← (Message)String
3:     switch M.Type do
4:         case TIMING_MESSAGE
5:             TimingMessageProc(M.Data)
6:         case TRILATERATION_MESSAGE
7:             TrilaterationMessageProc(M.Data)
8:         case CALIBRATION_MESSAGE_MESSAGE
9:         ... CalibrationMessageProc(M.Data)
10: end function
```

is designed to help both the server and the client differentiate the content (Fig. 3.2). The algorithms of the message process pipeline are shown in Alg. 3Alg. 4.

3.3 Synchronization Component

Currently, PTP [16] and NTP [30] are the most widely used synchronization protocols in the computer network field. There are many implementations of these protocols, including Chrony [11], NetTime [2], and Atomic Clock Sync [1]. The reason that a customized NTP synchronization scheme is built in our system rather than using off-the-shelf implementations is to reduce both the complexity of the system and the usage of resources:

1) Most implementations, such as Chrony, are designed to work in a network with multiple servers in a complex structure [32]. The network structure of our system is relatively simple, allowing us to build a customized NTP implementation that more precisely fits our requirements.

2) In contrast, compared with NTP, PTP is designed to achieve clock accuracy at the sub-microsecond level, which requires more network resources. However, the accuracy requirement for time synchronization in our proposed system is at the millisecond level (1 ms), which is 100 times lower than the accuracy provided by PTP (0.01 ms). Because the proposed system is based on LAN (Local Area Network), the time delays of mutual transmission are considered equal.

This server periodically sends synchronization messages to clients (in the server time T_1). When a client receives a synchronization message (in the client time T_2), it replies immediately (in the client time T_3). After that, the server receives the response (in server

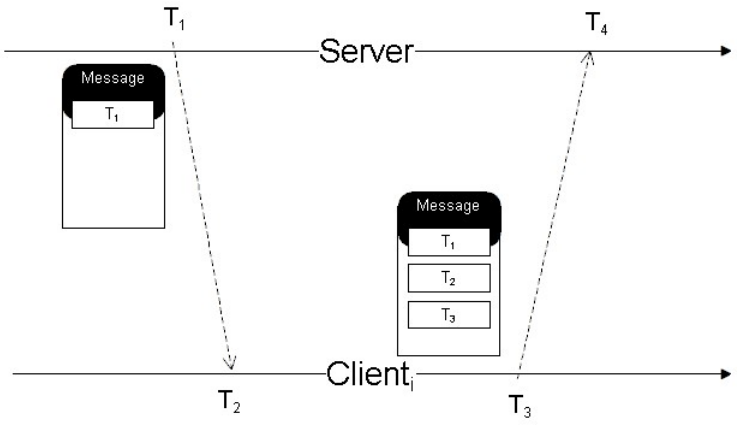


Figure 3.3: With the transmission between the server and the client, the local time of sending and receiving are stored in the synchronization message.

time T_4) of the synchronization message. The clock error and time delay between the server and client can be computed as

$$\begin{cases} T_2 - T_1 = D_i + E_i \\ T_4 - T_3 = D_i + E_i \end{cases} \tag{3.1}$$

where E_i denotes the clock error between the i-th client and the server and D_i indicates the time delay in transferring the data from the i-th client to the server. Derived from Equation 3.1, D_i and E_i can be computed as

$$\begin{cases} D_i = \frac{(T_4 - T_1) - (T_3 - T_2)}{2} \\ E_i = \frac{(-T_2 + T_1) + (T_4 - T_3)}{2} \end{cases} \tag{3.2}$$

Thus, the server can backtrack the sending time $T_{s,i}$ of a message (from the i-th client) based on its receiving time $T_{r,i}$, D_i, and E_i as

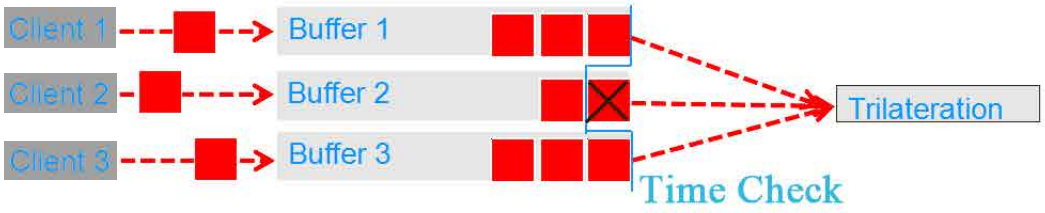


Figure 3.4: Red squares indicate the data frame sent by the clients. In this image, the first frame of buffer 2 is excluded by the time check.

$$T_{s,i} = T_{r,i} - D_i - E_i \tag{3.3}$$

All received data are stored in buffers, which are FIFO (first in first out) queues categorized by the serial number of the departure client (Fig. 3.4). The TCP/IP protocol protects the sequence in time of the data streams in the local network [24]. However, some of the data are occasionally lost during the transmission. In this case, the sending time is verified when the trilateration component extracts data from the buffer. In each frame of the server (Alg. 5), a time range is determined by adding the extra offset time (i.e., $\frac{1000}{30}$ ms at an fps of 30) to the current server time, where the data exceeding the mentioned time range are excluded.

3.4 Calibration Component

3.4.1 Calibration Wand Localization

The main purpose of this step is to generate the area that covers the calibration wand from the whole 2D depth image for further calculation. The color image and the corresponding

Algorithm 5 Data Buffer Time Check

Require: $Buffer$ The array of FIFO queues; each queue stores the data frames from one client. N Number of clients.

Ensure: Data frames that exceed the time range are excluded.

```
1: function TimeCheck(Buffer, N)
2:     LatestTime ← 0
3:     for i = 0 → N − 1 do
4:         if Buffer^i_End.Time > LatestTime then
5:             LatestTime ← Buffer^i_Top.Time
6:         end if
7:     end for
8:     for i = 0 → N − 1 do
9:         if Buffer^i_End.Time < LatestTime − 30 then
10:            DeleteEnd(Buffer^i)
11:        end if
12:    end for
13: end function
```

depth image of the calibration wand are shown in Fig.3.5. A bitmap of depth D is defined to indicate the depth image obtained by the Kinect, where D_{ij} refers to the distance in centimeters between the Kinect and the corresponding point of the object's surface. i and j indicate the horizontal and vertical coordinates of the depth image, respectively. Because the designed detection range of the Kinect is from 0.5 m to 6.5 m, an object in the depth image exceeding such a range results in inaccurate performance. The standardization of D is conducted via the following initialization:

$$D_{ij} = \begin{cases} 50 & D_{ij} < 50 \\ 650 & D_{ij} > 650 \\ D_{ij} & otherwise. \end{cases} \tag{3.4}$$

As shown in Fig.3.5, pixels constituting the image of the wand are orange. To obtain the data from the image, a iterated binarization procedure is conducted to differentiate

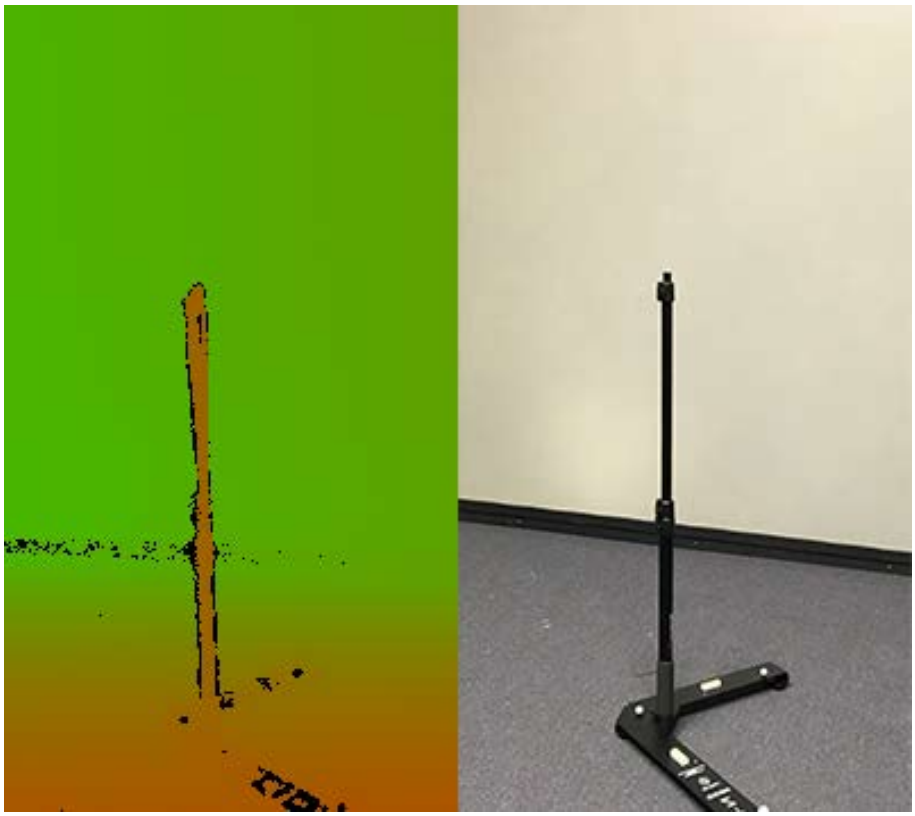


Figure 3.5: Color image (left) and depth image (right) of the calibration wand in a strongly disturbed background. The black pixels in the depth image refer to the noise and shadow.

the image pixels of the calibration wand from the background. The bitmap T is defined to indicate the result of the binarization, where elements of 1 or 0 indicate that the element in D corresponds to the calibration wand or not. In each iteration, T_{ij}^{k} (the element in the bitmap with coordinates (i, j) in the k-th iteration) is computed as

$$T_{ij}^{k} = \begin{cases} 1 & D_{ij} <= Ave\left(D_{ij} | T_{ij}^{k-1} = 1\right) \cdot C_{offset} \\ 0 & otherwise. \end{cases} \tag{3.5}$$

where all elements in the initial bitmap, i.e, T^1, are set as 1. C_{offset} indicates the parameter used to avoid the occurred offset, and $Ave\,(\bullet)$ refers to the average of the set $(\bullet)$. In this

(a)

(b)

Figure 3.6: The original depth image of the calibration wand (a) and the depth image after the binarization iteration algorithm. In (b), only the wand is kept in the image.

paper, the coefficient C_{offset} is chosen as 0.93.

$$\text{Min}(D_{ij}|T_{ij}^{k-1}=1)+WandDiameter >= \text{Max}(D_{ij}|T_{ij}^{k-1}=1)) \tag{3.6}$$

where $WandDiameter$ is 3.5 cm measured in practical experiments.

The iteration ends if the condition (Equation 3.6) is satisfied, meaning that only the wand in the depth image is chosen (Fig. 3.6). The full binarization algorithm is shown in Alg. 6

According to T, the boundary of the calibration wand in the depth image is determined by the difference in the radius. Therefore, the upper-left and bottom-right points of the image are defined as P_{ul} and P_{br}, respectively. The coordinate of the center point of the calibration wand's image can be calculated through these two points, which are defined as P_m.

According to the technical specification [37], the range of the viewing field of the Kinect v2 is 70° and 60° in the horizontal and vertical directions, respectively. Because the size of the depth image is 512×424 pixels, the central point of the image P_c is $(256, 212)$. The 3D coordinates of a point in the real world (denoted as $P'(x_{P'}, y_{P'}, z_{P'})$), corresponding to the point in the depth image (denoted as $P(x_P, y_P, z_P)$), are calculated as

$$\begin{cases} x_{P'} = & D_P \cdot tan(70° \cdot \frac{x_P - 256}{512}) \\ y_{P'} = & D_P \cdot tan(60° \cdot \frac{y_P - 212}{424}) \\ z_{P'} = & D_P \end{cases} \tag{3.7}$$

where the diagram is shown in Fig. 3.7.

Algorithm 6 Depth Image Binarization

Require: $Depth$ The bitmap of the depth image. $Height, Width$ The height and width of the depth image.

Ensure: $Result$ The bitmap with 0 and 1, where only the pixels of the wand's image are given by a 1 in $Result$.

```
1: function PickMinMax(Depth, Height, Width, Result)
2:     Min ← 2 × 10^31
3:     Max ← 0
4:     for i = 0 → Height − 1 do
5:         for j = 0 → Width − 1 do
6:             if Result_{i,j} = 1 then
7:                 if Depth_{i,j} < Min then
8:                     Min = Depth − i, j
9:                 end if
10:                if Depth_{i,j} > Max then
11:                    Max = Depth − i, j
12:                end if
13:            end if
14:        end for
15:    end for
16:    return Min, Max
17: end function
18: function Binarization(Depth, Height, Width)
19:    Memset(Result, 1)
20:    [Min, Max] =← PickMinMax(Depth, Height, Width, Result)
21:    while Min + 3.5 < Max do
22:        Average ← 0
23:        Count ← 0
24:        for i = 0 → Height − 1 do
25:            for j = 0 → Width − 1 do
26:                if Result_{i,j} = 1 then
27:                    Average ← Average + Depth_{i,j}
28:                    Count ← Count + 1
29:                end if
30:            end for
31:        end for
```

```
        Average ← Average ÷ Count × 0.93
        for i = 0 → Height − 1 do
            for j = 0 → Width − 1 do
                if Depth_{i,j} ≤ Average then
                    Resulti, j ← 1
                else
                    Result_{i,j} ← 0
                end if
            end for
        end for
    end while
    return Result
end function
```

However, the origin of the system's coordinates is the center of the calibration wand instead of the center of the calibration wand's surface. Therefore, once the coordinate of the point $P'_m(x_{P'_m}, y_{P'_m}, z_{P'_m})$ corresponding to P_m is calculated, the radius of the calibration wand (i.e., $x_{P_{br}} - x_{P_{ul}}$) needs to be considered as well. The coordinates of the center of the calibration wand in the coordinate system of the client $P'_o(x_{P'_o}, y_{P'_o}, z_{P'_o})$ are calculated as

$$\begin{cases} x_{P'_o} = & x_{P'_m} \\ y_{P'_o} = & y_{P'_m} \\ z_{P'_o} = & z_{P'_m} + (x_{P_{br}} - x_{P_{ul}}) \end{cases} \tag{3.8}$$

To enhance the robustness of this calibration wand recognition, the client sensor obtains a measurement several times in each calibration frame and compares the difference in the measurement results. The frame with a higher deviation in the measurement is ignored.

