A System for Hand Physiotherapy Using Leap Motion

Xueying LIU
Balliol College
University of Oxford
Supervised by Dr. Irina Voiculescu

A thesis submitted for the degree of
Master of Science in Computer Science

Trinity 2015

Abstract

There are many people have hand surgery around the world every year. The majority of them needs to have physiotherapy afterwards for recovery. To achieve that patients need to perform hand exercises at home. However, it is difficult to monitor their performance and guide them doing correctly. This project delivered a web-based system using Leap Motion sensor which could assist patients doing exercises at home and provide performance feedbacks. The dissertation mainly talks about the design and implementation of the system and presents algorithms to extract useful information as performance feedbacks from the collected data through Leap Motion. Besides, I used a data collection approach to find out the most suitable measurement parameters for evaluating the patients' performance and quantify the normal range of each performance measurement parameter. The system was successfully developed and achieved all initial requirements.

Acknowledgements

First and foremost, I would like to show my sincere gratitude to my project supervisor Dr. Irina Voiculescu for her support and guidance throughout this project. I also want to express my heartfelt thanks to the orthopaedic hand surgeon Associate Professor Dominic Furniss and the postdoctoral researcher Dr. Elise Pegg from Nuffield Orthopaedic Centre. They provided me a lot of information about hand recovery exercises and gave many useful suggestions about the user interface. Besides, I would like to thank all the volunteers who joined the testing and data collection process.

In close second, I would like to express my appreciation to everyone at Balliol College, especially my friends Jennifer, Bernardo, Xenatasha and my college advisor Professor Tom Melham. They gave me a warm, supportive family. I also want to thank lecturers and administrative staff at the Computer Science Department for their amazing lectures and academic assistance.

Finally, a huge thank to my parent (Mother: Aiqin Xue, Father: Shucai Liu) and my boyfriend Ke Chen for their great love and support.

Contents

1 Introduction **7**
- 1.1 Motivation . . . 7
- 1.2 Description of the work . . . 8
  - 1.2.1 Aims and objectives . . . 8
  - 1.2.2 Functional requirements . . . 8
- 1.3 Structure of the dissertation . . . 9

2 Background **10**
- 2.1 Hand recovery exercise . . . 10
  - 2.1.1 Description of required dynamic gestures . . . 10
- 2.2 Leap Motion . . . 13
  - 2.2.1 Overview of Leap Motion . . . 13
  - 2.2.2 Leap Motion API . . . 14
- 2.3 3D hand visualization API . . . 16
  - 2.3.1 WebGL . . . 16
  - 2.3.2 Three.js . . . 16
  - 2.3.3 LeapJS Plugins . . . 16

3 System design **17**
- 3.1 System diagram . . . 17
- 3.2 Database design . . . 18
- 3.3 Home Page design . . . 19
- 3.4 Exercise Assistant & Record Page design . . . 20
  - 3.4.1 Layout design . . . 20
  - 3.4.2 Interaction with Leap Motion . . . 21
  - 3.4.3 Data recording design . . . 23
- 3.5 Gesture analyzer design . . . 24
  - 3.5.1 Data preprocessing . . . 25
  - 3.5.2 Algorithm for valid information extraction . . . 29
- 3.6 Feedback Page design . . . 36

4 System implementation **38**
- 4.1 System Access & Home Page implementation . . . 39
  - 4.1.1 Register & Login implementation . . . 39
  - 4.1.2 Home Page implementation . . . 40
- 4.2 Hand Exercise Assistant implementation . . . 41
  - 4.2.1 Virtual Hand implementation . . . 42
  - 4.2.2 Real Hand implementation . . . 43
  - 4.2.3 Interaction implementation . . . 43
- 4.3 Algorithm implementation . . . 45

5 Data collection and analysis **48**
5.1 Performance measurement parameter selection 48
5.1.1 Initial assumption 48
5.1.2 Measurement parameter for Gesture 1 & 4 49
5.1.3 Measurement parameter for Gesture 2 & 5 49
5.1.4 Measurement parameter for Gesture 3 & 6 50
5.1.5 Measurement parameter selection summary 52
5.2 Data collection approach 53
5.3 Normal range of selected parameters 54
5.3.1 Normal range for Gesture 1 & 4 54
5.3.2 Normal range for Gesture 2 & 5 55
5.3.3 Normal range for Gesture 3 & 6 55

6 Test and evaluation **57**
6.1 Functionality examination 57
6.1.1 Functional test result 57
6.1.2 Special case handle test 59
6.2 Feedback accuracy evaluation 59
6.3 System limitation 61
6.4 User experience 62
6.4.1 Normal users 62
6.4.2 Doctor 62

7 Conclusion **63**
7.1 Achievement 63
7.2 Personal reflection 64
7.3 Future work 65

Bibliography **65**

A User feedback examples **68**

B Selected screen shots for data collection **70**

List of Figures

2.1 Dynamic Gesture 1 . . . 11
2.2 Bones of the human hand . . . 11
2.3 Dynamic Gesture 2 . . . 12
2.4 Dynamic Gesture 3 . . . 12
2.5 Leap Motion . . . 14
2.6 Leap Motion `Finger` object . . . 15
3.1 System diagram . . . 17
3.2 ER diagram of database . . . 18
3.3 Home Page . . . 19
3.4 Exercise Assistant & Record Page . . . 20
3.5 Animation interaction process . . . 23
3.6 Gesture analyzer . . . 24
3.7 A summary of measurement parameters discussed in Section 5.1 . . . 25
3.8 Calculate the angle between two vectors . . . 27
3.9 Data visualization before running the algorithm . . . 29
3.10 Data visualization after running the algorithm . . . 30
3.11 Structure of the input 2 (output 1) in Figure 3.6 . . . 35
3.12 One table feedback example for Gesture 2 . . . 36
3.13 One graph feedback example for Gesture 2 . . . 37
4.1 Source code structure . . . 38
4.2 Screen shots of Register & Login Page . . . 39
4.3 Screen shots of the Home Page . . . 40
4.4 Screen shots of Exercise Assistant & Record Page . . . 42
4.5 Screen shot of main codes in `virtualHand.js` . . . 42
4.6 Screen Shot of selected codes in `realHand.js` for hands interaction . . . 44
4.7 Screen Shot of the selected codes in `singleFeedback.php` . . . 46
4.8 Frame Structure . . . 47
5.1 Extra parameter for Gesture 2 & 5 . . . 50
5.2 Extra parameter for Gesture 3 & 6 . . . 51
5.3 Linear relation between minimal sphere radius and palm width (millimeters) 52
5.4 Frequency distribution of the minimal α . . . 54
5.5 Frequency distribution of the minimal θ . . . 55
5.6 Frequency distribution of minimal difference . . . 56
6.1 One example of testing different bend angles . . . 59
6.2 Graph test example . . . 60
6.3 One special graph test example . . . 61
A.1 Screen shot of Gesture 1 Feedback Page (1) . . . 68
A.2 Screen shot of Gesture 1 Feedback Page (2) . . . 69
A.3 Screen shot of Gesture 6 Feedback Page (1) . . . 69
A.4 Screen shot of Gesture 6 Feedback Page (2) . . . 69
B.1 Sample pictures for the data collection . . . 70

List of Tables

2.1 Exercises summarization (names and requirements) 13
3.1 6 steps of Figure 3.5 23
3.2 Input and output for algorithm 2 26
3.3 Position parameters triple & corresponding angle 28
5.1 Summary of the measurement parameter for each exercise 52
6.1 Functionality examination table 58

Chapter 1

Introduction

This chapter provides readers an introduction to this project. Section 1.1 illustrates the motivation of the project. Section 1.2 specifies the description of the work, including functional requirements, aims and objectives. The overall structure of this dissertation is described in Section 1.3.

1.1 Motivation

Around the world, many people have hand injury issues which may caused by accident, functional degradation, overuse and so on. According to the data from Hospital Episode Statistics (HES)[4], in England for trauma only (contains traumatic amputation, injuries of nerves, blood vessels, muscle, tendon at wrist and hand), there are more than 20,000 people had emergency hand surgery every year since 2006. Besides, the demand for elective hand surgery (non-emergency situation) increased significantly in England. The research[5] predicted that, in 2030, there will be around 170,000 surgeries, compared with 87,582 in 2011.

After hand surgery, the majority of patients need to have physiotherapy afterwards to help them maximize their hand functions and full recovery. As part of this, the patient will need to continuously do hand recovery exercises in a relatively long period. It is unrealistic to ask patients go to hospital being guided and monitored by doctors every day. Therefore, the patients should perform required recovery exercises at home. However, there are several issues for patients doing exercises at home. They maybe fail to always do the exercises correctly and repeat required number of times for each exercise. They also do not know their performance of each practice. The project aimed for solving these issues by developing a web-based system which allows patients to do hand recovery exercises at home timely and correctly, stores data on how patients are doing and provides feedbacks to them.

The Leap Motion[15] is a hand tracking sensor which could detect the hand movement of its environments. Benefit from this detecting feature, the idea of this project was to develop a system for hand physiotherapy using the sensor Leap Motion. The system was created in collaboration with clinicians from the Nuffield Orthopaedic Hospital (It is one of Oxford University Hospitals and provides treatment for bone and joint issues for over 80 years[24]). The orthopaedic hand surgeon Associate Professor Dominic Furniss based at the Nuffield Orthopaedic Centre provided a set of specific required hand exercises for the system.

1.2 Description of the work

1.2.1 Aims and objectives

The aim of the project was to create a web-based system for hand physiotherapy using Leap Motion. The system could provide users' hand visualization as a 3D hand model and aid patients doing hand recovery exercises correctly. The system is able to store users' hand movement data collected from the Leap Motion and provide feedbacks to patients using a variety of measurement parameters (different hand recovery exercises use different parameters). The values of the parameters reflect users' performance and are calculated from the collected data. Besides, the project used data collection approach (collect data from healthy hands) to quantify the normal range of the key measurement parameter for each required exercise. The normal range defines a healthy hand standard and generally describes a normal performance.

The followings are objectives:

1. Design and implement a user interface which could provide hand visualization, interact with the Leap Motion to aid patients with specific required recovery exercises (mentioned in Section 2.1) though animation and video instruction, collect hand motion tracking data (raw data) provided by Leap Motion and give performance feedbacks to patients.

2. Design and implement algorithms for the collected raw data processing, including noise detection, measurement parameters calculation and useful information extraction.

3. Although Leap Motion could track hand motion, it is not always as precise as human eyes and the data provided by Leap Motion is limited. Therefore, it is crucial to explore the solution of the following questions.

    - What are the limitations of detecting required hand motions using Leap Motion and how to overcome those issues?
    - Which parameters provided by the Leap Motion API could be used for required dynamic gestures analysis? What is the most suitable measurement parameter to evaluate the overall performance of each required exercise, aiming to provide clinically meaningful feedbacks based on users' performance?

4. Quantify the normal range of the key measurement parameter for each required exercise based on the collected data from healthy hands to define a normal (healthy) performance.

1.2.2 Functional requirements

In order to create a good system for assisting patients doing hand recovery exercises, we had a series of meetings with an orthopaedic hand surgeon Associate Professor Dominic Furniss and a postdoctoral researcher Dr. Elise Pegg from the Nuffield Orthopaedic Centre. They provided us three types of hand recovery exercises as the required ones for the system. They also specified the functions that the system was expected to achieve in order to assist patients. Besides, they showed us the whole process of how they guide patients doing the exercise correctly and evaluate patients' performances. Based on the collected information from them, functional requirements are listed below, which are guidelines of system design and implementation.

The functional requirements of this system are:

1. The system shall enable users to register and allow valid users access the system.
2. The system shall enable the user to view a list of their assigned hand exercises, including exercise information such as exercise assigned date, number of repeats for each exercise, whether finish successfully etc.
3. The system shall enable the user to access three different specific required exercises for both hands.
4. The system shall enable the user to know how to operate and start a hand exercise.
5. The system shall guide the user to complete hand exercise correctly.
   (a) The system shall provide the visualization of users' hand which detected by Leap Motion.
   (b) The system shall enable the user to follow the movement of the animation instruction to complete the exercise.
   (c) The system shall provide video instructions of how to perform each required exercise.
   (d) The system shall enable the user to have preparation time before starting an exercise.
   (e) The system shall give users a clear signal (such as word count, color signal, word instructions and progress tracking) to help them move to next step smoothly.
6. The system shall store users' hand movement data collected from Leap Motion.
7. The system shall analyze the collected data and show users the performance feedbacks of each completed exercise.

1.3 Structure of the dissertation

Chapter 2 explains the background information of the project, which helps readers have a better understanding of the rest chapter. The background includes the description of three types of required hand exercises, a brief introduction of Leap Motion sensor, and several APIs for 3D hand visualization. They are all frequently mentioned in the dissertation.
Chapter 3 presents the design of the system based on functional requirements mentioned in Section 1.2.2, which includes the design of system structure, user interface, the interaction with Leap Motion, data storage and several algorithms I designed for analyzing the raw data and generating performance feedbacks for users.
Chapter 4 talks about how I implement the system based on the design.
Chapter 5 provides the analysis and discussion of the measurement parameters I chose to evaluate the overall performance of each required exercise. Besides, the chapter reports the normal range of key measurement parameters I got from the data collection approach.
Chapter 6 shows the test and evaluation of the whole system.
Chapter 7 concludes the achievements of the project, personal reflection and suggestions of future work.

Chapter 2

Background

This chapter talks about the background information of the project, including the description of all hand recovery exercises which the system could assist patients to do, the introduction of Leap Motion sensor and useful APIs for achieving 3D hand visualization.

2.1 Hand recovery exercise

The system is particularly designed for patients who have injured fingers. After a hand surgery, patients feel very painful and even could not bend their joints as healthy people. In order to recover their hand functions, they need to repeatedly do hand recovery exercises in a long time. The orthopaedic hand surgeon Associate Professor Dominic Furniss provided us three types of recovery exercises which are frequently used after hand surgery for maximizing hand functions in the real world. The system mainly focuses on these three specific hand exercises.

2.1.1 Description of required dynamic gestures

In this project, the hand recovery exercises are called dynamic gestures. Since the three specific exercises apply to both hands, the system supports 6 dynamic gestures for right and left hand in total. These gestures are frequently mentioned in the dissertation. To make the description more clear, this section explains the detail information of each gesture and labels each gesture with a unique name (Gesture 1, 2, 3, 4, 5, 6).

Each gesture has two stages shown in Figure 2.1, Figure 2.3 and Figure 2.4. One is starting from the initial position to the goal position. The other one is starting from the goal position back to initial place. Each gesture only focuses on the performance of 4 fingers including index finger, middle finger, ring finger and little finger (exclude thumb). Three types of exercises are explained one by one below with an exercise summarization table at the end of this section.

Exercise No.1: Gesture 1 & 4

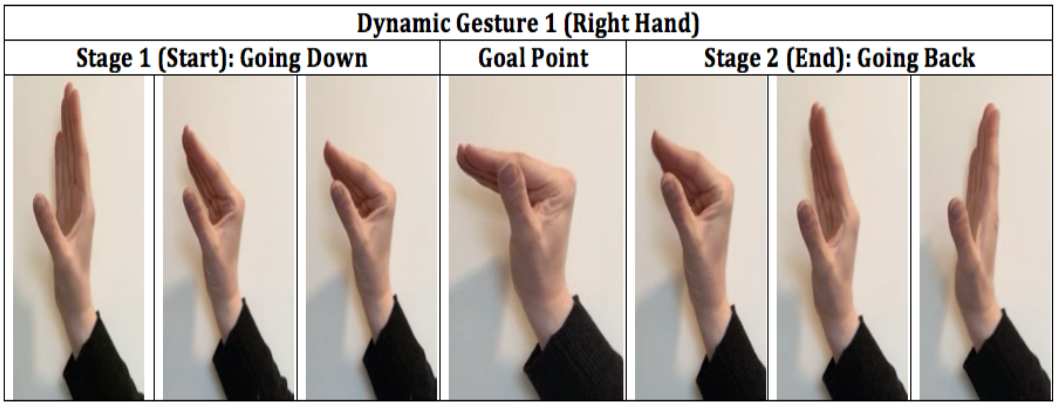
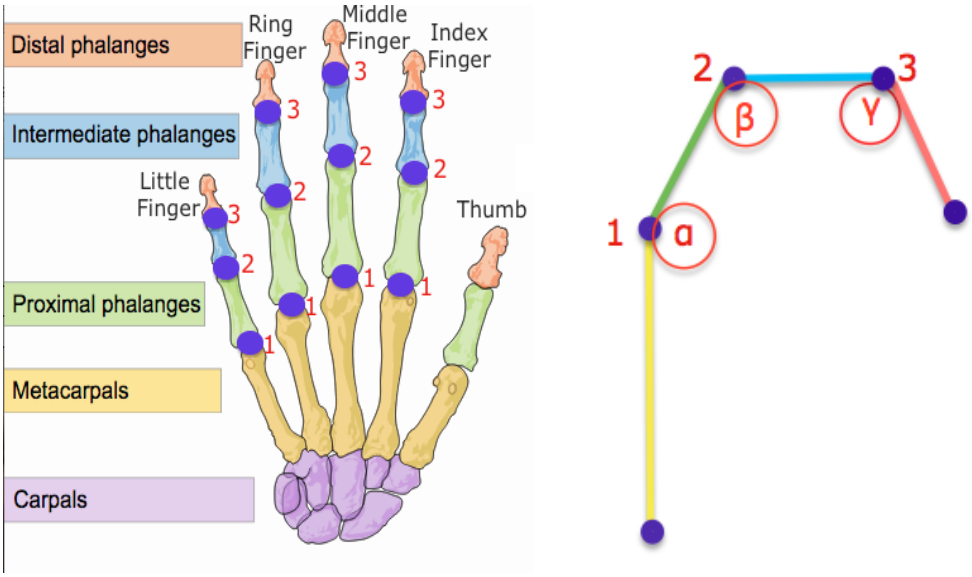


Figure 2.1: Dynamic Gesture 1

Figure 2.1 shows the first exercise for the right hand, which called **Gesture 1**. The first exercise for left hand follow the exact same motion as Figure 2.1 which called **Gesture 4**. The key point for this exercise is to practice the joints between proximal phalanges and metacarpals shown in Figure 2.2(a), which are all the joints labeled by 1. Figure 2.2(b) shows the side view of human finger with bone structure. The perfect goal position is the middle image of Figure 2.1. It requires people bend the joint to make the angle α in Figure 2.2(b) reaching almost 90 degrees (at the initial position, all fingers keep straight, the α is around 180 degrees). In clinical diagnosis, the key evaluation factor for Gesture 1 & 4 is the minimal angle α that the patient could achieve at the goal position for each finger (except thumb). The system is expected to show users the minimal value of α that each finger achieves at their goal position which allow users to compare with the perfect gesture requirement mentioned above and know how far to reach the goal.



(a) Front view[23] (b) Side view

Figure 2.2: Bones of the human hand

Exercise No.2: Gesture 2 & 5

Figure 2.3 shows the second exercise for the right hand, which called **Gesture 2**. The second exercise for left hand follow the exact same motion as Figure 2.3 which called **Gesture 5**.

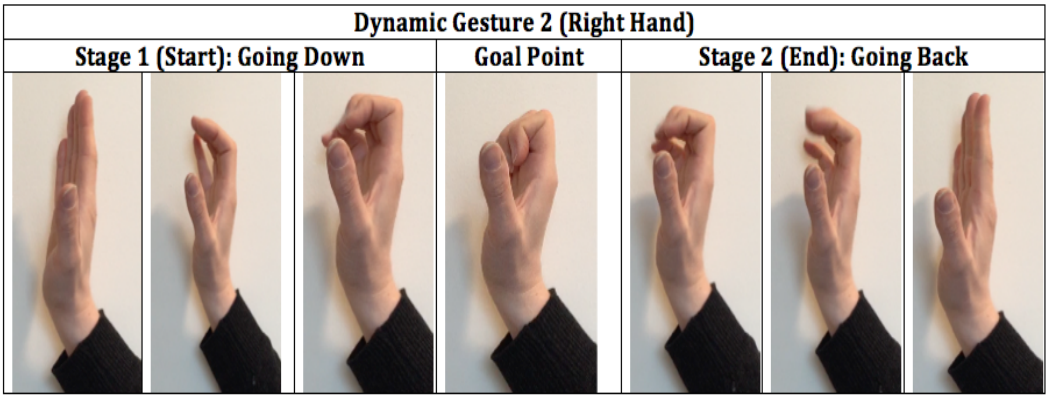


Figure 2.3: Dynamic Gesture 2

The key point for this exercise is to practice the joints between distal phalanges and proximal phalanges shown in Figure 2.2(a), which are all the joints labeled by 2 and 3. The perfect goal position is the middle image of Figure 2.3. It requires people bend joints to make the angle β in Figure 2.2(b) below 90 degrees and angle γ in Figure 2.2(b) around 110 degrees (at the initial position, all fingers keep straight, the β and γ are all around 180 degrees). In clinical diagnosis, the key evaluation factors for Gesture 2 & 5 are the minimal angle β and γ that the patient could achieve at the goal position for each finger (except thumb). The system is expected to show users the minimal value of β and γ that each finger achieves at their goal position.

Exercise No.3: Gesture 3 & 6

Figure 2.4 shows the third exercise for the right hand, which called **Gesture 3**. The third exercise for left hand follow the exact same motion as Figure 2.4 which called **Gesture 6**. This exercise needs patients to make a fist and let the fingers touch the palm.

