Finger identification-based hand gestures and point cloud-based 3D motion gestures for Natural User Interface

Graduate School of Systems and Information Engineering

University of Tsukuba

March ２０１６

Unseok Lee

Abstract

In recent years, many researches have been proposed about gestures in 2D/3D for designing Natural User Interface(NUI). In NUI, the needs for the communication methods have been evolved; it aims at more natural way to communicate with computers. Recently, with developments of technologies, the direction for designing the NUI is changing rapidly; our body became the interface to communicate with the computers. This is the new trend for designing the interfaces. On this point, we focused on hand gestures and 3D motion gestures to design more natural interfaces. Our research focused on how can we design more natural interfaces with powerful and robust gesture recognition.

Our research is composed of two main parts. A finger identification-based hand gestures and point cloud-based 3D motion gestures. We aim at designing the use of NUI easier both beginners and experts, and designing the interactions occur as naturally as possible, without having the users go through an artificial step such as training.

Finger identification-based hand gestures aim at designing natural hand gestures with robust recognition. Also, it aims at overcoming previous hand gesture recognition problems, such as gestures being too simple when using a depth camera and lack of robustness in vision-based approaches. We designed three classifiers for overcoming these problems; finger counting classifier, finger identification classifier and vector matching classifier. Our system recognizes natural hand poses and gestures through the classifiers. The natural interactions were implemented using the proposed gesture recognition system.

We designed applications such as Air-drawing, Multi-touch, Mapping and Finger mouse interface using the system. The interfaces provided natural and intuitive interactions with gestures. Also, good gesture recognition results are obtained using our proposed system. The results demonstrated robustness of our recognition system and naturalness of our proposed interfaces.

However, we found out that our proposed system has limitations. The system works well with 2D gestures, such as directions, but is not sensitive enough to detect complex 3D gestures. Also, interactions between hands and objects in 2D have many limitations, thus we decided to improve our system from 2D gestures to 3D motion gestures.

We designed point cloud-based 3D motion gesture interface for overcoming the limitations. By improving the system, various directional gestures (e.g., upper-right, upper-left, ...) can be designed, and the directions can be discriminated by their angles and movement speed.

The interface, in short, can provide more sensitive gestures and robust gesture recognition using the 3D point cloud approach. For these, we implemented a MOtion Interface framework ("MOI toolkit"). It provides one-hand, two-hand and hand-object motion gesture recognition as libraries. Also, 3D object tracking and collision detection function are provided as modules in the toolkit. The toolkit provides pre-defined (e.g., swipe, multi-touch, ...) gestures. Our system can learn new motion gestures (i.e., user-defined gesture) as well, because it uses a powerful 3D processing library and machine learning algorithms. The developers who do not have knowledge about 3D gestures can easily map the gestures to their own application. It can be expected to make promising natural user interfaces with our open toolkit.

We designed simple NUI applications using our toolkit, such as Touch-Pair, Google Street View controller and Google Earth controller. The applications can be used as guidelines how we can use the toolkit for a natural user interface. Our final goals are adding more gesture types such as object-object, body-object in the next version of MOI toolkit. This way, the toolkit can be more useful for users and developers.

In result, we can summarize our contributions as three things. First, we proposed finger identification based robust hand gesture recognition method. We obtained that improved hand gesture recognition accuracy and increased hand gesture vocabularies using the method. Second, we proposed point cloud based 3D motion gesture interface. It has improved our finger identification based approach. We obtained more robust gesture recognition accuracy using the interface. Also, more various and sensitive hand gestures can be designed by improving finger identification based method(2D) to point cloud based method(3D). Finally, we designed natural user interfaces using the proposed method. We proposed natural and intuitive interactions in finger identification based approach (i.e., Air-drawing, Multi-touch, Mapping and Finger mouse), also implemented 3D interactions in point cloud based approach (i.e., Touch-Pair, Application controller). We believe that more natural, intuitive and immersive interfaces can be designed using our proposed techniques.

Acknowledgments

I am heartily thankful to Professor Jiro Tanaka, my thesis supervisor, for his many valuable suggestions, precise directions, and kind encouragement. I also greatly thank Associate Professor Shin Takahashi, Associate Professor Buntarou Shizuki and Assistant Professor Simona Vasilache for their kind advice and suggestions regarding this research. I would like to thank Professor Kazuhiro Fukui, Professor Akihisa Ohya and Professor Tomoo Inoue for their many useful comments on this research. I am also grateful to all the members of the IPLAB, Interactive Programming Laboratory, University of Tsukuba, for giving me many opportunities to discuss my research with them and helping me in my experiments. I would like to thank my wife (Minseon Kim) for her continuous support and encouragement. Finally, I extend my sincere gratitude to my parents and family members for all their efforts and their wishes.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Many low-cost depth cameras have released in recent years (e.g., Microsoft Kinect, Intel RealSense, ...). Also, technologies for the devices have been developed rapidly, such as a Microsoft Kinect Software Development Kit(SDK) [5] and Intel RealSense SDK [6]. These SDKs provide powerful modules, such as a hand recognition, a body recognition and a speech recognition. All of the provided modules are for designing Natural User Interfaces(NUI) finally [7]. The NUI means some interface (or some system) that is easy to use for both beginners and experts, and easy to learn how to control the systems [8]. In detail, it is not users trying to use the interfaces, but the interfaces come to users naturally. However, users easy to use with traditional interaction methods like mice and keyboards are often unwilling to change to new alternative interfaces. Ideally, new interfaces should be designed more accessible to users without requiring long periods of learning and adaptation [9].
In short, the natural user interface has to design intuitively, naturally and easily to use on the users' side [10]. For example, we can play tennis games or do web shopping by using the hand and body tracking technologies, these technologies lead to the users reality and immersion of the interactions.
Also, the 2D and 3D processing technologies have been developed in recent years (e.g., OpenCV [11], Point Cloud Library(PCL) [12] , ...). New possibilities are provided in NUI filed with the development of these technologies. Especially, the release of PCL is providing simple methods to handle 3D processing. It means that even the users who do not know 3D processing well, they can design NUI with the 3D processing. Therefore, many researches [13] [14] are proposed using the 3D processing technologies. These researches provide three dimensional, immersive and intuitive interactions. By using three dimensional processing for

hand tracking, especially, we can design various directional hand gestures, and also design three dimensional interactions between hands and objects. More intuitive and immersive interfaces are expected to use these methods.

In this thesis, we focus on designing NUI using finger identification based hand gesture and point cloud based 3D motion gesture recognition methods. All of the proposed methods are for designing more natural, immersive and intuitive interfaces. We will discuss the reasons that use hand gesture recognition with finger identification, and 3D motion gestures with point cloud. Also, we will describe robustness of our method and naturalness of our interfaces in detail.

1.1 Motivation and research goal

1.1.1 Finger identification-based hand gesture interactions

Interfaces using hand gestures are a popular field in Human Computer Interaction(HCI). Many researches have already been proposed in this field [15], [16]. They mostly provided hand gesture recognition using vision-based, glove-based and depth-based. However, vision-based recognition is highly influenced from rough condition (e.g., noise, dark environment). The glove-based approach is unnatural, because users have to wear the devices.
Recently, a new possibility is provided to HCI field with the development of the sensors such as Kinect and Leap Motion. This development has made possible robust hand gesture recognition in bad condition such as dark environment and rough background, thus many depth-based hand gesture recognition researches proposed recently [17], [18]. However, almost all of the proposed researches are simple and unfamiliar gestures [19], [20].
In this research, we propose a new finger identification method and hand gesture recognition [21]. Our research aims at making hand gesture recognition more robust. Our proposed method is provided for designing of more natural and intuitive interfaces with gestures. We propose drawing interaction, multi-touch interaction, mapping interaction and finger mouse interaction using the proposed method. We evaluate our proposed method and each interaction to demonstrate the usability and robustness of our method.

1.1.2 Point cloud-based 3D motion gestures and interactions

We have already proposed finger identification-based hand gestures and the interactions using our proposed method. However, we found out that our research has limitations. In this method, hand gestures are designed in 2D, so the method provided simple gestures only compared with 3D hand gestures. We analyzed related researches of gestures in 3D even our finger identification provided robustness and natural feeling in 2D; advantages of the 3D gestures, practical usages. By analyzing the advantages of the 3D gestures, we decided to improve our previous method (i.e., finger identification-based approach); 2D to 3D. The expected advantages of 3D gestures are as below:

We can expect more sensitive gestures with 3D. The hand gestures can be designed in various directions (e.g., front, back, up, down, left, right, upper-right, upper-left,...). We can also expect various hand gesture combinations with their directions and movements. The motion gestures can be in 3D, which means that we can distinguish gestures by hand movements as well. Another possibility in 3D motions is hand-object interaction. We can expect to make tangible interaction with objects, and motion gestures using a depth sensor. Another motivation for this research is the already developed robust 3D processing libraries. These are designed to use 3D processing for robot perception mainly. Using these libraries has many possibilities. Because of many powerful 3D processing in the libraries, we can make robust and interesting interactions for human with a depth camera.

To overcome the above mentioned limitations and apply the advantages to our research, we decide to make 3D motion gesture interface with a toolkit("MOI toolkit"). It is provided as libraries, and has powerful functions with point cloud and gesture recognition libraries. Our proposed toolkit is aimed at developers who want to use 3D motion and map them to their own applications easily. One hand motion, two hands and hand-object motion gestures are provided in the library.

We designed a hand-object interface example, using our toolkit. It is called "Touch-Pair" interface. It aims at easily connecting the cyber world(digital object) and the real world(physical object) for the interaction.

1.2 Dissertation organization

The presented dissertation is structured as follows. Chapter 1 is an introduction that the scope of the dissertation. In Chapter 2, we introduce the finger identification-based hand gestures which designs interfaces more naturally and intuitively. Chapter 3 introduces point cloud-based 3D motion gestures and useful tools to use motion gestures easily. Finally, Chapter 4 provides our research contributions and expected future works.

Chapter 2

Finger identification-based hand gesture interface

This chapter describes the design of a finger identification-based hand gesture interface, our proposed prototype of the interfaces for interactions. The design and architecture of the interfaces are presented, including detailed descriptions of the system algorithms. We described the reason that uses a finger identification algorithm in the first steps. We conducted an experiment to evaluate the usability factors of interaction accuracy and gesture classification accuracy. Finally, we described discussion about the successes and limitations of our approach, and conclude with the results.

Our themes are: design a new finger identification method for improving hand gesture vocabularies and robustness of gesture recognition; propose a gesture version new interactions using a depth camera.

2.1 Importance of finger identification

The interface using the hand gesture is a popular field in Human Computer Interaction(HCI). Recently, new horizons are open to the HCI field with the development of sensors and technology [22] such as Kinect, Depth-Sense and Leap motion. This development has made possible robust hand recognition in bad conditions such as dark light and rough background. This depth-based sensors and technologies provide a robust recognition [23], but there are limitations in the hand gesture vocabulary. The proposed hand gesture recognition researches have these limitations. In current fingertip tracking in the hand gesture

researches, for instance, the input fingertip recognition of the index and middle fingers are interpreted in the same manner. However, we can increase the interaction bandwidth of hand gesture in a depth camera by using finger identification method. Because the method provides additional information (i.e., one of the fingers: thumb, index, middle, ring or little) to the system.

In short, by discriminating fingers, the finger input vocabulary volume is increased, and various finger combination inputs are possible, especially multi-touch gestures (e.g., multi-finger interaction [24], [25]). We believe that the proposed method leads more possibilities for designing natural interfaces.

We focused on two advantages to improve hand gesture recognition and design natural interfaces:

Advantage 1: The hand gesture input vocabularies are increased

Advantage 2: Natural multi-touch gestures can be designed using our method

The first advantage is because of increasing the combination of input fingers (i.e., one fingertip, multiple fingertips) [26], [27]. Take for example, a two-fingertip multiple gestures performed with the thumb and index finger, and the index finger and middle finger. The five completely different gesture inputs can be recognized in two fingertips case, while the current researches being used would recognize them as same input. It means that exists 31 different cases of hand gestures with fingertips (i.e., 5 cases for one and four fingertips, 10 cases for two and three fingertips and 1 case for five fingertips). We do not use all of the cases, but we mapped the gesture cases to multi-touch gestures. The second advantage is that more various multi-touch interactions [8] can be designed and mapped using our proposed method, also these gestures are familiar to use and easy to memorize, because the gestures are used on the smart phone. The hand gesture recognition accuracy is increased using our finger identification method, due to identifying fingers exactly.

In other word, the hand gesture recognition robustness and the hand gesture vocabularies are increased using the finger identification method, finally, the better NUI can be designed based on the method. Therefore, the method is important in hand gestures and NUI filed.

2.2 Design goals

In natural user interface, *natural* refers to the user's behavior and feeling during the experience. The natural user interface must mirror users' capabilities, meet users' needs, take full advantage of users' capacities and fit users' task and context demands [8]. And the NUI experiences focus on the joy of doing; the pleasure comes from the interaction, not the accomplishment.

There are many basic design principles of designing NUI except to we mentioned. In this chapter, we focus on implementing finger identification based hand gestures for designing NUI using the 3D depth-sensing input modalities.

Our system has two main principles of design:

1. Design finger identification based hand gestures: We design the finger identification based hand gestures; our system can provide many gesture input vocabularies by the method, and increase the accuracy of hand gesture recognition. We can provide a natural feeling of use, and intuitiveness.

2. Design hand gestures with the example interfaces: The example interfaces with gestures are provided using finger identification method. We design the gestures ensuring the each interaction 'feels' natural and familiar such as multi-touch, swipe gestures. The designed examples of interface using our gestures showed similar feedbacks with the existing interface (i.e., multi-touch on the screen surface) when we perform the gestures, however, the interface using our gestures present different feeling. In short, we design the familiar gestures (i.e., popular gesture) with a new feeling of use.

2.3 Related work

Many hand gesture recognition techniques are proposed for natural user interface (NUI) [28], [29], [30]. The vision-based, glove-based and depth based approaches are proposed in many researches. In this section, we describe the related works in details and their limitations, also the advantage of our proposed method with the works.

2.3.1 Vision based hand gesture recognition

Employing hand gesture recognition for HCI, such interfaces have been studied and developed by many researchers over the past 30 years [31]. Many methods of vision based

hand gesture recognition are proposed in many areas [32]. The main areas can be divided four classes: medical systems and assistive technologies; crisis management and disaster relief; entertainment; and human-robot interaction [9].

Chu et al. [33] presented a self-portrait interface using vision based hand gesture recognition with fingertip tracking. They provided interesting interaction to control camera by the user's hand gestures. The system segmented hands and face by skin color. The fingertips were tracked by segmented hands, then the users can control the self-portrait interface. The interface itself is useful, but the robustness of recognition are decreased in many conditions such as darkness, black and white man hands and hands next to the face, because they used skin color segmentation. In addition, system functions are limited when user moves far from the camera because they do not use depth data for recognition. Also, the small depth sensors are released [34], so that the depth based hand gesture can be applied to the self-portrait interface.

Yeo et al. [35] presented hand tracking and gesture recognition system using low-cost hardware. They provided fingertip tracking with removing a face by using skin color segmentation and Haar-cascade face detection. Also, they provided static hand gesture recognition. It is fast and useful for the hand gesture recognition. However, it does not provide finger identification, and still have the limitations in many conditions as we mentioned above a Chu's research.

A hand tracking and recognition system[36] presented fingertips tracking based on color recognition. The system is initialized by sampling colors from the hand (i.e., color profile), thus the system overcame the limitations about differences of color by private hands (e.g., black man). The system implemented good recognition results, but it still has limitations about weakness on light, also does not provide finger identification.

2.3.2 Glove based hand gesture recognition

The glove based hand gesture recognition systems are proposed, because it it simple to set up and fast for recognition. On the topic of hand gesture recognition, the input device must capture the position of each finger on the hand, as well as the hand's orientation in its rotation axis (i.e., several degrees of freedom of movement). In this point, the glove based approaches are robust to recognize.

Kenn et al. [37] presented an interface for wearable computing applications using glove based finger recognition. His paper implemented finger identification for hand gesture. The research provided rich user experience based on accurate recognition rate. However, it is not practical in all situations because the user cannot always carry a glove. The glove is both unnatural and heavy. This condition is not suitable for natural interaction.

Wang et al. [38] presented a real-time hand tracking system with a color glove. It can reconstruct the pose of the hand from a single image of the hand wearing multi-colored glove. A user wears the glove that they designed, then the single camera capture the scene. Because the system can reconstruct a hand with 26 degrees of freedom, the system is robust to hand tracking. Also, the research demonstrated applications that finger spelling, animation control by using their method. It demonstrated good results. However, the system did not design the finger identification method and has to wear the glove, so that the system has still has limitations for NUI designs.

2.3.3 Depth based hand gesture recognition

After the Kinect is released on Nov. 2010, many interesting hand gesture recognition researched are implemented and proposed. Because the low-cost sensor provides color and depth information simultaneously, we can apply computer vision processing in 2D and 3D technologies to our data from the sensor.

Yang et al. [39] proposed a hand motion gesture recognition system using a Kinect depth data. Main proposal is a hand tracking algorithm in depth image by calculating a hand weighted probability. They proposed hand motion gestures such as wave, forward/backward, move up/down and left/right into a media player application using the algorithm. This system demonstrated the possibility by using Kinect for hand motion gesture recognition in a contactless UI. However, the implemented system was not able to recognize fingertip and identify fingers. Therefore, they were not able to provide robust and natural gestures, such as pinch gesture, spread gesture and flick gesture.

Raheja et al. [40] proposed a method to recognize and track fingertips and center of palm using a Kinect. Their proposed system tracks the fingertip by calculating depth image segmentation of hand regions, and applying a big circular filter for dividing a palm of a hand, finally finding the fingertips by using the distance map of a hand. The tracking showed good results, however the system has the weakness of users' movements (e.g., move

a hand front or back). Also, they do not implement any interactions, hand gestures and the finger identification. Thus, this approach is not enough for NUI designing.

Tang [41] presented hand gesture recognition using a Kinect. He used full body tracker developed by researchers for classifying the region of hands, and then the system tracks the hand gestures by integrating color and depth information in a support vector machines library. Also, it provided the hand gesture recognition using the method. However, the system does not evaluate robustness of the recognition, and the proposed hand gestures are unnatural and simple (i.e., open or close a hand), while our proposed methods can be designed various and natural hand gestures.

Ren et al. [22] presented a hand gesture recognition system with a Kinect. The proposed system consists of two major modules, hand detection and gesture recognition. It detects the hand shape using depth and color information. They applied a novel shape distance metric for improving hand gesture recognition because the Kinect only provides the low resolution of depth information. They implemented demos (i.e., a rock-paper-scissors game and arithmetic computation) for demonstrating the performance of the system in two real-life applications. However, the system cannot recognize the direction of the hands because it used hand shape-based detection without vector matching. Also, it cannot discriminate fingers, means that cannot recognize complicate hand gestures.

In short, most of the proposed hand gesture interaction techniques have limitations in designing natural interactions, that is to say our proposals aim to overcome all the problems we mentioned above.

2.4 Our proposed techniques and interfaces

In this section, we describe our hand gesture recognition techniques in details and interfaces using the techniques. Our main points of recognition are based on the finger identification and vector matching method. By using the methods can improve the hand gesture recognition accuracy and input vocabularies.

2.4.1 Finger identification-based hand gesture recognition technique

2.4.1.1 Hand gesture recognition pipeline

Our proposed method has three different classifiers for robust hand gesture recognition [42]. The classifiers are composed of a finger counting classifier, a finger identification classifier and a vector matching classifier (see Figure 2.1). The finger counting classifier is composed of fingertips tracking method, and the finger identification classifier is composed of method for identification and hand pose estimation. Finally, the vector matching classifier is composed of distance matching, direction matching and degree matching methods. The hand gesture recognition is performed using Our proposed classifiers from the finger counting to the vector matching, the hand gesture recognition accuracy is improved by passing through all classifiers. Each classifier is described below in details.

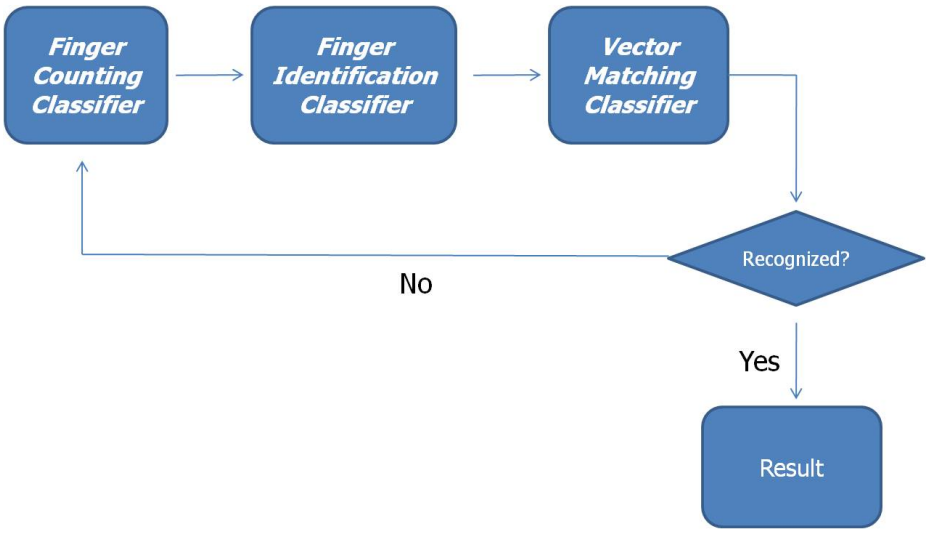

Figure 2.1: Our proposed recognition pipeline and Three classifiers

2.4.1.2 Fingertips tracking for finger counting classifier

Our system performs a depth segmentation for fingertip tracking. The system obtains depth data values from a Kinect, then those data values are segmented by pre-defined a threshold value(i.e., certain distance measured by the system). The depth pixel based clustering are performed using k-means with the segmented depth data values. In this step, the k is two, because our system separates the hands to two groups. The system

operates on a real time basis, and whenever there is the change of the input depth data source, the k-mean method is continuously applied. The contour detection is performed using tracing algorithm, and convex hull is calculated around the contour. Also, our system calculates palm of each hand with their center points. The fingertips are obtained by using the intersected point of the hand contour and the convex hull. The intersected points are candidate sets for fingertips. Finally, fingertips are chosen using three point alignment algorithm (see (a) of Figure 2.2). The finger counting classifier counts the number of fingertips, and passes to the finger identification classifier. It is an important process, because the finger number used as the basic data for robust hand gesture recognition in next classifier.