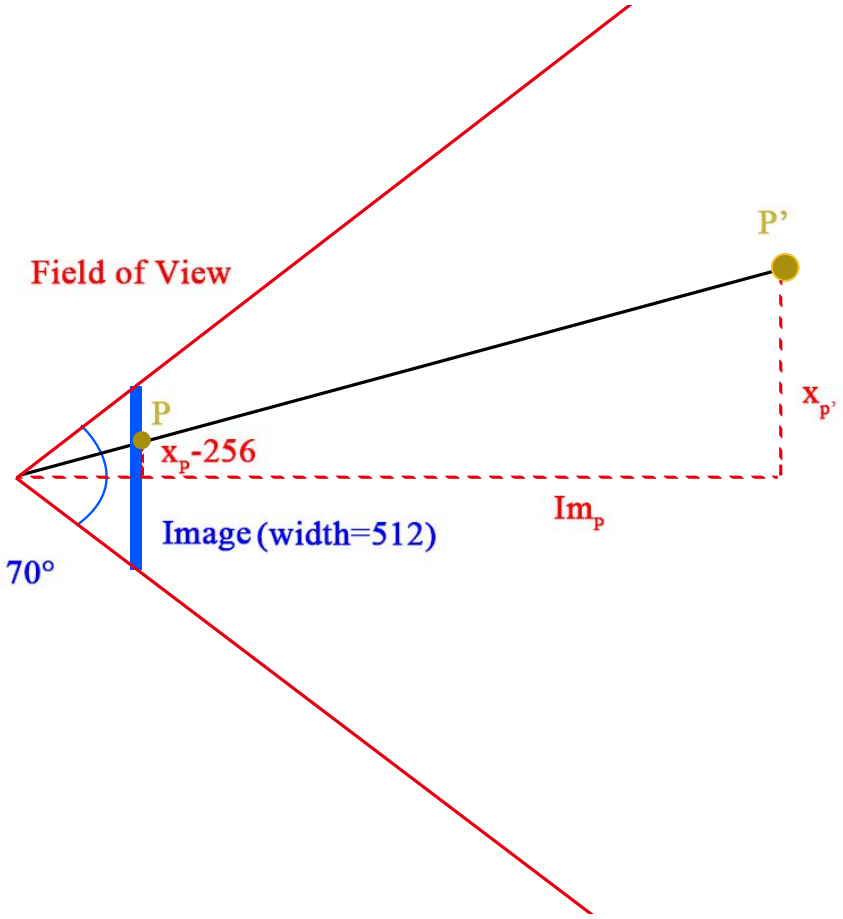


Figure 3.7: The geometric relationship between coordinates in the depth image and those in the real world of the x-axis. Here, P' is the point in the real world, and P is the corresponding point in the depth image. The blue line indicates the shot of the depth sensor.

3.4.2 Transformation from Client's Coordinates to Server's Coordinates

In our proposed method, all Kinects are located to be perpendicular to the ground surface (Fig. 3.8), meaning that the z-axes of all coordinate systems (including server's and clients' coordinate systems) are parallel. Therefore, there is no rotation transformation along the z-axis. The rotation angles are calculated by the differences between the coordinate

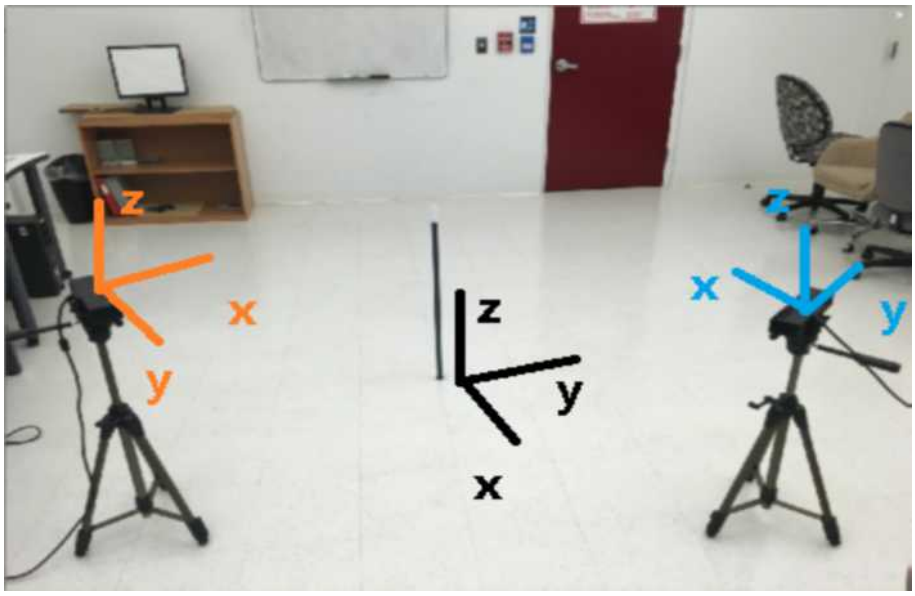


Figure 3.8: The system's coordinates (black) and those of the other two Kinect sensors' local coordinates (orange and blue). Their z-axes are all parallel.

movements (along the x-axis and y-axis) within the server's and clients' coordinate systems.

The client obtains the position of the calibration wand twice, thereby defining a trajectory of the calibration wand. Here, we refer to this trajectory as the y-axis in the system's coordinates. The rotation angle θ of the y-axis can be calculated as

$$\theta = arccos\left(\frac{\Delta y'}{\Delta y}\right) \tag{3.9}$$

where Δ refers to the change operator and y' and y indicate the y coordinate of the calibration wand in the client's coordinates and in the system's coordinates, respectively.

Given the relative displacement of the origins and the rotation angle of the axes, a point in the client's coordinates $(P'(x_{P'}, y_{P'}, z_{P'}))$ can be transformed into its corresponding point in the system's coordinates $(P^s(x_{P^s}, y_{P^s}, z_{P^s}))$ as

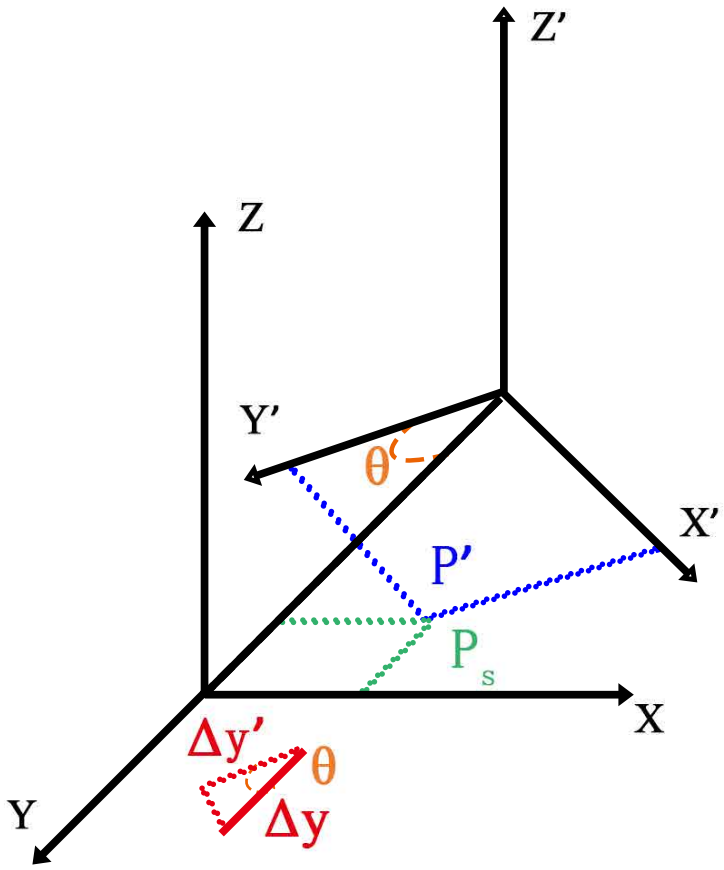


Figure 3.9: Defining θ, P_s, P', Δy and $\Delta y'$ in the transformation from the client's coordinates to the system's coordinates.

$$\begin{cases} x_{P^s} = & x_{P'} \cdot cos(\theta) - y_{P'} \cdot sin(\theta) \\ y_{P^s} = & x_{P'} \cdot sin(\theta) + y_{P'} \cdot cos(\theta) \\ z_{P^s} = & z_{P'} \end{cases} \tag{3.10}$$

3.5 Trilateration Component

This component is used to combine the measurements from each sensor to obtain a more accurate trilateration result [33]. The coordinate transformation (Section 3.4) is conducted through the result from the calibration component before the measurement is applied to the nonlinear trilateration calculation. Once this is performed, this trilateration component starts the process whereby the server continuously receives measurements from clients that have the coordinates of the human body's joints.

3.5.1 Linearized Equation

The basic idea of how to obtain an accurate trilateration result is to treat the joint point P_c as the intersection of several spheres [44] [27], whose centers P_{s_i} are the positions of n sensors. And exact distances between the sensor and joint point are r_i, which indicate the radii of the spheres as well. The formula for those spheres is

$$\left(x_{P_c} - x_{P_{s_i}}\right)^2 + \left(y_{P_c} - y_{P_{s_i}}\right)^2 + \left(z_{P_c} - z_{P_{s_i}}\right)^2 = r_i^2 \tag{3.11}$$

The intersection point of those n spheres' surfaces can be obtained by solving the system of quadratic equations. However, this solution technique is not feasible because it produces

a system of nonlinear equations of high degree.

$$\begin{cases} \left(x_{P_c}-x_{P_{s_1}}\right)\left(x_{P_{s_2}}-x_{P_{s_1}}\right)+\left(y_{P_c}-y_{P_{s_1}}\right)\left(y_{P_{s_2}}-y_{P_{s_1}}\right) \\ \quad +\left(z_{P_c}-z_{P_{s_1}}\right)\left(z_{P_{s_2}}-z_{P_{s_1}}\right)=\frac{\left(r_1^2-r_2^2+d_{12}^2\right)}{2}=b_1 \\ \left(x_{P_c}-x_{P_{s_1}}\right)\left(x_{P_{s_3}}-x_{P_{s_1}}\right)+\left(y_{P_c}-y_{P_{s_1}}\right)\left(y_{P_{s_3}}-y_{P_{s_1}}\right) \\ \quad +\left(z_{P_c}-z_{P_{s_1}}\right)\left(z_{P_{s_3}}-z_{P_{s_1}}\right)=\frac{\left(r_1^2-r_3^2+d_{13}^2\right)}{2}=b_2 \\ \vdots \\ \left(x_{P_c}-x_{P_{s_1}}\right)\left(x_{P_{s_n}}-x_{P_{s_1}}\right)+\left(y_{P_c}-y_{P_{s_1}}\right)\left(y_{P_{s_n}}-y_{P_{s_1}}\right) \\ \quad +\left(z_{P_c}-z_{P_{s_1}}\right)\left(z_{P_{s_n}}-z_{P_{s_1}}\right)=\frac{\left(r_1^2-r_n^2+d_{1n}^2\right)}{2}=b_n \end{cases} \tag{3.12}$$

where

$$d_{1i}=\sqrt{\left(x_{P_{s_1}}-x_{P_{s_i}}\right)^2+\left(y_{P_{s_1}}-y_{P_{s_i}}\right)^2+\left(z_{P_{s_1}}-z_{P_{s_i}}\right)^2} \tag{3.13}$$

$$A\vec{x}=\vec{b} \tag{3.14}$$

where

$$\begin{gathered} A=\begin{pmatrix} x_{P_{s_2}}-x_{P_{s_1}} & y_{P_{s_2}}-y_{P_{s_1}} & z_{P_{s_2}}-z_{P_{s_1}} \\ x_{P_{s_3}}-x_{P_{s_1}} & y_{P_{s_3}}-y_{P_{s_1}} & z_{P_{s_3}}-z_{P_{s_1}} \\ \vdots & \vdots & \vdots \\ x_{P_{s_n}}-x_{P_{s_1}} & y_{P_{s_n}}-y_{P_{s_1}} & z_{P_{s_n}}-z_{P_{s_1}} \end{pmatrix}, \\ \vec{x}=\begin{pmatrix} x_{P_c}-x_{P_{s_1}} \\ y_{P_c}-y_{P_{s_1}} \\ z_{P_c}-z_{P_{s_1}} \end{pmatrix}, \vec{b}=\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}. \end{gathered} \tag{3.15}$$

The linearized equation method applies a geometrical principle to convert the problem into one of solving linear equations. The equation of one sensor in the system (e.g., the first sensor) would be chosen as the linearizing tool to regroup and simplify the equations; the results are shown in Equation 3.12, which can be easily written in matrix form, as shown in Equation 3.14. Thus, the system has in total (n-1) linear equations with 3 unknown values, meaning that result can be obtained by applying Gaussian elimination through any 3 equations from the system. Giving the sequence number of selected equations c, d, e, the Gaussian matrix leads to

$$G = \left(\begin{array}{ccc|c} A_{c1} & A_{c2} & A_{c3} & b_c \\ A_{d1} & A_{d2} & A_{d3} & b_d \\ A_{e1} & A_{e2} & A_{e3} & b_e \end{array}\right) \tag{3.16}$$

When the data input to the system of equations are accurate, solving those equations can produce an accurate result. However, approximate data are always used in practice, meaning that direct results would be unacceptable.

The accuracy of the linearized equation method can be improved by combining results from different sets of equations chosen to solve when there are in total n sensors ($n > 3$). The formula for this advanced linearized equation for obtaining the p_c method are shown below:

$$p_c = \overline{\{p_c'\}} \tag{3.17}$$

where $\{p_c'\}$ indicates the array of results obtained by choosing different sets of equations from Equation3.12 applying the combinatorics principle. Thus, the length of the $\{p_c'\}$ is

C_{n-1}^{3}.