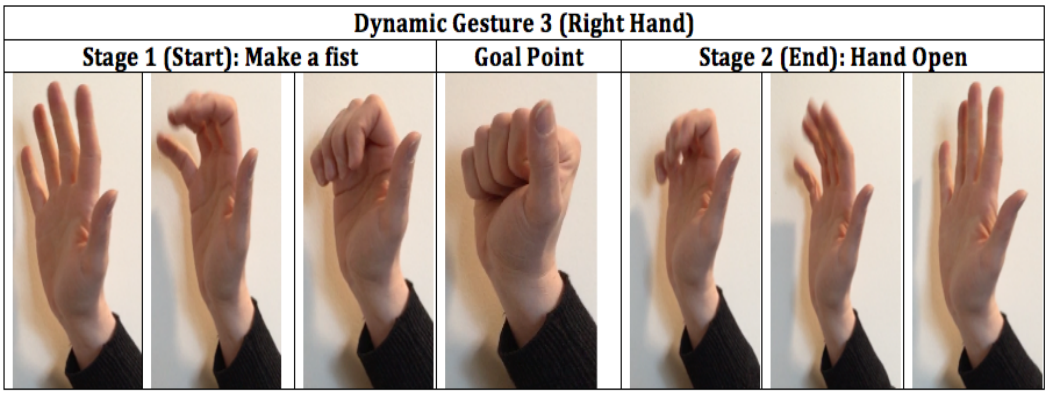


Figure 2.4: Dynamic Gesture 3

The key point for this exercise is to practice the joints between distal phalanges and metacarpals shown in Figure 2.2(a), which are all the joints labeled by 1, 2 and 3. The perfect goal position is the middle image of Figure 2.4. It requires people bend joints to make the angle α in Figure 2.2(b) around 90 degrees, β around 90 degrees and γ around 110 degrees (at the initial position, all fingers keep straight, the α, β and γ are all around 180 degrees). In clinical diagnosis, the key evaluation factors for Gesture 3 & 6 are the minimal angle α, β and γ that the patient could achieve at the goal position for each finger (except thumb). The system is expected to show users the minimal value of α, β and γ that each finger achieves at their goal position.

Exercises summary

We labeled all exercises and their clinical meaningful information (each finger joint bend angles α, β and γ shown in Figure 2.2(b)) required by doctors. The names shown in Table 2.1 will be used directly in the later part of the dissertation. In table 2.1, the Requirements column means the recovery goal that patients should achieve for each gesture meets a perfect standard (healthy standard). The bend angles α, β and γ are expected to show to patients as part of the feedback.

Exercise ID	Right Hand Exercise Name	Left Hand Exercise Name	Requirements
No. 1	Gesture 1	Gesture 4	α almost 90°
No. 2	Gesture 2	Gesture 5	$\beta < 90°$ γ around 110°
No. 3	Gesture 3	Gesture 6	α & β around 90° γ around 110°

Table 2.1: Exercises summarization (names and requirements)

2.2 Leap Motion

This section talks about the overview of Leap Motion sensor including why I choose Leap Motion, how Leap Motion works and its features. Besides, I introduce the Leap Motion API in section 2.2.2.

2.2.1 Overview of Leap Motion

The Leap Motion controller[18] is a tiny USB peripheral device released in the middle of 2013. It used two cameras, which acts like human vision (each person has two eyes) and three infrared LEDs to track hand motions [15]. The device could use 2D images collected from the cameras to reconstruct 3D data which shows what the sensor detects and sees. It is very easy to use. People only need to plug the Controller into the computer and download the software to use this device. Figure 2.5(a)[22] is an example.

There are some similar motion controller devices one example is Kinect[8] which was developed by Microsoft. The reason to choose Leap Motion is that Kinect is more suitable for the whole body tracking. However, Leap motion is good for hand tracking. It provides hands gestures to fingerbone-level tracking and allow people access hundreds of data points provided by its APIs [14]. Since the project aims for Hand Physiotherapy which needs to access the data of hands and fingers. Leap Motion is the best choice. Besides, Leap Motion is precise, accurate and sensitive which achieved by tracking ten fingers up to 1/100th of a millimeter and people's movements at a rate of over 200 fps (frames per second)[15].

(a) Get started with the Leap Motion controller [22] (b) Hand sphere[13]

Figure 2.5: Leap Motion

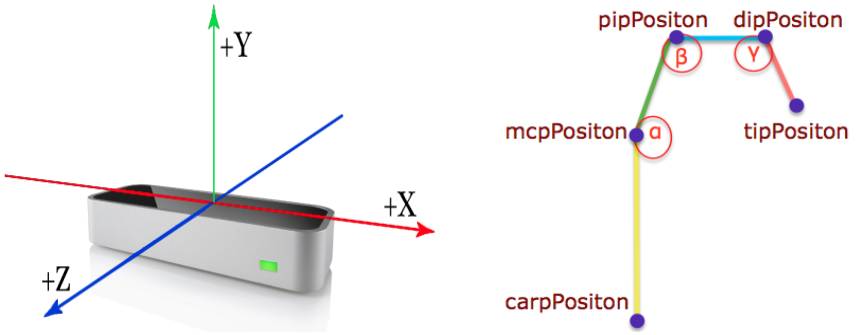
2.2.2 Leap Motion API

The hand physiotherapy web-based system collects patients' hand movement data through the Leap Motion JavaScript API. The collected data records the movements of how patients perform required dynamic gestures over the sensor. The recorded data of a completed dynamic gesture could be considered as a combination of multiple snapshots stored in chronological order. We could say the set of data reflects the whole hand movements. The Leap Motion controller is able to track hand movements of its environment as a set of snapshots called `frames` in Leap Motion API [12]. Each individual frame represents the hand information of an unit detecting moment and it stores various of tracking objects including `Hand`, `Arm`, `Finger` and so on. The Leap Motion is able to update the frame information with the frame rate 60 fps [12]. When the patient moves the hand over the sensor, the sensor keeps capturing the frame information, and the system stores all captured frame data in one set as the recording of how the patient is doing.

The Table 2.1 shows the joints bend angle information expected to be provided to patients. Besides, it is crucial for patients to know how long it takes to complete one exercise, because the time could indirectly reflect the smoothness of the whole process, usually it takes longer time for patients to complete one exercise if they feel pain. Time is an important measurement component. Based on what I need to provide to patients, for this project, I chose to use the following tracking objects, `Finger`, `Frame` and `Hand` included inside each individual frame.

From Finger Object:

Leap motion provides a `Finger` object, which allows the system access the information of five fingers. The bend angles could be calculated from the position of adjacent joints. The following five parameters are the joint positions provided by `Finger` Object. Figure 2.6(b) shows the corresponding joint positions for each parameter. The five parameters mentioned below are all position vectors in three-dimensional based on the Leap Motion coordinate system shown in Figure 2.6(a). The Leap Motion system uses a right-handed Cartesian coordinate system[10] and the coordinate origin is located at the top of the device.



(a) Leap Motion coordinate system[10] (b) Parameters visualization

Figure 2.6: Leap Motion `Finger` object

(1): `carpPosition`**[11]**
Position Vector which describes the base end of the metacarpals position.

(2): `mcpPosition`**[11]**
Position Vector which describes the position of the joint between the metacarpals and proximal phalanges.

(3): `pipPosition`**[11]**
Position Vector which describes the position of the joint between the proximal phalanges and intermediate phalanges.

(4): `dipPosition`**[11]**
Position Vector which describes the base of the distal phalanges position.

(5): `tipPosition`**[17]**
Position Vector which describes the finger tip position.

From Frame Object:

Each `Frame` contains one parameter called `timestamp` which used for calculating how long it takes to complete one exercise.

(1): `timestamp`**[12]**
Record the time captured by the frame in microseconds from the start of Leap Motion.

From Hand Object:

Leap Motion has `Hand` object. Inside `Hand` object, there are two parameters (Section 5.1.4 explains why we choose them) also used for the system.

(1): `grabStrength`**[13]**
It represents the degree of hand open or close with range 0 to 1. The value will be 0 if we open our hands and keep all fingers straight. The value will be 1, if Leap Motion detected that the user doing a grab pose.

(2): `sphereRadius`**[13]**
Leap Motion provides a sphere model which fitting the curve of the users' hand. The sphere is located as a ball holding by the hand (Figure 2.5(b)[13] is an example.). This parameter is the radius of the sphere.

2.3 3D hand visualization API

The system could provide user hand visualization as a 3D hand model. In order to achieve this, the following related APIs which will be directly mentioned in later chapter are introduced in this section.

2.3.1 WebGL

`WebGL`[6] (Web Graphics Library) is a JavaScript API which is designed for rendering interactive 3D computer graphics in any compatible web browser such as most versions of Chrome, Safari 8 and 9, IE 11 and so on. `WebGL`[6] uses the HTML element called `Canvas`. The `Canvas` could be considered as a container which allows `WebGL` render into.

2.3.2 Three.js

`Three.js`[25] is a JavaScript API used to create and display animated 3D computer graphics on a Web browser using `WebGL`. It could be considered as an easy way to use `WebGL`. The main library Rigged Hand (Section 2.3.3) I used to provide 3D hand visualization in the system is achieved with `Three.js`.

2.3.3 LeapJS Plugins

LeapJS Plugins[16] are a set of independent plugins which add extra functions of the Leap Motion JavaScript API. Based on the requirements of the project, I only used the following two plugins.

(1): Rigged Hand[16]
It adds 3D Hands which detected by the Leap Motion into web pages using `Three.js`. It is used for achieving 3D hand visualization.

(2): Playback[16]
It could play a prerecorded file with particular `JSON` format, which is used for developing the animation instruction assisting patients doing hand exercises. It also provides extra functions such as pause, stop and repeat playback when the recording is ended.

Chapter 3

System design

This chapter talks about the design of the Hand Physiotherapy system. Section 3.1 shows the system diagram including the main parts of the system and how they link to each other. Section 3.2 describes the database design. The system user interface design covers three main parts which are the Home Page design on Section 3.3, the Exercise Assistant & Record Page design on Section 3.4 and the Feedback Page design on Section 3.6. Additionally, several algorithms I designed for analyzing collected data in order to provide performance feedbacks to users are discussed in Section 3.5.

3.1 System diagram

The Hand Physiotherapy system is a web-based system. Web-based system allows users easily access through a web browser, which is very convenient for users to do hand recovery exercises and check their recovery progress using the Internet at home, working place or anywhere they want. Besides, the Leap Motion JavaScript API (Section 2.2.2) and LeapJS Plugins (Section 2.3.3) for web application provide tools which are exactly what the Hand Physiotherapy system needs. Considering the user requirements and system development, I chose to design a web-based system. The whole project addressed only the functionality issue which aimed to achieve all functional requirements discussed in Section 1.2.2 (Other issues such as security issue will not be discussed and they are mentioned as future work in Section 7.3).

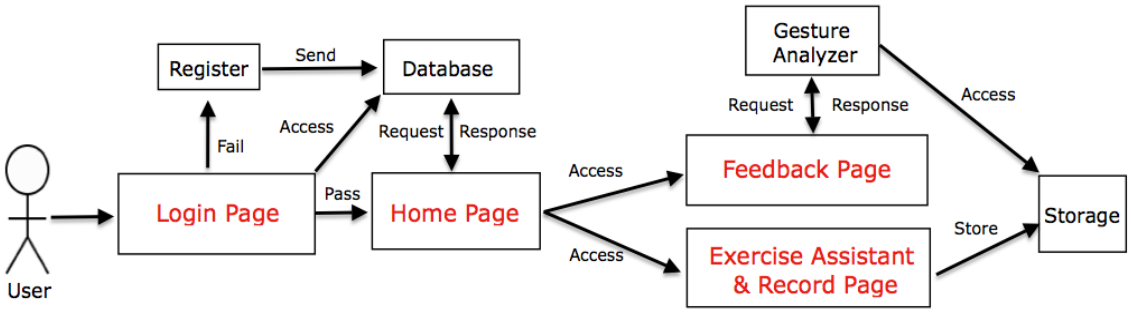


Figure 3.1: System diagram

Figure 3.1 is the system diagram, which covers all main components of the system and shows their relations through arrows. Based on the functionality, the system could be divided into 4 main web pages which are colored in red on Figure 3.1.

By following arrows step by step, we could know how patients interact with the system. Firstly, users (patients) will access a Login Page which they need to do identity checking by searching the database. For a new user, register is required. An existing user could move to the Home Page successfully. The Home Page was designed for two main purposes, viewing all assigned hand exercises for the users from the database and navigating to another two main web pages of the system. One is the Exercise Assistant and Record Page which allows the user to do hand exercises under the instructions and also record the user's performance into the Storage. The other one is the Feedback Page, which enables users to check the feedback of each completed exercise. All feedbacks are generated by the Gesture Analyzer. If the user needs the feedback of a particular completed exercise, the Gesture Analyzer could access the Storage, find out the corresponding collected performance data and run algorithms in order to get the feedback results used for the Feedback Page.

3.2 Database design

The aim of the database is to store patients' information and their corresponding assigned exercises. Figure 3.2 is the ER diagram of the database. There are mainly two entities. One is called `patient_info` which used for recording the basic information of users. The attributes of this entity including a unique patient ID `pid` (primary key), patient real name (`name`), `username` and `password` used in the Login Page, `age`, `gender` and `note` (some extra information worth to be recorded).

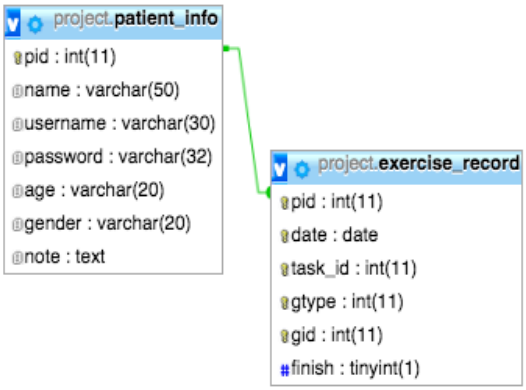


Figure 3.2: ER diagram of database

The other one is called `exercise_record` which records all assigned exercises for all registered users. Each record in this table represents one exercise for one patient. The record includes the following components. The first one is `pid`, which tells us this record is belongs to which patient. The second one is the date of this exercise called `date`. There is a set of exercises for users, which includes all gestures (Section 2.1.1) and `gtype` means the record for which required gesture (could be 1 to 6). Patients need to do a set of exercises multiple times. Each time could be called one task and `task_id` represents the task number of this record. For each gesture, users should repeat doing several times and `gid` means the record for which repeat. Besides, there is one attribute called `finish` which represents whether this exercise is completed or not and 0 means not finished yet and 1 means this exercise finished successfully. The relation of two entities is one-to-many, one patient could have multiples exercise records. However, one exercise record only belongs to one patient.

3.3 Home Page design

Figure 3.3 was created by using Balsamiq Mockups[1]. It shows the design of the Home Page. The Home Page should allow users to view all assigned exercises, provide links to access another system functionalities and show guide information to help them using the system.

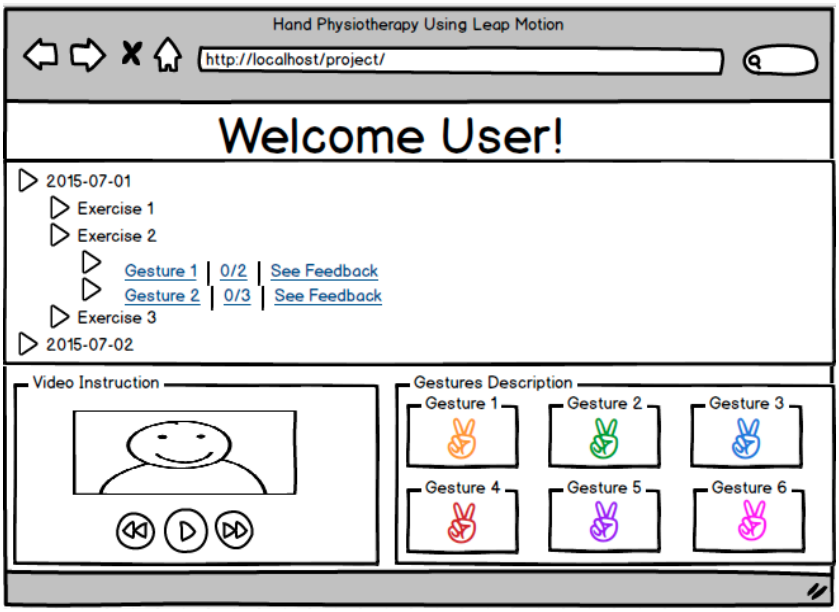


Figure 3.3: Home Page

At the top of this page, all assigned exercises for this patient are listed and ordered by date. Since the exercise records for one patient could be many. In order to save space and help users to search the exercise quicker, I decided to show exercise records by using the list which is able to fold and unfold. In Figure 3.3, there is one triangle before each list element. Users could click the triangle to unfold the detail information below that element. For example, in Figure 3.3, comparing with the exercise lists for 2015-07-01 and 2015-07-02, all exercises for 2015-07-01 are unfolded. At the beginning, most exercise records are folded for saving space.

The Home Page allows users to easily navigate to other functional web pages. For example, there are several required gestures (Section 2.1.1) under Exercise 2 in Figure 3.3. If the user clicks on the gesture name such as Gesture 1, the system will jump to the interface introduce in Section 3.4 which mainly used for helping the user do exercises. There is a small progress bar next to the gesture name such as 0/3, which shows how many times that the user has already completed and how many times the user required to repeat for this gesture. For a completed exercise, the user could click on the See Feedback, which could link to the Feedback Page (Section 3.6) of this exercise.

This page also provides the video instruction about how to use this system such as how to access each function, how to start one exercise etc. Besides, each gesture is named using numbers such as Gesture 1, Gesture 2. The user may be confused about matching the name with the actual hand motion. Therefore, the page also provides a quick description of each gesture using the animated `GIF` image (transformed from corresponding hand motion video).

3.4 Exercise Assistant & Record Page design

The exercise Assistant & Record Page allows users to access the key function of the system which assists the patient to do exercises and records the patient's performance using Leap Motion. This section discusses the design of this part, including the layout of this page on section 3.4.1, the interaction with the Leap Motion to complete one dynamic gesture on section 3.4.2 and how to record the performance on section 3.4.3.

3.4.1 Layout design

Figure 3.4 (created by using Balsamiq Mockups[1]) shows the design of exercise page which could help the patient do a particular required dynamic gesture. There are four areas labeled with number in Figure 3.4.

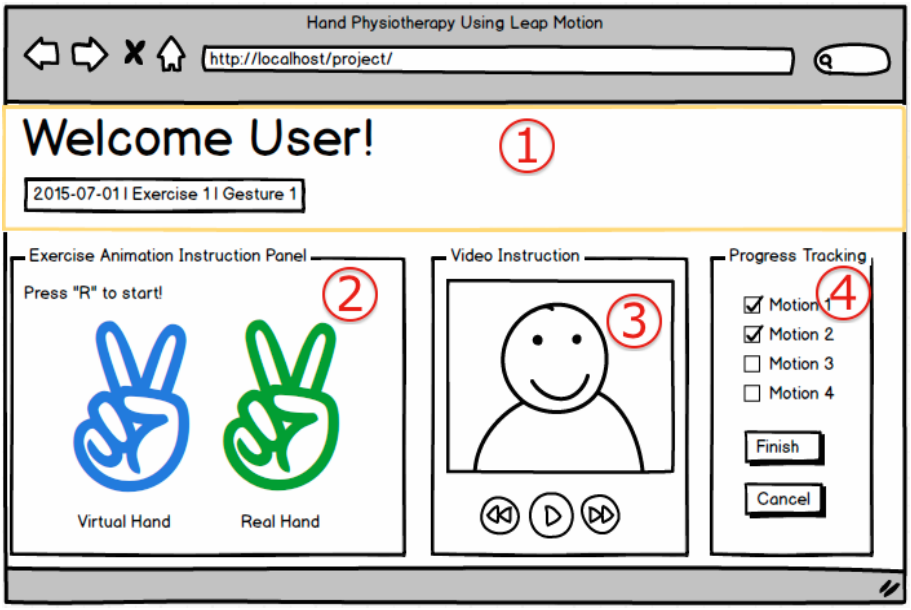


Figure 3.4: Exercise Assistant & Record Page

➤**Area 1:** Shows current gesture information (date, task).
➤**Area 2:** Provides the animation instruction for the user to follow. This area is the main interaction between the user and the system to complete one gesture. In the left corner of this area, there are word instructions to guide users what they should do next. For example, press keyboard R to start the exercise, provide a countdown to allow users to prepare before the motion start and tell them whether the exercise is finished. In the center of this area the user is able to see two 3D hands, which are called Virtual Hand (the Blue one) and Real Hand (the green one). This area contains the major interaction of Leap Motion, the detail design is explained in Section 3.4.2.
➤**Area 3:** Provides a video instruction (one orthopaedic hand surgeon helped us record the video) of how to perform this gesture.
➤**Area 4:** Provides progress tracking. Each gesture needs users to repeat several times. This area lists all repeats and the corresponding check box will be selected when users finished that repeat successfully.

3.4.2 Interaction with Leap Motion

This section talks about the detail design of the Area 2 in Figure 3.4, which contains the main interaction with the Leap Motion sensor. The design of this part could be divided into four components, which are the design of Virtual Hand, Real Hand, hand exercise process and how to link the Virtual Hand, Real Hand to complete one exercise.

In the real world, a good way to tell people how to do a hand motion correctly is to perform for them personally step by step, and allow people to follow. The design of this Area 2 used the same idea. We used two hands in this area called Virtual Hand and Real Hand.

▼Virtual Hand:
The blue one in Area 2 is Virtual Hand which performs the perfect hand motion of this gesture and guides users when to stop or move in the whole exercise process. The users are expected to completely follow what the Virtual Hand does. The perfect hand motion is prerecorded using a Leap Recorder[19] which is a playback tool. I used it to record and edit perfect hand motions for each required gestures (Section 2.1.1) and then downloaded the recorded snippets into `JSON` files. I used the Playback (Section 2.3.3) to play and control (such as pause, stop etc) the prerecorded hand motions. When users access a particular gesture, the system will load and play the corresponding prerecorded data as the animation instruction and show to the users.