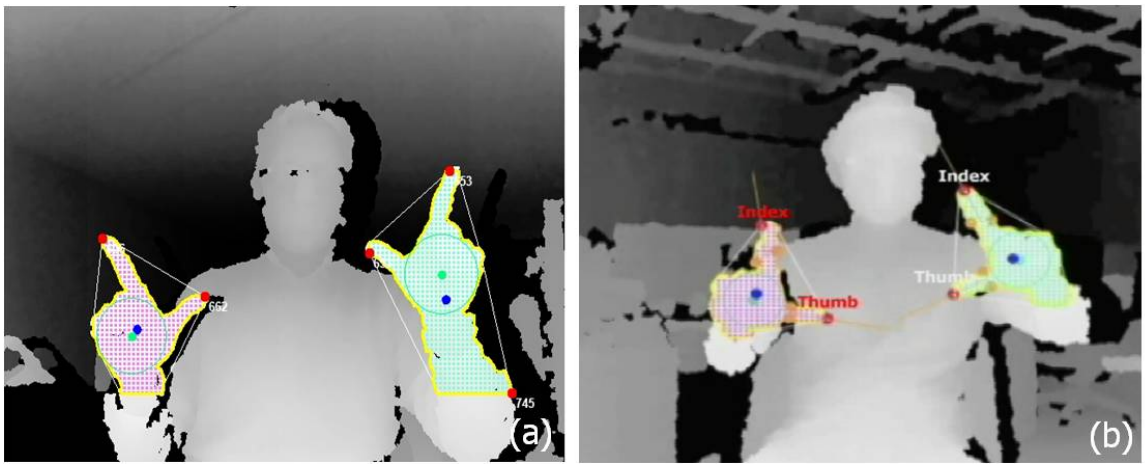

Figure 2.2: (a) Two fingertip tracking of each hand result and (b) Thumb and index finger identification result

White line: Convex hull, Yellow line: Contour, Red point: Fingertip
Green circle: Palm, Green point: Center of palm, Blue point: Center of a hand cluster
Yellow point of Figure 2.2 (b): Support calculating points

2.4.1.3 Finger identification classifier

After finger counting, we put the data to the finger identification classifier. The finger identification is a powerful method for robust hand gesture recognition, and can be extended to various promising gestures (i.e., input gesture vocabularies). Our system divided two cases to identify the fingers. It is based on the fingertips number.
The first case is under five fingertips (i.e., one to four fingertips). The hand pose estimation

is performed only this case. Our classifier has 30 different hand shape database. Because one fingertip and four fingertips have five different hand pose cases, two fingertips and three fingertips have 10 different hand pose cases. However, the cluster size of a hand can be difference per a person. So, we made five different cluster size to overcome this problem. In result, our system has 150 hands pose databases. For hand pose estimation, system performs three steps, that is finger counting, cluster size comparing (see (a) and (b) of Figure 2.3) and vector direction of a center point in the palm to fingertips (see (c) and (d) of Figure 2.3). The (b) of Figure 2.2 showed finger identification results.

On the other hand, the second case is when users extend five fingers. The system performs different calculations. The easiest method checks the thumb and index finger. Because, the distance of the thumb and forefinger is due to be the biggest among all adjacent fingers. The baby finger is distinguished the finger, which the head of a family which is apart from the thumb is far. The central finger is distinguished one which is the closest to forefinger in the meantime. The remaining one is the ring finger. The mentioned methods work well, except special cases. Robust and various gestures can be expected using the proposed method.

Algorithm 1 Hand Gesture Recognition Algorithm

Input: Segmented hand raw data ∈ Depth raw data
Output: Hand gesture name result

Initialization $FingerCount = 0$, $IdentificatedResult = 0$
$DetectedHandCluster = NULL$
$FingerCount = FingertipTracking(HandRawData)$
$IdentificatedResult = FingerIdentification(FingerCount)$
while $i \leq HandClusterDB.size$ **do**
  if ClusterSize(IdentificatedResult)= HandClusterDB.current.size **then**
    DetectedHandCluster=HandClusterDB.current
    break
  end if
  HandClusterDB.next
  i++
end while
while $j \leq DetectedHandCluster.size$ **do**
  if Distance(IdentificatedResult)=DetectedHandCluster[j].distance and
Degree(IdentificatedResult)=DetectedHandCluster[j].degree **then**
    return HandGestureName
  end if
  j++
end while

2.4.1.4 Vector matching classifier

This classifier designed for improving hand gesture recognition accuracy with the finger identification. The vector matching classifier recognizes by using three components; the direction, distance of each fingertip and degrees of fingertips. In addition, our system calculates the hand size (i.e., hand cluster size) to classify a group of database by size for improving recognition accuracy shown as (a) of Figure 2.3. The five different types of a fingertip in cluster size between 80 and 90 shown as (b) of Figure 2.3. Our system calculates same gestures from the database.

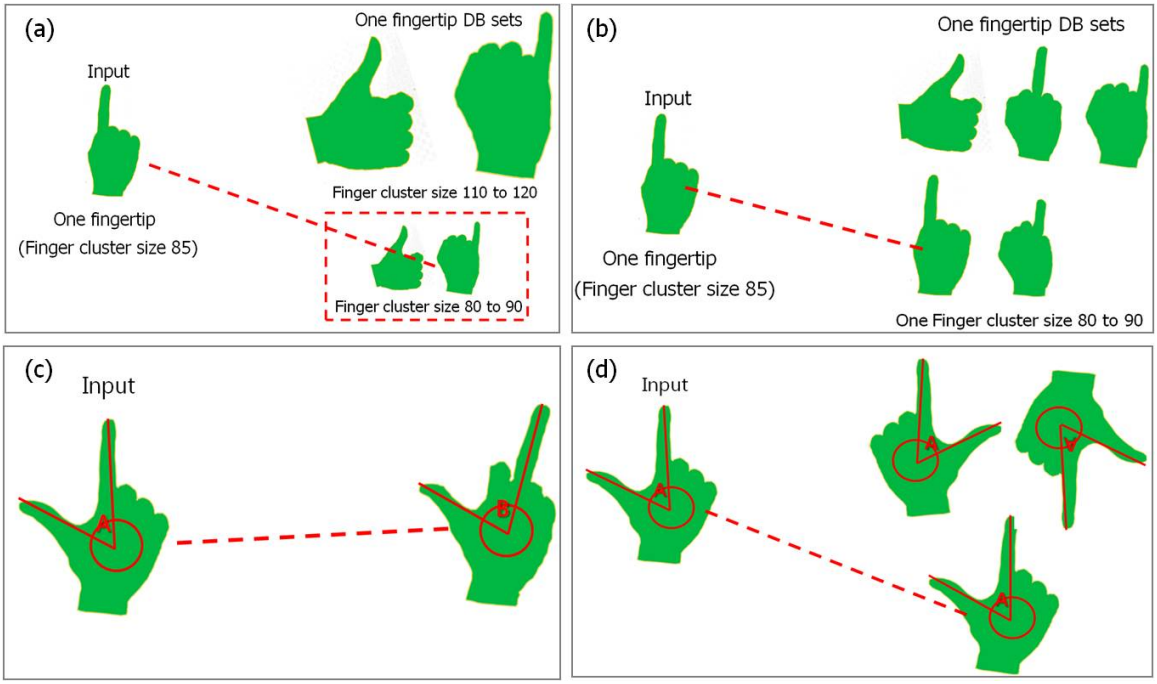

Figure 2.3: Example of hand gesture recognition using vector matching

(a) Finger cluster size matching: Find the matched cluster size in Databases
(b) Fingertip number matching: Detect the same pose of the hand in the Database
(c) Vector matching by degree: Compare the A and B
(d) Vector matching by direction: Compare the distance and direction of lines

Each component is used for discriminating similar gestures; for instance, in (c) of Figure 2.3, it is almost similar gesture when the system calculates. Thus, the system can discrim-

inate the small difference of the gestures using three components. From the direction and distance, a line is drawn based on a center point of palm to each fingertip. Our system calculates the distance and the direction of lines (e.g., upper-right, down,...). The cases of degree calculations are divided into three. That is, one vector degree of two fingertips case shown as (d) of Figure 2.3, two vector degrees of three fingertips cases, and three vector degrees of four fingertips cases; no vector degree in the one fingertip case, each vector is used to discriminate pre-define gestures by calculating the degrees; certain degrees are defined by the system.
In results, the gesture recognition accuracy improves through these three steps (i.e., distance, direction, degree).

2.4.2 Natural interfaces using our proposed technique

We explain our designed natural user interface such as air-drawing, multi-touch manipulation, mapping manipulation and finger mouse, and each interaction in the interfaces. Our proposed interfaces are used our proposed classifiers. We explained the interfaces and their interactions in detail.

2.4.2.1 Air-drawing interface

Our air-drawing interface implemented correctly finger painting interactions, by using depth-based finger gestures. We designed painting, drawing line gesture for these interactions. First, painting gesture is recognized when user extends his/her index finger, then the line is drawn along the path of movement. Second, drawing line gesture is recognized when user extends their thumb and index finger (see (a) of Figure 2.4), then the line is drawn depending on the coordinate value of the thumb and index finger. The same interactions are made in the case of using both hands at simultaneously.

2.4.2.2 Multi-touch interface

The multi-touch interface is implemented well mapped with multi-touch interactions on a touch screen (e.g., smart phone, smart pad). It provides resizing (i.e., pinch, spread gesture) and rotating natural interaction with the images. In order to resize the image, the system tracks the number of fingertips. When the system detects two fingertips (i.e., thumb and index finger of one hand or index finger of left and right hand), it changes to

resizing interaction mode. In resizing interaction mode, the system computes the distance between two fingertips. If the distance between the fingertips becomes larger than a certain value found experimentally, zoom-in interaction will be performed and the size of the image is expanded. On the contrary, if the distance between fingertips is shorter than a certain value, zoom-out interaction will be performed and the size of the image is reduced.
In order to rotate the image, the system computes each fingertip's position (i.e., thumb and index finger coordinates of one hand or index finger coordinates for left and right hand). If the left index finger is raised upwards and the right index finger is moved downward, then the image is rotated clockwise. On the contrary, if the right index finger is raised upwards and the left index finger is moved downwards, then the image is rotated counter-clockwise. When using only one hand, similar interactions are performed. If the thumb is raised upwards and the index finger is moved downwards, then the rotating interaction is performed in a clockwise (see (b) of Figure 2.4).

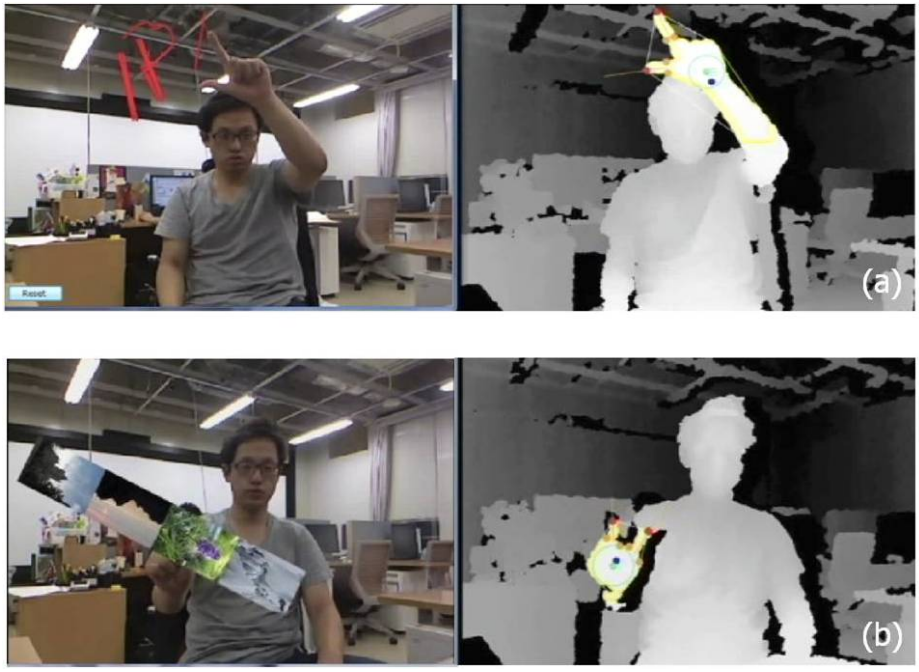

Figure 2.4: Our proposed interaction(1)

(a) Air-drawing interaction: Drawing a line using thumb and index finger
(b) Multi-touch interaction: Rotating interaction using thumb and index finger

2.4.2.3 Mapping interface

A mapping interface implements interactions with video surface on 3D space using depth data. We designed the gestures for mapping, selecting surface and time shift interaction with video player.

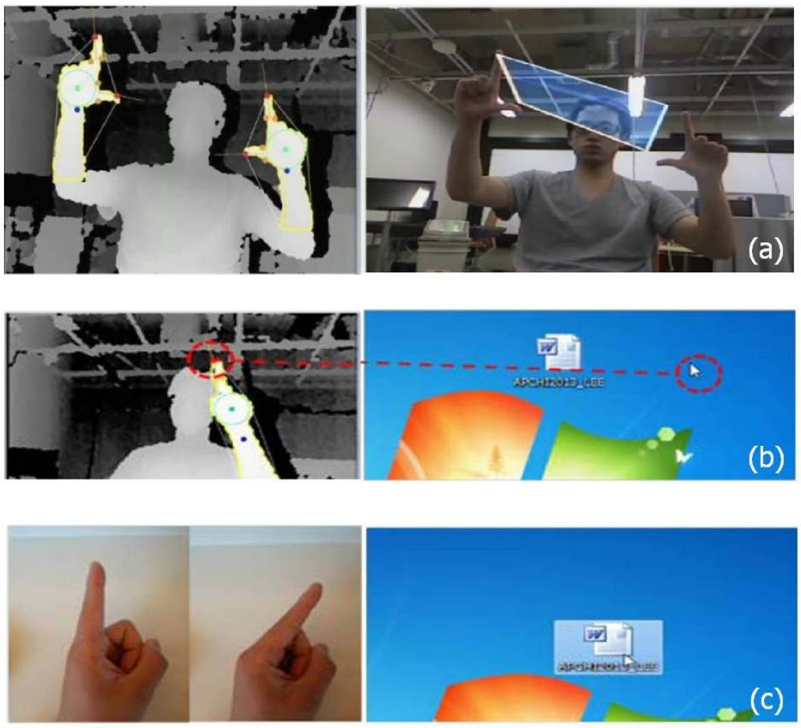

Figure 2.5: Our proposed interaction(2)

(a) Mapping interaction: Create video surfaces using thumb and index finger of each hand
(b) Finger mouse interaction: Drag mouse cursor using index finger
(c) Finger click interaction: Click moving the index finger forward and backward

Mapping interactions are made when the system detects two fingertips on each hand. The video surface to play the video is created depending on the coordinates of the two fingertips of each hand shown as (a) of Figure 2.5 (i.e., 4 sets of coordinates, thumb and index finger coordinate value for each hand). User can control the size of the surface and mapping position by moving four points (thumb and index of each hand). The surfaces depth value can be controlled by moving a hand or both hands forward/backward. We can create a new surface in front of the surface that was created before, when both hands are moved forward.

On the contrary, the new surface can be created behind the surface that was created before, when both hands are moved backward. In the case of moving a hand forward and the other hand backward, an almost diamond-shaped surface will be created. If shown from the front, it appears as a diamond surface. However, it can be shown as a rectangular surface when it is shown from different angles.
Selecting interactions occur when an index fingertip is on the surface that the user wants to control (i.e., it means that the depth value of index fingertip and the surface is same). This interaction is needed when multiple surfaces are created. The selected surface is shown with a white color border. Other interactions can be performed with the selected surface. Time shifting interaction with the selected surface occurs when the system detects five fingertips of the right hand and one fingertip of the left hand. The user can control the video player's time bar by moving his/her left fingertip.

2.4.2.4 Finger mouse interface

The finger mouse interface is implemented well mapped with mouse click interactions. It provides mouse functions using hand gestures (i.e., drag, click, double click gesture) and mouse right button interaction with L gesture using thumb and index finger recognition. This interface provides natural interaction with a computer.
In order to drag the mouse cursor, the system tracks the number of fingertips. When the system detects fingertip of index finger, it changes to the dragging interaction mode (see b of Figure 2.5). In dragging interaction mode, the system computes the coordination value of index fingertip. If the fingertips move to position, mouse cursor also moves in the same direction and position.
In order to perform click interaction, the system computes the depth value of index fingertip. If the index finger is moved forward and backward one time (see (c) of Figure 2.5), the mouse click interaction is performed. For mouse double click, the same process is needed with mouse click interaction. However, double click interaction needs to perform two times of click interaction quickly. It calculates the depth value of index fingertip as well. The mouse right button interaction is made when the use when the system detects thumb and index fingertips(L shape gesture).

2.5 System evaluation

2.5.1 Purpose

In this section, we evaluated the effectiveness of our proposed method; three classifiers, interactions. We conducted interaction accuracy experiments for evaluating the recognition accuracy of designed interactions, and gesture classification experiments for demonstrating the effectiveness of our classifiers. Finally, we conducted user questionaries for demonstrating naturalness and intuitiveness in user experience. We analyzed the results of all experiments, then found out the problems. In short, the purpose of the experiments is not only demonstrating the effectiveness of our method but also improving our system.

2.5.2 Interaction accuracy experiments

In this experiment, we evaluated recognition accuracy with our proposed method; finger identification and designed finger gestures. The experiments were performed on a computer with Intel Core i5 CPU 2.67GHz and 4.0 GB RAM, using a Microsoft Kinect for Xbox 360. We performed the experiments with ten volunteers. Our experiments are designed to evaluate six gestures; extending all fingers for finger identification, drawing, pinch, spread, rotate and mapping gesture. After thoroughly explaining all our gestures, each volunteer performed a gesture 100 times, in each condition; normal, dark, rough condition (i.e., rough condition means with noise or obstacles between back of hands). We checked whether the system recognized the gesture or not.

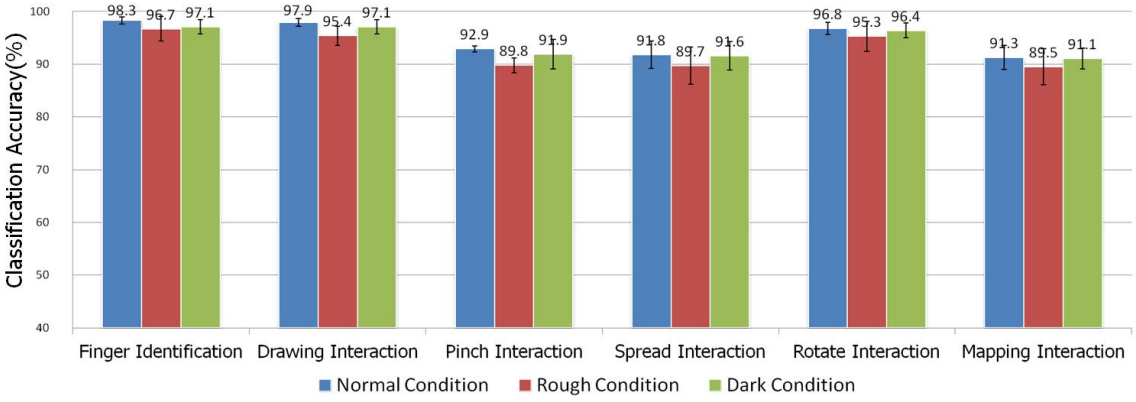

Figure 2.6: Interaction accuracy results

Our proposed method for finger identification and hand gesture recognition with depth data has shown high accuracy over 89 percentages for all gestures. Almost the same accuracy of average with normal and dark conditions, and relatively low accuracy in rough conditions (e.g., with noise or obstacles in the background). However, the system was not influenced at all by dark and rough condition. We found that finger identification, drawing and rotating gesture obtained a relatively high accuracy of recognition, because the coordination value of x and y is changed when the gestures are performed; not z-axis coordination value.
We found that pinch, spread and mapping interactions had a relatively low recognition rate. The pinch interaction was recognized as spread interaction, because users moved their hands rapidly, before the pinch interaction was recognized. The spread interaction was recognized as pinch interaction in the similar way. Also, the pinch and spread gestures' depth value is changed dynamically; when users performed the gesture by a hand, the x, y and z coordination value are changed rapidly.

In addition, the mapping interaction (see (a) of Figure 2.5) has a high time complexity compared to other gestures, because the system calculates depth data from fingertips on both hands. Therefore, the recognition accuracy is decreased in case the user moves their hand rapidly, but it can be improved by upgrading our system hardware or using GPU processing.

2.5.3 Gesture classification accuracy experiments

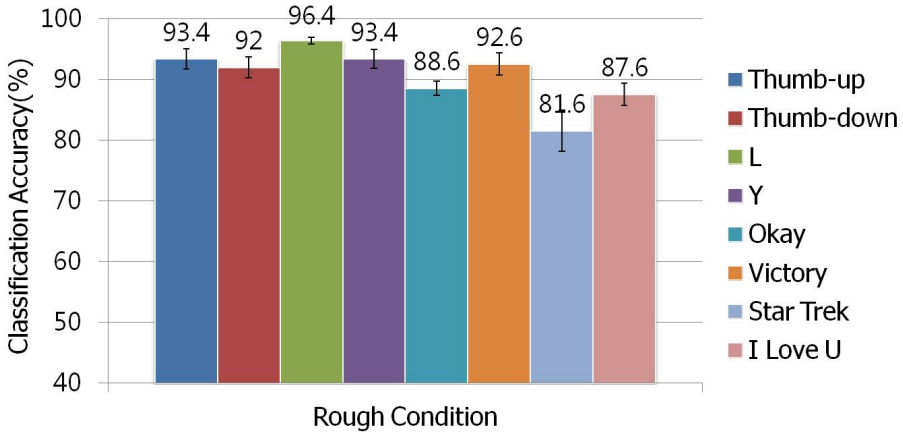

Figure 2.7: Gesture classification accuracy results

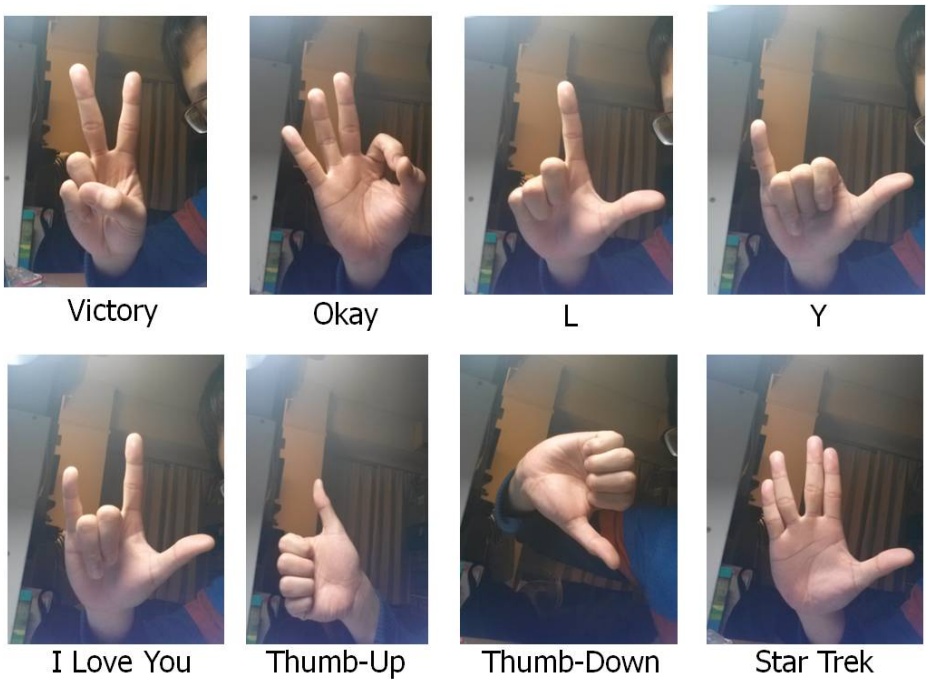

Figure 2.8: List of gestures [1]

To evaluate this experiment, five volunteers performed the eight gestures for 100 times: 50 times with the left hand and 50 times with the right hand. We evaluated eight gestures; Thumb-up, Thumb-down, L, Y, Okay, Victory, Star Trek [43], I Love U (see Figure 2.8); the gestures are designed [1] to demonstrate the effectiveness of our proposed three classifiers. We used finger counting, finger identification and vector matching (i.e., the degrees and distances of center point of palm and a fingertip) for evaluating each gestures.

