Finger-to-Thumb: A Gesture-based Menu Selection Technique

Wei Lin

(Master's Program in Computer Science)

Advised by Buntarou Shizuki

Submitted to the Graduate School of
Systems and Information Engineering
in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering
at the
University of Tsukuba

March 2017

Abstract

The aim of this thesis is to investigate the development of hand gesture-based interaction, summarize the pros and cons of some representative gestural interfaces, and propose our own solution to the remaining problems in earlier researches.

We firstly introduce the concept of gesture-based menu selection techniques and how they can have a profound influence on user experience (UX). The historical development of hand gesture recognition technologies is presented, as well as the discussion of their advantages and disadvantages.

We then propose Finger-to-Thumb, a gesture-based menu selection technique. Finger-to-Thumb uses Leap Motion as the finger tracking device so that it can avoid the requirement for on-body sensors.

The Finger-to-Thumb system contains three kinds of gestures: touch gestures, swipe gestures and Touch-then-Swipe gestures. Touch gestures are the gestures that you touch your thumb to other fingers of the same hand, hence are also called Finger-to-Thumb gestures. Touch gestures are used to activate selection events. For instance, you can execute an index touch gesture to select the button mapped to the index finger. Swipe gestures are mid-air horizontal or vertical movement of the hands, which are used to switch between button panels (each button panel contains 4 buttons). Touch-then-Swipe gestures are the combination of touch gestures and swipe gestures which are executed by holding the touch postures while moving the hands. Touch-then-Swipe gestures are used to simulate slider functions, such as the volume slider in a gesture controlled music player application. We also give a detailed description of the recognition algorithm of each gesture.

The experiment part presents three experiments we conducted to investigate the performance and usability of Finger-to-Thumb. The result showed that the participants were prefer Finger-to-Thumb than the widely-used Move-and-Tap system because Finger-to-Thumb was simpler, faster, had no need for the movement of hands, and could provide tactile feedback.

The thesis then introduces two applications developed based on Finger-to-Thumb: a gesture controlled fighting game and a gesture controlled music player.

The last part of the thesis states the future work and conclusion of our research work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Augmented Reality (AR) and Virtual Reality (VR) technologies have developed very fast in recent years. Actually, several kinds of commercial products, from the full feature ones (e.g., HTC Vive, Oculus Rift, PlayStation VR) to the mobile ones (e.g., Samsung Gear VR, Freefly VR, Zeiss VR One), were already on sale there. These products are expected to bring in a brand new generation of digital entertainment experience.

Usually, AR and VR are implemented by a combination of computer technologies such as 3D display technology, sensor technology, etc. Among them, the design of input interface can also be a great challenge since our familiar input devices like mouse and keyboard are not competent now. Instead, gestural interfaces are preferred due to their more natural and intuitive way of interaction.

1.1 Menu Selection Techniques

In gesture-based interaction, menu selection techniques are the methods users rely on to interact with the menus. And menus serve to provide a means for system control [BKJLP04, CBJL09, DL10]. Therefore, menu selection techniques can have a profound impact on user experience.

Many gesture-based menu selection techniques had been researched and developed before, and most of them work in this pattern:

- Firstly, the user moves one of his/her hands in front of a camera (the system may also

require the user to simultaneously keep a pointing-like gesture [LLM13, VB05]), which consequently moves the cursor on the screen corresponding to the hand's movement.

- Then, after the user succeeded in moving the cursor to his/her desired button, he/she can confirm his/her selection (click the button) by executing a predefined gesture (e.g., releasing a previously executed pinch [GN12], or using thumb trigger gestures [GWB04, LLM13]) or just keeping that posture for a couple of seconds (using dwell time) [HAR95, WP03].
- Finally, the system handles the task assigned to that button.

We also developed a such kind of system for comparison purpose, which however, uses an mid-air tap gesture to confirm the selection, similar to introduced in [VB05] . We call this system *Move-and-Tap* in the thesis. Move-and-Tap seems to be satisfying in the aspect of intuition because its operation is easy to understand with a perceptible affordance. However, Move-and-Tap can also suffer from problems such as a long execution time (from moving to selecting), and may cause fatigue to users by the consecutive moving of arms.

1.2 Research Purpose

This thesis investigates the development of gesture-based menu selection techniques, summarizing some representative gestural interfaces with the discussion of their advantages and disadvantages. Then we propose Finger-to-Thumb: a gesture-based menu selection technique, evaluating its performance and usability by experiments.

1.3 Approach

We propose a new depth-based way to detect the Finger-to-Thumb gestures and implement it as a menu selection technique. We use Leap Motion as our finger tracking device, which can provide real-time depth data of the fingers in 3D space. Each finger except the thumb is mapped to a command built-in the system (e.g., click a button), and users can execute the command by touching his/her thumb finger with the corresponding finger. Although our prototype system and applications both work in desktop mode, we expect our approach can be used in AR and VR environments also, which becomes our future work and will be discussed in Chapter 6.

1.4 The Organization of this Thesis

In Chapter 1, we give a brief introduction to the background of research in gesture-based interaction and the definition of menu selection technique. Our research purpose and an overview of our approach are also described in this chapter.

Chapter 2 is our literature review about gestural interfaces. It contains the development of gesture-based interaction with some representative work, whose pros and cons will also be discussed.

Chapter 3 is a detailed description of our proposed approach. We firstly introduce our finger tracing device - Leap Motion about its hardware and software. Then, we introduce the Finger-to-Thumb gestures, listing all the touch, swipe and Touch-then-Swipe gestures we used in the system. Finally, is the statement of algorithm, including a comparison of the old one with the improved one.

To verify the usability and performance of our approach, we conducted three experiments which are recorded in Chapter 4. It states the purpose, participants, apparatus (setting up), procedure, result and discussion of each experiment. From this chapter we can conclude the benefit and limitations about the Finger-to-Thumb system.

Chapter 5 introduces the applications developed by the Finger-to-Thumb system. One is a simple game named Shuriken Fighting, and another is a gesture-controlled music player called Finger Player.

Chapter 6 is about our future work, especially the expectation of AR and VR applications.

Chapter 7 is the conclusion.

Chapter 2

Related Work

This chapter states our literature review on hand gesture interaction scholarly articles. We introduce the development of hand gesture recognition technologies and design concepts of gestural interfaces, in addition, we also compare and discuss their advantages and disadvantages.

2.1 Gesture-based Interaction

We human beings use hand gestures for a long history to express emotions and convey orders. Hand gestures help us interact with surroundings quickly and accurately. Besides, it's also an effective communication way for the disabled people. Human hand gestures are so simple and intuitive that even users having no technique knowledge can handle it well [CRDR11, SS15]. For this reason, in computer science field, direct use of hands as the input device has been a hot research topic for many years.

The early work on hand gesture interaction started around 1980s [Bol80]. This work, which is called "Put-that-there", combined voice and pointing gesture to interact with a large-screen graphics display surface. Users can draw shapes by speaking out the shape and pointing to a desired position to specify the drawing spot. The combination of voice and gesture greatly improved the usability of the system because the pointing gesture helps distinguish the intentional and unintentional voices, i.e, the system only recognizes the voice while it detects a pointing gesture. Besides, the point-and-speak pattern is a natural expression that is easy to understand and execute.

2.2 Hand Gesture Recognition Technologies

"Put-that-there" used position sensors to communicate the user's hand position to the computer. For the evolution of hand gesture recognition technologies, we can roughly divide it into three phases: glove-based phase, vision-based, and depth-based.

2.2.1 Glove-based Recognition Technologies

Glove-based recognition technologies use data gloves to recognize the user's hand gestures. A data glove (also called a weird glove, or cyber glove) is a glove-like input device with certain sensor units attached to capture such data like the global position/rotation, bending degree, or moving speed of the fingers and joints. These information is then mapped to different hand gestures and interpreted by a computer. The great advantage of data gloves is that the gesture recognition process is simple and direct, with no need for any kinds of pre-processing. This feature not only makes it work properly with even very limited processing power, but also improves its recognition accuracy.

The first data glove was created by Electronic Visualization Laboratory in 1977, called The Sayre Glove [SZ94]. It was able to measure finger bending with the help of photoelectric sensors [SZ94, DSD08]. Today's data gloves can provide with accurate real time 3D data of fingers and arms, and are much easier to wear. Figure 2.1 shows the P5 Glove sold by Fifth Dimension Technologies, which is capable to track hand and arm movement. However, it is inevitable that these on-body devices sometimes will be obtrusive and inconvenient.

Figure 2.1: The P5 Glove.

Bowman et al. [BWCL01, BW01] used a pair of data gloves (Fakespace Pinch Gloves™) to develop VR applications, including a menu system with which the user can select icons by touching his/her thumb to the other fingers of the same hand. Although the gestures adopted in this work are similar to our Finger-to-Thumb gestures, our implementation is not based on data gloves, which is the primary difference.

2.2.2 Vision-based Recognition Technologies

In order to achieve a more natural way of gesture interaction, researchers proposed vision-based recognition technologies, which remain the hottest topics in the field of hand gesture recognition technology. Vision-based recognition technologies usually use RGB cameras to acquire a sequence of color images, then use image processing techniques to classify gestures. They may sometimes rely on a very complex image processing architecture system including segmentation, feature extraction, and machine learning based classification. Figure 2.2 illustrates the flowchart of a typical vision-based gesture recognition system.

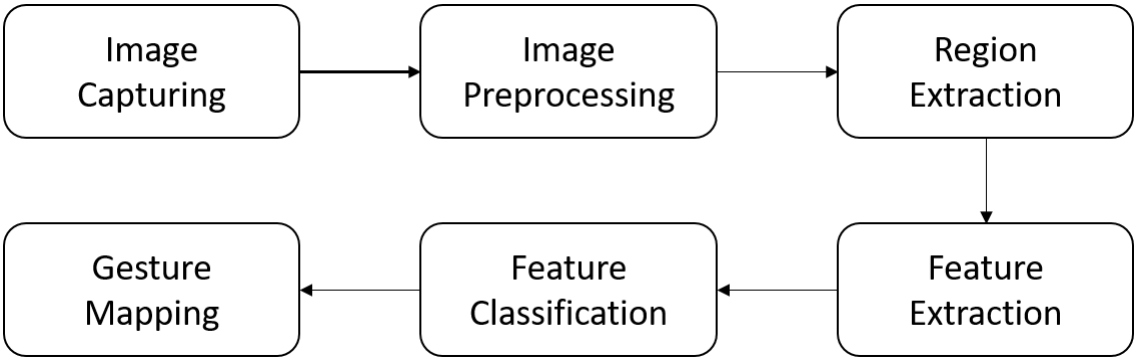


Figure 2.2: The flowchart of a typical vision-based gesture recognition system.

Vision-based recognition technologies eliminated the requirement for on-body devices, allowing users to interact with the system with their bare hands, which are more convenient and comfortable compared to glove-based recognition [LC11]. Nevertheless, there exist drawbacks in this approach too. The main problem is, however, due to its vision-based implementation, which means the performance of the system can be easily influenced by lighting conditions and it may be totally unreliable when used in a dark environment.

To the best of our knowledge, the first vision-based gesture recognition system was developed in 1980s [Pre14] by the MIT Media Laboratory to track the positions of the body and limbs for real-time computer animation [SZ94]. Nowadays, there are many different implementations of vision-based recognition from using color markers [SMT12] to complicated as Artificial Neural Networks (ANN) [TRR13].

Sasaki et al. [SKA+06] presented an input interface for AR environment which allowed the user to select icons by touching fingers of his/her one hand with the index finger of the other hand. The system displayed icons directly on the user's hand, which not merely was a good interface metaphor, but also saved time of searching buttons.

Suzuki et al. [SMT12] proposed an interaction technique for interactive surfaces called Finger-Specific-Interaction (FSI). They developed a prototype system using color markers to differentiate between multiple users as well as their different fingers. The system achieved a high number of input primitives (equal to the number of fingers used) and also was able to be operated in an eyes-free mode.

Pac-pac [FSMK10] is a tabletop entertainment system using pinch gestures, which are recognized by a ceiling camera above the table. This system allows users to shoot bullets by performing a pinch gesture towards the target. The pinch gesture recognition technique, which was earlier introduced in [Wil06], detects the ellipse surrounded by the user's thumb and index (mostly) finger to track the position of the corresponding hand. When the camera is unable to detect the enclosed ellipse that existed in previous frames, it sends a opening event indicating the hand is opened.

2.2.3 Depth-based Recognition Technologies

Recently, the invention of depth-based recognition technologies have inspired researchers to use depth-sensing devices for acquiring 3D data to track hand motions and recognize hand gestures. This trend is still speeding up thanks to the advent of low-cost depth cameras [WCL16]. Figure 2.3 is the Xtion PRO LIVE, a depth camera sold by ASUS.

An IR (infrared) projector and an IR camera constitute the essential components of a depth camera. The IR projector emits a pattern of IR light (like a starry sky of dots) which can be seen by the IR camera, then the IR camera sends this dot pattern to the depth sensor's processor, from where the depth information are worked out by analyzing the displacement of the dots.

Depth-based recognition technologies utilize depth cameras to acquire depth information of the hands, based on which, the hand parameters (position, rotation, moving speed, etc.) are calculated. In contrast to vision-based ones, few researchers relied only on depth information to develop their hand gesture recognition systems [KKKA12, KZL12, MZ11, ORS12, CWL15, LT13].