▼Real Hand:
The green one in Area 2 is Real Hand. When users doing exercises, it is crucial that they could visualize the detected movement that they performed over the Leap Motion sensor on the screen. This is also very convenient for them to compare their performances and the perfect Virtual Hand movement. Therefore, I used the Rigged Hand (Section 2.3.3) to achieve the user's hand 3D visualization on the screen, called Real Hand.

Besides, for all required dynamic gestures in the project, it is important to see the joints bend angles that users could achieve. Therefore, I set up the position of the camera along the negative direction of the x-axis and make the camera look at the coordinate origin in order to provide a side view of the hand, which allows users to view the bend angles they could achieve (Virtual Hand used the same API and camera settings for hand visualization).

The interface shown in Figure 3.4 is particularly for doing one gesture. It may need the user to repeatedly perform the gesture several times. All required repeats should be continually completed. There is a little preparation time between each repeat. Additionally, from section 2.1.1, we know that, each gesture has two stages (start position to a goal position then back to initial position). There is also a little preparation time for users come back from the goal position to the initial place. The users may confuse about when they are at a preparation stage, when they should move their hands. In order to solve this, except for following the Virtual Hand, I also added color changing in the Real Hand design. The Real Hand has two colors. When the Real Hand is in purple, which means users are in a preparation stage, they should not move. When the Real Hand turn green, users should follow the Virtual Hand movement and do the same motion. The color change is a good single to help users quickly response and follow Virtual Hand motion.

▼Exercise process design:
A complete exercise contains several repeats of a dynamic gesture. Before each repeat, users should have time to prepare. I decide to use count down with 3, 2, 1 to give users enough time to get ready. Inside each repeat, users need to hold the pose for 2 second in their goal position (the middle image of Figure 2.1, 2.3, 2.4). I decided to use countdown with Ready, Go to allow users holding the pose.

▼Interaction signal design:
In order to link the Real Hand and Virtual Hand to achieve a complete exercise, users need to tell the system when they are ready to start, when they reach the goal position they could achieve and when they finish the current repeat. There are several ways could allow users to send the signal to the system. At the beginning, we tried to use keyboard control (press one key to send the signal). However, based on the feedbacks from doctors, it was not a convenient way (For example: patients may forget to press the key in time, or they have to press too many times if the gesture needs to repeat multiple times). Therefore, I decided to use a static pose as a signal. (Reason: Leap Motion is a very sensitive sensor. The velocities of the finger in dynamic and static stage are easily detected. Furthermore, this matching users' habit, when we finish and ready for the next step, usually we stop moving).

▼Signal:
The signal we used in the system could be defined that users could continue holding their hands and keep hand static for 0.5 second.

To detect users' hands are in a static state, I used one parameter provided by Leap Motion API called `tipVelocity`[17] which records the tip moving speed in millimeters per second. There is no absolutely static pose in real life, therefore even though we think our hands are not moving there is still a speed which could be detected by Leap Motion. Based on long time observation, when the `tipVelocity` of one finger below 10 millimeters per second (Leap Motion is very sensitive, when our fingers slightly move a little bit, the value of `tipVelocity` could be much larger than 10 and this value works really well in practice), we could say that the finger is not moving at that moment.

Since the system mainly focus on the movement of four fingers, therefore, I added the `tipVelocity` of four fingers except thumb, if the sum value less than 40 millimeters per second. We could say the user's hand is stable at that moment. To defined a stable status, the system monitors the sum of `tipVelocity` for continues 0.5 second. If during this time period, the sum value could continuously satisfy the requirement, we could say this is a valid signal for a static pose.

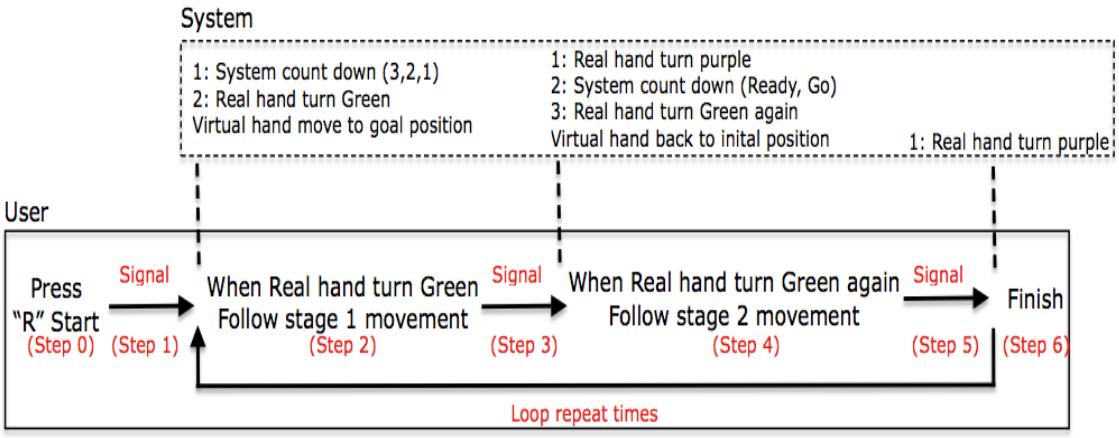


Figure 3.5: Animation interaction process

Figure 3.5 shows how the user and the system interaction in this animation instruction panel to complete one exercise. The solid rectangle describes users' action and the dotted rectangle shows how the system responses. Figure 3.5 could be explained by using Table 3.1. All the keywords (Real Hand, Virtual Hand, Signal) I used in Figure 3.5 and Table 3.1 were explained at the beginning of this section.

Step 0:	Users press keyboard R to start the system.
Step 1:	Users send a signal to notice the system they are ready. The system receives the signal, starts counting down from 3 to 1, changes the Real Hand to green and activate the Virtual Hand moving to the goal position then pause.
Step 2:	Users find out Real Hand color turns to green. They follow the movement of Virtual Hand reaching goal position.
Step 3:	Users send a signal to notice the system they are ready to go back to the initial position. The system receives the signal and turns the Real Hand into purple, then starts counting down (Ready Go), finally turns the Real Hand into green again and makes the Virtual Hand moving back to the initial position and stop.
Step 4:	Users find out the Real Hand color turns to green again. They follow the movement of the Virtual Hand and back to the initial position.
Step 5:	Users send a signal to notice the system they finished the current repeat. The system receives the signal and turns the Real Hand into purple, then checks whether users finish all repeats, if not Go back to Step 1, if yes, Go to Step 6.
Step 6:	Users finish this exercise successfully.

Table 3.1: 6 steps of Figure 3.5

The user's performance is recorded and evaluated only in Step 2 and Step 4, when the Real Hand are in green.

3.4.3 Data recording design

The system is able to store how the patient is doing and we use a data format called JSON[27] (It is mainly used for sending data between the server and the web-based system) to store the collected data. This section discusses the design of data recording by answering two questions. One is what we should collect, the second one is when we should start or stop collecting the data.

Section 2.2.2 mentioned that Leap Motion could continuously (60 fps) update the `frame` information when the user puts the hand over the Leap Motion sensor. Each `frame` contains various of hand tracking information. However, we only collected the tracking information discussed in Section 2.2.2 such as `carpPosition`, `mcpPosition`, `timestamp`, `sphereRadius` etc. Because, those tracking information covers all required gestures needs for the further analysis. Besides, the collected data will be stored into one file, we want to minimize the size of the file, therefore, we only stored the necessary information for each gesture. When the user starts performing required gestures, each single moment is stored in one `frame`. The collected result will be a set of `frames` provided by the Leap Motion.

However, we do not need to store all detected `frames`. Based on the design, the system uses hand color change from purple to green to tell users they should start moving and following the instruction. Therefore, users do the exercises only when their hands are green. We only need to store the `frames` information when users' hands are green. In other words, the system will only collect data in Step 2 and Step 4 of Table 3.1.

Each registered user has its own folder on the server to store all completed exercises data. When users click the Finish button on Figure 3.4 Area 4, the system will check whether the user successfully finished all repeats (a complete exercise means users are required to continuously perform all repeats). If not, the system will notice the user 'not finish'. If yes, the collected data will be stored into the folder. The users could also click the Cancel button to restart the exercise.

3.5 Gesture analyzer design

The aim of the Gesture Analyzer in Figure 3.6 is generating the performance result of a completed exercise by running algorithms on the corresponding raw data (collected from the Leap Motion, discussed in Section 3.4.3). Since the system evaluates the overall performance of each dynamic gesture based on the value of the performance measurement parameters discussed in Section 5.1, it is crucial to transform the raw data into the useful measurement parameters' value we need. This section explains the methods I used and the algorithms I designed for achieving this goal. Figure 3.6 shows the two main steps of how the Gesture Analyzer transforms the raw data into what we want to show to the users. Section 3.5.1 and 3.5.2 discuss the two steps respectively.

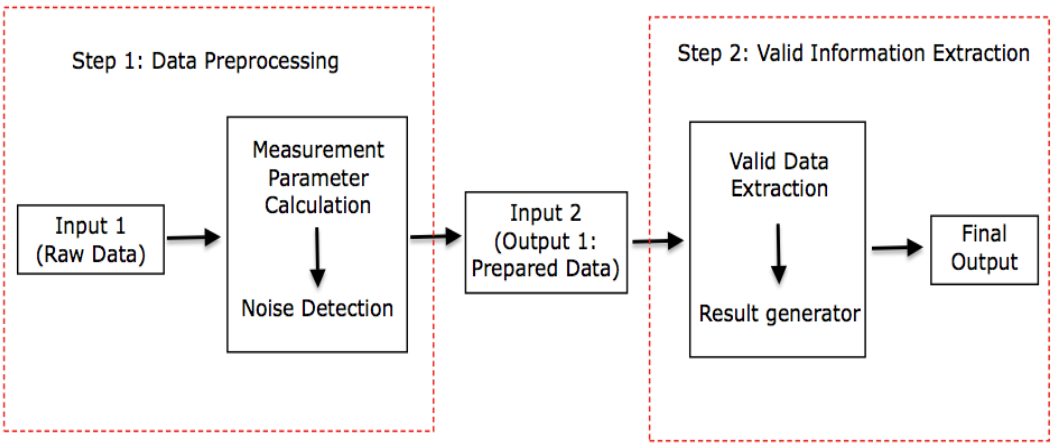


Figure 3.6: Gesture analyzer

3.5.1 Data preprocessing

The section talks about the Step 1 in Figure 3.6 which includes three components, the input, preprocessing algorithms and output.

Input 1:

The input of Step 1 is the raw data collected from Leap Motion (mentioned in Section 3.4.3), which is a set of `frames`. Each `frame` contains hand tracking information introduced in Section 2.2.2 such as `pipPosition`, `mcpPosition`, `timestamp` etc.

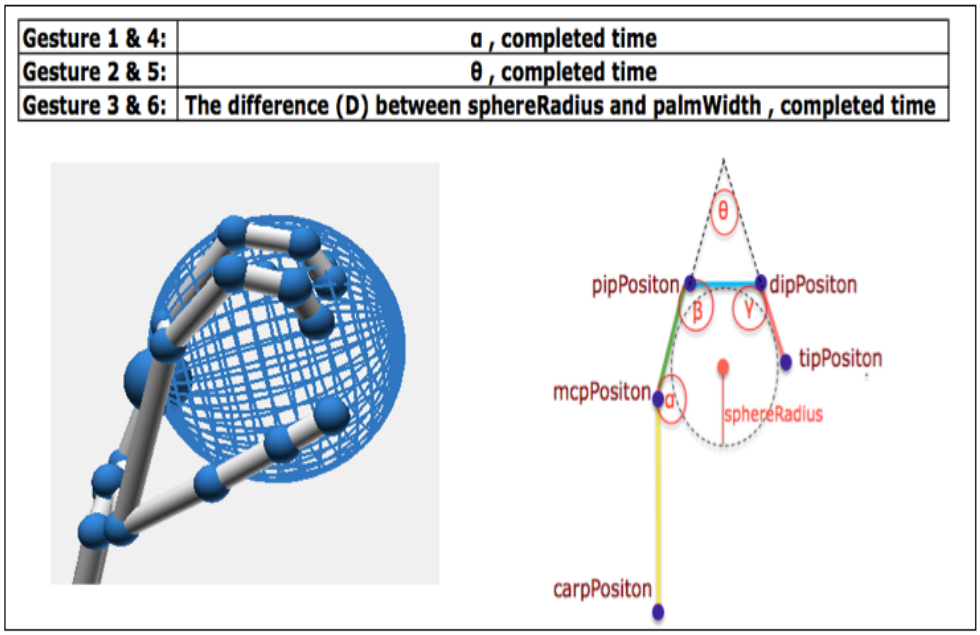


Figure 3.7: A summary of measurement parameters discussed in Section 5.1

The system evaluates the users' overall performance by using the measurement parameters shown in Figure 3.7. Figure 3.7 is a summary of Section 5.1. The top of the Figure 3.7 shows what we should provide to users to show their performance of each gesture, including how long it takes to complete one gesture, the value of α, θ and the difference(D) between `sphereRadius` and `palmWidth`. The bottom of Figure 3.7 uses finger side view to visually explain the meaning of each measurement parameter. Figure 3.7 shows the final output that all algorithms (mentioned in Section 3.5) work together and expect to get (It aimed for helping readers quickly understand the aim of all algorithms). The detail information of each measurement parameter and why I choose them to evaluate users' performance are discussed in Section 5.1.

Preprocessing algorithm:

The raw data does not directly contain this information in Figure 3.7. Therefore, The aim of data preprocessing is to transform the raw data into a set of information we want to analyze which records how the value of the measurement parameter for each gesture changed at each time point. In general, there are two steps for the data preprocessing algorithm shown in Figure 3.6, which includes the calculation of measurement parameters and noise detection. The noise data is very rare. In this project, the noise data means that at one time point, a particular measurement parameter's value such as the angle α

or θ does not make sense. For example, If we find that the angle of α less than 90 (human finger could not bend to that angle), we will remove this frame data and define it as noise.

Output 1:

The output of the Step 1 stored in an associative array (a set of (key, value) pairs). The key is the time and the value is the corresponding measurement parameter's value at that time point. The whole output represents how the value of the measurement parameter varies along the time line. Figure 3.11 shows the structure of Output 1.

Algorithm 1 is the data preprocessing algorithm for Gesture 3 and 6 (Exercise No.3) shown below.

Algorithm 1 Data Preprocessing Algorithm for Gesture 3,6

```
1:  procedure DATA PREPROCESSING1(R)
2:  //Input R: raw data, a list of frames
3:  //Output A: an associative array. Key: time, Value: D (difference) in Figure 3.7
4:  //Step 1: Get data from input set R (a set of frames)
5:      for all frames fr in R do
6:          time ← fr.timeStamp
7:          radius ← fr.sphereRadius
8:          width ← fr.palmWidth
9:  //Step 2: calculate D and store into output array A
10:         A[time] = radius - width
11:     Return A
```

The data preprocessing algorithm for Gesture 1, 2, 4, 5 (Exercise No.1 and No.2) is called Algorithm 2. It is a slightly different with data preprocessing algorithm for the Gesture 3, 6 (Exercise No.3). Because, the measurement parameters for Gesture 1, 2, 4, 5 focus on joints bend angles α and θ (each finger has its own α and θ, the gesture needs us to consider 4 fingers) in Figure 3.7. Algorithm 2 has to use a helper algorithm called Algorithm 3 to achieve angles calculation. Table 3.2 explains the input and output for Algorithm 2.

Input:	**R:** a list of raw data **g:** the row data belongs to which gesture result ($g \in \{1, 2, 4, 5\}$)
Output:	**A:** a set of associative array, Key of each associative array is time For Gesture 1, 4: Value of each associative array is α in Figure 3.7 For Gesture 2, 5: Value of each associative array is θ in Figure 3.7 **A** contains 4 elements for index, middle, ring and little finger respectively.

Table 3.2: Input and output for algorithm 2

Algorithm 2 Data Preprocessing Algorithm for Gesture 1,2,4,5

```
1: procedure DATA PREPROCESSING2(R,g)
2: //Step 1: Create 4 associative array to store 4 fingers preprocessing result
3:     index = {}                          ▷ the associative array store index finger data
4:     middle = {}                         ▷ the associative array store middle finger data
5:     ring = {}                           ▷ the associative array store ring finger data
6:     little = {}                         ▷ the associative array store index little data
7: //Step 2: Get data from input set R (a set of frames)
8:     for all frames fr in R do
9:         time ← fr.timeStamp
10:        fingersSet ← fr.fingers         ▷ fingersSet store all joints position for 4 fingers
11: //Step 3: do the calculation for 4 fingers one by one
12:        for all fingers fi in fingersSet do
13:            //currentFinger ∈ {index, middle, ring, little}
14:            currentFinger ← fi.getCorrespondingResultArray
15:            angle ← fi.getKeyAngle(g,fi)   ▷ getKeyAngle(g,fi) explain in Algorithm 3
16: //Step 4: Check not noise data
17:            if angle ≠ -1 then
18: //Step 5: Store into associative array (key: time, value: required angle)
19:                currentFinger[time] = angle
20: //Step 6: Store each finger result to the output set A
21:    A = { index, middle, ring, little}
22:    Return A
```

The bend angle calculation:

Inside Algorithm 2, there is one method called `getKeyAngle` which takes gesture ID and a set of finger joints positions data as input and the output is the angle we want. This method is mainly used for calculating the measurement parameters shown in Figure 3.7, α for Gesture 1 and 4, θ for Gesture 2 and 5. The detail of `getKeyAngle` method shown in Algorithm 3. The approach I used to do the angle calculation is called dot product of two vectors and it is the key idea behind the main function of Algorithm 3 called `vectorAngle`.

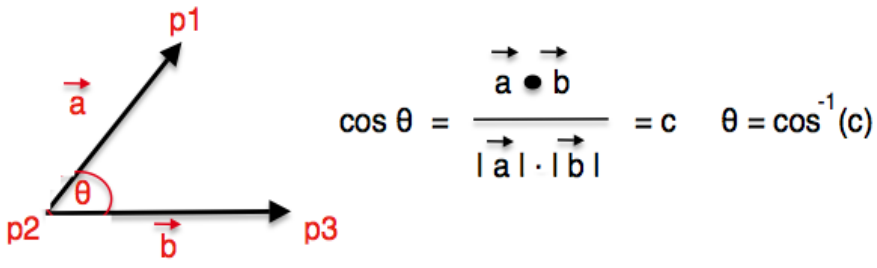


Figure 3.8: Calculate the angle between two vectors

The function `vectorAngle` takes three joints positions. Assume they are p1, p2 and p3 shown in Figure 3.8. $\vec{a}$ and $\vec{b}$ could be calculated. If p1 = (x1, y1, z1), and p2 = (x2, y2, z2), then $\vec{a}$ = p1 -p2 = (x1 - x2, y1 -y2, z1 -z2). The dot product of two vectors $\vec{a}$ and $\vec{b}$ is $\vec{a} \bullet \vec{b} = |\vec{a}||\vec{b}| \cos\theta$ [3]. The $\cos\theta$ could be calculated by using the dot product of two vectors divided by the magnitudes of the two vectors [21] shown in Figure 3.8. Then the inverse of cosine function could be used to get the value of θ.

$$\text{If } \vec{a} = (a_x, a_y, a_z),\ \vec{b} = (b_x, b_y, b_z) \text{ then } \vec{a} \bullet \vec{b} = a_x b_x + a_y b_y + a_z b_z$$

$$|\vec{a}| = \sqrt{(a_x)^2 + (a_y)^2 + (a_z)^2},\ |\vec{b}| = \sqrt{(b_x)^2 + (b_y)^2 + (b_z)^2}$$

Therefore, three joints positions are enough for calculating the bend angle. That is how `vectorAngle` function works in Algorithm 3. Table 3.3 is a reference table of which Leap Motion parameters (three joints position introduced in section 2.2.2) could be used for calculating required bend angles.

Angle we want	Position parameters
α	carPosition, mcpPosition, pipPosition
β	mcpPosition, pipPosition , dipPosition
γ	pipPosition , dipPosition, tipPosition
θ	β, γ

Table 3.3: Position parameters triple & corresponding angle

Algorithm 3 is shown below. It is a helper function for Algorithm 2 used on Line 15.

Algorithm 3 Required angle calculation Algorithm

```
1: procedure getKeyAngle(g,fi)
2: // Input: g (Gesture ID), fi: a set of joints positions for one finger (need for angle calculation)
3: // Output: The required angle (could be α or θ, or -1 which means noise data)
4: //Step 1: check gesture ID
5: // For gesture 1 and 4
6:     if g == 1 || g == 4 then
7:         pip ← fi.pipPosition                          ▷ get needed joints positions
8:         mcp ← fi.mcpPosition
9:         carp ← fi.carpPosition
10: //Step 2: calculate angle α
11:        α ← vectorAngle(pip,mcp, carp)                ▷ α in Table 3.3
12: //Step 3: check noise, noise return -1
13:        if α < 90 then
14:            Return -1
15:        else
16:            Return α
17: // For gesture 2 and 5
18:    else if g == 2 || g == 5 then
19:        tip ← fi.tipPosition                          ▷ get needed joints positions
20:        dip ← fi.dipPosition
21:        pip ← fi.pipPosition
22:        mcp ← fi.mcpPosition
23: //Step 2: calculate angle θ
24:        β ← vectorAngle(dip,pip,mcp)                  ▷ β, γ, θ in Table 3.3
25:        γ ← vectorAngle(tip,dip, pip)
26:        θ ← 180 - (360 - (β + γ))                     ▷ θ in Section 5.1.3
27: //Step 3: check noise, noise return -1
28:        if θ < 0 then
29:            Return -1
30:        else
31:            Return θ
```

3.5.2 Algorithm for valid information extraction

This section discusses the Step 2 in Figure 3.6. From Figure 3.6, we know that the input (input structure shown in Figure 3.11) of this step is also the output of Section 3.5.1. We get an associated array of the time and the value of the measurement parameter at that time point on last Section 3.5.1. Figure 3.9 is the visualization of the output associated array. For this line chart, the horizontal direction (x axis) is labeled by time, and the vertical direction (y axis) is labeled by the gesture measurement parameter, which are α, θ, and D in Figure 3.7. The blue line curve in Figure 3.9, clearly represents how the value of the measurement parameter changed along the time line when the user performed the gesture.