The best case is the character 'L' having the accuracy of 96.4%. The worst case is the 'Star Trek' gesture having the accuracy of 81.6%. The Star Trek gesture the number of fingers changes according to the user and the angle calculation is needed. Therefore, the performing speed is slower than the other gesture. The accuracy is low. However, in the arbitrary environment, all gestures had more than 80% result. To overcome intersected fingers problem, our system calculates the cluster size of the intersected area and intersected size between convex hull and hand contour. These methods improved recognition rate.

2.5.4 User Questionaries

We did a questionnaire for evaluating each interface. The evaluation was progressed with 15 men and 5 women. We illustrated simply about the function of each interface. 20 users evaluated each interface on four items such as easy to use, the naturalness of the interaction with the gestures and intuitiveness of interfaces [8]. Five kinds of scores were made for each item; one to five scores (i.e., five is the highest score). The average score was obtained; the average scores of four interfaces (i.e., air-drawing, multi-touch, mapping and finger mouse).

Table 2.1: Average of each interface

Questions	Average
Was it easy to learn usages in each interface?	4.75
Was it natural to perform interactions and gestures using one hand?	4.65
Was it natural to perform interaction and gestures using both hands?	3.75
Was using the interface intuitive?	4.45

Table 2.1 shows that each interface obtained good result except in the case of using both hands. Because the system often occurs recognition errors in mapping interface and multi-touch interface, the users felt inconvenience relatively to use in real-time; particularly, pinch and spread gestures. However, our system was demonstrated effectiveness of the recognition accuracy even the pinch and spread gestures; we obtained over 90% in normal condition. 'Multi-touch interface' obtained a higher score in each content than the other interface ,because the multi-touch interaction is widely used and familiar to the users.
In the result, all interfaces obtained good scores, the average results can demonstrate intuitiveness, naturalness and easiness of the system; of course, partially demonstrated. Therefore, we conclude that the proposed interfaces are natural interaction with our proposed method and each experiment.

2.6 Summary and discussion

2.6.1 Discussion

In this chapter, we conducted three type experiments; interaction accuracy, gesture classification accuracy using three classifiers, and user questionaries.
In the interaction accuracy experiment, we obtained good recognition results (i.e., over 89%

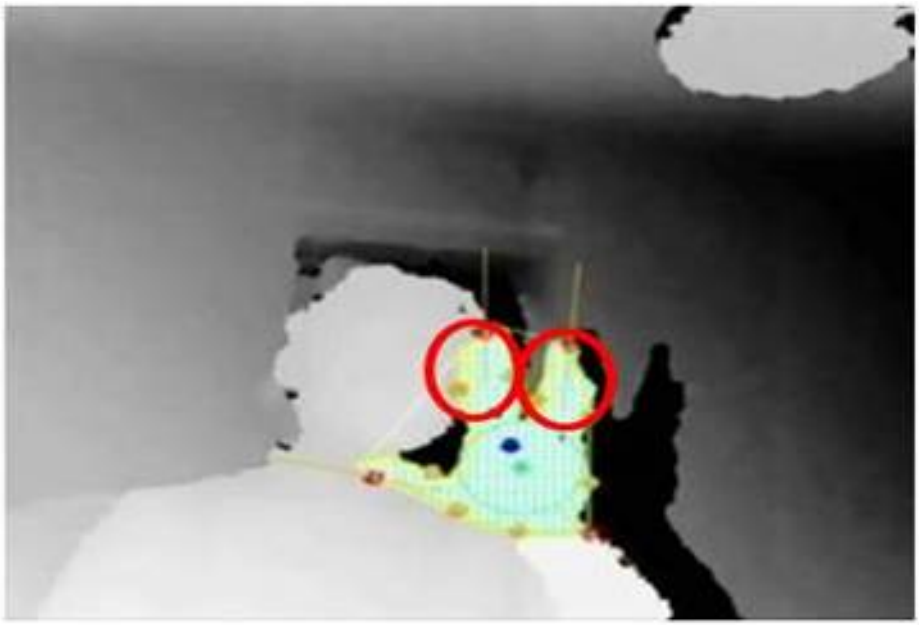

Figure 2.9: Special case of a hand gesture(the red circles are adjacent fingers)

for all gestures). Also, we found out that the light does not have an effect to the results, while the rough condition has much influence to the recognition accuracy; because of noise in clusters. We obtained over 96.7% in our main technology (i.e., finger identification). It demonstrated the method can provide many possibilities to natural user interface.

In gesture classification accuracy experiments, we demonstrated the effectiveness of our propose classifiers. However, we found out that there is the worst case of recognition accuracy; a star trek gesture, because two fingers are adjacent; the index and middle finger, the ring and little finger shown as Figure 2.9. We can solve this problem by calculating the cluster size of adjacent fingers using the hand's palm and the fingertips of adjacent fingers; the cluster size of adjacent fingers is bigger than a finger.

In user questionaries, we evaluated our proposed system with the components of NUI [8]; the contents of questions. Almost volunteers felt that propose interfaces are natural and intuitive to use. However, they felt inconvenience using both hands; the volunteers were familiar with multi-touch gestures with a hand, because almost volunteers used smart phone. We received comments from the users about the needs for applying these interactions in practical situations, such as games, Google Street View and Google Earth. The users also suggested the use of our proposed interactions on larger displays.

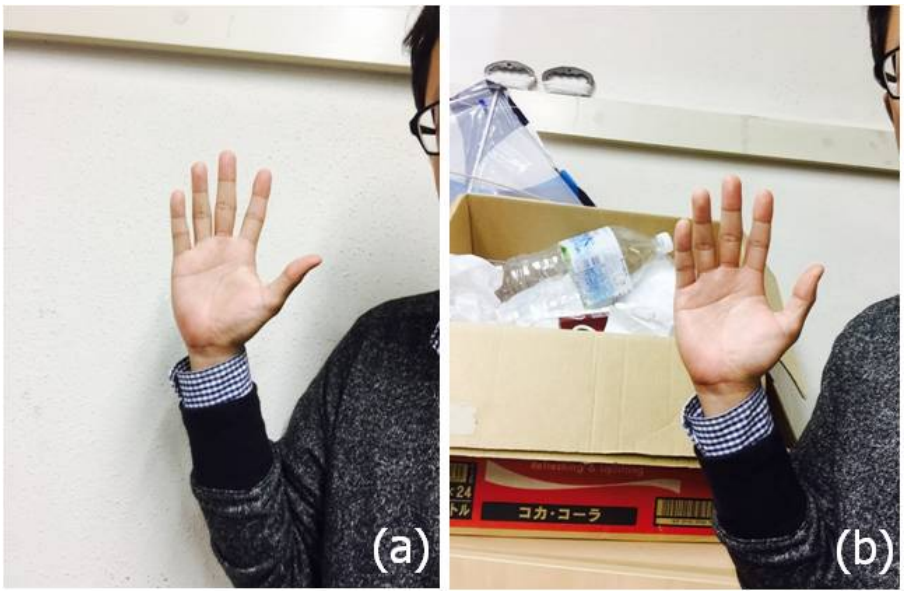

Figure 2.10: Evaluation condition

(a) Normal condition: White background without noise or obstacles
(b) Rough condition: Background with noise or obstacles between back of a hand

2.6.2 Summary

In this chapter, we presented fingertip tracking and finger identification based hand gesture recognition techniques using a depth camera. We used the clustering algorithm for hand detection and Graham Scan/Tracing algorithm for fingertip tracking. Also, we proposed new finger identification for improving gesture recognition accuracy and increasing gesture input vocabularies. We designed three gesture classifiers; finger counting, finger identification with hand pose and vector matching. The classifiers are designed for robust hand gesture recognition. We overcame previous problems; weakness of light and rough condition, limited gesture input vocabularies, unnatural gestures. We demonstrated our system's robustness and naturalness by evaluating three experiments; interaction and gesture experiments, user questionaries.

Our system provided natural finger interaction and gestures in depth-based gesture version such as drawing, pinch, spread, rotating and mapping gestures. We presented the interfaces using our proposed gestures with three classifiers. These interfaces are air-drawing, multi-touch, mapping, finger mouse. The interfaces demonstrated the usability factors, nat-

uralness and robustness by the evaluations with good recognition results. Of course, our approaches have limitations; the system provided pinch and spread gesture, it is still the 2D hand gestures even the system used three dimensional depth value. This is the motivations for upgrading our research 2D to 3D. It will describe in details in Chapter 3.

In this research, we can extend to support more practical functionalities of interactions using the proposed method in this chapter. Our system can be applied an augmented reality interface using HMD with the hand gestures. It will need to apply object tracking for marker-less interface. We believe that this interface can include practical functions and will be more natural to use.

Chapter 3

Point cloud-based 3D motion gesture interface

In this chapter, we describe the design of a point cloud-based 3D motion gesture interface (i.e., MOI toolkit), and prototype system using the toolkit (i.e., Touch-Pair [44]) for natural interactions. The design and architecture of the interfaces are presented, including detailed descriptions of the system algorithms. We described the reason that uses 3D motion gestures in the first steps. We conducted an experiment to evaluate the usability factors of tangible interaction accuracy and motion gesture classification accuracy. Finally, we described discussion about the successes and limitations of our approach, and conclude with the results.

Our themes are: designing a 3D motion gesture interface with a toolkit; hand motion, hand-object motion and tangible interaction are proposed in 3D motion gesture. Also, we design a natural and intuitive interface for connecting the cyber world and the real world using the toolkit.

3.1 Importance of 3D motion gesture

We implemented finger identification based hand gesture recognition system in Chapter 2. It provided robust hand gesture recognition, and we designed NUI using the proposed method. However, we found out that it still has limitations (i.e., 2D hand gestures). The biggest limitation of the method is unnatural. It means the users felt gesture as the input, because all gestures are three dimensional in the real world. In our previous method, we

should consider hand pose and hand degree of freedom, because the propose method cannot recognize when the hand is turning to hand's side visible. Thus, it has limitation to subtle gestures and various directions. Therefore, we aim at the interface comes to users naturally (i.e., the interface recognizes user's intentions). The 3D motion gesture can provide this naturalness and intuitive feeling with robust 3D gesture recognition [45], [46].
Many technologies demonstrated the effectiveness of 3D gesture to natural interaction designing, and overcame we mentioned limitations [6] [47]. Many 3D gestures based interfaces are proposed in many areas such as games [48] [49], medical applications [50] [51], consumer electronics; specifically for control of large screen smart TVs [52] [53], these interfaces have been popular [54]. The users felt that the interface is natural and intuitive to use [47]. In addition, the 3D motion recorder is released recently in Kinect SDK [5], and many samples are developed and applied to many areas (e.g., sports, movies and games). It demonstrated the 3D motion is useful and robust to use. This is the first reason for deciding 3D motion gestures in our research. It is not only improving 2D gesture limitations, but also very promising gestures for designing NUI.

Recently, Point Cloud Library(PCL) 1.8.0 version was released [12], the new possibilities for designing NUI using 3D processing are opened. The library includes 3D processing modules such as filters, features, segmentation, registration, so on. The modules provide 3D processing technique, and we can apply these modules to real-time 3D motion gesture recognition and interactions. This is the second reason for deciding 3D motion gestures in our research. The hand motion gesture using the modules provides three dimensional gesture (i.e., various directions, subtle gesture) and hand-object interaction with natural. 3D hand tracking and 3D object tracking are different research area, it is difficult to implement each tracking algorithm, however we can handle all of these easily using the PCL. We can design 3D hand motion gestures and 3D hand-object gestures with the technique.
We believe that our proposed 3D method can provide more possibilities for designing natural interfaces. We focused on two challenges to design natural interfaces with 3D motion gesture:

Challenge 1: Design 3D motion interfaces with a toolkit: We design an interface of 3D gesture library; with 3D hand motion gestures and 3D hand-object modules.

Challenge 2: Design natural interfaces and 3D gestures: Our system supports easy way to design 3D gestures what users' want to use; user-defined gestures, and designs Touch-Pair interface; natural tangible interaction.

In result, we can overcome the mentioned limitations by achieving successful implementation of the challenges, and the PCL libraries made changes to handle 3D tracking easily. Thus, we can design more natural and intuitive interface with 3D motion gestures.

3.2 Design goals

In this chapter, we focus on implementing 3D hand gestures and 3D motion gestures for designing NUI using the 3D depth-sensing input modalities. We design a gesture toolkit and the NUI examples using the toolkit. The designed interfaces and gestures cannot satisfy all users, because there are differences of preferring interfaces by the users. That is why the function of user-defined gesture input has to be provided with pre-defined gestures in NUI. In this thesis, we aim at designing interfaces with 3D gestures and tools that supports users or application developers to designing gesture and interface easily. Our system has three main principles of design:

1. Design tools of gestures and interfaces that supports the users and developers for designing their own interface easily: We design a 3D gesture library as an interface. The interface is provided as a library file, can be extended to users' applications. Also, tools are provided as executable applications. The tools are third party applications, it can be used controller in any applications on the users' side. In short, the developers can use these tools in their own applications, the users can use tools as a third party application to control their existing applications (e.g., internet browser, games).

2. Support easy method to design 3D gestures, whatever users want to use: We design our system to support user-defined gestures using the provided library (i.e., a toolkit). And our system can be trained by the gestures. We can provide natural gestures whoever use our system, because the user can design their own gestures by supporting our user-defined gesture tools.

We design the system with the interfaces, gestures, tools. We can expect to design more natural, intuitive and immersive natural interfaces in 3D.

3. Design 3D gestures with the example interfaces: The 3D gestures are implemented using 3D point cloud data. We design the 3D gestures ensuring the each interaction 'feels' natural and familiar such as 3D hand gestures, Touch-Pair, Hand-Object interface. The provided example interfaces provide naturalness and intuitiveness of 3D interactions.

3.3 Related work

3.3.1 3D hand gesture

A 3D gesture is a specific pattern that can be extracted from a continuous data stream that contains 3D position, 3D orientation, and/or 3D motion information. In other words, a 3D gesture is a pattern that can be identified in space, whether it is a device moving in the air such as a mobile phone or a game controller, or a user's hand or whole body [54]. Many 3D hand motion and gestures are proposed [55], we focus on the dynamic three dimensional hand movements; 3D hand motion.

Sanchez-Riera et al. [56] proposed 3D hand gesture algorithm. It tried to overcome two main problems of hand gesture tracking. That is the great number of degrees of freedom of the hand, and rapid movements for natural gestures. It uses 81 hand gesture database set, and use deep learning to train all gestures. Because of using a learning gesture, it obtained good tracking result with pre-defined gestures. However, it does not implement tangible interactions and motion gesture interactions. It still has limitations, that is static and limited gesture only. Our system attempts to overcome these problems.

Kristensson et al. [57] proposed one-handed and two-handed gesture with pre-defined gestures. They used the Microsoft Kinect 3D full-body motion tracking sensor to design a bimanual continuous gesture interface that recognizes one-handed and two-handed gestures while they are being articulated by the user. Their system continuously tracks both hands of the user via the skeleton data from the Kinect, and the system has an input zone to determine whether the user's hands are inside or outside the zone. Finally, it provides continuous bimanual hand gestures, and evaluates effectiveness of the gesture recognition accuracy. It obtained pretty good results of the recognition. However, because of using hand joint data, there are simple gestures only, it means that it is static hand gestures. Also, hand-object interactions and tangible interactions do not implement, so that it still has limitations with the simple gestures.

Hackenberg et al. [58] presented a technique implementing barehanded interaction with virtual 3D content using a Kinect. They implemented 3D multi-touch systems. They developed algorithms of finger and palm extraction and gesture recognition for the systems. It provided pose estimation for hand tracking and used machine learning for 3D gesture recognition. In multi-touch system, they implemented basic 3D interaction such as selection, translation,rotation and scaling, in a three-dimensional context. It implemented 6 degrees

of freedom of hands, and mapped well 2D multi-touch to 3D. However, the proposed method still has limitations. This research does not provide hand-object interactions, and it cannot support over 7 degrees of freedom of hands. It did not demonstrate its robustness and usefulness enough. In short, the proposed research is not enough to design natural use for interactions.

Colaco et al. [59] presented a 3D gesture sensing prototype and interaction with Head-Mounted Displays(HMD). Also, it implemented a single hand tracking technology using a 2D RGB camera and a Time-Of-Flight(TOF) sensor. They have contributions in designing a low-power prototype, so that it is suitable for mobile devices. They designed 3D motion gesture tracking with their prototype. The designed gestures are circle, point-and-click, swipe and zoom in-out. The gestures are performed with interaction on the HMD. The menu activates menu and controlling picture interface are designed using the gestures. It was well designed interfaces for Augmented Reality(AR) or Virtual Reality(VR) and obtained good recognition results. However, because it used a 2D camera and a TOF sensor the gestures are static and simple, also the hand-object interactions did not implemented. It has multiple hands overlapping problem as well.

Our system overcomes these mentioned problems. Our proposed methods do not have the limitation of degree of freedom of hands because our system uses 3D point cloud data, and are robust to recognize 3D hand gestures. Also, it provides hand-object interactions. Finally, we have implemented a toolkit and Touch-Pair interface (i.e., the example using our proposed method). The toolkit aims at users or developers who do not know 3D gestures can use the gestures easily.

3.3.2 Depth-based touch sensing technologies

In recent years, depth-based cameras and related technology have developed rapidly. Research on obtaining 3D data on objects using depth information has also made progress. The framework for 3D sensing using depth cameras has been improved remarkably. Klompmaker et al. [60] implemented tangible interactions using a depth camera and a 3D sensing framework. They have implemented touch detection and object interaction, supporting multi-touch and tangible interactions with arbitrary objects. They used images from a depth camera to determine whether a user's finger touched the object. However, they were unable to support 3D touching and dynamic pairing between objects for tangible interac-

tions. Wilson et al. [61] presented depth-sensing cameras to detect touch on a tabletop, using the camera to compare the current input depth image against a model of the touch surface. The interactive surface does not need to be instrumented in advance for the interaction, and this approach allows touch sensing on non-flat surfaces. The system provided a simple example of tangible interaction with a book. It recognized touched points of a book. However, they only supported simple touch recognition, the recognition was applied the book's surface (i.e, almost 2D touch), and could not address touch in any direction with 3D objects.

Our proposed technologies overcame all problems by providing hand-object tangible interactions. It provides three dimensional touch recognition, also 3D motion gestures with the objects.

3.3.3 Tangible interactions with objects

The interactions with physical objects are important areas in Tangible User Interface(TUI). Many researches are proposed with several kinds of sensors [62], [63]. They mainly presented hand-object, tangible interactions.

"Digital Desktop" by Wellner [64] was used in an early attempt to merge the physical and digital worlds. They implemented a digital working space on a physical desktop where physical paper served as an electronic document. The interaction with papers was by means of bare fingers. This research provided tangible interaction with a desk, and useful examples are presented. Nishi et al. [65] presented augmented desk interface with registering real objects on a user's desktop based on a user indicating a region on the desk by making a snapshot gesture with four fingers. A color histogram was used to model the object and a pointing gesture was used to trigger the recognition. They proposed natural and intuitive interaction with a desk, and it is useful. However, these researches are tangible interaction on the desk surface (i.e., 2D touch), thus it has many limitations for 3D interactions.

"Icon Sticker" [66], based on this idea, is similar. Icon Sticker is a paper representation of digital content. It consists of transferring icons from the computer screen to paper, so they can be handled in the real world and used to access digital content directly. An icon is first converted into a corresponding barcode, which is printed on a sticker. Then the sticker can be attached to a physical object. To access the icon, the user scans the barcode on the sticker with a barcode scanner. "Web Sticker" [67] uses barcodes to represent online

information. It is similar to Icon Sticker, but instead of icons, it manages Web bookmarks. They use a handheld device with a barcode-reading function to capture the input and display related information. Here were also attempts to improve tagging of physical objects for a more natural tangible interaction. However, the research did not provide 3D gestures, and unnatural because it attached sticker and used barcode.

Ono et al. [68] presented tangible interaction using their proposed method. They used acoustic sensing to have touch input capability of objects. They proposed prototype sensor, and provide tangible interactions by attaching a sensor to the objects. Also, they designed interesting interactions such as much player, multi-functional input device using everyday objects. The system demonstrated robustness of touch or grasp recognition, and useful applications. However, in gestures, they used pre-defined gestures and static gestures. Also, the system has to attach a sensor to the object for the interactions with wires. Additional sensors are attached when users want to use multiple objects. It is the unnatural way in NUI. Our research overcame these problems, also we can design natural tangible interactions with 3D motion gestures.

Although many previous tangible interaction researches have used physical objects for interactions [69], most of them are additional sensor based approaches, and they do not support 3D hand-object motion gestures or need additional sensors for increasing interactive objects. Thus, to overcome this, we proposed a robust 3D hand-object motion gesture recognition method that detects touch in three dimensions. The system supports dynamic pairing (i.e., increasing or changing the interactive objects) between physical and digital objects without additional sensors, and makes physical objects accessible to touch anywhere.

3.4 Our proposed techniques and interface

In this section, we describe our 3D motion gesture recognition technologies in detail, and provide a toolkit with usages, and we implemented the examples using our proposed method.

3.4.1 3D motion gesture interface

We designed 3D motion gesture interface base on Point Cloud Library(PCL)[12] that provided 3D data processing modules. The pipeline of the interface is shown as Figure 3.2. Our interface provided three different modes; one hand, two hands, hand-object. The

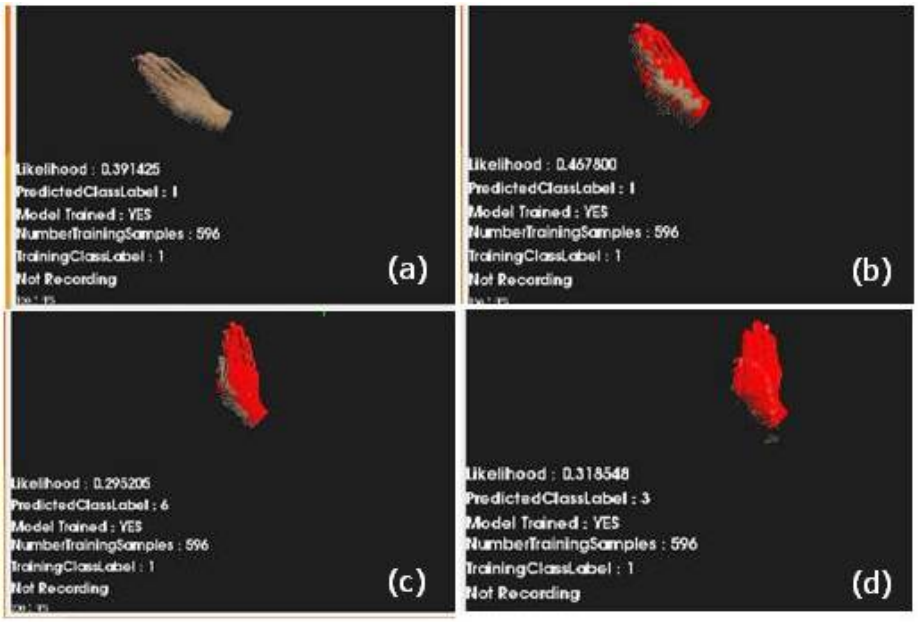

Figure 3.1: 3D hand motion gestures in our interface(swipe a hand rightward) (a) Initial hand poses and (b), (c), (d) Detect differences of motion over time sequence(red color of the points)

interface was designed for 3D NUI with 3D motion gestures in real-time. Our interface performs pre-processing (i.e., filter, downsampling of point cloud), because the 3D data processing has high time complexity in real-time.