Since vision-based recognition technologies and depth-based recognition technologies both rely on image processing techniques, the work flow of these two approaches are similar that

Figure 2.3: The IR projector and IR camera on Xtion PRO LIVE.

contains steps like background subtraction, hand region extraction, hand feature extraction, etc. However, the greatest merit of depth-based recognition technologies is that they can work in various lighting conditions where the RGB cameras cannot.

Lee et al. [LT13] proposed a novel finger identification and hand gesture recognition technique with Microsoft Kinect depth data. Their gesture recognition accuracy reached over 91% and the results almost stayed the same under different lighting conditions (normal and dark). But this work could only detect 2D gestures.

2.2.4 Other Sensor-based Recognition Technologies

There are many other sensor-based recognition technologies having been researched and developed.

WristFlex [DP14] is an on-body gestural interface that can detect finger pinch gestures with the accuracy higher than 80% and the gesture execution time about 1.6 seconds. It measures pressure distribution around the wrist with an array of force sensitive resistors which is worn around that wrist. Then it uses a machine learning algorithm to classify

finger pinch gestures. The real-time performance of their system achieved an accuracy of 80.5% (SD: (±)8.7%).

We think the biggest advantage of WristFlex is its capability of wirelessness, which hugely expands its usage in real life, especially, for outdoor uses. However, in the aspect of gesture recognition, our approach is not only more accurate (> 90%), faster (0.996 sec), but also with no need for any on-body devices.

ThumbRing [TWHH16] is an input device for item selection on head-mounted displays (HMDs) and smart glasses. ThumbRing is a ring-like device with an inertial measurement unit (IMU) worn on the thumb to track finger motions. By arranging an item to a finger segment, users can touch and slide finger segments with their thumb to select the item. To resist shake in mobile conditions such as walking, another IMU is attached to the back of the hand to compute relative angles between the hand and the thumb. Sliding and touching the segments with the thumb provide privacy, subtlety, natural tactile feedback and similar input area as smartphones. Their user study showed that the mean selection time (ST) for all trials without errors was 1763.59 ms (SD = 386.89 ms), and the error rate (ER) was 16.13%.

The ThumbRing's interface has a very similar usage as ours. Moreover, it is capable of portability. However, ThumbRing requires users to wear an extra on-body device and its complex procedure of confirming a selection increases the average selection time. On the other hand, their pilot study which figured out users' feeling of most comfortable regions on fingers for touching (to select) can be a guideline for our application developing.

As we can see, though these on-body sensor based recognition technologies may obtain a high gesture recognition accuracy, their demands for wearing extra devices sometimes can be obtrusive and uncomfortable.

2.3 Design of Hand Gesture Interfaces

In addition to the gesture recognition technologies, the design of gestural interfaces also could have a deep influence on user experience (UX) of the system. Up to now, researchers have proposed and investigated many gesture sets for different applications, while the design principle is considerably the same: to be fast, high-accuracy and effortless (we focus on mid-air hand gestures in this thesis).

A recent comparative evaluation in [KL14] compared several hand gesture based selection techniques, including Finger-Count, Hand-n-Hold, Thumbs-up, as well as 3D Marking menus [RO12]. The results of their studies showed that the Finger-Count menu technique

was significantly faster than the other menu techniques and was the most preferred one by the participants. This is because the Finger-Count menu technique has a more reliable selection confirmation mechanism. And with Finger-Count menus, there is no need for users to keep moving their hands which can greatly relieve the feeling of fatigue.

The characteristic of Finger-Count gestures enlightens us to implement Finger-to-Thumb gestures as a menu selection technique as they have similar properties.

Jude et al. [JPG16] implemented and evaluated three different gestures for users to manipulate objects on a screen, which were referred to as "*grasp*", "*grab*", and "*pinch*". According to the results of their user study, the *grasp* was strongly preferred than the other two gestures, and the second one is *pinch*.

The reason why the *pinch* was less preferred is that while in their experiment, participants were asked to keep the ring and pinky finger outstretched while performing the pinch gesture, which could induce more fatigue. In our Finger-to-Thumb system, users can do pinch gestures with other non-pinching fingers stay in a natural and comfortable posture.

Huang et al. presented DigitSpace [HCY+16], a thumb-to-fingers touch interface based on magnetic tracking technique. They conducted three user studies to investigate the comfortable areas of human fingers when touched by the thumb finger. Their results provided some design considerations for thumb-to-fingers touch interfaces (such as the influence of hand anatomy and the tactile feedback between fingers).

rapMenu [RMB08], which originates from Marking Menus [Kur93, KB93] and FlowMenu [GW00], is a circle shaped menu design for hand gesture based remote control. It uses wrist rotation gestures and finger touch gestures to navigate menu items on a circular layout. One selection task can be completed in two steps:

1. The user rotate his/her wrist to activate four menu items on part of the circular layout.
2. After the four menu items are highlighted, the user can select his/her desired one by a pinch gesture (each finger except the thumb is mapped to one of the activated menu items).

While rapMenu focuses on menu layout designing to improve the efficiency of selection, our Finger-to-Thumb system even has no need to concern about menu layouts since we directly map menu items to users' fingers so that it can get rid of the influence of menu layouts.

Chapter 3

Finger-to-Thumb System

The Finger-to-Thumb system uses real time depth data of users' hands and fingers, based on which gestures are recognized by our detection algorithm. Users can use these gestures to interact with applications such as menu item selection.

3.1 Leap Motion

We use Leap Motion as our hand tracking device because it can provide us with real time 3D data of the hands. Besides, the non-wearable device can help us get rid of the inconvenience from on-body sensors. Moreover, since Leap Motion is specialized for hand gesture recognition, their detailed API documentation can help us build our applications more smoothly. Many researchers also adopted Leap Motion in their hand gesture interaction systems, such as hand writing [KU16], sign language recognizing [MK16, PAC13, SN16] and games [MT15, LWT+15].

3.1.1 Leap Motion Controller

The Leap Motion Controller, shown in Figure 3.1, is a compact motion tracking device developed and sold by Leap Motion, Inc. It is designed to be placed on a physical desktop, facing upward, and connected to a computer via USB while being used. The core parts of a Leap Motion Controller consist of two cameras and three infrared LEDs that are used to track infrared light.

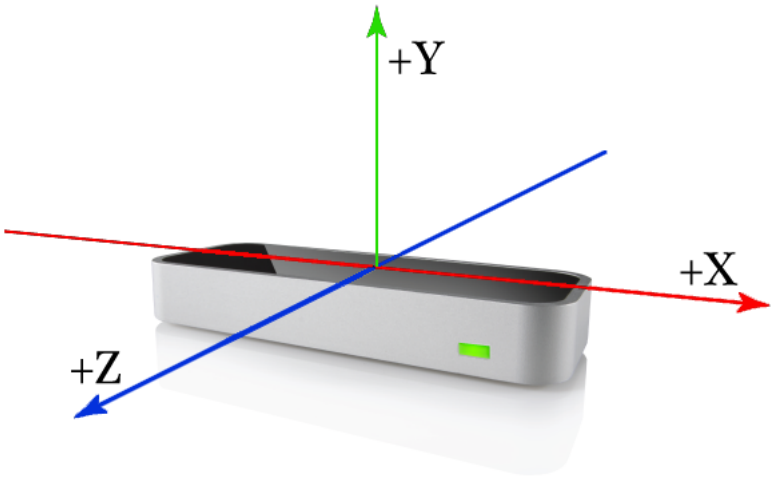


Figure 3.1: The Leap Motion controller uses a right-handed coordinate system.

Users can execute hand gestures above the Leap Motion Controller within an inverted pyramid like interaction space, whose limited range is about 60 centimeters [Col14]. The Leap Motion Controller reads the sensor data into its own local memory to perform any necessary resolution adjustments. Then, the adjusted sensor data is streamed via USB to the Leap Motion tracking software.

3.1.2 Leap Motion Software

The Leap Motion software (Leap Service) installed on our computer processes the sensor data (images) to reconstruct a 3D representation of what the device sees. After which, the tracked information such as hands and fingers are extracted. Finally, the Leap Service feeds the results, which are expressed as a series of frames, into a transport protocol, through which, our client library can retrieve the tracking data.

3.2 Gestures

The gesture set of Finger-to-Thumb system mainly but not only contains touch gestures that performed as touching your thumb to other fingers. Additionally, we defined swipe gestures and Touch-then-Swipe gestures to enrich the interaction.

3.2.1 Touch Gestures

Touch gestures are the foremost ones in our system, i.e., users mostly depend on them to interact with the system. They are respectively index touch, middle touch, ring touch and pinky touch, illustrated in Figure 3.2.

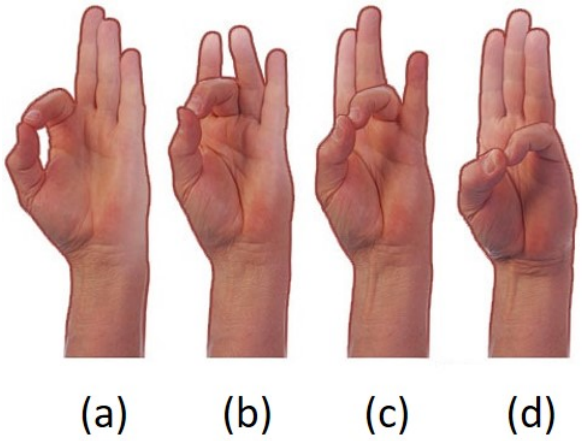


Figure 3.2: Finger-to-Thumb gestures performed by left hand, (a) index touch, (b) middle touch, (c) ring touch and (d) pinky touch.

Due to their appearance, we named them Finger-to-Thumb gestures to distinguish from general touch gestures. Actually, this kind of gestures have already been widely introduced by researchers in their gestural interfaces [BWCL01, BW01, FSMK10, DP14, TWHH16, HCY+16].

We choose Finger-to-Thumb gestures to implement the menu selection mechanism mainly due to three reasons:

- First, they are fast. Since these gestures are very simple, and are so familiar to we human beings. There is almost no need for practice before we get used to them.
- Second, touching our fingers to our thumb will produce a tactile feedback which can be an explicit cue for our selection.
- Third, because they are fast and have tactile feedback, we can accomplish eyes-free interaction, that is, we don't have to keep watching on our fingers while manipulating the system.

Users can quickly select a menu item by touching his thumb to the corresponding finger. Assume there are four buttons named A, B, C, D that are respectively mapped to the index

finger, middle finger, ring finger and pinky finger. A user can select Button A by performing an index touch gesture, i.e., touching the thumb to the index finger, as Figure 3.3 shows.

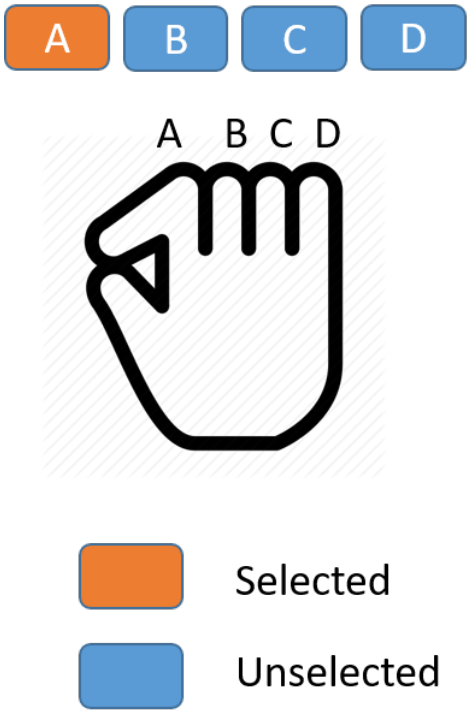
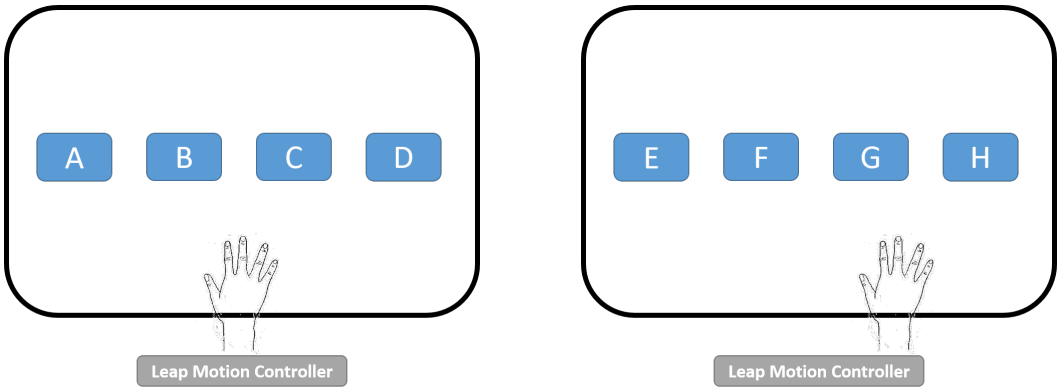


Figure 3.3: Performing index touch gesture to select Button A.

The advantages of Finger-to-Thumb gestures are much suitable for a fast, real-time selection scenario but with few menu items. Since we only have four fingers that can be touched by the thumb on one single hand, even if we used both our hands, the input vocabulary could barely increase to at most eight gestures. An evil trick is to allow multi-touch (e.g., simultaneously touch the index finger and middle finger with the thumb), such that we can have as much as 15 (2^4 - 1) gestures for one hand. Apparently, this is not a realistic solution because it requires users to remember a large number of gestures which is an absolutely terrible design.