Figure 3.9 shows the change of α which starts around 170 degrees and then the user bend the joint, the angle starts reducing and reaching to a minimal value around 120 degrees. That is the limitation of the user could achieve at the goal position. The minimal value stabilized for a while, because the system keeps recording the data until it detects that the user's hand is not moving for a continuous period around half second.

Then the system starts counting down and gives the user fixed 2 seconds to prepare. After that, the user moving their hands from the goal position to the initial position. The value of α back to 170 degrees again. Finally, the system keeps recording until the system detects that the user stops moving. It is clearly shown from the curve that, the data we have so far contains some extra information and it could not describe the user real hand motion accurately. The extra information is called invalid data in this project. The system aims for removing all invalid data and getting a correct feedback result for the user.

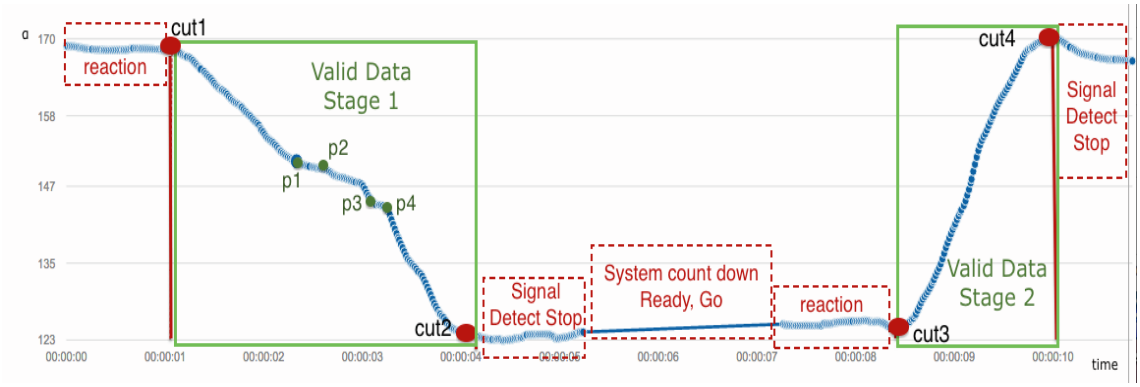


Figure 3.9: Data visualization before running the algorithm

What is the invalid data?

Invalid data in this project is the data which should not be counted as part of the user's real gesture performance. There are three types of invalid data shown in Figure 3.9. One complete required gesture could be considered as two stages which are separated by system count down.

The first invalid data called 'reaction' data, which happened at the beginning of each stage. The system starts collecting data when the user's hand turns green (discussed in Section 3.4.3), when the user finds out the hand turns green, usually, it takes the user a short time to react then start moving. Therefore, the reaction data (the two places called reaction in Figure 3.9) should not be counted as part of the user's hand motion, since we want to know the actual start time that the user moved the hand. In Figure 3.9, the point cut1 and cut3 could be considered as the real starting point of each stage.

The second invalid data is the system count down with is a fixed 2 second shown in the middle of Figure 3.9. It just gives users a preparation time which is ready for moving their hands back to initial position.

The third invalid data is called 'Signal Detect Stop' (Figure 3.9) which happened at the end of each stage. The system detects whether users want to stop by checking whether their hands are not moving for continuous half second (the static pose called signal mentioned in Section 3.4.2). Before the system detects that stop signal, the system will still collect the data. Therefore, we should move this part and find the actual stop position. In Figure 3.9, the point cut2 and cut4 could be considered as the real finish point of each stage.

The feature of valid data:

The whole curve of Figure 3.9 is a set of data point, each element in the set is labeled by its location in the data set, from 0 to the (length-1). There are several special elements will be used for explanation, there are labeled with a name along the curve. In Figure 3.9, we could say the data inside the green rectangle is the valid data, which is between the points labeled with cut1, cut2 and cut3, cut4. The valid data has the following features.

For stage 1, the curve is going down. The value of start and end points should nearly be the maximum and minimum measurement parameter's value of the Stage 1 (since the global min, max for Stage 1 are possible to come from the invalid data set). For stage 2, the curve is going up. The value of start and end point should nearly be the minimum and maximum measurement parameter's value of the Stage 2. The overall trend of the curve is going down or going up. There is another feature for the curve. The curve could be very smooth, such as the curve between cut3 and cut4 in Figure 3.9. It could also be not smooth, such as the curve between cut1 and cut2 in Figure 3.9. Sometime, the curve is going down sharply, but it could also be very stable. The curve could even be going up and down in Stage 1 or Stage 2. No matter what types of curve we got, the overall trend for Stage 1 and 2 are not change which are going down and up respectively.

Aim and ideas of the algorithm:

The aim of the step 2 in Figure 3.6 is to do the valid data extraction on the output of data preprocessing (Section 3.5.1), remove all invalid data and get the correct performance result for users, which means that we extract valid data from Figure 3.9 to Figure 3.10. Since this changing pattern is exactly same for all gestures. The curves for θ and D are also following the same pattern. Therefore, Algorithm 4 could be used for all gestures.

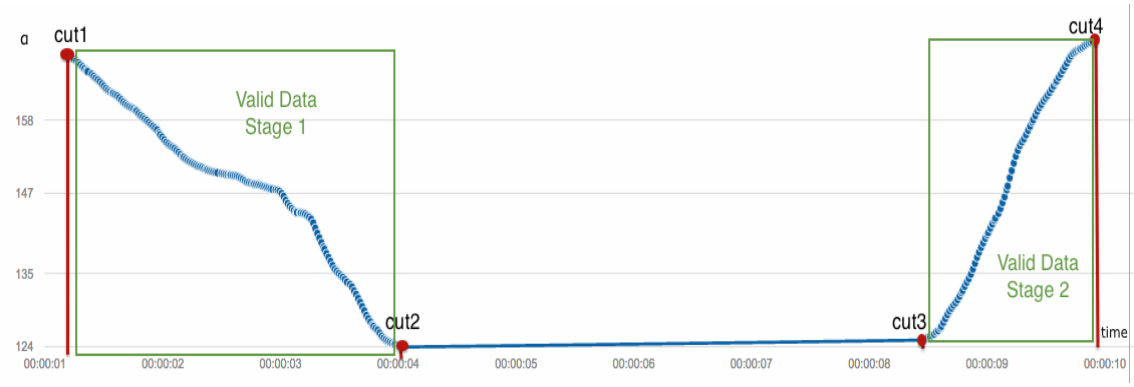


Figure 3.10: Data visualization after running the algorithm

The ideas of Algorithm 4 were based on the features of the valid data mentioned above. We deal with Stage 1 and Stage 2 situation separately.

For Stage 1, the whole curve of stage 1 is divided into multiple small segments which have the same trend. One segment is represented by a pair (a, b) which means the segment from point a to point b, a and b are location information. The system will only analyze the segments which are contentiously going down (Going down means calculate the gradient of an adjacent pair, if the gradient less than 0 and the absolute value larger than a threshold, we could say going down). One satisfied example could be the pair (cut1, p1) in Figure 3.9, the curve between cut1 and p1 is continuously going down. The reason we only focus on the segments with reducing trends are that the overall trend for valid data of Stage 1 is going down. The segments in reducing trends are possible to be part of valid data. Then we choose the one which is most likely to be valid data. We choose the segment with the largest length to be the most likely one. And we check whether it is good enough by comparing with the min and max value of that segment to the global min and max value of Stage 1. If the difference smaller than a threshold, we could say it is good enough. If not, we keep merging other segments into one until we satisfy the requirement.

Here is a simple example for Stage 1 in Figure 3.9:
(We considered that the curve from p1 to p2 and p3 to p4 is stable)
We have a set of segments S covers all reducing trend for Stage 1 in Figure 3.9.
S = {(cut1, p1), (p2, p3), (p4, cut2) } The longest element is (cut1, p1).
There are 3 steps to achieve Stage 1 valid data extraction.

Step 1: Choose (cut1, p1). The value at point cut1 near global max. However, the value at p1 is far away from global min.

Step 2: Merge with the next segment. Merger (cut1, p1) and (p2, p3), we get (cut1, p3). However, the value at p3 is far away from global min. (If far away from global max, merge with a segment located before the chosen one)

Step 3: Merge with next segment. Merge (cut1, p3) and (p4, cut2), we get (cut1, cut2). The value at cut2 is near global min. The pair (cut1, cut2) is good enough. The data between cut1 and cut2 is valid.

For Stage 2, we follow the similar idea, the whole curve of stage 2 is divided into multiple small segments which have the same trend. The system will only analyze the segments which are contentiously going up (Going up means calculate the gradient of an adjacent pair, if the gradient larger than 0 and the absolute value larger than a threshold, we could say going up). The reason we only focus on the segments with increasing trends are that the overall trend for valid data of Stage 2 is going up. The segments in increasing trends are possible to be part of valid data. Then we choose the one which is most likely to be valid data. We choose the segment with the largest length to be the most likely one. And check whether it is good enough by comparing with the min and max value of that segment to the global min and max value.

The detail of Algorithm 4 for valid data extraction is shown below.

Algorithm 4 Valid Information Extraction Algorithm

```
1: procedure EXTRACTION(A)
2: // Input A: an associative array: Key: time, Value: measurement parameter 's value at that time point.
3: // Output Points: 4 cutting points which is the cut1 to cut4 in Figure 3.9, which gives the boundary location information of valid data
4:
5: //Step 1: Get global min,max of Stage 1 and Stage 2 for compassion in later step
6:     stage1Array ← A.getFrontStageData          ▷ all data before system count down
7:     stage2Array ← A.getBackStageData           ▷ all data after system count down
8:     min1 ← stage1Array.getMin                  ▷ get min parameter's value of Stage 1
9:     max1 ← stage1Array.getMax                  ▷ get max parameter's value of Stage 1
10:    min2 ← stage2Array.getMin                  ▷ get min parameter's value of Stage 2
11:    max2 ← stage2Array.getMax                  ▷ get max parameter's value of Stage 2
12:
13: //Step 2: Calculate gradients of all adjacent pairs of input A
14:    timeStamps ← A.getAllKeys          ▷ all keys of A, all time (x axis) information.
15:    length ← A.getSize                                  ▷ how many elements in A
16:    gradients = {}                          ▷ Store all gradients of all adjacent pairs
17:    for all i = 0 to length-2 do
18:        x1 ← timeStamps[i]                  ▷ x2-x1, difference of x axis (Key: time)
19:        x2 ← timeStamps[i+1]
20:        y1 ← A[x1]       ▷ y2-y1, difference of y axis (Value: measurement parameter)
21:        y2 ← A[x2]
22:        gradients[i] = (y2-y1)/(x2-x1)
23:
24: //Step 3: find segments for stage 1 and stage 2
25:    gradients1 ← gradients.getStage1                ▷ get all gradients of stage 1
26:    gradients2 ← gradients.getStage2                ▷ get all gradients of stage 2
27:
28: //Stage 1 segments: gradient < 0, |gradient| > threshold, record all continuous pairs
    One example of segments for Stage 1: the segment (a,b) means that all gradients value
    satisfy the condition, from the ath to bth element of gradients1
29:    segments1 ← gradients1.getSegments          ▷ get the segments set for Stage 1
30:
31: //Stage 2 segments: gradient > 0, |gradient| > threshold, record all continuous pairs
32:    segments2 ← gradients2.getSegments          ▷ get the segments set for Stage 2
33:
34: //Step 4: find cut point for stage 1 and stage 2
35:    (cut1,cut2) ← findCutPointForStage1(segments1,min1,max1,A)    ▷ Algorithm 5
36:    (cut3,cut4) ← findCutPointForStage2(segments2,min2,max2,A)    ▷ Algorithm 6
37:    return (cut1, cut2, cut3, cut4)
```

There is one simple example shown before about how to find the cut point cut1 and cut2 of Figure 3.9 through segments merging and min/max value comparison. Algorithm 5 and Algorithm 6 are the detail processes of how to achieve that for Stage 1 and Stage 2 respectively (two algorithms are slightly different in the comparison process). They both take 4 elements as input. The first one is the segments of its own stage, the second and third are the global min and max value of its own stage, and same input data A as Algo-

rithm 4 (use for accessing the keys and values of the array for comparison). The output for Algorithm 5 are cut1 and cut2 (start and finish point for Stage 1). The output for Algorithm 6 are cut3 and cut4 (start and finish point for Stage 2).

Algorithm 5 and Algorithm 6 are the key helper functions for Algorithm 4. Two functions `findCutPointForStage1` and `findCutPointForStage2` in Algorithm 4 Line 35 and 36 are Algorithm 5 and Algorithm 6 respectively.

Algorithm 5 Find Cut Point For Stage1

```
1: procedure FINDCUTPOINTFORSTAGE1(segments,min,max,A)
2:     cut1 ← 0                                                    ▷ initial result
3:     cut2 ← 0                                                    ▷ initial result
4: //Step 1: find the longest segment among segments
5:     longestSegment ← segments.getLongest     ▷ the segment covers most data points
6: //Step2: Check whether the longest is good enough
7:     error ← 2                                                   ▷ accept error range
8:     cut1 ← longestSegment[0]        ▷ longestSegment = (a,b) longestSegment[0] = a
9:     cut2 ← longestSegment[1]                                    ▷ longestSegment[1] = b
10:    keys ← A.getAllkeys                                         ▷ get all keys (Key: time)
11:    currentMax ← A[keys[cut1]]                                  ▷ get current max value from A
12:    currentMin ←A[keys[cut2]]                                   ▷ get current min value from A
13: //Step3: For max boundary
14: //case 1: good enough
15:    if max - currentMax <= error then
16: //current cut1 is good enough
17: //case 2: Not good keep merge
18:    else
19:        while segment' ← getBeforeSegment() do          ▷ Comment 1 (on next page)
20:            maxValue' = A[keys[segment'[0]]      ▷ merge first, then check current max
21:            if maxValue' >= currentMax then                      ▷ max point updates
22:                currentMax ← maxValue'
23:                cut1 ← segment'[0]                               ▷ update boundary
24:                if max - currentMax <= error  then ▷ if fail, keep merge previous one
25:                    Break                ▷ go out of loop, current cut1 is good enough.
26:
27: //Step4: For min boundary
28: //case 1: good enough
29:    if currentMin - min <= error then
30: //current cut2 is good enough
31: //case 2: Not good keep merge
32:    else
33:        while segment' ← getNextSegment() do          ▷ Comment 2 (on next page)
34:            minValue' = A[keys[segment'[1]]      ▷ merge first, then check current min
35:            if minValue' <= currentMin then                      ▷ min point updates
36:                currentMin ← minValue'
37:                cut2 ← segment'[1]                               ▷ update boundary
38:                if currentMin - min <= error  then       ▷ if fail, keep merge next one
39:                    Break                ▷ go out of loop, current cut2 is good enough.
40:    return (cut1, cut2)
```

Two comments used in Algorithm 5 and 6:
Comment 1: If currentSegment location in segments is i, then getBeforeSegment() return the element in i-1 position, it will stop if the before segment does not exist.
Comment 2: If currentSegment location in segments is i, then getNextSegment() return the element in i+1 position, it will stop if the next segment does not exist.

Algorithm 6 Find Cut Point For Stage2

```
1:  procedure FINDCUTPOINTFORSTAGE2(segments,min,max,A)
2:      cut3 ← 0                                                    ▷ initial result
3:      cut4 ← 0                                                    ▷ initial result
4:  Step 1: find the longest segment among segments
5:      longestSegment ← segments.getLongest     ▷ the segment covers most data points
6:  //Step2: Check whether the longest is good enough
7:      error ← 2                                                   ▷ accept error range
8:      cut3 ← longestSegment[0]        ▷ longestSegment = (a,b) longestSegment[0] = a
9:      cut4 ← longestSegment[1]                                    ▷ longestSegment[1] = b
10:     keys ← A.getAllkeys                                         ▷ get all keys (Key: time)
11:     currentMax ← A[keys[cut4]]                                  ▷ get current max value from A
12:     currentMin ←A[keys[cut3]]                                   ▷ get current min value from A
13: //Step3: For max boundary
14: //case 1: good enough
15:     if max - currentMax <= error then
16: //current cut4 is good enough
17: //case 2: Not good keep merge
18:     else
19:         while segment' ← getNextSegment() do                    ▷ Comment 2 above
20:             maxValue' = A[keys[segment'[1]]      ▷ merge first, then check current max
21:             if maxValue' >= currentMax then                     ▷ max point updates
22:                 currentMax ← maxValue'
23:                 cut4 ← segment'[1]                              ▷ update boundary
24:                 if max - currentMax <= error  then      ▷ if fail, keep merge next one
25:                     Break                  ▷ go out of loop, current cut4 is good enough.
26:
27: //Step4: For min boundary
28: //case 1: good enough
29:     if currentMin - min <= error then
30: //current cut3 is good enough
31: //case 2: Not good keep merge
32:     else
33:         while segment' ← getBeforeSegment() do                  ▷ Comment 1 above
34:             minValue' = A[keys[segment'[0]]      ▷ merge first, then check current min
35:             if minValue' <= currentMin then                     ▷ min point updates
36:                 currentMin ← minValue'
37:                 cut3 ← segment'[0]                              ▷ update boundary
38:                 if currentMin - min <= error  then  ▷ if fail, keep merge previous one
39:                     Break                  ▷ go out of loop, current cut3 is good enough.
40:     return (cut3, cut4)
```

Result generator:

The final performance result is generated by using all algorithms mentioned above. We firstly do the data preprocessing in order to get the associative array shown in Figure 3.11 (ID in the Figure represents the location of each (Key, Value) pair in the array). Then we do the valid information extraction and find out the real user's movement data (among all collected data), which is the data inside two red rectangles in Figure 3.11. The cut1 to cut4 in Figure 3.11 were found by using Algorithm 4.

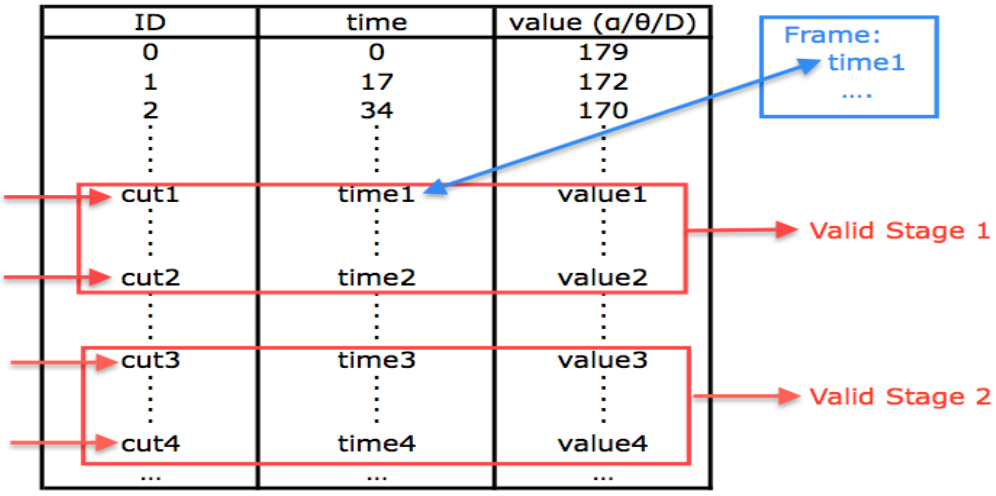


Figure 3.11: Structure of the input 2 (output 1) in Figure 3.6

First key result:

Figure 3.7 shows what we should provide to users as the overall performance feedback. The common measurement parameter, the time (how long it takes to complete the exercise) should be calculated using (time2 - time1) + (time4 - time3) (in Figure 3.11) which is the time for valid stage 1 plus the time for valid stage 2 (Figure 2.1, 2.3 and 2.4 explains that one complete required gesture contains two stages). The time is the first key result.

Second key result:

In addition, in order to access the overall performance of the gesture, we merge valid stage 1 and stage 2 together and find the min and max value of the gesture measurement parameter among value1, value2, value3 and value4 (they are the min and max value for their corresponding stage). The global min and max value represents the boundary of how the user could achieve in the whole gesture performance process. In this project, the global minimal value (achieved at the gesture goal position) of the measurement parameter for the corresponding gesture is more important. Because it is the limitation of what the user could achieve. The measurement parameter α for Gesture 1 & 4 and θ for Gesture 2 & 5 (Figure 3.7), have four minimal values respectively, for four fingers. We used the average to represent the global minimal value. The minimal value is the second key result.

Other results:

There are some other results which may be helpful to users. One example could be the max value we mentioned in the last section (second key result) which help us find out the range of hand motion. Besides, given the min and max value, we could find their corresponding key such as the key for value1 in Figure 3.11 is time1. The time keeps

changing and it is a unique value. We could use the time to search the corresponding frame information from the collected data (the blue part in Figure 3.11). Therefore, if we know the global min and max value, we could find the corresponding two frames. Each frame captures a hand moving moment and contains useful hand tracking information (Section 2.2.2). We could provide more information based on what contains inside the two frames to find out what happened at that moment in which the user reached the hand movement limitation. Section 3.6 below has more detail information about the results, including which information contains in the final feedback result and how to show the feedback to the user.

3.6 Feedback Page design

This section talks about the design of the Feedback Page including which information should be included in the Feedback Page (listed below) and how to show the feedbacks (explained by using one feedback example for Gesture 2 generated by the system shown in Figure 3.12 and Figure 3.13).