The techniques of our interface composed of three main parts. That is 3D recognition and tracking, machine learning and gesture prediction modules. The 3D recognition and tracking modules provided hand tacking, object tracking and hand-object collision detection in 3D. The hand segmented depth and color information, and the object segmented by Plane model segmentation and Euclidean cluster extraction [70]. The segmented 3D point cloud data (i.e., hands and objects) are shown on the 'Scan objects' part of Figure 3.3.

The machine learning modules provided Dynamic Time Warping(DTW) classifier [71] for dynamic motion gesture, Adaptive Naive Bayes Classifier(ANBC) [72] for static gestures. The gesture prediction modules perform prediction with the classifiers (i.e., DTW or ANBC) using pre-defined gesture databases or user-defined gesture databases. The pre-defined gestures are designed based on [8]; swipe, multi-touch. Users can design their own gesture, then record the gesture for training the system. The natural and interesting 3D gesture can be designed using our proposed interface.

3.4.1.1 Interface pipeline

We describe our interface pipeline in sequence; our system provided six processes for interactions. First, the users or developers decide a gesture type in select mode (i.e. one hand, two hands, hand-object). Then, the hands segmentation and scanning objects are performed for 3D tracking with the filtering. The objects are tracked by using a scanned object database. The collision detection is performed when the users selected hand-object mode. The 3D motion gestures can be recorded, and our system is trained by recorded 3D point cloud data. Also, we can use pre-defined motion gesture database because our interface provided record gesture database (e.g., swipe, multi-touch, ...). In this case, we do not need record and train of gestures. The 3D motion gestures can be predicted by recorded data or pre-defined data. Our system calculates the change of points amount when the system performed gesture prediction. The predicted gesture's result is returned to the users' side applications, and interactions are performed by the result.

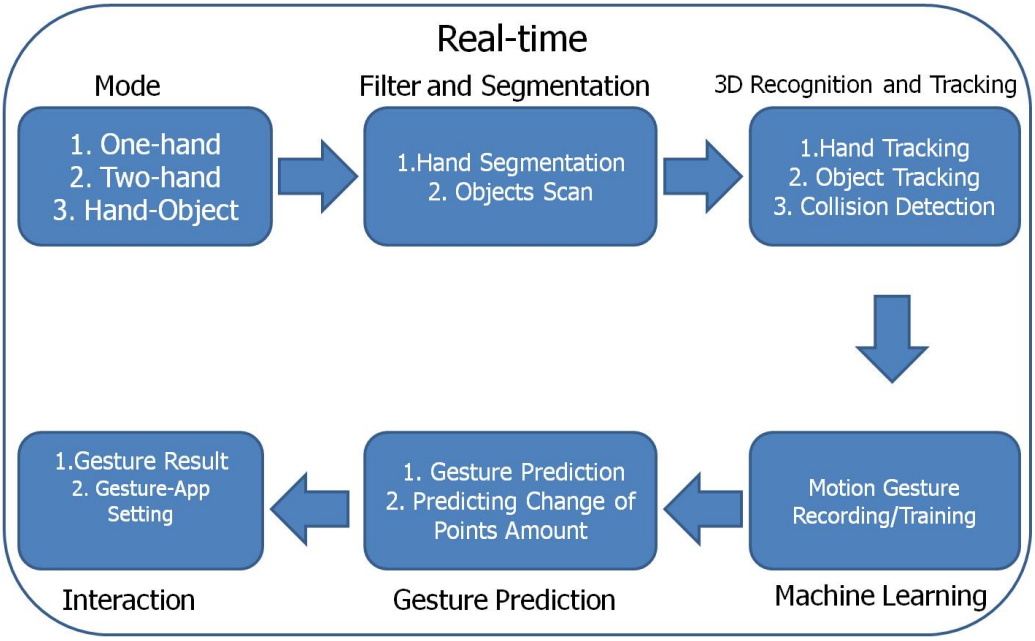

Figure 3.2: 3D motion interface pipeline

3.4.1.2 3D recognition and tracking

We describe the 3D recognition and tracking modules in details (see the right diagram of Figure 3.7). Our system used color and depth information from a Kinect for 3D hand tracking. We segment 3D point cloud of the hands by depth information first (i.e., depth threshold). The segmented areas are defined as 'interactive area', and then the hands are segmented by the skin color; erode first then dilate to eliminate the noises. The center points of the hands in 3D are calculated by a 3D centroid function in PCL. Our system can record 3D motion gestures based on the center points of the hands. Our system attempts to track the fingertips of hands in 3D. Our system creates a distance map to produce skeletons of hands; it makes the tracking easier. However, the tracking error occurs in real-time; because inaccuracy points are obtained from the Kinect sensors. We should do post-processing for improving recognition accuracy; 3D registration and reconstruction [73].

There are two steps for 3D object tracking; we provided 6 Degrees Of Freedom(DOF) of

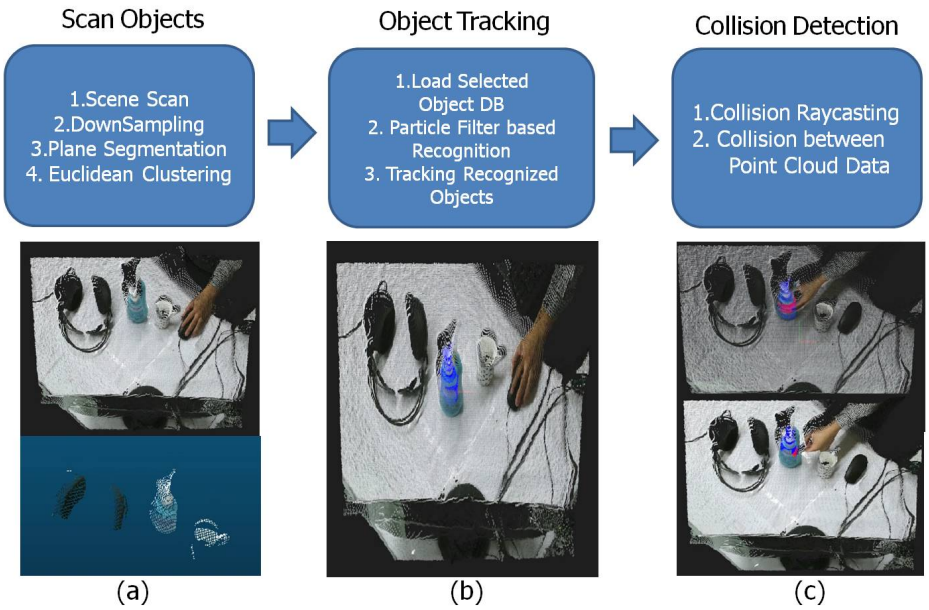

Figure 3.3: (a) Point cloud based scene(top), Scan result of objects(bottom) and (b) Object tracking(blue color) (c) Collision detection(red color)

objects. The first step scans objects, second is 3D object tracking with particle filters [74], [75]. In scan objects, our system captures 3D point cloud data of all scenes after applying distance filter and downsampling; the downsampling is important to improving time com-

plexity, however, GPU processing is better than downsampling with remaining current point cloud amounts (i.e., remaining robustness of recognition). Plane segmentation is performed after pre-processing. Only objects remain on a desk after doing the plane segmentation. Euclidean clustering performs for storing each remained object to pcd files; a file used in PCL. A user or an application selects an object what they want to track. Our system tracks the object using a pcd file based with the particle filter. The particle filter is performed the processes of prediction, weighting and resampling by every time. Our system predicts the position and rotation with normals, so that we can predict 6-D pose of objects. The tracking is kept even the object be made movements rapidly.
Our basic 3D object tracking is performed as below [76]

- At first, using previous Pariticles information about position and rotation, it will predict each position and rotation of them at the next frame.
- Next, we calculate weights of those particles with the likelihood formula 3.1; we can select which likelihood function you use.
- Finally, we use the evaluate function which compares real point cloud data from a depth sensor with the predicted particles, and resample particles.

The particle filter based 3D object tracking performed the weighting using the equation 3.1.

$$\begin{aligned} weight^{(i)} &= \sum_{j} L_{distance}(p_j, q_j) L_{color}(p_j, q_j) \\ L_{distance}(p_j, q_j) &= \frac{1}{1 + \alpha |p_j - q_j|^2} \\ L_{color}(p_j, q_j) &= \frac{1}{1 + \beta |p_{jcolor} - q_{jcolor}|^2} \end{aligned} \tag{3.1}$$

p_j: A point of the hypothesis point cloud
q_j: The nearest point of the input point cloud to p_j

3.4.1.3 Collision detection

The collision detection is an important area in tangible interaction [77], [78]. The collision performance decides the naturalness and intuitiveness of tangible interaction. We should consider the reason that we used octree data structure; because it is suitable for

Algorithm 2 Collision Raycasting Algorithm

Input: Origin Vector, Direction Vector *points* ∈ Segmented Hand Points
Output: Intersected Octree Node, after k iterations

Initialization $Origin = (0, 0, 0)$, $Direction$ = (hand[i].x,hand[i].y,hand[i].z)
CurrentNode = FirstNode
while $i \leq k$ **do**
  if hand[i].points != OctreeNode.iterator **then**
    go to GetSubNode
    while CurrentNode has intersected points **do**
      Save the Points
    end while
    GetNextNode
  else
    Save the Points
  end if
end while

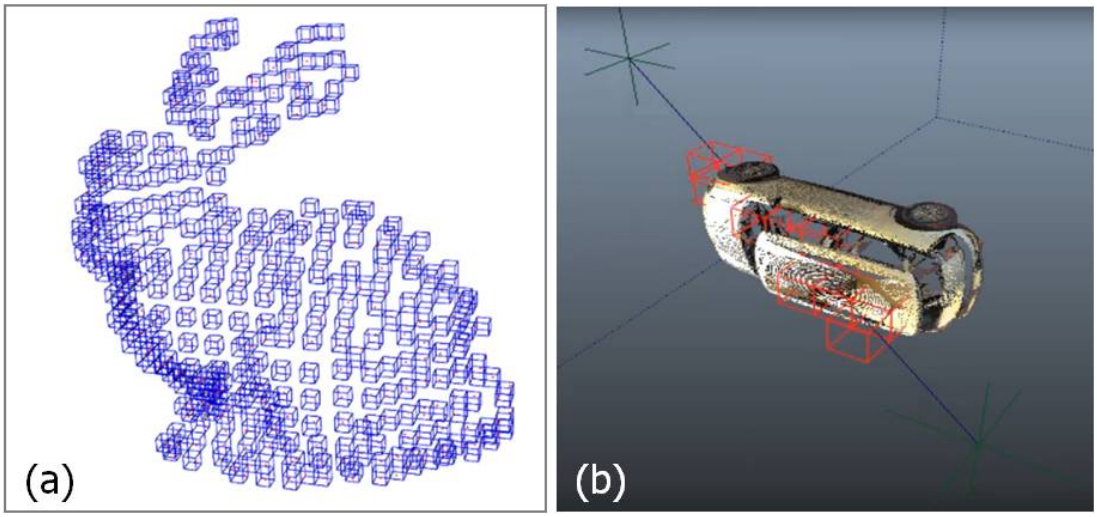

Figure 3.4: (a) Octree in PCL(creating a hierarchical tree data structure from point cloud data) [2] and (b) Collision detection(red hexahedra) between a line and a car of point cloud using Octree-Raycasting [3]

performing voxel (i.e., 3D pixels) search, radius search and neighbor search. Basically, the collision detection is based on searching intersected points of objects. The octree is appropriate for searching intersected points in real-time.

At this point, our hand-object interaction uses collision points between a hand and an object. The collision detection is performed based on the octree data structure. We used the octree based approach because the octree data structure is fast enough to recognize collision in

real-time as we mentioned. Especially, we used the method of octree ray-casting (see Figure 3.4), it is suitable for 3D collision detection using the difference of points distance. The ray-casting method needs an origin point and direction for detecting intersected points (see the (b) of the Figure 3.4). In our system, we set the origin point as zero points, and direction as hand points. It detects the collision points between a hand and an object (see the red points in Figure 3.9). The detected collision points will be used for gesture recording, and training the gestures to our system for 3D interactions.

3.4.1.4 Motion gesture training and prediction

The motion gesture training and prediction is performed when users or applications want to design their own 3D gestures. In basically, our system can record their 3D motion gestures (i.e., movements of the 3D point cloud data in time sequence). In a hand or two hand mode, our system records 3D point cloud data of hands' motion, And in hand-object mode, the system records the collide 3D point cloud data of hands and an object. In other words, our system recognizes the collision points of hands and an object, then records the points in time sequence.

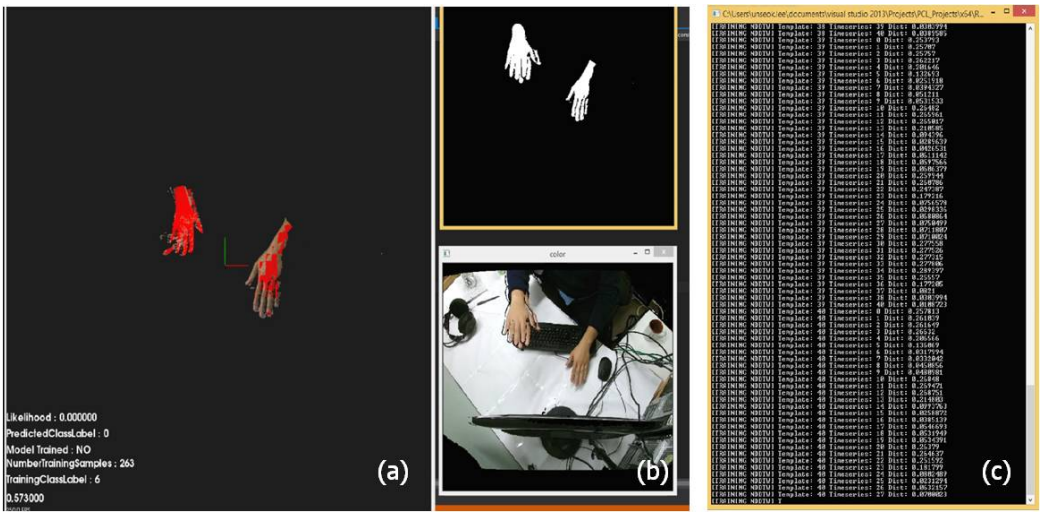

Figure 3.5: (a) Motion gesture recording and (b) Binary image(top), RGB image(bottom) and (c) Creating motion gesture templates using DTW

(a) The red point means movement of the hand; moving point

(b) The binary image obtained from the RGB image by skin segmentation

(c) The template data are created by recorded data, then our system is trained by the

template automatically

The users can design their own 3D motion gestures by recording (see (a) of Figure 3.5). The system is trained by using the recorded gestures, and the gesture prediction is ready after the training. Gesture templates are made with the system is trained. The gesture template's number is related with recording time and a number of iterations, also it is related to training time (i.e., much training time, many recorded data). For the system, we used DTW and ANBC machine training algorithm, because the algorithm is designed for static and dynamic gesture training and prediction. These are implemented in Gesture Recognition Toolkit [4].

Predictions of gestures are performed in an almost similar way with gesture training. The predictions are performed using gesture templates, and also our system uses detection of changes of 3D hand point cloud amount as well (see Figure 3.5); the red points in the hands means the results of change detection based on octree data structure. The system can detect the sensitive of gestures using detecting the changes, it means that the system discriminates the gestures by detecting how the hand moved in a second.

For example, right-swipe gesture, (a) moved a hand 20 centimeters in one second (b) moved 20 centimeters in two seconds (c) moved 19 centimeters in a second, the (a), (b) and (c) are discriminated different motion gestures.

In result, the users can design various motion gestures(i.e., various input vocabularies) using the machine learning, also we improved accuracy of gesture recognition and classification using the method.

3.4.1.5 Automatic gesture spotting for motion prediction

One of the key advantages of Gesture Recognition Toolkit(GRT) [4] is that the GRT can automatically reject movements that are not any of the gestures (e.g., beginning the motion for prediction) that we have trained the system to recognize. In the GRT, this is called null rejection. By null rejection, our system estimates a threshold value that rejects samples which are unlikely to belong to any of the gestures (see white color in the bottom of (b), (c) on Figure 3.6). Thus, the system can recognize meaningful gestures from continuous hand motion or motion of collision points using the GRT functions.

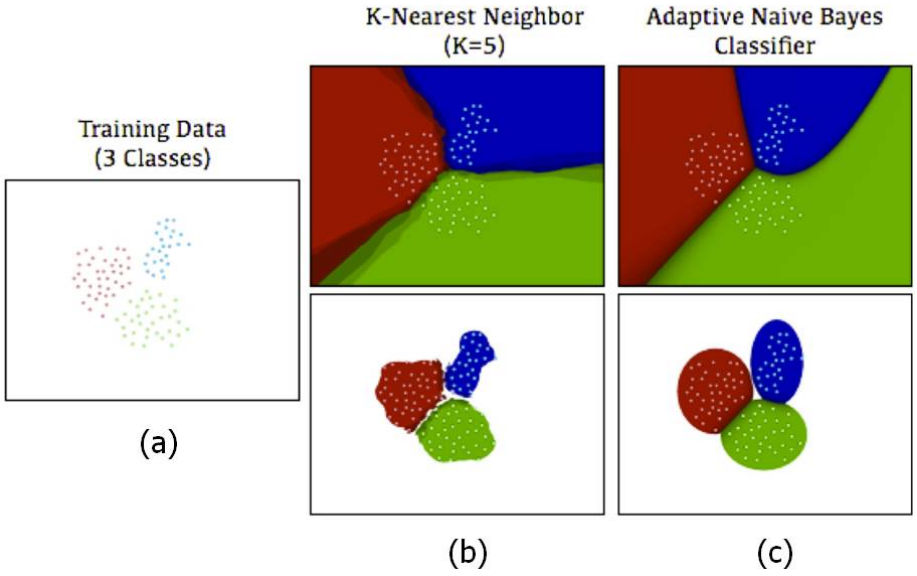

Figure 3.6: Gesture spotting example [4]

(a) Training data (three different gestures)
(b), (c) Predicted gesture label without null rejection (top), with null rejection (bottom)
Red for gesture 1, green for gesture 2, blue for gesture 3, white for null gesture

3.4.2 MOI toolkit

A MOI toolkit (Motion interface of IPLAB) is fusion tools for designing 3D gesture easily. We implemented this toolkit, because of the difficulties of 3D gesture implementation even 3D motion gesture has usefulness in designing NUI. Our toolkit aims at easy portability, extension, convenience and intuitiveness of the use on users' side. Our toolkit includes many technologies such as parts of 3D process modules; 3D hand tracking, object tracking, collision detection and machine learning. The users can use all of the technologies just by input our toolkit.

The toolkit provided as a library file. The developer who wants to input gestures to their own applications can designed by input the file to their applications. The modules are provided as we mentioned such as 3D recognition and tracking, gesture training and prediction (see Figure 3.7); the function and classes. The toolkit returns the results of predicted gestures by the modules. We implemented simple examples for showing how to use our toolkit; one or two hands motion interface, hand-object motion interface.

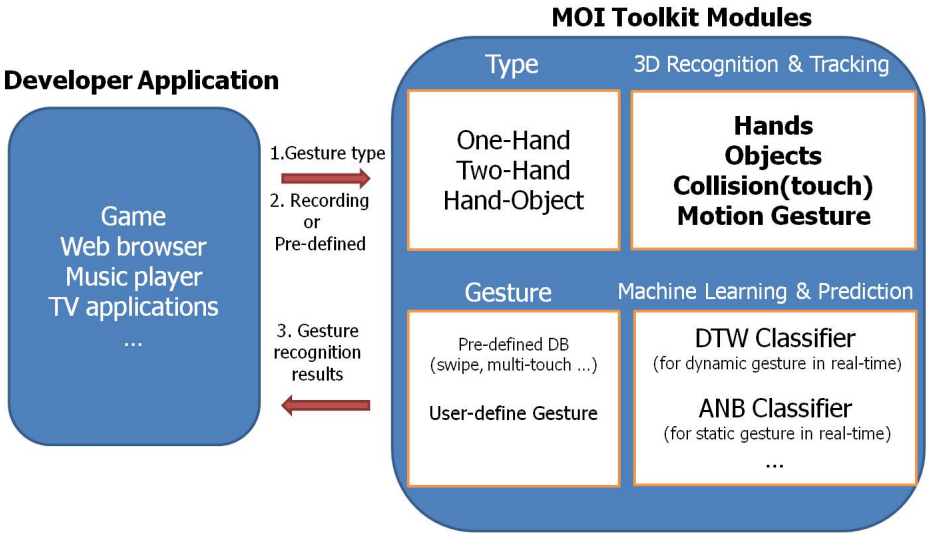

Figure 3.7: MOI toolkit interface

3.4.2.1 3D hand motion interface

We implemented simple hand motion gesture examples for showing usage of the toolkit. The implemented examples are "Google Streetview" controller using one hand, "Google Earth" controller using two hands motion gestures. We recorded six 3D motion gestures for one hand; swipe up, down, left, right, front and back. We can control the Google Street View; go or back, up or down ,turn left or right (see (a) of Figure 3.8). Also, we designed six 3D motion gestures for two hand; swipe left or right by each hand, zoom in or out, rotation of clockwise or counter-clockwise (see (b) of Figure 3.8). We can control the Google Earth; rotate left or right and zooming in or out, rotate the earth clockwise or opposite.

The examples used center point of each hand as basic input first, and then the difference of the change amount of hands is detected for classifying gestures (see Figure 3.1). The predicting gesture is performed using gesture templates. In the example, we used 596 samples for six gestures classification in one hand interaction, and 269 samples are used for six gestures classification in two hand interaction. We can expect to robust recognition accuracy by increasing the samples; the number of times of recording the gesture. Because it is based on the center points of hands and fingertips in 3D, the interface provides low time complexity and robust accuracy with recording just six times (i.e., about 100 samples per one time) for one hand motion gesture and ten times for two hand motion gestures.