3.2.2 Swipe Gestures

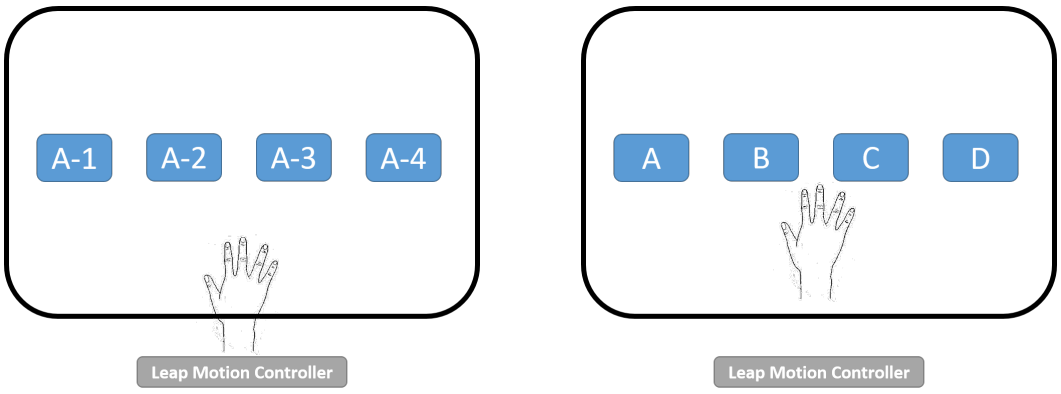
Inspired by the research of asymmetric bimanual gestures [Gui87], we came up with the idea of using gestures done by the other hand to switch activated menu items. Some researchers also have proved that bimanual interaction outperforms unimanual interactions [HPPK98, WHM12].



(a) (b)

Figure 3.4: (a): Button panel with Button A, B, C, D; (b): Using a right swipe gesture to switch to the button panel with Button E, F, G, H.

We choose swipe gestures because of their simpleness and good metaphor for switching. Users can switch button planes using left or right swipe gestures. For instance, there are totally 8 buttons (A, B, C, D, E, F, G, H) among which only 4 can be displayed simultaneously on the screen. At first, Button A, B, C, D are displayed. If the user wants to select Button E, he/she can perform a right swipe to switch to the panel with Button E, F, G, H displayed, as shown in Figure 3.4. Then an index touch gesture accomplishes the task. Further, the user can do a left swipe gesture to return to the former panel (with Button A, B, C, D) if he/she wants to select a button from it. And if there is a sub-menu under the selected menu item (e.g., Button A-1, A-2, A-3, A-4 under Button A), the user can use a up swipe gesture to go back to the parent menu, as shown in Figure 3.5.

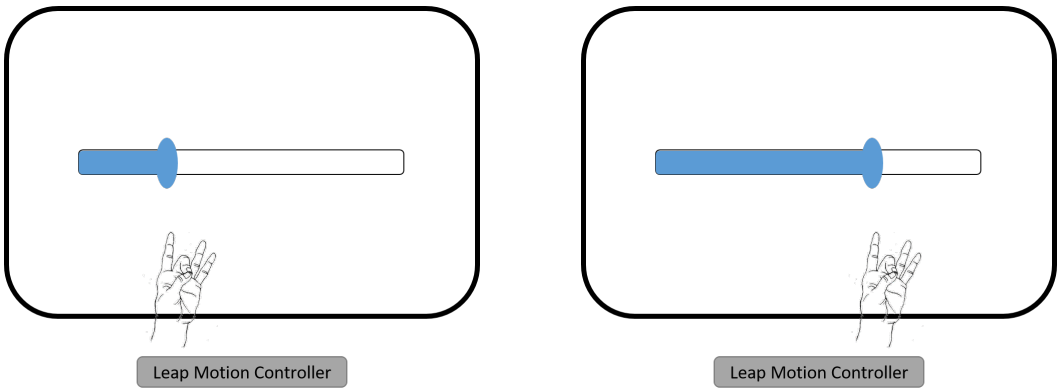


(a) (b)

Figure 3.5: (a): The sub-menu buttons under Button A; (b): Using a up swipe gesture to go back to the parent menu.

3.2.3 Touch-then-Swipe Gestures

A Touch-then-Swipe gesture is the combination of a touch gesture and a swipe gesture. That is, you keep the touch gesture and move your hand as a swipe gesture. We use Touch-then-Swipe gestures to adjust slider components in the *FingerPlayer*, which is a gestural music player application that will be introduced in Chapter 5. Figure 3.6 gives a diagrammatic sketch about its usage.



(a) (b)

Figure 3.6: A middle Touch-then-Swipe gesture from left (a) to right (b).

3.3 Gesture Detection Algorithms

We have introduced our designed gestures which can be summarized in Table 3.1 with their corresponding functions. Since Leap Motion is only able to provide us with 3D data of hands and fingers, we must develop our own algorithms to detect gestures. In this section, we describe the detection algorithm for each kind of gesture in detail.

3.3.1 Touch Gesture Detection

Actually, our touch gesture detection algorithm has experienced an improvement. The deprecated one is much simpler. It firstly calculates the Euclidean distance between the thumb and other fingers. Figure 3.7 shows this distance between the thumb and the index finger of the left hand. Then we use a threshold value for each finger to decide whether there is a touch happening or not. The threshold value for each finger was adjusted by our observation (in our prototype system developed by Unity, it was set to 30 Unity units). This approach

Table 3.1: Gestures of Finger-to-Thumb system

Gesture Types	Gestures	Functions
Touch Gestures	Index Touch	Item Selection
	Middle Touch	Item Selection
	Ring Touch	Item Selection
	Pinky Touch	Item Selection
Swipe Gestures	Right Swipe	Switch to the next Button Panel
	Left Swipe	Switch to the previous Button Panel
	Up Swipe	Switch to the parent Button Panel
Touch-then-Swipe Gestures	Right Slide	Turn up a horizontal slider bar
	Left Slide	Turn down a horizontal slider bar
	Up Slide	Turn up a vertical slider bar
	Down Slide	Turn down a vertical slider bar

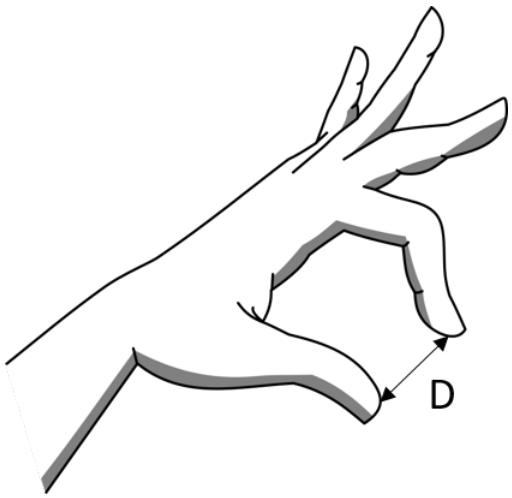


Figure 3.7: The Euclidean distance (D) between the thumb and the index finger of the left hand.

would suffer a problem that, some people have difficulty in completely outstretching other fingers while doing a specific Finger-to-thumb gesture. And this may cause a relatively low recognition accuracy for that touch gesture, especially for the middle and pinky fingers. Figure 3.8 shows the unconsciously bended ring finger is influencing the detection of the middle touch gesture as its distance between the thumb may also surpass the threshold value.

Figure 3.8: The unconsciously bended ring finger is influencing the detection of the middle touch gesture as its distance between the thumb may also surpass the threshold value.

We modified the detection algorithm which now not only alleviated this problem, but achieved a higher recognition accuracy for all touch gestures. The new detection algorithm works as follows:

1. Check if there is a pinch pose in the current frame by valuating the *PinchStrength* property of the hand being detected. *PinchStrength* is supposed to be a measure of the holding strength of a pinch hand pose. It is zero for an open hand, and increases to 1.0 when a pinch hand pose is recognized.

2. If we detect a pinch pose, get the 3D position data of each fingertip of that pinched hand. If not, continue Step 1.

3. Calculate the Euclidean distances between the thumb fingertip and other fingertips. The finger with the smallest distance will be considered as the pincher (the finger touching with the thumb).

4. The system executes the task mapped to the pincher.

Our preliminary experiment compared accuracy of the improved algorithm with the old algorithm and their results are summarized in Figure 3.9. We can see that the new algorithm works much better than the old one, especially for the ring and pinky touch detection. The mean accuracy among these gestures is 97.13% for the new algorithm and 86.73% for the old one, which indicates that we approximately achieved a 10.4% improvement.

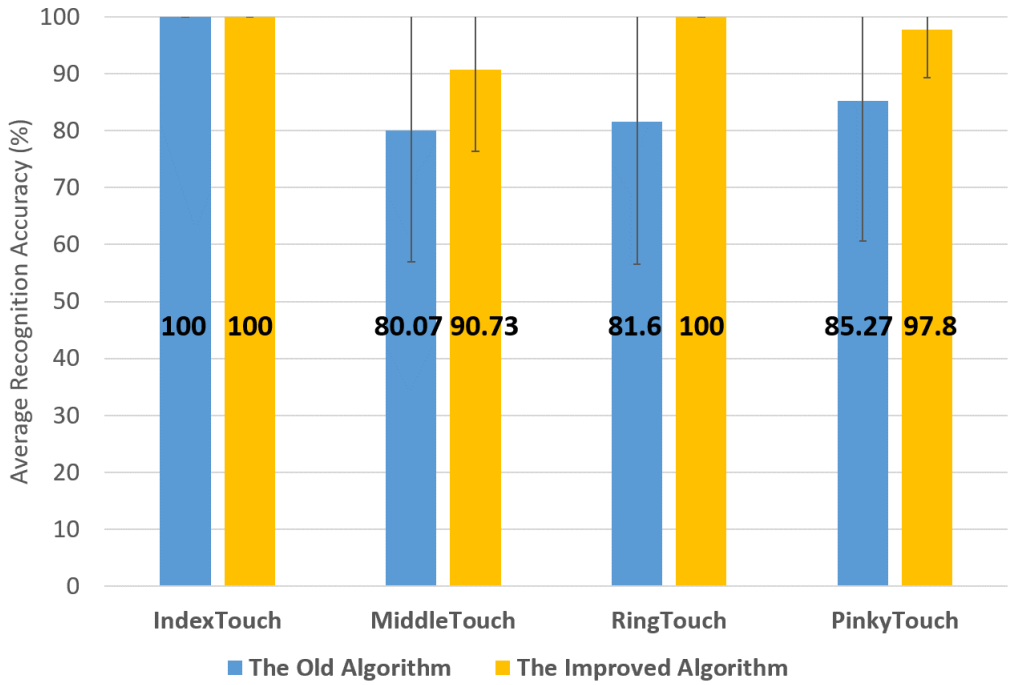


Figure 3.9: Touch gesture accuracy comparison of old and new algorithms with standard deviation error bars.

3.3.2 Swipe Gesture Detection

Although the Leap Motion API provides a Swipe Gesture detection solution, we can not integrate it directly due to its incapacity to distinguish directions. We modified the detection algorithm of swipe gestures to make it able to distinguish four directions: up, down, left, and right. The modified algorithm is described below:

1. While the Leap Motion detects a swipe type gesture, it provides us its direction vector - $Direction$.

2. We compare the absolute values of $Direction.x$ and $Direction.y$. If $Direction.x > Direction.y$, we believe this is a horizontal swipe gesture, otherwise, a vertical swipe gesture.

3. If it is a horizontal swipe gesture, we then see if $Direction.x > 0$, and if so, it is detected as a right swipe gesture, if not, it is a left swipe gesture.

4. If it is a vertical swipe gesture, we instead to see if $Direction.y > 0$ to define whether it's a up or down swipe gesture.

Another problem is the Leap Motion reports swipe gestures in a frame-based way so it would report multiple swipe gestures even the user only performed once. We used a flag variable to make the system able to detect discrete gestures.

3.3.3 Touch-then-Swipe Gesture Detection

A Touch-then-Swipe gesture is the combination of a touch gesture and a swipe gesture, the same as its detection algorithm. That is, only when we have detected a touch gesture, we enable the system to detect swipe gestures. If the touch gesture stopped, the system also stops the detection of swipe gestures.

Chapter 4

Experiments

To investigate the accuracy and performance of our Finger-to-Thumb system, we conducted three experiments. We expect through these experiments, we can verify that our system is qualified for a menu selection technique, that is, to be fast, high-accuracy and effortless.

4.1 Experiment 1 - Accuracy

A usable gestural interface firstly depends on its recognition accuracy. We must let our system clearly understand what the user wants to do.

4.1.1 Purpose of the Experiment

The first experiment was carried out to measure the average recognition accuracy of each gesture in our system, including touch and swipe gestures.

4.1.2 Participants

Five participants (4 male, 1 female) aged from 24 to 26 joined this experiment as volunteers. All of them were right-handed, and were graduate students majored in computer science. Among them, only one ever used a gestural interface and none were familiar with it. Participants were given an informed consent form (Appendix A) explaining the purpose

and potential effects of this experiment. They were also told that they could stop at any time.

4.1.3 Apparatus

The experimental setup, shown in Figure 4.1, consisted of a Lenovo ThinkPad E440 laptop (specs: Intel(R) Core(TM) i3-4000M CPU processor, 4GB RAM, Intel HD Graphics 4600 integrated graphics card) and a Leap Motion Controller connected with the laptop via USB. The experiment system software was developed using Unity 5.3 (C# scripting) with Leap Motion Orion SDK and its Unity Core Assets.