3.5.2 Linear Least Square

Applying the linear least square method to the system of linear equations in Equation3.14 can obtain a more accurate result than the result obtained by directly solving 4 equations from the system of linear equations. In the trilateration procedure, r_i is the only approximate parameter; the goal of the procedure is to obtain the $\vec{x}'$ that minimizes the objective function S, which is given by

$$S(\vec{x}') = \vec{r}^{\mathrm{T}}\vec{r} = \left(\vec{b} - A\vec{x}'\right)^{\mathrm{T}}\left(\vec{b} - A\vec{x}'\right) \tag{3.18}$$

This minimization problem has a unique solution, given by solving the normal equations:

$$\left(A^{\mathrm{T}}A\right)\vec{x}' = A^{\mathrm{T}}\vec{b} \tag{3.19}$$

Thus, the best fit coefficients are expressed as

$$\vec{x}' = \left(A^{\mathrm{T}}A\right)^{-1}A^{\mathrm{T}}\vec{b} \tag{3.20}$$

However, the accuracy and precision of the result obtained by solving using the linear least square method cannot meet the requirements in practice [8]. Moreover, when $A^{\mathrm{T}}A$ is singular [17] or poorly conditioned [18], meaning that the inverse of such a matrix cannot be accurately obtained, SVD (Singular Value Decomposition) is required.

3.5.3 Nonlinear Least Square

Specifically, the nonlinear least square method calculates an accurate position estimation with approximately accurate input measurements. First, the estimation error [15] of the current trilaterated joint coordinate $P(x_P, y_P, z_P)$ with respect to the i-th sensor $f_i(P)$ is defined as

$$f_i(P) = \sqrt{(x_P - x_i)^2 + (y_P - y_i)^2 + (z_P - z_i)^2} - r_i \tag{3.21}$$

where (x_i, y_i, z_i) denotes the coordinates of the i-th sensor and r_i denotes the distance between the true joint position and the i-th sensor's position. Here, an objective function F is defined as the sum of the square of the estimation errors. Obtaining an optimized joint position by nonlinear trilateration is equivalent to solving the following minimization problem regarding F:

$$\min F(P) = \sum_{i=1}^{n} f_i(P)^2 \tag{3.22}$$

Considering the balance between the complexity and accuracy, Newton iteration is chosen [33] [35] to solve the optimization problem in Equation 3.22. More specifically, the partial derivative of the objective function F with respect to x, y, and z yields

$$\begin{cases} \frac{\partial F}{\partial x} = 2\sum_{i=1}^{n} f_i(\frac{\partial f_i}{\partial x}) \\ \frac{\partial F}{\partial y} = 2\sum_{i=1}^{n} f_i(\frac{\partial f_i}{\partial y}) \\ \frac{\partial F}{\partial z} = 2\sum_{i=1}^{n} f_i(\frac{\partial f_i}{\partial z}) \end{cases} \tag{3.23}$$

Referring to Newton iteration, the vectors $\vec{g}$ and $\vec{R}$ are defined as follows:

$$\underbrace{\begin{pmatrix} \frac{\partial F}{\partial x} \\ \frac{\partial F}{\partial y} \\ \frac{\partial F}{\partial z} \end{pmatrix}}_{\vec{g}} = 2 \underbrace{\begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial x} & \frac{\partial f_n}{\partial y} & \frac{\partial f_n}{\partial z} \end{pmatrix}^{\mathrm{T}}}_{J} \underbrace{\begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}}_{\vec{f}}, \tag{3.24}$$

and

$$\vec{R} = \begin{pmatrix} x_P \\ y_P \\ z_P \end{pmatrix} \tag{3.25}$$

Assume that the intermediate solution of $\vec{R}$ in the k-th iteration is denoted as $\vec{R}_k$. According to the Newton iteration, the solution in the next iteration $\vec{R}_{k+1}$ can be computed as

$$\vec{R}_{k+1} = \vec{R}_k - \left(J_k^{\mathrm{T}} J_k\right)^{-1} J_k^{\mathrm{T}} \vec{f}_k \tag{3.26}$$

The initial solution $\vec{R}_1$ can be obtained by the linear least square method [52]. Regarding the iteration rule in Equation 3.26, on the right side, we provide the calculation of $J^T J$ and $J^{\mathrm{T}} \vec{f}$ as follows:

$$J^{\mathrm{T}} J = \begin{pmatrix} A_1 & A_2 & A_3 \end{pmatrix} \tag{3.27}$$

where

$$A_1 = \begin{pmatrix} \sum_{i=1}^{n} \frac{(x_P - x_i)^2}{(f_i + r_i)^2} \\ \sum_{i=1}^{n} \frac{(x_P - x_i)(y_P - y_i)}{(f_i + r_i)^2} \\ \sum_{i=1}^{n} \frac{(x_P - x_i)(z_P - z_i)}{(f_i + r_i)^2} \end{pmatrix}$$

$$A_2 = \begin{pmatrix} \sum_{i=1}^{n} \frac{(x_P - x_i)(y_P - y_i)}{(f_i + r_i)^2} \\ \sum_{i=1}^{n} \frac{(y_P - y_i)^2}{(f_i + r_i)^2} \\ \sum_{i=1}^{n} \frac{(y_P - y_i)(z_P - z_i)}{(f_i + r_i)^2} \end{pmatrix} \tag{3.28}$$

$$A_3 = \begin{pmatrix} \sum_{i=1}^{n} \frac{(x_P - x_i)(z_P - z_i)}{(f_i + r_i)^2} \\ \sum_{i=1}^{n} \frac{(y_P - y_i)(z_P - z_i)}{(f_i + r_i)^2} \\ \sum_{i=1}^{n} \frac{(z_P - z_i)^2}{(f_i + r_i)^2} \end{pmatrix}$$

and

$$J^{\mathrm{T}} \vec{f} = \begin{pmatrix} \sum_{i=1}^{n} \frac{(x_P - x_i) f_i}{(f_i + r_i)} \\ \sum_{i=1}^{n} \frac{(y_P - y_i) f_i}{(f_i + r_i)} \\ \sum_{i=1}^{n} \frac{(z_P - z_i) f_i}{(f_i + r_i)} \end{pmatrix} \tag{3.29}$$

Because $J^{\mathrm{T}}J$ is a matrix with a size of 3 by 3, an adjunct method [45] is applied to calculate its inverse matrix to improve the efficiency as follows:

$$\left(J^{\mathrm{T}}J\right)^{-1} = \frac{\left(J^{\mathrm{T}}J\right)'}{\left|J^{\mathrm{T}}J\right|} \tag{3.30}$$

where $\left(J^{\mathrm{T}}J\right)'$ indicates the adjunct, which is the transpose of the cofactor matrix of $J^{\mathrm{T}}J$, and $\left|J^{\mathrm{T}}J\right|$ denotes the determinant of $J^{\mathrm{T}}J$.

Finally, the iteration is designed to stop when the two adjacent iterative results of the objective function $F(P)$ are sufficiently small, i.e.,

Table 3.1: Comparison of Efficiency and Accuracy with Different Measurement Error Ranges and $C_{threshold}$

$C_{threshold}$	Measurement Error Range					
	(15, 30] cm		[5, 15] cm		[0, 5) cm	
	Propagation Error (cm)	Time Consumption (μs)	Propagation Error (cm)	Time Consumption (μs)	Propagation Error (cm)	Time Consumption (μs)
1	21.4	< 0.1	17.2	< 0.1	9.3	< 0.1
1×10^{-2}	15.7	0.6	12.0	0.4	7.5	0.3
1×10^{-4}	14.2	11	7.3	10	3.9	7
1×10^{-6}	13.7	1040	6.6	783	3.6	651

$$F^k(P) - F^{k-1}(P) < C_{threshold} \tag{3.31}$$

where $F^k(P)$ and $F^{k-1}(P)$ denote the results of the objective function in the k-th and $(k-1)$-th iterations, respectively, and $C_{threshold}$ denotes the threshold of the adjacent iterative result. The full algorithm of the Newton Iteration method is shown in Alg. 7.

In this paper, the experiment used to determine $C_{threshold}$ is conducted such that the trilateration input (i.e., 4 Kinect sensor measurements) is simulated with different errors and thresholds to test the time efficiency and accuracy. Each set of experiments takes 5000 sets of data, such that the standard deviation of the result is less than 0.001%. The results of the experiments are shown in Table 3.1. According to the results, when $C_{threshold}$ decreases, the accuracy improves and the time consumption increases. When $C_{threshold}$ is set to 1×10^{-6}, compared with the case when $C_{threshold}$ is 1×10^{-4}, the accuracy improves slightly, whereas the time consumption apparently increases. For instance, the time consumption is more than 1 millisecond when the mean measurement error is larger than 15 cm. Thus, in this paper, $C_{threshold}$ is chosen as 1×10^{-4}.

Algorithm 7 Newton Iteration Method

Require: *Initial* The matrix of the initial solution of the joint position. P Array of sensor positions. D Array of obtained distances between the joint and the sensors. N The number of sensors.

Ensure: *Result* The optimized joint position.

function ERROR($Result, P, D$)
  $E \leftarrow 0$
  for $i \leftarrow 0 \rightarrow N-1$ **do**
    $E \leftarrow E + (\text{EUCLID}(Result, P_i) - D)^2$
  end for
  return E
end function
function NEWTOWNITERATION($Initial, P, D$)
  $Result \leftarrow Initial$
  $Current \leftarrow \text{ERROR}(Result, P, D)$
  do
    for $i \leftarrow 0 \rightarrow N-1$ **do**
      $J^{\mathrm{T}}J_{0,0} \leftarrow J^T J_{0,0}(x_{Result} - x_{P_i})^2 \div SensorError^2$
      $J^{\mathrm{T}}J_{0,1} \leftarrow J^T J_{0,1}(x_{Result} - x_{P_i}) * (y_{Result} - y_{P_i}) \div SensorError^2$
      $J^{\mathrm{T}}J_{0,2} \leftarrow J^T J_{0,2}(x_{Result} - x_{P_i}) * (z_{Result} - z_{P_i}) \div SensorError^2$
      $J^{\mathrm{T}}J_{1,1} \leftarrow J^T J_{1,1}(y_{Result} - y_{P_i})^2 \div SensorError^2$
      $J^{\mathrm{T}}J_{1,2} \leftarrow J^T J_{1,2}(y_{Result} - y_{P_i}) * (z_{Result} - z_{P_i}) \div SensorError^2$
      $J^{\mathrm{T}}J_{2,2} \leftarrow J^T J_{2,2}(z_{Result} - z_{P_i})^2 \div SensorError^2$
      $x_{J^{\mathrm{T}}\vec{f}} \leftarrow (x_{Result} - x_{P_i}) * (SensorError - R_i) \div SensorError$
      $y_{J^{\mathrm{T}}\vec{f}} \leftarrow (y_{Result} - y_{P_i}) * (SensorError - R_i) \div SensorError$
      $z_{J^{\mathrm{T}}\vec{f}} \leftarrow (z_{Result} - z_{P_i}) * (SensorError - R_i) \div SensorError$
    end for
    $J^{\mathrm{T}}J_{1,0} \leftarrow J^{\mathrm{T}}J_{0,1}$
    $J^{\mathrm{T}}J_{2,0} \leftarrow J^{\mathrm{T}}J_{0,2}$
    $J^{\mathrm{T}}J_{2,1} \leftarrow J^{\mathrm{T}}J_{1,2}$
    $Result \leftarrow Result - J^{\mathrm{T}}J^{-1}J^{\mathrm{T}}\vec{f}$
    $Last \leftarrow Current$
    $Current \leftarrow \text{ERROR}(Result, P, D)$
  while $Last - Current > 1 \times 10^{-4}$
  return*Result*
end function

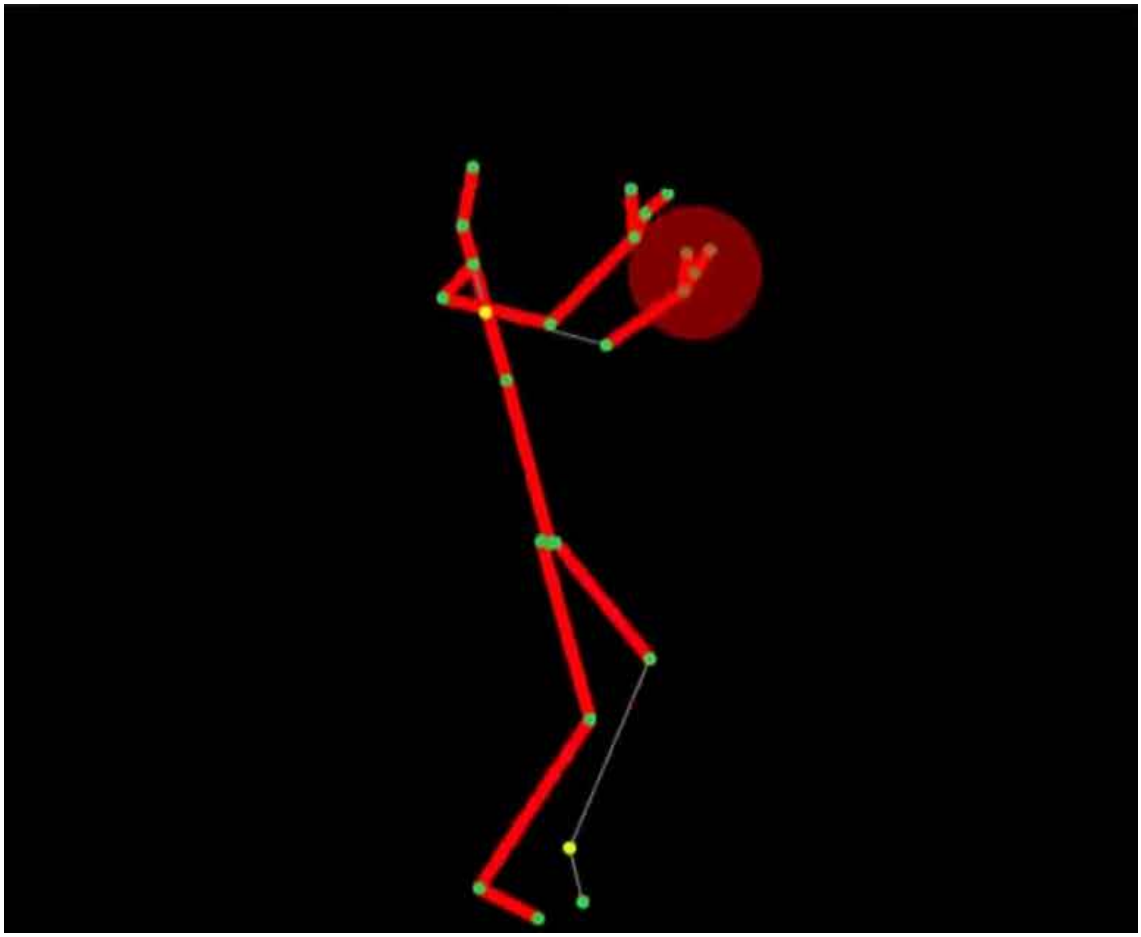


Figure 3.10: The direct body tracking output of the Kinect API. Red thick segments indicate tracking body parts, while white thin segments indicate inferred body parts.