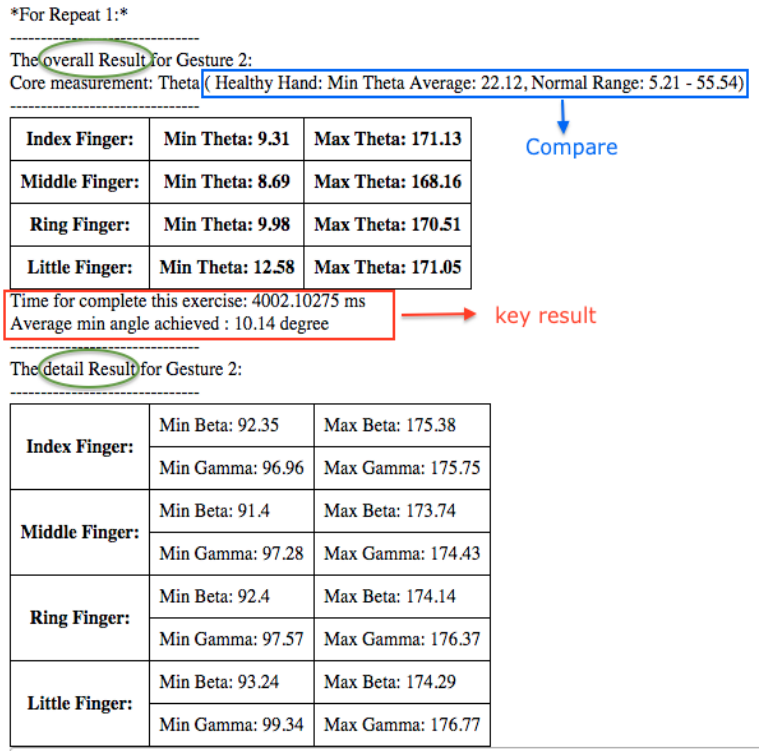


Figure 3.12: One table feedback example for Gesture 2

What's inside the Feedback Page?

1. Overall result:
   Section 5.1.5 explains which measurement parameters are used for evaluating the user's overall performance. At the beginning of the feedback, the system shows the overall result. For example, the red rectangle in Figure 3.12 (Gesture 2 use θ to evaluate overall performance) shows the complete time and average minimal value of θ (they are the key result mentioned at the end of section 3.5.2). Besides, the minimal and maximal values of θ for each finger are also shown above the red rectangle.

2. Compare information:
   The blue rectangle in Figure 3.12 shows the normal range of θ and the average minimal θ achieved by healthy people (the result of Section 5.3). Patients could compare their minimal θ in red rectangle with the information inside the blue rectangle to find out how well they performed.

3. Detail information:
   Table 2.1 in Section 2.1.1 summarizes what detail information the doctors wants the system to show. For example, for Gesture 2, the values of β and γ for each finger are required to provide to patients. Therefore, below the red rectangle in Figure 3.12, we provide those required information (they are called other results at the end of section 3.5.2) for the patients.

4. The whole process graph:
   The gesture performed by the user is a dynamic process. The system uses a line chart to represent the process by showing how the value of the measurement parameter changed along the time line, and the line chart clearly reflects the changing trend. Figure 3.13 is a graph feedback example for Gesture 2. Each color in Figure 3.13 represents how the value of θ change along the time line for one finger.

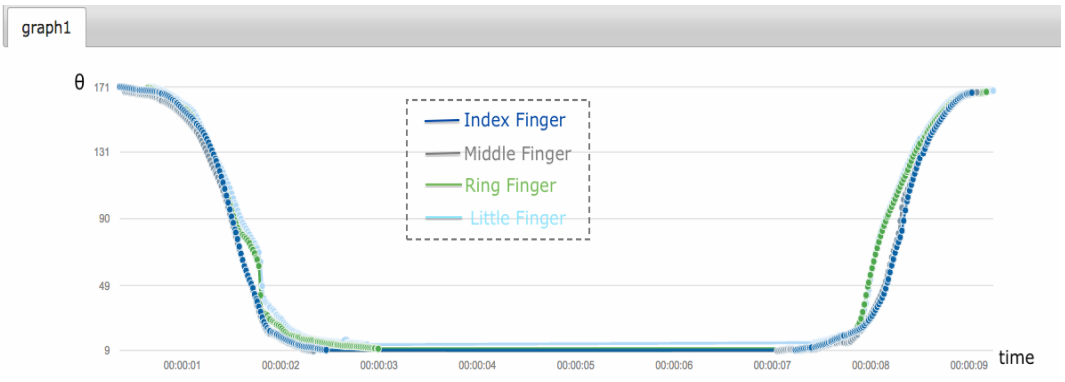


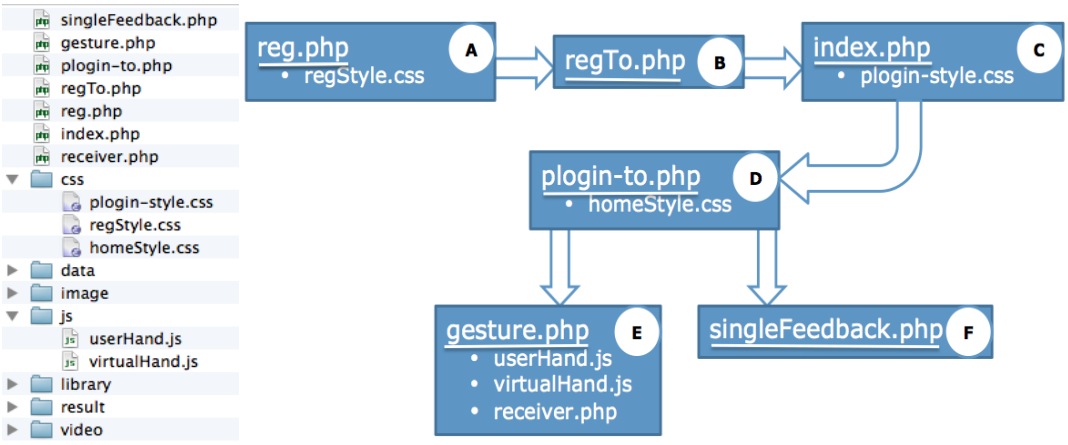
Figure 3.13: One graph feedback example for Gesture 2

From Section 2.1, we know that one exercise contains two gestures for the right and left hand. The Feedback Page structure for two hand gestures are the same. For example, the Exercise 2 contains Gesture 2 and Gesture 4, this section only shows the feedback example of Gesture 2 in Figure 3.12 and 3.13, the feedback for Gesture 4 follows the same structure by using tables and graphs to provide all information the user needs to know for Exercise 2. Each exercise's feedback contains the four parts listed above. Appendix A shows the Feedback Page for Exercise 1 (Gesture 1 & 4) and Exercise 3 (Gesture 3 & 6) generated by the system.

Chapter 4

System implementation

This chapter shows how the system was implemented based on the design of Chapter 3. The Hand Physiotherapy system is a web-based system which contains several web pages implemented based on its functionality. For the Front-End development, I used JavaScript, HTML and CSS, which aimed for building, styling and programing web pages respectively. For the Back-End, I used PHP for web server programming and SQL for accessing the database.



(a) (b)

Figure 4.1: Source code structure

Figure 4.1(a) shows all source codes for the project. There are seven folders. The `css` folder stores all styling files for the system and the data folder stored 6 perfect required hand movements in JSON format, they are used for Virtual Hand (Section 3.4.2) movement. The `image` and `video` folder store all media resources applied in the system. Besides, the `result` folder stores all registered users' performance. The `library` folder provides all APIs I used in the system, such as the Leap Motion JavaScript API, LeapJS Plugins (mentioned in Section 2.2, 2.3) etc. The `js` folder stores all JavaScript files I implemented for the system. All of these folders provide support for the main functions of the system.

The rest files in the project folder are the implementation of main system functionalities which consists of six components with the relationship shown in Figure 4.1(b) (the external stylesheet and javescript links of that page are listed inside each component). The components from A to D allow users access the system, including register, login and get into the Home Page of the system and their detail implementation is explained in Section 4.1. Section 4.2 talks about component E in Figure 4.1(b), which allows users to do the hand

recovery exercises. The component F aims for proving feedbacks of users' performances, which contains the implementation of the algorithms mentioned in section 3.5. Section 4.3 discusses how these algorithms implemented in order to get the feedback result.

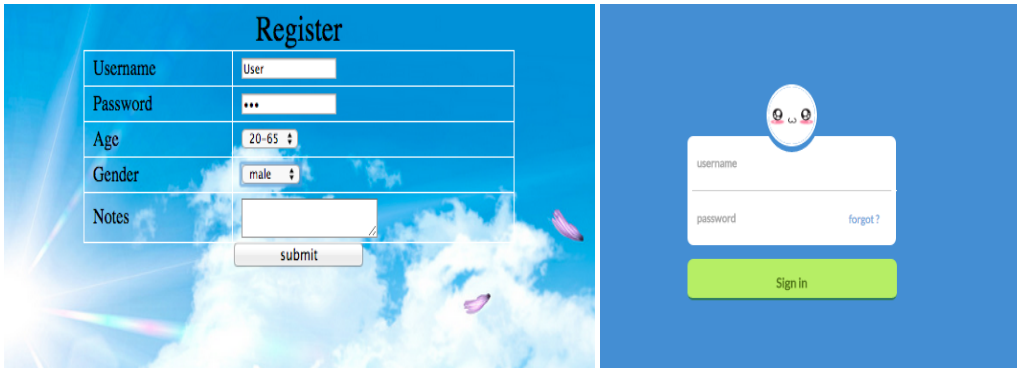
4.1 System Access & Home Page implementation

This section talks about component A to D in Figure 4.1(b) which contains the system access (register & login) and the Home Page implementation.

4.1.1 Register & Login implementation

All users need to register first, the component A (`reg.php`) is the implementation of the Register Page shown in Figure 4.2(a). In this web page, users need to enter their personal information, and all input data will be submitted to the database. I used HTML <`form`> element defined below to collect users' input information such as username, gender, age and so on. Two form attributes, `action` and `method` were used below. When users click on the submit button, all form data is sent to the file called `regTo.php` using `post` method (HTTP post transaction).

```
<form action="regTo.php" method="post">
...<input type="text" name="username"/>
...<input type="submit" value="submit"/></form>
```



(a) Register Page (b) Login Page

Figure 4.2: Screen shots of Register & Login Page

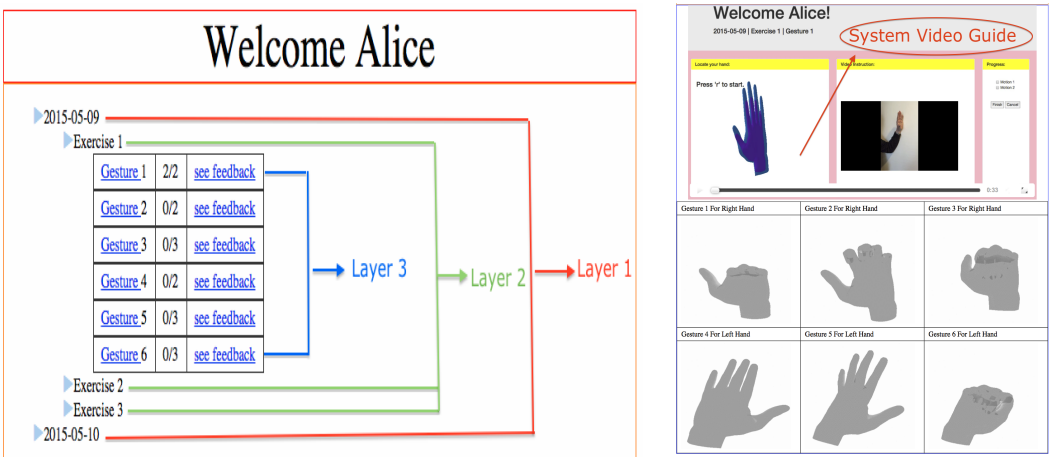
The component B in Figure 4.1(b) `regTo.php` is used for connecting, storing the form information to the database and also initializing hand exercises for users. I used the open source database MySQL running on my own computer for the project. The following code shows how I connect and insert data into the database. I created one MySQL database called `project`, there are two tables based on the design of Section 3.2, the table called `patient_info` stores patients' information. For each new user, there will be one new record in this table. I added the record using SQL query `INSERT INTO` statement. The following codes are the basic 4 steps to achieve this. I also used the `INSERT INTO` statement to add new exercise records into the second table called `exercise_record`, which gives users' initial hand exercises covering all required dynamic gestures (as one of the future work, there will be a doctor interface, which could set up more exercises to each user, this project only focused on patients' interface).

```
//setp 1: connect
$db = new MySQLi('127.0.0.1', $admin, $adminPassword);...
//step 2 select database
$db->select_db('project');
//step 3: insert query
$sql="insert into patient_info values (NULL, '$name', '$username',
'$password', '$age', '$gender', '$note')";
//step 4: run query to insert
$db->query($sql); ...
```

After a successfully register process, the system automatically jumps to the Login Page shown in Figure 4.2(b). The component C in Figure 4.1(b) `index.php` is the implementation of Login Page. The system used HTML <`form`> element to collect users' input and send their login information into the component D in Figure 4.1(b) called `plogin-to.php`.

4.1.2 Home Page implementation

The `plogin-to.php` is the implementation of the Home Page in Section 3.3. Firstly, the system will check whether the login information is valid. For a valid user, the system shows all exercises of this user, Figure 4.3(a) is an example.



(a) Home Page exercise list (b) Home Page system guide

Figure 4.3: Screen shots of the Home Page

Access database:

I wrote SQL statements to access all exercise information of each user. The Home Page exercise list shown in Figure 4.3(a) could be summarized that each day has several exercises and each exercise contains six required dynamic gestures. Each gesture may need to repeat several times. We divided the exercise list into three layers.

The first layer (colored in red in Figure 4.3(a)) required the system to show which dates the user was assigned exercises. Therefore, I run the query below to find out all distinct dates that the user was assigned exercises.

```
// Get all distint date through the unique patient ID (pid)
$sql= "SELECT DISTINCT date FROM exercise_record where
pid = $pid ORDER by date";
```

The second layer (colored in green in Figure 4.3(a)) is that for each individual date, the user has several exercises (Exercise 1,2,3). Therefore, I run the query below to find out all distinct exercises for that day.

```
//Get all exercise for each day through, pid and date information
$sql = "SELECT DISTINCT task_id FROM exercise_record where pid=$pid
and date='$date' ORDER by task_id";
```

The third layer (colored in blue in Figure 4.3(a)) is that, for each exercise of a particular date, there are 6 required gestures, for each gesture, users may be requested to repeatedly doing several times, which shows in this format of (Finished/OverallRepeat), for example (0/2) means the gesture need to repeat twice and so far, 0 repeat had finished. I run the query below to find out how many repeats were needed for each gesture. To check how many repeats finished yet, we just need to add '`and finish=1`' (`finish` is one column of `exercise_record` table, 1 means finished) at the end of the statement below.

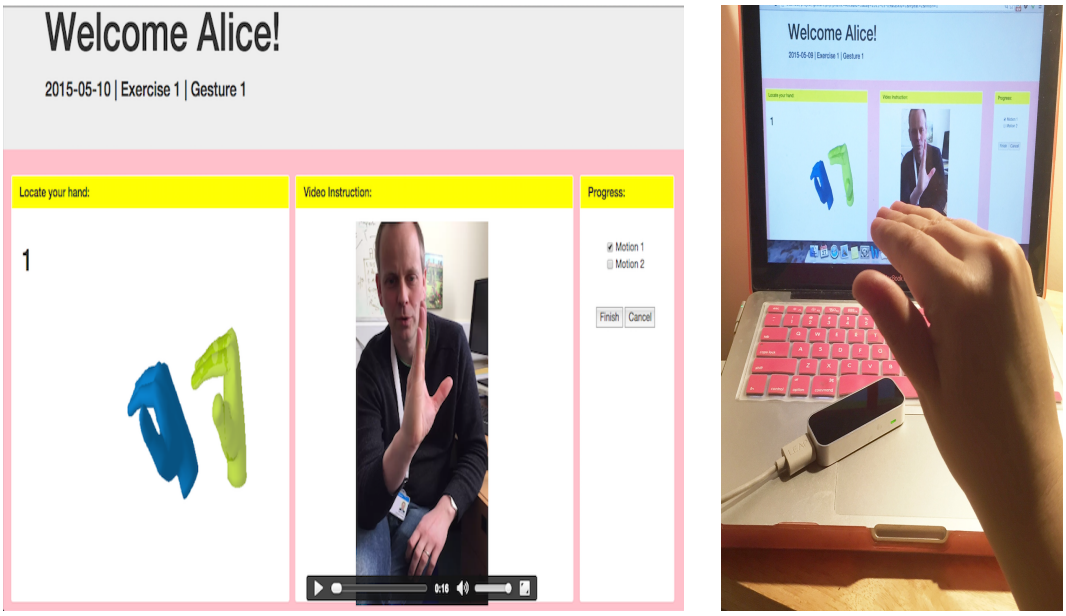
```
//Get each gesture repeat time through pid,date,task_id,gtype(gesture ID)
$sql = "SELECT COUNT(gid) AS cgid FROM `exercise_record` WHERE pid=$pid
and date='$date' and task_id=$taskId and gtype=$k";
```

To show all the information in a designed structure in Section 3.3, I chose to use HTML <ul> and <li> Tag, which represent unordered list and list item respectively. In order to achieve fold down performance, I used one function (fold down list[2]) to achieve hidden or unhidden list children. Besides, I used HTML `Table` element (using <table>,<tr>,<td> tag) to show six required gestures, users could click the gesture name (Gesture 1..6) to access into the corresponding page to do a new hand exercise or click 'see feedback' in Figure 4.3(a) to check the performance of a completed exercise.

The Home Page could also provide system guide information shown in Figure 4.3(b) which includes a video about how to use the system (used HTML <video> element to show video on the web page) and animated `GIF` images to describe each gesture (used HTML <img> Tag to show images on the web page) based on the design on Section 3.3.

4.2 Hand Exercise Assistant implementation

This section explains the implementation of Figure 4.1(b) component E `gesture.php` and three included files `virtualHand.js`, `userHand.js` and `receiver.php`. Figure 4.4(a) is the implementation result of `gesture.php`. It is the main web page for users doing a particular hand recovery exercise. For example, if the user puts the hand over the Leap Motion sensor shown in Figure 4.4(b), we could get the main user interface shown in Figure 4.4(a). The layout of this page applied Bootstrap Grid System[26] which provides a fast and simple way for the web page layout. The key implementations of this page are the Virtual Hand (the blue one in Figure 4.4(a)), Real Hand (the green one in Figure 4.4(a)) and how these two hands interact with each other.



(a) (b)

Figure 4.4: Screen shots of Exercise Assistant & Record Page

4.2.1 Virtual Hand implementation

The `virtualHand.js` is the implementation of Virtual Hand which mainly achieved be using two LeapJS Plugins mentioned in Section 2.3.3. In order to use the two plugins, it was required to create a controller for the Virtual Hand using `new Leap.Controller()` provided by Leap Motion API. There are mainly two steps to achieve our goal shown in Figure 4.5. First of all, I used the playback plugin to play the prerecorded dynamic gesture (stored in the data folder with the path on line 8) as the animation instruction, then initialized the basic play options. Secondly, I applied the riggedHand plugin (on line 15) with initial setting such as scale, bone color in order to add the Leap-Enabled hand (the blue hand in Figure 4.4(a)).

```
5   //Step 1: Use LeapJS Plugins Playback
6   controller.use('playback', {
7       // a JSON file of preprecorded dynamic gesture
8       recording: './data/g'+gid+'.json',
9       //initial settings
10      timeBetweenLoops: 1000, pauseOnHand: false, autoPlay: true, loop:false
11  }).on('riggedHand.meshAdded', function(handMesh, leapHand){
12      handMesh.material.opacity = 1;
13  });
14  //Step 2: JS Rigged Hand Plugin: add Leap-Enabled hand
15  controller.use('riggedHand', {
16      //initial settings
17      scale: 2,
18      boneColors: function (boneMesh, leapHand){
19          return {
20              hue: 0.564, saturation: 0.9, lightness: 0.5
21          };
22      }
23  });
```

Figure 4.5: Screen shot of main codes in `virtualHand.js`

In order to put the virtual hand in the right place, I used the following codes to get the rendered 3D hand model and located it in the right place on the web page shown in Figure 4.4 through unique HTML element ID called `UserHandPanel`.

```
var r1 = virtualHandController1.plugins.riggedHand.renderer;...
document.getElementById("UserHandPanel").appendChild(r1.domElement);
```

4.2.2 Real Hand implementation

The `userHand.js` is the implementation of the Real Hand (the green one shown in Figure 4.4(a)). Similar to the Virtual Hand, I created another controller for the Real Hand which used for connecting the system and the Leap Motion sensor in order to get the hand tracking data. I also used the riggedHand plugin with different color settings to add the 3D real hand model to the page.

The Virtual Hand and Real Hand were located in the same section. The only difference is that the Virtual Hand was set up in a fixed position, but the Real Hand's position is not fixed in that section. Users could move their hands to a comfortable position they want within that section (two hands could be apart or overlapped). The real hand implementation mainly achieved by using one method called `on` shown below, which used for defining the callback function. The `userHandController` will keep passing the most recent `frame` to the callback function with a frame rate, the callback function operates the input frame data (mainly for status update and data storage).

```
var userHandController = new Leap.Controller;
userHandController.on('frame',function(frame){
... status update, data storage...}
```

For data storage, there is one boolean value called `store` control whether the `frame` data needed to be stored. The value of `store` will be updated based on the current status (discuss in Section 4.2.3). I created one array called `allHandDataStore` in the code, which used for storing all users' performance of this gesture practice (it may contain several repeats). The first element of the array stores basic exercise information such as exercise date, gesture ID, etc. Each repeat performance will be stored into the array called `handDataStore` temporarily, when one repeat finished, add the current `handDataStore` into `allHandDataStore`, and clean `handDataStore` ready for storing next repeat.