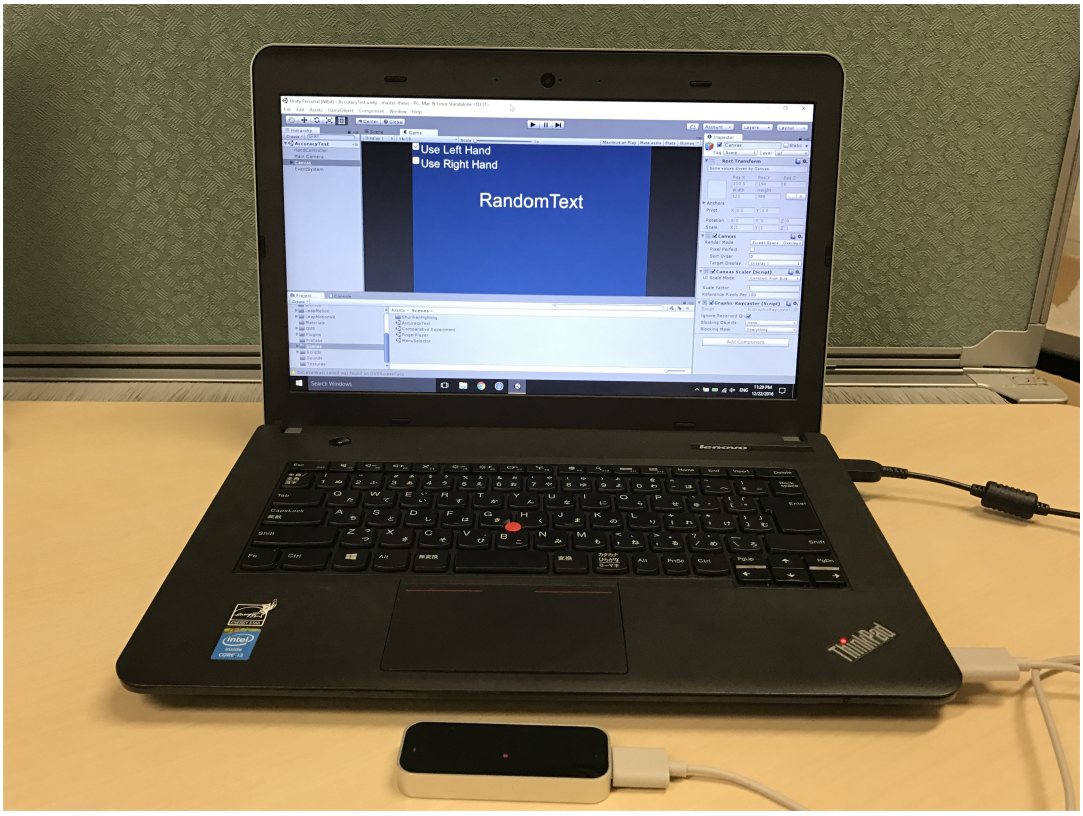


Figure 4.1: The experimental setup.

During the experiment, participants lifted their hands approximately 10 to 25 centimeters over the Leap Motion Controller so that their hands could be detected properly, as Figure 4.2 illustrates.

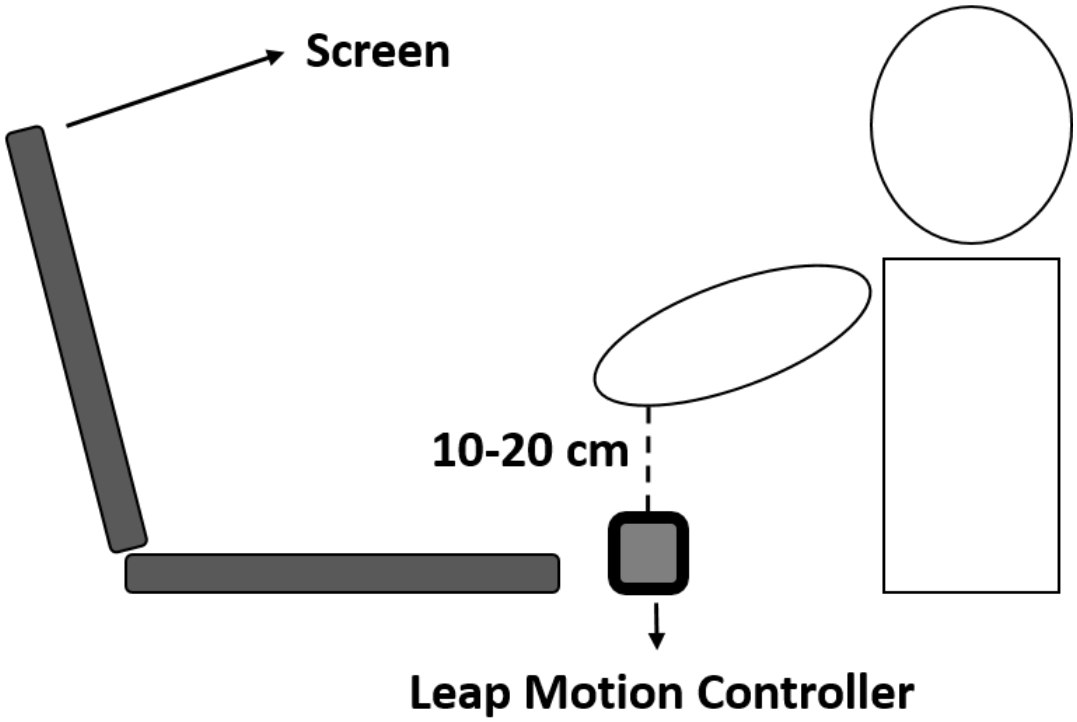


Figure 4.2: Diagrammatic sketch of the experimental setup's side view.

4.1.4 Procedure

The procedure of this experiment for each participant is described as below:

1. The experimenter explained how to use the system, including all the gestures.
2. The participant began with a 2 minutes' training phase, during which he/she practiced with the system. The system also allowed the participant to choose his preferred hand (handedness) to do the gestures.
3. The experiment phase consisted of 3 sessions. In each session, the participant was asked to do 20 number of gestures randomized by the system. These gestures were index touch, middle touch, ring touch, pinky touch, left swipe, right swipe and up swipe, totaled 7 kinds. Figure 4.3 shows a participant is performing the ring touch gesture.
4. After one session, the participant took a 1 minute's break, then continued with the next 20 tasks.
5. After all three sessions were done, the participant was asked to fill out a questionnaire (Appendix B) about his/her impression. The whole experiment took about 10 minutes for each participant.

4.1.5 Result and Discussion

Figure 4.4 shows the average recognition accuracy of each gesture tested in this experiment. We can see that all these gestures can be correctly detected with a high accuracy over 90%. We observed that sometimes when the participant was performing a middle touch gesture, the 3D model of his/her hand was however being rendered as an index touch gesture, as shown in Figure 4.5. Once the participant adjusted the angle of his/her hand slightly, the mis-recognition disappeared. This is the reason why middle touch gestures had a relatively low accuracy, and it may be due to some hardware limitations of the Leap Motion Controller. We also observed that some participants were unable to completely outstretch other fingers while performing a certain touch gesture, which furthermore, decreased accuracy for that gesture. This phenomenon was much more explicit when we used the old algorithm to detect touch gestures. For up swipe gestures, the relatively low accuracy was caused by the situations that some participants would put their hands down soon after they completed a up swipe gesture, which led to a wrong recognition of a down swipe gesture.

From the questionnaires, we obtained the feedback that most participants (4 of 5) thought these gestures were very easy to perform. And none of the participants said they felt tired

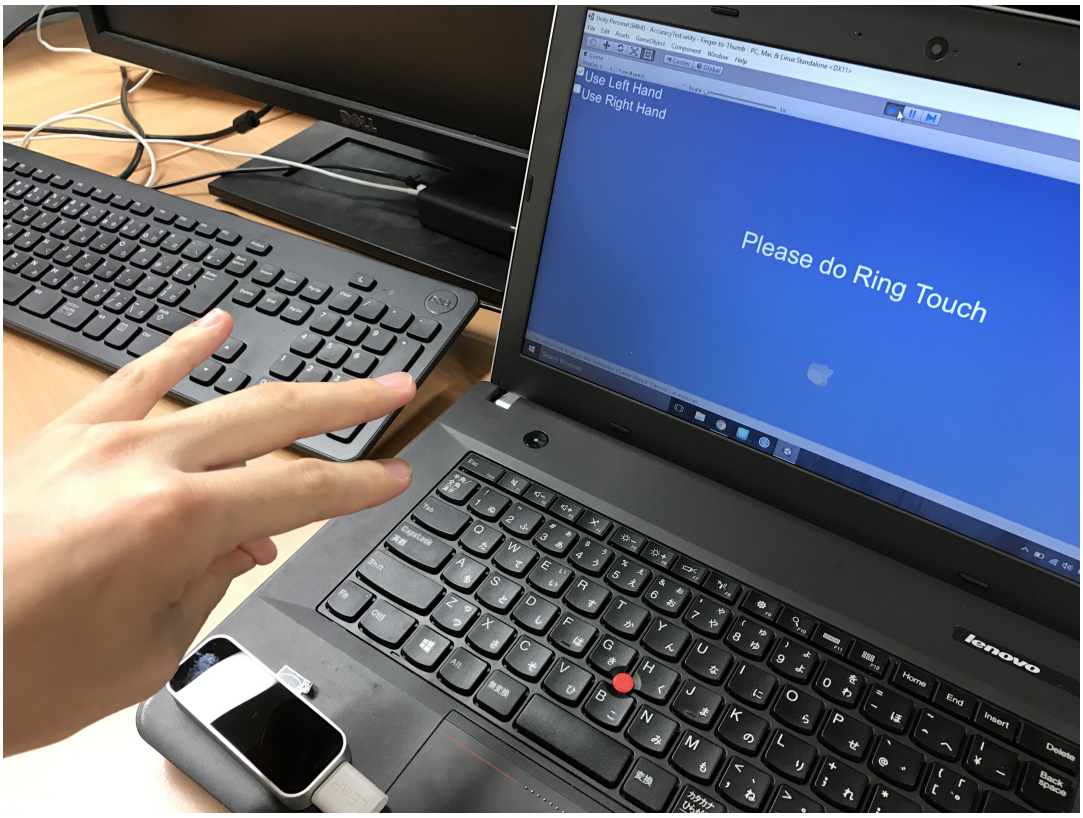


Figure 4.3: A participant is performing the ring touch gesture instructed by the system.

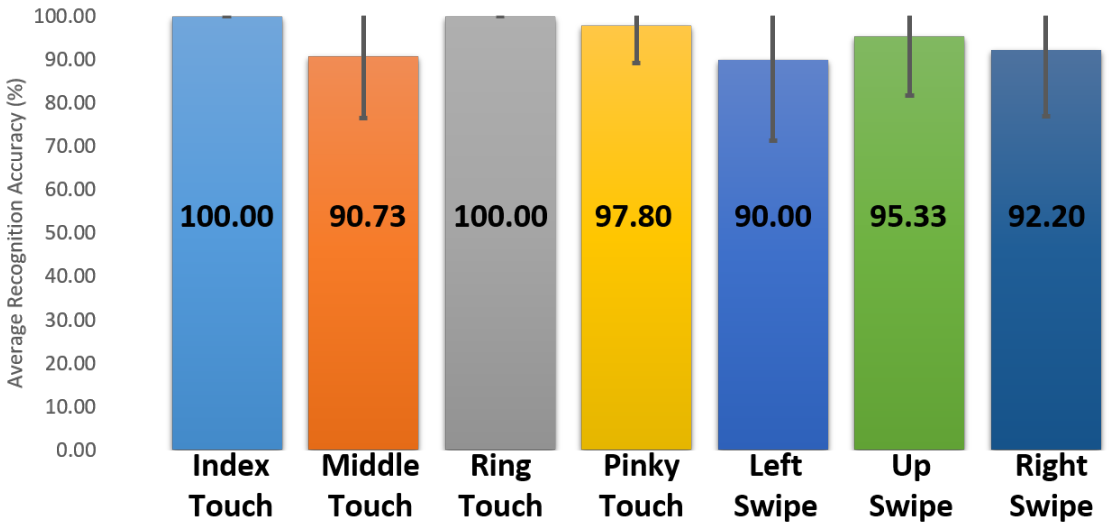


Figure 4.4: Average recognition accuracy of touch gestures and swipe gestures with standard deviation error bars.

after the experiment. However, some participants said that they prefer performing touch gestures and swipe gestures with the same hand, which later inspired us to come up with the design of Touch-then-Swipe gestures.