3.6 Occlusion Compensation Component

The Kinect v2 is designed to obtain inferred results when it is not able to make a direct measurement, leading to the accuracy of the measurement being insufficient compared with the directly tracked results. The inferred results are first marked by the Kinect's SDK, as shown in Fig. 3.10. In the image, the participant's left arm and left leg are partly occluded by the right side of their body. In addition, the client analyzes the positions of limbs to determine if the results are affected by occlusion.

Because the image of the human body is abstracted as combined segments in 3D coordinates, the occlusion of limbs in this paper is defined as the crossing of segments on the projection in the Kinect's view. The vector product from the computational geometry principle [28] is applied to determine the crossing of segments V_1 and V_2. Two segments are crossed when 6 conditions in Equation 3.32 are met.

$$\begin{cases} \text{Min}(x_{V_1,1}, x_{V_1,2}) - \text{Max}(x_{V_2,1}, x_{V_2,2}) & <= 0 \\ \text{Min}(x_{V_2,1}, x_{V_2,2}) - \text{Max}(x_{V_1,1}, x_{V_1,2}) & <= 0 \\ \text{Min}(y_{V_1,1}, y_{V_1,2}) - \text{Max}(y_{V_2,1}, y_{V_2,2}) & <= 0 \\ \text{Min}(y_{V_2,1}, y_{V_2,2}) - \text{Max}(y_{V_1,1}, y_{V_1,2}) & <= 0 \\ VectorProduct \quad (x_{V_1,1} - x_{V_2,1}, y_{V_1,1} - y_{V_2,1}, & \\ \qquad x_{V_2,2} - x_{V_2,1}, y_{V_2,2} - y_{V_2,1}) \cdot & \\ VectorProduct \quad (x_{V_1,2} - x_{V_2,1}, y_{V_1,2} - y_{V_2,1}, & \\ \qquad x_{V_2,2} - x_{V_2,1}, y_{V_2,2} - y_{V_2,1}) & < 0 \\ VectorProduct \quad (x_{V_2,1} - x_{V_1,1}, y_{V_2,1} - y_{V_1,1}, & \\ \qquad x_{V_1,2} - x_{V_1,1}, y_{V_1,2} - y_{V_1,1}) \cdot & \\ VectorProduct \quad (x_{V_2,2} - x_{V_1,1}, y_{V_2,2} - y_{V_1,1}, & \\ \qquad x_{V_1,2} - x_{V_1,1}, y_{V_1,2} - y_{V_1,1}) & < 0 \end{cases} \tag{3.32}$$

where $x_{V_i,j}$ indicates the x coordinate of the j-th point's coordinates in segment V_i and similarly for $y_{V_i,j}$ and $z_{V_i,j}$. $VectorProduct(x_1, x_2, y_1, y_2)$ refers to the vector product operator, which is $(x_1 \cdot y_2 - x_2 \cdot y_1)$.

It is noted that two crossed segments are intersected at a single point unless they are overlapped. Furthermore, the sequence of crossed segments is determined by the spacial

distance to the Kinect sensor (i.e., the coordinate z) of the intersection point. Using the result of the vector product of two segments, the coordinates of the intersection point are calculated, and their sequence can be further obtained. Therefore, the occluded limbs are determined by calculating the intersection points of all pairs of segments in the image of the human body and comparing the z coordinate of the intersection point. The occluded data are marked by the client before being sent to server. Here, data with the minimum time of occlusion [54] are transferred to the next procedure, and data from at least half of the clients are selected. The equations used to obtain the z coordinate of the intersection point $p_{Intersection,1}$ and $p_{Intersection,2}$ are shown in Equation: 3.33

$$\begin{cases} z_{p_{Intersection,1}} = & (z_{V_1,2} - z_{V_1,1}) \cdot ((y_{V_1,1} - y_{V_2,1}) \cdot (x_{V_2,2} - x_{V_2,1}) - (x_{V_1,1} - x_{V_2,1}) \cdot (y_{V_2,2} - y_{V_2,1})) \\ & / ((x_{V_1,2} - x_{V_1,1}) \cdot (y_{V_2,2} - y_{V_2,1}) - (x_{V_2,2} - x_{V_2,1}) \cdot (y_{V_1,2} - y_{V_1,1})) + z_{V_1,1}; \\ z_{p_{Intersection,2}} = & (z_{V_2,2} - z_{V_2,1}) \cdot ((x_{V_1,1} - x_{V_2,1}) \cdot (y_{V_1,2} - y_{V_1,1}) - (y_{V_1,1} - y_{V_2,1}) \cdot (x_{V_1,2} - x_{V_1,1})) / \\ & ((x_{V_2,2} - x_{V_2,1}) \cdot (y_{V_1,2} - y_{V_1,1}) - (x_{V_1,2} - x_{V_1,1}) \cdot (y_{V_2,2} - y_{V_2,1})) + z_{V_2,1}; \end{cases} \tag{3.33}$$

The algorithm used to exclude inaccurate results on the server side is shown in Alg. 8.

3.7 Visualization Component

3.7.1 Real-time Skeleton Visualization

After completing the trilateration procedure, the server calculates the joint position data of the current time period. Here, the component is design to draw the z-axis projection of the participant's skeleton on the application's interface, meaning that the z coordinates

Algorithm 8 Exclude Inaccurate Results

Require: *Shade* Array of the times for which the joint is occluded by others from each client. *Joint* Array of joint position data from each client. n Number of clients.

Ensure: *Available* Number of joint data that will be processed by the trilateration component.

1: **function** ExcludeInaccurateResult($Shade, n$)
2: Sort($Joint, Shade$) ▷ Sort both the *Joint* and *Shade* arrays in ascending order by comparing the *Shade* as the key.
3: $Available \leftarrow 1$
4: **while** $Shade_{Available} \leq 0 \vee Available \geq$ Floor$(n/2)$ **do**
5: $Available \leftarrow Available + 1$
6: **end while**
7: **return** *Available*
8: **end function**

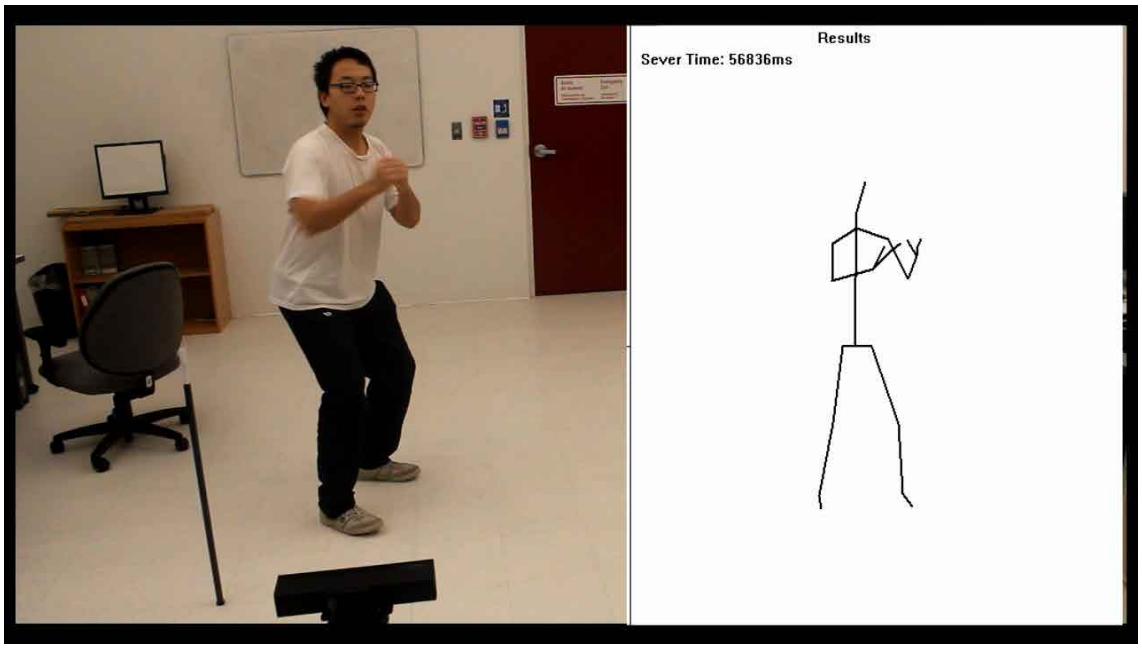


Figure 3.11: The image of the participant (left) and the visualization result shown on the screen (right). Here, the system's coordinates are set to be opposite from the real world in the measurement; therefore, the visualization results are reversed.

of the joint position data are excluded in the drawing. Once the position data of a new frame are transmitted to the visualization component of the visualization, the component first normalizes the coordinates in terms of scale and relative position and then erases the image in the interface and draws the new projection.

Microsoft's Windows-specific standard library **windows.h** provides functions for drawing graphs in 32-bit Windows applications. The algorithm design of the feature is shown in Alg. 9, and the result and the corresponding participant's image are shown in Fig. 3.11

3.7.2 Human Body Kinematic Visualization

Furthermore, with the Inverse Kinematics (IK) [48] module and the Computed Muscle Control (CMC) [46] module from OpenSim [12], the server then obtains the joint angles and translations as coordinated values that best match the trilateration result and generates a set of muscle excitations that produce a coordinated muscle-driven simulation of the model's movement.

The Inverse Kinematics are used to solve a least-squares problem that minimizes the differences between the measured joint positions from the trilateration results and the model's virtual joint positions, subject to joint constraints. The weighted squared error is formulated as

$$SquaredError = \sum_{i=1}^{jointpositions} w_i(\bar{x}_i^{subject} - \bar{x}_i^{model})^2 + \sum_{j=1}^{jointangles} \omega_j(\theta_j^{subject} - \theta_j^{model})^2 \tag{3.34}$$

Algorithm 9 Real-time Skeleton Visualization

Require: $Joint$ The joint position data of the participant, $hwnd$ The handle of the window. $ImageWidth$, $ImageHeight$ The width and height of the visualization area.

Ensure: The projection of the participant's skeleton on the screen.

function DrawLine($Joint_1, Joint_2$)
    $x_1 \leftarrow$ CoordinateTransform($x_{Joint_1} - x_{Datum}, ImageWidth$)
    $y_1 \leftarrow$ CoordinateTransform($y_{Joint_1} - y_{Datum}, ImageHeight$)
    $x_2 \leftarrow$ CoordinateTransform($x_{Joint_2} - x_{Datum}, ImageWidth$)
    $y_2 \leftarrow$ CoordinateTransform($y_{Joint_2} - y_{Datum}, ImageHeight$)
    MoveToEx(x_1, y_1)
    LineTo(x_2, y_2)
end function
function DrawSkeleton($Joint, hwnd$)
    $hdc \leftarrow$ BeginPaint($hwnd$)
    FillRect(hdc)
    $Datum \leftarrow Joint_{SpineBase}$ ▷ Set the datum of the image
    DrawLine($Joint_{Head}, Joint_{Neck}$)
    DrawLine($Joint_{Neck}, Joint_{SpineShoulder}$)
    DrawLine($Joint_{SpineShoulder}, Joint_{SpineMid}$)
    DrawLine($Joint_{SpineMid}, Joint_{SpineBase}$)
    DrawLine($Joint_{SpineShoulder}, Joint_{ShoulderRight}$)
    DrawLine($Joint_{SpineShoulder}, Joint_{ShoulderLeft}$)
    DrawLine($Joint_{SpineBase}, Joint_{HipRight}$)
    DrawLine($Joint_{SpineBase}, Joint_{HipLeft}$) ▷ Draw torso
    DrawLine($Joint_{ShoulderRight}, Joint_{ElbowRight}$)
    DrawLine($Joint_{ElbowRight}, Joint_{WristRight}$)
    DrawLine($Joint_{WristRight}, Joint_{HandRight}$)
    DrawLine($Joint_{HandRight}, Joint_{HandTipRight}$)
    DrawLine($Joint_{WristRight}, Joint_{ThumbRight}$) ▷ Draw right arm
    DrawLine($Joint_{ShoulderLeft}, Joint_{ElbowLeft}$)
    DrawLine($Joint_{ElbowLeft}, Joint_{WristLeft}$)
    DrawLine($Joint_{WristLeft}, Joint_{HandLeft}$)
    DrawLine($Joint_{HandLeft}, Joint_{HandTipLeft}$)
    DrawLine($Joint_{WristLeft}, Joint_{ThumbLeft}$) ▷ Draw left arm
    DrawLine($Joint_{HipRight}, Joint_{KneeRight}$)
    DrawLine($Joint_{KneeRight}, Joint_{AnkleRight}$)
    DrawLine($Joint_{AnkleRight}, Joint_{FootRight}$) ▷ Draw right leg
    DrawLine($Joint_{HipLeft}, Joint_{KneeLeft}$)
    DrawLine($Joint_{KneeLeft}, Joint_{AnkleLeft}$)
    DrawLine($Joint_{AnkleLeft}, Joint_{FootLeft}$) ▷ Draw left leg
    EndPaint(hdc)
end function

where $\bar{x}_i^{subject}$ and $\bar{x}_i^{model}$ are the three-dimensional positions of the i^{th} joint for the subject and model, $\theta_j^{subject}$ and θ_j^{model} are the vlues of the j^{th} joint angle for the subject and model, and w_i and ω_j are factors that allow joint positions and joint angles to be weighted differently.

The CMC uses a static optimization criterion to allocate forces to synergistic muscles and proportional-derivative control to generate a forward dynamic simulation that closely tracks the kinematics. However, the full state equations representing the activation and contraction dynamics of the muscles are incorporated into the forward dynamic simulation as well. The activation dynamics are modeled by relating the change in muscle activation and excitation with the time rate. The contraction dynamics are modeled by a lumped-parameter system that accounts for the force-length-velocity propperties of the muscle and elastic properties of a tendon.

To improve the efficiency of the system, the joint position data are recorded in a temporary file as a time sequence after being obtained by the server. The whole joint movement record is transfered to the IK module and CMC module as soon as the tracking procedure is completed, and the results are shown on the screen.