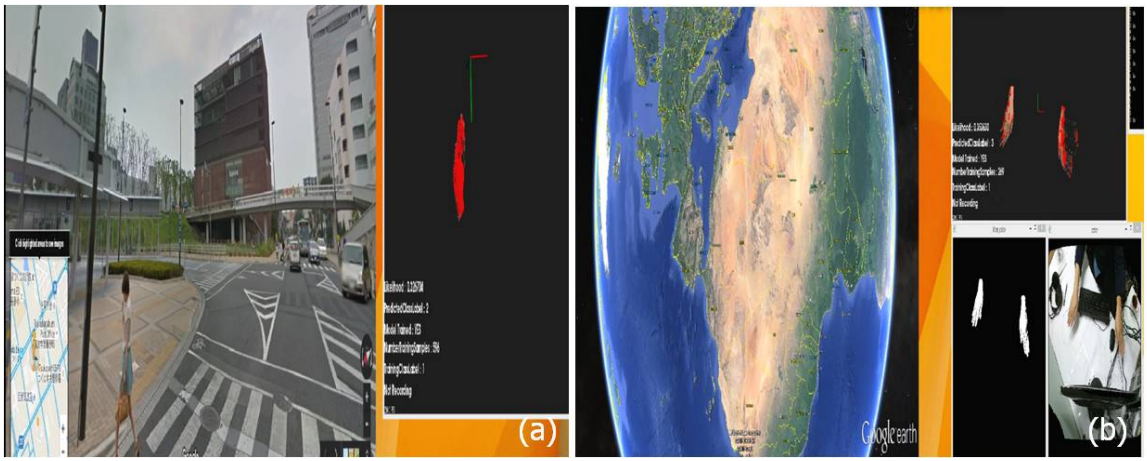

Figure 3.8: (a) Control Google Street View using pre-defined one hand motion gestures and (b) Control Google Earth using pre-defined two hand motion gestures

It is few iterations in cognitive machine learning areas [4], however, our interface provided fast, easy and robust machine learning of 3D hand gestures with DTW.

3.4.2.2 3D hand-object motion interface

The researches of a hand and physical object interactions have been conducted for many years. The interactions provide intuitiveness of feeling, because the users touch the real objects with some feedbacks [79], [80]. The proposed tangible interactions have to attach a sensor or multiple sensors to physical objects mainly (i.e., unnatural), even the researches have robustness of accuracy. This is the motivation for designing point cloud based hand-object interactions in 3D. The hand-object interface means the interactions between a hand and an object (e.g., a cup, a cube box, ...) using a depth camera. For instance, the interactions can be designed such as touch a cup, grasp a cup, the pouring action with a cup. It can be considered the tangible interactions using 3D point cloud data from a depth camera. The interface can recognize touched points; collision points of them. We implemented a hand-object motion gesture example using the proposed method. It is music volume controller by using physical objects; it is green cube box. The paired digital object is a computer. We trained clockwise and counter-clockwise motion gestures with the paired object. For these, our system calculates hand-object collision points, and detects differences of the change amount of collision data (see Figure 3.9). Finally, the system predicts gesture using trained collision gesture templates.

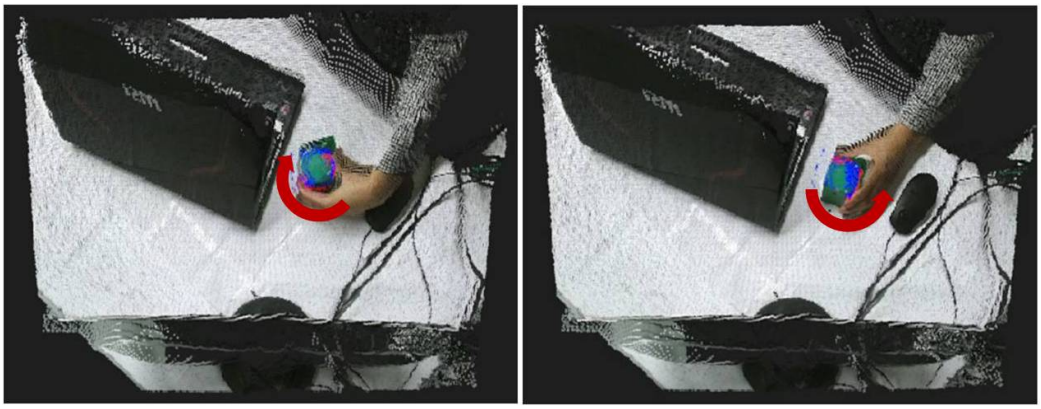

Figure 3.9: Computer music volume control using pre-defined clockwise(left) and counter-clockwise(right) motion gesture

3.5 Touch-Pair interface

In this section, we describe Touch-Pair interface using our MOI toolkit. The interface was designed for connecting cyber world and real world easily; the cyber world is digital object, the real world is physical objects. In previous tangible research, it is difficult to connect the cyber and real objects; it needs additional sensors or pre-setup devices. Also, we inspired from that the depth camera can be a touch sensor [61].

Our proposed interface provided a dynamic paring technique without additional sensors. The fusion interaction (i.e., tangible and 3D motion gestures) can be provided with our interface; it can be considered an example of hand-object motion interface. Because our interface used two Kinect sensors, the hardware setup was changed (i.e., the position and angles of the sensord). We mainly used the official kinect SDK for MOI toolkit, and we used OpenNI for Touch-Pair interface; in other word, the software setup had been changed a little, because the OpenNI is different with the official SDK for obtaining raw data from Kinect cameras. Also, official SDK does not support multiple Kinect sensors simultaneously. The hardware and software setup were changed, but the interface still provides powerful tracking technique and gesture recognition.

Moreover, it provides more robust 3D object recognition, because it used two cameras (i.e., few invisible areas of objects) and surface reconstruction method. We describe the interface and technologies in details.

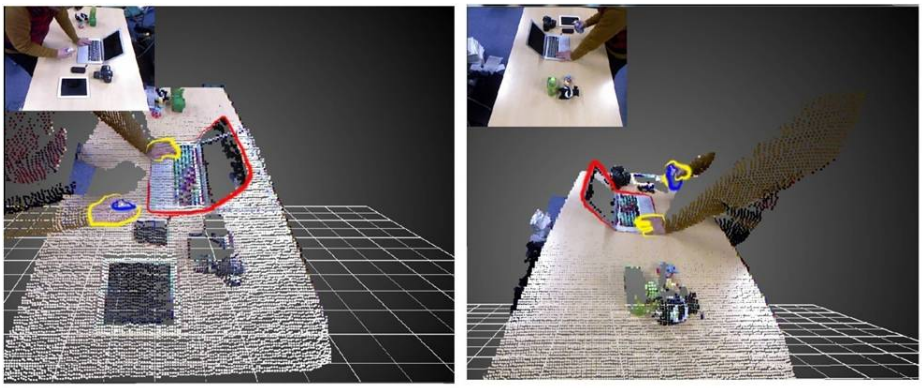

Figure 3.10: TouchPair interface from camera 1(left) and camera 2(right)

The red color: Touched digital object (i.e., a computer)
The blue color: Touched physical object (i.e., a plastic bottle)
The yellow color: Tracked hands

3.5.1 Motivation

We use touching actions in our daily life. The action is performed when we drink water, control remote controller for watching television. In other word, touching action is natural, user-friendly and intuitive. Recently, touch action is widely used with smart phone and smart pad. Also, researches for tangible interaction with the daily physical object are increased in HCI, because the touch gesture has been popular. Many researches are proposed about this area; adding tangibility to objects by attaching sensors [81], [82]. The objects are almost physical object, and the tangible interactions are designed to control digital objects (e.g., computers, smart phones,...) by the physical objects. However, those researches have limitations. They need to take efforts for attaching sensor with wire to the objects, and they need additional sensors to the object if they want to make interaction with other objects. In daily life, attaching/detaching sensors are inconvenienced and unnatural with many wires. Attaching sensors have good recognition result in tangible, but they cannot provide motion gestures as well. Therefore, we designed an interface to overcome mentioned problem.

Our interface aims at changing tangible objects dynamically without additional sensors; we can add tangibility to the objects or remove tangibility from the objects freely, and also adding motion gestures to the object as well. In short, our system can add tangible and 3D

motion gestures to the objects; fusion interactions using the pre-setup depth cameras. The key goals of our interface are two parts. First, *be fast and easy to set up.* Second, *be connected virtual world and real world very easy and natural way.* The interface can be an example of hand-object interaction in MOI toolkit. It can be expected many extended possibilities for natural user interface.

3.5.2 Touch-Pair system

3.5.2.1 Hardware and software

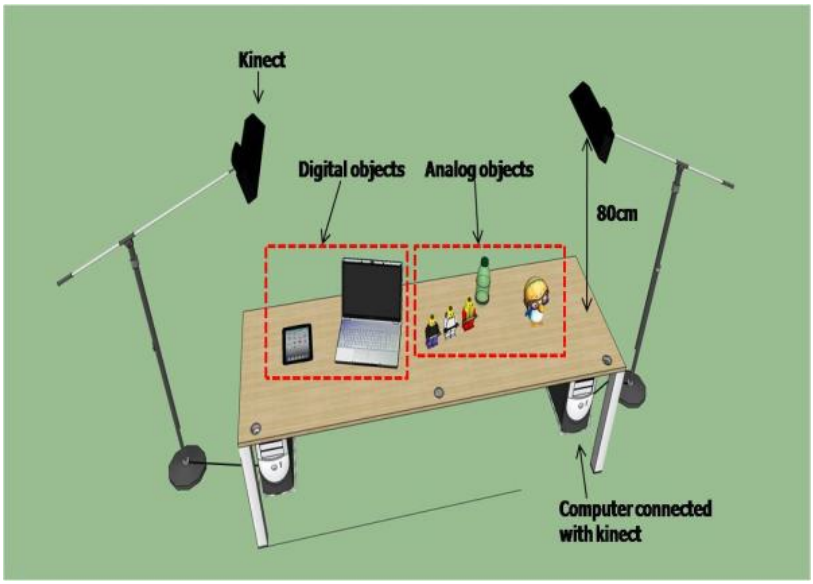

Figure 3.11: TouchPair System Configuration

Our system consists of two Microsoft Kinect sensors for Xbox 360 with stands. Two computers (Intel i7 2.4Hz, 8GB RAM, and GeForce GTX 660M graphics card) are used to handle the 3D point cloud data in real-time. A desk and the pairing object for interaction are installed. The physical and digital objects are randomly placed (see Figure 3.11). The Kinect sensors are set at 80 cm from the desk. The computers are connected to each Kinect sensor, and the digital objects have wireless internet or Bluetooth connections to the computer (i.e., installed in the bottom of the desk). Our system uses OpenNI in OpenFrameworks [83] for the Kinect sensors, and a point cloud method that provides an example of add-ons in the frameworks; the Point Cloud Library. The frameworks provide multiple Kinect sensors; not the offical Kinect SDK. The system obtains the raw data of

3D point cloud, and maps the RGB data to the 3D point cloud; the PointXYZRGB data in PCL.

The touched points and movement of the points are based on pairing and recognition of 3D motion gestures. The proposed system was implemented on a Microsoft Windows 7 platform. The pairing recognition module was implemented in Visual Studio 2010 and OpenNI 1.5.4 [84]. Our system used two computers and two Kinect sensors; two computers for real-time 3D processing (i.e., computation complexity), two sensors for robustness of 3D touch recognition.

3.5.2.2 Architecture

The entire system consists of three major modules (see Figure 3.12); 3D calibration and reconstruction, Touch recognition and learning gestures, and Prediction of trained gestures. The input data are obtained by the two Kinect sensors, on the left and right. It calibrates each 3D point cloud data from two cameras, and processes the data. In the case of using one camera, there are parts that cannot be captured the point cloud such as both sides of a cup. Because one camera cannot capture all areas of objects, even if camera sets top-down direction up; in 3D scan, it needs moving a camera or an object for capturing all areas of the object. By using only one camera, it is difficult to find out the touched location and touching gestures, because it cannot cover all sides of objects. Our interface sets up two cameras with 3D processing (i.e., reconstruction [85]); the cameras capture both sides of objects, 3D processing is performed for filling holes in surfaces.

Our system sets up two cameras in proper position that was found out empirically (see Figure 3.11); 80 centimeters from a desk. The system performed the 3D calibration first with 3D registration, and 3D reconstruction to obtain reconstructed data from calibrated data. We can recognize touching of almost all sides when using these methods. We can find the location of the touch as well. In addition, the difficult side of recognizing (e.g., bottom of the object) is estimated using the surface estimation (i.e., moving least squares smoothing [86]). The process is detailed below.

1) 3D calibration and reconstruction module: In this module, we calibrate 3D point cloud data of each object, obtained from the two Kinect sensors (see (a) of Figure 3.14); our system performs translation and rotation for 3D registration of each different point cloud data from the sensors. The system makes a 3D reconstruction model using with a calibrated

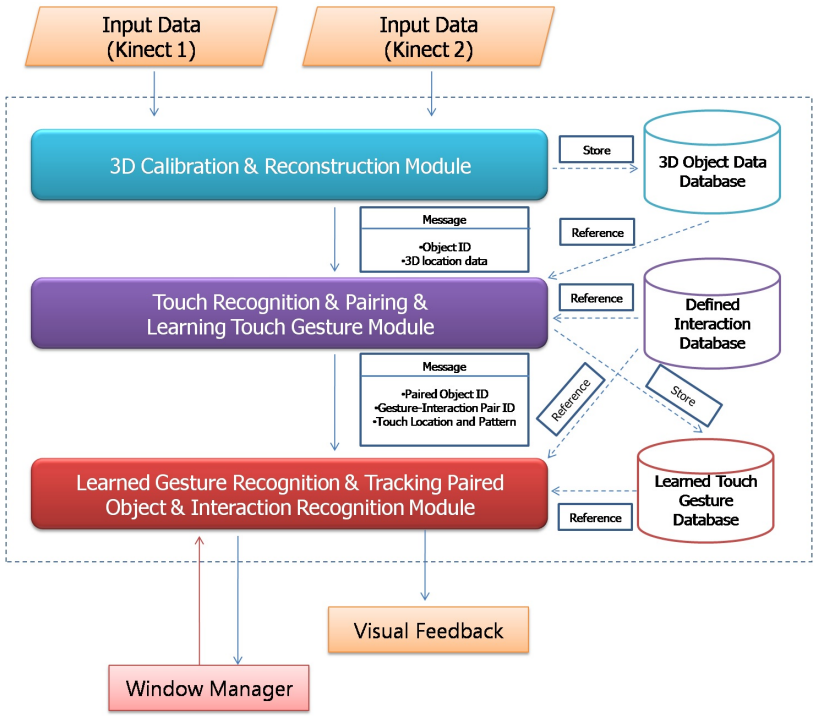

Figure 3.12: TouchPair System Architecture

3D point cloud data (see (b) of Figure 3.14); the calibration performed based on the object recognition. The surface reconstruction is performed with smoothing and resampling the surface using Moving Least Squares(MLS) method. This module stores calibrated and reconstructed data in a database as meshes, which is then used by the touch-recognition module. After storage, the module sends messages to the touch-recognition and pairing module about object ID and object location using the 3D point cloud data.

2) Touch recognition, pairing, and learning touch gesture module: In this module, we implemented pairing and learning touch gesture with touch recognition method. The process is detailed as follows.

- Touch recognition and pairing: Touch is recognized in terms of the depth and position of the object and hands using 3D tracking. Using the previous depth information from the 3D reconstruction based on the point cloud, the system determines whether the hand touched the object, and if so, the position of the object. The system recognizes the time of touching between the user's 3D hand point cloud data and the object 3D point cloud data, then determines whether

they are paired. 3D point cloud data of the paired physical object are stored and sent to the tracking module with information on the object type. We defined a limited object database.

- Learning touch gesture: In this method, the system identificates touched position of hands and fingers with objects. In addition, our system stores movement of the fingers and gestures to the database. A touched object is stored with its 3D vision image for tracking after paired. Gestures and interaction pair with touched objects can be stored in the database by the system as well. They can be used tangible interactions with learned touch gestures.

3) Learned gesture recognition, tracking paired object, and interaction recognition module: In this module, our proposed system recognizes learned touch gesture and tracks paired object. In addition, a tangible interaction is implemented with touched position and gestures using data from the database. The process is detailed as follows:

- Learned gesture recognition: The learned touch gesture recognition is implemented using learned touch gesture database that was stored by the learning touch gesture module. The system recognizes touched position and gestures applied to the objects. The paired object and interactions are prepared after recognizing the touching gesture and the touched objects.
- Tracking paired object and interaction recognition: After pairing, the system tracks the physical object based on stored 3D point cloud data. The paired digital object can be tracked; however, the paired objects do not commonly move. The paired physical object is tangible, based on 3D physical object data, from the 3D calibration and reconstruction module. We can make an interaction with the digital object in this module; the interaction is shown by visual feedback.

3.5.2.3 Touch paring method using 3D point cloud data

Our proposed method uses 3D point cloud data processing of Kinect depth data. A point cloud itself is a set of data points in a coordinate system. We measured a large number of points on the surface of an object using PCL [12]. The system obtains RGB data from two Kinect sensors (see Figure 3.10) and assigns them to the depth area. However,

not all directions of the object can be reconstructed. Thus, we find the most appropriate location for the Kinects and position them so that they cover most of the experimental space.

1) Touch gesture and recognition: Our touch pair system recognizes the touch actions of users' hands based on depth. The system calculates the depth of each point between a user's hand and the object by filtering closer data. The flow of recognition is as follows.

- Calculation of all depth points: Calculate all points of the physical and digital objects and the user's hand on the table.
- Determination of finger position: The system calculates the minimum and maximum depths of the finger by using defined thresholds because we hold our fingers in specific ways when we touch something.
- Determination of hand position: The system calculates the minimum and maximum depths of all fingers and the palm; from the front view, the system uses depth information from both sensors simultaneously.
- Determination of grasping: Using depth data on users' hands and on objects collected from both sensors simultaneously, we found certain threshold values for recognizing the act of grasping.

2) Physical-digital object pairing: Our system performs time calculations between touched physical and digital objects versus touched object-object pairing. When the system recognizes the objects that the user wants to pair, the color is changed. The red color refers to the digital object and the physical object is blue. The recognized hand is shown in yellow using 3D point cloud data. The main steps are as follows.

- Time calculation: For pairing, the user maintains a touching posture for a few seconds after touching is recognized between the objects. The color is changed after the pairing.
- Tracking a paired set: To track paired objects, the system calculates 3D point cloud data continuously, which are provided by the real-time reconstruction module in Figure 3.12. The color of the tracked object is shown.
- Changing a paired set: To change a paired object set, a pairing gesture is made for some period of time. The user touches what he/she wants to pair. After

a few seconds, performing the pairing gesture (see Figure 3.10) will change the pair set as indicated by the color feedback.

3.5.3 Touch recognition techniques

In this subsection, we describe about recognition technology touching 3D object. Our system classified a kind of hand touch patterns. In addition, our system determines whether the object is touched or not in real time. We illustrate about the method recognizing the location of the touched part as well. The system uses RGB-D images from two Kinect cameras and point cloud data for recognizing.

3.5.3.1 Touch pattern of object

The pattern of the touch is various and it is different according to each research. In this thesis, we classify patterns into three types of touch in order to use tangible interaction with existing objects. Our system classifies into finger touch, hand touch and grasp. Finger and hand touch are top-down touch(see (a) and (b) of Figure 3.13). Finger touching only or finger and palm touching are distinguished.

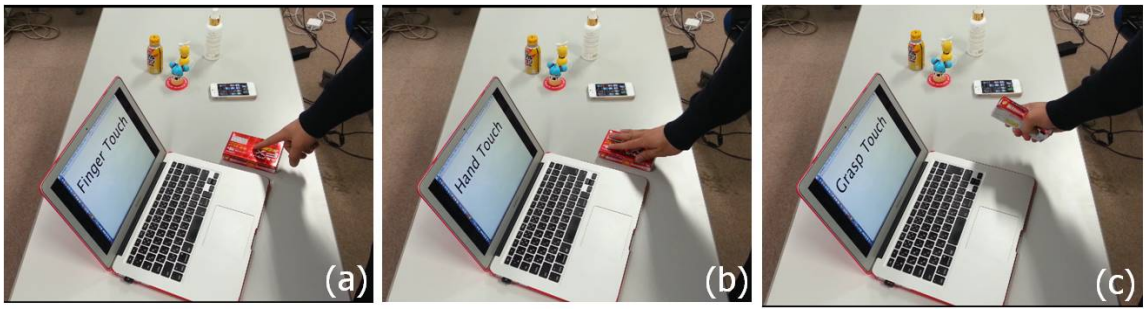

Figure 3.13: (a) Finger touch and (b) Hand touch and (c) Grasp

The above mentioned touch patterns are quite simple touch patterns. On the other hand, grasp is more complicated and has various patterns. It happens when we pick the object up or hold it. For example, picking a pencil and grasping a P.E.T bottle are considered grasp touches. Grasp touches can be classified according to object's hardness and size. We touch fingers and palm to the object when we pick the cup up and touch fingers only when we pick the pencil up. A grasp can be classified as finger touch or hand touch.
The system stores user's touched pattern and location value for making tangible objects

from existing objects. The touch patterns are classified as described above and become the fundamental initial point for the interaction.

3.5.3.2 Touch recognition flow

Our proposed system classifies three parts (see Figure 3.12). The system performs these methods in order. First is 3D calibration and reconstruction. The system performs calibration process on the data from RGB-D camera1 and camera2. The data from RGB-D camera2 rotates 180 degrees based on y-axis because camera1 and camera2's data location are opposite.

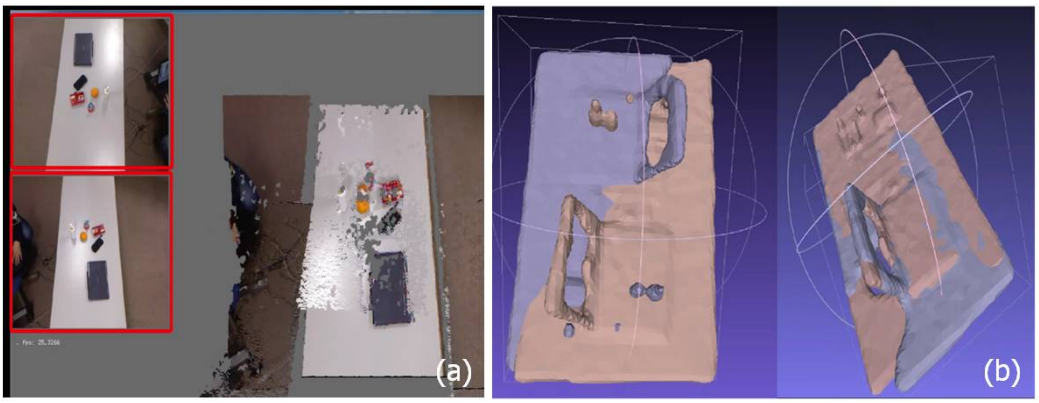

Figure 3.14: (a) Calibration of two camera's data and (b) Reconstructed surface

The system makes points in each data based on the 3D model and vision image when the system performs rotating (see (b) of Figure 3.14). An object id is generated using 3D shape and vision image data in real time. The system can recognize 3D location using point cloud data. We can store the touched position of the object by using these data. The stored database consists of a structure similar to Table 3.1. The system reconstructs hand model using point cloud data from two cameras. The hand index is assigned to each hand. The mesh and depth value of each finger in hands are stored in the database. It calibrates objects using vision and depth image from two cameras for generating the object id. There is 3D value such as meshes, depth as well from each camera. It provides 3D shape, location and vision image of each object. The system can discriminate objects using these data. In result, the stored 3D object database is used for touch recognition, touched position of the object and learning gesture as well. The second is touch recognition and gesture learning. The touch recognition recognizes by using the point cloud and reconstructed 3D model

estimation. Our system uses point cloud data when we do simple touch recognition. It is that case of finger touch or hand touch. The system recognizes the location of the hand and objects using depth based point cloud data. We defined the depth value of an object surface as $D_{surface}$. It has defined the maximum value as D_{max} that is slightly above the $D_{surface}$. It has defined upper part of finger joint as D_{min} as well.

The system determines when users touch the object surface when the finger joint is between D_{max} and D_{min} [61]. The touched area is point cloud data between a hand and an object. The hand touch is recognized because all finger joints value and palm value are between each $D_{surface}$ value. These recognized and touched data can be stored when the user wants to make touch gestures. The database is stored by our system consisting of structure similar to Table 3.2 for learning touch gestures. The table value is stored when the user makes a gesture with the touching object. The hand index in the Table 3.2 means discriminating touched hand with the object. The touched finger value with the object are stored as mesh, depth and vision value.The first index of the finger's array represents a thumb to little finger. The second index of the finger's array represents the joint of the finger. Palm value recognizes and stores when the user touches the object with finger and palm. The stored vision images and each value are used for recognizing objects as well.

$$Finger[i][j] - Threshold \leq Finger[i][j] \leq Finger[i][j] + Threshold(i \neq 0) \tag{3.2}$$

$$D_{min}[i][j] \leq D_{Finger[i][j]} \times 0.95 \leq D_{max}[i][j](i \geq 0) \tag{3.3}$$

Finally, the system provides learned gesture recognition and detects defined interactions. These methods refer to the stored database (see Table 3.2). The system recognizes touch gesture when the user does the same gesture as the user-defined touch gesture. However, it is difficult to touch exactly the same position as defined. Therefore, we implemented the recognition formula (see equation 3.2). The system tracks the touched position from

Table 3.1: 3D object database structure

	Kinect 1	**Kinect 2**
Object	Hand Index(Left or Right hand) Hand Value(Mesh, Depth value)	
	Mesh value Depth value Vision image	Mesh value Depth value Vision image

Table 3.2: Learning touch gesture database structure

	Kinect 1	**Kinect 2**
Object	Hand Index (Left or Right hand)	
	Finger [0][0,1,2](Thumb)	Finger [0][0,1,2](Thumb)
	Finger [1][0,1,2](Index)	Finger [1][0,1,2](Index)
	Finger [2][0,1,2](Middle)	Finger [2][0,1,2](Middle)
	Finger [3][0,1,2](Ring)	Finger [3][0,1,2](Ring)
	Finger [4][0,1,2](Little)	Finger [4][0,1,2](Little)
	Palm value	Palm value
	Vision image	Vision image

Table 3.3: System defined interaction database

	Object
Interaction	Interaction type Gesture-Function Pair ID Touch Pattern Vision image

thumb to little finger. Each touched finger between the thresholds (equation 3.2) can be detected. An appropriate value for the threshold is 0.3 for each depth value (x, y, z axis value). It had good results in all experiments. After detecting learned touch recognition, the system tracks the user's gesture and records mesh and point cloud motion. The touch motion gesture and pre-defined interaction can be paired by user. The interaction type is determined as well. The interaction returns feedback when the user does touch gesture using these databases.

3.5.3.3 Touch recognition technique

The touch recognition is conducted using point cloud data and 3D model estimation. Each finger is divided into three parts as the finger joint for robust touch recognition. Each finger joints have 3D point cloud data. Our system tracks the location of touch between joint and object in each joint. Our system calculates the number of touching point cloud data and their motion for each joint. Our system determines touch recognition based on a formula that found experimentally (see equation 3.3). The system determines whether 95 percentages of the finger joint cloud data are touched same position that the user defined or not for each finger joint. Our system can estimate the area that cannot see by camera because the system knows D_{min} and D_{max} value of each finger joint. The finger point cloud data between D_{min} and D_{max} is considered touched, even if cameras cannot see all the

Algorithm 3 Touch-Pair Algorithm

Input: Raw data of objects, 3D point cloud data, Segmented hands
Output: PairID, Object number, Collision points,

```
Initialization PairID = NULL, Object = NULL TouchPairMode = TRUE
if TouchPairMode == FALSE then
    return
end if
while IsHandsTracking != FALSE and IsObjectTracking != FALSE do
    if CollisionDetected then
        for ObjectDatabase do
            Save ObjectID of collide object
        end for
        TouchPairMode = FALSE
    end if
end while
```

areas. The system can recognize 3D touch area with objects using this proposed method. The error rate of recognition is low because each finger joints are managed separately. The system can determine using three finger joints data even if a one finger joint cannot recognize well. The system uses middle joint of each fingers mainly when the all three finger joints touched with object as well. Because the system can estimate the first and third joint point cloud data using middle joint data when all finger joints touched.

3.5.3.4 Touch-Pair potential applications

After pairing objects, the system can be used in various tangible interactions. We described tangible interactions using proposed touch recognition technique. We defined the interaction functions and learned touch gesture to the interaction in basically using touch pairing. We implemented five types of interaction. They are flight game, map control, music control, instrument and drawing using existing object touching gesture. We described in detail their function and gestures.

1) Flight game and map control interaction: We designed a flight joystick using physical objects (see (a) of Figure 3.15). Such a joystick can be used when playing a flight or gun game. Our system made a pairing between a notebook PC and a lotion bottle with a cap as a game controller. The game can be controlled by pushing the lotion bottle's cap (see (b) of Figure 3.15) and moving it. The movements can be recognized based on the cloud data for the object and the hand. We also designed a map controller with paired objects

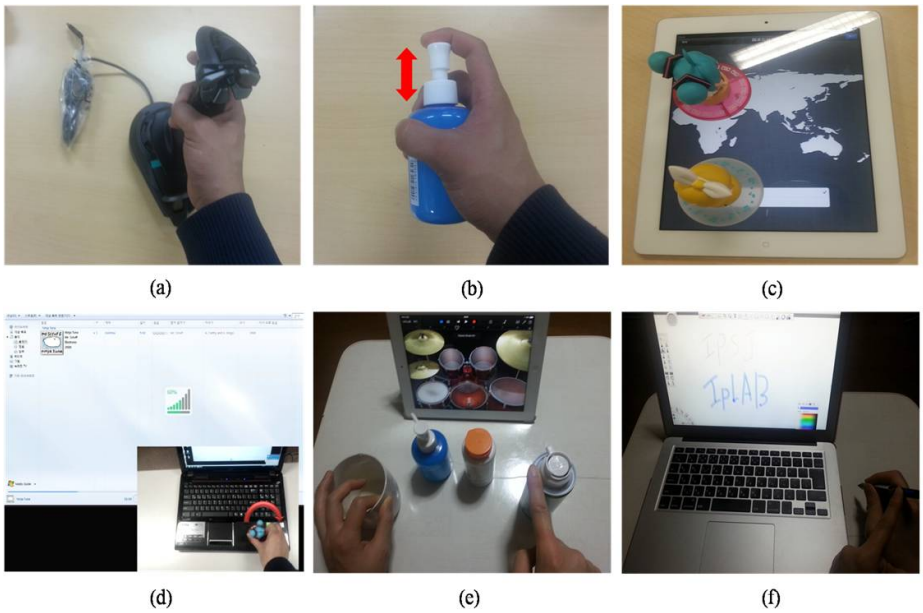

Figure 3.15: Potential applications of Touch-Pair

(see (c) of Figure 3.15). Our system tracks the two toys and the touched position; the map is moved to provide feedback.

2) Music player interaction: We implemented simple interaction. It is a volume controller using toys. The system stored by calibrating and reconstructing the toy object. The user did clockwise or counter clockwise rotating for learning gesture. The system stores touched location of the object and hand motion of gesture to the database. The user makes a connection between the gesture and volume up or down interactions. After that, the music player's volume is changed when the user does clockwise or counter clockwise touch gesture (see (d) of Figure 3.15).

3) Instrument interaction: We implemented instrument controller using existing objects. The objects are being the part of drum instruments in this interaction. The multiple objects touch recognition is conducted for doing learning gesture. These recognition use not only the depth and mesh value of objects, but also vision data of objects. Because it has to discriminate objects and multiple touch recognition with discriminated objects. There are eight instruments in drum on the pad (see (e) of Figure 3.15). Each existing object is defined controlling two instruments. The user defines touch gesture to the object. This interaction can control the volume and play styles by using learned gesture.

4) Drawing interaction: The drawing interaction is provided. The interaction is on the

existing pen object as input device. Our system tracks pen's movement to interact, drawing feedback. The drawing is performed when a user is writing action on the desk. We can write the character by writing action. The proposed interaction can change pen color and control thickness by using the touch pen (see (f) of Figure 3.15). The system obtains color changing feedback when the user draws after touching user defined location of the pen. We can extend to adding more functionality using touch recognition technique. It can be used as real tablet pen.

3.6 System evaluation

3.6.1 System setup

We conducted evaluations in two different system environments; Touch-Pair and 3D motion gesture using the MOI toolkit. Touch-Pair used two first version of Kinect sensors (see (a) of Figure 3.16), and the examples of 3D motion gesture used a Kinect second version (see (b) of Figure 3.16); one hand, two hand, hand-object gestures and interactions. The second version of Kinect provides 1920×1080 resolution of RGB and 512×424 resolution of Depth, while first version provided 640×480 resolution of RGB and 320×240 resolution of Depth [87]. Also, the degrees of view are improved; 57 degrees to 70 degrees in horizontal field, 43 degrees to 60 degrees in vertical field. Because the Kinect v2 brought big improvements in depth and RGB (i.e., almost 1.6 times in depth, 3 times in RGB), our system can capture the hand-object motion gesture by a Kinect v2. However, Only one sensor still

Figure 3.16: (a) Touch-Pair system setup and (b) 3D motion gesture system setup

has limitations for Touch-Pair; the limitation of view degrees. Therefore, we conducted the Touch-Pair evaluation by two Kinects, the 3D motion gestures evaluation by a Kinect v2.

3.6.2 Tangible experiment

The experiment is evaluating the effectiveness for tangibility of object using a depth camera. We evaluate seven different objects in Figure 3.22 (i.e., Toy, Black doll, Cube, Pet bottle, Smart phone, Camera). The classification accuracy is evaluated for finger touch, grasp (see Figure 3.13). The five volunteers are participated in conducting these evaluations. We explained each touch gestures and evaluation manners enough. The volunteers conducted 100 times for each object. The results were obtained as 93.4% for the finger collision detection, Over 80.2% for grasp collision detection. We conclude hardness and size of object has many effects on the results and precise grasp collision detection had difficulties in invisible areas. However, simple tangible interaction is no problem, the training makes the system robust to detect motion gestures.

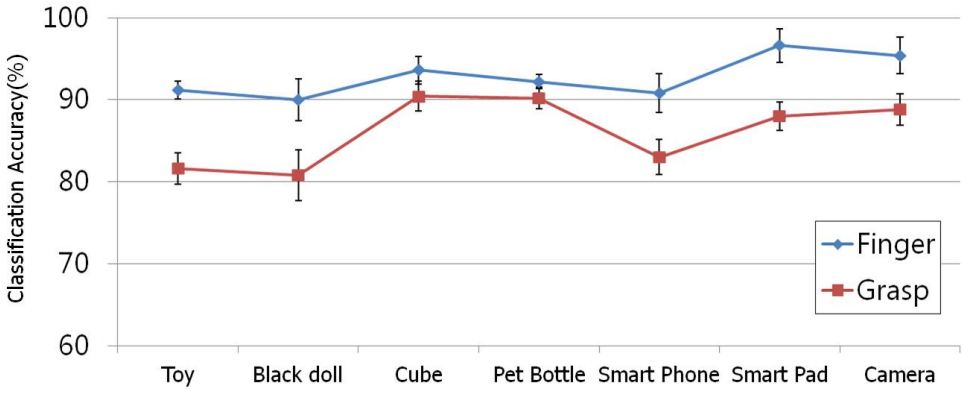

Figure 3.17: Tangible recognition accuracy results

3.6.3 Motion gesture experiment

The experiment is evaluating the effectiveness for one-hand, two-hand, hand-object gestures using a depth camera. We evaluate six different one-hand, two-hand respectively. It evaluated 100 times for each gesture, the result is classification accuracy. The results were obtained as over 92% for hand motion gesture in 1000 training samples (see Figure

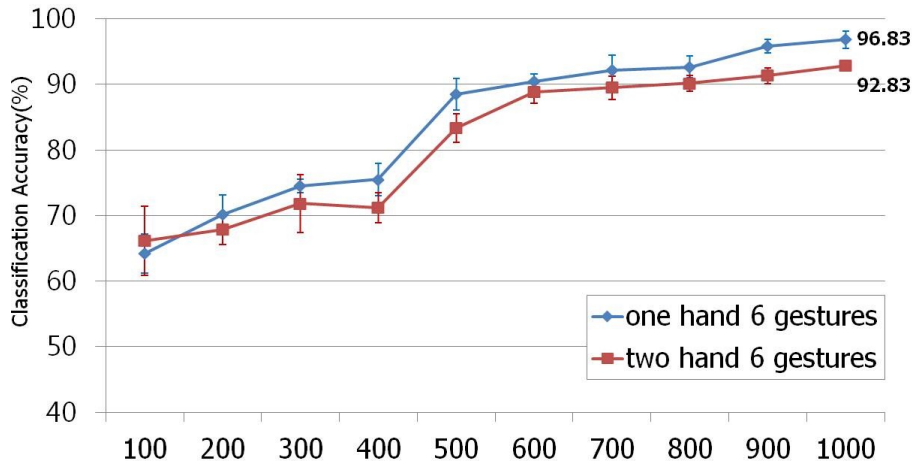

Figure 3.18: 3D motion gesture results1

3.18), over 84.5% for hand-object motion gesture in 1300 training samples (see Figure 3.19). One round of training makes almost 100 samples. We need 7 to 10 training rounds in common use. In short, our proposed methods are suitable for natural using with user-defined gestures.

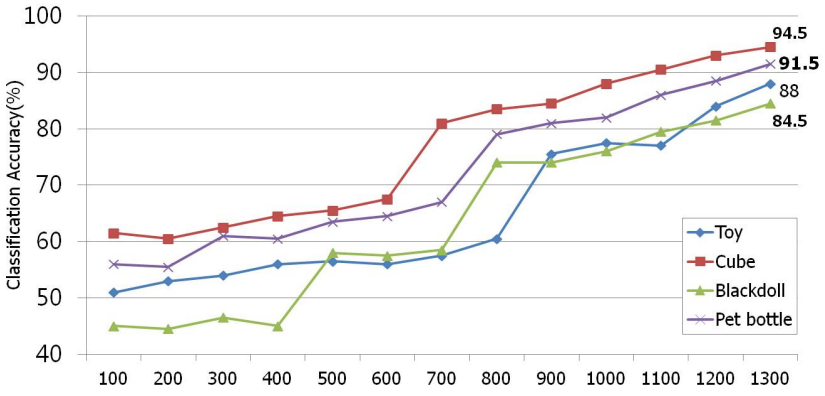

Figure 3.19: 3D motion gesture results2

3.6.4 Touch-Pair recognition accuracy experiment

Four physical objects and three digital objects were used. We evaluated finger touching, hand touching, grasping, and object-object touching for each object. The experiments were performed on computers (Intel Core i7 CPU, 2.5 GHz, and 8.0 GB RAM) using two

Microsoft Kinect sensors for Xbox.

We performed the experiments with 10 volunteers. We explained each touch pairing method sufficiently. Then a volunteer performed four touching gestures to each object 100 times. We defined three second as the period for completing pairing via touching. When the system recognized a pairing, it showed color feedback. Touch recognition success alone was not counted. The participants were allowed to touch physical objects only with the finger and digital objects only with the hand. Table 3.4 shows the average pairing recognition success for the 10 volunteers.

Our proposed system showed over 90% pairing recognition with the objects provided. We found that the average finger-based pairing recognition rate was higher than hand-based pairing. Finger pairing was recognized best when one or two fingers were used. Hand pairing requires checking whether the palm is touching. Thus, hand-based pairing recognition was less accurate than finger-based pairing. In addition, the success rate for touching a smart phone was lower than that for touching the other objects. This may have been because of the size of the object. Most adult hands are bigger than most smart phones. Thus, it becomes difficult for the system to find the positions of the finger and palm.

Table 3.4: Finger and hand touch pairing results

	Note PC		Smart Pad		Smart Phone	
	finger	*hand*	*finger*	*hand*	*finger*	*hand*
Toy	98.3%	95.4%	99.1%	95.3%	95.3%	93.2%
Black Doll	95.7%	93.3%	96.2%	94%	92.2%	90.1%
Green Cube	98.4%	96.7%	99.3%	95.7%	96%	94%
Pet Bottle	96.7%	95.3%	97.1%	95.4%	93%	91%

Table 3.5 shows the results for another pairing method, such as grasping and object-object pairing recognition. The experiments were performed in the same way as described in Table 3.4. Grasping-based pairing recognition accuracy was over 85% with the objects provided. This method uses point data from many directions. Generally, the front, side, and back surfaces of an object are touched when holding something in the hand. Thus, grasping has to be determined by analyzing the data from many directions.

Thus, on the whole, the recognition rate was low. We also found that the recognition rate differed by object size and hardness. A plastic toy, green cube, and bottle are relatively

Table 3.5: Grasping and object-object touch pairing recognition results

	Note PC		Smart Pad		Smart Phone	
	grasp	*object*	*grasp*	*object*	*grasp*	*object*
Toy	91.4%	94.3%	89.1%	98.1%	87.3%	94.2%
Black Doll	85.4%	95.4%	85.1%	97%	85.2%	91.1%
Green Cube	91.2%	96.2%	90.1%	96.7%	87%	96%
Pet Bottle	90.1%	97.1%	90.2%	97.4%	88%	94.3%

hard. However, a doll is very soft. When the user touches or grasps a softer object, the system has difficulty determining touch depth. Thus, recognition accuracy was lowest for the doll among all objects provided. However, generally, the recognition rate was high.

3.6.5 3D object recognition accuracy experiment

3.6.5.1 Single object recognition

In this experiment, we present results for 3D object recognition with a 3D object database. Eighteen existing objects were used for evaluation. We divide two group the objects by intensity. The group was divided three group again by object size. The size based group includes three objects in the result. The intensity value one means soft objects and value two means hard objects. The size value one means small size such as a pen, eraser and card and value two means normal size such as beverage can, cup and plastic lotion bottle. The three means bigger size such as cap, shoes, books, etc. We do not need to consider very big object because our system implemented the object on the desk (see

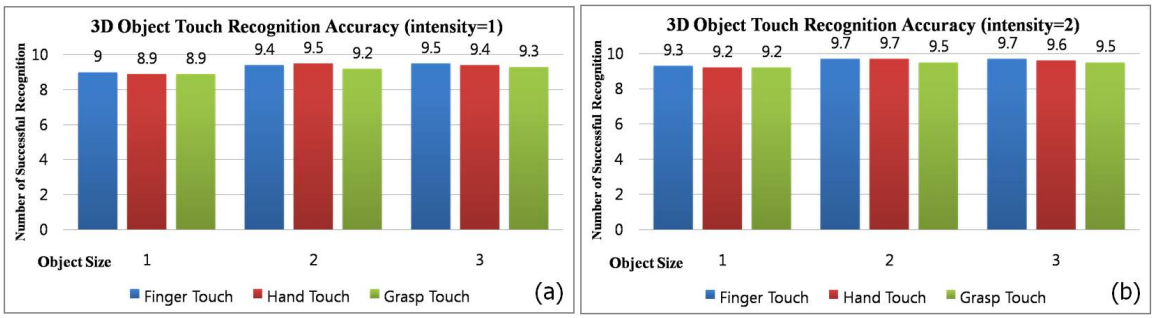

Figure 3.20: (a) Recognition accuracy in intensity one and (b) Accuracy in intensity two

Figure 3.11). The experiments were performed on a computer with two Intel Core i7 CPU 2.5GHz and 8.0 GB RAM computers and two Microsoft Kinect for Xbox 360. We performed the experiments with ten volunteers. We explained each touch method enough. We touch the object by each touch method and stored 3D object database to all objects. The place of object where volunteers touched is indicated with the sticker. The sticker instructs the place to be touched to volunteers in recognition experiment. A volunteer was performed 10 times three touching method to each object. We checked whether the system recognizes touching or not. The graph shows the average of succeeding number to each object from ten volunteers.

In result, we can obtain good recognition result in Figure 3.20. We can find out, the result is lower in size one with comparing another size. The grasp touch is lower in each size as well. We found out the reasons because small size object is difficult to catch all area from the camera. The area that cannot catch is estimated. The grasp touch recognition complicates comparing with the other touch recognition as well. Because the system has to consider over two sides when grasp touch is happening

We obtained higher results in (b) than (a) of Figure 3.20. All size and touch method are obtained over 90 percentages successful recognition results in (b) of Figure 3.20. The recognition rate of hard objects is higher than soft objects. Because The depth value was a little changed when the volunteer touched soft objects. In result, the hard and big size object was obtained almost highest recognition rate.

3.6.5.2 Multiple object recognition

The volunteers are chosen objects what they want to use. The object size and intensity was not considered for natural. Because the size and intensity of objects are not considered so much when we use existing object in real life. In this experiment, we already stored 3D object database to all objects and indicated with small sticker. They choose two objects at first. After that, they touch one object of them. The system checked whether the system recognizes touching or not and the correct object id (see Table 3.1). They choose two objects again what they want to use. After that, they do the same way. The object number is increased from three to five and did same way to experiment. The experiments were done ten times for each object number. The result graph shows the average of succeeding number with different touch method from ten volunteers. The result was obtained over 90

percentage for all situations (see Figure 3.21). The result shows that the number objects did not have an effect on the recognition accuracy. We limit the number of objects to five

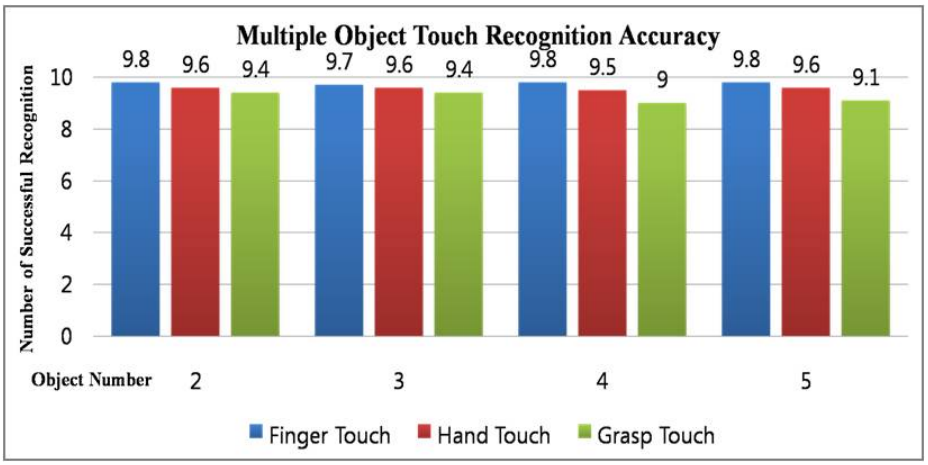

Figure 3.21: Multiple object touch recognition accuracy

because the size of a desk in this experiment. However, the recognition rate does not lower so much even if the object number is increased. The object size and intensity have an effect on the recognition accuracy more than object number.

3.7 Summary and discussion

3.7.1 Discussion

3.7.1.1 Touch-Pair

In our experiments, we found out that the touch and gesture recognition with 3D objects can be obtained good recognition accuracy rate. Our proposed method recognized 3D object touched location and learned gesture using RGB-D Kinect cameras without attaching additional sensors. We obtained good result over 90 percentage succeed recognition in all experiments. We found out that the size and intensity of objects have an effect on the experiment result. The number of objects had not an effect to the result so much. It means the estimated area did not recognize well relatively. The big size and hard object can capture all areas to touch recognition. The recognition rate is high because it is not an estimation. However, the results between each experiment do not need to consider so much for using natural tangible interactions. The proposed system can cover their interactions

and gestures.

In real life, we expected natural our actions to be interactions in general. For instance, a feedback happens when we grip the cup. The interaction occurs when we tap the objects once or twice. A feedback is happening when we do clockwise or counter clockwise gesture with objects as real controller. It is interested and useful when the actions that used in real life are used.

The proposed interactions are designed by considering these points. The natural gestures are connected with interactions. The music controller, instrument controller and drawing interaction are the function of being frequently used. The designed gestures are friendly and being frequently used as well. Therefore, the volunteers who participated in evaluation practiced well, and became expert to use the proposed gesture and function in learned gesture experiments. In result, we can obtain good results and proved system usability and robustness.

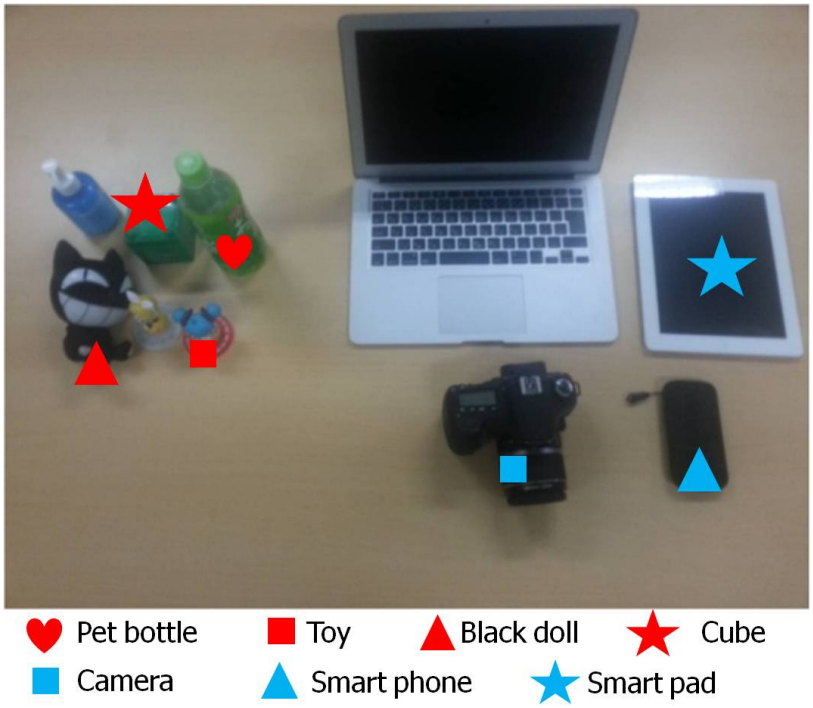

Figure 3.22: The objects used in evaluation

3.7.1.2 Tangible experiment

We evaluated the tangible experiments for demonstrating the robustness of recognition in touch. We obtained over 90%(SD=2.55%) recognition results in finger touch detection, and 80.8%(SD=3.11%) in grasp detection. As the results, we found out the hardness and size of objects have many effects to the recognition accuracy. The black doll showed worst recognition accuracy, because the object is soft; the system is difficult to detect touching. Also the toy showed lower recognition accuracy relatively; the toy is very small size, so it is difficult to detect precise touch gestures.
The grasp recognition accuracy showed lower than finger touch, because there are invisible areas (i.e., Dead-Zone, see Figure 3.23) when the volunteers conducted grasp gestures. The precise grasp touch is the challenges in tangible interface research areas, however, our system provided simple tangible gesture recognition enough. The detection of tangible gestures can be robust with system training.

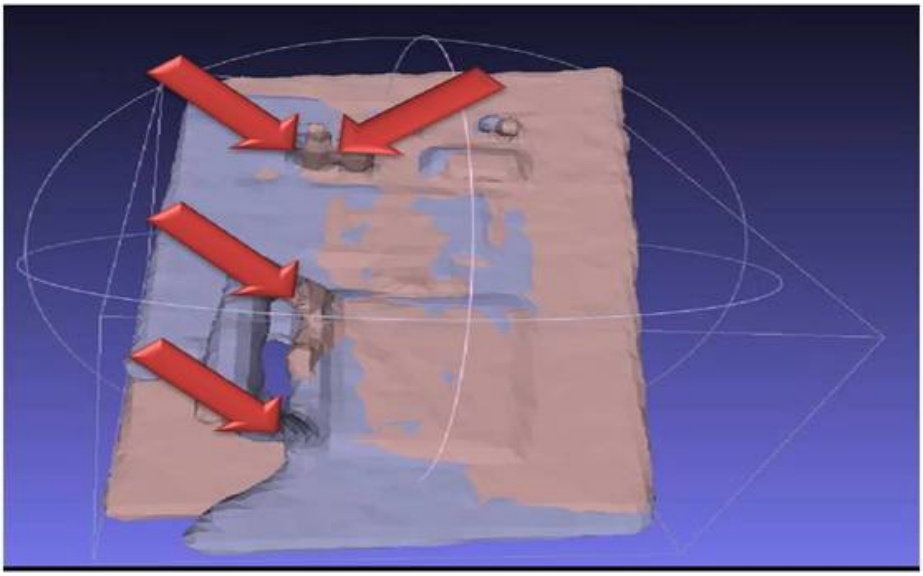

Figure 3.23: Dead-Zone(red arrows) of recognition

3.7.1.3 Motion gesture experiment

We evaluated three type experiments of 3D motion gestures; one hand, two hands, hand-object. In experiments, we conducted six gestures for one hand and two hands. We obtained over 94% classification accuracy. The accuracy was low until 400 training samples

shown as Figure 3.18. The accuracy was increased rapidly between 400 and 500 training samples. Because the six different swipe gestures are difficult to discriminate under 100 samples for a gesture in the DTW machine learning algorithm. We found out that one hundred samples for one gesture are suitable for robust recognition; the recognition accuracy is similar in 600 training samples and 1000 training samples. One round of training generates almost 100 samples; one round training takes 3 seconds. It means that 7 to 10 rounds of training is enough for six gestures. Therefore, our system provided practical use in real life; it takes a few minutes of training while another machine algorithm needs over 1000 training samples for a gesture. Also, we combined change amount of hand (i.e., octree-based change detection 3.1) and training samples. It provides more robust accuracy of recognition.
Also, we obtained over 84.5% from hand-object motion gestures. We conducted two motion gestures with different objects; clockwise, counter-clockwise motion gestures. In this experiment, we found out the hand-object gestures need more training samples (i.e., 1300 samples for practical use) than hand only motion. We analyzed the results; because the collision points of a hand and an object were changed dynamically, when the volunteers performed the gestures. It means the rotation and position of collision points changed fast and various directions.
Also, the 'Black doll' and 'Toy' obtained low accuracy relatively. We found out that the results were related on the size and hardness of the object; same reasons in the tangible experiments. Because of the difference of size and hardness, the point of increasing accuracy rapidly was different. The 'Cube' was increased the accuracy rapidly, between 600 and 700 training samples. The 'Pet bottle' was between 700 and 800 samples. However, all accuracy results in the 1300 training samples are enough to use in real life. The 1300 training takes 13 rounds of training and takes about 40 seconds for training. We conclude the motion gestures using our interface are enough to use in common.

3.7.2 Summary

In this chapter, we proposed new MOI toolkit interface for overcoming our 2D gesture recognition limitations. We designed the interface to use tangible and 3D motion gestures easily; from the beginners to the experts. Our interface designed with easy portability, extension, convenience and intuitiveness of use on the users' side. At this point, our interface provided as a library. The users or developers use our interface on their own just by input

the library.

Our interface has provided powerful modules; 3D recognition and tracking, gesture learning and prediction. Our modules can recognize hands, objects and collision in 3D based on PCL. The robust recognition provided possibilities of designing stable NUI; the technique is stable. Also, the machine learning modules can train an predict 3D gesture. The robust and practical technique provided new possibilities of user-defined gestures; user-friendly, extensional. Our proposed interface is more powerful in hand-object gestures and interactions. We implemented simple tangible interaction using a depth camera and 3D motion gestures. The advantages of our interface are that we can use both of them simultaneously. It meant that we can design the movements and touch of objects using our interface while previous researches only provided each.

In Touch-Pair interface, we designed the interface that connects the cyber world and real world easily, dynamically. The interface can design tangible and motion gesture interaction without additional sensors. It means that the users cannot feel artificial things; natural and intuitive. Also, we can change the object to preferred dynamically. We believe that Touch-Pair interface made big progress for NUI.

In result, we expect that everyone can access NUI easily. It means that we have provided many possibilities of next natural interface designing; many research (e.g., motion gesture for mobile devices [88]) can apply our interface and technologies with naturalness and robustness.

Chapter 4

Conclusion

We have described the researches of finger identification-based hand gesture interface for designing natural user interface. We introduced several interfaces using our proposed finger identification techniques: 1) Air-drawing 2) Multi-touch 3) Mapping, and 4) Finger mouse interface.

Also, we introduced promising interfaces using 3D motion gestures: 1) MOI toolkit 2) Hand motion 3) Hand-object, and 4) Touch-Pair interface.

From the research starting point, we found some limitations of the conventional approaches regarding designing natural user interface using hand gestures, then we proposed our approach, that is, to use finger identification based hand gestures to interact. After this, we obtained good results, our proposed interface, and implemented sample rich interfaces. However, we found out later that the finger identification based approach still has limitations to design three dimensional natural interfaces, thus we decided to implement point cloud based 3D motion gesture to overcome the limitations. Finnaly, we implemented a rich interface; MOI toolkit interface. We provided not only powerful 3D motion gesture techniques, but also an easy way to design 3D gestures using the interface. We provided examples of 3D motion gestures to show how to use the interface.

The experiments demonstrated the usability and showed that the proposed interface is more useful for designing natural user interface than the conventional gesture interfaces. We believe the proposed interfaces are a promising concept for the future development of designing natural interfaces and interactions.

4.1 Contributions

Our research goals are design of natural user interfaces that are easy to use for both beginners and experts, and letting the interaction come to users naturally. For these, we proposed finger identification-based robust hand gesture recognition. We implemented natural interactions (i.e., drawing, multi-touch, mapping, finger mouse).
Also, we proposed MOI toolkit for overcoming our finger identification-based gesture recognition limitations (i.e., 2D gestures). The toolkit was designed to easily allow to use tangible and 3D motion gesture by users. The one-hand, two-hand and hand-object motion gestures were designed in the toolkit. It supports both pre-defined 3D gestures (e.g., swipe, multi-touch) and user-defined 3D gestures. In hand-object, we designed an interface called "Touch-Pair". It provided natural interaction without additional sensors, and can change the preferred target object dynamically. We can easily connect the cyber world and the real world through the interface. Each proposed method was evaluated along with its gestures and interactions. In the evaluations, we obtained good results. We can conclude that our system is robust for recognition in 2D and 3D, and the designed interactions are natural and intuitive to use. It means that our methods have many possibilities in designing natural user interfaces.
In Chapter 2, we can summarize the contributions as below:

- We provided finger identification based robust hand gesture recognition method.
- Gesture version of depth-based natural interfaces were provided; drawing, multi-touch, mapping, finger mouse interface.
- Gesture input vocabularies were increased.

In Chapter 3, we can summarize the contributions as below:

- We provided point cloud based robust hand-object 3D motion gesture interfaces (i.e., MOI toolkit, Touch-Pair).
- 3D gesture input vocabularies were increased, and new possibilities were provided using our 3D motion gesture interface (i.e., interactions using both tangible and 3D motion gestures).
- Hand motion and hand-object motion gesture framework were provided as a library (i.e., increasing extension, easy to use both beginners and experts).

- Cyber and real world can be connected easily and naturally using a Touch-Pair interface.
- Practical machine training and prediction algorithm were provided; users can design their own 3D gestures easily in short time.
- The natural interfaces were provided; Google Street View, Google Earth and Music volume controller interfaces.

4.2 Future work

Our proposed interfaces demonstrated that the interfaces can be a promising technique for designing natural user interface. In the near future, our interface can apply to Augmented Reality(AR), Virtual Reality(VR) using 3D hand gestures [89], [90], [91]. In particular, the virtual reality of games can be designed with an Oculus Rift device [92], because our interface provided 3D hand tracking and gesture recognition. The interface using 3D hand gestures can provide immersive interaction and natural feedbacks. We can control virtual objects using our interface (i.e., MOI toolkit).

And also, we can expect to design natural and immersive AR interface with see-through type of Head Mounted Display(HMD) [93], [94]. The mid-air gestures and tangible interaction with objects can be designed using our interface in AR [95], [96]. Robust and natural gesture recognition using our interface provides better user experiences in VR and AR. Also, we can improve devices of the Touch-Pair interface; first version of Kinect for windows (i.e., Microsoft Kinect v1) to second version of Kinect for windows (i.e., Microsoft Kinect v2). The Touch-Pair using two Kinect v2 can provide more possibilities; it can capture 140 degrees in horizontal, 120 degree in vertical. We can expect to improve the robustness of recognition by using two Kinect v2 devices.

Finally, we will improve our toolkit, and release as a library. Our MOI toolkit has one-hand, two-hand and hand-object in the current version (v0.1). We aim at improving the collision algorithm and adding more gesture type such as object-object, body-object motion gesture [97] in version 0.2. The next version can be expected to provide users with easier methods to design motion gestures for body and objects as well.

Appendix A

Details of hand gesture recognition

A.1 Finger identification algorithm

Our system provided finger identification. The identification algorithm is divided by extending finger numbers (i.e., five fingertips or under four fingertips). The algorithm described in detail as below:

Algorithm 4 Finger Identification Algorithm

Input: Fingertip counting results ∈ Fingertip tracking raw data
Output: Finger identification results

```
1: Initialization IdentificatedResult = NULL, TwoHand = FALSE
2: if FingerCount ≥ 6 then
3:    TwoHand=TRUE
4: end if
5: if TwoHand!=TRUE then //One hand
6:    if FingerCount ≤ 4 then
7:       VectorMatching(HandPoseDB)
8:    else//Five fingertips extended
9:       while FingertipAndConvexHull.size!=NULL do
10:         MaxDistanceSet(FingertipAndConvexHull[i],Thumb,Index)
11:         MaxDistance(FingertipAndConvexHull[i],Thumb,Little)
12:         MinDistance(FingertipAndConvexHull[i],Index, Middle)
13:         Remaining finger is Ring finger
14:         i++
15:      end while
16:   end if
17: else//Two hand
18:    VectorMatching(HandPoseDB)
19: end if
```

A.2 Hand pose databases

30 × 5 hands pose set were implemented as databases. The combination of hand gestures with ring finger often not used, because the extending ring finger is an inconvenience; unnatural. Nevertheless, our databases have all cases of fingertips for hand gesture recognition, because it has possibilities that some users want to use the ring finger. We provided 30 fingertips as databases (see Figure A.1).

- One fingertip: 5 different cases
- Two fingertips: 10 different cases
- Three fingertips: 10 different cases
- Four fingertips: 5 different cases

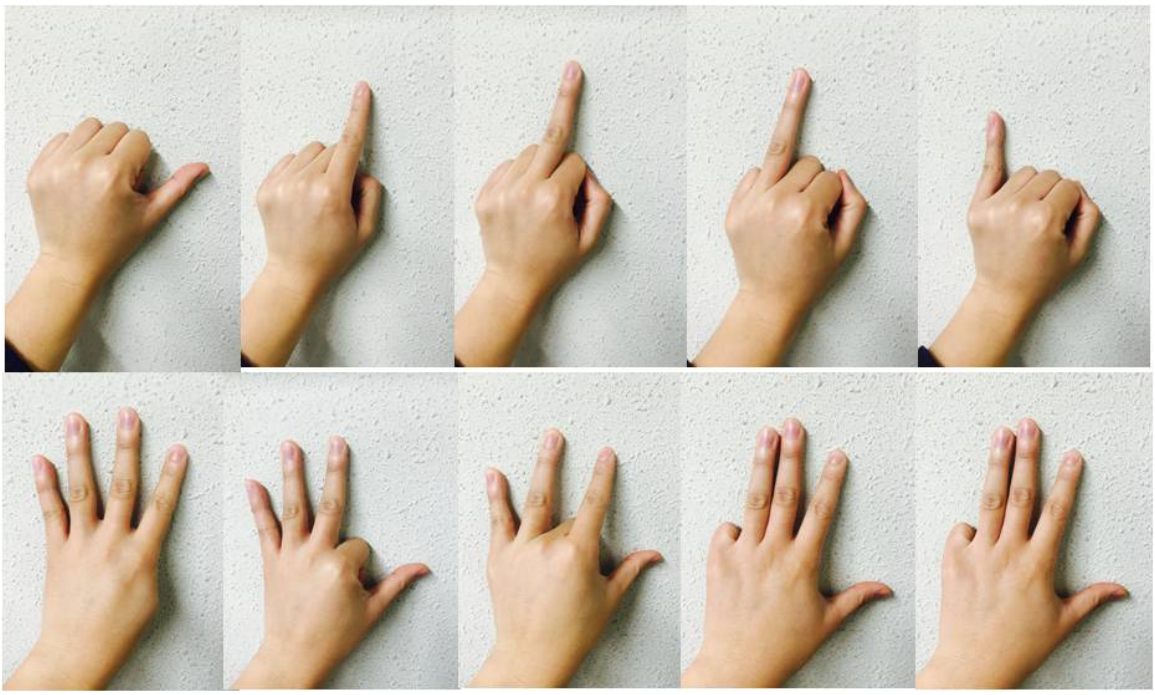

Figure A.1: Hand pose databases of one fingertip(top) and four fingertips(bottom)

Also, we divided five different hand cluster database by hand area of the cluster (see Figure A.2). The hand size is different when a user comes to the camera near, also the hand size is different by users. Because of these reasons, we defined five different hand cluster databases.

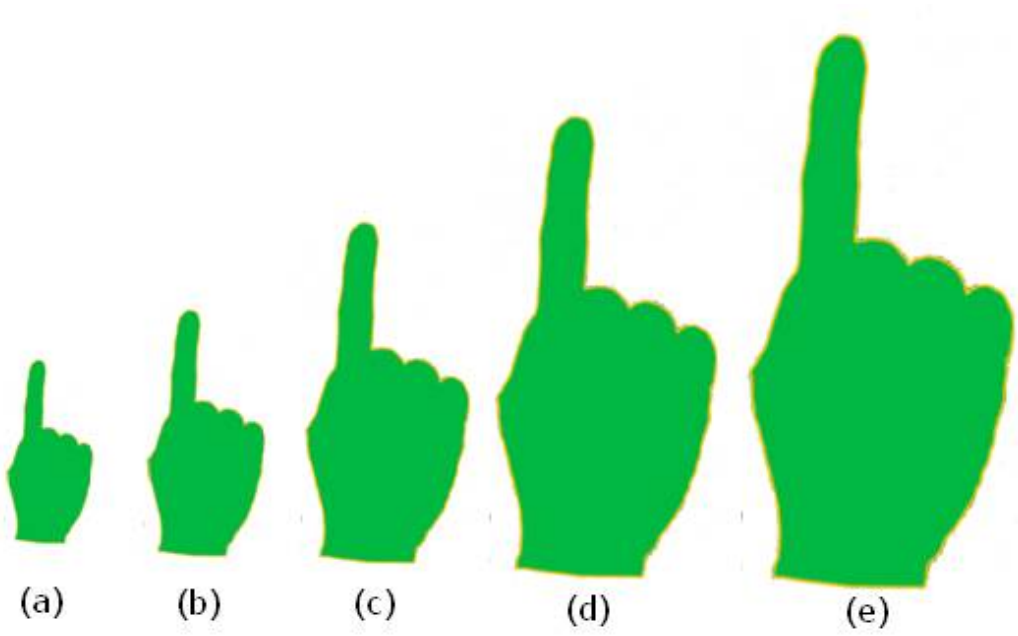

Figure A.2: A hand pose database of index finger example by area of hand cluster

(a) Hand cluster size between 80 and 90

(b) Hand cluster size between 90 and 100

(c) Hand cluster size between 100 and 110

(d) Hand cluster size between 110 and 120

(e) Hand cluster size between 120 and 130

Appendix B

Details of Touch-Pair interface

B.1 Touch-Pair

B.1.1 Pipeline

The "Touch-Pair" interface pipeline is almost same with the "MOI toolkit" interface. The difference between the two interfaces is re-pair mode by pressing a 'p' key. The main point in Touch-Pair interface is a dynamic pairing between a physical object and a digital object. Our interface provided the re-pairing mode for changing the paring target.

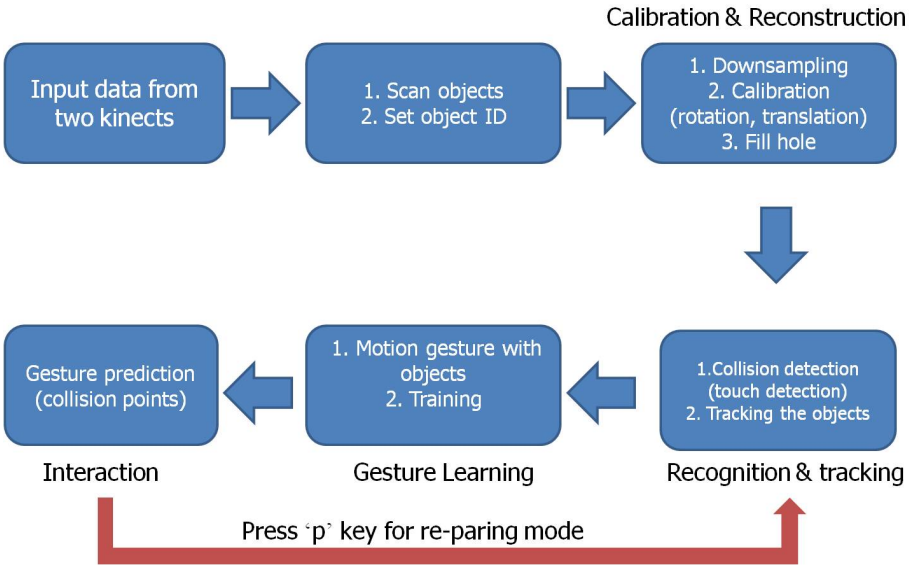

Figure B.1: Touch-Pair interface pipeline

B.1.2 Plane touch detection

Our system used plane touch recognition except grasp touch (see c of Figure 3.13). The plane touch recognition is based on corresponding the depth value of a hand and an object; the system estimates corresponding hand's point area and object's point area by computing each depth value.

$$D_{min} \leq HandDepthValue \leq D_{max} \tag{B.1}$$

D_{min}: Minimum depth value of the object

D_{max}: Maximum depth value of the object

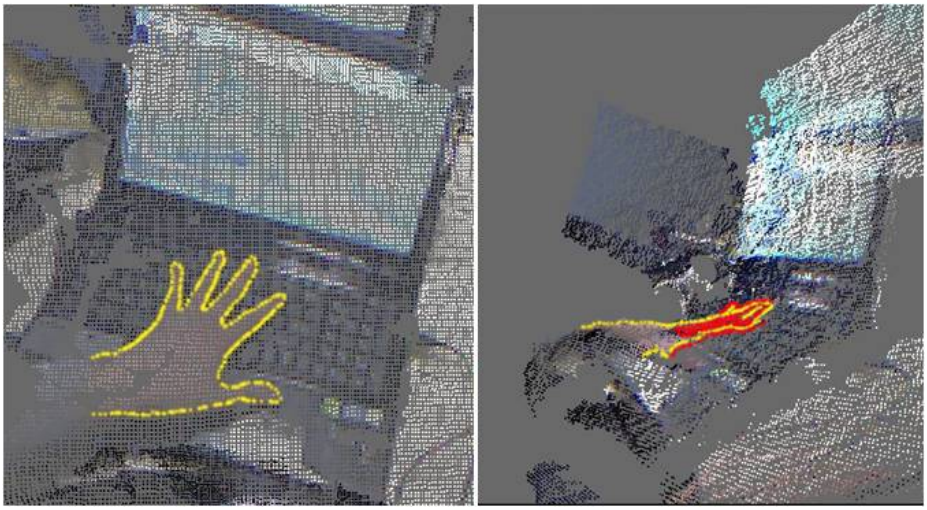

Figure B.2: Yellow line: a tracked hand(left), Red color: touch points

Bibliography

[1] List of gestures. [Online]. Available: https://en.wikipedia.org/wiki/List_of_gestures

[2] Octree in pcl. [Online]. Available: http://www.pointclouds.org/documentation

[3] Octree raycasting. [Online]. Available: https://youtu.be/aw0VExw2zy8

[4] N. Gillian and J. A. Paradiso, "The gesture recognition toolkit," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3483–3487, Jan. 2014.

[5] Microsoft kinect for windows sdk. [Online]. Available: https://dev.windows.com/en-us/kinect/tools

[6] Intel realsense sdk. [Online]. Available: https://dev.windows.com/en-us/kinect/tools

[7] S. W. Choi, W. J. Kim, and C. H. Lee, "Interactive display robot: Projector robot with natural user interface," in *Proceedings of the 8th ACM/IEEE International Conference on Human-robot Interaction(HRI '13)*, 2013, pp. 109–110.

[8] D. Wigdor and D. Wixon, "Brave nui world: Designing natural user interfaces for touch and gesture." Morgan Kaufmann Publishers Inc., 2011, pp. 759–760.

[9] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan, "Vision-based hand-gesture applications," in *Commun. ACM*, vol. 54, no. 2, 2011, pp. 60–71.

[10] N. Villaroman, D. Rowe, and B. Swan, "Teaching natural user interaction using openni and the microsoft kinect sensor," in *Proceedings of the 2011 Conference on Information Technology Education(SIGITE '11)*, 2011, pp. 227–232.

[11] Open computer vision library. [Online]. Available: http://opencv.org

[12] Point cloud library. [Online]. Available: http://www.pointclouds.org

[13] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology(UIST '11)*, 2011, pp. 559–568.

[14] M. Korn and J. Pauli, "Kinfu mot: Kinectfusion with moving objects tracking," in *Proceedings of the 10th International Conference on Computer Vision Theory and Applications (VISIGRAPP 2015)*, 2015, pp. 648–657.

[15] C. H. Yu, W. W. Peng, S. F. Yang-Mao, Y. Wang, W. Chinthammit, and H. B. L. Duh, "A hand gesture control framework on smart glasses," in *SIGGRAPH Asia 2015 Mobile Graphics and Interactive Applications(SA '15)*, 2015, pp. 16:1–16:4.

[16] Y. S. Hsiao, J. Sanchez-Riera, T. Lim, K. L. Hua, and W. H. Cheng, "Lared: A large rgb-d extensible hand gesture dataset," in *Proceedings of the 5th ACM Multimedia Systems Conference(MMSys '14)*, 2014, pp. 53–58.

[17] H. Karam and J. Tanaka, "Two-handed interactive menu: An application of asymmetric bimanual gestures and depth based selection techniques," in *Human Interface and the Management of Information. Information and Knowledge Design and Evaluation*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2014, vol. 8521, pp. 187–198.

[18] F. Dominio, M. Donadeo, G. Marin, P. Zanuttigh, and G. M. Cortelazzo, "Hand gesture recognition with depth data," in *Proceedings of the 4th ACM/IEEE International Workshop on Analysis and Retrieval of Tracked Events and Motion in Imagery Stream(ARTEMIS '13)*, 2013, pp. 9–16.

[19] M. Kamel Boulos, B. Blanchard, C. Walker, J. Montero, A. Tripathy, and R. Gutierrez-Osuna, "Web gis in practice x: a microsoft kinect natural user interface for google earth navigation," *International Journal of Health Geographics*, vol. 10, no. 1, 2011.

[20] L. Xu, Y. Fang, K. Wang, and J. Li, "Plug&touch: A mobile interaction solution for large display via vision-based hand gesture detection," in *Proceedings of the 20th ACM International Conference on Multimedia(MM '12)*, 2012, pp. 1177–1180.

[21] U. Lee and J. Tanaka, "Finger controller: Natural user interaction using finger gestures," in *Human-Computer Interaction. Interaction Modalities and Techniques*, ser. Lecture Notes in Computer Science, 2013, vol. 8007, pp. 281–290.

[22] Z. Ren, J. Meng, J. Yuan, and Z. Zhang, "Robust hand gesture recognition with kinect sensor," in *Proceedings of the 19th ACM International Conference on Multimedia(MM '11)*, 2011, pp. 759–760.

[23] H. Liang, J. Yuan, and D. Thalmann, "3d fingertip and palm tracking in depth image sequences," in *Proceedings of the 20th ACM International Conference on Multimedia(MM '12)*, 2012, pp. 785–788.

[24] A. Colley, J. Väyrynen, and J. Häkkilä, "In-car touch screen interaction: Comparing standard, finger-specific and multi-finger interaction," in *Proceedings of the 4th International Symposium on Pervasive Displays(PerDis '15)*, 2015, pp. 131–137.

[25] S. Murugappan, Vinayak, N. Elmqvist, and K. Ramani, "Extended multitouch: Recovering touch posture and differentiating users using a depth camera," in *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology(UIST '12)*, 2012, pp. 487–496.

[26] Y. Suzuki, K. Misue, and J. Tanaka, "A potential exploration of finger-specific interaction," in *Proceedings of the 10th Asia Pacific Conference on Computer Human Interaction(APCHI '12)*, 2012, pp. 389–392.

[27] A. Colley and J. Häkkilä, "Exploring finger specific touch screen interaction for mobile phone user interfaces," in *Proceedings of the 26th Australian Computer-Human Interaction Conference on Designing Futures: The Future of Design(OzCHI '14)*, 2014, pp. 539–548.

[28] J. Choi, B.-K. Seo, and J. I. Park, "Robust hand detection for augmented reality interface," in *Proceedings of the 8th International Conference on Virtual Reality Continuum and Its Applications in Industry(VRCAI '09)*, 2009, pp. 319–321.

[29] E. Ohn-Bar, C. Tran, and M. Trivedi, "Hand gesture-based visual user interface for infotainment," in *Proceedings of the 4th International Conference on Automotive User*

Interfaces and Interactive Vehicular Applications(AutomotiveUI '12), 2012, pp. 111–115.

[30] Z. Yang, Y. Li, Y. Zheng, W. Chen, and X. Zheng, "An interaction system using mixed hand gestures," in *Proceedings of the 10th Asia Pacific Conference on Computer Human Interaction(APCHI '12)*, 2012, pp. 125–132.

[31] S. Lenman, L. Bretzner, and B. Thuresson, "Using marking menus to develop command sets for computer vision based hand gesture interfaces," in *Proceedings of the Second Nordic Conference on Human-computer Interaction(NordiCHI '02)*, 2002, pp. 239–242.

[32] M. I. Boulabiar, T. Burger, F. Poirier, and G. Coppin, "A low-cost natural user interaction based on a camera hand-gestures recognizer," in *Human-Computer Interaction. Interaction Techniques and Environments*, ser. Lecture Notes in Computer Science, J. Jacko, Ed. Springer Berlin Heidelberg, 2011, vol. 6762, pp. 214–221.

[33] S. Chu and J. Tanaka, "Hand gesture for taking self portrait," in *Human-Computer Interaction. Interaction Techniques and Environments(HCII '11)*, July 2011, pp. 238–247.

[34] Mobile 3d camera. [Online]. Available: http://structure.io

[35] H. S. Yeo, B. G. Lee, and H. Lim, "Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware," in *Multimedia Tools Appl.*, vol. 74, no. 8, Apr. 2015, pp. 2687–2715.

[36] S. Andresen. (2013) Hand detection using color recognition. [Online]. Available: https://github.com/simena86/handDetectionCV

[37] H. Kenn, F. V. Megen, and R. Sugar, "A glove-based gesture interface for wearable computing applications," in *IFAWC 4th international forum on applied wearable computing*, 2007, pp. 169–177.

[38] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," in *ACM SIGGRAPH 2009 Papers*, 2009, pp. 63:1–63:8.

[39] C. Yang, Y. Jang, J. Beh, D. Han, and H. Ko, "Gesture recognition using depth-based hand tracking for contactless controller application," in *Consumer Electronics (ICCE)*, Jan 2012, pp. 297–298.

[40] J. Raheja, A. Chaudhary, and K. Singal, "Tracking of fingertips and centers of palm using kinect," in *Computational Intelligence, Modelling and Simulation (CIMSiM)*, Sept 2011, pp. 248–252.

[41] M. Tang. (2012) Recognizing hand gestures with microsoft kinect. [Online]. Available: http://uran.donetsk.ua/ masters/2012/fknt/sobolev/library/article5.pdf

[42] U. Lee and J. Tanaka, "Finger identification and hand gesture recognition techniques for natural user interface," in *Proceedings of the 11th Asia Pacific Conference on Computer Human Interaction(APCHI '13)*, 2013, pp. 274–279.

[43] Star trek gesture. [Online]. Available: https://en.wikipedia.org/wiki/Vulcan_salute

[44] U. Lee and J. Tanaka, "Touchpair : Dynamic analog-digital object pairing for tangible interaction using 3d point cloud data," in *The Seventh International Conference on Advances in Computer-Human Interactions(ACHI '14)*, 2014, pp. 166–171.

[45] M. Chen, G. AlRegib, and B.-H. Juang, "6dmg: A new 6d motion gesture database," in *Proceedings of the 3rd Multimedia Systems Conference(MMSys '12)*, 2012, pp. 83–88.

[46] D. Ashbrook and T. Starner, "Magic: A motion gesture design tool," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems(CHI '10)*, 2010, pp. 2159–2168.

[47] Leap motion and application store. [Online]. Available: https://www.leapmotion.com

[48] S. Cheema and J. J. LaViola, "Wizard of wii: Toward understanding player experience in first person games with 3d gestures," in *Proceedings of the 6th International Conference on Foundations of Digital Games(FDG '11)*, 2011, pp. 265–267.

[49] J. J. LaViola, Jr., "Context aware 3d gesture recognition for games and virtual reality," in *ACM SIGGRAPH 2015 Courses*, 2015, pp. 10:1–10:61.

[50] A. Bigdelou, L. Schwarz, T. Benz, and N. Navab, "A flexible platform for developing context-aware 3d gesture-based interfaces," in *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces(IUI '12)*, 2012, pp. 335–336.

[51] L. Schwarz, A. Bigdelou, and N. Navab, "Learning gestures for customizable human-computer interaction in the operating room," in *Medical Image Computing and*

Computer-Assisted Intervention MICCAI 2011, ser. Lecture Notes in Computer Science, G. Fichtinger, A. Martel, and T. Peters, Eds. Springer Berlin Heidelberg, 2011, vol. 6891, pp. 129–136.

[52] S. Lee, M. Sohn, D. Kim, B. Kim, and H. Kim, "Smart tv interaction system using face and hand gesture recognition," in *Consumer Electronics (ICCE), 2013 IEEE International Conference on*, Jan 2013, pp. 173–174.

[53] M. Takahashi, M. Fujii, M. Naemura, and S. Satoh, "Human gesture recognition system for tv viewing using time-of-flight camera," *Multimedia Tools and Applications*, vol. 62, no. 3, pp. 761–783, 2013.

[54] J. J. LaViola, Jr., "An introduction to 3d gestural interfaces," in *ACM SIGGRAPH 2014 Courses*, 2014, pp. 25:1–25:42.

[55] O. Hilliges, D. Kim, S. Izadi, M. Weiss, and A. Wilson, "Holodesk: Direct 3d interactions with a situated see-through display," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems(CHI '12)*, 2012, pp. 2421–2430.

[56] J. Sanchez-Riera, Y. S. Hsiao, T. Lim, K. L. Hua, and W. H. Cheng, "A robust tracking algorithm for 3d hand gesture with rapid hand motion through deep learning," in *Multimedia and Expo Workshops (ICMEW), 2014 IEEE International Conference on*, July 2014, pp. 1–6.

[57] P. O. Kristensson, T. Nicholson, and A. Quigley, "Continuous recognition of one-handed and two-handed gestures using 3d full-body motion tracking sensors," in *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces(IUI '12)*, 2012, pp. 89–92.

[58] G. Hackenberg, R. McCall, and W. Broll, "Lightweight palm and finger tracking for real-time 3d gesture control," in *Virtual Reality Conference (VR)*, March 2011, pp. 19–26.

[59] A. Colaço, A. Kirmani, H. S. Yang, N. W. Gong, C. Schmandt, and V. K. Goyal, "Mime: Compact, low power 3d gesture sensing for interaction with head mounted displays," in *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology(UIST '13)*, 2013, pp. 227–236.

[60] F. Klompmaker, K. Nebe, and A. Fast, "dsensingni: A framework for advanced tangible interaction using a depth camera," in *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction(TEI '12)*, 2012, pp. 217–224.

[61] A. D. Wilson, "Using a depth camera as a touch sensor," in *ACM International Conference on Interactive Tabletops and Surfaces(ITS '10)*, 2010, pp. 69–72.

[62] Q. Bonnard, P. Jermann, A. Legge, F. Kaplan, and P. Dillenbourg, "Tangible paper interfaces: Interpreting pupils' manipulations," in *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces(ITS '12)*, 2012, pp. 133–142.

[63] J. Ou, "Jing hua: Interacting with virtual flowers in a physical garden," in *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction(TEI '12)*, 2012, pp. 391–392.

[64] P. Wellner, "The digitaldesk calculator: Tangible manipulation on a desk top display," in *Proceedings of the 4th Annual ACM Symposium on User Interface Software and Technology(UIST '91)*, 1991, pp. 27–33.

[65] N. Takahiro, S. Yoichi, and K. Hideki, "Interactive object registration and recognition for augmented desk interface," in *IFIP conference on human-computer interface Interact*, March 2001, pp. 250–246.

[66] S. Itiro and M. Yoshiaki, "Iconstickers: Converting computer icons into real paper icons," in *Human-Computer Interaction: Ergonomics and User Interfaces, Proceedings of HCI International(the 8th International Conference on Human-Computer Interaction)*, August 1999, pp. 271–275.

[67] P. Ljungstrand, J. Redström, and L. E. Holmquist, "Webstickers: Using physical tokens to access, manage and share bookmarks to the web," in *Proceedings of DARE 2000 on Designing Augmented Reality Environments(DARE '00)*, 2000, pp. 23–31.

[68] M. Ono, B. Shizuki, and J. Tanaka, "Touch & activate: Adding interactivity to existing objects using active acoustic sensing," in *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology(UIST '13)*, 2013, pp. 31–40.

[69] H. Song, H. Benko, F. Guimbretiere, S. Izadi, X. Cao, and K. Hinckley, "Grips and gestures on a multi-touch pen," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems(CHI '11)*, 2011, pp. 1323–1332.

[70] R. B. Rusu, "Semantic 3d object maps for everyday manipulation in human living environments," Ph.D. dissertation, Computer Science department, Technische Universitaet Muenchen, Germany, October 2009.

[71] N. Gillian, B. Knapp, and S. O'Modhrain, "Recognition of multivariate temporal musical gestures using n-dimensional dynamic time warping," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2011, pp. 337–342.

[72] N. Gillian, R. B. Knapp, and S. O'Modhrain, "An adaptive classification algorithm for semiotic musical gestures," in *the 8th Sound and Music Computing Conference (SCM2011), Padova, Italy*, 2011.

[73] P. Azad, D. Münch, T. Asfour, and R. Dillmann, "6-dof model-based tracking of arbitrarily shaped 3d objects," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 5204–5209.

[74] C. Choi and H. Christensen, "Rgb-d object tracking: A particle filter approach on gpu," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 2013, pp. 1084–1091.

[75] C. Choi and H. I. Christensen, "Robust 3d visual tracking using particle filtering on the special euclidean group:a combined approach of keypoint and edge features," in *The International Journal of Robotics Research*, 2012, pp. 498–519.

[76] Documentations and tutorials of pcl object tracking in real time. [Online]. Available: http://pointclouds.org/documentation/tutorials/tracking.php

[77] R. Wimmer and S. Boring, "Handsense: Discriminating different ways of grasping and holding a tangible user interface," in *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction(TEI '09)*, 2009, pp. 359–362.

[78] R. Wimmer, "Flyeye: Grasp-sensitive surfaces using optical fiber," in *Proceedings of the Fourth International Conference on Tangible, Embedded, and Embodied Interaction(TEI '10)*, 2010, pp. 245–248.

[79] M. Sato, I. Poupyrev, and C. Harrison, "Touché: Enhancing touch interaction on humans, screens, liquids, and everyday objects," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems(CHI '12)*, 2012, pp. 483–492.

[80] C. Harrison, M. Sato, and I. Poupyrev, "Capacitive fingerprinting: Exploring user differentiation by sensing electrical properties of the human body," in *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology(UIST '12)*, 2012, pp. 537–544.

[81] L. P. Cheng, M. H. Lee, C. Y. Wu, F. I. Hsiao, Y. T. Liu, H. S. Liang, Y. C. Chiu, M. S. Lee, and M. Y. Chen, "Irotategrasp: Automatic screen rotation based on grasp of mobile devices," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems(CHI '13)*, 2013, pp. 3051–3054.

[82] M. Goel, J. Wobbrock, and S. Patel, "Gripsense: Using built-in sensors to detect hand posture and pressure on commodity mobile phones," in *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology(UIST '12)*, 2012, pp. 545–554.

[83] Openframeworks. [Online]. Available: http://www.openframeworks.cc

[84] Openni. [Online]. Available: http://structure.io/openni

[85] M. Berger, J. A. Levine, L. G. Nonato, G. Taubin, and C. T. Silva, "A benchmark for surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 2, pp. 20:1–20:17, Apr. 2013.

[86] S. Schaefer, T. McPhail, and J. Warren, "Image deformation using moving least squares(siggraph '06)," 2006, pp. 533–540.

[87] How does the kinect v2 compare to the kinect v1(hardware specifications). [Online]. Available: http://zugara.com/how-does-the-kinect-2-compare-to-the-kinect-1

[88] J. Ruiz and Y. Li, "Doubleflip: A motion gesture delimiter for interaction," in *Adjunct Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology(UIST '10)*, 2010, pp. 449–450.

[89] Y. Jang, S. T. Noh, H. J. Chang, T. K. Kim, and W. Woo, "3d finger cape: Clicking action and position estimation under self-occlusions in egocentric viewpoint," *Visual-*

ization and Computer Graphics, IEEE Transactions on, vol. 21, no. 4, pp. 501–510, April 2015.

[90] M. Lee, M. Billinghurst, W. Baek, R. Green, and W. Woo, "A usability study of multimodal input in an augmented reality environment," *Virtual Reality*, vol. 17, no. 4, pp. 293–305, 2013.

[91] H. Bai, G. Lee, and M. Billinghurst, "Free-hand gesture interfaces for an augmented exhibition podium," in *Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction(OzCHI '15)*, 2015, pp. 182–186.

[92] Oculus rift. [Online]. Available: https://www.oculus.com

[93] K. Hong, J. Yeom, C. Jang, J. Hong, and B. Lee, "Full-color lens-array holographic optical element for three-dimensional optical see-through augmented reality," *Opt. Lett.*, vol. 39, no. 1, pp. 127–130, Jan 2014.

[94] D. Cummings, G. Lucchese, M. Prasad, C. Aikens, J. Ho, and T. Hammond, "Haptic and ar interface for paratrooper coordination," in *Proceedings of the 13th International Conference of the NZ Chapter of the ACM's Special Interest Group on Human-Computer Interaction(CHINZ '12)*, 2012, pp. 52–55.

[95] S. Neale, W. Chinthammit, C. Lueg, and P. Nixon, "Natural interactions between augmented virtual objects," in *Proceedings of the 23rd Australian Computer-Human Interaction Conference(OzCHI '11)*, 2011, pp. 229–232.

[96] J. H. Seo, J. Storey, J. Chavez, D. Reyna, J. Suh, and M. Pine, "Arnatomy: Tangible ar app for learning gross anatomy," in *ACM SIGGRAPH 2014 Posters*, 2014, pp. 25:1–25:1.

[97] T. Chaves, L. Figueiredo, A. Gama, C. de Araujo, and V. Teichrieb, "Human body motion and gestures recognition based on checkpoints," in *Virtual and Augmented Reality (SVR), 2012 14th Symposium on*, 2012, pp. 271–278.

List of Publications

Journals

Unseok Lee and Jiro Tanaka, "Touch Recognition Technique for Dynamic Touch Pairing System and Tangible Interaction with Real Objects," *International Journal on Advances in Intelligent Systems*, Vol.7, No.3 & 4, 2014, pp.482-492.

Conference proceedings

Unseok Lee and Jiro Tanaka. "Finger Controller: Natural User Interaction using Finger Gestures," *Human-Computer Interaction, Part IV, HCII2013*, pp.281-290, Las Vegas, USA, July 21-26, 2013.

Unseok Lee and Jiro Tanaka. "Finger Identification and Hand Gesture Recognition Techniques for Natural User Interface," *APCHI '13 Proceedings of the 11th Asia Pacific Conference on Computer Human Interaction*, pp.274-279, Bangalore, India, September 24-27, 2013.

Unseok Lee and Jiro Tanaka. "TouchPair : Dynamic Analog-Digital Object Pairing for Tangible Interaction using 3D Point Cloud Data," *The Seventh International Conference on Advances in Computer-Human Interactions* , pp.166-171, Barcelona, Spain, March 23-27, 2014.