4.2.3 Interaction implementation

The Figure 3.5 defined the interaction process. This section talks about how to achieve this through status updates. In `userHand.js`, there is a boolean value '`green`' which used for controlling the color of the user's hand based on the design. Another boolean value '`go`' controls whether the system need to detect the user' stop signal (hold the hand for 0.5 second) under the condition that the user has not finished all repeats yet. There are three places need to check the user's signal in Figure 3.5 (three red Signal signs over the arrow of that Figure). The user follows the movement of the Virtual Hand, therefore when the Virtual Hand stops moving, the system should start detecting whether the user's hand is stopped.

We check whether the user's hand could hold for a fixed duration as the stop signal. We get the difference of the current `frame` time point and the `startTimeStamp` and check whether the difference larger than the fixed duration. In the code, there is one value called `startTimeStamp` which always records the start time point of a continuous stop signal and it will keep updating using one method called readyToStart. The main codes for Real and Virtual hand interaction shown in Figure 4.6 (When the boolean value `go` is true and this exercise has not finished yet, we run the code below).

```
78  //Satisfy static requirements
79  if(vol <= 40){
80      //Continue holding for a fix duration, valid signal
81      if(frame.timestamp - startTimeStamp >= duration){
82      //for a phase finished signal: movement finsihed hand color change, only store data when hand is green
83          green = false; store = false;
84          //Middle Point: The start point of a new repeat(not first one) also the finish point of last repeat
85          if(phase == 0 && notFirstTime){
86              //finished exercise +1, store last repeat data, clean array ready for next, tick the checkbox
87              gfin ++;  allHandDataStore.push(handDataStore);  handDataStore = new Array;
88              document.getElementById(gfin).checked = true;
89          }
90          switch(phase) {
91          case 0:   // phase 0: hand going down to goal position
92              notFirstTime = true; go = false;
93              if(gfin == gre){ // all repeat completed
94                  showInDiv("countDown", "<h3>Perfect! Job Done! </h3>");
95              }else{ // System count down 3,2,1, Play the virtual hand
96                  showInDiv("countDown", "<h1>3</h1>");
97                  setTimeout(function(){showInDiv("countDown", "<h1>2</h1>");}, 1000);
98                  setTimeout(function(){ showInDiv("countDown", "<h1>1</h1>");}, 2000);
99                  setTimeout(function(){ virtualHandController1.plugins.playback.player.play();
100                     green = true; store = true;}, 3000);
101                 setTimeout(function(){ //after forward time, virtual hand pause, move to phase 1, check signal
102                     virtualHandController1.plugins.playback.player.pause(); readyToStart();
103                     phase = 1; go = true}, forward);
104             }
105             break;
106         case 1: //phase 1;  going back : System count down: ready, Go Play the rest movement of virtual hand
107             go = false; showInDiv("countDown", "<h1>ready</h1>");
108             setTimeout(function(){showInDiv("countDown", "<h1>GO</h1>");}, 1000);
109             setTimeout(function(){ virtualHandController1.plugins.playback.player.play();
110                 green = true; store = true;}, 2000); // after back time, virtual hand pause, back to phase 0, check signal
111             setTimeout(function(){ virtualHandController1.plugins.playback.player.pause();
112                 readyToStart(); phase = 0; go = true;}, back);
113             break;
114         default:
115             break;
116         }
117     }
118 }else{ //Not satisfy static requirement find a new start point
119     readyToStart();
120 }
```

Figure 4.6: Screen Shot of selected codes in `realHand.js` for hands interaction

The codes start with checking the sum of `tipVelocity` (`vol`), if the `vol` satisfies the static status requirement (explained in Section 3.4.2 signal), we check whether this status continuously holds for a fixed duration. If not (on line 118), the system runs `readyToStart` function to find a new start time point for `startTimeStamp`. If yes, this means the system detects a valid signal successfully. There are 3 situations phase 0, 1 and one middle point shown in the Figure 4.6(on line 91, 106 and 84). For each repeat, there are two phases corresponding two stages of each hand gesture.

- Phase 0 (on line 91) means the user's hand should move from initial to goal position.
- Phase 1 (on line 106) means the user's hand should move from goal to initial position.

In phase 0 and 1, I both used `setTimeout` methods with two parameters, one is the function and the other one is the time. The `setTimeout` operation is that the system will wait for a fix time (the second parameter) and then execute the first parameter the function. There is one method called `showInDiv`, given a unique HTML element ID and the word instructions, `showInDiv` will add the word instruction into a fix location which is the left top corner of the hands visualization panel.

In phase 0, I used `setTimeout` method to achieve the count down from 3 to 1 (on line 96 - 99). Then the system actives the Virtual Hand controller to play the animation (on line 99). The boolean value, `green` and `store` update. Users could move their hands and the system starts recording their performance. The Virtual Hand's movement pause when the Virtual hand reaching the goal position (on line 102). The system updates the value of `phase`, `go`, the system moves to the next phase, and starts detecting the user's stop moving signal.

In phase 1, I used `setTimeout` method to achieve the count down from ready to go (on line 107 - 108). Then the system actives the Virtual Hand controller (on line 109) to continue play the animation again from the goal position back to the initial position and pause again (on line 111) if a completed the virtual hand movement finished. The system updates the value of `phase`, `go`, moves back to phase 0 again and starts detecting users' stop moving signal based on the design.

For a middle point which is the start point of a new repeat and the end point of last finished repeat (on line 84). The system stored the last repeat performance data into the `allHandDataStore` array, clean the `handDataStore` array for the next repeat and update the progress tracking panel by ticking the corresponding checkbox on the right part of the Figure 4.4(a) (on line 87, 88).

When all repeats finished, users click on the Finish button, all data in `allHandDataStore` will be converted into JSON by using library function `JSON.stringify` which transfers the array to a JSON string. Then I used the `post` method in jQuery (a JavaScript library) to send all data into `receiver.php`. The aim of `receiver.php` was to store the JSON string into the result folder for the user by using library function called `file_put_contents`. After clicking the Finish button, this gesture was completed successfully.

4.3 Algorithm implementation

The Section 3.5 and 3.6 talk about algorithms I designed for processing the collected raw data and feedback design for users. The last component F `singleFeedback.php` in Figure 4.1(b) is the implementation of Feedback Page, including the implementation of all designed algorithms and feedback visualization.

For each single gesture, it may need users repeatedly doing several times. The feedback for a single gesture should contain the performance of all repeats. Figure 4.7 shows the code for system providing the feedback for one repeat performance (other repeats follow the same way to generate the result).

```
77  //Step 1: prepare: calculate angle and remove noise
78  $preparedSet = []; //store the result for raw data preprocessing
79  if($gestureID == 3 || $gestureID == 6){
80      $preparedSet = prepareForG3AndG6($currentRepeat,$gestureID);
81      $minmaxsMiddle = seperateCutReference($preparedSet);
82      //step 2: extraction
83      $cutpoints = getCutPoints($preparedSet,$minmaxsMiddle);
84      $resultPoint = findMinMaxKey($cutpoints,$preparedSet);
85      //Step 3: generate result:
86      for($fi = 0; $fi < 4; $fi++){
87          $resultPoints[$fi] = $resultPoint; //store for detail
88      }
89      $overallResult[0] = generateOverallResult($resultPoint,$cutpoints,$preparedSet);
90      $drawData[0] = [$preparedSet, $cutpoints];
91  }else{ //For Gesture 1,2,4,5
92      //Step 1: prepare: calculate angle and remove noise
93      $preparedSet = angleCalulationAndNoiseRemove($currentRepeat,$gestureID);
94      for($fi = 0; $fi < 4; $fi++){
95          $minmaxsMiddle = seperateCutReference($preparedSet[$fi]);
96          //step 2: extraction
97          $cutpoints = getCutPoints($preparedSet[$fi],$minmaxsMiddle);
98          $resultPoint = findMinMaxKey($cutpoints,$preparedSet[$fi]);
99          //Step 3: generate result:
100         $resultPoints[$fi] = $resultPoint;//for gesture 2,5 detail
101         $overallResult[$fi] = generateOverallResult($resultPoint,$cutpoints,$preparedSet[$fi]);
102         $drawData[$fi] = [$preparedSet[$fi], $cutpoints];
103     }
104 }
105 //Step 4: view overall result
106 viewOverallResult($overallResult,$gestureID);
107
108 //Step 5: videw and generate detail result
109 if($gestureID == 3 || $gestureID == 6 || $gestureID == 2 || $gestureID == 5){
110     for($fi = 0; $fi < 4; $fi++){
111         $currRe = $resultPoints[$fi];
112         generateDetailResult($currRe,$currentRepeat,$gestureID,$fi);
113     }
114 }
115 //Step 6: visulization
116 $draw["graph$r"] = drawGraph($gestureID,$drawData);
```

Figure 4.7: Screen Shot of the selected codes in `singleFeedback.php`

Providing performance feedback to users contains two processes. The first one is the feedback calculation process and the second one is the feedback visualization process. Figure 4.7 clearly shows 6 steps by providing the feedback for one repeat. Among all these steps, the step 1 to step 3 used for feedback calculation process and the step 4 to step 6 used for the feedback visualization process.

Step 1 - 3:

Section 3.5 discusses how to analyze the collected raw data in order to get the feedback result. The first step (mentioned in Section 3.5.1) is the raw data preprocessing and Gesture 3,6 and Gesture 1,2,4,5 have different preprocessing algorithms. This reflects on the real implementation. The step 1 for Gesture 3,6 is shown on line 80, using `prepareForG3AndG6` method which was exactly the implementation version of Algorithm 1 in Section 3.5.1. The step 1 for Gesture 1,2,4,5 shown on line 93, using `angleCalulationAndNoiseRemove` method which was exactly the implementation version of Algorithm 2 in Section 3.5.1. For these two methods, they both took the array called `currentRepeat` and the output stored in an array called `preparedSet`. The `currentRepeat` is the raw data. I used the library function `json_decode` to convert a JSON string (stores hand tracking data) into a PHP object which is an array of `frames`. Figure 4.8 is a single `frame` structure example. The data structure of `preparedSet` is the associative array (a set of key and value pair), Figure 3.11 is an example.

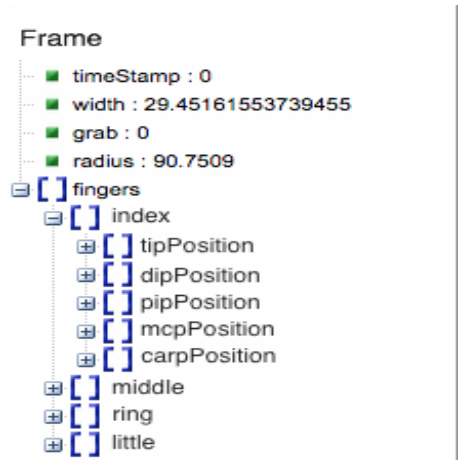


Figure 4.8: Frame Structure

Then all gestures use the `seperateCutReference` method on line 81 and 95. It is a preparation method for the Step 2 valid information extraction, shown on line 83 and 97 using `getCutPoints` method which is exactly the implementation version of Algorithm 4 in Section 3.5.2. The step 3 is generating the overall result on line 89 and 101 using a method called `generateOverallResult`. The output of this method covers the First and Second key results mentioned at the end of Section 3.5.2.

Step 4 - 6:

Step 1 to Step 3 applied algorithms and get the overall performance result. For step 4, I used the method `viewOverallResult` on line 106 to print the overall result on the web page in a table format. As we discussed in Section 3.6. The system could also provide the detail information. For gesture 2, 3, 5, 6, except the overall result, it is very helpful for patients to know extra information. Therefore, I implemented one method called `generateDetailResult` on line 112 (Step 5), to calculate and print the detail performance shown on the web page. The calculation for the detail result is simple. From the overall result, we know two unique time points which make the overall measurement parameter to be max and min. I search all raw data using the two unique time points and get the corresponding two `frames` to calculate the detail information (just the bend angles introduced in Table 2.1). The last Step 6 is the data visualization. I applied a line chart library called `morris.js` [9] to plot the data point which reflects the whole process of the dynamic gesture. The final implementation output of the component F in Figure 4.1(b) shown in Figure 3.12, Figure 3.13, and all figures in Appendix A. They are selected screen shots of the Feedback Page generated by the system.

Chapter 5

Data collection and analysis

The previous two chapters (Chapter 3 and 4) discuss how I achieve the objective 1 and 2 mentioned in section 1.2.1. The chapter focuses on the rest of the objectives. Section 5.1 talks about how I select the measurement parameters for evaluating the overall performance of each gesture, including the issues I met during the selection process and how to overcome those issues. This section gives the answers of the exploring questions proposed in objective 3 in section 1.2.1. Besides, the chapter summarizes the normal range of the key selected measurement parameter for each gesture and discusses what a healthy person could achieve. The process of how I collected the data to get the normal range is explained in this chapter as well.

5.1 Performance measurement parameter selection

The system is expected to provide feedbacks to patients, which could reflect the overall performance of how well they completed the required dynamic gestures. In this project, we use the measurement parameter to evaluate patients' performance.

What is the measurement parameter?
The measurement parameter could be considered as a patient overall performance result which expressed in number. For example, when patients bend their fingers to perform a required exercise, the average bend angle of all fingers achieved by patients could be one performance measurement parameter for that exercise. Different exercises focus on recovering different hand joints. Therefore, the three types of required exercises mentioned in section 2.1 use different measurement parameters.

The approach I used to find the most suitable measurement parameter for each exercise could be divided into two steps which explain in the following sections. Firstly, in section 5.1.1, we gave an initial assumption about which parameters should be used to measure each gesture based on the information we collected from doctors. Then we found out whether the initial assumption works as we expected based on the observation and analysis of the collected data. If some issues were found, we tried to solve them. The section 5.1.2, 5.1.3 and 5.1.4 discuss the measurement parameter we finally selected for three required exercises respectively. Section 5.1.5 is the summarization of measurement parameter selection.

5.1.1 Initial assumption

The evaluation of dynamic gesture mainly depends on two key points. The first one is how long it takes to complete one exercise. To reach a same goal, usually the users without

hand issues may finish slightly easier and quicker. For all exercises, the common measurement parameter is the `time`. The second one is the limitation of the patient could reach, such as which is the minimal bend angle that users could achieve, this reflects the range of human hand function (something that users are able to do). The measurement parameter for each exercise is different.

Our initial assumption was decided based on section 2.1.1. On that section, I discuss the key evaluation factors (called α, β, γ) for all exercises using in clinical diagnosis, which are summarized in Table 2.1. The initial ideas is that the doctors usually use the minimal value of α, β, γ that patients achieved (based on what doctors actually see) at the goal position to evaluate their performance face to face. Then we could follow the same idea to choose the measurement parameters for the system.

Since we talk about the overall performance and each gesture focuses on four fingers, we use the minimal α, β, γ at the goal position and then calculate the average value of four fingers as the overall performance measurement parameter. Our assumption is that the value α, β, γ detected by the Leap Motion could be used as the measurement parameters. Based on Table 2.1 we assume:
(1) The measurement parameter for Gesture 1 & 4 : α.
(2) The measurement parameters for Gesture 2 & 5 : β and γ.
(3) The measurement parameters for Gesture 3 & 6 : α, β and γ.

α, β, γ are mainly used for measuring patients' performance of the required gestures in the real world under human vision. However, the system uses the data collected from Leap Motion to evaluate patients' performance. Leap Motion is not always as precise as human eyes. The question is that whether α, β, γ are still suitable to be used as the measurement parameters for the system. According to data collection and long time observation, I found that some of the assumptions above did not work as we expected. The following sections provide details explanation, including the encountered problems, how to solve the problems and what is the most suitable parameter for evaluating the overall performance of each gesture.

5.1.2 Measurement parameter for Gesture 1 & 4

For Gesture 1 & 4, we chose α as the measurement parameter. Based on the collected data, I found that Leap Motion could detect this angle α very well (the normal range could be found in Section 5.3.1). We did not find any issues and the initial assumption worked as we expected. Each finger (only focus on 4 fingers) has its own minimal value of α at the goal position. I calculate the average value α of four fingers as the overall performance measurement parameter for this gesture.

5.1.3 Measurement parameter for Gesture 2 & 5

Initially, we chose β and γ as the measurement parameters. According to Table 2.1, we expected that $\beta < 90°$ and γ around $110°$. However, it did not work as we expected and Leap Motion has limited range issue for Gesture 2 & 5.

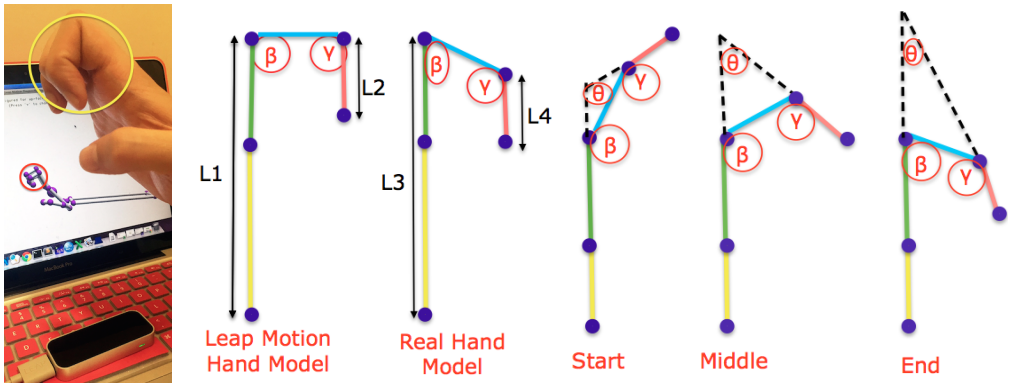
Limited range issue:

I found that among more than 100 healthy hands. No one could reaching the goal which makes $\beta < 90°$. The majority of people could reach almost $90°$ for β. For Gesture 2 & 5, β is a very important parameter to evaluate users' performance. The doctors hope that

patients could make $\beta < 90°$. With this limit range issues of Leap Motion, it is not very accurate to evaluate the performance of this gesture by using β directly.

Problem solving:

Based on long time observation and study, I found why the β could never less than 90° and how to evaluate this gesture in a better way.



(a) (b) (c)

Figure 5.1: Extra parameter for Gesture 2 & 5

Figure 5.1(a) shows an interaction between the real user's hand and Leap Motion. The red circle in Figure 5.1(a) is the hand detected by Leap Motion which is consistent with Leap Motion API. The yellow circle is the real user's hand. Comparing with the red and yellow circle, We could find the limitation of Leap Motion clearly. The detected result by the Leap Motion could not correctly reflect the real user hand's situation.

Leap Motion has built in its own hand model which could be abstracted as the left geometrical finger shape on Figure 5.1(b), however, the real people's hands are in the shape of the right geometrical finger on Figure 5.1(b). Therefore, the individual value of β & γ could not perfectly reflect the actual angles. Our initial assumption that selecting β & γ as the measurement parameter for Gesture 2 & 5 is not a good choice.

However, I found that the sum of β & γ could be used for the gesture evaluation. Because, for a perfect gesture in Figure 5.1(b), L1, L2 and L3, L4 should almost be parallel. Therefore, the sum of β & γ is the same (almost 180°) for both situations. The sum of β & γ could be used for a better evaluation component than using them individually for this gesture. The angle θ in Figure 5.1(c) could be calculated from the sum of β & γ by using the features of the triangle, θ = 180 - (360 - (β + γ)). Theoretically from Figure 5.1(c), the value of θ should between 180 to 0 (Start - Middel - End in Figure 5.1(c)). The angle θ almost covers all situations of this gesture, which is the most suitable evaluation component for Gesture 2 & 5. Each finger (only focus on 4 fingers) has its own minimal value of θ at the goal position. I calculate the average value θ of four fingers as the overall performance measurement parameter for this gesture.

5.1.4 Measurement parameter for Gesture 3 & 6

Initially, we chose α, β and γ as the measurement parameters. However, our assumption did not work as we expected and Leap Motion has self-occlusion issue for Gesture 3 & 6.

Self-occlusion issue:

Gesture 3 & 6 could be considered as making a fist. When we make a perfect fist, finger tips are hidden in the palm, which leads to an unstable and inaccurate result. The values of α, β and γ are not guaranteed to be the actual angles that users achieved. If we evaluate Gesture 3 & 6 by using the values of α, β and γ directly, the result is not convincing enough. Besides, it is not a good approach to evaluate the overall performance using multiple parameters. For this gesture, each finger has its own minimal values of α, β and γ in the goal position. We are required to focus on 4 fingers' performance. There are 12 values should be considered in total for this gesture. Based on two reasons mentioned above, the initial assumption is not the best choice and it is necessary to find other measurement parameters which could better reflect Gesture 3 & 6.

Problem solving:

The hardware limitation of Leap Motion could not be solved in short time. With many attempts of different parameters and various combination and calculation, I found a better component to evaluate this gesture.

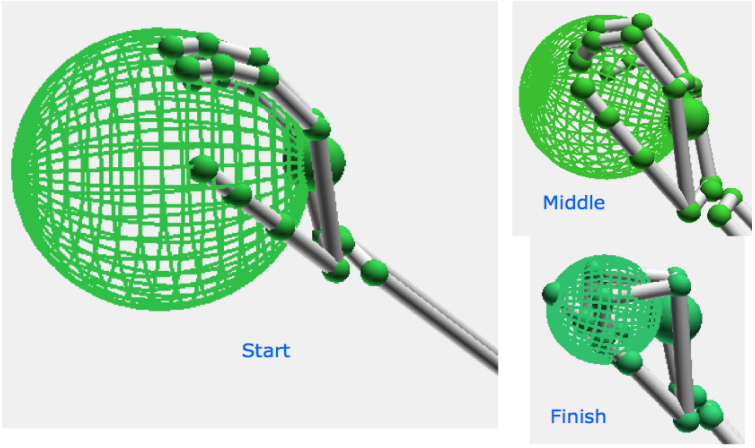


Figure 5.2: Extra parameter for Gesture 3 & 6

In Leap Motion API, there are two parameters could possibly used for evaluating Gesture 3 & 6. One is `grabStrength`, the other is `sphereRadius` defined in Section 2.2.2. Gesture 3 & 6 require users making a fist which contains the process of hand open and close. Therefore, `grabStrength` could be used. Besides, the sphere well fits the curve of the hand, and the `sphereRadius` parameter reflects the degree of finger motion. Therefore, it could also be considered.

Based on the observation and collected data of `grabStrength` parameter, I found that all healthy volunteers could achieve 1 (1 is the upper limit which means Leap Motion detects that the user's hand is closed mentioned in section 2.2.2). Furthermore, this parameter is very sensitive, it could reach to 1 before users making a complete fist. Therefore, it could not differentiate a perfect or good gesture. For example, the `grabStrength` for Figure 5.2 (Figure 5.2 was generated by using the Leap Motion Diagnostic Visualizer tool[20]) Middle and Finish gesture are all 1. The `grabStrength` is not a suitable measurement parameter for this gesture.

However, `sphereRadius` works extraordinarily well. The whole motion of this gesture could be considered as user holds a ball from a large size such as Figure 5.2 Start gesture

to a small size ball such as Figure 5.2 Finish gesture. Based on the observation of healthy volunteers, everyone has a minimal size of the sphere, when they make a complete fist (all fingers touch the palm). Since `sphereRadius` reflects the size of the sphere. I decided to use `sphereRadius` to evaluate this gesture. The minimal value of `sphereRadius` is also the limitation of Leap Motion could detect for Gesture 3 & 6.

Based on the collected data, I found that although all volunteers performed perfectly to complete the exercise, the minimal value of `sphereRadius` are not the same. The reason is that, the sizes of our hands are slightly different. Since the sphere fits the curve of the users' hand [13], the value of `sphereRadius` will be affected by both factors, the hand pose and the hand size as well. I attempted many parameters combinations to find out how to quantify the minimal value of `sphereRadius` for different hands. Finally, I found that, the minimal value of `sphereRadius` for a perfect gesture could be estimated by the width of the palm, which approximately be the distance between the index and little finger. The `palmWidth` and minimal `sphereRadius` has linear relation and the line y = x is used to approximate that relation shown in Figure 5.3. For a perfect gesture, the minimal value of `sphereRadius` is expected to be similar to the value of `palmWidth`. I finally chose the difference(D) between `sphereRadius` and `palmWidth` (D = `sphereRadius` - `palmWidth`) to be the performance measurement parameter for Gesture 3 & 6. The difference(D) means how far the patients could reach the goal (estimated by `palmWidth`).

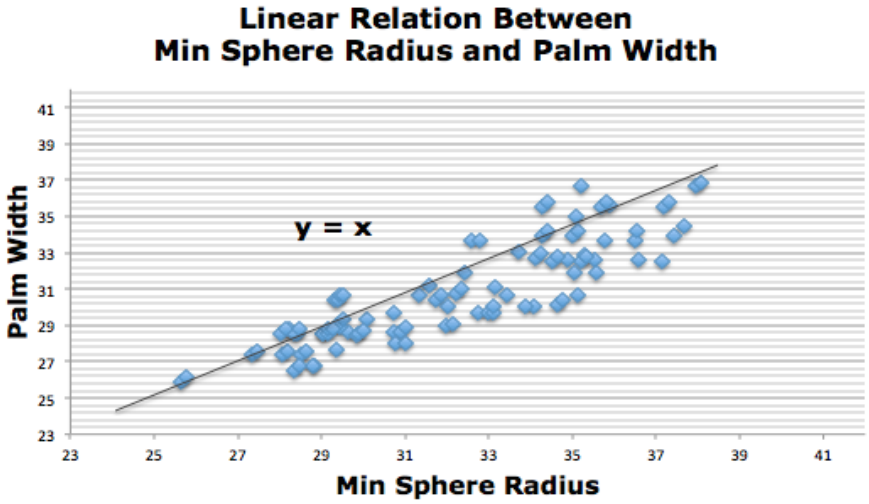


Figure 5.3: Linear relation between minimal sphere radius and palm width (millimeters)

5.1.5 Measurement parameter selection summary

According to section 5.1.2, 5.1.3 and 5.1.4, we found out what is the measurement parameter for each exercise except the common parameter `time`. Table 5.1 is the summary of the final selection. We use the minimal value of each measurement parameter achieved by patients in the goal position to evaluate their overall performances.

Gesture 1 & 4:	Average α of 4 fingers
Gesture 2 & 5:	Average θ of 4 fingers
Gesture 3 & 6:	The difference(D) between `sphereRadius` and `palmWidth`

Table 5.1: Summary of the measurement parameter for each exercise

5.2 Data collection approach

Section 5.1 discusses the whole process of selecting most suitable measurement parameters for each exercise and provides a conclusion on Section 5.1.5. This section explains the data collection approach we used to quantify the normal range of selected measurement parameters. In order to find the normal range, I aimed to make a substantial data collection. With limited time, I asked 60 healthy people put their hands over the Leap Motion sensor and doing the required exercises correctly by using the system introduced in Chapter 3 and 4. The collected data from the system was used in Section 5.3.

Why choose healthy people not patients?

Considering the diversity, 60 volunteers come from different background, ages, countries and occupations (Appendix B shows several selected screen shots for data collection). They are healthy people. Healthy people in this project mean that they do not have injured fingers, their hands are fully functional, easily bend their joints without any issues. They could achieve perfect performance for each required exercise in the data collection process.

There are three reasons for choosing healthy people for data collection. Firstly, compared with patients, healthy people without any injured fingers are able to bend their finger joints in a larger range. This could help us to find out what is the normal range of hand motion detected by Leap Motion. Secondly, to have a deep understanding of the variation and similarity of different hands detected by Leap Motion, we need to collect data from many different control subjects. In the short term, this goal could be better and easily achieved by collecting data from healthy people. Moreover, it is crucial to find out or give a definition of 'good performance', this needs to do data analysis from healthy controls in order to find out what healthy people could achieve. Therefore, I choose to collect data from healthy people.

Data collection process for each volunteer:

The whole data collection process could be divided into 6 steps below. During the whole process, I did the video recording of the user's hand motion and wrote down useful information I found.

- Step 1: Explain the aim of the project to the user.
- Step 2: Register the user information such as name, ages etc.
- Step 3: Let the user see the video instruction about how to use the system.
- Step 4: Let the user access a new exercise.
- Step 5: Let the user see the corresponding gesture performance instruction first and then ask the user to follow the movement of the animation instruction to complete the exercise correctly.
- Step 6: Go back to Step 5 until there are no new exercises.

If a user is successfully registered, the system automatically sets up standard exercises. The user has the options to do the standard exercises for both hands or only one hand. There are two types of standard exercises, the first one requires each dynamic gesture repeats 3 times, the second one only needs to do once. User options completely depend on the time they have. On average, the whole process needs 30 minutes (If the user chooses to do exercise for both hands, and each gesture repeats 3 times). We considered that one

person's right and left hand are different. Since, most of the people doing exercises by using both hands. In all, the data I collected for section 5.3 came from 60 healthy people with more than 100 different hands.

5.3 Normal range of selected parameters

This normal range of each selected parameter is very important. Because it helps us understand what a good or normal performance should be using Leap Motion sensor. The normal range represents a healthy standard, which is a clear goal for patients to achieve. The following sections talk about the normal range we found for each exercise based on the collected data from Section 5.2.

5.3.1 Normal range for Gesture 1 & 4

Based on Section 5.1.5, we used the minimal value of α which users could achieve to measurement their overall performance for Gesture 1 & 4. Figure 5.4 is generated by using SPSS Statistics[7] which is a statistical analysis tool. It shows the frequency distribution of the minimum α based on the collected data from all healthy volunteers who had perfect performance in the data collection process (more than 100 different hands, one hand contains four α achieved by four fingers).

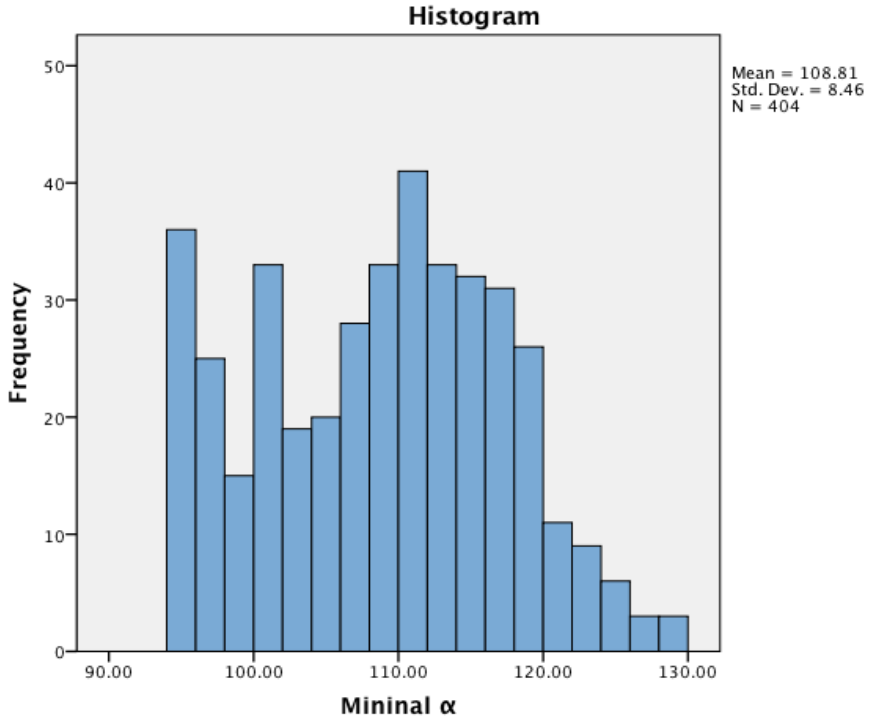


Figure 5.4: Frequency distribution of the minimal α

Based on Figure 5.4 we found that the normal range of α is between 94.65° and 129.91°. One average, healthy people could achieve 108.81°.

5.3.2 Normal range for Gesture 2 & 5

Based on Section 5.1.5, we used the minimal value of θ which users could achieve to measure their performance for Gesture 2 & 5. Figure 5.5 (generated by using SPSS Statistics[7]) shows the frequency distribution of the minimal θ based on the collected data from healthy people with perfect performance in the data collection process. The data came from more than 100 different hands, one hand contains four θ achieved by four fingers.

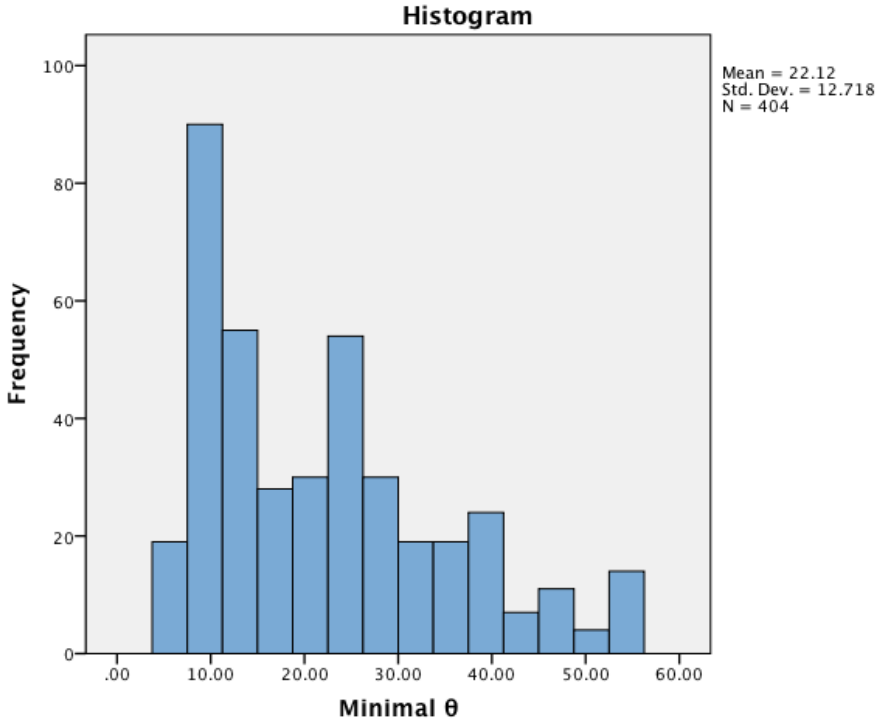


Figure 5.5: Frequency distribution of the minimal θ

From Figure 5.5, the normal range of θ is between 5.21° and 55.54°. According to the observation, healthy volunteers performed differently for this exercise. Although, their performances are all good, visually, some of them have a larger range to bend their joints than other volunteers. The detected result from Leap Motion on Figure 5.5 tells us, on average, healthy controls could reach 22.12°.

5.3.3 Normal range for Gesture 3 & 6

Based on Section 5.1.5, we used the minimal value D (difference between `sphereRadius` and `palmWidth`) which users could achieve to measure their performance for Gesture 3 & 6. Figure 5.6 (generated by using SPSS Statistics[7]) shows the frequency distribution of the minimal difference between `sphereRadius` and `palmWidth` based on the collected data from healthy people (more than 100 different hands) with perfect performance in the data collection process.

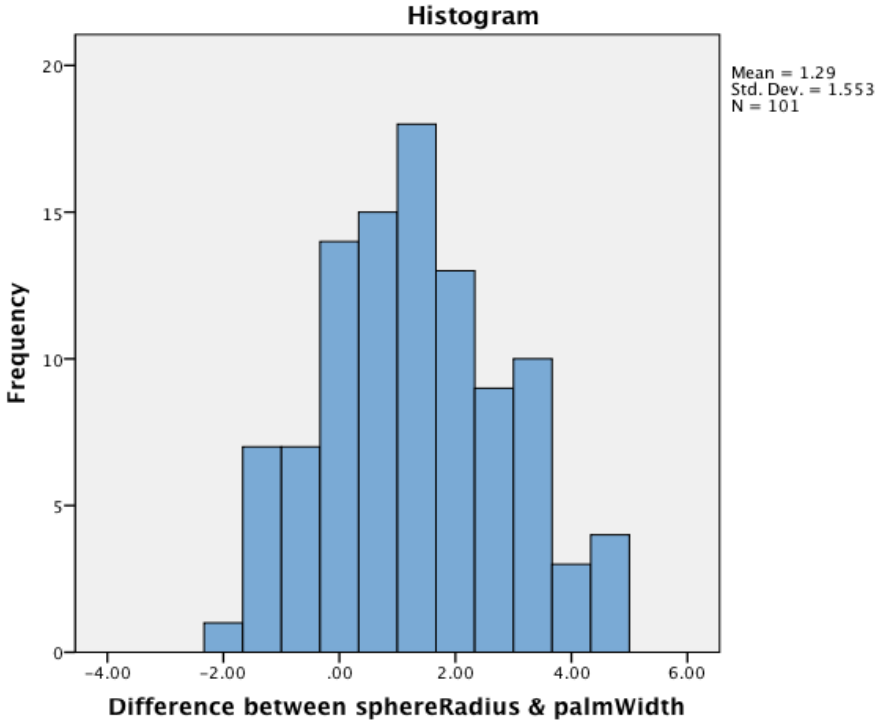


Figure 5.6: Frequency distribution of minimal difference

The normal range of minimal difference(D) is approximately between -2.32 to 4.61 millimeter. The difference for the majority of healthy people detected by the Leap Motion is 1.29. For a perfect performance of Gesture 3 & 6, the minimal value of `sphereRadius` that the user achieved at the goal position should be around the value of the user's `palmWidth`.

Chapter 6

Test and evaluation

The chapter shows the test and evaluation of the Hand Physiotherapy system. Section 6.1 talks about the functionality examination, which briefly summarizes how the system meet the functional requirements mentioned in Section 1.2.2 and how the system react in some special situations. Section 6.2 shows the feedback accuracy evaluation. The system limitation is discussed in Section 6.3. Section 6.4 presents some feedbacks from the participants about how they felt when they used the system.

6.1 Functionality examination

In Section 1.2.2, there are seven functional requirements. The whole system was implemented according to them. It is important to test whether the system satisfies all functional requirements. This section firstly shows the test result and then explains how system handle special situations.

6.1.1 Functional test result

The table 6.1 summarizes the test result by comparing the expected behavior and how the system really behaves. The last column of the table tells whether the system passes the test.

Test ID	Functional Requirements	System Behavior	Result
1	The system shall enable users register and allow registered users access the system.	1. Users could register their information through a Register Page. 2. Users could login into the system by entering the username and password through a Login Page.	Pass
2	The system shall enable the user to view a list of their assigned hand exercises.	If the user could login to the system successfully, then the user could access the Home Page which lists all assigned exercises for the user ordered by date. Users could choose to fold or unfold the list elements to check the detail exercise information.	Pass

3	The system shall enable the user to access three different specific required exercises for both hands.	Each required exercise for both hands is labeled with different name (Gesture 1 to Gesture 6). Users click the name, they could go to the corresponding Exercise Assistant Page.	Pass
4	The system shall enable the user to know how to operate and start a hand exercise.	Users could watch the instructional video on the Home page to learn how to use the system.	Pass
5	The system shall guide the user to complete hand exercise correctly.	1. In the Exercise Assistant Page, users could watch the gesture introduction video. 2. Users could also follow word instructions to activate an animation instruction (Virtual Hand). 3. Users could visualize their detected hand (called Real Hand) by Leap Motion. 4. Users could move their hands over the Leap Motion sensor, and make the Real Hand follow the same movement of the Virtual Hand. 5. The Real Hand's color will be changed when the Real Hand turns green, users move their hands, if they finished one movement, they could hold their hands for continuous 0.5 second and the system will lead you to the next step.	Pass
6	The system shall store user performance data collected from Leap Motion.	Users could click the finish button in the Exercise Assistant Page, all hand performance data will be stored into a particular folder created for the user.	Pass
7	The system shall enable the user to get their performance results for each completed exercise.	In the Home Page, next to each gesture name, there is one element called 'see feedback'. Users could click this element and the corresponding performance result will be shown on the Feedback Page.	Pass

Table 6.1: Functionality examination table

6.1.2 Special case handle test

Table 6.1 mainly discusses the normal situation. However, there are some exceptional situation could happen. It is important to test the system could work smoothly and be able to handle the special situation. The key special cases may happen in the system are listed below.

1. If a user login in with an invalid username or password, the system will show 'invalid user'.
2. In the process of a user doing a hand exercise, if the user' hand does not put over the Leap Motion sensor. The system will notice the user 'No Hand detected, Please retry'.
3. If the user click the 'Finish' button before completing the exercise, the system will notice the user 'not finish'.
4. If a user attempts to check the feedback result of an incomplete exercise, the system will tell the user 'There is no feedbacks for this exercise so far and please finish the exercise first'.

The system could be able to handle special situations and notice the user that why it is not working.

6.2 Feedback accuracy evaluation

The feedback accuracy reflects in 3 aspects shown below.

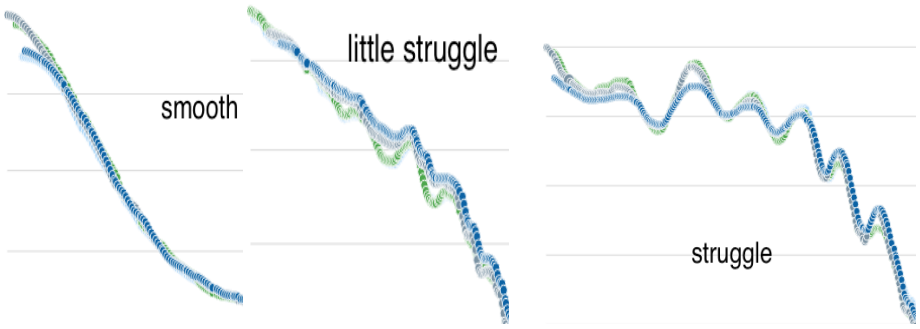
The first one is the correctness of the boundary value we get for each gesture measurement parameter. We wonder whether the boundary value we get could actually reflect the user's real performance over the sensor. For example, for a completed performance of Gesture 1 or Gesture 4, the feedback (Figure A.1 in Appendix A is an example) provided by the system includes the minimal value of bend angle α. We want to know whether the minimal value we get from the system is consistent with what we actually see. The 'consistent' does not mean exactly same in this project. Because, there are many factors could affect the result. The first one is the Leap Motion detection limitation (The sensor is unlikely to be 100% accuracy). The second factor is the human vision error. We look at the user's hand performing Gesture 1. It is unlikely to tell the exact bend angle we see. Therefore, we roughly compare the value we see and the value we get from the system for each test user. In this project, the 'consistent' means we accept the result under a certain allowance.

Figure 6.1: One example of testing different bend angles

I tried to bend the joints to different levels (simulating different situations) one example could be Figure 6.1. I tested all gestures feedbacks. The results we got from the system were consistent with what we saw. I also showed the feedback system to the orthopaedic hand surgeon Associate Professor Dominic Furniss who designed the required gestures of this system, he was happy with the result.

The second one is the correctness of the gesture performance time we get from the feedback result. Section 3.5.2 introduces the extraction algorithm which aims to remove all invalid data. Because, in the collected data set, it contains the real hand movement data and also the signal or reaction data. We only want to extract the hand movement data in order to find out how long it really takes for that movement. I went through each test recording video and used the stopwatch to manually count users' real movement times and compare with the system results. I found that, it takes around 3.5 seconds to complete one exercise on average and the error is around 0.15 milliseconds which is small enough and would not affect the final results.