Chapter 4

Experimental Results

4.1 Experimental Setup

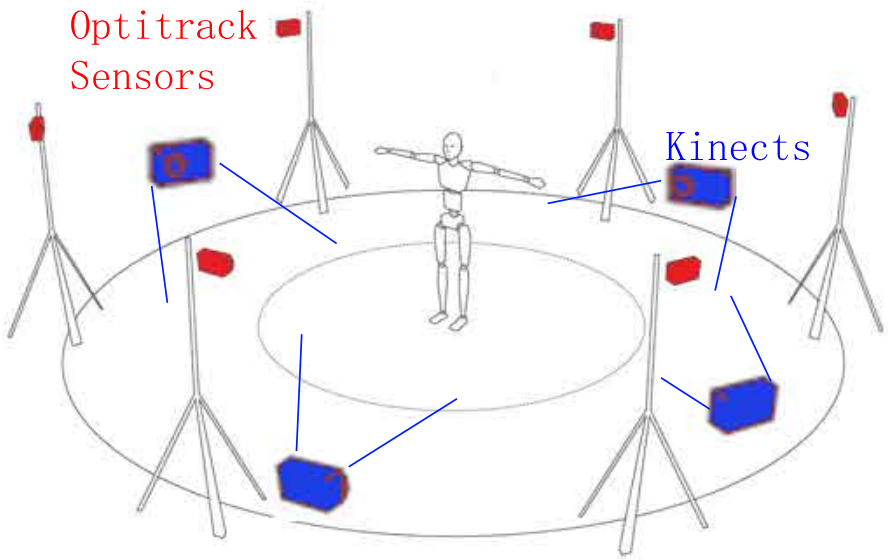
Optitrack is a marker-based optical motion capture system with multiple IR camera sensors. It is chosen as the ground-truth in the experiment because of its high accuracy (mean absolute error ¡ 1 mm) [4]. Specifically, Optitrack Flex V100:R2 [25] is the type of camera, and Arena v1.7 [14] is the corresponding software's version. A total of 6 Optitrack R2 cameras are placed around the measurement area at a radius of 2 m (the best measurement range for both Optitrack R2 and Kinect v2). All cameras are connected by a USB hub and controlled by a central computer. A total of 4 Kinects (i.e., clients) are placed around the measurement area, and a laptop computer (Processor: Intel(R) Core(TM) i7-4700MQ CPU @ 2.40 GHz, RAM: 8.00 GB) is chosen as the server. In addition, clocks from the server of the proposed system and the Optitrack are synchronized before each set of experiments to guarantee that the clock error is less than 5 ms. The experimental setup

and corresponding real-time tracking visualization are shown in Fig. 4.1 where (a) shows the layout of the Optitrack and Kinect v2 sensors. (b) is the snapshot of the experiment scene as well as the real-time tracking visualization results. A total of 16 participants with heights of 170 ± 17 cm and weights of 76.5 ± 23.5 kg participated in the following experiments. As the requirement of the Optitrack, all participants wear a pure black body suit with 4 IR markers placed at each tracked joint during the measurement.

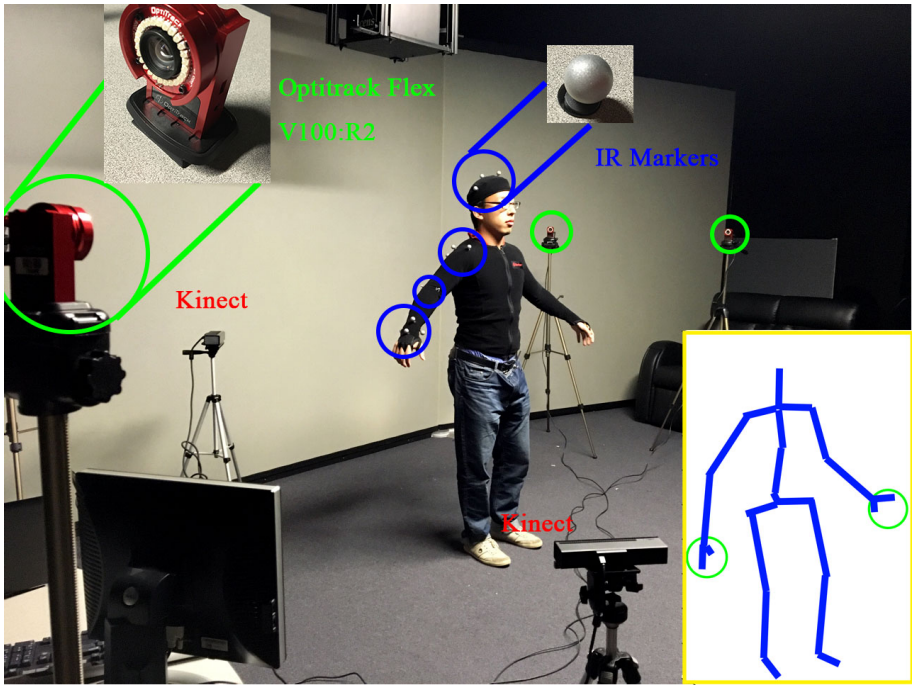
Furthermore, The Delsys Trigno^{TM} Wireless Smart Sensor System[3] is chosen as the benchmark for evaluating the accuracy of our proposed system. The Trigno^{TM} System is a type of inertial-based motion capture system, where each sensor has a built-in triaxial accelerometer, which can perceive a change in the sensor's acceleration and output it with a change in voltage.

The whole Trigno^{TM} System costs approximately $1200, and this includes the software package, the server, and 16-channel sensors. In contrast, our system only costs approximately half of that amount due to the low price of the Kinect v2 sensors.

To evaluate the accuracy of our tracking system, we place 6 Trigno^{TM} smart sensors on the 6 joints of the lower limbs, including the left hip, left knee, left ankle, right hip, right knee, and right ankle, as shown in Figure4.2. Correspondingly, these six joints are measured by our proposed system simultaneously. In the experiment, a 2.5 s gait movement is measured using both systems and compared. The tester is a 24-year-old male subject with a height of 176 cm and weight of 79 kg who moves his left and right feet forward and back one by one. The average distance between all the joints and the Kinects is 346 cm, as measured by a laser distance meter (distance measurement precision: ± 2 mm). Because the Trigno^{TM} System is based on an inertial mechanism [20] based on acceleration rather



(a)



(b)

Figure 4.1: Experimental set-up. (a) The layout of the Optitrack and Kinect v2 sensors in the experiment. (b) Snapshot of the experiment, where the green and blue circles indicate the Optitrack sensors and the IR markers that are placed on the tracked human joints. The visualization component's output is shown at the bottom-right corner, where the blue lines indicate the limbs and the green circles indicate the hand tracking.

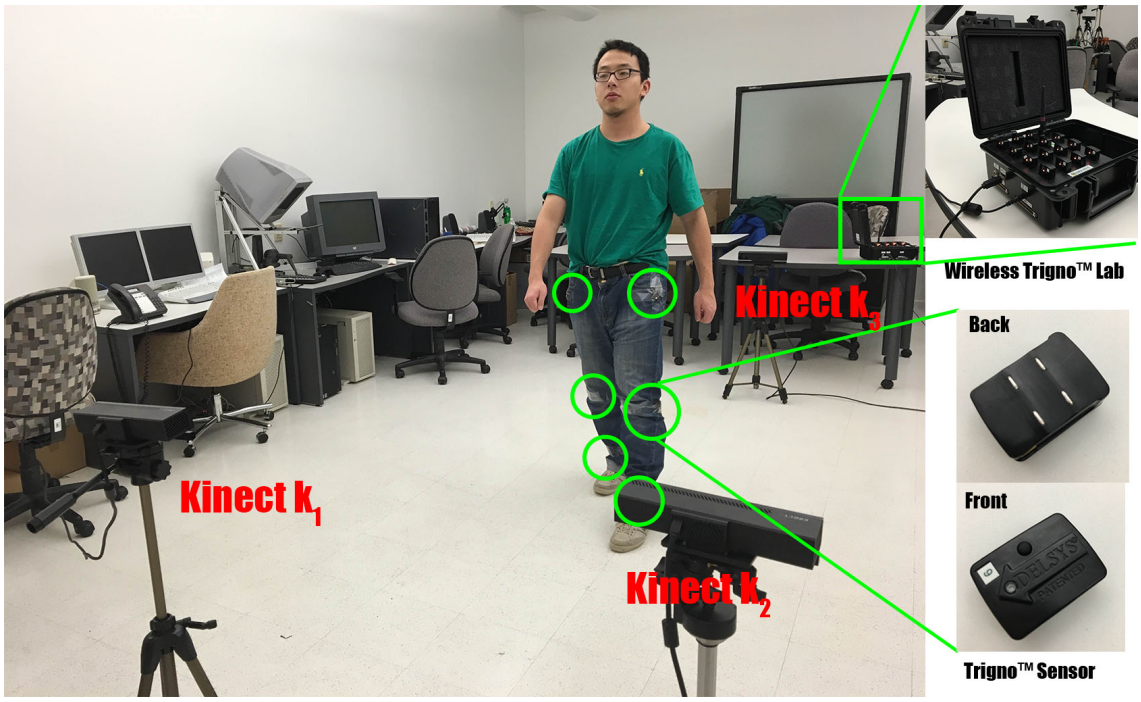


Figure 4.2: Accuracy comparison. Two motion capture systems (our proposed gait tracking system and the Delsys Trigno^{TM} Smart Sensor System) are compared to evaluate the accuracy of our system. Green circles indicate the positions of the tracked joints in human gait movement for both systems.

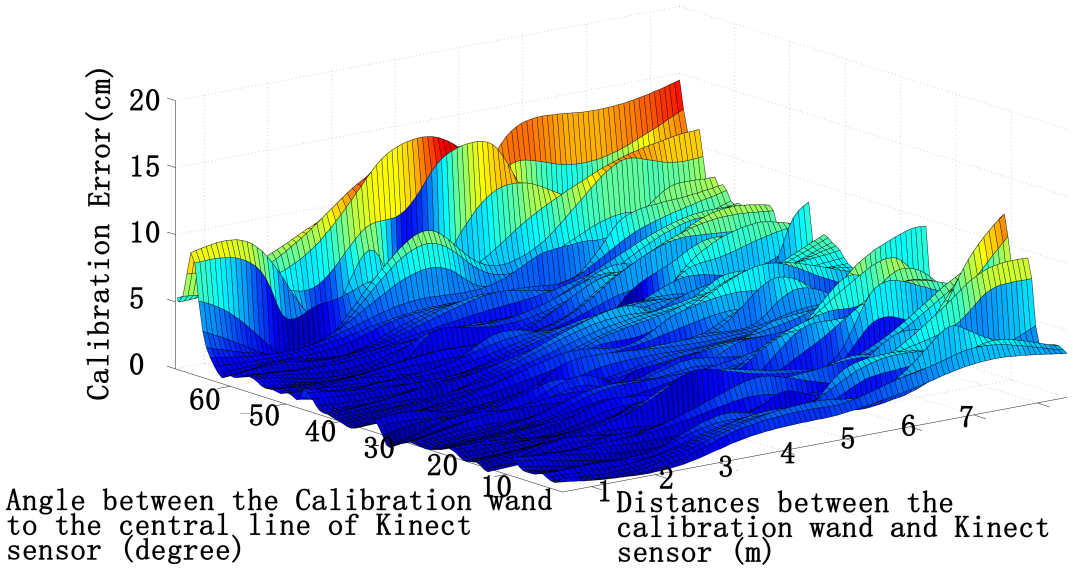


Figure 4.3: Calibration error distribution along the distance (between the calibration wand and the Kinect sensor) and the angle (between the calibration wand and the central line of the Kinect sensor).

than position data, the acceleration data have to be transformed into position data before comparison.

4.2 Calibration Evaluation

The accuracy of the calibration component is tested by comparing the measurement between the proposed system and the laser meter. The calibration results are based on different distances between the calibration wand and a sensor as well as the angle between the calibration wand and the central line of the Kinect sensor. Here, 100 data samples are collected. The relation between the calibration error and the aforementioned distance and angle is shown in Fig. 4.3.

The specifications of Kinect v2 provide its best measuring range of 0.5 m - 3 m, and

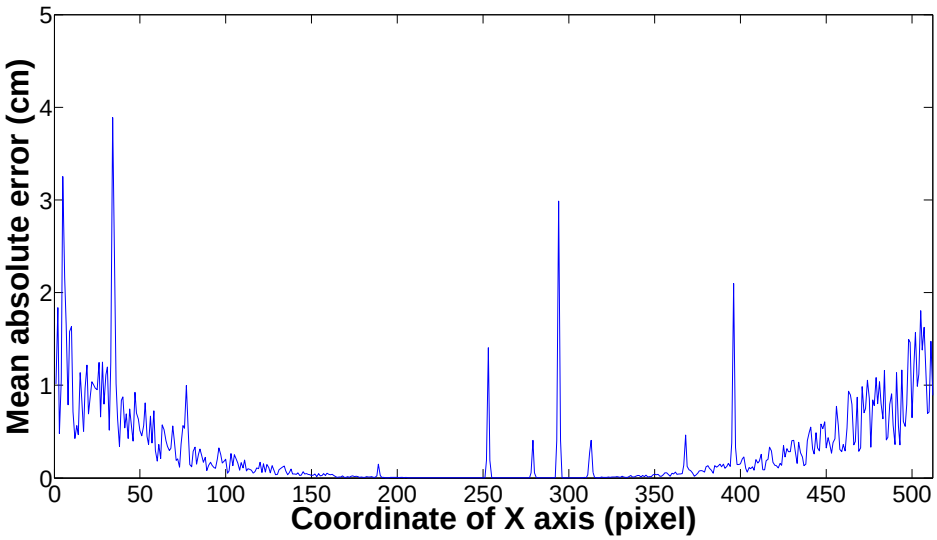


Figure 4.4: Mean absolute error of the pixels located in the middle row of depth image.

the obtained experimental results coincide with this. As shown in Fig. 4.3, the calibration error apparently increases if the calibration wand is very close to the sensor (i.e., less than 0.8 m). The sensor is unable to effectively conduct the calibration if the distance is less than 0.5 m because only 20% of the calibration results are valid. On the other hand, if the distance between the wand and sensor is greater than 3 m, the calibration error apparently increases because of the resolution limitation of the Kinect v2 sensor. To conclude, the distance between the calibration wand and sensors should be kept between 0.8 m and 3 m.