4.2 Experiment 2 - Comparison with Move-and-Tap

We have introduced some other menu selection techniques in Section 1.1. For comparison, we developed a Move-and-Tap system which works as follows:

1. The user moves one of his/her hands over the Leap Motion Controller, which consequently moves the cursor-like indicator on the screen corresponding to the hand's movement.
2. After the user succeeded in moving the indicator to his/her desired button, he/she can confirm his/her selection (click the button) by executing an in-air tap (towards the screen) gesture.
3. The system handles the task assigned to that button (in this experiment, we simply

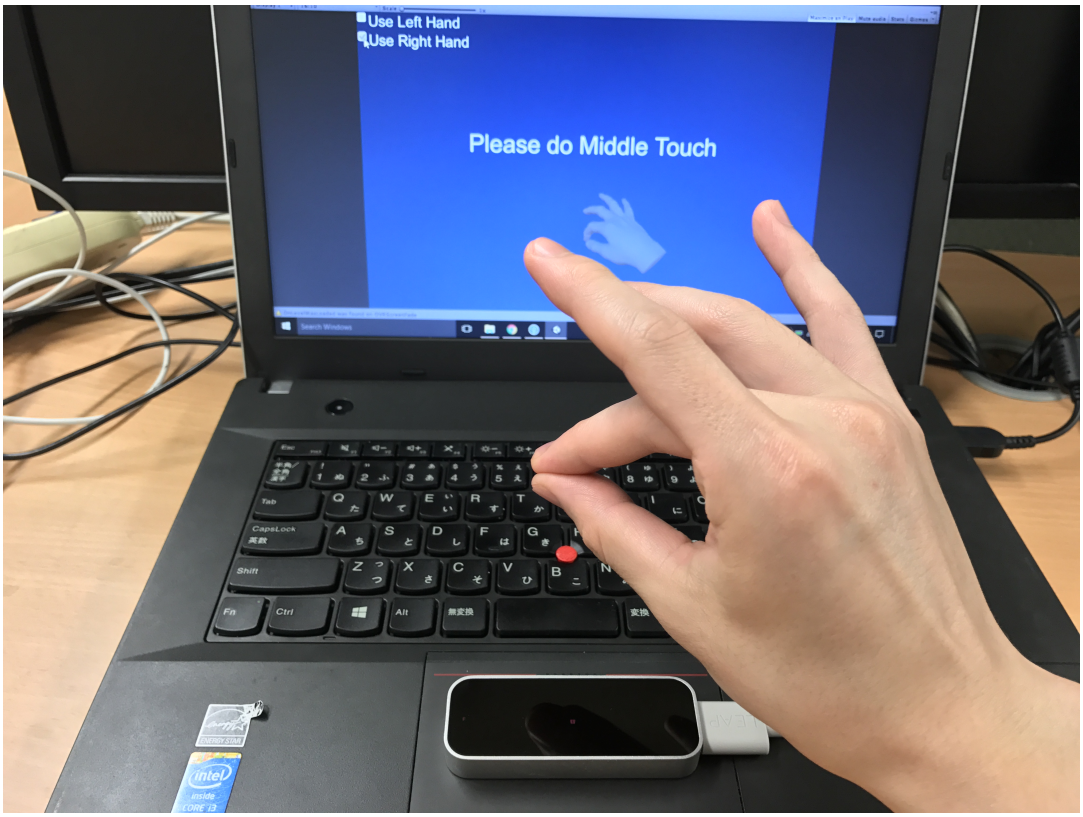


Figure 4.5: The middle touch gesture is rendered as an index touch gesture by the Leap Motion.

changed the button's color once it was selected).

This section describes the experiment that we conducted to compare the performance between Finger-to-Thumb and Move-and-Tap.

4.2.1 Purpose of the Experiment

Experiment 2 was carried out to compare the performance of the two gestural menu selection techniques: Finger-to-Thumb versus Move-and-Tap. We expect our Finger-to-Thumb system to outperform the Move-and-Tap system, achieving faster gesture execution time, lower error rate and causing less fatigue to the users.

4.2.2 Participants

8 volunteers (7 males and 1 female) majored in computer science took part in this experiment. They aged from 23 to 25 and were all right handed. None of the participants were familiar with gestural interfaces. The same informed consent was given as Experiment 1 did.

4.2.3 Apparatus

The experiment setup was actually the same as described in last section, other than the software was redeveloped to a button selection application which can also automatically record experimental data.

4.2.4 Measurements

We took three measurements into account for this experiment:

1. The execution time of each gesture, i.e., the time cost to complete one gesture task.
2. The error rate of each gesture. Error rate reflects not only the accuracy, but also how that gesture is easy to understand.

3. The fatigue degree of our system. Fatigue degree is, however, a qualitative measurement voted by the participants. It tells whether our system is effortless or not.

Execution time T is the amount of time (in seconds) that passed from when the system displays a randomized instruction text (e.g., Please select Button 1) to when the participant correctly completes the corresponding selection task. The system continues displaying another instruction text after the corresponding selection task is done. We made it to instruct the participant to complete 15 selection tasks in one session. Thus, after each session, we would have 15 number of execution time being outputted and their mean value was auto calculated by the system.

Error rate ER is calculated by $ER = (ToG - ToC)/ToG$, where ToG means the total number of gestures the participant actually did in one session, while ToC is the total number of gestures the system asked the participant to do, in our settings , $ToC = 15$. Our experiment system counted every gesture the participant did during one session which was then assigned to ToC. Hence we could directly read the error rate calculated by the system.

Fatigue degree FD is, however, a qualitative measurement derived from questionnaires. We used a Likert-type scale [Lik32] dividing the fatigue degree into 5 levels. The highest score 5 means most tired and the lowest score 1 means least tired.

4.2.5 Procedure

Experiment 2 was conducted by the following steps:

1. Participants began with a 2 minutes' training phase. During the training phase, participants learned how to use these two systems and practiced with them. A 1 minute's break was given after the training phase.
2. Then, the experiment phase started. To avoid order effect, participants were divided equally into two groups hence each group had 4 participants. The participants in Group A were asked to use the Finger-to-Thumb system first for two sessions, and then turned to use the Move-and-Tap system for another two sessions. The participants in Group B did the same tasks but in a reversed order. Table 4.1 presents the session agenda of each group.
3. In each session, the participant was asked to do 15 selection tasks under the instruction of the system. Figure 4.6 shows the participant is doing the task of selecting Button 3. The total number of selection tasks one participant should complete was 60 (15 tasks $\times$ 4 sessions).

4. Including the 1 minute break after each session, the experiment lasted nearly 15 minutes.

5. After the experiment, participants were asked to fill out a questionnaire (Appendix C) about their impressions on these two systems.

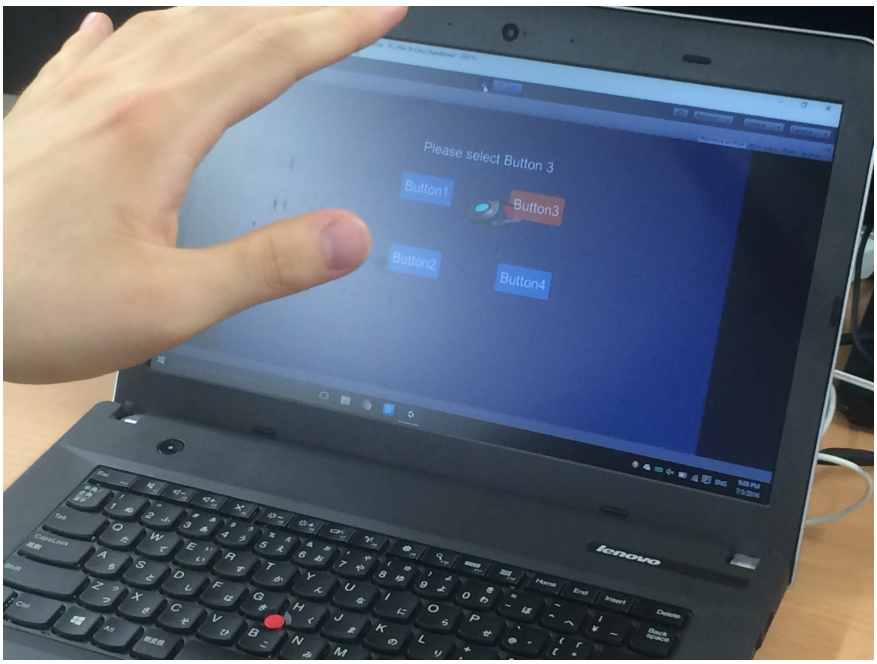


Figure 4.6: The participant is doing the task of selecting Button 3.

Table 4.1: Session agenda of Experiment 2 (avoiding order effect)

	Group A	Group B
Session 1	Finger-to-Thumb	Move-and-Tap
Session 2	Finger-to-Thumb	Move-and-Tap
Session 3	Move-and-Tap	Finger-to-Thumb
Session 4	Move-and-Tap	Finger-to-Thumb

4.2.6 Result and Discussion

Execution time comparison between Finger-to-Thumb and Move-and-Tap is presented in Figure 4.7. The average execution time of Finger-to-Thumb system was 0.996 second with a standard deviation equaled to 0.121. For the Move-and-Tap system, the average execution time was 2.147 seconds with the standard deviation being 0.382. Consequently, Finger-to-Thumb was about 1 second faster than Move-and-TAP for each selection task. The

t-test ($t = -8.227, p < 0.005$) also indicated that there was significant difference between Finger-to-Thumb and Move-and-Tap.

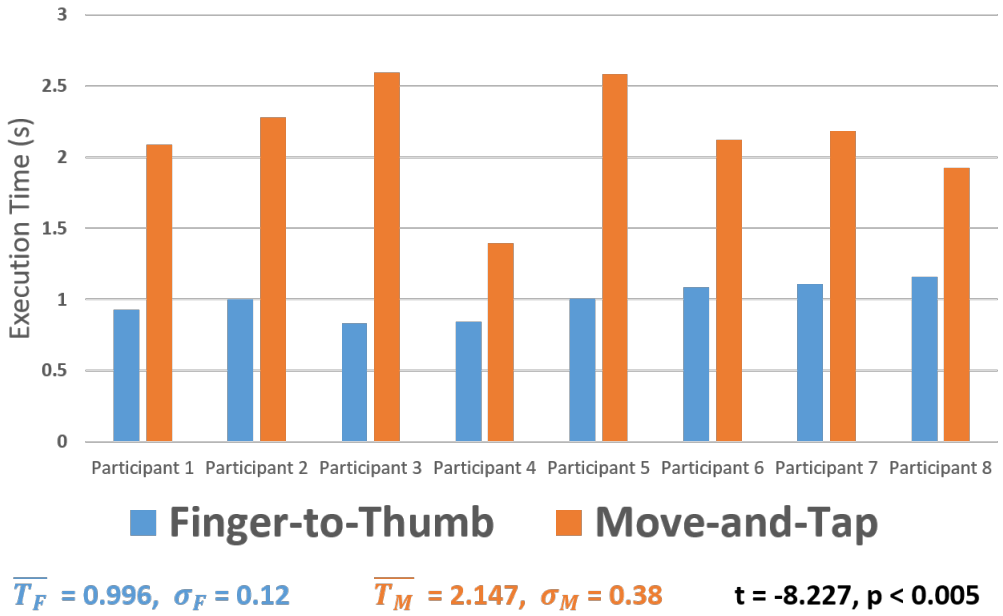


Figure 4.7: Finger-to-Thumb vs Move-and-Tap - Execution time comparison.

Error rate comparison between Finger-to-Thumb and Move-and-Tap is presented in Figure 4.7. The average error rate of Finger-to-Thumb was 16.4% ($SD = 5.22$), smaller than Move-and-Taps 17.3% ($SD = 6.98$), however there was no significant difference between them according to the t-test ($t = -0.413, p = 0.692$).

The collected questionnaires told us that the Finger-to-Thumb system got a average fatigue degree score of 1.8 points, lower than the Move-and-Tap system's 3 points. And this result indicated the participants thought Finger-to-Thumb was easier to use than Move-and-Tap. A more explicit evidence was that all the participants voted they prefer the Finger-to-Thumb system than the Move-and-Tap system. We think it is because the selection mechanism of Finger-to-Thumb does not require the user to keep moving their hands and an in-air tap gesture is much power-consuming than a touch gesture.

On the other hand, we also found that some participants complained they sometimes would get confused thinking which finger should be touched while they were using the Finger-to-Thumb system. Although this "delay" problem would get alleviated as they kept using the system, we are considering improving the visual feedback by dynamically drawing the buttons on the 3D hand model. This will be discussed in Chapter 6 as one of our future work.

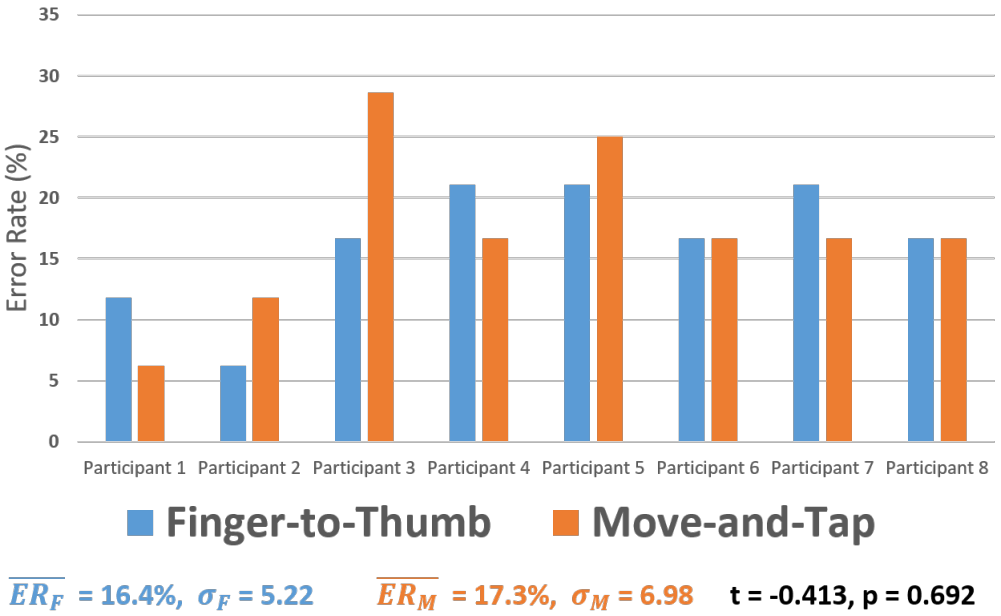


Figure 4.8: Finger-to-Thumb vs Move-and-Tap - Error rate comparison.