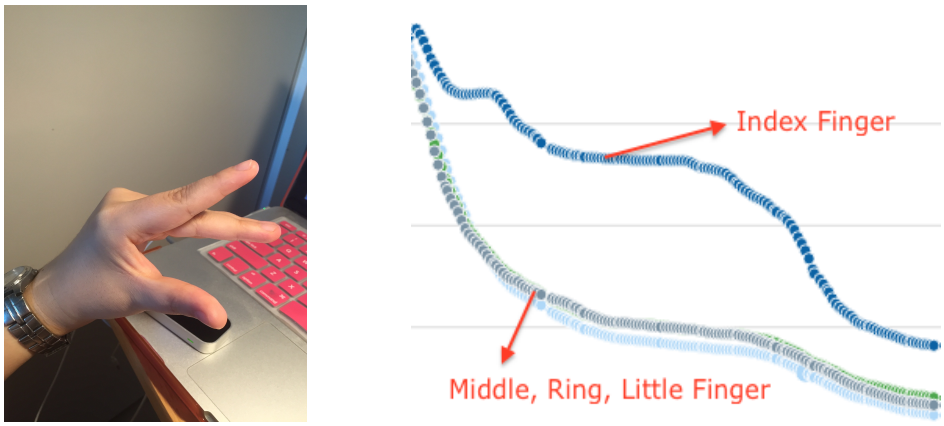
Inside feedback page, the system also plots the performance of the whole hand motion using line chart. The third evaluation aspect is that whether the graph could reflect the whole dynamic hand movement. In order to test this, we simulated different situations, one example could be the Figure 6.2. It shows the bend angle changes for four fingers along the timeline (each finger has one unique color). The Figure 6.2(a) represent a very smooth hand motion. The Figure 6.2(b), we simulated to perform a little struggle situation (if users feel painful, they could not move their hands as smooth as healthy people, they may have a some small range fluctuation during the whole process). The Figure 6.2(c), we simulated a struggle situation, users may try to bend the joints up and down in order to reach the goal position. The graph contains a larger range of fluctuation. Based on the graph, we found out that the feedback line chart could provide us useful information and reflect the dynamic gesture process (smooth or maybe struggle a little bit etc).



(a) (b) (c)

Figure 6.2: Graph test example

In addition, we also simulated another special situation that one particular finger could not bend as much as other fingers. Figure 6.3(a) is an example, the index finger could not bend any further. The corresponding line chart perfectly reflects this situation on figure 6.3(b), the blue line for index finger is clearly different with other lines (blue line performed worse).



(a) (b)

Figure 6.3: One special graph test example

6.3 System limitation

The system has interaction with Leap Motion in order to track users' hand motion. When users put their hands over the Leap Motion sensor, the system provides their hands' visualization detected by Leap Motion. Most of the time, the detected hand could correctly reflect real users' hand situation over the sensor. However, according to the observation, there are a few cases show that the Leap Motion could not always correctly detected the real hand pose. Therefore, the limitation discusses in this section includes the limitation of the Leap Motion which restricts the functionality of the system (affect final feedback result) and the limitation of the system itself.

1. The first limitation comes from Leap Motion. The user bended fingers joints, however the detected hand showed that the user's fingers were straight. This showed an inconsistent situation between the detected result and real performance. This situation may cause by many reasons such as the user's hand was not located in a good detected position, the surface of the Leap Motion is not clean enough which affects the use of two Leap Motion cameras and the occlusion issues may happened etc. The system provides feedback based on the detected data from Leap Motion. In this case, the system could not differentiate that whether the user could not bend their joints or the result we got was caused by the detected issues of the Leap Motion. The system fails to give an accurate result under this situation.

2. Another limitation from Leap Motion is the delay issues. If users do the hand motion very fast over the Leap Motion sensor. Most of the time, the detect hand could reflect the hand motion with the same speed of how users performed. However, there were still a few cases that one or more fingers moved obvious more slowly than the real situation. For this delay situation, by analyzing the collected data from Leap Motion, the system could not differentiate whether users moved slowly or the delay issues happened. The system fails to give an accurate result under this situation.

3. This limitation comes from the system itself. In Section 3.4.2, I explained that the system and users are interacting with each other using a signal which needs users hold their hands for a continuous period. We could consider the signal as a static pose (static: tip velocity of the fingers should less than a fix value). If the system detects the signal, it provides instructions to guide users what they should do next. However, for some of the users who are shaking or have trembling hands (if their

tip velocity in a shaking state, does not meet a static pose requirement), the system could not detect their signals. In this situation, the system are not able to work properly.

For the first two limitations, if users find that the detected result and the real situation are totally different, users are recommended to redo the exercise and trying to find a better detection position.

6.4 User experience

This section summarizes some comments from all participants who tried the system so far.

6.4.1 Normal users

People feel fun to use the system, in general, there is no issues for most of the people get used to it very fast, especially for the adolescence and young adult. However, for the people with over 60 or the people do not use new technology very often, they tend to feel a little nervous at the beginning, but still could learn how to use it quickly, which is also shown that the system is likely to be accepted by the public. After using the system, a few healthy people say they maybe need the system to practice if their hands are overused in the real life, and people are very supported to apply sensor into the Hand Physiotherapy. There are also few negative feedbacks such as some detection issues mentioned in section 6.3 (point 1 and 2).

6.4.2 Doctor

It is crucial to get some feedbacks from the doctor. Since in the future, the system feedback data should be useful for doctors to evaluate patients' recovery situation. Besides, this project had a cooperative relationship with the an orthopaedic hand surgeon, and a post-doctoral researcher based at the Nuffield Orthopaedic Centre. Therefore, I also collected some feedbacks from the orthopaedic hand surgeon Associate Professor Dominic Furniss. He said: "In terms of the original brief for the initial development of the Leap Motion device to measure hand function and motivate patients for physiotherapy post injury or surgery, the software works well. We were particularly impressed with the innovative solutions to give a single measurement for composite functions (Exercise 2 and Exercise 3). The data that is generated by the software is useful and clinical relevant. We are very excited to continue with this project in the future, and we think that it will produce a product that can be used in the clinical setting".

Chapter 7

Conclusion

This chapter firstly summarizes the achievement of the project on Section 7.1. Section 7.2 talks about the person reflection including the challenges I met during the project, the connection to the taught course and what I learned from this project. The future work is pointed out at the end of this chapter.

7.1 Achievement

As it shows in the previous chapter, the project has achieved the initial goal we set. The completed works listed below:

1. A web-based hand physiotherapy system using Leap Motion was successfully created, which meets all functional requirements. The system has a variety of useful functionalities. The system could provide 3D hand visualization, allow users access hand exercises, assist users doing exercises correctly through video and animation instructions, evaluate and show performance feedbacks to patients. Besides, the interface is user friendly, the layout of the interface is clear and well structured. The system keeps providing word instructions (such as ready, go, well done) to guide patients using the system smoothly and being motivated and encouraged, which is important for the recovery system.

2. The project presents some issues I found while analyzing the required dynamic gestures and also proposed possible solutions to solve them on Section 5.1. This could be potentially helping people who will analyze similar dynamic gestures using Leap Motion in the future.

3. The project proposed the algorithm for extracting valid data among collected raw data, the idea of the algorithm is generally suitable for many similar analysis with the value of one parameter changing follows a particular pattern.

4. The project found the most suitable measurement parameters for required gestures and quantified the normal range of these measurement parameters shown in Section 5.3, including the normal range of α for Gesture 1 & 4, θ for Gesture 2 & 5 and the difference (D) between sphereRadius and palmWidth for Gesture 3 & 6. The result collected from over 100 different healthy hands and it is important to define a normal performance, as a healthy standard for comparing the patients' performance.

Overall the project was a successfully completed.

7.2 Personal reflection

Challenges:
During the whole development process, there were three main challenges. The first one was making the system exactly matching doctors' expectation in short time. Since, the main goal of this system was to provide a tool which assists patients doing exercises correctly using Leap Motion. This needed me to get familiar with Leap Motion API quickly to understand what we could achieve from Leap Motion. It also needed me to deeply understand the real procedure that how doctors guide their patients and the detail information of each required recovery exercise in order to better reflect the real procedure into the system. It was difficult to deliver the tool which perfectly matching all requirements directly. I started with a very basic version and keep modified based on the feedbacks I got in a series of meetings with clinicians from the Nuffield Orthopaedic Hospital. This took me an amount of time and finally the system exactly matched what doctors' expected.

The second challenge came from the data analysis process. I gradually found many limitations of Leap Motion to accurately reflect some of the required gestures. I spent a lot time to try different measurement parameter combinations and finally found the most suitable measurement parameters for each required exercise. Although, it may not be the best one (still need to be tested in a long time). It provided a solution which is very near to the perfect answer and it is also the best option we could get so far. The third challenge was the algorithm design for manipulating the raw data to get the result we want, this needed many attempts to deliver the final version of the algorithm in order to handle different situations very well.

Connection to the taught courses:
All the knowledge I learned from the taught courses in Michaelmas and Hilary Term helped me completed the project. For example, the technique for converting 3 joints position (in 3 dimensional coordinate system) to bend angle through dot product was learned in the Computer Animation course. Additionally, I applied a large amount of knowledge I learned in the Visual Analytics course. I learned the basic knowledge about HTML, CSS, JavaScript in the lecture and those knowledge are important to help me implement the web-based system. I also learned how to implement data plotting tool and show its visualization on the web. The system Feedback Page provides the visualization (line chart) of the collected data which reflects the whole process of hand motion. Besides, the knowledge I learned from the other courses provided me various of thinking aspects which supported me solving all technique challenges I met through the project.

What I learned:
I gained experience of web-based system development. I found it is difficult to meet all requirements, especially the requirements may be modified during the whole development process. From this experience, I could quickly and accurately deliver what users expected next time. I also learned how to apply the new knowledge and problem solving techniques I learned in the taught courses to solve real issues I met during the project. I felt that my problem solving skills improved dramatically. Besides, I also learned from my supervisor Dr. Irina Voiculescu that it is important to be confident with your ideas, our potential beyond our expectation.

Overall, I am happy with the outcome of the project, that achieved all goals we set up at the beginning of the Trinity Term.

7.3 Future work

The future work of the system listed below.

1. We could long time track several real patients' recovery process of using the system and find out how well the system performed to help patients completely recover and anything we could modify to assist them better. Besides, we could also widely collect data from real patients to quantify the range of their hand motion.

2. The system could add more functions in the future. The system will provide the interface for recording and analyzing more gestures (other hand recovery exercises). The system will provide different ways to give feedbacks to patients based on the situation of each patient. For example, for the patients who injured both hand fingers, the system could use the normal range distribution result in Chapter 5 to check whether the patients gradually recovering from a bad situation to a normal situation. For the people who injured one hand, we could use the healthy hand data as a healthy standard to evaluate the progress of the unhealthy hand.

3. This project mainly focused on the functionality of the patient interface. The future work could be designing and implementing the interface for doctors. The interface could allow doctors to set up exercises to patients, view their patients' exercise results stored in the patient performance result folder, and send their diagnosis to patients. Furthermore, the doctors should only have the permissions to access its own patients' information and more importantly, the security issues should be considered.

Bibliography

[1] LLC Balsamiq Studios. Balsamiq Mockups . https://balsamiq.com/. [Online; Accessed 23-August-2015].

[2] Reg Borenstein. Fold down list. https://gist.github.com/atduskgreg/5432907. [Online; Accessed 23-April-2015].

[3] Stephen Cameron. Computer Animation Vectors & Transformations. University Lecture, 2014.

[4] Health & Social Care Information Centre. Hospital Episode Statistics. http://www.hscic.gov.uk/hes. [Online; Accessed 03-Augest-2015].

[5] Bebbington, Emily and Furniss, Dominic. Linear regression analysis of Hospital Episode Statistics predicts a large increase in demand for elective hand surgery in England. *Journal of Plastic, Reconstructive & Aesthetic Surgery*, 68(2):243–251, 2015.

[6] Mozilla Foundation. WebGL . https://www.khronos.org/webgl/. [Online; Accessed 23-July-2015].

[7] IBM. SPSS Statistics. http://www-01.ibm.com/software/analytics/spss/products/statistics/. [Online; Accessed 22-August-2015].

[8] Microsoft. Kinect for Windows. https://www.microsoft.com/en-us/kinectforwindows/. [Online; Accessed 20-August-2015].

[9] Morris.js. Line & Area Charts. http://morrisjs.github.io/morris.js/lines.html. [Online; Accessed 23-April-2015].

[10] LEAP MOTION. API Overview. https://developer.leapmotion.com/documentation/javascript/devguide/Leap_Overview.html. [Online; Accessed 23-July-2015].

[11] LEAP MOTION. Finger. https://developer.leapmotion.com/documentation/javascript/api/Leap.Finger.html. [Online; Accessed 23-July-2015].

[12] LEAP MOTION. Frame. https://developer.leapmotion.com/documentation/javascript/api/Leap.Frame.html. [Online; Accessed 23-July-2015].

[13] LEAP MOTION. Hand. https://developer.leapmotion.com/documentation/javascript/api/Leap.Hand.html. [Online; Accessed 23-July-2015].

[14] LEAP MOTION. Introducing the skeletal tracking model. https://developer.leapmotion.com/documentation/javascript/devguide/Intro_Skeleton_API.html. [Online; 29-August-2015].

[15] LEAP MOTION. Leap Motion for Mac and PC. https://www.leapmotion.com/product/desktop. [Online; 29-August-2015].

[16] Leap Motion. LeapJS Plugins . https://github.com/leapmotion/leapjs-plugins. [Online; Accessed 23-July-2015].

[17] LEAP MOTION. Pointable. https://developer.leapmotion.com/documentation/javascript/api/Leap.Pointable.html. [Online; Accessed 23-July-2015].

[18] LEAP MOTION. Reach into new worlds. https://www.leapmotion.com. [Online; Accessed 29-August-2015].

[19] LEAP MOTION. Record and Playback Your Hand Gestures on the Web. http://blog.leapmotion.com/record-playback-hand-gestures-web/. [Online; Accessed 23-July-2015].

[20] LEAP MOTION. Using the Diagnostic Visualizer. https://developer.leapmotion.com/documentation/cpp/supplements/Leap_Visualizer.html. [Online; 29-August-2015].

[21] Rick Parent. *Computer Animation: Algorithms and Techniques*, chapter Background Information and Techniques. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 2 edition, 2002.

[22] RobotShop. Leap Motion 3D Motion Controller. http://www.robotshop.com/en/leap-motion-3d-motion-controller.html. [Online; Accessed 22-April-2015].

[23] Bibi Saint-Pol. Bones of the human hand. https://en.wikipedia.org/wiki/Hand#/media/File:Scheme_human_hand_bones-en.svg. [Online; Accessed 23-July-2015].

[24] Oxford University Hospitals NHS Trust. Nuffield Orthopaedic Centre. http://www.ouh.nhs.uk/hospitals/noc/default.aspx. [Online; Accessed 03-Augest-2015].

[25] Ricardo Cabello , Branislav Ulicny and other contributors. three.js. http://threejs.org/. [Online; Accessed 23-July-2015].

[26] W3Schools.com. Bootstrap Grid System. http://www.w3schools.com/bootstrap/bootstrap_grid_system.asp. [Online; Accessed 23-April-2015].

[27] W3Schools.com. JSON Tutorial . http://www.w3schools.com/json/. [Online; Accessed 22-August-2015].

Appendix A

User feedback examples

Feedback for Gesture 1 & 4:

Figure A.1 and Figure A.2 are the screen shots of Gesture 1 Feedback Page (The Feedback Pages for Gesture 1 & 4 are similar, I just provide the Gesture 1 example). The Gesture 1 may contain several repeats. The example on Figure A.1 required the user repeating 3 times. The overall results are listed one by one (For Gesture 1 & 4, the detail and overall results are the same). Figure A.2 shows the graph of 3 repeats. Users could click the corresponding tab to check the performance of each repeat.

For Repeat 1:

The overall Result for Gesture 1:
Core measurement: Alpha (Healthy Hand: Min Alpha Average 108.81, Normal Range: 94.65 - 129.91)

Index Finger:	**Min Alpha: 112.69**	**Max Alpha: 166.52**
Middle Finger:	**Min Alpha: 113.36**	**Max Alpha: 177.22**
Ring Finger:	**Min Alpha: 111.72**	**Max Alpha: 176.79**
Little Finger:	**Min Alpha: 111.69**	**Max Alpha: 167.89**

Time for complete this exercise: 3145.801 ms
Average min angle achieved : 112.365 degree

For Repeat 2:

The overall Result for Gesture 1:
Core measurement: Alpha (Healthy Hand: Min Alpha Average 108.81, Normal Range: 94.65 - 129.91)

Index Finger:	**Min Alpha: 124.87**	**Max Alpha: 166.23**
Middle Finger:	**Min Alpha: 125.43**	**Max Alpha: 178.18**
Ring Finger:	**Min Alpha: 127.27**	**Max Alpha: 174.4**
Little Finger:	**Min Alpha: 127.77**	**Max Alpha: 165.88**

Time for complete this exercise: 3373.4925 ms
Average min angle achieved : 126.335 degree

For Repeat 3:

The overall Result for Gesture 1:
Core measurement: Alpha (Healthy Hand: Min Alpha Average 108.81, Normal Range: 94.65 - 129.91)

Index Finger:	**Min Alpha: 113.31**	**Max Alpha: 165.48**
Middle Finger:	**Min Alpha: 114.04**	**Max Alpha: 178.05**
Ring Finger:	**Min Alpha: 116.15**	**Max Alpha: 174.54**
Little Finger:	**Min Alpha: 117.92**	**Max Alpha: 163.71**

Time for complete this exercise: 3488.4275 ms
Average min angle achieved : 115.355 degree

Figure A.1: Screen shot of Gesture 1 Feedback Page (1)

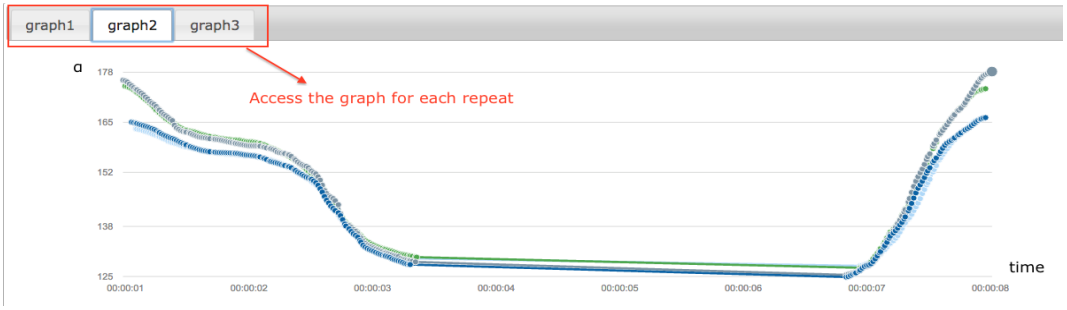


Figure A.2: Screen shot of Gesture 1 Feedback Page (2)

Feedback for Gesture 3 & 6

Figure A.3 and Figure A.4 are the screen shots of Gesture 6 Feedback Page (The Feedback Pages for Gesture 3 & 6 are similar, I just provide the Gesture 6 example). The example on Figure A.3 required the user perform this Gesture 6 only once. The overall and detail feedback results are shown in the table. Figure A.4 is the graph for representing the whole process of the user's performance.

For Repeat 1:

The overall Result for Gesture 6:

Core measurement: the different(D) between SphereRaius and palmWidth (Healthy Hand: Min D Average: 1.29, Normal Range: -2.32 - 4.61)

Min D: -0.46	**Max D: 241.49**

Time for complete this exercise: 4015.32 ms

The detail Result for Gesture 6:

Index Finger:	Min Alpha: 112.05	Max Alpha: 174.27
	Min Beta: 96.29	Max Beta: 176.06
	Min Gamma: 137.51	Max Gamma: 168.73
Middle Finger:	Min Alpha: 108.57	Max Alpha: 177.98
	Min Beta: 96.34	Max Beta: 175.64
	Min Gamma: 139.41	Max Gamma: 168.03
Ring Finger:	Min Alpha: 109.58	Max Alpha: 176.78
	Min Beta: 96.35	Max Beta: 176.01
	Min Gamma: 139.16	Max Gamma: 168.06
Little Finger:	Min Alpha: 110.92	Max Alpha: 170.82
	Min Beta: 96.37	Max Beta: 175.98
	Min Gamma: 138.18	Max Gamma: 167.7

Figure A.3: Screen shot of Gesture 6 Feedback Page (1)

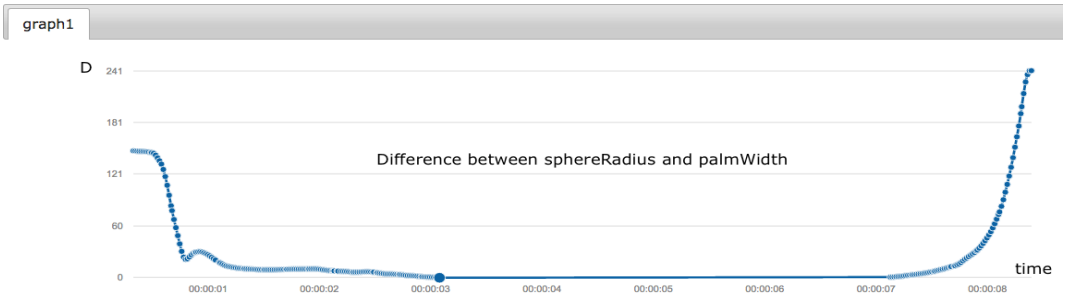


Figure A.4: Screen shot of Gesture 6 Feedback Page (2)

Appendix B

Selected screen shots for data collection

Figure B.1: Sample pictures for the data collection