The reason that the calibration accuracy is affected by the mentioned angle is the distortion of the depth image. As shown in Fig. 4.3, the calibration error increases when the calibration wand leaves the center of the Kinect's field of view. A further experiment to test the sensor's distortion is conducted, where a flat plain is measured 2 m away from the Kinect v2, and 1000 frames of the depth image are obtained. The mean absolute errors of every pixel in the depth map are calculated, and the errors of the pixels located in the

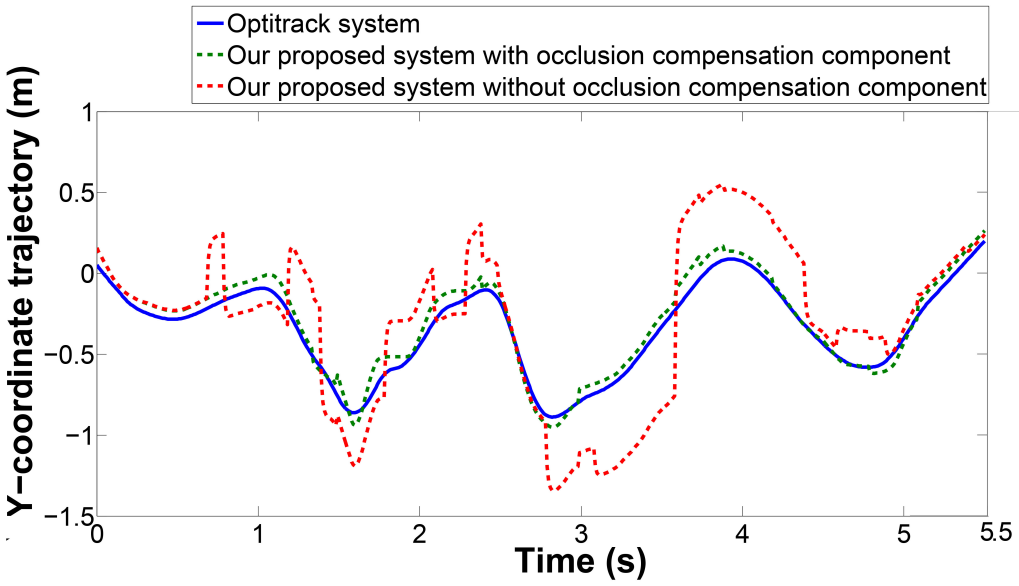
middle row of the map are extracted and shown in Fig. 4.4. It is shown that the mean absolute error on the dual edges of the depth image (pixels with x coordinates from 0 - 150 and 350-520) is much higher than that in the middle of the depth image, indicating that the recommended angle between the wand and the central line of the Kinect is less than 26°. Here, the large peaks in this figure are due to the noise in the depth image.

Based on the above analysis, within the recommended measurement range (i.e., distance of between 0.8 m and 3 m and angle of less than 26°), 50 sets of calibration samples are collected and analyzed. The mean absolute error of the calibration results is 0.73 ± 0.85 cm.

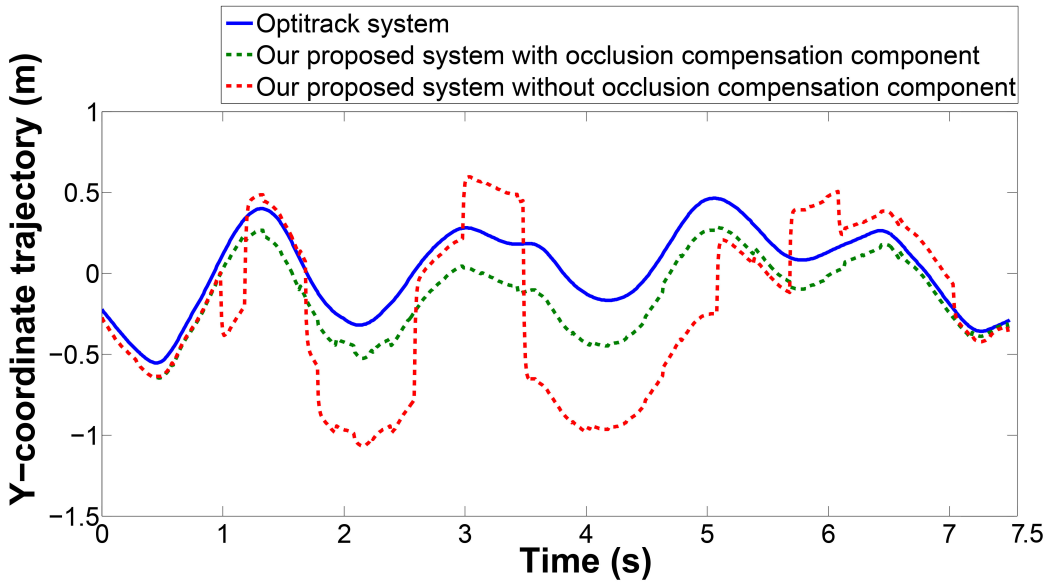
4.3 Occlusion Compensation Verification

To evaluate the effect of the occlusion compensation component on the accuracy of the occluded joints' measurement, the trajectories of joints with and without the component applied are compared together with the ground truth (Optitrack system).

In the experiment, two experiments are designed wherein 4 Kinects are set up in such a way that some of them would obtain an occluded measurement. In Experiment 1, every participant swings his/her arms for a duration of 15 s. Here, the arms are partly occluded by the body, and the shoulder, wrist, and elbow of the occluded arm are selected to be analyzed. In Experiment 2, every participant crosses his legs and performs a gait cycle for a duration of 15 s, where the hip, knee, and ankle of both legs are selected to be analyzed. The y coordinates of the tracked joints (elbow joint in Experiment 1 and ankle joint in Experiment 2) are shown in Fig. 4.5.



(a)



(b)

Figure 4.5: The y-coordinate trajectory of the joint of (a) the elbow in Experiment 1 and (b) the ankle in Experiment 2. The blue line indicates the ground-truth measured by the Optitrack system. The green and red dots show the tracking results with and without the occlusion compensation component applied, respectively.

In Fig. 4.5a, the arm swings 2 times for a duration of 6 seconds; it is occluded by the body from 2.5 s to 5.0 s in the field of view of one of the Kinects. It is shown that the result from the system without applying the occlusion compensation component is affected by the occluded measurements because the Kinect still returns its inferred result when it is not able to track the joint directly. When applying the occlusion compensation component, our proposed system can exclude the inferred result from the occluded Kinect if there is an direct tracking from other Kinect sensors. Thus, the tracking results with the occlusion compensation component (green dotted curve) are in better agreement with the Optitrack's ground-truth (blue curve) compared with those without the collusion compensation component (red dot curve) in the time period between 2.5 s and 5.0 s.

In Fig. 4.5b, the participant steps forward and backward 4 times for a duration of 7 s, where one of the legs is occluded during the middle 2 cycles from 3 s to 6.5 s. Similarly, the tracking results when applying the occlusion compensation component (green dotted curve) are better than those without the occlusion compensation component (red dotted curve). In contrast, the overall tracking results for both cases are not as good as the results in Experiment 1 because the body tracking algorithm of the Kinect makes frequent mistakes when distinguishing the body posture for closing knees.

A total of 50 sets of occluded measurement samples are collected in Experiments 1 and 2, respectively. Compared with the ground-truth, the mean absolute errors with the occlusion compensation component are 13.3 cm and 19.4 cm for the arm joints (shoulder, elbow, and wrist) and leg joints (hip, knee, and ankle) compared with that without the occlusion compensation component, which are 18.5 cm and 23.2 cm, respectively (Table 4.1).

Table 4.1: Tracking Error with and without the occlusion compensation component

Tracked Joints	Mean Absolute Error (cm)	
	With Occlusion Compensation Component	Without Occlusion Compensation Component
Arm Joints (shoulder, elbow, and wrist)	13.3	18.5
Leg Joints (hip, knee, and ankle)	19.4	23.2

Table 4.2: Comparison of Three Trilateration Methods

Method	Mean absolute error (cm)		Standard Deviation (cm)	
	3 Kinects	4 Kinects	3 Kinects	4 Kinects
Geometric method[51]	11.6	11.6	13.5	13.5
Linear least square[52]	11.6	8.2	13.5	9.7
Nonlinear least square	11.8	6.5	12.6	8.9

4.4 Trilateration Method Comparison

To compare the accuracy of different trilateration methods, practical measurements are made, and a corresponding ground truth is obtained by a laser meter. A total of 100 sets of data of a motionless object measured by a Kinect are collected to facilitate the comparison of three methods. The distances between the object and the Kinects are guaranteed to be 200 cm. The results are shown in Table 4.2.

The original geometric method picks stable equations to obtain the result, meaning that it cannot obtain increased accuracy by adding more sensors into the system. On the other hand, when the number of sensors meets the lower limit of the requirement, the linear least square method essentially becomes equivalent to the geometric methods in that they obtain the same result. The use of one extra sensor results in almost the same accuracy

compared with the advanced equation method (accuracy difference of less than 7%); in addition, it obtains a better precision, which is 27% higher.

The nonlinear least square method has a different objective function. Considering the limitation on the number of iterations, it may obtain worse results compared with other linear methods. When the threshold for stopping the iteration is changed to 1×10^{-6}, it can obtain the same result but at an intolerable time cost, which affects the real-time feature of the system. However, both the accuracy and precision are improved apparently compared with no extra sensor; the mean absolute error is reduced by 45%, and the standard deviation is reduced by 29%.

According to the results, it is shown that these methods achieve an equal accuracy and precision when the number of sensors satisfies the lower limit. However, with increased number of sensors, the least square methods obtain better results compared with equation methods. Nonlinear least squares achieves the best performance in the experiment.

4.5 Accuracy Comparison between Trilateration Systems

In this experiment, the Optitrack system and the Kinect clients run simultaneously to measure the movements of body parts. Four joints of the human upper body are selected (i.e., head, shoulders, elbows and wrists) for tracking. Each participant completes 4 cycles of gaits for a duration of 35 s. To quantitatively evaluate the overall performance of the proposed system, three trilateration systems (the linear trilateration system [52], the geometric trilateration system [51] and our proposed nonlinear trilateration system) are

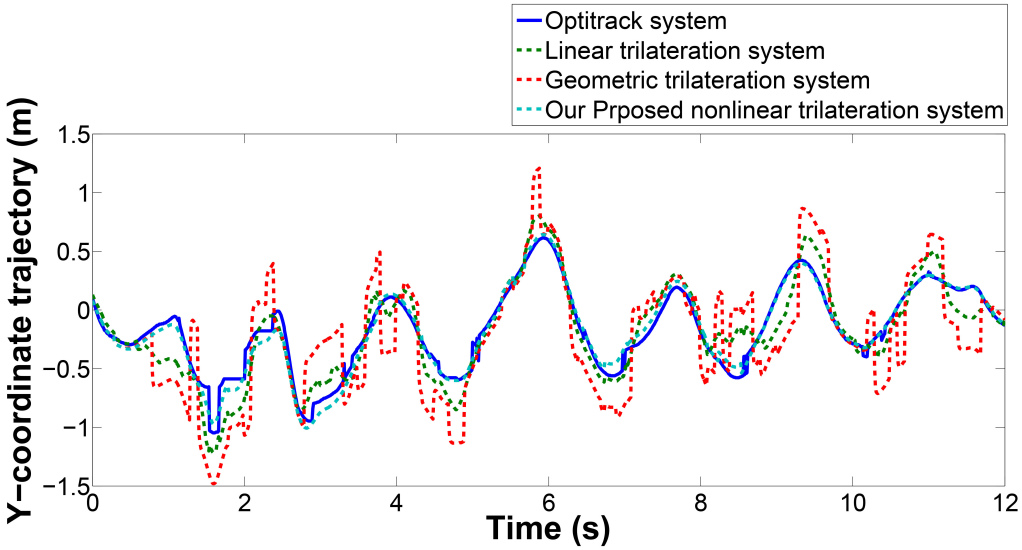


Figure 4.6: The y coordinates of the trajectories of the right wrist, where the blue, green, red, and cyan curves indicate its movement measured by the Optitrack system, the linear trilateration system, the geometric trilateration system, and our proposed nonlinear trilateration system, respectively.

applied to process the measurement data obtained from the Kinect sensors, based on which the overall accuracy and the accuracy for different body parts are analyzed (shown as follows). Here, for different coordinate settings in the trilateration systems, the analysis is based on the relative movement of joints in the same time period.

Fig. 4.6 shows the tracking results of one participant who conducts 6 gait cycles with arm swings, meaning that his right wrist makes periodic movements. Comparing the results from the above-mentioned four trilateration systems, a large deviation (maximum of 5 cm) occurs in the extreme swinging positions, whereas it is less than 1 cm in the other gait phases for all three trilateration systems. This is because the Kinect sensor has a large measurement error when the tracked joint has an abrupt velocity variance. Our proposed nonlinear trilateration system has the smallest tracking error compared with that of the other trilateration systems due to the robustness feature of its nonlinear optimization

Table 4.3: Tracking accuracy comparison between the linear trilateration system, the geometric trilateration system, and our proposed nonlinear trilateration system

Tracked Joints	Mean Absolute Error (cm)		
	Linear Trilateration System [52]	Geometric Trilateration System [51]	Our Proposed Nonlinear Trilateration System
Head	5.3	5.5	5.3
Shoulder	10.5	11.8	7.6
Elbow	8.6	13.9	7.3
Wrist	18.3	25.0	14.2
Average	10.8	14.1	8.7

solution.

By comparing the mean absolute errors of the tracked joints in the above-mentioned different motion capture systems (shown in Table 4.3), an accuracy evaluation is quantitatively provided. The maximum error occurs in the measurement of the wrists, which has the longest movement trajectory and largest velocity variance. One significant factor concerning how the accuracy of the Kinect sensor is strongly affected by velocity is the low frame rate of the Kinect's depth sensor, which is at most 30 Hz. On the other hand, the Kinect's resolution also has a limitation in measuring the end-side joints (e.g., wrists and ankles), which correspond to fewer pixels in the depth image.