4.3 Experiment 3 - Eyes-free

Eyes-free interaction refers to the interaction that not depends on any visual feedbacks. Users can quickly execute some tasks without keep watching on the screen. This not only benefits the disabled but improves efficiency for all the users. We designed this experiment to find whether our Finger-to-Thumb system could enable eyes-free interaction. Since the participants and setup were the same as Experiment 2, we no longer state them in this section.

4.3.1 Procedure

Experiment 3's procedure is as below:

1. As DigitSpace [HCY+16]'s design, we placed an opaque carton-obstacle between the participant and his/her gesture hand (the hand used to perform gestures) to prevent the participant looking his/her gesture hand, as Figure 4.9 shows.
2. The participant kept watching on the screen, which provided visual instructions, completing the tasks (the same as described in Experiment 2) without looking his/her hands.

3. This experiment contained two repeating sessions. Including the 1 minute's break after each session, it cost nearly 5 minutes for each participant.

Figure 4.9: The opaque carton-obstacle the participant's sight so that he/her cannot watch his/her hand during the experiment.

4.3.2 Result and Discussion

The comparison of execution time between eyes-free mode and normal (non-eyes-free) mode is illustrated in Figure 4.11. The average execution time was 0.996 second ($SD = 0.12$) for the normal mode, and 1.004 seconds ($SD = 0.09$) for the normal mode. The t-test ($t = -0.290, p = 0.780$) showed that there was no significant difference between them.

The comparison of error rate between eyes-free mode and normal (non-eyes-free) mode is illustrated in Figure 4.12. The average error rate of the normal mode was 16.4% ($SD = 5.22$), the same as the eyes-free mode but with a different standard deviation ($SD = 3.75$).

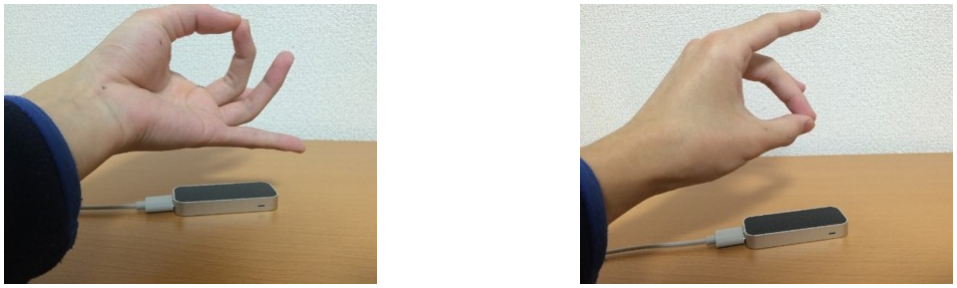
The t-test ($t = 0.004, p = 0.996$) also showed that there was no significant difference between them.

These two results suggested that our Finger-to-Thumb system can enable eyes-free interaction.

4.4 Pros and Cons of Finger-to-Thumb

This chapter summarized all the experiments we have conducted to evaluate the performance and usability of Finger-to-Thumb menu selection technique. From the results, we can conclude that the pros of Finger-to-Thumb contains: fast (less than 1 second to complete a selection task), high accuracy (all of the gestures got a recognition rate over 90%) and effortless (a low fatigue degree voted by participants). Furthermore, we also verified that the Finger-to-Thumb system can enable eyes-free interaction, which could be very useful for developing future applications.

Nevertheless, there are still some limitations in the Finger-to-Thumb system. One is the occlusion problem: when the user executes touch gestures with his/her back of hand facing the Leap Motion Controller, his/her back of hand will block the detection of that gesture, for this reason, the user had to execute touch gestures with his/her palm facing the Leap Motion Controller, as shown in Figure 4.10 Another limitation is since we only have four fingers on one hand, when there are a lot of items in the menu, it may force users to execute multiple swipe gestures to reach their desired menu item, which may cause tiredness.



(a) (b)

Figure 4.10: Occlusion problem of the Finger-to-Thumb system. (a): The Leap Motion Controller cannot see the index touch gesture due to occlusion. (b): Users have to do touch gestures with their palm facing the Leap Motion Controller.

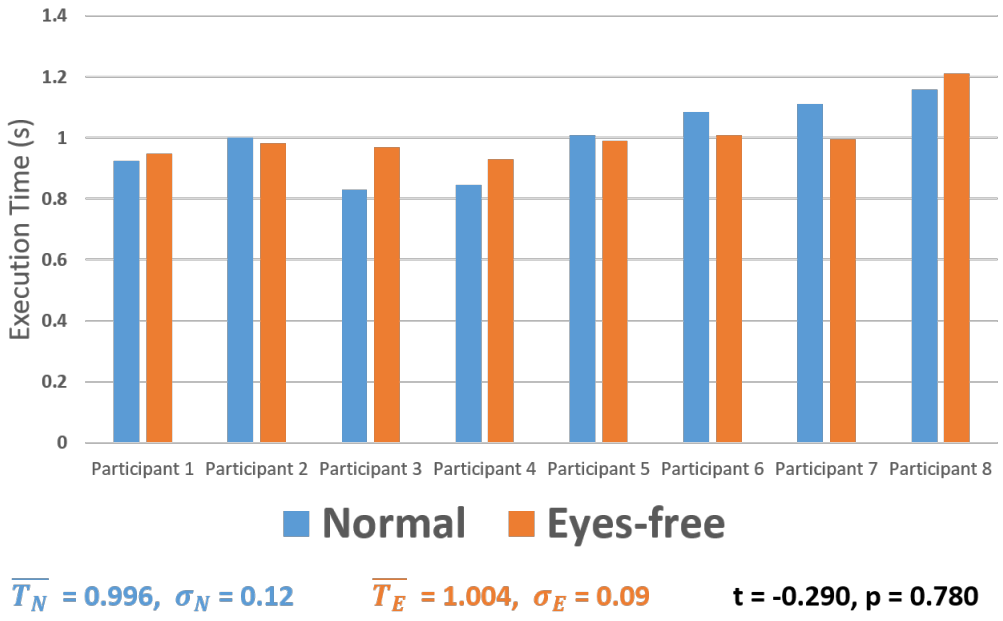


Figure 4.11: Comparison of execution time between eyes-free mode and normal (non-eyes-free) mode.

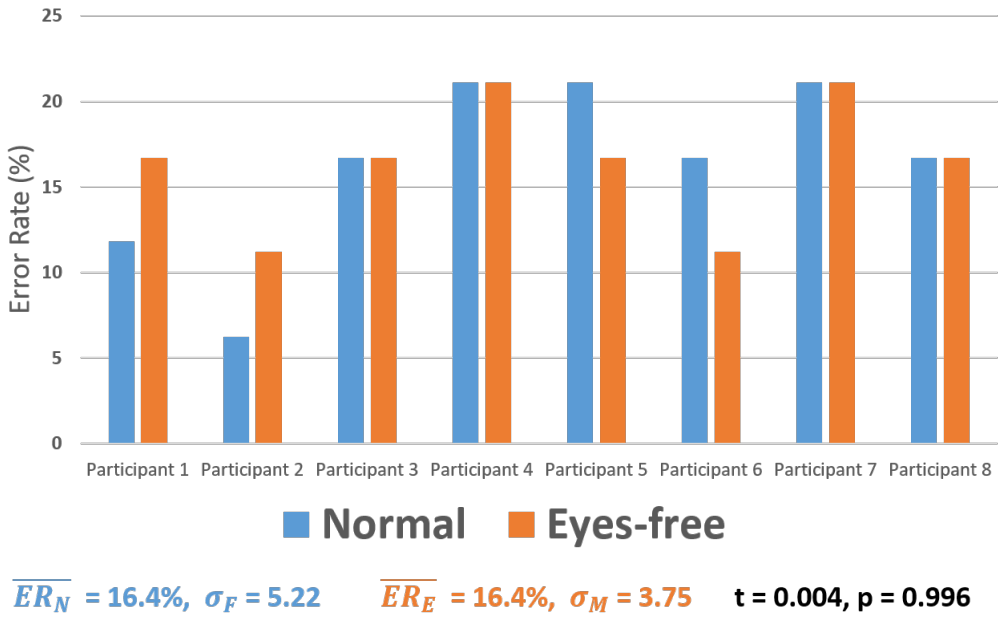


Figure 4.12: Comparison of error rate between eyes-free mode and normal (non-eyes-free) mode.

Chapter 5

Applications

We have discussed that the Finger-to-Thumb menu selection technique is rather suitable for small scale selection tasks. This chapter introduces two gesture controlled applications developed based on Finger-to-Thumb: ShurikenFighting and FingerPlayer.

5.1 ShurikenFighting

The capability of real time interaction makes it a good candidate to develop game applications. We developed *ShurikenFighting*, a gesture controlled game that can be played with the Finger-to-Thumb gestures.

5.1.1 Gameplay

There are two Ninja characters, respectively, standing on the left side and right side of the main scene. A player firstly chooses his/her preferred hand (to execute touch gestures), which accordingly, determines which Ninja he/she will play as. Then, the player can throw a Shuriken towards the Ninja in the opposite direction by executing a Finger-to-Thumb gesture. Different touch gestures (index touch, middle touch, ring touch and pinky touch) are mapped to different heights, where the Shuriken will be thrown out. For instance, you can throw out a Shuriken at the lowest height by executing an index touch gesture. If the opposite Ninja failed shooting down the coming Shuriken in time (by throwing out a Shuriken at the same height), he will get "hit" (actually the Ninja character does not physically get hit by the Shuriken, we say the Ninja get hit when the opposite Shuriken

reached the edge of the corresponding side) and lose health points.

5.1.2 Game modes

There are two game modes in this game: a PvE mode for players to practice with AI, and a PvP mode for playing against other human players. Players can switch game modes from the start screen, as Figure 5.1 shows.

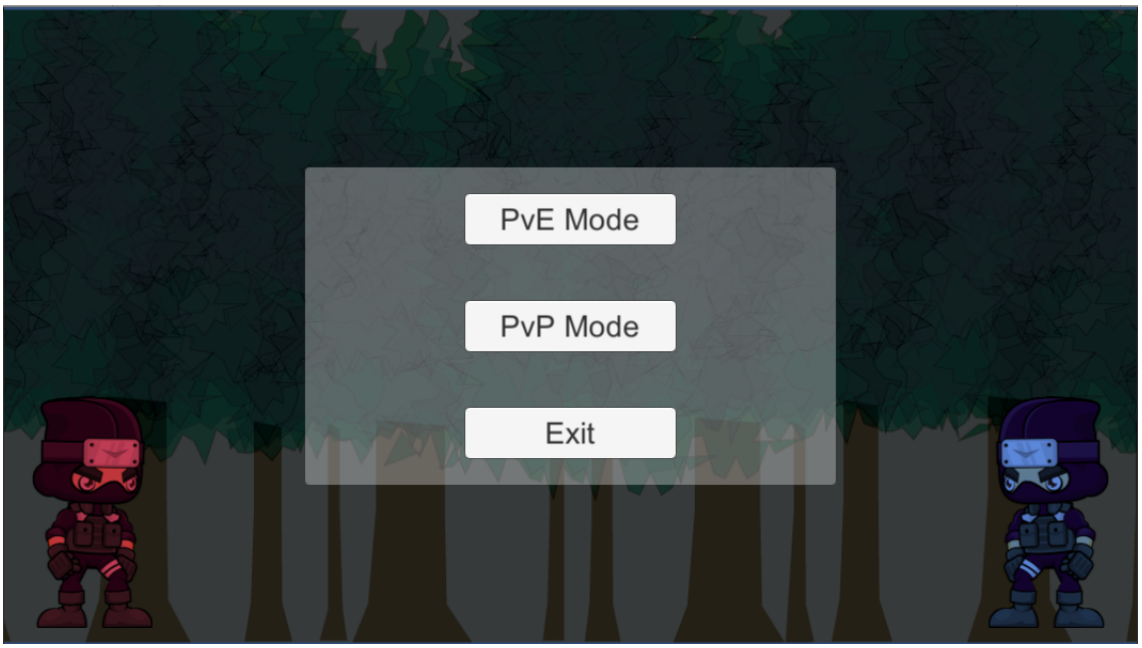


Figure 5.1: Start screen of *ShurikenFighting*. Players can switch between PvE mode and PvP mode here.

In the PvE mode, players choose to play as one of the two Ninja characters and the remaining one will be controlled by AI. During the game, players must keep throwing Shurikens (executing touch gestures) to shoot down the Shurikens thrown by the AI controlled Ninja. Since the AI controlled Ninja has an infinite health points, players cannot literally defeat him but accumulate scores when they success in shooting down Shurikensthrown by the opposite enemy (Every time the player successes in shooting down one Shuriken, his/her score increases by one point). In contrast, the player controlled Ninja could die if his health points decrease to zero due to hit by the enemy's Shurikens, and at which time, the game ends. Figure 5.2 shows the player is playing as the left side Ninja and trying to shoot down the Shurikens thrown by the enemy.

Figure 5.2: The player is playing as the left Ninja, throwing Shurikens to shoot down the enemy's Shurikens. The current acquired score is displayed in the middle top on the screen. The yellow bar on the upper left corner represents the player's remaining health points.

The PvP mode allows players to play against each other. This time, each side's Ninja has the same amount of health points that would decrease if the Ninja got hit. The game ends when either one side of the two Ninjas died.