Furthermore, the three trilateration systems achieve a similar accuracy for tracking the head and see large differences in accuracy for the other three joints (shoulder, elbow, and wrist). Because of the similar accuracy, the Kinect sensor is able to achieve accurate head tracking because the head's motion is relatively stable and hardly occluded. The lowest accuracy of the geometric trilateration system is due to its restriction on the number of sensors in that only the geometric trilateration system applies 3 Kinect sensors, whereas the other systems apply 4 Kinect sensors. Due to the feature of the nonlinear least square

optimization approach, our proposed nonlinear trilateration system obtains the highest accuracy and robustness [33]. Specifically, our proposed system achieves a higher average accuracy by 24.1% and 38.3% compared with the linear and geometric trilateration systems, respectively.

As shown in Figure4.7, the two systems obtain the 6 joint positions in a total time interval of 2.5 s and report the triaxial results after standardization and transformation. In the first 0.6 s, the tester moves his left foot forward and moves it back in the next 0.6 s. During this process, it is shown that the largest changes occur in the left ankle's x-axis and z-axis positions. The y-axis position of the left ankle exhibits tiny changes because the direction of the left leg's movement is perpendicular to the y-axis of the coordinate system. Compared with the other two joints (i.e., the left knee and the left hip), a larger movement is observed from the ankle joint, as compared with that from the hip and knee joints. Similarly, the movement of the lower-right limb is measured in the time interval between 1.2 s and 2.5 s, which shows the same motion characteristics as those of the left leg.

We record the measurement from the Trigno^{TM} System as the benchmark and provide the quantitative accuracy evaluation of our proposed system by computing the standard deviation of the measurement difference between the two motion capture systems (shown in Table 4.4). Based on the comparison of the different axes of the same joint, the maximum errors always occur on the x axis, along which joints make the maximum movements. On the other hand, the technical specification of the Trigno^{TM} System shows that the standard deviation of the voltage output error of the sensors is ± 0.233 V (i.e., ± 0.94 cm after transformation). Furthermore, we obtain the standard deviation of the measurement

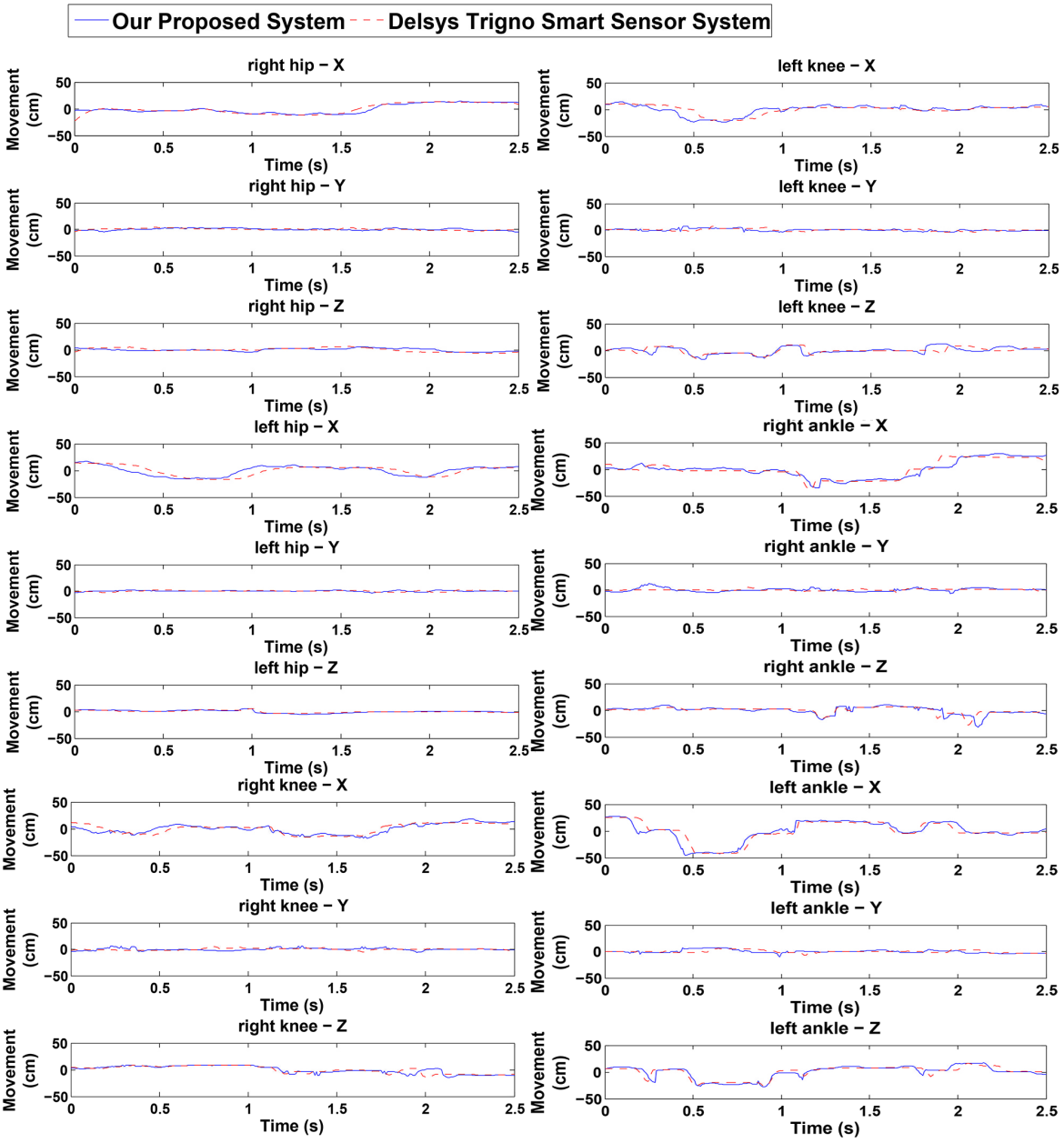


Figure 4.7: Accuracy comparison results. The measured 3D coordinates of 6 joints (left hip, left knee, left ankle, right hip, right knee, and right ankle) using both our proposed gait tracking system and the Delsys Trigno^{TM} Wireless Smart Sensor System are compared. The blue solid line shows the trilateration measurement of our proposed system; the green dashed line shows the inertia measurement of the Delsys System.

Table 4.4: Statistics of the Standard Deviation of the Difference between the Two Motion Capture Systems in Figure4.7

Figure	Standard Deviation (cm)
Right hip - X axis	5.26
Right hip - Y axis	1.28
Right hip - Z axis	1.87
Right knee - X axis	4.48
Right knee - Y axis	1.16
Right knee - Z axis	1.19
Right ankle - X axis	3.58
Right ankle - Y axis	2.33
Right ankle - Z axis	3.40
Left hip - X axis	4.12
Left hip - Y axis	1.87
Left hip - Z axis	3.89
Left knee - X axis	6.90
Left knee - Y axis	2.60
Left knee - Z axis	4.37
Left ankle - X axis	5.81
Left ankle - Y axis	2.27
Left ankle - Z axis	4.24
Total	3.98

error of our proposed system by adding the total standard deviation of the measurement difference between the two motion systems as ±4.92 cm=±(0.94 cm+3.98 cm) under the condition that the average distance from all joints to the Kinect sensors is 346 cm.

4.6 Efficiency Analysis

An experiment is conducted to evaluate the system's performance on real-time processing and clock synchronization. Here, our proposed system consists of 1 server and 4 clients, where the server and client times for sending and receiving messages and completing specific functions (such as the server's trilateration computation and occlusion compensation and the client's human joint observation) are recorded separately. Thus, the real-time processing characteristic of the proposed system can be evaluated by calculating the time cost, starting from obtaining the depth image on the client side to completing the trilateration computation on the server side. The clock synchronization can be tested by comparing the difference between the local time of the clients and the synchronized time on the server side. In total, 18000 frames (i.e., 600 s at an fps of 30) of data are recorded in the experiment. The statistical results of the synchronization results are shown in Table 4.5.

The frames excluded by the synchronization component due to the error in the network transmission (3 frames) and those whose processing time is greater than $\frac{1000}{30} \approx 33$ ms (7 frame) would affect the real-time processing feature (Table 4.5). Thus, our proposed system guarantees the real-time feature in 99.94% (17990 frames in 18000 frames) of its measurement process. It is shown that all clients dynamically synchronize their clocks with the server such that all clock errors are less than 4 ms and 99.36% of the clock errors are

Table 4.5: The statistics of the processing time cost and the clock error

Time Range (ms)	Frame Statistics Based on Processing Time Cost	Frame Statistics Based on Clock Error *
[33,+∞)	7	0
[22,33)	11926	0
[11,22)	6064	0
[4,11)	0	0
[2,4)	0	258
(0,2)	0	71740

* The clock errors are recorded from 4 clients; there are a total of $18000 \times 4 = 72000$ sets of clock error samples.

less than 2 ms.

4.7 Visualization Results

With the Inverse Kinematics and Static Optimization modules from OpenSim, the system analyzes the joint position data from the trilateration result and then produces the visualization result. There are a total of 1000 frames of user's movement recorded by the system within 30 s, which includes a total of 7 gait cycles. To reduce iterant work, we finally choose the first 4.5 s of the record that already includes a full gait cycle as an example to verify the visualization work.

As shown in Fig. 4.8, the system generates the user's musculoskeltal model with the Inverse Kinematics module and simulates the movement of the user's first three steps based on the trilateration results. The user is a 24-year-old male, and we represent the user's musculoskeletal system by a linkage actuated by 9 muscles (long head of biceps femoris, sartorius, psoas, vastus, rectus femoris, tensor fasciae latae, gracilis, gluteus and gemellus). He completes a full gait cycle, including two steps, from 0.5 s to 4.5 s, and each

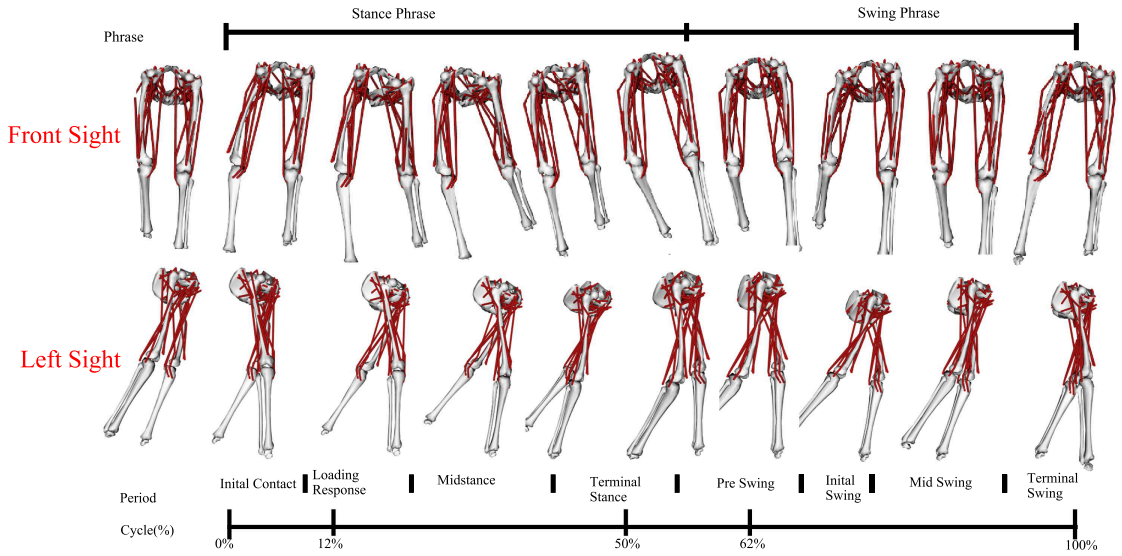


Figure 4.8: Model of user's musculoskletal system generated by OpenSim. The two figures show the nine phases of the model in the left and front sight every 0.5 s from 0 s to 4.5 s.

step constitutes a phase. The first phrase of the cycle is called the stance phase, which is shown in Fig. 4.8 from 0.5 s to 2.5 s. In this phase, the user sets his right leg as the supporting leg and moves his left leg. The second phase is from 2.5 s to 4.5 s and is called the swing phase, in which the user performs the opposite action as the stance phase by setting the left leg as the supporting leg and moving the right leg.

Fig. 4.9 shows the results of the simulation of the force produced by muscles. These nine muscles produce a force when the user lifts and stretches his leg. To perform a step, the hip and knee extensors are activated at the beginning by the force produced by the sartorius and quadriceps (Sar, Psoas, Vas_int, Glut) and then stretched by the force produced by the quadriceps and biceps femoris (Bifemlh, Rect_fem, Tfi, Grac, and Gem).

Comparing the figure of the sartorius (the first figure Bifemlh in Fig. 4.9) with Fig. 4.8, this shows that this muscle is the main cause of the knee flexion. From 1.5 s to 3.5 s,

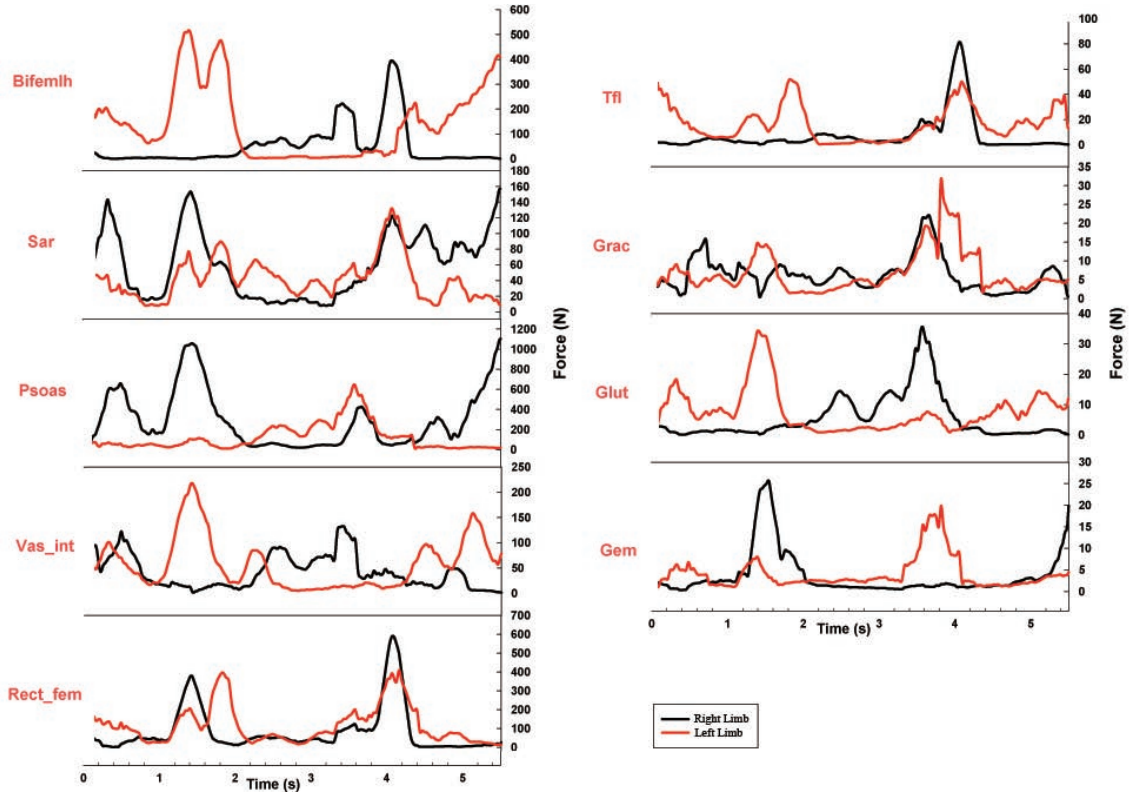


Figure 4.9: Simulation results of muscles. The nine figures from top to bottom plot the force produced by nine groups of muscles within a time range from 0.0 s to 5.5 s.

the user takes a step using the left leg and bends his left knee. The sartorius of the left leg produces at most 500 N to complete such an action. Similarly, when the user takes a step on the left leg from 3.5 s to 4.5 s, the sartorius of the right leg produces at most 400 N to complete the action.

Chapter 5

Conclusions and Future Work

5.0.1 Conclusion

In our proposed system, the calibration component registers the relative positions of sensors and transforms coordinates for further calculation. Then, the occlusion compensation component filters the inferred measurements from the Kinect v2 sensors for the trilateration component, which combines the measurements from multiple sensors to obtain the optimized human body tracking. At all times, the synchronization component coordinates between the clients and server, which enables the real-time feature of the system. The aforementioned 4 components are quantitatively evaluated experimentally, therein verifying the validity of the proposed motion capture system.

There are still some factors affecting the performance of our proposed system. For instance, the Kinect v2 sensor's accuracy suffers from a limitation related on its likelihood human body tracking algorithm's accuracy. Simultaneously, when tracking body joints with high velocity and high acceleration, the Kinect v2's depth sensor's limitation of a low

frame rate (40 Hz) increases the measurement error from the unsynchronized clock error.

5.0.2 Future Work

In this thesis, a multi-Kinect human body tracking system based on a nonlinear trilateration method is developed. We can work on several areas to further improve the accuracy and the robustness of our proposed system.

- **Embedded client:** It is complex to applicate the client's function on a computer of which computational resources are substantially greater than the requirements of the client. If the client's function can be applicated based on embedded systems, where an embedded chipset rather than personal computers are used as clients, the cost of the system and the difficulty of the set-up can be decreased.

- **Moving wand calibration:** Currently, the proposed system calibrates the sensors' positions based on the trajectory of the calibration wand's single movement. The accuracy of the calibration can be further improved by calibrating based on the trajectories of continuous and non-vertical calibration wand movements. Furthermore, the calibration procedure can be designed based on a divide-and-conquer algorithm, which can average the error based on the different measurement distances and angles.

- **Advanced occlusion detection:** The proposed system applies a computational geometric principle to detect the crossing of the human limbs. However, there are other situations that lead to occluded measurements, including the complete overlapping of images and other objects blocking a sensor's field of view in the measurement area. With advanced occlusion detection methods, the trilateration results will not

be affected by the occluded measurement, which is less accurate compared with direct tracking measurements.

References

[1] Atomic clock sync information. "www.worldtimeserver.com/atomic-clock/" Last viewed on 2015/12/08.

[2] Nettime. "www.timesynctool.com/" Last viewed on 2016/02/05.

[3] Technical Specification of Trigno^{tm} Lab. "http://www.delsys.com/products/wireless-emg/trigno-lab/" Last viewed on 2015/10/20.

[4] V100:R2 Data Sheet - OptiTrack, 2012. "http://www.optitrack.com/static/documents/V100-R2viewed on 2016/07/11.

[5] C. Albitar, C. Graebling, and C. Doignon. Robust structured light coding for 3D reconstruction. In *Proceedings of IEEE 11th International Conference on Computer Vision*, pages 1–6, 2007.

[6] H. Bacakoglu and S. Kamel, M. A three-step camera calibration method. *IEEE Transactions on Instrumentation and Measurement*, 46(5):1165–1172, 1997.

[7] W. Bailey, S. and B. Bodenheimer. A comparison of motion capture data recorded from a Vicon system and a Microsoft Kinect sensor. In *Proceedings of ACM Symposium on Applied Perception*, pages 121–121, 2012.

[8] W. Bode, H. and E. Shannon, C. A simplified derivation of linear least square smoothing and prediction theory. In *Proceedings of the IRE*, volume 38, pages 417–425. IEEE, 1950.

[9] M. Budiu, J. Shotton, G. Murray, D., and M. Finocchio. Parallelizing the training of the Kinect body parts labeling algorithm. In *Proceedings of Big Learning: Algorithms, Systems and Tools for Learning at Scale*, pages 1–6, 2011.

[10] S. Corazza, L. Muendermann, A. M. Chaudhari, T. Demattio, C. Cobelli, and T. P. Andriacchi. A markerless motion capture system to study musculoskeletal biomechanics: Visual hull and simulated annealing approach. *Annals of Biomedical Engineering,*, 34(6):1019–1029, 2006.

[11] R. Curnow and M. Lichvar. Chrony. "https://chrony.tuxfamily.org/" Last viewed on 2016/03/06.

[12] L. Delp, S., C. Anderson, Frank, S. Arnold, Allison, P. Loan, A. Habib, T. John, C., E Guendelman, and G. Thelen, D. Opensim: open-source software to create and analyze dynamic simulations of movement. *IEEE Transactions on Biomedical Engineering*, 54(11):1940–1950, 2007.

[13] T. Dutta. Evaluation of the Kinect sensor for 3-D kinematic measurement in the workplace. *Applied Ergonomics*, 43(4):645–649, 2012.

[14] D. Freeman, N. Evans, R.l Lister, A. Antley, G. Dunn, and M. Slater. Height, social comparison, and paranoia: an immersive virtual reality experimental study. *Psychiatry Research*, 218(3):348–352, 2014.

[15] E. Gill, P. and W. Murray. Algorithms for the solution of the nonlinear least-squares problem. *SIAM Journal on Numerical Analysis*, 15(5):977–992, 1978.

[16] G. Giorgi and C. Narduzzi. Modeling and simulation analysis of ptp clock servo. In *Proceedings of IEEE International Symposium on Precision Clock Synchronization for Measurement*, pages 155–161, 2007.

[17] H. Golub, G. and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970.

[18] H. Haskell, K. and J. Hanson, R. An algorithm for linear least squares problems with equality and nonnegativity constraints. *Mathematical Programming*, 21(1):98–118, 1981.

[19] C. Heinze, G. K. Germany, R., S. Spyropoulos, S. Hussmann, and C. Perwass. Automated robust metric calibration algorithm for multifocus plenoptic cameras. *IEEE Transactions on Instrumentation and Measurement*, 65(5):1197–1205, 2016.

[20] I. Hunter, K. Seeley, M., T. Hopkins, J., C. Carr, and J. Franson, J. Emg activity during positive-pressure treadmill running. *Journal of Electromyography and Kinesiology*, 24(3):348–352, 2014.

[21] G. Johansson. Visual perception of biological motion and a model for its analysis. *Attention, Perception, & Psychophysics*, 14(2):201–211, 1973.

[22] A. Jones and J. Ohlund. *Network Programming for Microsoft Windows*. Microsoft Press, 1999.

[23] K. Khoshelham and S. Elberink. Accuracy and resolution of Kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[24] T. Lakshman, U. Madhow, and B. Suter. Window-based error recovery and flow control with a slow acknowledgement channel: a study of tcp/ip performance. In *Proceddings of Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1199–1209, 1997.

[25] N. Lehrer, Y. Chen, M. Duff, L. Wolf, S., and T. Rikakis. Exploring the bases for a mixed reality stroke rehabilitation system, part ii: design of interactive feedback for upper limb rehabilitation. *Journal of Neuroengineering and Rehabilitation*, 8(1):1, 2011.

[26] Z. Li, Z Duan, G Chen, and L Huang. Consensus of multiagent systems and synchronization of complex networks: A unified viewpoint. *IEEE Transactions on Circuits and Systems*, 57(1):213–224, 2010.

[27] K. Meghani, S. and M. Asif. Localization of wsn node based on rtt toa using ultra wide band & 802.15. 4a channel. In *Proceddings of Networking, Sensing and Control*, pages 380–385, 2014.

[28] X. Mei, H. Ling, Y. Wu, E. Blasch, and L. Bai. Minimum error bounded efficient L1 tracker with occlusion detection. In *Proccedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1257–1264, 2011.

[29] D. Metcalf, C., R. Robinson, J. Malpass, A., P. Bogle, T., A. Dell, T., C. Harris, and H. Demain, S. Markerless motion capture and measurement of hand kinematics: Val-

idation and application to home-based upper limb rehabilitation. *IEEE Transactions on Biomedical Engineering*, 60(8):2184–2192, 2013.

[30] Mills and L. David. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, 1991.

[31] B. Moeslund, T. and E. Granum. A survey of computer vision-based human motion capture. *ComputerVision and Image Understanding*, 81(3):231–268, 2001.

[32] E. Motter, A., C. Zhou, and J. Kurths. Enhancing complex-network synchronization. *Europhysics Letters*, 69(3):334, 2005.

[33] S. Murphy, W. Determination of a position using approximate distances and trilateration. *Colorado School of Mines*, 2007.

[34] K. Oh. Motion-controlled video entertainment system, 1997. U.S. Patent No. 5,616,078.

[35] R. Pan, Vi.and Schreiber. An improved newton iteration for the generalized inverse of a matrix, with applications. *SIAM Journal on Scientific and Statistical Computing*, 12(5):1109–1130, 1991.

[36] M. Park, J. Park, and S. Jung. An implementation of QR code based on-line mobile augmented reality system. *Journal of Korea Multimedia Society*, 15(8):1004–1016, 2012.

[37] O. Patsadu, C. Nukoolkit, and B. Watanapa. Human gesture recognition using Kinect camera. In *Proceedings of International Joint Conference in Computer Science and Software Engineering*, pages 28–32, 2012.

[38] P. Peltola, M. Valtonen, and J. Vanhala. A portable and low-cost 3D tracking system using four-point planar square calibration. In *Proceedings of International Conference of Indoor Positioning and Indoor Navigation*, pages 1–9, 2012.

[39] A. Pfister, M. West, A., S. Bronner, and A. Noah, J. Comparative abilities of Microsoft Kinect and Vicon 3D motion capture for gait analysis. *Journal of Medical Engineering & Technology*, 38(5):274–280, 2014.

[40] G. Plouffe and A. Cretu. Static and dynamic hand gesture recognition in depth data using dynamic time warping. *IEEE Transactions on Instrumentation and Measurement*, pages 305–316, 2016.

[41] J. Preis, M. Kessel, M. Werner, and C. Linnhoff-Popien. Gait recognition with Kinect. In *Proceedings of 1st International Workshop on Kinect in Pervasive Computing*, pages 1–4, 2012.

[42] C. Scheel and O. Staadt. Mobile 3D gaze tracking calibration. In *Proceedings of 12th Conference on Computer and Robot Vision*, pages 176–183, 2015.

[43] A. Schmitz, M. Ye, R. Shapiro, and R Yang. Accuracy and repeatability of joint angles measured using a single camera markerless motion capture system. *Journal of Biomechanics*, 47(2):587–591, 2014.

[44] H. Staras and N. Honickman, S. The accuracy of vehicle location by trilateration in a dense urban environment. *IEEE Transactions on Vehicular Technology*, 21(1):38–43, 1972.

[45] K. Tai, H. Teranishi, Y. Arai, and T. Tagawa. The kinetics of hydrolytic polymerization of ε-caprolactam. ii. determination of the kinetic and thermodynamic constants by least-squares curve fitting. *Journal of Applied Polymer Science*, 25(1):77–87, 1980.

[46] G. Thelen, Darryl and Frank C Anderson. Using computed muscle control to generate forward dynamic simulations of human walking from experimental data. *Journal of Bacteriologyiomechanics*, 39(6):1107–1115, 2006.

[47] D. Thewlis, C. Bishop, N. Daniell, and G. Paul. *A Comparison of Two Commercially Available Motion Capture Systems for Gait Analysis: High-end vs Low-cost*. 2011.

[48] L-CT Wang and Chih-Cheng Chen. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 7(4):489–499, 1991.

[49] D. Webster and O. Celik. Experimental evaluation of Microsoft Kinect's accuracy and capture rate for stroke rehabilitation applications. In *Proceedings of IEEE Haptics Symposium*, pages 455–460, 2014.

[50] B. Yang, H. Dong, and A. El Saddik. Development of a self-calibrated motion capture system by nonlinear trilateration of multiple Kinects v2. *IEEE Transactions on Industrial Informatics*, Submitted.

[51] L. Yang, B. Yang, H. Dong, and A. El Saddik. 3-D markerless tracking of human gait by geometric trilateration of multiple Kinects. *IEEE Systems Journal*, 2016.

[52] L. Yang, L. Zhang, H. Dong, A. Alelaiwi, and A. El Saddik. Evaluating and improving the depth accuracy of Kinect for Windows v2. *IEEE Sensors Journal*, 15(8):4275–4285, 2015.

[53] S. Yang, R., H. Chan, Y., R. Gong, M. Nguyen, G. Strozzi, A., P. Delmas, G. Gimel'farb, and R. Ababou. Multi-Kinect scene reconstruction: Calibration and depth inconsistencies. In *Proceedings of 28th International Conference of Image and Vision Computing*, pages 47–52, 2013.

[54] Z. Yue, K. Zhou, S., and R. Chellappa. Robust two-camera tracking using homography. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages iii–1, 2004.

[55] V. Zordan, B. Majkowska, A. Chiu, and M. Fast. Dynamic response for motion capture animation. *ACM Transactions on Graphics*, 24(3):697–701, 2005.