5.2 FingerPlayer

FingerPlayer is a gesture controlled music player that supports all the main functions built in a common music player, including play/pause, play next/previous, fastforward, backforward and volume up/down. Figure 5.3 shows the main display of *FingerPlayer*.

5.2.1 Gesture Mapping

FingerPlayer uses touch gestures and Touch-then-Swipe gestures to control the music player. Each gesture is mapped to a unique function as shown in Table 5.1.

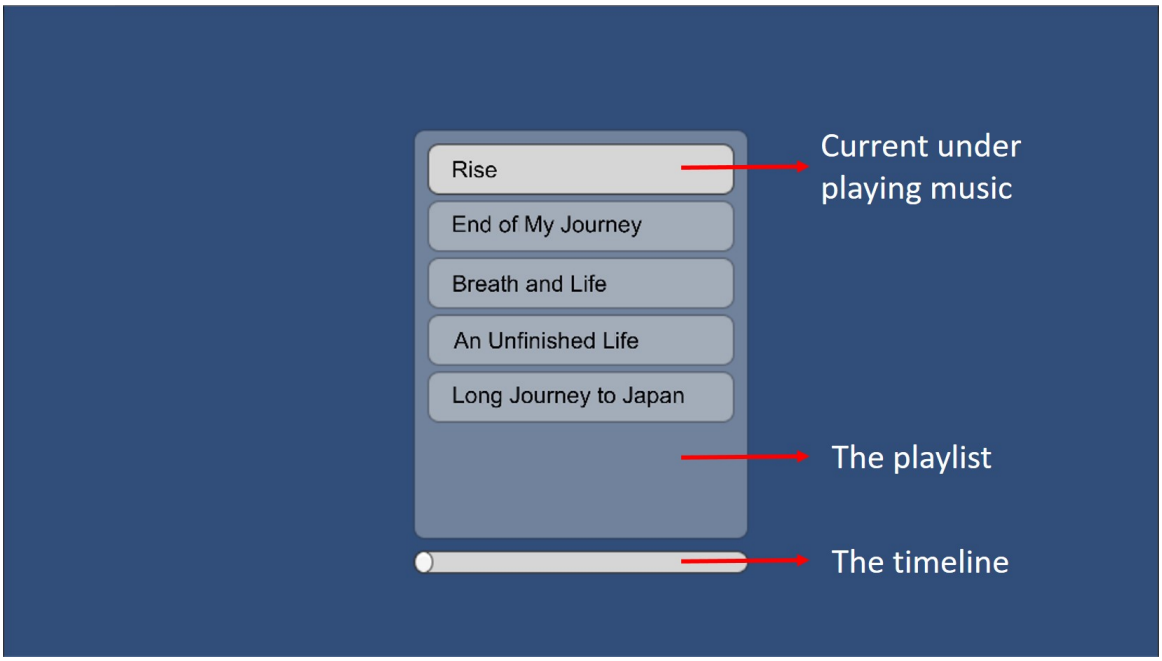


Figure 5.3: The main display of *FingerPlayer*.

Table 5.1: *FingerPlayer* gestures and their corresponding functions

Gestures	Functions
Index Touch	Play/Pause
Middle Touch	Play Previous
Ring Touch	Play Next
Index Touch-then-Swipe (swipe to left)	Backforward
Index Touch-then-Swipe (swipe to right)	Fastforward
Pinky Touch-then-Swipe (swipe to left)	Volume down
Index Touch-then-Swipe (swipe to right)	Volume up

Index touch gestures are used to play and pause the music. When *FingerPlayer* detects an index touch, it firstly checks if currently is playing a piece of music. If so, it pauses the music; and if not, it begins to play the music.

Middle touch gestures are used to simulate the play previous function, i.e., let the player play the previous music in the playlist. Oppositely, ring touch gestures are used to simulate the play next function (play the next music in the playlist).

Volume up/down is, however, adjusted by pinky Touch-then-Swipe gestures. When the system detects a pinky touch gesture, it continues to check whether that pinky touch posture is maintained during the movement of the hand. A left pinky Touch-then-Swipe gesture

will turn down the volume, and the right counterpart will turn up the volume. A volume slider will be displayed if the system detects pinky Touch-then-Swipe gestures in order to provide visual feedback, as Figure 5.4 shows.

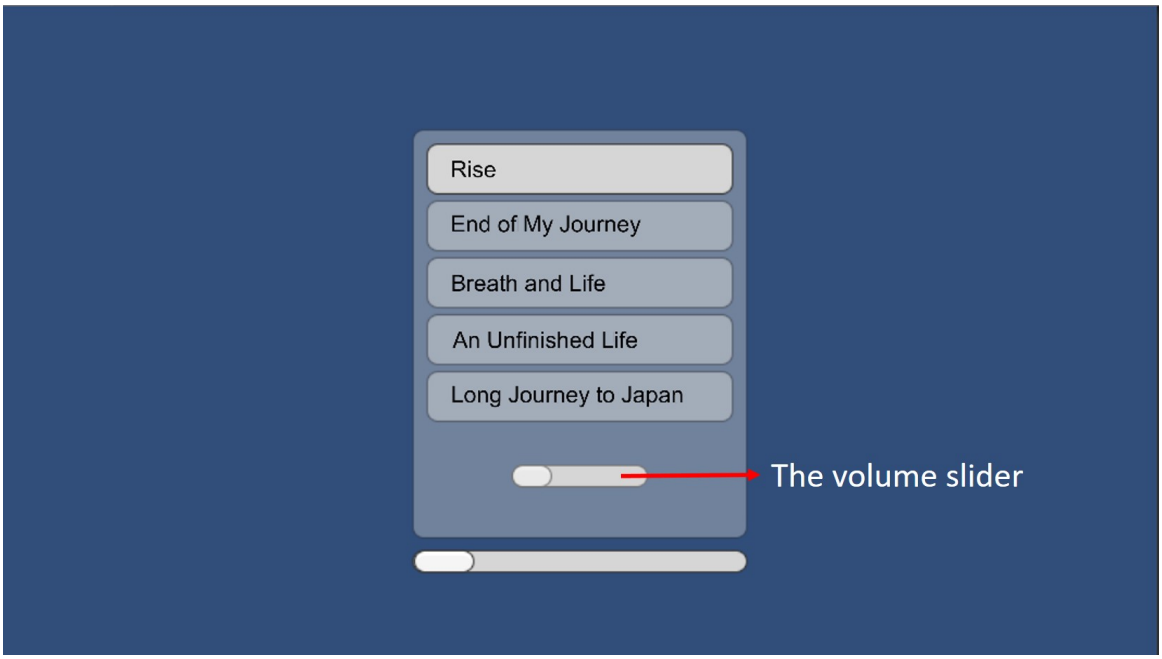


Figure 5.4: The volume bar shows up when the system detects a pinky Touch-then-Swipe gesture.

Touch-then-Swipe gestures are also applied to the index finger. An index Touch-then-Swipe gesture allows the user to adjust the timeline to seek new playback time he/she wants. The left index Touch-then-Swipe gesture implements fastforward function, and the right one implements backforward function. The adjust scale is based on the length of the song, hence we can restrain the movement within a rather small space (at least within the limited detection range of the Leap Motion Controller).

Chapter 6

Future Work

We discuss our future work in this chapter, including the improvement of our menu selector application (used in Experiment 2) and the expectation of AR/VR applications.

6.1 Management of Large Number of Menu Items

We have showed that the limitation of Finger-to-Thumb is that it only allows users to select among 4 menu items simultaneously. When there are more than 4 menu items designed in the system, users must do swipe gestures to activate other menu items to make them selectable.

Although swipe gestures are simple and familiar with us since they are widely applied in current touch screens, frequent use of mid-air swipe gestures performed by the other hand (other than the hand that used to perform touch gestures) may induce additional fatigue. Some participants also reported that they wished to perform the touch gestures and swipe gestures with the same hand in the Experiment 1. However, performing touch gesture and swipe gesture with the same hand is not such efficient as the transition between these two kinds of gestures is not smooth enough.

Instead of swipe gestures, we are considering other gestures that can be performed smoothly between touch gestures with the same hand, like the rotation gesture of the wrist presented in [RMB08]. We believe a well designed switch gesture can greatly mitigate the limitation of Finger-to-Thumb.

6.2 FingerButtons

In the menu selection application (Section 4.2), since buttons were aligned statically on the screen, some participants complained that they sometimes felt confused about the mapping relations between fingers and buttons. Although this problem can be alleviated as they got familiar with the system, we think a more intuitive visual feedback can help user get rid of remembering the mapping relations. The solution we have come up with is *FingerButtons*, a hand menu interface with buttons drawn on the fingers, as Figure 6.1 illustrates. The 3D hand models rendered by Leap Motion are true (non-reversing) mirror images of real hands. They reflect the real time movement of users' hands, hence can provide an intuitive visual feedback.

With *FingerButtons*, novice users do not need to remember the mapping relations between fingers and buttons, since the interface provides a real time visual feedback that informs which finger is mapped to which button. While expert users can execute Finger-to-Thumb gestures in an eyes-free way (turn off the visual feedback), which has been verified in Section 4.3.

6.3 Applications in AR/VR Environments

Gestural interfaces provide a more immersive and natural interaction way than traditional input devices (mouses and keyboards) while used in AR/VR environments. We think Finger-to-Thumb is also eligible for AR/VR applications, especially with *FingerButtons*. It can not only give intuitive visual feedback, but also avoid the requirement of looking for menu items as they are right on your fingers.

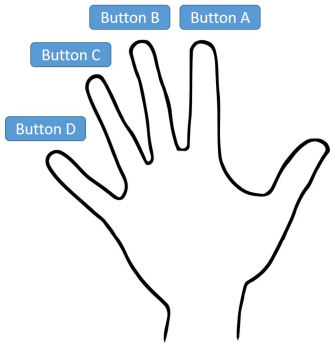


Figure 6.1: *FingerButtons*: a hand menu interface with buttons drawn on fingers.

Chapter 7

Conclusion

In this thesis, we firstly introduced the related work of gesture-based interaction, including the development of hand gesture recognition technologies and design concepts of hand gestural interfaces. We also discussed the advantages and disadvantages of each hand gesture recognition technology with some example researches.

Considering the remaining problems in the earlier researches, we proposed Finger-to-Thumb: a gesture-based menu selection technique. Finger-to-Thumb uses Leap Motion as the finger tracking device so that it can avoid the requirement for on-body sensors.

We defined three kinds of gestures for the Finger-to-Thumb system: touch gestures, swipe gestures and Touch-then-Swipe gestures. Touch gestures are the gestures that you touch your thumb to other fingers on the same hand, hence are also called Finger-to-Thumb gestures. Touch gestures are used to activate selection events. For instance you can execute an index touch gesture to select the button mapped to the index finger. Swipe gestures are in-air horizontal or vertical moving gestures of the hands, which are used to switch between button panels. Touch-then-Swipe gestures are however the combination of touch gestures and swipe gestures that are executed by holding the touch postures while moving the hands. Touch-then-Swipe gestures are used to simulate slider functions, such as the volume slider in a gesture controlled music player application.

The detection algorithm of each gesture was described in Section 3.3.

Our experiments verified that Finger-to-Thumb is fast, high-accuracy and effortless. The average execution time of a selection task was less than 1 second, which greatly outperformed the widely-used Move-and-Tap system. The preliminary study showed that all the touch gestures, as well as swipe gestures, achieved a mean recognition accuracy over 90%.

Furthermore, the collected questionnaires indicated that participants were satisfied with Finger-to-Thumb because of its fast speed and few feeling of tiredness. Besides, we also verified Finger-to-Thumb can enable eyes-free interaction, which can be a convenient feature for application development.

We developed two applications based on Finger-to-Thumb: *ShurikenFighting* and *Finger-Player*. The first one is a simple fighting game uses solely touch gestures (Finger-to-Thumb gestures). The second one is a gesture controlled music player that uses touch gestures and Touch-then-Swipe gestures.

The future work consists of *FingerButtons* and applications for AR/VR environments. *FingerButtons* can be an improvement for our menu selection application that can dynamically display buttons on the user's 3D hand model (distribute on each finger).

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor, Dr. Buntarou Shizuki, for his continuous advice and guidance throughout my master study. His patience and conscientiousness influenced deeply on my attitude towards research and study, even daily life.

I also want to thank my former supervisor Professor Jiro Tanaka and other professors of the Interactive Programming Laboratory: Associate Professor Shin Takahashi, Associate Professor Satoshi Saga and Assistant Professor Simona Vasilache, for their valuable comments and constructive suggestions.

Besides, all the laboratory members, especially those in NERF team and WAVE team helped me a lot throughout my graduate student life, their kindness will never be forgotten.

Finally, I would thank my dear mother and father, without whose support I could never have this chance of studying abroad.

Bibliography

[BKJLP04] Doug Bowman, Ernst Kruijff, Joseph J. Jr. LaViola, and Ivan P. Poupyrev. 3D User Interfaces: Theory and Practice, CourseSmart eTextbook, 2004.

[Bol80] Richard A. Bolt. "Put-that-there": Voice and Gesture at the Graphics Interface. *SIGGRAPH Computer Graphics*, 14(3):262–270, July 1980.

[BW01] Doug A Bowman and Chadwick A Wingrave. Design and evaluation of menu systems for immersive virtual environments. In *Virtual Reality, 2001. Proceedings. IEEE*, pages 149–156. IEEE, 2001.

[BWCL01] Doug A. Bowman, Chadwick A. Wingrave, Joshua M. Campbell, and Vinh Q. Ly. Using Pinch Gloves™ for both Natural and Abstract Interaction Techniques in Virtual Environments. In *Proceedings HCI International 2001*, pages 629–633, 2001.

[CBJL09] Dustin B. Chertoff, Ross W. Byers, and Joseph J. Jr. LaViola. An Exploration of Menu Techniques Using a 3D Game Input Device. In *Proceedings of the 4th International Conference on Foundations of Digital Games*, FDG '09, pages 256–262, New York, NY, USA, 2009. ACM.

[Col14] Alex Colgan. How Does the Leap Motion Controllere Work? `http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work`, August 2014. Accessed December 19th 2016.

[CRDR11] Ankit Chaudhary, Jagdish Lal Raheja, Karen Das, and Sonia Raheja. Intelligent approaches to interact with machines using hand gesture recognition in natural way: a survey. *International Journal of Computer Science & Engineering Survey (IJCSES)*, 2(1):122–132, February 2011.

[CWL15] Wei-Lun Chen, Chih-Hung Wu, and Chang Hong Lin. Depth-based hand gesture recognition using hand movements and defects. In *2015 International Symposium on Next-Generation Electronics (ISNE)*, pages 1–4. IEEE, 2015.

[DL10] Barney Dalgarno and Mark J.W. Lee. What are the learning affordances of 3-D virtual environments? *British Journal of Educational Technology*, 41(1):10–32, 2010.

[DP14] Artem Dementyev and Joseph A. Paradiso. WristFlex: Low-power Gesture Input with Wrist-worn Pressure Sensors. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 161–166, New York, NY, USA, 2014. ACM.

[DSD08] L. Dipietro, A. M. Sabatini, and P. Dario. A Survey of Glove-Based Systems and Their Applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(4):461–482, July 2008.

[FSMK10] Kentaro Fukuchi, Toshiki Sato, Haruko Mamiya, and Hideki Koike. Pac-pac: pinching gesture recognition for tabletop entertainment system. In *Proceedings of the International Conference on Advanced Visual Interfaces*, pages 267–273. ACM, 2010.

[GN12] François Guimbretière and Chau Nguyen. Bimanual Marking Menu for Near Surface Interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 825–828, New York, NY, USA, 2012. ACM.

[Gui87] Yves Guiard. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *Journal of motor behavior*, 19(4):486–517, 1987.

[GW00] François Guimbretiére and Terry Winograd. FlowMenu: Combining Command, Text, and Data Entry. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, UIST '00, pages 213–216, New York, NY, USA, 2000. ACM.

[GWB04] Tovi Grossman, Daniel Wigdor, and Ravin Balakrishnan. Multi-finger Gestural Interaction with 3D Volumetric Displays. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, UIST '04, pages 61–70, New York, NY, USA, 2004. ACM.

[HAR95] John Paulin Hansen, Allan W Andersen, and Peter Roed. Eye-gaze control of multimedia systems. *Advances in Human Factors/Ergonomics*, 20:37–42, 1995.

[HCY+16] Da-Yuan Huang, Liwei Chan, Shuo Yang, Fan Wang, Rong-Hao Liang, De-Nian Yang, Yi-Ping Hung, and Bing-Yu Chen. DigitSpace: Designing Thumb-to-Fingers Touch Interfaces for One-Handed and Eyes-Free Interactions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 1526–1537, New York, NY, USA, 2016. ACM.

[HPPK98] Ken Hinckley, Randy Pausch, Dennis Proffitt, and Neal F. Kassell. Two-handed Virtual Manipulation. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 5(3):260–302, September 1998.

[JPG16] Alvin Jude, G. Michael Poor, and Darren Guinness. Grasp, Grab or Pinch? Identifying User Preference for In-Air Gestural Manipulation. In *Proceedings of the 2016 Symposium on Spatial User Interaction*, SUI '16, pages 219–219, New York, NY, USA, 2016. ACM.

[KB93] Gordon Kurtenbach and William Buxton. The Limits of Expert Performance Using Hierarchic Marking Menus. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, pages 482–487, New York, NY, USA, 1993. ACM.

[KKKA12] Cem Keskin, Furkan Kiraç, Yunus Emre Kara, and Laie Akarun. Randomized decision forests for static and dynamic hand shape classification. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 31–36. IEEE, 2012.

[KL14] Arun Kulshreshth and Joseph J. LaViola, Jr.. Exploring the Usefulness of Finger-based 3D Gesture Menu Selection. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 1093–1102, New York, NY, USA, 2014. ACM.

[KU16] Satoshi Kamaishi and Ryuya Uda. Biometric Authentication by Handwriting Using Leap Motion. In *Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication*, IMCOM '16, pages 36:1–36:5, New York, NY, USA, 2016. ACM.

[Kur93] Gordon Paul Kurtenbach. *The design and evaluation of marking menus*. PhD thesis, University of Toronto, 1993.

[KZL12] Alexey Kurakin, Zhengyou Zhang, and Zicheng Liu. A real time system for dynamic hand gesture recognition with a depth sensor. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 1975–1979. IEEE, 2012.

[LC11] Luigi Lamberti and Francesco Camastra. Real-time hand gesture recognition using a color glove. In *International Conference on Image Analysis and Processing*, pages 365–373. Springer, 2011.

[Lik32] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

[LLM13] Julien-Charles Lévesque, Denis Laurendeau, and Marielle Mokhtari. An asymmetric bimanual gestural interface for immersive virtual environments. In *International Conference on Virtual, Augmented and Mixed Reality*, pages 192–201. Springer, 2013.

[LT13] Unseok Lee and Jiro Tanaka. Finger identification and hand gesture recognition techniques for natural user interface. In *Proceedings of the 11th Asia Pacific Conference on Computer Human Interaction*, pages 274–279. ACM, 2013.

[LWT+15] Po-Wei Lee, Han-Yu Wang, Ying-Chao Tung, Jhe-Wei Lin, and Andries Valstar. TranSection: Hand-Based Interaction for Playing a Game Within a Virtual Reality Game. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '15, pages 73–76, New York, NY, USA, 2015. ACM.

[MK16] Rajesh B. Mapari and Govind Kharat. American Static Signs Recognition Using Leap Motion Sensor. In *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*, ICTCS '16, pages 67:1–67:5, New York, NY, USA, 2016. ACM.

[MT15] Christiane Moser and Manfred Tscheligi. Physics-based Gaming: Exploring Touch vs. Mid-air Gesture Input. In *Proceedings of the 14th International Conference on Interaction Design and Children*, IDC '15, pages 291–294, New York, NY, USA, 2015. ACM.

[MZ11] David Minnen and Zahoor Zafrulla. Towards robust cross-user hand tracking and shape recognition. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1235–1241. IEEE, 2011.

[ORS12] Serban Oprisescu, Christoph Rasche, and Bochao Su. Automatic static hand gesture recognition using ToF cameras. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 2748–2751. IEEE, 2012.

[PAC13] Leigh Ellen Potter, Jake Araullo, and Lewis Carter. The Leap Motion Controller: A View on Sign Language. In *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*, OzCHI '13, pages 175–178, New York, NY, USA, 2013. ACM.

[Pre14] Prashan Premaratne. Historical Development of Hand Gesture Recognition. In *Human Computer Interaction Using Hand Gestures*, pages 5–29. Springer, 2014.

[RMB08] Tao Ni Ryan, P. McMahan, and Doug A. Bowman. Tech-note: rapMenu: remote menu selection using freehand gestural input. In *3D User Interfaces, 2008. 3DUI 2008. IEEE Symposium on*, pages 55–58. IEEE, 2008.

[RO12] Gang Ren and Eamonn O'Neill. 3D marking menu selection with freehand gestures. In *3D User Interfaces (3DUI), 2012 IEEE Symposium on*, pages 61–68. IEEE, 2012.

[SKA+06] Hiroshi Sasaki, Tomohiro Kuroda, Peter Antoniac, Yoshitsugu Manabe, and Kunihiro Chihara. Hand-menu system: a deviceless virtual input interface for wearable computers. *Journal of Control Engineering and Applied Informatics*, 8(2):44–53, 2006.

[SMT12] Yu Suzuki, Kazuo Misue, and Jiro Tanaka. A Potential Exploration of Finger-Specific Interaction. In *Proceedings of the 10th Asia Pacific Conference on Computer Human Interaction (APCHI12)*, pages 389–392, 2012.

[SN16] Merkourios Simos and Nikolaos Nikolaidis. Greek Sign Language Alphabet Recognition Using the Leap Motion Device. In *Proceedings of the 9th Hellenic Conference on Artificial Intelligence*, SETN '16, pages 34:1–34:4, New York, NY, USA, 2016. ACM.

[SS15] Praveen Kumar Sharma and Shreya Sharma. Evolution of Hand Gesture Recognition: A Review. *International Journal of Engineering and Computer Science ISSN*, 4(1):9962–9965, January 2015.

[SZ94] David J. Sturman and David Zeltzer. A Survey of Glove-based Input. *IEEE Computer Graphics and Applications*, 14(1):30–39, January 1994.

[TRR13] Paulo Trigueiros, Fernando Ribeiro, and Luís Paulo Reis. Vision-based gesture recognition system for humancomputer interaction. *Computational Vision and Medical Image Processing IV: VIPIMAGE*, pages 137–142, 2013.

[TWHH16] Hsin-Ruey Tsai, Cheng-Yuan Wu, Lee-Ting Huang, and Yi-Ping Hung. ThumbRing: Private Interactions Using One-handed Thumb Motion Input on Finger Segments. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, MobileHCI '16, pages 791–798, New York, NY, USA, 2016. ACM.

[VB05] Daniel Vogel and Ravin Balakrishnan. Distant Freehand Pointing and Clicking on Very Large, High Resolution Displays. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*, UIST '05, pages 33–42, New York, NY, USA, 2005. ACM.

[WCL16] Chih-Hung Wu, Wei-Lun Chen, and Chang Hong Lin. Depth-based Hand Gesture Recognition. *Multimedia Tools and Applications*, 75(12):7065–7086, June 2016.

[WHM12] Julie Wagner, Stéphane Huot, and Wendy Mackay. BiTouch and BiPad: Designing Bimanual Interaction for Hand-held Tablets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 2317–2326, New York, NY, USA, 2012. ACM.

[Wil06] Andrew D. Wilson. Robust Computer Vision-based Detection of Pinching for One and Two-handed Gesture Input. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, UIST '06, pages 255–258, New York, NY, USA, 2006. ACM.

[WP03] Andrew Wilson and Hubert Pham. Pointing in Intelligent Environments with the WorldCursor. In *INTERACT*, 2003.

Appendices

This part gives the informed consent form and questionnaires we used in the experiments.

Appendix A Informed Consent Form

Informed Consent Form

Experiment Purpose

The purpose of this experiment is to compare the performance of our proposed gesture based menu selection technique - the Finger-to-Thumb system with the traditional Move-and-Tap system. And, to find whether Finger-to-Thumb can achieve eyes-free interaction.

Participants will be asked to use both these two menu selection techniques and fill out an anonymous questionnaire after the experiment. The experiment will last approximately half an hour, but you can choose to stop at any time.

Please note that none of the experimental tasks is a test of your personal intelligence or ability. The objective is to test the performance and usability of our research system.

Confidentiality

During the experiment, the system will automatically record the time you spent on each task, and an experimenter will keep observing the whole procedure to record the total number of gestures you have done.

All data will be no related to your personal information so that your anonymity will be protected in any research papers and presentations that result from this work.

Record of Consent

Your signature below indicates that you have understood the information about our experiment and consent to your participation. The participation is voluntary and you may refuse to answer certain questions on the questionnaire and withdraw from the study at any time with no penalty. If you have further questions related to this research, please contact the researcher.

Participant: Date:

Experimenter: Date:

Appendix B Experiment 1 Questionnaire

Experiment 1 Questionnaire

Thank you very much for participating the experiments. We would like to ask you a few questions about your experience during the experiments. Since this is an anonymous questionnaire, there is no need to worry about privacy problems.

Information about you:

1. What is your age?

2. What is your gender? Male Female

3. Are you left-handed or right-handed? Left Right

4. How familiar are you with gestural interfaces?
   A. Never used B. Rarely used
   C. Occasionally used D. Frequently used

Questions about your experience:

Please answer the following questions on a scale of 1-5, where 1 is **Strongly Disagree** and 5 is **Strongly Agree**. Please circle the number that represents your best answer.

1. I felt it was easy to execute the touch gestures.

   1 2 3 4 5

2. I felt it was easy to execute the swipe gestures.

   1 2 3 4 5

3. I felt tired after the experiment.

   1 2 3 4 5

If you have any comments or suggestion, please write them down below:

Appendix C Experiment 2 Questionnaire

Experiment 2 Questionnaire

Thank you very much for participating the experiments. We would like to ask you a few questions about your experience during the experiments. Since this is an anonymous questionnaire, there is no need to worry about privacy problems.

Information about you:

1. What is your age?

2. What is your gender? Male Female

3. Are you left-handed or right-handed? Left Right

4. How familiar are you with gestural interfaces?
   A. Never used B. Rarely used
   C. Occasionally used D. Frequently used

Questions about your experience:

Please answer the following questions on a scale of 1-5, where 1 is **Strongly Disagree** and 5 is **Strongly Agree**. Please circle the number that represents your best answer.

1. I felt tired after using the Finger-to-Thumb system.

   1 2 3 4 5

2. I felt tired after using the Move-and-Tap system.

   1 2 3 4 5

3. I prefer the Finger-to-Thumb system to the Move-and-Tap system.

   1 2 3 4 5

If you have any comments or suggestion, please write them down below: