Mid-air hand interaction with optical tracking for 3D modelling

Cui, Jian

2019

Cui, J. (2019). Mid-air hand interaction with optical tracking for 3D modelling. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/83267>

<https://doi.org/10.32657/10220/48020>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**MID-AIR HAND INTERACTION WITH OPTICAL
TRACKING FOR 3D MODELLING**

CUI JIAN

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

2019

CUI JIAN

MID-AIR HAND INTERACTION WITH OPTICAL TRACKING FOR 3D MODELLING

CUI JIAN

School of Computer Science and Engineering

A thesis submitted to
Nanyang Technological University & Technische Universität Darmstadt
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2019

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

[Input Date Here]

8-Feb.-2019

.....

Date

[Input Signature Here]

.....

[Input Name Here]

Cui Jian

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

[Input Date Here]

8.02.2019

Date

[Input Supervisor Signature Here]

A. Sourin

[Input Supervisor Name Here]

Alexei Sourin

Authorship Attribution Statement

This thesis contains material from 4 paper(s) published in the following peer-reviewed journal(s) and conferences where I was the first and/or corresponding author.

Parts of each Chapter was published as Jian Cui, Alexei Sourin, “Mid-air Interaction with Optical Tracking for 3D Modeling.” *Computers & Graphics*, Volume 74, Pages 1-11, 2018, ISSN 0097-8493, <https://doi.org/10.1016/j.cag.2018.04.004>.

The contributions of the co-authors are as follows:

- I prepared the manuscript drafts. The manuscript was revised by Prof. Alexei Sourin.

Parts of Chapter 7 was published as Jian Cui and Alexei Sourin, “Interactive Shape Modeling Using Leap Motion Controller.” *SIGGRAPH Asia 2017 Technical Briefs*, ACM, 2017.

The contributions of the co-authors are as follows:

- I prepared the manuscript drafts. The manuscript was revised by Prof. Alexei Sourin.

Parts of Chapter 4 was published as Jian Cui and Alexei Sourin, "Feasibility Study on Free Hand Geometric Modelling Using Leap Motion in VRML/X3D," 2014 International Conference on Cyberworlds (CW), 2014.

The contributions of the co-authors are as follows:

- I prepared the manuscript drafts. The manuscript was revised by Prof. Alexei Sourin.

Parts of Chapter 5 was published as Jian Cui, Arjan Kuijper, Dieter W. Fellner, and Alexei Sourin, “Exploration of natural free-hand interaction for shape modeling using leap motion controller.” 2016 International Conference on Cyberworlds (CW), 2016.

The contributions of the co-authors are as follows:

- I prepared the manuscript drafts. The manuscript was revised by Prof. Alexei Sourin and Prof. Arjan Kuijper. The project was coordinated by Prof. Dieter W. Fellner.

[Input Date Here]

8-Feb.-2019

.....

Date

[Input Signature Here]

.....

[Input Name Here]

Cui Jian

Abstract

Compared to common 2D interaction done with mouse and other 2D-tracking devices, 3D hand tracking with low-cost optical cameras can provide more degrees of freedom, as well as natural gestures when shape modeling and assembling are done in virtual spaces. However, though quite precise, the optical tracking devices cannot avoid problems intrinsic to hand interaction, such as hand tremor and jump release, and they also introduce an additional problem of occlusion. This thesis investigates whether interaction during 3D modeling can be improved by using optical sensors so that 3D tasks can be performed in a way similar to interaction in real life and as efficient as when using common 2D-tracking based interaction while still minimizing the intrinsic problems of precise hand manipulations and optical problems.

After surveying the relevant works and analyzing technical capabilities of the commonly available optical sensors, two approaches are thoroughly investigated for the natural mid-air hand interaction in precise 3D modeling – they are collision-based and gesture-based interaction. For collision-based methods, a set of virtual interaction techniques is proposed to realistically simulate real-life manipulation and deformation with one and two hands. For gesture-based interaction, a core set of interaction techniques is also devised which allows natural real-life interaction ways to be used. In addition, algorithms are proposed for both collision-based and gesture-based interaction to enhance the precision while minimizing the problems of hand tremor and jump release. However, the results show that virtual interaction designed with collision-based methods is always slower than real-

life interaction due to missing force feedback. Although common gesture-based interaction is less affected by its problem, it still cannot completely get rid of the problems of occlusion and jump release.

Eventually, a new method of gesture-based interaction is proposed to use hands in a way similar to how it is done when playing the Theremin – an electronic musical instrument controlled without physical contact by hands of the performer. It is suggested that the dominant hand controls manipulation and deformation of objects while the non-dominant hand controls grasping, releasing and variable precision of interaction. Based on this method, a generic set of reliable and precise gesture-based interaction techniques is designed and implemented for various manipulation and deformation tasks. It is then proved with the user studies that for the tasks involving 3D manipulations and deformations, the proposed way of hand interaction is easy to learn, not affected by the common problems of hand tracking, as well as more convenient and faster than common 2D interaction done with mouse for some 3D tasks.

Acknowledgements

First of all, I would like to thank my supervisor Associate Professor Alexei Sourin for his invaluable instructions and guidance. I am grateful to him for his help throughout my time as a research student at Nanyang Technological University. I also thank Professor Arjan Kuijper for giving me valuable advice and guiding my research at TU Darmstadt. It is credit to their efforts and ideas that my research has reached this point.

I also wish to express my gratitude to the School of Computer Science and Engineering and Nanyang Technological University for providing me the opportunity and financial support to further my study and research. Special thanks go to the Joint PhD Degree Programme NTU - TU Darmstadt and Fraunhofer Singapore for supporting my project.

I wish to thank my parents for their support. I would like to thank my wife Jin Luying for her selfless concern and encouragement. Praises and thanks to the God, the Almighty, for His showers of blessings throughout my research work to complete the research successfully.

谨以此论文献给我的妻子金璐颖女士

TABLE OF CONTENTS

CHAPTER 1	INTRODUCTION	1
1.1	BACKGROUND AND MOTIVATION	1
1.2	PROPOSED APPROACH.....	3
1.3	REPORT ORGANIZATION.....	4
CHAPTER 2	SHAPE MODELING – FROM REAL-LIFE TO DIGITAL	5
2.1	MODELING FINE SHAPES WITH HUMAN HANDS IN REAL LIFE	5
2.2	SHAPE MODELING USING 2D TRACKING DEVICES.....	9
2.3	3D HAND MOTION TRACKING TECHNOLOGY	12
2.3.1	<i>Optical tracking technologies.....</i>	<i>12</i>
2.3.2	<i>Glove-based hand tracking systems.....</i>	<i>16</i>
2.4	NATURAL HAND INTERACTION ALGORITHMICALLY	19
2.4.1	<i>Collision-based interaction</i>	<i>19</i>
2.4.2	<i>Gesture-based interaction.....</i>	<i>22</i>
2.5	SUMMARY	26
2.6	FURTHER RESEARCH PLANS.....	28
CHAPTER 3	OPTICAL SENSORS SELECTION AND TEST	30
3.1	OVERVIEW OF OPTICAL SENSORS FOR HAND TRACKING.....	30
3.2	TESTING OF THE SELECTED SENSORS.....	32
3.2.1	<i>Experiment Setup</i>	<i>32</i>
3.2.2	<i>Tasks and procedures.....</i>	<i>33</i>
3.2.3	<i>Results</i>	<i>35</i>
3.3	SUMMARY AND DESIGNING GUIDELINES	38
CHAPTER 4	NATURAL COLLISION-BASED INTERACTION	40
4.1	OPERATIONS TO BE STUDIED	40
4.2	COLLISION BETWEEN VIRTUAL HANDS AND OBJECTS.....	41
4.3	TAKING.....	42
4.4	POSITIONING	44
4.5	RELEASING	45
4.6	DEFORMATION	46
4.7	USING PHYSICS ENGINE WITH UNITY	47
4.7.1	<i>Taking.....</i>	<i>49</i>
4.7.2	<i>Positioning.....</i>	<i>53</i>
4.7.3	<i>Releasing</i>	<i>55</i>
4.7.4	<i>Deformation</i>	<i>56</i>
4.8	INTERACTION WITHOUT A PHYSICS ENGINE.....	58
4.8.1	<i>Taking.....</i>	<i>59</i>
4.8.2	<i>Positioning.....</i>	<i>59</i>

4.8.3	<i>Releasing</i>	62
4.8.4	<i>Deformation with one and two hands</i>	64
4.8.5	<i>Additional algorithm on virtual finger correction</i>	67
4.9	IMPLEMENTATION	71
4.9.1	<i>Platform selection</i>	71
4.9.2	<i>Plugin implementation</i>	73
4.9.3	<i>Implementation of mathematical models</i>	75
4.9.4	<i>Code structure on FVRML</i>	77
4.10	USER STUDY	79
4.10.1	<i>Tasks</i>	80
4.10.2	<i>Apparatus</i>	81
4.10.3	<i>Participants</i>	82
4.10.4	<i>Procedure</i>	82
4.10.5	<i>Results</i>	82
4.11	SUMMARY	84
CHAPTER 5	NATURAL GESTURE-BASED INTERACTION	86
5.1	OBJECTIVE	86
5.2	DESIGN OF INTERACTION TECHNIQUES	87
5.2.1	<i>Taking</i>	87
5.2.2	<i>Positioning</i>	88
5.2.3	<i>Releasing</i>	92
5.2.4	<i>Deformation with one or two hands</i>	93
5.3	USER TEST	95
5.3.1	<i>Tasks</i>	95
5.3.2	<i>Procedure</i>	97
5.3.3	<i>Implementation details</i>	97
5.3.4	<i>Results</i>	98
5.4	A SECOND USER TEST	100
5.4.1	<i>Functionality and interaction design</i>	100
5.4.2	<i>Implementation</i>	102
5.4.3	<i>Tasks</i>	104
5.4.4	<i>Procedure</i>	104
5.4.5	<i>Results</i>	105
5.5	SUMMARY	107
CHAPTER 6	MAKING SHAPES LIKE PLAYING MUSIC	109
6.1	PRINCIPLES OF INTERACTION	109
6.2	DESIGN OF THE ACTUAL GESTURES AND INTERACTION USING LEAP MOTION CONTROLLER	112
6.2.1	<i>Motion space</i>	112
6.2.2	<i>Taking</i>	113
6.2.3	<i>Positioning</i>	114
6.2.4	<i>Releasing</i>	117

6.2.5	<i>Deformation</i>	118
6.2.6	<i>Additional operations for interaction control</i>	127
6.2.7	<i>Adaptive calibration algorithm</i>	129
6.3	EVALUATION	132
6.3.1	<i>Task</i>	133
6.3.2	<i>Apparatus</i>	135
6.3.3	<i>Procedure</i>	136
6.3.4	<i>Results</i>	137
6.4	DISCUSSION	142
6.5	SUMMARY	143
CHAPTER 7	CONCLUSION	145
CHAPTER 8	FUTURE WORKS	148
8.1	SEPARATING DOFS OF THE VIRTUAL HAND AMONG TWO HANDS	149
8.2	MULTIPLE SENSORS SETUP	150
REFERENCES		152

LIST OF FIGURES

Figure 2-1 Hand joints and their degrees of freedom	5
Figure 2-2 Motion of hand joints: (a) and (b) are motion capabilities of fingers; (c), (d) and (e) are motion capabilities of the thumb.	6
Figure 2-3 Glove-based systems. (a) CyberGlove. Figure is from website [38]. (b) HumanGlove. Figure is from website [39]. (c) 5DT Data Glove. Figure is from website [40]. (d) PinchGLove. Figure is from website [41].	17
Figure 3-1 Setups of sensors. (a) Intel RealSense SR300. (b) uSens Fingo. (c) Desktop mode of Leap Motion controller. (d) VR mode of Leap Motion controller.	33
Figure 3-2 Hand gestures for testing. (a)-(d) correspond to Gesture I–IV. Intel RealSense SR300 is used to illustrate the gestures. Hand orientation was changed according to other setups of sensors.	35
Figure 4-1 The pinching gesture that is implemented in the interaction engine for the operation of taking.	50
Figure 4-2 The geometrical mesh models created in Autodesk Maya and their corresponding colliders generated in Unity. (a), (b) and (c) are the original mesh models of the subdivided cube, the human lower yaw model, and the human skull model respectively in Autodesk, while (d), (e) and (f) are their corresponding colliders generated by Unity.	51
Figure 4-3 The efficiency test of the taking operation. The scene was occupied with 200 skull models.	52
Figure 4-4 Testing on the operation of positioning. (a) Taking-based ways of positioning. (b) Non-prehensile interaction ways.	54
Figure 4-5 Testing on soft-body deformation. (a) The object deformation when it collides with the ground. (b) Deformation by the palm. (c) Deformation by fingers. (d) Model crash.	57
Figure 4-6 Illustration of the positioning technique with one and two hands. (a) Local coordinate system of a hand. (b) The positioning technique with one hand. (c) The positioning technique with two hands.	60
Figure 4-7 Turning an object around an axis.	62
Figure 4-8 An example of making hand imprints using implicit functions.	65

Figure 4-9 An example of twisting by two hands.	66
Figure 4-10 The correction algorithm for finger penetration. (a) Correction of finger sinking. (b) Correction of finger passing-through.	67
Figure 4-11 Flowchart of the complete finger correction algorithm.	71
Figure 4-12 Routing coordinates of the index finger to the interactive java script in VRML and linking them to the position of the moving ball.	74
Figure 4-13 Pinching a piece of elastic clay. Fingerprints are created by moving two offset functions centered on C1 and C2	76
Figure 4-14 Code structure in VRML.....	79
Figure 4-15 Real-life and virtual tasks in the experiments. (a) putting a cap onto a container, (b) screwing a nut onto a bolt, (c) making elastic deformation with one hand, and (d) twisting deformation done by two hands.	80
Figure 4-16 Apparatus and experiment setup.	81
Figure 5-1 Recognition of the taking gesture.	88
Figure 5-2 The interaction technique for rotation around a fixed axis.	91
Figure 5-3 Scaling by two hands.	95
Figure 5-4 The experiment setup. Interaction space and objects. (a) The real-life interaction setup. (b) The virtual interaction setup	96
Figure 5-5 The 3D widget for hand interaction.	101
Figure 5-6 Constraining interaction techniques for (a) directional scaling, (b) uniform scaling, (c) twisting, (d) tapering, (e) rotation, and (f) translation.....	102
Figure 5-7 The modeling process.	102
Figure 5-8 Menu of the modeling prototype.....	103
Figure 5-9 The virtual modeling task.....	104
Figure 5-10 Selected works of participants.	106
Figure 6-1 (a) Playing the Theremin (Moog Etherwave Standard.) (b) Theremin-style gestures proposed for 3D modeling in mid-air.	111
Figure 6-2 The interaction space of two hands.	113

Figure 6-3 The interaction techniques for (a) twisting, (b) inflating, (c) dragging, and (d) bending.....	120
Figure 6-4 The interaction technique for scene manipulation.	128
Figure 6-5 The adaptive calibration algorithm.	131
Figure 6-6 Implementations of the Theremin-style interaction for natural interaction tasks studied in previous sections. (a) Manipulating the nut. (b) Rotating the bolt by rotating the scene. (c) and (d) are manipulating a pencil and a book.....	132
Figure 6-7 The shape modeling task in Autodesk Maya.	134
Figure 6-8 The setup for the user study for the (a) mid-air group and (b) Maya group. .	136
Figure 6-9 Average time spent on each step of the task.	138
Figure 6-10 Average number of operations on each step.	139
Figure 6-11 Results of the user questionnaire.....	141

LIST OF TABLES

Table 3-1 Comparison of tracking range and frames of frequency.	36
Table 3-2 Standard deviation of the measured positions and angles	37
Table 3-3 Average errors of the separation distance in the pinching gesture.	38
Table 4-1 Time spent on real-life and virtual interaction of each task.	82
Table 5-1 Comparison on efficiency of real-life and virtual interaction.	98
Table 5-2 Results of user questionnaire	99
Table 5-3. Subject evaluation results for the modeling task.	105

Chapter 1 Introduction

1.1 Background and motivation

We use our hands in many ways for various modeling and assembling tasks. For example, different ways of grasping are developed for taking objects with different sizes and shapes. To deform an object, we may poke, press or smooth it with one or several fingers or any other parts of a hand. Two hands are used for twisting, bending or squeezing, where motions of wrist, shoulder, or the whole arm are involved. Moreover, the second hand can be used either in a symmetric way, e.g., to hold large objects with two hands, or for auxiliary control, e.g., to constrain motion of the object. The developed hand interaction ways allow us to deal with most basic modeling tasks. However, when working on precision tasks two intrinsic problems of hand interaction exist: hand tremor and jump release. Hands tend to tremble when there is no physical support. Jump release causes objects to slightly displace when the fingers are being removed from them. These problems seriously affect the precision of object placement, especially for small components.

Historically, 3D modeling with computer uses 2D tracking devices, such as mouse and touchpads, which are based on tracking positions of one or a few points on a plane. Although such interaction can minimize the negative impact of tremor and jump release in the modeling process, the plethora of natural hand gestures was replaced with only a few basic 3D manipulations which are often rather unnatural and even inefficient when implemented using successive 2D tracking operations. For example, an arbitrary 3D rotation is split into several subsequent rotations around principal axes.

Progress in computer vision allows mid-air hand motion to be tracked by affordable optical sensors, such as Leap Motion controller, Intel RealSense SR300 (or called BlasterX Senz3D), and uSens Fingo. Breakthroughs were also made in predicting self-occluded hand, e.g., [1, 2], which, until recently, was a serious obstacle when using optical tracking devices. These advances of optical tracking have boosted many applications and research in different areas, such as mid-air keyboard typing [3], interaction with large displays [4], and biometric recognition [5]. Most of the applications are for VR games and entertainment, as it can be seen in the application store of Leap Motion controller [6].

Compared to the traditional 2D tracking devices, 3D hand motion can provide more degrees of freedom, as well as natural gestures for shape modelling. Interaction of shape modeling can be potentially done in a more natural and efficient way with optical sensors. On the other side, however, it still remains a challenge to use optical hand tracking for precision tasks, such as virtual assembling and modeling, since the precision of interaction is inevitably affected by the additionally introduced problems, such as hand tremor, jump release, and the occlusion problem of optical tracking. The thesis is therefore motivated to solve the observed problems and improve the interaction of existing shape modeling systems. Specifically, it is hypothesized that **shape modeling can be improved by using optical sensors, so that 3D tasks can be done in a way similar to interaction in real life and as efficiently as using common 2D-tracking based interaction, while the intrinsic problems of precise hand motion and optical tracking are minimized.**

1.2 Proposed approach

With reference to whether collision with virtual hands (driven by real hands) is involved, existing natural interaction techniques can be classified into two types: collision-based and gesture-based methods. Collision-based methods mimic natural interaction by implementing collisions between virtual hands and objects, while gesture-based methods control different interaction with a set of pre-defined gestures. In order to verify the hypothesis, experiments are to be conducted in both directions to compare real-life and virtual interactions. Specifically, the following research and development work has to be done.

1) To investigate whether collision-based methods can be used to achieve precise interaction required for shape modelling and virtual assembling. The expected research novelty here is in new algorithms and tracking methods which will allow to perform such collision-based interaction using optical tracking devices. The proposed methods have to be tested with interim user studies. This research is described in Chapter 4.

2) To investigate whether gesture-based methods can be used to achieve precise interaction required for shape modelling and virtual assembling. The expected research novelty here is in new hand-tracking algorithms which will allow to perform such gesture-based interaction using optical tracking devices. The proposed methods have to be tested with interim user studies. This research is described in Chapter 5.

3) To select the best performing technique and to undertake a final user study comparing the proposed mid-air hand interaction with mouse-based interaction while working on a typical shape modelling task and using one of the commonly used modeling tools. This research is described in Chapter 6.

1.3 Report organization

In the following parts of the thesis, related works on human hand motion capabilities and limitations, bottlenecks of 2D tracking devices, mid-air hand tracking technologies and interaction techniques are surveyed in Chapter 2. Existing available optical sensors for articulated hand tracking are reviewed, selected and tested in Chapter 3. Natural collision-based techniques are designed and studied in Chapter 4. Natural gesture-based techniques are designed and studied in Chapter 5. A special theremin-style interaction way on gesture-based interaction are studied in Chapter 6 which also continues with the user studies. The conclusion is made in Chapter 7, and the future work is outlined in Chapter 8.

Chapter 2 Shape Modeling – From Real-life to Digital

In this chapter, a survey is conducted on related works from real-life interaction ways to virtual interaction techniques for shape modeling. Human hand and finger motion capabilities and limitations when doing real-life modeling tasks are illustrated in Section 2.1. Precision, interaction techniques and problems of 2D tracking devices in shape modeling are discussed in Section 2.2. 3D hand tracking technologies are surveyed in Section 2.3. Existing natural interaction techniques are surveyed in Section 2.4, which is followed by a summary in Section 2.5. Further research plans for the thesis based on the survey are listed in Section 2.6.

2.1 Modeling fine shapes with human hands in real life

Figure 2-1 Hand joints and their degrees of freedom

Human hand motion has 27 Degrees of Freedom (DOFs) in total, of which local finger motion has 21 DOFs and global motion has 6 DOFs for the whole hand translation and rotation. With reference to local finger motion, each finger has 4 DOFs and the thumb motion has 5 DOFs, as illustrated in Figure 2-1.

Figure 2-2 Motion of hand joints: (a) and (b) are motion capabilities of fingers; (c), (d) and (e) are motion capabilities of the thumb.

A finger can move in two major ways – flexion and extension. We can flex our fingers to grab or hold objects or extend our fingers to release objects. The middle, ring and pinky have the same single muscle for flexion, so that the flexion of one finger will inevitably involve the motion of the others. The MCP joints allow fingers to extend towards the backside of the hand, which is called hyper-extension, as illustrated in Figure 2-2 (a). The other DOF of MCP joints allow fingers to abduct and adduct within the palm plane.

Abduction refers to the motion when fingers move away from the hand midline (ray of middle finger), while adduction refers to the motion in the opposite direction, as can be seen in Figure 2-2 (b). Due to the fact that the middle finger is already on the midline of hand, its motion, in no matter which direction, is called abduction.

The thumb is regarded as being responsible for half function of a hand. The joints of Thumb IP and Thumb MP allow a thumb to flex and extend just like fingers, as shown in Figure 2-2 (c). The TMCP joint allows unique circumduction movement of the thumb so that it can touch other fingertips, as shown in Figure 2-2 (d). The movements of abduction, adduction, and retropulsion of the thumb are described in Figure 2-2 (e).

These DOFs of hand motion allow us to dexterously use our hands in real-life interaction tasks. For example, 33 different grasp types, which consist of different combinations of individual finger postures, were found and classified in [7]. People can easily plan how to grasp objects of different sizes, weights and shapes. In addition, hand motor skills have also been developed with maturation of age and experience. Typically, hand motor skills can be classified into two groups: gross motor skills and fine motor skills (refer to, e.g., [8]). Gross motor skills involve coordinating large muscles of arm and shoulders to move hands in a large space while performing various gestures like waving, punching, swinging, etc. In contrast, fine motor skills involve small movements of wrist and fingers. They are used in such precise tasks as threading a needle, using instruments like scissors and tweezers, writing, etc. The two types of skills usually work together in the process of completing an interaction task.

The developed interaction skills, especially the fine motor skills, are crucial for completing precise modeling tasks, such as pottery, claying, sculpting, and mechanical

components assembly. To deform something, we may poke, press or smooth it by one or several fingers or other parts of a hand. We may use two hands for twisting, bending or squeezing, which involves motion of wrist, shoulder, or the whole arm. Moreover, the second hand can be used either in a symmetric way, e.g., to hold large objects with two hands, or for auxiliary control, e.g., to fix motion of the object.

Although our hands are capable of dealing with most of interaction in real life, they are still highly constrained so that they can only be used for a very specific set of precision modeling tasks. In various studies, (e.g., [9]) it was observed that precise in-hand manipulation (the ability to move objects by fingers in a hand) had the best performance only when the sizes of objects range from 48 mm to 59 mm. Due to the constraints imposed on finger joint motion, in-hand rotation capabilities are also quite limited depending on the orientation of rotation axis [9, 10]. Interaction becomes extremely difficult when the size of an object is smaller than 5 mm, e.g., when tightening a small nut on a screw. Although precision tools (such as a tweezers) can be used, *hand tremor* still makes it too difficult to do operations at such scale -- the amplitude varies between $0.4 \text{ mm} \pm 0.2 \text{ mm}$ for young individuals and $1.1 \text{ mm} \pm 0.6 \text{ mm}$ for old individuals [11, 12]. Hand tremor is also closely associated with *jump release*, which may happen when fingers are not simultaneously removed from the surface of the object being carefully placed. A little momentum created by the delayed finger may slightly displace the object. Furthermore, these problems become more serious due to arm fatigue after long-time operations.

It was observed that natural free hand motion can provide not only abundant degrees of freedom but also dexterous interaction ways for virtual shape modeling. However, direct transferring of our real-life hand interaction ways to virtual environments may not be

always feasible to achieve high efficiency of interactive shape modeling, even though hand tracking devices are quite precise and are able to capture sub-millimetre displacements of the fingers. On the other hand, the intrinsic problems of hand manipulations can be potentially reduced by using constraints and varying precision and coordinate mapping in virtual modeling environments similar to using tools and magnifying glasses in real life. Keeping in mind these limitations of hand motion, virtual interaction techniques that can minimize or totally avoid negative effects of these problems will be explored.

2.2 Shape modeling using 2D tracking devices

Digital shape modeling unites various techniques used for making virtual objects (including their geometry and visual appearance), defined by creating and manipulating points, curves, surfaces, as well as deformations and Boolean (Set-Theoretic) operations.

In many cases of interactive shape modeling, however, 3D modeling is performed through a series of 2D drawings and displacements since this can be conveniently done with 2D tracking devices such as mouse, trackpad, joystick, keyboard, etc. For example, freeform surface modeling deals with construction and editing surfaces which can be created, for example, by filling the profile curves, or by evolving or extruding curves along another plane curve. Sweeping and extrusion are created by moving curves, surfaces or solid primitives along 3D sweeping trajectories which can also be defined on a plane. Sketch-based modeling allows 2D contours to be drawn and then inflated to 3D shapes. Digital sculpting deforms shapes by displacing their individual points or parts along straight lines. Various global deformations, such as scaling, tapering, twisting and bending, are often simplified as 2D operations by using handles and side bars. When Constructive Solid

Geometry (CSG) builds shapes by combining them using Boolean operations, positioning of each shape with reference to another is also interactively decomposed to displacements along a straight line or plane rotations about principal axes.

Indeed, virtual modeling can be more precisely done with 2D-tracking devices than that done with bare hands in real-life interaction. Hand tremor is less serious with an arm resting on the desktop rather than held in the mid-air space. The problem of jump release can also be solved when releasing an object with fingers is replaced by releasing a mouse button. In addition, 2D-tracking devices have a high precision for tracking hand motion on the desktop. Tests showed that the actual precision of a common mouse sensor ADNS6010, with the stated precision of 12.7 μm (2000 dpi), was 0.144 mm (when moving 1.024mm) for static measuring and reached 0.187 mm for dynamic measuring [13]. The sub-millimetre accuracy allows a standard mouse to be used precisely for the purpose of shape modeling.

Although common 2D-tracking devices can be efficiently used for 2D tasks (e.g., sketching and manipulation on a plane) and some 3D tasks that can be mapped into 2D (e.g., 3D object selection with the ray casting method), they become inefficient when it comes to 3D operations due to limited motion space and degrees of freedom. The plethora of natural hand gestures has to be replaced with only a few basic 3D manipulations which are often unnatural and even inefficient when implemented using successive 2D-tracking operations. Therefore, special interaction techniques were developed for such tasks to be performed with 2D-tracking devices.

3D widget (also called 3D gizmo or manipulator) is a common method used by most shape modeling tools, e.g., in Autodesk Maya. Manipulation of an object is performed by interacting with translation, rotation and scale widgets embedded in the target 3D object.

Some special widgets were also designed for deformations, such as tapering, twisting, bending and a combination of them [14]. The widgets interactively control the parameters, including range, extent and direction of deformation. They can be efficiently and precisely used for 2DOF operations but become inefficient for arbitrary operations involving more degrees of freedom. For example, aligning two objects has to be done by performing a sequence of translations and rotations.

Aside from widgets, 3D tasks have also been implicitly mapped to 2D space using various intuitive techniques. Virtual trackball, Shoemake's Arcball and Two-arc Valuator methods allow 2DOF of 3D rotation to be controlled simultaneously by a mouse. The efficiency of these 3D rotation techniques were compared in [15]. A sliding method was proposed for 3D translation in [16], where the object being moved was always sliding on another one. Although interaction using 2D tracking devices can be more efficient than that using 3D tracking for some specific cases, e.g., [17], the insufficient DOFs always constrain the convenience of interaction.

Interaction using 2D-tracking devices can be potentially improved by using 3D hand tracking, which provides not only additional degrees of freedom but also natural interaction ways for virtual interaction. In order to explore this idea, existing hand tracking devices, as well as related natural virtual interaction techniques, are surveyed. It is supposed that in order to achieve the desired accuracy of modeling, a candidate tracking device should have at least sub-millimetre accuracy which is comparable to that of a common optical mouse.

2.3 3D hand motion tracking technology

As classified by [18], common motion tracking technologies fall into the following five categories – mechanical, inertial, acoustic, magnetic and optical tracking. By reviewing the existing hand tracking devices, optical tracking technology is dominantly used due to its high precision, low latency, unobstructiveness and affordability. However, the intrinsic occlusion problem limits its accuracy for highly-articulated finger motion. In comparison, glove-based system is free of occlusion by tracking bending angles of finger joints through sensors sewed inside a glove. However, the expensive price and cumbersome setup prevents such systems from being wide use in common daily interaction. In this section, existing hand tracking devices based on the two mainstream hand tracking technologies introduced above are surveyed, in order to understand whether existing technology for hand motion tracking allows for a sub-millimetre accuracy to be comparable to a common optical mouse.

2.3.1 Optical tracking technologies

Optical tracking systems consist of two components: light sources and optical sensors. The light sources can be either active ones which emits lights, or passive ones which reflect ambient lights. With regards to where to place the optical sensors, the tracking technologies can be classified into inside-out and outside-in tracking. In inside-out tracking the optical sensors are put inside the tracked device, while in outside-in tracking they are placed at stationary location.

For inside-out tracking, the optical sensor inside the device looks out to find light emitters or other reference objects to determine its own position and orientation. The technology has been used by some hand-held game controllers for tracking 6-DOF hand

motion in a room area. For example, the Wii Remote, as the primary controller for Nintendo's Wii console, has the infrared (IR) camera inside to sense the infrared lights emitted from edges of the console's Sensor Bar. The relative distance to the console's Sensor Bar is calculated through triangulation by Wii CPU [19]. However, since motion of the controller can only be tracked when the Sensor Bar is in its visibility range, it is used as an on-screen pointer rather than for tracking hand motion in all poses. By contrast, the HTC Vive tracking system minimizes occlusion by using two base stations which emit a sequence of infrared light pulses into the room. The location and orientation of a Vive controller can be tracked by decoding the information of the IR pulse received by the array of IR sensors inside the controller.

For outside-in tracking, the optical sensors are placed at stationary locations. Typical representatives of this method are Playstation Move and Oculus touch. Playstation is the motion game controller for Sony Playstation game console. The controller has an orb at head which can glow any color by RGB light-emitting diodes (LEDs) [20]. The light can be tracked by a Playstation Eye or Playstation camera, and the position of the controller can therefore be calculated according to the size of lights captured by the camera. The Oculus Touch is a hand-held controller for Oculus Rift. It contains a set of infrared LEDs which blinks in a specific pattern. The lights received by the IR sensors placed on desk are used to track motion of the controller by comparing it with the known pattern.

The HTC Vive controller and Oculus Touch are declared to have the best accuracy among all of existing consumer-level motion controllers. According to [21], the Vive tracker had a jitter level of 0.3 mm when a single base station was used, and an static accuracy about 2 mm when measured by a ruler. The submillimeter jitter level was also

confirmed by [22], which declared the standard deviation of the recorded position was always less than 0.5 mm. In a more professional setup, where Vive was measured by two laser rangefinders with accuracy of 1.6 mm, the results showed that the error of the Vive tracker was less than 1 cm on average and at most 1.5 cm when the two base stations were placed with distances of 6.3 m and 7.6 m [23]. In addition, the tracking accuracy of Oculus Rift and HTC Vive was compared by Adrian *et. al.* [24]. Their results showed that the Oculus Rift and Vive had quite similar accuracies of 7.6 ± 5.3 mm and 7.5 ± 6.9 mm, respectively, at static normal sitting positions (1.3m) within their recommended working areas of 2.75 m^2 for Oculus and 6.25 m^2 for Vive. Both of them had a submillimeter level of jitters which were 0.12 ± 0.18 mm for Oculus and 0.23 ± 0.33 mm for Vive. According to the observations, current best consumer-level motion controllers have a sub-millimeter level of jittering, which can reach the hand tremor level of a normal person. However, the accuracy for static position tracking is still not as high as sub-millimeter level of a standard mouse. In addition, holding a controller prevents further use of dexterous finger motion in virtual interaction, which has up to 21 DOFs.

The submillimeter level of accuracy can be achieved by a professional motion tracking system such as Vicon, OptiTrack and VisualEyes. These systems can track several reflective markers put on the target object simultaneously at a high refreshing rate. For example, the VICON system is measured to have an accuracy up to $76\text{ }\mu\text{m}$ and precision (noise) of $15\text{ }\mu\text{m}$ in a four camera configuration [25]. Precision of OptiTrack Prime 41 tracking system is up to 0.2 mm [26]. Although such systems have been successfully used for whole body motion tracking, they are still too vulnerable to occlusion when tracking the highly articulated hand and finger motion. In addition, they are also very expensive for

common daily use, e.g. it costs 5999 USD for one professional motion tracking camera Prime 41 [27].

Another group of outside-in tracking technology uses vision-based techniques for markerless hand tracking. Such technology uses affordable optical cameras, which can be IR cameras, color cameras or depth camera, to directly capture images of bare hand motion and use vision-based techniques to reconstruct the hand model. However, the occlusion problem is a bottleneck for such methods – it is challenging to recognize the configuration of the hand when most parts of a hand are occluded in the captured image. Algorithms for tracking highly-occluded finger motion are developing fast in recent years, thanks to the increasing computing power of GPU and progress in machine learning. For example, deep learning methods such as RDF [28], convolutional Neural Network (CNN) [2], as well as their combination [29] have been successfully used for tracking articulated hand and finger motion. Hand motion in serious occlusion cases, such as crossing two hands [1] and holding objects [2], can also be plausibly interpreted now. The precision of tracking was considered to be further improved by calibrating the size of the virtual hand model according to that of users' hands [30].

At the same time, many consumer-level short-range IR cameras specially designed for hand tracking are gradually becoming commercialized and popular. Typical devices are Leap Motion controller [31], uSens Fingo [32], and BlasterX Sens3D [33]. These short-range sensors are designed for desktop use and they have shown high precision in tracking small-scale hand motion. For example, the static accuracy of Leap Motion controller, when tracking the tip of a pen, is within 0.2 mm, and 2.5 mm, when the pen is moving [34]. Another test [35] showed that the accuracy of tracking of a static fingertip was less than

0.01 mm in an area of 30 cm above the device, while the overall accuracy was below 0.5 mm in its entire sensory area. However, the sub-millimeter accuracy is not stable when fingertips were too close. Tests in [36] showed that Leap Motion controller had error of 0.878 cm when the distance between the thumb tip and index fingertip was 1 cm. Only when the separation distance was larger than 5 cm, the error was smaller than 0.04 cm. Another work [35] reported that the device failed to track the constant distance (21.36 mm, $sd=0.023$ mm) between tips of two sticks – the error was beyond 0.8 cm.

2.3.2 Glove-based hand tracking systems

In comparison to optical systems, glove-based systems are free of occlusion by using gloves that are in physical contact with users' hands. A glove-based system is defined as 'a system composed of an array of sensors, electronics for data acquisition/processing and power supply, and support for the sensors that can be worn on the user's hand.' [37]. Typically, the major component of the system is a cloth glove made of elastic fabric, like lycra, where bend sensors are sewn inside at finger joints. The bend sensors track bending angle of the finger joints by measuring the amount of deflection caused to them. Different technologies are used for making such sensors, such as conductive ink, fiber-optic, conductive fabric, etc.

Figure 2-3 Glove-based systems. (a) CyberGlove. Figure is from website [38]. (b) HumanGlove. Figure is from website [39]. (c) 5DT Data Glove. Figure is from website [40]. (d) PinchGlove. Figure is from website [41].

Glove-based systems have a long history of development. The first glove-based system Sayre Glove was invented by Massachusetts Institute of Technology (MIT) in 1977 [37]. Commercialization of glove-based systems started in the end of 1980s. Typical representatives of early professional glove-based systems are CyberGlove [38], HumanGlove [39], PinchGlove [42], and 5DT Data Glove [40]. The price of such a professional glove-based system is always very expensive, e.g., CyberGlove II has a price of at least \$10k per glove. Entry-level gloves, such as VMG Lite [43] and Manus VR [44], have relatively lower prices of around \$500-\$800, but their tracking capabilities are also very limited – each glove only has less than 12 bending sensors inside.

When using glove-based systems, absolute positions of fingertips cannot be directly tracked, but can only be calculated according to the global hand position and bending extent of the fingers. Although using bend sensors can avoid occlusion, such technology has its own problems including variations of hand sizes, nonlinearity, cross-coupling and noise, which seriously affect the tracking precision. Since each glove has a fixed size, the sensors

sewn inside it cannot be placed exactly at finger joints to precisely measure the bending angles for different hand sizes. The cross-coupling problem is caused by inevitable stretching of neighbourhood sensors during finger motion. It was reported in [45] that the cross-coupling problem could cause error of more than 20° . Therefore, when tracking a pinching gesture using CyberGlove, the actual separation distance between the thumb tip and the index fingertip was measured as 7.53 cm from raw data, and 5.25 cm after linear calibration [46]. Since simple linear calibration methods cannot solve the problem of cross-coupling, efforts have also been made in the direction of using parabolic functions to synthesize training data [47, 48], considering it as an interpolation problem [49], or using additional vision systems to calibrate with a multiple linear regression method [50]. However, these methods can only adjust joint angles to get a better tracked hand posture, while fingertip positions cannot be tracked precisely. In comparison, machine learning methods showed a good performance for calibrating fingertip positions [46] in solving the problem of cross-coupling. However, the method is computational demanding, so that the frame refreshing rate is inevitably slowed down in applications.

Therefore, although glove-based tracking is a relatively old and mature technology, there are still many problems limiting motion of fingers to be precisely tracked. Other disadvantages of such systems are their complex and cumbersome setups and expensive prices. Hence, such systems may not be the best choice for using for shape modeling on a daily base.

2.4 Natural hand interaction algorithmically

With reference to whether collision with virtual hands (driven by real hands) is involved, natural interaction techniques can be classified into two types: collision-based and gesture-based methods. They are introduced respectively in the following parts of this section.

2.4.1 Collision-based interaction

Collision-based methods mimic natural interaction by implementing collisions between virtual hands and objects. The collision model of a virtual hand, controlled by the user's real hand, can be constructed as a skeleton model or a simplified metaphor. The interaction between virtual objects and virtual hands can be driven by a physics engine (a third-party software that provides an approximation of certain physical system such as rigid body dynamics with collision detection) or purely algorithmically.

Interaction driven by a physics engine

When using a physics engine, real-life interaction ways, such as pushing, hitting, and throwing, can be realistically simulated in real-time after collision models and rigid bodies are assigned to virtual hands and objects. However, it becomes problematic when trying to grab virtual objects – the objects will jump away due to some penalty algorithms implemented in the engine to prevent penetration. Additional algorithms thus are needed to be implemented to correct finger penetration and define frictions in the contact area. Borst et. al. [51] proposed a spring-damper model, which prevents penetration by loosely coupling a virtual hand with a real hand through a set of linear and torsional virtual spring-

dampers. In [52], motion of a virtual hand followed a set of constraints including non-penetration constraints, limited angle of joint motion and fidelity of tracked hand motion. The constraints were set priority and solved by a physics engine proposed in [53]. It was also proposed to use soft finger pads [54, 55] in the virtual hand model in order to properly simulate contact response when grasping objects with sharp edges.

Although the proposed methods can effectively prevent finger penetration visually, the offset between real and virtual fingers still exists when the object is being grasped. The mismatch between them introduces additional problems for virtual interaction. For example, releasing a virtual object requires additional motion of real fingers, just like the object to be released is “sticky”. Therefore, instead of solving existing problems of hand motion, such as hand tremor and jump release, these methods make the problems even worse in virtual interaction. In addition, physics-based simulation is computationally expensive. It has additional requirements on the size and type of a shape model. For example, in Unity game development platform with built-in physics engine (www.unity3d.com), a mesh collider can only be generated on a convex mesh which has less than 256 triangles. Therefore, such methods may not be always feasible to be implemented in shape modeling applications, where the size of a mesh can easily grow beyond the maximal limits of the physics engine.

Interaction with explicit methods

An alternative way is to use pseudo-physical methods, which define motion of objects explicitly rather than by simulating physical responses. For example, the heuristic grasping method is commonly used to mimic taking virtual objects with hands. The method assumes

an object is being firmly grasped by a hand so that its relative transformation to the hand is fixed during manipulation. The motion starts when the defined grasping condition is satisfied, e.g., the object collides with the thumb and multiple fingers simultaneously [56].

Finger-based manipulation is commonly implemented by putting a set of point sensors on fingers and by defining the object motion through the motion of sensors. For example, the representative spherical plane method (RSPM) [57] was proposed to simulate object manipulation with three fingers. In the method, three sensors are placed on three fingertips forming a “FingerTip Triangle” (FTT), and the object motion is defined by the deformation of FTT. The method was further extended to simulate manipulation with two fingers by adding a third point on the root between the two fingers. Ronan et al. [58] proposed to use friction cones to simulate forces in the contact area between fingers and objects. When a finger collides with a virtual object, whether it exerts force or slides on the surface of the object is decided by the simulated Coulomb law. Another sensor-based grasping method [59] drives interaction through grasp pairs rather than single sensors. Valid grasp pairs, which exert forces in opposite directions, are extracted from a sequence of frames and used to define the object motion.

Collision-based methods were also implemented for interactive mesh deformation. For example, clay deformation by fingers was simulated by warping the space around a mesh using Compactly Supported Radial Basis Functions, or CSRBF, centered on fingertips [60]. Another work [61] proposed to use the whole hand to deform a point cloud, from which a smooth surface is generated. With application to product design [62], the material’s elastic property was simulated by constructing the deformable shape using a volume-like mass-

spring-damping model. Deformation of the elastic shape was calculated from the estimated amount of force exerted by a virtual hand.

2.4.2 Gesture-based interaction

Instead of computing collision, gesture-based methods control interaction with a set of pre-defined gestures. Based on the work of Quek et al. [63] on gesture taxonomy for human computer interaction, Maria Karam and M.C. Schraefel [64] classified gestures into five types: deictic, manipulations, semaphores, gesticulation, and language gestures. Deictic gestures involve pointing to the target for identification in the context. Manipulative gestures are the ones used to relocate or deform an object by relating its movement\deformation with hand or arm movements. Semaphoric gesture is defined as “any gesturing system that employs a stylized dictionary of static or dynamic hand or arm gestures” [64]. They serve as symbols to activate different functionalities of the system. Gesticulation refers to gestures that comes with conversations, e.g., to depict the shape of an object or to convey the emotion. Sign language gestures are linguistically linked with words which are arranged according to grammar structures to convey sentences. Among the five kinds of gestures, manipulative gestures are most related to the research and they will be further surveyed. Deictic and Semaphoric gestures are also necessary for controlling functionalities of a modeling system, but they are not the major focus of the thesis.

Gesture tracking technologies

Hand gestures can be captured through physical interaction with the interfacing device, like data glove, accelerometers, multi-touch screen, game controllers and etc. A more

affordable and common method is to use camera-based systems combined with vision-based techniques to track and recognize mid-air gestures. The vision-based techniques can be further classified into two categories – appearance-based and model-based methods [65].

Appearance-based methods recognize gestures by matching the hand appearance in the image with a set of predefined template gestures. The gesture templates can be generated from image sequences, e.g., history images or MHIs [66], or features derived from the images, e.g., contours and edges [67], hand orientation histograms [68], and fingertip positions [69].

Model-based methods establish a model of hand, e.g., volumetric model, geometric primitives, or skeletal model, and update parameters of the model according to the captured images of hand motion. The gestures are further defined by the extracting geometric information from the model. For example, the pinch gesture can be defined by measuring the distance between the thumb tip and index fingertip when the skeleton hand model is available. Model-based methods have caught much research attention in recent years. For example, Zhou Ren et al. proposed a novel distance metric call Finger-Earth Mover's Distance (FEMD) to recognize finger gestures of similar configurations using Kinect sensor [70, 71]. Toby Sharp et al. proposed a real-time tracking system based on a depth camera, which can accurately reconstruct complex hand poses across a variety of subjects [72]. Andrea Tagliasacchi et al. proposed real-time registration process that accurately reconstructs hand poses by fitting a 3D articulated hand model to depth images [73]. Anastasia Tkach et al. proposed a method which used sphere-meshes as a geometric representation for real-time hand tracking [74]. Srinath Sridhar et al. proposed a 3D articulated Gaussian mixture alignment strategy for tracking of hand-object interaction

using a single commodity RGB-D camera [75]. Ge Liuhao et. al. proposed a 3D CNN-based method to accurately regress full 3D hand pose in a single pass using the hand depth image as input [75].

Interaction techniques

With reference to whether one or two hands are used, common manipulative gestures can be classified into unimanual gesture and bimanual gesture. For unimanual gestures, the most frequently used ones are the pointing gesture and the pinching gesture. The pointing gesture, as a kind of deictic gesture, has been commonly used for object selection in virtual interaction interface. It was implemented by casting a ray from the hand to collide with object being pointed [76, 77]. Mimicking how we take objects with thumb and fingers, the pinching gesture is naturally used for taking and manipulating virtual objects [78]. The gestures can be recognized by measuring the separation distance between thumb tip and index fingertip from the skeleton hand model or from the 2D captured images [79].

Bimanual gestures can be further classified into symmetric ones and asymmetric ones. Bimanual symmetric gestures use two hands as the same role, such as skipping and weightlifting. Bimanual asymmetric gestures are those which use both hands in an asymmetric way, e.g. gestures used for playing a stringed musical instrument. Based on observations on how we use our hands in daily activities, Guiad proposed the “kinematic chain model” [80] which have been widely used as guidelines for designing asymmetric bimanual interaction with computer 1) The left hand manipulates the spatial reference, and right hand performs actions on the reference frame; 2) Right hand has a higher movement frequency and larger movement amplitude than the left hand. The right hand can produce

finer movement than the left; 3) Left hand proceeds in an action. Several works were proposed to apply Guiard's framework to devise bimanual applications. For example, the Toolglass system implemented bimanual interaction where one hand controls the Toolglass and the other hand manipulates a cursor for interaction [81]. Daniel P. Mapes et al. proposed a two-handed direction manipulation interface which allows users to translate, rotate, scale, align and glue objects in the virtual environment [82]. Cutler et al. [83] built a virtual interaction system on the Responsive Workbench, where five bimanual symmetric gestures were designed, which included symmetric scale, slide-and-turn, turntable, grab-and-carry, and grab-and twirl. The symmetric scale operation was further used in [82, 84]. Based on Guiard's principle, a bimanual interaction interface was designed for neurosurgical planning [85, 86]. In their system, the bimanual virtual cutting operation was mapped from real hand movements when physical objects were held in hands as tactile cues. In addition, bimanual deformation techniques for space wrapping was also proposed in [87].

The works illustrated above were proposed based on interaction with a physical interfacing device, which could be data glove, accelerometers, multi-touch screen, Responsive Workbench and etc. Bimanual interaction techniques with bare hand interaction based on vision-based tracking systems were also proposed for virtual modeling. For example, a handle bar metaphor [88] was proposed for manipulating one or multiple objects, which is performed by moving two hands in 3D space just like manipulating a handle bar. A "sheet-of-paper" method was proposed in [89] in a virtual assembly system for manipulating the scene in a way similar to pinching a sheet of paper. Grasping and releasing an object was implemented by moving two hands closer or apart [90]. The interaction technique was designed for optical devices that cannot track any gestures

robustly. With reference to shape modeling applications, Murugappan et al. [91] proposed a system, where the profile of generalized cylinders was specified by moving two hands in a large space in the room. De Araújo et al [92] developed a modeling system which allowed virtual mockups to be created through gestures in a semi-immersive environment – 2D shapes were drawn on a multi-touch surface and then extruded above the surface by hand tracking with Microsoft Kinect. Kang et al. [93] developed a gestural interface and used it in a CAD solid modeling system for rapid creation of 3D conceptual shapes. However, it must be noted that the interaction techniques discussed above were designed with either Microsoft Kinect or multi-camera-based systems, which required hands to move in a large space but neglect dexterous finger motion.

Gesture-based interaction involving small-scale hand and finger motion has been implemented with short-range optical sensors for hand tracking, as it can be found in the official app store [6] of Leap Motion controller. For example, an application named “sculpting” [94] uses two spherical brushes, controlled by two index fingertips, to edit subtle features of a simulated clay. The applications Pinch Draw [95] and Pinch Move [96] implemented mid-air hand writing and precise manipulation of a cube with the pinching gesture. Waving, pointing and opening gestures were jointly used to control menu browsing, selection and manipulation respectively in an application called Form and Function 3D [97], which was designed as an anatomy educational tool.

2.5 Summary

In this chapter, surveys were conducted on human hand motion capabilities, interaction techniques with 2D tracking devices, 3D hand motion tracking technologies and natural

interaction techniques. It was observed that existing interaction of shape modeling implemented with 2D tracking devices can be potentially improved by using 3D hand motion, which provides more degrees of freedom and natural gestures for modeling. By comparison to the other 3D hand tracking technologies, optical sensors are more affordable, popular and convenient for shape modelling purpose on a daily base. The consumer-level optical sensors are selected for the research due to their sub-millimetre precision, affordable prices, convenience, and non-obtrusive ways of tracking. However, using such systems for precision tasks still remains challenging, because the precision of virtual interaction is constrained by both the intrinsic hand motion problems, such as tremor and jump release, and the occlusion problem of optical tracking. Using glove-based systems instead to avoid the occlusion problem is not a feasible solution, since such systems have their intrinsic problems of nonlinearity, cross-coupling and noise, which seriously affect the precision. Therefore, the problems are considered to be solved algorithmically by proposing novel interaction techniques.

With respect to interaction techniques, both collision-based and gesture-based methods have their own advantages and intrinsic problems for implementing efficient and natural interaction. However, there is no apparent evidence on which method is more promising in achieving the objective of the thesis. Therefore, both directions will be explored with the goal to solve the observed problems and finally allow optical hand tracking to be reliably used for shape modeling.

2.6 Further research plans

The next step is to further evaluate existing optical sensors in aspects of their tracking range, refreshing rate, and accuracy on different gestures. The test results will be used as references and foundations for further research in the thesis.

Based on the capabilities and limitations of the optical sensors, a set of interaction techniques will be designed accordingly and tested through user studies. In order to compare virtual interaction with real-life interaction, a group of participants will be invited to test both virtual and real-life interaction on a set of same tasks. The simulated virtual environment will be constructed as similar as the real-life scenario and the virtual tasks are implemented by mimicking real-life interaction tasks. The efficiency of interaction will be evaluated by comparing the time spent on virtual and real-life interaction. In addition, it is also of major concern whether the proposed interaction techniques can be comfortably and precisely used by different users.

The final task is to compare the two 3D modeling interfaces, which are based on 3D optical tracking and common 2D-tracking devices. They will be compared on a common 3D modeling task, which involves several basic manipulation and deformation operations. A group of participants will be invited to test the task on these two conditions. The two kinds of interfaces will be evaluated not only on efficiency, but also on aspects of learnability, mental load, comfort, controllability and precision. It is expected that 3D interaction with optical tracking can be significantly faster than 2D-tracking based interaction, while still better than or as good as 2D-tracking based interface in the other aspects. The efficiency of interaction will be quantitatively measured by the time spent on

completing the task, while the other aspects of learnability, mental load, comfortableness, controllability and precision will be evaluated through user questionnaires.

Chapter 3 Optical Sensors Selection and Test

Before designing any actual interaction techniques, both capabilities and limitations of existing optical hand tracking technologies should be understood. In this chapter, optical sensors for hand tracking are surveyed and tested, based on which the one with the best precision of tracking is selected. It is also expected that the statistics results of tests on the devices can be used as references and foundations for further research in the thesis.

3.1 Overview of optical sensors for hand tracking

Although there are many kinds of optical sensors for hand tracking in the market, only a few of them can track both hand and finger motion in 3D space – they are Leap Motion controller, Intel RealSense SR300 (or called BlasterX Senz3D), and uSens Fingo. All of them are close-range optical sensors with affordable prices ranging from \$79.9 to \$149. Microsoft Kinect for Xbox One was also used for the purpose of hand and finger tracking [72, 98]. However, such appealing hand tracking capabilities are not available from the official SDK. In addition, Microsoft Kinect as a product line was discontinued in October 2017. Therefore, only Leap Motion controller, Intel RealSense SR300 and uSens Fingo are selected as the candidate devices for the test. Their technologies, hardware and software are briefly introduced.

Published in July of 2013 by a startup company Leap Motion, Inc, Leap Motion controller is the pioneering sensor that commercializes optical technologies of articulated hand tracking. The sensor is a small device designed to be placed on the top of a desk and track hand motion above it. The hardware consists of two monochromatic infrared (IR) cameras and three IR LEDs. The LEDs cast infrared light with wavelength of 850

nanometers [99] (invisible to human eyes) to objects in front of it, which are further tracked by the infrared cameras. Limited by the propagation range of the LED light, tracking range of the device is within 60 cm in front of it. The cameras capture grayscale stereo images and send them to the software for analysis. The initial version of SDK allowed the static tip positions to be tracked with an accuracy of 0.2 mm [34]. However, the tracking was very vulnerable to finger occlusion at that time. After several updates since its first publication, the current SDK Orion Beta can track fingers even when they are slightly occluded. The current Orion Beta SDK only supports Windows, while the previous V2 version supports Mac and Linux. SDKs are provided in Unity, C++, C#, Objective-C, Java, Python, JavaScript, and Unreal Engine.

As a successor of Intel® RealSense™ camera F200, Intel RealSense SR300 (or called BlasterX Senz3D) is a 3D close-range depth camera published by Intel Cooperation in May of 2016. The device consists of an IR camera, a color camera, and an IR Laser Projector. The depth sensor has indoor tracking range of 0.2-1.5m. The video camera can provide 1080p full HD video image quality at up to 30 frames per second (FPS), or 720p HD video image quality at up to 60 FPS [100]. The device can only be used with Microsoft Windows 10 (64-bit) or Higher. It uses the Intel RealSense SDK, which supports Microsoft Visual Studio and Unity.

uSens Fingo is a recent hand tracking camera published in the beginning of 2017 by a Silicon Valley startup uSens founded in 2013. Different from other technologies, the device uses a single camera module with an embedded dual-lens (stereo) camera [101]. It is designed as a VR product to work with headsets, including Samsung Gear VR, HTC Vive, Oculus Rift, Google Daydream and Google Cardboard. It can work on both desktops (with

Windows 7+) and cellphones (with Android 4.4+). The current SDK only supports Unity, while the versions for C++ and Java are declared to come soon.

3.2 Testing of the selected sensors

3.2.1 Experiment Setup

Apparatus includes a desktop computer, a ruler and the three optical sensors -- Leap Motion controller, uSens Fingo and Intel RealSense SR300. The desktop computer is Alienware Auroro R5 with the Intel® Core™ i7-6700 Processor (4-Cores, 8MB Cache, Turbo Boost 2.0, up to 4.0GHz) and 32GB (4X8GB) DDR4 2133MHz SDRAM Memory. The Graphic card is NVIDIA® GeForce® GTX 1080 with 8GB GDDR5X. The monitor is 24-inches with a resolution of 1920*1200.

The three devices have different setups. Intel RealSense SR300 was put on the top of the monitor while facing the users (Figure 3-1 (a)). uSens Fingo was fixed in the mid-air with additional tools while facing to the monitor in order to track hand motion in front of it (Figure 3-1 (b)). Leap Motion controller has two tracking modes – desktop mode and VR mode. Both of the two modes were tested in the experiment. For the desktop mode, Leap Motion controller was placed on the top of a desk to track hand motion above it (Figure 3-1 (c)). It was placed in the same way as uSens Fingo for the VR mode (Figure 3-1 (d)). All of the devices were connected to the desktop computer through USB 3.0.

The most recent SDKs of the three devices were installed, which were Intel(R) RealSense (TM) SDK Runtime version v6 for Intel RealSense SR300, Leap Motion Orion 3.2.1 for Leap Motion controller, and Fingo SDK 1.2.4 for uSens Fingo. Hand tracking data was obtained with their sample codes.

Figure 3-1 Setups of sensors. (a) Intel RealSense SR300. (b) uSens Fingo. (c) Desktop mode of Leap Motion controller. (d) VR mode of Leap Motion controller.

3.2.2 Tasks and procedures

Three experiments were designed to evaluate tracking range, frames frequency (fps) and accuracy of the three sensors. In the first experiment, tracking range of the sensors was measured by moving hands over the devices until they reached the borders where fingers could no longer be accurately tracked. Frames frequencies (fps) were obtained from their default visualizers when motion of one or two hands were being tracked.

The second experiment tested consistency of the sensors in two cases. In the first case, palm was facing to a device while all fingers were extended to be clearly visible (Gesture I), as shown in Figure 3-2 (a). In the second case, the same gesture as Gesture I was

performed but the palm was vertical with reference to the device so that most parts of the hand were occluded (Gesture II), as shown in Figure 3-2(b). In both cases, the hand was kept still at the best tracking position for 20 seconds while the elbow was rested on the desk. The tracked positions of the palm and the tips of the thumb and the middle finger, as well as the palm normal direction were recorded every other 2 seconds.

The third experiment evaluated the accuracy of pinching virtual objects. The movement of pinching was performed by moving the thumb tip towards the index fingertip until they were connected. Two variations of this gesture were tested. In both cases the thumb and the index finger moved in the same way, while the other three fingers were extended in the first case (Gesture III, Figure 3-2 (c)) and naturally bended in the second case (Gesture IV, Figure 3-2 (d)). The second case was more difficult to be precisely tracked due to finger occlusion. The tracked separation distance between the thumb tip and the index fingertip was measured (it should be ideally zero) when the actual fingers were connected. This process was iterated for 10 times.

Figure 3-2 Hand gestures for testing. (a)-(d) correspond to Gesture I–IV. Intel RealSense SR300 is used to illustrate the gestures. Hand orientation was changed according to other setups of sensors.

3.2.3 Results

Visibility range and refreshing rate

The field of view, depth of tracking and frames frequencies of the three devices are shown in Table 3-1. The field of view data of the three sensors is from the official website of Usens Fingo [102], Intel RealSense SR300 [103], and Leap Motion controller [104]. Data of depth and fps were measured in the experiment.

Table 3-1 Comparison of tracking range and frames of frequency.

	Field of view (degree)		Depth (meter)	Frames frequencies (fps)
	Horizontal	Vertical		
Leap Motion controller	150	NA	0.25-0.75	108-110
uSens Fingo	115	90	0.05-0.4	65-70
Intel RealSense SR300	72.5±2	60±2	0.3-0.6	53-54

According to the results, the three devices have common length of depth range which is around 0.3-0.5 m. This tracking range allows hand motion to be tracked in a confined area above the desk. This area is extended when using VR setups of Leap Motion controller and uSens Fingo, since the sensors can move along with head motion in this mode. By comparison, Leap Motion controller has the largest tracking range and highest frames frequency among the three sensors.

Accuracy of tracking

As it can be seen in Table 3-2, Leap Motion controller has the overall better precision than the other two sensors. It can track palm position with sub-millimeter accuracy (0.8 mm) when the palm is wide open (Posture I). The actual precision of the device is higher than the measured results. It was used to track hand tremor [5], which is $0.4 \text{ mm} \pm 0.2 \text{ mm}$ for young individuals [11, 12]. Although it is not as precise when tracking the occluded hand (Posture II), the overall precision of tracking palm position is less than 2 mm. However, fingertips cannot be tracked as precisely as the palm. The accuracy is less than 4

mm when the hand is static and open. The results also show that the accuracy of Leap Motion controller is similar in desktop mode and VR mode.

Table 3-2 Standard deviation of the measured positions and angles

		Leap Motion (Desktop mode)	Leap Motion (VR mode)	uSens Fingo	RealSense SR300
Gesture I	Palm position (mm)	0.8	0.8	4.6	15.9
	Palm normal (deg.)	0.5	0.9	2.5	6.2
	Thumb tip (mm)	1.6	1.3	4.6	18.0
	Middle tip (mm)	1.4	1.3	4.5	25.4
Gesture II	Palm position (mm)	1.9	1.3	13.0	34.4
	Palm normal (deg.)	0.6	7.6	4.6	15.0
	Thumb tip (mm)	3.2	2.2	14.1	48.2
	Middle tip (mm)	3.4	2.1	9.1	35.5

By comparison, uSens Fingo and Intel RealSense SR300 are less accurate. Different from Leap Motion controller, uSens Fingo can track palm position and fingertip positions with similar accuracies, which are less than 5mm for the best case (Posture I) and 15mm for the occlusion case (Posture II). The performance of Intel RealSense SR300 is very unstable. Therefore, it will not be considered as the priority device for the research.

As it can be seen in Table 3-3, all of the devices cannot track fingertip positions as precisely in the pinching gesture as when the hand is open. By comparison, Leap Motion controller has the best precision around 7 mm in the desktop mode. However, the worst precision of the device is not limited to 20 mm. Therefore, it can be concluded from the results that virtual objects with sizes below 20 mm cannot be taken by the virtual hands. For safe manipulation, the sizes of virtual objects have to be set such that they could be

grasped with the distance of at least 30 mm between the thumb and fingers. This accords with the previous tests surveyed in Section 2.3.2.

Table 3-3 Average errors of the separation distance in the pinching gesture.

		Leap Motion (Desktop mode)	Leap Motion (VR mode)	uSens Fingo	RealSense SR300
Gesture III	Average error (mm)	6.9±0.8	7.7±1.0	46.3±0.9	67.9±3.9
Gesture IV	Average error (mm)	19.3±2.7	8.6±1.1	30.8±1.2	20.3±6.7

Further testing of Leap Motion controller

Further testing of Leap Motion controller showed that the best tracking range of the device, where hand and finger motion can be reliably tracked, was around 40 cm width and 40 cm depth horizontally, with a reliable vertical tracking area starting from 10 cm above the device to the height more than the maximum reach of hands when sitting in front of the desk. One hand can slightly overlap the other without loss of tracking. Palm position and orientation could be more reliably tracked than finger motion, so that they should be preferred over using finger tracking as input for 3D modeling. Finger motion can only be tracked when they are not occluded.

3.3 Summary and designing guidelines

Results from the test show that Leap Motion controller has the best performance in all of the aspects of tracking range, frames frequency and accuracy among the three devices. In addition, the current Intel RealSense Software Development Kit (v.2) supported by Creative Senz3D does not support hand tracking anymore. Therefore, Leap motion

controller was selected as a major testing device for the research, while the other devices may also be alternatively used and their tracking capabilities are kept in mind.

According to the results of the experiment, the following guidelines are summarized for further interaction design:

- Since the palm position can be more precisely tracked than fingertips, it should be preferred over fingertips to be used as input for 3D modeling.
- The best quality of tracking is when the hand is wide open while the palm is facing to the device.
- The size of a virtual object should be at least larger than 30 mm in order to be safely taken.
- Occlusions should be minimized since they seriously affect the accuracy of all optical sensors.

Chapter 4 Natural Collision-based Interaction

This chapter investigates whether collision-based interaction with optical hand tracking by Leap Motion controller can be used as naturally and efficiently as real-life interaction. Common interaction operations are listed in Section 4.1 and how they can be implemented virtually are illustrated in Section 4.2 – 4.6 respectively. The virtual interaction in an environment, when a physics engine is available, is tested in Section 4.7. Interaction techniques based on the situation when a physics engine is not available are designed in 4.8. The proposed interaction techniques are implemented in Section 4.9 and a user study is conducted in Section 4.10. The contributions of this chapter are finally summarized in Section 4.11.

4.1 Operations to be studied

The objective of this chapter is to study whether collision-based interaction with optical hand tracking by Leap Motion controller can be used as efficiently as real-life interaction when doing precise assembling and deforming tasks within the reach of hands. The interaction has to be done with hands over the table within the best tracking area with a width of 40 cm and a height from 10 cm to 40 cm right above the device. The sizes of virtual objects have to be such that they can be taken by hand with the distance of at least 30 mm between the thumb and fingers (as tested in Section 3.2.3).

The basic operations to be tested are taking, positioning, releasing and deformation. Taking (also known as pinching, seizing, grasping, gripping or holding) refers to picking up the object so that it moves along with hand motion. Different hand postures (with up to 33 types [7]) can be used in this operation depending on the object sizes, shapes and interaction

intent. Positioning refers to changing the position and orientation of the object. The adjustment can be done by rotating the wrist, moving the arm or colliding with the object by any parts of the hand without taking it up (e.g., pushing, turning, or poking). The second hand can also be used in the process to extend the interaction capability of the first hand. Releasing is the inverse operation of taking – it refers to removing palm or fingers from the object so that it is no longer in contact with the hand. Deformation refers to changing the shape of elastic objects by pressing with fingers or palm as well as twisting, bending or stretching by one or two hands.

4.2 Collision between virtual hands and objects

The first step of simulating natural collision-based interaction is to assign collision models to virtual hands and objects. The efficiency and performance of interaction depend on what collision models are used. For example, the collision model of a virtual hand can be constructed with a set of point colliders attached to fingertips or simple primitives (e.g., cylinders or boxes), or using volumetric models. When complex collision models are used, space partitioning algorithms (e.g., bounding volume hierarchy) are necessary for accelerating the computation of collision.

Collision-based interaction can be realistically simulated in real-time by many physics engines, which are mostly used for real-time physics simulation in games. Popular game-oriented engines are NVIDIA PhysX [105], Bullet Physics [106], and ODE [107]. These physics engines come with collision engines that can track collision between objects including simple primitives such as box, sphere, cylinder and capsule, or a convex mesh which meets certain conditions (e.g., for Unity, the mesh should have less than 256

vertices). Collision with a concave mesh can also be tracked by the engines but the mesh has to be static. A concave mesh can move in response to collision with other objects only when it is broken into separate convex parts and joined together as a group, or modeled by primitive colliders or a low-resolution convex mesh collider. In order to make the game objects behave realistically under the control of the physics engine, rigid bodies need to be assigned to them so that they can receive forces and torques from collision or external sources [108].

The ways how the operations of taking, positioning, releasing and deformation can be implemented with collision-based techniques are respectively investigated in Sections 4.3-4.6. In each section, it is considered how these operations can be implemented by common ways under the condition without a physics engine, as well as whether the operations can be more efficiently and precisely performed under the best condition when a physics engine is available. This chapter was partially published in [109].

4.3 Taking

The operation of “taking” refers to establishing firm contact between the hand and the object, so that the object can follow hand motion rather than slipping away. This can be done by closing fingers to the thumb while clamping the object in-between, wrapping fingers around the object when it has a cylindrical shape, or pinching by thumb and index finger when the object is thin or small.

In virtual interaction, whether an object is firmly taken by a hand can be decided by two methods – the explicit method and implicit method. The explicit method defines some conditions explicitly to tell the system when the event of taking is triggered. For example,

mimicking the common way of taking by pinching the object with thumb and fingers, the condition of virtual taking can be defined as when the object collides with tips of thumb and fingers simultaneously. The explicit method can plausibly recognize the specific gestures defined by the conditions, but fails to recognize other gestures which are beyond the conditions.

The implicit method purely relies on the simulated physics to drive the virtual interaction rather than defining specific conditions. A virtual object can be taken when enough friction forces are generated from the contact areas between hand and virtual objects to resist the gravity in the environment. In comparison to the explicit method, this method can simulate taking in a way more similar to that done in real life but it requires complex implementation and heavy computation.

The explicit methods have been commonly used in both conditions with or without a physics engine. Even on the condition when a physics engine is available, implicit ways of taking cannot be immediately used due to the problem that the virtual object will bounce away while the fingers are being closed to take it. This is because interpenetration between virtual fingers and objects is not allowed and the engine will generate forces to push them apart. However, finger penetration is inevitable when the virtual hand is coupled with the real hand motion which is without any constraints. Therefore, additional algorithms have to be implemented to correct finger penetration and to define the friction between virtual hands and objects. This results in large amount of implementation works to be done (as discussed in Section 2.4.1) since there are not any existing development toolkits that can be found for such purposes. Nevertheless, a physics engine still provides advantages in tracking collision

between virtual hands and objects more efficiently, since relevant algorithms are optimized within the engine architecture.

The operation of taking is further tested in Section 4.7.1 in a physics-driven environment. In Section 4.8.1, an explicit taking method is designed for the condition when a physics engine is not available. The method is well adapted to the hand tracking capability of Leap Motion controller.

4.4 Positioning

Positioning refers to adjusting the location and orientation of the object to the desired posture. It can be done by using fingers, wrist or the arm after the object is taken by a hand, or through none-prehensile interaction ways (e.g., by pushing and turning) driven by collision with any parts of a hand without taking it up. The second hand can also be used in the process to extend the interaction capability of the first hand, e.g., to stabilize motion of the object or increase the rotation angle.

In the virtual interaction, the operation of positioning can also be implemented with the explicit methods and implicit methods. The first kind of methods explicitly defines object motion according to geometric data extracted from hand and finger motion. Transformation of an object after it is taken by a hand, is commonly simulated [110] by making the object a child node in the scene graph hierarchy, so that its transformation is updated in every frame by inheriting that of the hand. This method assumes firm contact between hand and object while local motions of fingers are neglected. In-hand manipulation (done with fingers when the object is taken) and none-prehensile ways, driven by collision with any parts of the hand, require collision sensors to be put on the corresponding parts of

a hand. Transformation of the object should be defined by the motions of one or multiple sensors in contact with the object, e.g., [58].

The implicit methods allow object motion to be driven by various physical variables, including the torque and momentum generated from the collision area between hand and the object, gravity, friction, as well as the instantaneous velocity of the object. Such methods rely on a physics engine to calculate motions of objects in response to the simulated physics. Non-prehensile interaction ways (such as pushing, turning, or hitting) which are driven by collision with any parts of a hand can be directly simulated by a physics engine after colliders and rigid bodies are assigned to virtual objects. Motions of objects in response to collision with hands can be controlled by adjusting object mass, gravity, as well as parameters that control collision behaviours (e.g., contact offset and rest offset [111]). However, interaction ways after the object is taken by a hand requires additional algorithms to be implemented to correct hand penetration and define friction between virtual hands and objects, e.g., [51, 53].

The common positioning ways (provided by the Leap Motion interaction engine) in a physics-based environment are further tested in Section 4.7.2. Both taking-based and non-prehensile positioning ways based on the condition without a physics engine are proposed in Section 4.8.2.

4.5 Releasing

Releasing is the inverse operation of taking – it refers to removing palm or fingers from the object so that it is no longer in contact with the hand.

Releasing in virtual interaction can also be implemented by explicit and implicit ways corresponding to how taking is implemented. Specifically, for the explicit ways of taking, releasing is triggered when the conditions of taking are no longer satisfied, while in implicit ways the object is released when the generated frictions from the contact areas can no longer resist to the gravity of the object. The precision of releasing is affected by the problem of jump release which happens when fingers are not simultaneously removed from the surface of the object while it is being released. A little momentum created by the delayed finger motion may slightly displace the object. The problem is further affected by arm fatigue which is resulted from long time operations.

Although the observed problems for releasing still cannot be solved in principle even when a physics engine is used, the precision could be increased in certain circumstances when the motion of the object is constrained by the physical environment. For example, when placing an object (with positive gravity value) on another object or on the ground, the location and orientation of the object can be automatically adjusted until it reaches the static equilibrium state in contact with the surface.

Releasing an object in a physics environment is further tested in Section 4.7.3, while the releasing algorithm for the condition without a physics engine is proposed in Section 4.8.3 with concentration on improving the precision and minimizing the impact of hand tremor and jump release.

4.6 Deformation

In real-life interaction, a piece of elastic material can be deformed locally by pinching or pressing it with individual fingers, resulting in fingerprints left on it. The depth of

fingerprints can be adjusted by controlling the forces exerted on the contact area. Large-scale global deformation can be made when using two hands, e.g., by rolling, stretching, pressing or twisting.

Virtual deformations done with hand tracking can be implemented on shapes with different representations. When a polygon mesh model is used, fingerprints can be created on the model by displacing the vertices on the contact area defined by field functions (e.g., Compactly Supported Radial Basis Functions [112]) centered on the fingertips. When the shape is defined by implicit functions, local fingerprints can be created by subtracting implicit functions defined for the fingers by Boolean operations. Global deformations (such as bending, twisting, tilting, etc.) of these models can be defined by non-linear functions [113]. Cellular automata methods (e.g., [114]) or volumetric models (e.g., [115]) can be used to realistically simulate clay deformation, where the clay in the contact area can be transported to low density areas to maintain the density distribution equalized.

Common game-based physics engines (e.g., Bullet, PhysX, Unreal Engine, etc.) allows real-time simulation of soft-body deformations such as rope, cloth or any other shapes constructed by the spring-mass model. How well these models can be used for the purpose of real-time deformation with hand tracking is tested in Section 4.7.4. Controllable deformation techniques done with virtual hands are proposed in Section 4.8.4 for the condition when a physics engine is not available.

4.7 Using physics engine with Unity

In this section, it is investigated how well the operations of taking, positioning, releasing and deformation of objects can be simulated in the best condition when a physics

engine is available. The testing is conducted on the platform of Unity (unity3d.com), which is a free-licensed (personal version) game engine. Unity has its own integrated physics engine to be used for developing 3D games and simulation. The choice is majorly driven by its wide popularity, cross-platform applications, as well as its seamless workflow with other mainstream platforms (e.g., Autodesk Maya) and devices (e.g., Oculus Rift and HTC Vive).

The Leap Motion company provides several hand interaction assets [116] for the Unity platform. The core asset is the dependency for all the other modules. It implements a minimal interface to transfer and visualize hand tracking data from Leap Motion controller. The interaction engine allows for extra hand interaction ways (e.g., grasping and throwing), which are not supported in the physics engine, to be used on the platform. The Graphic Renderer optimizes rendering of a computationally efficient interface for mobile platforms, while the hand module asset provides different geometrical hand models. These hand interaction assets allow for quick prototyping of hand interaction in the Unity platform and they have been used for developing applications of Leap Motion controller, e.g., in [117].

The following subsections investigate how efficiently the operations of taking, positioning, releasing and deforming can be done on the Unity platform (version 2017.4.1f1) is investigated. The core asset (version 4.3.4) and the interaction engine (version 1.1.1) were used in the experiment. The graphic model of the virtual hand used in the experiment was constructed as a simple low-poly model, and the collision model matched the geometric model. Virtual objects of different shapes and sizes were used in the study. The desktop computer used for experiments was Alienware Auroro R5 with the Intel Core i7-6700 Processor (4-Cores, 8MB Cache, Turbo 61Boost 2.0, up to 4.0GHz) and

32GB (4X8GB) DDR4 2133MHz 62 SDRAM Memory. The Graphics card was NVIDIA GeForce 63 GTX 1080 with 8GB GDDR5X. The monitor was 24 inches with a resolution of 1920 x 1200.

4.7.1 Taking

As it was observed in Section 4.3, the operation of taking is not directly supported by a standard physics engine – virtual objects will bounce away when the fingers are closed. The interaction engine uses explicit methods (as illustrated in Section 4.3) to implement this operation while the rigid body simulation by the physics engine is turned off to allow finger penetration. The physics engine is still used to track collision between virtual hands and objects, and the conditions for taking are explicitly defined by the interaction engine. Therefore, tests were separately conducted on both the recognition algorithms of the taking gestures (done by the interaction engine) and the quality and efficiency of collision tracking (done by the physics engine).

The tests show that an object can be taken only when it collides with the tips of the thumb and the fingers simultaneously, while both the thumb and the fingers are bent. The defined conditions allow for recognition of basic pinching gestures (see Figure 4-1) while other commonly used ways of taking (e.g., by bending fingers to wrap around the object, by holding with two hands, and by closing thumb to palm) cannot be tracked.

Figure 4-1 The pinching gesture that is implemented in the interaction engine for the operation of taking.

The shape models supported by Unity are defined with polygon mesh. Simple primitives, such as sphere, cylinder, box and capsule, can be directly created in the software, while more complex mesh models can be created from scripts or imported from third-party modeling tools, e.g., Autodesk Maya. Although there are no limitations on the size of the mesh, the mesh colliders implemented by the physics engine have to be convex with less than 256 vertices. The colliders can only be built in Unity but they are not allowed to be imported from any third-party software. Therefore, the precision of taking depends on to which extent the colliders fit the graphic models.

Figure 4-2 The geometrical mesh models created in Autodesk Maya and their corresponding colliders generated in Unity. (a), (b) and (c) are the original mesh models of the subdivided cube, the human lower jaw model, and the human skull model respectively in Autodesk, while (d), (e) and (f) are their corresponding colliders generated by Unity.

In order to find out the precision limits when taking different objects, three mesh models were created in Autodesk Maya and imported into Unity for the test. The models included a subdivided convex cube (vertices 602, edges 1200, faces 600, see Figure 4-2 (a)), a concave human lower jaw model (vertices 602, edges 1200, faces 600, see Figure 4-2 (b)) and a concave human skull model (vertices 1184, edges 3296, faces 2066, see Figure 4-2 (c)). Mesh colliders were added to the objects in Unity after the objects were declared as concave. The results show that Unity can estimate the geometry of the mesh and build a simplified collider rather than making one that exactly matches the graphical mesh model. The algorithm worked well on the subdivided cube model -- the generated colliders could well match the mesh but with much less surfaces, as it can be seen in Figure 4-2 (d).

However, there is an obvious mismatch between the graphical model and the automatically generated collider for the concave lower jaw model, as it can be seen in Figure 4-2 (e), because the model cannot well fit to a convex mesh collider. For the complex human skull model, a larger enclosing collider (with around 50 faces) was generated which could roughly match the geometry by covering the shape (see Figure 4-2 (f)).

Figure 4-3 The efficiency test of the taking operation. The scene was occupied with 200 skull models.

Since the operation of taking is driven by the collision between virtual hands and objects, the efficiency of tracking should be influenced by the number and complexity of colliders used in the scene. Therefore, the efficiency was tested by increasing the number of colliders in the scene. The skull model with the convex collider (of around 50 faces) was used for the efficiency test of the taking operation. The colliders of virtual hands were modeled with 16 box primitives (three box colliders for each finger and one for the palm), which could well match the graphic model. The results show that the taking operation could be quite efficiently tracked during the whole process of increasing the number of skull from 1 to 200. The FPS (frames per second) remained at around 65 during the whole process. Although the potential capability of the physics engine may allow for more skulls to be

simulated, the testing stopped at this point since the collider number had become large enough for normal virtual assembly or modeling tasks.

To summarize the results, the physics-based environment allows shape models defined with polygon mesh to be taken by the basic pinching gesture, which is performed by closing thumb and one or several fingers simultaneously. The major advantages that can be provided by a physics engine for the operation of taking are 1) quick building of simple colliders for virtual objects and 2) efficient tracking of collision between virtual hands and objects. The provided functionalities are sufficient for making game applications without requiring a high precision of interaction, while it becomes prohibitive to build a precise collider for a complex mesh – the model has to be broken into small convex pieces before importing to Unity for generating colliders that can be supported by the platform.

4.7.2 Positioning

As discussed in Section 4.4, natural positioning ways can be classified into taking-based ways and non-prehensile ways (e.g., by pushing and turning). The taking-based ways have been implemented in the interaction engine of Leap Motion, while non-prehensile ways can be directly used in the physics simulation of Unity. These two kinds of positioning ways were respectively tested.

The taking-based ways are implemented in the interaction engine with explicit methods – motion of the object is explicitly defined according to the hand translation and wrist rotation after the object is taken by the hand. The method worked well for translation in the interaction space, but it became inconvenient for rotation – the magnitude of rotation was highly constrained by the limited rotation extent of the wrist. In real-life interaction, large-

scale rotation can be achieved by passing the object to the second hand, or using in-hand fingers motions when the size of the object is small. However, both of these two interaction ways are not supported by the interaction engine. The virtual object fell down most of the times when it was passed to the other hand (see Figure 4-4(a)). Therefore, in order to make large-scale of rotation the virtual object had to be placed on the supporting surface and taken again.

Non-prehensile interaction ways, including pushing (translating) or turning (rotating) the object, done by using one or multiple fingers were tested. The friction between the object and the ground plane was set to a small value of 0.02, so that the object could be easily pushed by a hand while not sliding away. The bounciness was set to 0 with the aim to prevent the object from jumping away when finger penetration happened.

Figure 4-4 Testing on the operation of positioning. (a) Taking-based ways of positioning. (b) Non-prehensile interaction ways.

The results show that the virtual object could be reliably pushed or turned when only one finger was used (see Figure 4-4 (b)). However, when multiple fingers were used, the object jumped away despite the bounciness set at 0. The jumping problem also happened

due to occasionally unstable tracking of finger motion. These problems limited the non-prehensile interaction ways to be precisely used for adjusting the posture of the object.

To conclude, the taking-based interaction ways work fine for object translation but they are not very convenient for object rotation – the extent of rotation is limited by the rotation range of the wrist. However, the natural interaction way, which can increase extent of rotation by passing the object to the other hand, is not well supported by the interaction engine. The non-prehensile ways can be plausibly simulated in the environment, but the standard physics engine cannot be properly tailored to be used for precise hand interaction tasks – the jumping problem limits the precision of interaction. These observed problems are considered to be solved by the novel interaction techniques further illustrated in Section 4.8.2.

4.7.3 Releasing

Releasing is the operation inverse to taking. It is implemented with the explicit methods in the interaction engine. Tests were conducted to investigate what kind of releasing gestures can be recognized and how precise a virtual object can be released.

The results show that a virtual object can be released when either of these two conditions are satisfied: 1) the tips of the thumb or the fingers are no longer in contact with the object; 2) either the thumb or the fingers are fully extended. The second condition can alleviate the “sticky release” problem to some extent when the fingers completely sink into objects with large sizes.

Although motion of the released object is constrained in the physics-based environment since the orientation of the object can be automatically adjusted when it fell

down, it was still too difficult to precisely release an object to the target place due to the problems of hand tremor and jump release. A little momentum created by the delayed finger motion always displaced the object. Novel algorithms are further proposed in Section 4.8.3 to solve the observed jump release problem.

4.7.4 Deformation

The integrated physics engine of Unity PhysX allows for real-time simulation of soft-body deformations such as rope, cloth or any other shapes constructed by the spring-mass model. The Unity platform allows rope and cloth simulation to be used, but the most useful soft object simulation is not accessible in the software. Therefore, it was considered to use the third-party asset DefKit [118] to test the soft-body simulation provided by the PhysX engine. DefKit is a free asset that can be downloaded from Unity asset store. It tetrahedralizes 3D objects and connects all these rigidbodies through mass-spring model (MSM), and the simulation is left to the Unity's built-in PhysX engine. The physics parameters of the PhysX engine, such as springs stiffness or friction, are allowed to be adjusted in the asset.

Figure 4-5 Testing on soft-body deformation. (a) The object deformation when it collides with the ground. (b) Deformation by the palm. (c) Deformation by fingers. (d) Model crash.

The example bunny model (with 524 points, 2607 edges, 950 triangles and 1609 tetras) provided in the asset was used for the test. The model was initially placed in the mid-air and it fell down in the beginning of simulation. The results showed that the deformation of the model was quite realistically simulated when it collided with the ground plane (see Figure 4-5 (a)) and the whole palm (see Figure 4-5 (b)). However, the deformation became less accurate when fingers were used to deform the object -- they could easily penetrate the soft body (see Figure 4-5 (c)). It was because the collision model of the soft body was rather simplified so that the interactive rendering speed of simulation could be maintained. The collision was only tested for the vertices of the mesh of the soft object. In addition, the soft model was so fragile that it could be easily ruined during interaction (see Figure 4-5 (d)).

Therefore, the existing available physical deformation models provided by the physics engine are still not precise and stable enough to be used for real-time modeling done with hand tracking.

According to the observations, although the standard soft body simulation provided by a common physics engine allows for some coarse deformations to be done by palm or whole hand, it is still not precise and stable enough to be used for local deformations done with fingers in modeling tasks. In addition, the colliders provided by the physics engine are not allowed to be used for deformable objects. Therefore, very incremental benefits can be expected from a physics engine for precise deformation tasks done with virtual hands. The deformation models are considered to be implemented with algorithms in Section 4.8.4.

4.8 Interaction without a physics engine

As observed in the previous section, a physics engine indeed provides some benefits and convenience for simulation of virtual assembling and modeling tasks done with virtual hands. However, there are still some problems when using them for precise modeling tasks, since the physics engines are commonly designed for game simulation rather than being well tailored for precise tasks. Therefore, it is considered whether the alternative interaction techniques designed without using a physics engine can be more precisely and reliably used. In the following part of this section, interaction techniques for the operations of taking, positioning, releasing and deformation are respectively proposed in Section 4.8.1-4.8.4 with a focus on solving the observed problems. In addition, finger correction algorithms are proposed in Section 4.8.5 in order to solve the finger penetration problem during interaction.

4.8.1 Taking

In the Unity platform, the virtual operation of taking was implemented with explicit methods by the interaction engine while the collision between hands and objects were tracked by the physics engine. The explicit method allows basic pinching gestures to be recognized. However, the object falls off hand when a user tries to pass the object to the other hand for continuous rotation (as observed in Section 4.7.2). This is because the system gets confused on the intermediate state when the object is simultaneously taken by two hands – it fails to distinguish which hand is taking the object. In fact, this case should be treated as a separate bimanual taking case, which is different from the unimanual taking case. Therefore, methods for both unimanual and bimanual taking are proposed.

Suppose collision between virtual hands and objects can be tracked (relevant algorithms can be found in Section 4.9.3). Similar to the method implemented in the interaction engine, a virtual object can be taken by one hand when it collides with the tips of the thumb and the fingers simultaneously (unimanual taking). The bimanual taking mode is activated when the object is taken by two hands simultaneously. These two modes can be switched when adding or removing the second hand so that the object will not fall off when it is passed from one hand to the other one.

4.8.2 Positioning

Positioning ways were classified into taking-based and non-prehensile interaction ways. As observed in Section 4.7.2, the taking-based ways were implemented in the interaction engine with explicit methods while the non-prehensile interaction ways are directly supported by the physics engine.

Figure 4-6 Illustration of the positioning technique with one and two hands. (a) Local coordinate system of a hand. (b) The positioning technique with one hand. (c) The positioning technique with two hands.

Following the discussion in the previous section, the positioning techniques are separately considered as unimanual and bimanual. When the object is taken by one hand, the object is fixed in the local coordinate system of the virtual hand (Figure 4-6 (a)) so that it moves along with the whole hand motion, as it can be seen in Figure 4-6 (b). When the object is taken by two hands simultaneously, translation of the object is calculated by the translation of the mid-point of the two hands and the increment of rotation is calculated by rotation of the vector connecting two hands, as it can be seen in Figure 4-6 (c). For both cases, transformation of the object is accumulated from the transformation computed between each two consecutive frames, so that these two positioning ways can be seamlessly

switched. By incorporating the bimanual mode, natural interaction ways done with two hands can be reliably used to further increase the extent of rotation.

Although non-prehensile interaction ways, such as pushing and turning, can be plausibly simulated in a physics-based environment, a standard physics engine cannot properly deal with finger penetration, which limits its use in precise hand interaction tasks. Therefore, alternative explicit methods are considered to be used in order to have full control of object motion in every frame so that the observed problems can be avoided.

The non-prehensile interaction ways of pushing (translation) and turning (rotation) are implemented with separate explicit methods. This is because it is difficult to come up with plausible and controllable ways that can control translation and rotation simultaneously using explicit methods without simulating physical factors such as gravity, velocity, acceleration, etc.

In real-life interaction, an object can be pushed after contact is established between the object and any part of the hand. Similarly, a virtual object can be pushed when the finger involved in collision moves in the opposite direction of the normal from the collision point. Specifically, suppose the hand collides with a virtual object in the frame n at point C_n . The object is pushed by $(\overrightarrow{C_n C_{n+1}} \cdot \vec{n})\vec{n}$ in the normal direction at the next frame $n+1$, if $\overrightarrow{C_n C_{n+1}} \cdot \vec{n} > 0$. When the object is in contact with multiple fingers simultaneously, motion of the object becomes a sum of motions created by individual fingers. The collision between hand and the object is evaluated again at the next frame and the process is iterated.

Figure 4-7 Turning an object around an axis.

For the operation of turning (rotation), it is assumed that motion of the object is constrained at its center or an axis so that the translation is fixed. When a virtual finger collides with the object at frame n , the vector $\vec{V}_n$ starting from the rotation center and ending at the contact point is tracked. If the finger was in contact with the object in the previous frame $n-1$, rotation of the object is computed from the change of the vectors $\vec{V}_{n-1}$ and $\vec{V}_n$ in two consecutive frames of $n-1$ and n (see Figure 4-7 for the case of axis-constrained rotation). When several fingers are involved in the rotation simultaneously, the vector resulting the largest rotation angle will be used.

4.8.3 Releasing

Releasing is the operation inverse to taking. A virtual object can be released when it is no longer in contact with the hand. The observed problems of this operation are hand tremor and jump release, while they remain unsolved in the physics-based simulation. In this

connection, a method is proposed to enhance the precision of releasing and to solve the observed problems. The idea of the method is to adjust the precision of controlling according to hand motion speed. The precision is enhanced when the hand motion speed is low for precise releasing, while a maximum motion speed is set to prevent the jumping problem (caused by unstable tracking).

In the algorithm, transformation of the object is scaled according to hand motion speed $\vec{V}_H$, which is estimated by the displacement of the hands in two consecutive frames. The object transformation is scaled down (stabilized for releasing), when the magnitude of $\vec{V}_H$ is below a threshold v_{min} , which resembles the case when the users naturally slow down hand motion speed to carefully place the object. The object is frozen when $\vec{V}_H$ exceeds the threshold v_{max} to prevent jumping. Details of the algorithm are illustrated as follows:

Step1: Compute the relative transformation of hand between current frame n and previous frame $n-1$

$$W_{rel}^n = W_{hand}^n (W_{hand}^{n-1})^{-1} \quad (4.1)$$

Step2: Compute the expected transformation of the object from hand transformation for current frame and

$$W_{object}^{n-exp} = W_{rel}^n W_{object}^{n-1} \quad (4.2)$$

Step3: Interpolate the relative transformation of the object between the two consecutive frames

$$W_{object}^n = \mathcal{F}_{slerp}(W_{object}^{n-1}, W_{object}^{n-exp}, \alpha) \quad (0 \leq \alpha \leq 1) \quad (4.3)$$

where,

$$\alpha = f(\|\vec{V}_H\|) = \begin{cases} 0 & \|\vec{V}_H\| > v_{max} \\ 1 & v_{min} \leq \|\vec{V}_H\| \leq v_{max} \\ \|\vec{V}_H\|/v_{min} & \|\vec{V}_H\| < v_{min} \end{cases} \quad (4.4)$$

The idea of adjusting precision of object motion according to hand motion speed was initially explored in [119]. Their method was proposed for gesture-based interaction so that the offset between the virtual object and the hand, which is resulted from different motion speeds of them, is invisible to users. The offset is accumulated during interaction so that it has to be artificially reset when it grows too big, which makes the object jump around. In comparison, my algorithm based on collision-based interaction can be more reliably used – it allows the offset to be visible by displaying both the virtual hand and the object so that users can control it as they want without causing the object to jump.

4.8.4 Deformation with one and two hands

In real-life interaction, local deformations can be created by fingers, while large-scale global deformation can be made with two hands, e.g., by rolling, stretching, pressing or twisting. As observed in Section 4.7.4, the soft-body deformation provided by a physics engine is not precise and stable enough to be used for modeling tasks done with hands. Therefore, alternative deformation techniques are considered in this section.

Implementation of local deformation made by hands requires to 1) track collision between hands and objects and 2) use some methods to define the local deformation of the object. These can be done in different ways according to different shape models. For example, when the shape is defined with polygon mesh, collision can be tracked by calculating the distance between the mesh and points sampled on hand. Deformation can be made by displacing the vertices in the vicinity of collision area with hands. For example,

local deformation by fingers was simulated by centering Compactly Supported Radial Basis Functions, or CSRBF, on fingertips to warp the space around a mesh [112]. The modeling application “sculpting” [94] allows local deformation to be created with hands on the dynamic mesh model [120], which supports arbitrary changes of the topological structures of the mesh. When the shape is defined with volumetric or implicit models, local deformation can be created by subtracting or adding materials on the shape using Boolean operations. A pilot example of making fingerprints on implicit models is shown in Figure 5-8. In the example, the deformation is created by subtracting a group of ellipsoids defined with implicit functions centered on the finger bones from the sphere when the virtual hand collides with it.

Figure 4-8 An example of making hand imprints using implicit functions.

In common 3D modeling systems, global deformations (such as bending, twisting, tilting, etc.) are implemented with non-linear functions [113] that can warp the object space. Simulation of global deformations done with two hands requires to 1) make conditions to define the start and end of deformation, and 2) extract parameters from hand motion to

define the extent and direction of deformation. For example, the deformation can start when the virtual object is taken with two hands simultaneously and is deformed globally by tracking the hand rotation and translation. For the operation of twisting, the twisting angles from the two hands can be calculated by projecting the palm rotation angle onto the plane vertical to the rotation axis, as shown in Figure 4-9, while the twisting axis can be pre-fixed or computed according to location of hands. Scaling can be performed by taking the object with two hands and moving them apart or close. The extent of scaling can be defined according to the distance between the two hands. For the operation of bending, both translation and rotation of two hands should be tracked to define the extent and direction of deformation.

Figure 4-9 An example of twisting by two hands.

4.8.5 Additional algorithm on virtual finger correction

Figure 4-10 The correction algorithm for finger penetration. (a) Correction of finger sinking. (b) Correction of finger passing-through.

The problem of finger penetration, caused by unconstrained virtual finger motion in the mid-air, makes it difficult to take virtual objects with slender or flat shapes, thin edges, or overall small sizes. In addition, it is also confusing for the users to see the visual artefacts of finger penetration when performing the interaction tasks. The problem is not properly solved in the physics engine – it exerts a large impulse force to push the object away, which seriously affects the precision of interaction. Therefore, robust finger correction algorithms are proposed in this section to enhance the precision and capability of interaction.

The algorithms can deal with two penetration cases: 1) virtual fingers sink inside objects and 2) virtual fingers pass through virtual objects. For the first case, the algorithm opens the finger joints until the tip is outside touching the surface of the object (see Figure 4-10 (a)). For the second case, the algorithm first opens the finger until the tip is inside the object. Treating this as the initial finger posture, the first algorithm is executed again to push the finger out of the object (see Figure 4-10 (b)).

For the first case, when the fingertip $\vec{P}_{tip}^n$ is inside the object at frame n , the correction algorithm opens DIP, PIP and MCP joint angles of the finger (the joint names are illustrated

in Figure 2-1) simultaneously until the fingertip $\vec{P}_{tip}^n$ is moved outside while still touching the surface of the object, as it can be seen in Figure 4-10 (a).

The DIP, PIP, and MCP joint angles of the finger at frame n is denoted as $\mathbf{Q}_n = \{\theta_{DIP}^n, \theta_{PIP}^n, \theta_{MCP}^n\}$. Since the lengths of the finger bones are fixed, the tip position of the finger can be calculated from $\mathbf{Q}_n$ and the palm position $\vec{P}_{palm}^n$ at frame n by a function f .

$$\vec{P}_{tip}^n = f(\mathbf{Q}_n, \vec{P}_{palm}^n) \quad (4.5)$$

Suppose the set of maximum angles of joints DIP, PIP and MCP when the finger is fully extended is $\mathbf{Q}_{ext} = \{\theta_{DIP}^{max}, \theta_{PIP}^{max}, \theta_{MCP}^{max}\}$, then the fingertip position of the extended finger at current frame n can therefore be calculated from $\mathbf{Q}_{ext}$ and the palm position $\vec{P}_{palm}^n$ as:

$$\vec{P}_{tip}^{ext} = f(\mathbf{Q}_{ext}, \vec{P}_{palm}^n) \quad (4.6)$$

Suppose $\vec{P}_{tip}^{ext}$ is outside the object. The algorithm uses a binary searching method to find the best joints angles of DIP, PIP and MCP which allow the fingertip to be moved outside while still touching the surface of the object. The algorithm is as following:

if $\vec{P}_{tip}^n$ is inside the object **and** $\vec{P}_{tip}^{ext}$ is outside the object

$Q_{in} \leftarrow Q_{in};$

$Q_{out} \leftarrow Q_{ext};$

while $\|f(Q_{in}, \vec{P}_{palm}^n) - f(Q_{out}, \vec{P}_{palm}^n)\| > \varepsilon$

if $f(Q_{(in+out)/2}, \vec{P}_{palm}^n)$ is inside the object

$Q_{in} \leftarrow Q_{(in+out)/2};$

else

$Q_{out} \leftarrow Q_{(in+out)/2};$

end if

end while

output $Q_{out};$

end if

where

$$Q_{(in+out)/2} = \left\{ \frac{\theta_{DIP}^{in} + \theta_{DIP}^{out}}{2}, \frac{\theta_{PIP}^{in} + \theta_{PIP}^{out}}{2}, \frac{\theta_{MCP}^{in} + \theta_{MCP}^{out}}{2} \right\} \quad (4.7)$$

Next, it is considered to solve the second case of penetration – a finger completely passes through the virtual object. This problem happens when the virtual object is too thin or small. Suppose the finger passes through the object at frame n . In the previous frame $n-1$, the tracked finger must be inside the virtual object while the virtual finger is corrected and placed outside of the object. Therefore, the distance between the tips of the two fingers should be $D_{n-1} > \varepsilon$. In this way, the frame can be targeted by tracking whether D_n is discontinuous in two consecutive frames.

When the second case of penetration is observed, the algorithm (searching algorithm) extends the tracked finger gradually while computing whether the finger collides with any virtual objects during the process. If the finger collides with any virtual objects during the process, the first correction algorithm is executed to open the finger outside the object, as it can be seen in Figure 4-10 (b). If finger does not collide with any virtual objects, the finger posture will not be corrected. This case may happen when the finger has moved away from the virtual object.

The complete correction algorithm is presented in Figure 4-11. Different correction algorithms are executed according to different cases of penetration. The object is pushed when D_n is bigger than a threshold T in order to prevent one or multiple fingers from sinking too deeply inside the object so that they cannot be corrected outside.

Figure 4-11 Flowchart of the complete finger correction algorithm.

4.9 Implementation

4.9.1 Platform selection

The proposed interaction techniques in the previous section can be implemented on different platforms. The major limitation of using a physics-based environment (e.g., Unity) is the lack of support on interactive modeling for the deformation tasks. The provided soft-body models cannot meet the expected precision of modeling with hands. Indeed, common physics engines are majorly designed for game simulation, while 3D models are mostly

made in additional modeling tools (e.g., Autodesk Maya) and imported to the environment. It is therefore considered to select a modeling platform which comes with a rendering pipeline for simulation of the deformation tasks done with hands.

In this connection, Virtual Reality Modeling Language (VRML) and JavaScript (VRML scripts) have been selected as the programming language. This is motivated by the available extension of VRML and X3D -- Function-based extension of VRML and X3D (FVRML/FX3D) – which extend the capability of standard VRML and its successor X3D to support creating object's geometry and sophisticated graphics appearances by using analytic implicit, explicit and parametric functions [121]. Both static and time-dependent shapes and operations can be rendered. In contrast to the polygon-based models of VRML/X3D, function-based nodes greatly reduce the size of the models and provide an unlimited level of detail. Also, these nodes allow for using set-theoretic as well as any analytically-defined geometric operations. The extension was developed for collaborative shape modeling as well as for building shared virtual worlds.

In FVRML/FX3D, the object FRep [122] definitions are written straight in the VRML/X3D code as function scripts together with the standard definitions of VRML and X3D. Specifically, from any surface defined by $f(x,y,z)=0$, a solid object is then defined by inequality $f(x,y,z)\geq 0$, where points on the surface result in 0 values of the function, while inside the solid object the values are positive, and outside – negative. The FVRML/FX3D has been developed and used in the lab for more than ten years. It has been successfully used as an educational tool for students to visualize complex mathematical functions [123]. The current interactive tool uses mouse and keyboard as input devices, which limits the

interactivity for the students. This thesis also considers a possibility to add new modalities of using hands for interactive shape modeling.

In order to implement the proposed interaction techniques on the selected platform, further works need to be done on implementing 1) a plugin to make hand tracking from Leap Motion controller available on the platform, and 2) both deformation and collision models for hand-based interaction.

4.9.2 Plugin implementation

BS Contact, developed by Bitmanagement Software, is used as the visualizer of codes implemented with VRML/X3D. It is commonly used to display scenes and it is a plugin to several Internet browsers. The advantage of the platform is that it can provide an interface to support input devices by extending VRML/X3D through a DeviceSensor node for receiving the input data from the devices. The platform Bitmanagement BS Contact also allows the created virtual environments to receive and send data to many VR devices. Hence, a plugin was implemented to deliver hand tracking data from optical sensors to the scene created with VRML or X3D. This section of work has been published in [109].

The plugin was developed in C++ using Microsoft Visual Studio 2015 on Windows 10. It is contained in a dynamic-link library file, which needs to be registered with Microsoft Windows OS to write the hand tracking data into the standard VRML/X3D file. The current plugin allows hand tracking data from Leap Motion controller and Intel RealSense SR300 to be sent into the virtual environments. uSens Fingo will be also supported in the near future, since its official website declares the C++ interface will be published soon.

Figure 4-12 Routing coordinates of the index finger to the interactive java script in VRML and linking them to the position of the moving ball.

In the plugin, a class with the name of *CDevice* in the file *Device.h* was declared to store hand tracking data from different sensors. The class contained a method *CDevice::GetVal()*, which can extract hand tracking data from the APIs provided by the SDKs of the optical sensors. An in-process COM object was implemented to receive hand tracking data from *CDevice* sensors and write the data to the VRML file. Correspondingly in the VRML file, a *DeviceSensor* node needs to be declared for receiving the hand tracking data from the plugin. BS Contact instantiates and registers the COM object for the input devices that is requested by a *DeviceSensor* to accomplish the data communication. Initialization, and sending data to the fields in the *DeviceSensor* node was achieved by using several External Authoring Interface (EAI) functions provided by Bitmanagement Software SDK.

An example shown in Figure 4-12 illustrates how the plugin works. In the demo, motion of a red sphere is controlled by the index fingertip, which is tracked by Leap Motion controller. In the *VRML* codes, the *eventOut* field *skeleton1* is declared in the *DeviceSensor* node. This field contains 3D position data of all the fingertips received from the plugin. The field of *skeleton1* is routed to an *eventIn* field named *fingertip2* from the *LeapTipVisualizer* node, where the 3D position of the index fingertip is extracted and stored in *tip2*. Finally, the 3D index fingertip position *tip2* is routed to the translation of a red sphere, which is declared in the node of *Tipball1*, so that the sphere can move along with the motion of index finger.

4.9.3 Implementation of mathematical models

The collision and deformation models are designed using implicit functions which are written straight as function scripts in the code to be directly rendered by FVRML.

Collision model

The collision model of a virtual hand was constructed by a group of points sampled on the hand. The precision of the model can be adjusted by controlling the number of the sampled points. Collision between the sampled points on the virtual hand, e.g., $P_i(x_i, y_i, z_i)$, and an object whose geometry is defined by $f_0(x, y, z) > 0$ can be detected when $f_0(x_i, y_i, z_i) > 0$. When the object is transformed by T in the space, its collision with the hand can be computed from $f_0(T^{-1}(x_i, y_i, z_i))$, which can be regarded as when the point is inversely transformed by T^{-1} while the object is static. In a similar way, collision between the hand and the object deformed by D can be tracked by computing $f_0(D^{-1}(x_0, y_0, z_0))$.

The collision models of the objects when they are being deformed or manipulated in tasks are updated in real-time. Using collision models defined with implicit functions also allow for achieving an ultimate precision in tracking collision between fingertips and virtual objects. In contrast to common polygon-based models, function-based representations allow for a great reduction of the model sizes and provide an unlimited precision for collision tracking.

Deformation model

Making a set of deformations on the shape $f_0(x, y, z)$ is done by applying a set of deformation functions $D_1 \dots D_n$ on the initial function, resulting in a new shape defined by

$$f_d(x, y, z) = f_0(D_1^{-1}D_2^{-1} \dots D_n^{-1}(x, y, z)) \geq 0 \quad (4.8)$$

Figure 4-13 Pinching a piece of elastic clay. Fingerprints are created by moving two offset functions centered on C_1 and C_2 .

When using function-based representations, local deformations done with fingers are simulated by using field functions centered near the fingertips, as it can be seen in Figure 4-13, and by subtracting the offset functions from the shape function:

$$f_1 = f_0 - \sum_{i=1}^n \alpha_i f(r_i) \geq 0 \quad (4.9)$$

where

$$f(r_i) = \begin{cases} 0 & r_i > r_0 \\ (1 - r_i)^2 & 0 < r_i < r_0 \end{cases} \quad (4.10)$$

and

$$r_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2}. \quad (4.11)$$

$f(r_i)$ is an iso-scalar offset function defined in a sphere area, which is centered on (x_i, y_i, z_i) with a radius r_0 . n is the number of fingers touching the object — each finger is associated with the offset function $f(r_i)$.

For the twisting example of global deformation illustrated in Section 4.8.4, the deformation starts when the object is grasped by two hands simultaneously at its two ends. The axis of twisting is pre-fixed as the central axis of the cuboid. Suppose the twisting axis is the principle X -axis. The x coordinates of the object fall in the interval of $[x_{min}, x_{max}]$. The twisting angles of the two ends are θ_{left} and θ_{right} . The global deformation of twisting is defined by

$$\begin{aligned} \theta &= \frac{x - x_{max}}{x_{min} - x_{max}} \theta_{left} + \frac{x_{min} - x}{x_{min} - x_{max}} \theta_{right} \\ X &= x \\ Y &= y \cos(\theta) - z \sin(\theta) \\ Z &= z \sin(\theta) + y \cos(\theta). \end{aligned} \quad (4.12)$$

4.9.4 Code structure on FVRML

The simplified code structure of implementation is shown in Figure 4-14. The standard VRML file consists of different parts in the form of “node”. There are four kinds of nodes involved in the figure – device sensor node, shape node, transformation node and script

node. Different nodes communicate with each other by sending and receiving data in corresponding fields.

The device sensor node “*ds*” receives hand tracking data from the tracking device and sends the data to the script node “Modeller”. The script node receives the data and performs various calculation in functions written in JavaScript, including computation of the finger correction algorithms, calculation on object transformation and deformation, collision detection, updating the collision model of the object, etc. After computation, it sends data to corresponding transformation and object nodes for visualization. Virtual hands in the scene are visualized by shape nodes of spheres (hand joints) and cylinders (hand bones) defined in basic VRML shape nodes, while the geometry of virtual deformable object is written as function string in the “definition” field of the “FShape” node. The shape nodes are child nodes of transformation nodes, which define the translation, rotation and scaling of the objects. The transformation of the virtual hands and the shape are updated by the corresponding “SFVec3f” and “SFRotation” fields received from the “Modeller” node, while the geometry of the shape is updated by the “SFString” field “scriptout” from the “Modeller” node.

Figure 4-14 Code structure in VRML.

4.10 User study

In this section, a user study was conducted to compare virtual interaction with real-life interaction. It is expected that the proposed techniques should allow virtual tasks to be performed as naturally and efficiently as real-life interaction.

4.10.1 Tasks

Figure 4-15 Real-life and virtual tasks in the experiments. (a) putting a cap onto a container, (b) screwing a nut onto a bolt, (c) making elastic deformation with one hand, and (d) twisting deformation done by two hands.

Four interaction tasks have been selected for comparison between virtual and real-life interaction and they are (a) putting a cap onto a container, (b) screwing a nut onto a bolt, (c) making elastic deformation with one hand, and (d) twisting deformation done by two hands, as it can be seen in Figure 4-15. The tasks in virtual environment were made the same as the tasks in real-life conditions. These tasks cover the investigated basic operations of taking, positioning, releasing and deformation. They consist of different interaction ways done with fingers, wrist and one or two hands. Hand motion was restricted within the best tracking area, as tested in Section 3.2, with a width and depth of 40 cm and a height from 10 cm to 40 cm. 1:1 coordinate mapping of the tracking area to the virtual modeling space

was used. The sizes of virtual objects were such that they could be grasped with the distance of 3 cm between the thumb and the fingers, as measured in Section 3.2.3.

4.10.2 Apparatus

The following equipment was used: a desktop computer, a Leap Motion controller, a pair of bolt and nut, a clay container, as well as a set of air drying colored clay. The desktop computer is DELL Precision WorkStation T7500, with the Intel CPU (2.40GHz) and 12.0GB memory installed on. The Graphic card is NVIDIA Quadro 2000 with 1 GB memory. The monitor is 24 inches with a resolution of 1920x1200.

Figure 4-16 Apparatus and experiment setup.

A Leap Motion controller was connected to the desktop computer through a USB port. It was placed on the table facing up in front of the screen. The interaction height was approximately 20 cm above the desktop. A pair of bolt and nut and the set of air-drying colored clay were also placed on the desk for participants to interact with (Figure 4-16).

4.10.3 Participants

16 participants were invited for the study. Half of them were female. Average age was 28 ($sd=3.12$). 5 of them had experience with mid-air interaction. The mid-air interaction devices they previously used before included Leap Motion controller, Microsoft Kinect, Wii and HTC VIVE.

4.10.4 Procedure

The user study started with an introduction to the tasks, after which the participants were allowed to practice until they felt ready for the tasks. The formal part consisted of 4 experiments: clay container capping (Experiment 1), bolt and nut assembly (Experiment 2), clay pinching (Experiment 3), and bimanual twisting (Experiment 4). Each experiment contained a real-life task and a virtual task. Time spent on each task was recorded by the experimenter. There were $(1 \text{ (real-life)} + 1 \text{ (virtual)}) \times 4 \text{ (experiments)} = 8$ trails in total.

4.10.5 Results

Table 4-1 Time spent on real-life and virtual interaction of each task.

	Experiment 1		Experiment 2		Experiment 3		Experiment 4	
	Real	Virtual	Real	Virtual	Real	Virtual	Real	Virtual
Average time	3.8	9.1	4.4	17.2	3.5	6.6	3.2	7.6
Standard deviation	1.2	2.1	1.4	4.5	1.4	2.6	0.9	3.9

All of the participants successfully accomplished the virtual tasks with the same interaction ways they used in the corresponding real-life interaction tasks. They could comfortably use the proposed interaction techniques without being affected by the problems

when using a physics engine (as observed in Section 4.7). T-test was performed to compare between the time spent on virtual and real-life interaction for each experiment. However, with reference to Table 4-1, the results show that the virtual interaction is significantly slower than the real-life interaction ($p < 0.01$ for all the experiments). The biggest time difference occurs in Experiment 2, where quite similar hand movements were involved as in Experiment 1, but the sizes of virtual objects were smaller. Interaction with small objects was indeed more difficult in the virtual environment where force feedback is missing. It was also considered to improve the efficiency of virtual interaction by providing stereo display with Oculus Rift. However, the improvement was still incremental.

By further analyzing the results, it was found that collision-based virtual interaction cannot outperform real-life interaction in principle with the current setup using optical tracking where tactile feedback is missing. In real-life interaction, hand and finger motion can be dexterously and rapidly controlled according to the tactile feedback received from the collision with the objects. In the virtual interaction without tactile feedback, however, every movement becomes clumsy – more time has to be spent on reaching hands to objects and using exaggerated finger motion while waiting for the delayed visual signals (the comparison video is available from <https://youtu.be/p0AMElVVlGM>). The results are also confirmed by an experiment conducted by Garbaya and Zaldivar-Colado [124], who found the time spent on a virtual interaction task could be shortened from five times to twice as used in real-life interaction when force feedback was provided.

4.11 Summary

In this chapter, it was investigated whether collision-based interaction with optical hand tracking by Leap Motion controller could be used as naturally and efficiently as real-life interaction for virtual assembling and deforming objects within the reach of hands. Common interactions were classified into basic operations of taking, positioning, releasing and deformation, and they were tested in both environments with and without a physics engine.

The results show that a physics engine can indeed provide some benefits and convenience for simulation of virtual assembling and modeling tasks done with virtual hands. However, there are still some problems when using them for precise modeling tasks, since the physics engines are commonly designed for game simulation rather than being well tailored for precise tasks. The observed limitations of the physics engine are 1) It becomes difficult to build a precise collider for a complex mesh in the physics engine; 2) The jumping problem affects the precision of using non-prehensile ways for object positioning; 3) The soft-body deformation model is still not precise and stable enough to be used for local deformations done with fingers in modeling tasks. The observed limitations of the interaction engine provided by Leap Motion are 1) The extent of rotation is so limited by rotation range of the wrist when using the taking-based interaction ways; 3) The precision of releasing is affected by problems of hand tremor and jump release. 3) In addition, it also makes users confused when they see the visual artefacts of fingers penetration, which is allowed by the interaction engine.

With the goal of solving the observed problems, new interaction techniques were proposed based the condition without a physics engine. Specifically, the incremental

novelties have been achieved in designing 1) bimanual taking and positioning techniques that allow large-scale rotation done with two hands, 2) explicit methods for pushing and turning which are free of the jumping problem, 3) a method to enhance the precision of releasing, 4) deformation techniques done with one or two hands, and 5) finger correction algorithms that solves the penetration problem and enhance the capability of interaction. The proposed methods allow hand interaction to be done more precisely and reliably than using the standard interaction engine provided by Leap Motion, which makes them more useful in precise tasks, such as virtual assembly and shape modeling. The techniques can be used with different shape representations, collision or deformation models provided by different platforms. As a testbed, the interaction techniques were implemented with VRML with the function-based extension, while additional contribution has been achieved in the process to allow hand tracking available on the platform.

Virtual and real-life interaction tasks were compared in the user study. The results show that although the proposed interaction techniques can be reliably used, collision-based virtual interaction is intrinsically not as efficient as real-life interaction due to missing of tactile feedback in the current setup of optical hand tracking.

Chapter 5 Natural Gesture-based Interaction

5.1 Objective

According to the results obtained from the previous chapter, collision-based interaction does not allow virtual interaction to be done as efficient as real-life interaction, since more time has to be spent on adjusting fingers in collision with virtual objects. The main problem is that finger motion cannot be efficiently controlled without tactile feedback in virtual interaction. In comparison, gesture-based interaction should allow for higher efficiency by avoiding collision with virtual objects, since the gestures can be recognized anywhere in the tracking range of the optical sensor. In addition, the efficiency of interaction will not be influenced by the sizes of virtual objects when calculation of collision is replaced by recognition of a set of pre-defined gestures. Therefore, in this part it is investigated whether gesture-based interaction can be done as efficient and precise as real-life interaction. The interaction is considered to be done in a confined space above the desktop, which is the same interaction area with the experiment of collision-based interaction. There is no limitation on the sizes of objects, but they should be taken with a gesture where the distance between the thumb and fingers should be at least 30 mm (as tested in Section 3.2.3). As discussed in Section 4.1, the same set of basic interaction operations are to be designed including taking, positioning, releasing and deformation. This chapter was partially published in [125, 126].

5.2 Design of interaction techniques

Simulation of gesture-based interaction requires both gesture recognition algorithms and interaction techniques to be implemented. The gesture recognition algorithms track whether the gestures stored in the library are present, while the interaction techniques define how the object moves in response to hand motion. The Leap Motion API cannot provide much value to this study since it only implements some simple gesture recognition algorithms without providing corresponding interaction techniques. Therefore, a small set of essential interaction techniques are designed for the basic operations with an emphasis paid on the precision of interaction. The incremental novelty of the following algorithms lies in the rotation method (non-prehensile interaction) and the technique that enhances the precision of releasing. Most of the other interaction techniques are similar to those designed for collision-based interaction, except that they are based on gesture recognition rather than computing the collision between the virtual hand and the object.

5.2.1 Taking

There are many different ways (with up to 33 types [7]) to take objects. These gestures are formed mostly according to the object sizes, shapes and the intents of interaction. When performing the gesture for precise taking in the mid-air where finger motion is without any constraints, these real-life taking ways will fall into the kind which can be described as moving fingers towards the opposable thumb until they are close or connected (or called pinching). Depending on individual preferences, motions of index, middle, pinky finger or multiple ones of them can be involved in the process of taking. All these variations of this gesture can be recognized by tracking the minimum distance between the thumb and

fingers. Specifically, suppose the fingertip positions of five fingers are $\{\vec{P}_i\}_{i=1}^5$, where $i=1, 2, \dots, 5$ correspond to thumb, index, middle, ring and pinky respectively (See Figure 6-1). The distance between thumb and the four fingers are calculated one by one, and the object can be taken when the minimum of them is below a defined threshold D_T :

$$D_s = \min\{\|\vec{P}_1 P_i\|\}_{i=2}^5 < D_T \quad (5.1)$$

The proposed recognition algorithm allows the feature of the taking gesture to be recognized regardless of how many fingers are involved.

The bimanual taking mode is activated when both hands are performing the taking gesture. It can be seamlessly switched with the uni-manual mode by opening or closing the second hand.

Figure 5-1 Recognition of the taking gesture.

5.2.2 Positioning

As classified in Section 4.4, positioning of an object can be done by using taking-based ways and non-prehensile ways. Taking-based ways refer to the interaction happening after

the object is taken by one or two hands, while the non-prehensile ways refer to interaction done without taking up the object.

Algorithmically, the transformation of an object being positioned in frame n W_{object}^n is obtained by applying the relative transformation W_{rel}^n , calculated from hand motion, to the transformation of the object from the previous frame W_{object}^{n-1} .

$$W_{object}^n = W_{rel}^n W_{object}^{n-1} \quad (5.2)$$

In the following part of this section, it is discussed how W_{rel}^n can be defined for different kinds of positioning ways.

Taking-based interaction

Taking-based interaction ways are considered as unimanual and bimanual. As discussed in Section 4.8.2, the interaction done with two hands allows for the larger extent of rotation than using one hand. When an object is taken by one hand, its accumulated transformation from the frame n can be calculated from the relative transformation of the hand motion between current and previous frame.

$$W_{rel}^n = W_{hand}^n (W_{hand}^{n-1})^{-1} \quad (5.3)$$

When the object is taken by two hands simultaneously, the translation can be defined by the translation of the mid-point of the two hands, while the rotation is calculated by rotation of the vector connecting two hands. Suppose the positions of two hands in frames $n-1$ and n are $\overrightarrow{P_{h1}^{n-1}}, \overrightarrow{P_{h2}^{n-1}}$ and $\overrightarrow{P_{h1}^n}, \overrightarrow{P_{h2}^n}$. The translation of the mid-point between two hands is

$$\overrightarrow{d_{bi}^n} = (\overrightarrow{P_{h1}^n} + \overrightarrow{P_{h2}^n} - \overrightarrow{P_{h1}^{n-1}} - \overrightarrow{P_{h2}^{n-1}})/2 \quad (5.4)$$

The rotation axis and the rotation angle calculated from bimanual interaction are

$$\overrightarrow{N_{bi}^n} = \frac{\overrightarrow{P_{h1}^{n-1}P_{h2}^{n-1}} \times \overrightarrow{P_{h1}^nP_{h2}^n}}{\|\overrightarrow{P_{h1}^{n-1}P_{h2}^{n-1}}\| \cdot \|\overrightarrow{P_{h1}^nP_{h2}^n}\|} \quad (5.5a)$$

and

$$\theta_{bi}^n = \cos^{-1}\left(\frac{\overrightarrow{P_{h1}^{n-1}P_{h2}^{n-1}} \cdot \overrightarrow{P_{h1}^nP_{h2}^n}}{\|\overrightarrow{P_{h1}^{n-1}P_{h2}^{n-1}}\| \cdot \|\overrightarrow{P_{h1}^nP_{h2}^n}\|}\right) \quad (5.5b)$$

The transformation of the object accumulated from the frame n can be therefore defined by

$$W_{rel}^n = T(\overrightarrow{d_{bi}^n})R(\overrightarrow{N_{bi}^n}, \theta_{bi}^n) \quad (5.6)$$

where $T(\overrightarrow{d_{bi}^n})$ and $R(\overrightarrow{N_{bi}^n}, \theta_{bi}^n)$ are the translation matrix and rotation matrix of $\vec{d}$, and $\langle \overrightarrow{N_{bi}^n}, \theta_{bi}^n \rangle$.

Non-prehensile interaction techniques

Non-prehensile positioning ways involve using hands to push (translation) or turn (rotation) of the object without taking it up. The operations of pushing and center-fixed turning (implemented with collision-based techniques in Section 4.8.2) cannot be properly implemented with gesture-based techniques, since they inevitably involve computing collision between hand and objects. Therefore, only the operation of rotation around a fixed axis is considered.

Rotating the object around a fixed axis by hand requires collision to be established between fingers and the object, so that rotation of the object can be driven by movements of fingers, wrist or both of them. The palm can face to different directions during the process of rotation. The trend of rotation can be extracted from the motions of fingertips after they are projected onto the plane that is perpendicular to the rotation axis.

Figure 5-2 The interaction technique for rotation around a fixed axis.

Suppose $\vec{N}_p$ is the normal of the plane perpendicular to the rotation axis. Projecting the tip positions onto the plane results in five fingertips positions of $\{\vec{P}_i^n\}_{i=1}^5$ in the frame n and the points from the previous frame are $\{\vec{P}_i^{n-1}\}_{i=1}^5$. The geometrical center of the points on the plane is

$$\vec{P}_c^n = (\sum_{i=1}^5 \vec{P}_i^n) / 5. \quad (5.7)$$

Five vectors are generated by connecting the center to individual points $\{\vec{P}_c^n \vec{P}_i^n\}_{i=1}^5$ (see Figure 6-2). The rotation angle $\bar{\theta}^n$ is defined by the average of the rotation angles of the five vectors $\{\theta_i^n\}_{i=1}^5$.

$$\theta_i^n = \cos^{-1} \left(\frac{\vec{P}_c^n \vec{P}_i^{n-1} \cdot \vec{P}_c^n \vec{P}_i^n}{\|\vec{P}_c^n \vec{P}_i^{n-1}\| \cdot \|\vec{P}_c^n \vec{P}_i^n\|} \right) \cdot \text{sign}((\vec{P}_c^n \vec{P}_i^{n-1} \times \vec{P}_c^n \vec{P}_i^n) \cdot \vec{N}_p) \quad (5.8)$$

$$\bar{\theta}^n = (\sum_{i=1}^5 \theta_i^n) / 5 \quad (5.9)$$

$$W_{rel}^n = R(\vec{N}_p, \bar{\theta}^n) \quad (5.10)$$

where $R(\vec{N}_p, \bar{\theta}^n)$ is the rotation matrix of the rotation axis $\vec{N}_p$ angle $\bar{\theta}^n$.

The operation of rotation is activated when the condition of taking defined by Equation (5.1) is satisfied. The proposed method allows all types of the common hand movement ways (no matter using fingers or wrist, or which direction the palm is facing) to be recognized and mapped to the object rotation.

5.2.3 Releasing

Releasing is the operation inverse to taking. For gesture-based interaction, an object can be released when the conditions for taking are no longer satisfied. However, the precision of releasing is affected by the problems of hand tremor and jump release. Therefore, a “soft releasing” technique is proposed which allows the virtual object to be gradually released while the precision is enhanced in the process.

The method split the threshold D_T defined in Equation (5.1) into two thresholds D_{min} and D_{max} so that a “soft releasing range” $[D_{min}, D_{max}]$ is defined for D_s . The Equation (5.1) is also modified accordingly – the object can be moved into a normal scale when it is firmly taken ($D_s < D_{min}$) and completely released when $D_s > D_{max}$, while the motion of the object is stabilized in the “soft releasing range” $[D_{min}, D_{max}]$. Specifically, a scaling factor was calculated from this motion range where,

$$\alpha = f(D_s) = \begin{cases} 0 & D_s > D_{max} \\ (D_s - D_{min}) / (D_{max} - D_{min}) & D_{min} \leq D_s \leq D_{max} \\ 1 & D_s < D_{min} \end{cases} \quad (5.11)$$

The actual transformation of the object is scaled by α so that the Equation (5.2) is modified to

$$W_{object}^n = \mathcal{F}_{slerp}(W_{object}^{n-1}, W_{rel} W_{object}^{n-1}, \alpha) \quad (0 \leq \alpha \leq 1) \quad (5.12)$$

The method allows the motion of the object to be gradually stabilized during releasing so that the problem of jump release can be minimized. The proposed method is similar to the one used for releasing in collision-based interaction (see Section 4.8.3), while the difference is that it enhances the precision according the strength of taking instead of using hand motion speed. The proposed method can be more reliably used for gesture-based releasing because the users can directly know to which extent the object has been released by observing the object motion. In contrast, when using the speed-based algorithm, it is difficult for the users to know when the object is actually completely released without seeing the collision between the fingers and the object – the object could jump away when the users remove their hands while the object is not completely released.

5.2.4 Deformation with one or two hands

In real-life interaction, local deformations can be created by fingers, while large-scale global deformation can be made with two hands, e.g., by rolling, stretching, pressing or twisting. When using gesture-based interaction, local deformations with individual fingers are difficult to implement since they inevitably involve implementing collision between the object and the fingers. It can only be alternatively done by using some sculpting tools manipulated by the whole hand.

For global deformations done with two hands, the interaction techniques proposed for collision-based interaction in Section 4.8.4 can be used when collision is replaced with gestures for activating the operations. Specifically, when the object is taken by two hands simultaneously with the taking gestures defined in Section 5.2.1, it can be stretched or compressed by moving two hands away or closer, twisted by rotating the wrists, or bended

by moving the whole hand and wrist at the same time. The extent and direction of deformation can be calculated from translation and rotation of the hands. In addition, the object can also be squeezed using one hand by further closing the fingers after it is taken.

Two examples are illustrated on how the object can be scaled (or called squeezed, compressed and stretched) by one and two hands. When one hand is used, the object can be scaled by

$$\beta_{uni} = \begin{cases} D_s/D_T & D_s < D_T \\ 1 & D_s \geq D_T \end{cases} \quad (5.13)$$

D_s is the distance that measures the strength of taking and D_T is the threshold for activation of taking, as illustrated in Section 5.2.1. The formula (5.13) is used to define deformation of elastic objects which can bounce back to its original shape after the hand is open. When the object is plastic, the scaling factor β can only decrease in one operation, while it is frozen to the value after the hand is open.

When two hands are used, the scale of deformation can be defined by tracking the distance between two hands. Specifically, suppose the positions of two hands, when the object is firstly taken, are $\overrightarrow{P_{h1}^0}, \overrightarrow{P_{h2}^0}$ and their positions in frame n are $\overrightarrow{P_{h1}^n}, \overrightarrow{P_{h2}^n}$ (see Figure 5-3). The scaling factor can be defined by

$$\beta_{bi} = \left\| \overrightarrow{P_{h2}^n P_{h2}^n} \right\| / \left\| \overrightarrow{P_{h1}^0 P_{h2}^0} \right\| \quad (5.14)$$

Figure 5-3 Scaling by two hands.

The object can be both stretched and compressed when using two hands while it can only be squeezed when using one hand. The object can either be deformed in a specific direction or in a uniform way.

5.3 User test

In this section, gesture-based interaction is compared with collision-based and real-life interaction. The research hypothesis is that *gesture-based interaction allows for higher efficiency in comparison to collision-based interaction*. It is also hypothesized that *gesture-based interaction can be performed as efficiently as real-life interaction, while the observed problems of hand tremor and jump release can be solved*.

5.3.1 Tasks

The tasks considered in this user study involve interaction with different objects placed on the desk. The tasks required using one or two hands to interact with the objects placed on the desk, including 1) moving a Rubik's cube, 2) moving a rubber ball, 3) reorienting a

pencil, 4) relocating a glass, 5) placing a book, 6) rotating a cube, and 7) stretching a rubber ball. The objects were initially lined on the desk in front of the participant (see Figure 5-4 (a)). For the first five tasks, a participant is required to take the objects one by one and place them on a piece of paper with size of 20cm x 15cm, while for the last two tasks the objects were placed on the paper before the task begins.

The real-life interaction tasks are to be compared with corresponding virtual interaction tasks which are implemented with 1:1 coordinate mapping of the real-life interaction space. The involved hand motion is above desktop within the best tracking area of Leap Motion controller with a width and depth of 40 cm and a height from 10 cm to 40 cm. Both gesture-based and collision-based virtual interaction were implemented. When interaction is in the collision mode, the hand avatar is shown, and the interaction is driven by collision-based interaction techniques proposed in Section 4.8. For gesture-based interaction, the interaction is controlled by gesture-based techniques proposed in the previous section of this chapter. Locations of two hands were indicated by two spheres instead of displaying the whole hand avatar (see Figure 5-4 (b)). The participant can automatically proceed to the next task when the current task is done.

Figure 5-4 The experiment setup. Interaction space and objects. (a) The real-life interaction setup. (b) The virtual interaction setup

5.3.2 Procedure

The facilities include a rubic cube, a pencil, a rubber ball, a glass, a notebook, a stop watch, a desktop computer and a Leap Motion controller. The participants need to do the real-life interaction tasks at first and then proceed to two rounds of collision- and gesture-based virtual interaction. In order to counterbalance of the training effect, the participants were randomly decided whether to gesture-based or collision-based first. Each round consists of the same seven interaction tasks. The participants could practice in each round until they felt ready to proceed to the formal test, where the time spent for each task was recorded. After each round of virtual interaction, the participants were required to fill in a questionnaire based on a 5-point Likert scale, where 1 = strongly disagree and 5 = strong agree, and give reasons for their scores.

- Q1: It is easy to learn and acquire the interaction skills;
- Q2: It is easy to remember the interaction techniques;
- Q3: I feel comfortable during interaction;
- Q4: The system is easy to control;
- Q5: I can do the tasks as precisely as I want.

5.3.3 Implementation details

The interaction techniques were implemented with the same platform used for collision-based interaction, as mentioned in Section 4.9. Specifically, Virtual Reality Modeling Language (VRML) and its successor Extensible 3D (X3D) were used as the programming language, and Bitmanagement BS Contact VRML/X3D was used as the viewer. BS Contact provides an interface to support input devices by extending

VRML/X3D through a DeviceSensor Node for receiving the input data from the device [109].

5.3.4 Results

14 participants were invited for the experiment and 4 of them are female. The average age is 30.2 ($sd = 3.98$). 7 of them used 3D hand tracking devices before which include Leap Motion controller, HTC Vive controller, Oculus Touch and Wii remote.

Table 5-1 Comparison on efficiency of real-life and virtual interaction.

		Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
Real-life	Mean	1.80	1.92	1.99	2.23	2.20	1.65	2.84
	Std.	0.71	0.79	0.81	0.75	0.67	0.56	0.75
Gesture-based	Mean	5.35	5.79	7.32	6.12	10.46	4.36	5.72
	Std.	2.56	2.85	4.12	3.89	6.91	1.85	3.92
Collision-based	Mean	14.04	8.05	13.51	7.01	15.56	5.69	10.68
	Dev.	12.82	4.24	8.60	3.88	14.25	2.43	6.41

As shown in Table 5-1, it takes less time on average for participants to do gesture-based interaction than collision-based interaction for all the seven tasks. The mean time of gesture-based interaction is significantly less than that of collision-based interaction for Tasks 1, 2, 3, and 7 ($p < 0.05$). The participants' subjective evaluation suggests (See Table 5-2) that gesture-based interaction is on average better than collision-based interaction in all of the evaluated aspects of the questionnaire. Results also show that gesture-based interaction is significantly easier to remember ($p \approx 0.05$), more comfortable ($p < 0.05$), and more controllable ($p < 0.05$) than collision-based interaction. As reflected by participants, they had to spend a long time on controlling their virtual hands motion while missing of

force feedback adds to its difficulty. In comparison, gesture-based interaction can shorten the time spent on adjustment when the strict collision tracking is replaced with the set of flexible recognition algorithms which can robustly detect commonly used gestures even with individual variations. In addition, by hiding the virtual hands users can concentrate more on the tasks rather than being distracted from the hand tracking results.

Table 5-2 Results of user questionnaire

Questions		Q1	Q2	Q3	Q4	Q5
Gesture-based	Mean	4.36	4.64	4.14	3.86	3.43
	Std.	0.74	0.50	0.86	0.66	1.09
Collision-based	Mean	3.86	4.29	3.29	3.21	2.86
	Std.	1.10	0.83	1.14	1.05	1.23

However, both collision- and gesture- based virtual interaction are still far less efficient than real-life interaction. Indeed, people can do real-life tasks quite fast with motor skills which they started to develop since a very young age, while most participants were still new to mid-air interaction with Leap Motion controller. It is quite challenging for participants to do virtual interaction as fast in the virtual condition without force feedbacks and gravity simulation. However, it is still believed that the efficiency of virtual interaction can improve after more practice. With reference to precision, both gesture- and collision-based interaction got a neutral score from the subjective evaluation from participants. With the many DOFs of hand motion, it is indeed difficult for people to place objects precisely in virtual tasks. Although the interaction has been enhanced by techniques proposed in Section 5.2, most participants who used the device for the first time did not get enough time to get familiar with the interaction way.

Based on the results, gesture-based interaction is chosen for further research. It is further investigated whether gesture-based interaction can be precisely used in a shape modeling environment where alternative low-DOF operations are provided, with which motion of objects should no longer be affected by the imperfection of hand motion when its translation is limited to a fixed direction or its rotation is limited to a fixed axis.

5.4 A second user test

Having achieved the results in the previous section, it is considered to study the feasibility of using gesture-based interaction for a serious modeling task. In comparison to the previous real-life interaction tasks, a shape modeling task usually involves more operations, and demands a higher precision and longer duration. The gesture-based interaction is therefore to be evaluated in aspects learnability, mental load, comfort, controllability and precision in a practical modeling task. This part of work was partially published in [126].

5.4.1 Functionality and interaction design

Shape modeling can be viewed as building a complex shape from simple primitives, in the process of which a large number of manipulation and deformation operations are involved. In order to allow a model to be precisely built, common modeling systems provide 3D widgets which allow high-DOF operations to be divided into low-DOF operations distributed on the respective handles. Following this idea, a novel 3D widget is specially designed for gesture-based interaction. As it can be seen from Figure 5-5, the widget consists of three handles and each of them controls a 1-DOF operation. A handle can be

naturally taken and released with the gesture-based interaction techniques proposed in Section 5.2.

Figure 5-5 The 3D widget for hand interaction.

The steering-wheel metaphor was implemented for a set of operations including directional scaling, uniform scaling, twisting, tapering, rotation and translation (as can be seen in Figure 5-6). One handle has 1 DOF of translation (for the operations of directional scaling, uniform scaling and translation), or 1 DOF of rotation (for the operations of twisting and rotation). It can be squeezed by one hand for the tapering operation, which is implemented by the Function (5.14) proposed in Section 5.2.4. Details on interaction with the handles can be found in the paper [126].

Figure 5-6 Constraining interaction techniques for (a) directional scaling, (b) uniform scaling, (c) twisting, (d) tapering, (e) rotation, and (f) translation.

5.4.2 Implementation

A prototype modeling system was implemented with VRML and FVRML and the Bitmanagement BS Contact was used as the viewer. The plugin developed in Section 4.9.2 was used for receiving the hand tracking data from Leap Motion controller.

Modeling pipeline

Figure 5-7 The modeling process.

In the modeling workflow, the final shape S_n is constructed from a work-piece shape S_0 through a sequence of modification operations. In each operation, a new primitive P_i is

generated for making modifications on the work-piece shape. The primitive can be deformed and manipulated by hand and finally applied to the work-piece shape tool through Boolean operations. The workflow can be seen in Figure 5-7. The shapes were defined in FVRML as implicit functions, which were written straight in the VRML code as function scripts together with the standard definitions of VRML.

Interface

PRIMITIVE	COLORS	DEFORMATION	MANIPULATION	OPERATION
box		d-scale	depart/joint	union
sphere		u-scale	rotation	subtraction
cylinder		twist	move	resolution+
cone		taper	axis-rot	resolution-
tri-prim		undo	axis-trans	bbx+
torus		reset	free	bbx-
wedge		hide handle	reset handle	bbx show/hide

Figure 5-8 Menu of the modeling prototype.

Different functionalities of the modeling application are accessible from the menu displayed on top of window, as can be seen in Figure 5-8. At the beginning of a modeling session, the type of a primitive has to be selected from the items under PRIMITIVE. Its color can also be changed to the ones available under COLORS. Different deformation operations can be chosen under the DEFORMATION item, where d-scale is short for directional-scaling, and u-scale means uniform-scaling. Different manipulation operations, including both free and constraining interaction techniques, are distinguished by the items under the MANIPULATION menu. The functionalities of Union, subtraction, increasing or decreasing resolution, and adjusting bounding box are available in the OPERATION menu.

5.4.3 Tasks

The modeling task is to build a house model (as it can be seen in Figure 5-9) from simple primitives. The primitives should be generated one by one, deformed, and positioned at the target place. The interaction task involves placing and deforming 4 primitives and 3 times of Boolean operations (union and subtraction). Both constrained interaction techniques (as proposed in the previous section) and free manipulation techniques (proposed in Section 5.2) were provided. It is expected that the model can be precisely built when using a combination of the proposed free interaction techniques and the widget-based constrained techniques. The involved gesture-based interaction techniques will be evaluated by user questionnaire and the quality of models they built.

Figure 5-9 The virtual modeling task.

5.4.4 Procedure

The participants were seated in a chair in front of a desk where the Leap Motion controller was placed on the desktop. In the beginning, the participants were showed a

tutorial which illustrates the functionalities of the modeling tool. They were required to go through the tutorial and to understand the relevant operations, after which they could proceed to build the house model, as shown in Figure 5-9. When the task was done, the participants were required to answer a questionnaire.

- Q1: It is easy to learn and acquire the interaction skills;
- Q2: It is easy to remember the interaction techniques;
- Q3: I feel comfortable during interaction;
- Q4: The system is easy to control;
- Q5: I can do the tasks as precisely as I want.
- Q6: What are the good and bad parts of the system for you to do the modeling task precisely?

The first five questions should be answered with reasons based on a 5-point Likert scale, where 1 = strongly disagree and 5 = strong agree, and the sixth question should be illustrated by participants.

5.4.5 Results

Table 5-3. Subject evaluation results for the modeling task.

Questions	Q1	Q2	Q3	Q4	Q5
Mean	4.07	4.14	3.43	3.64	3.21
Std.	0.62	0.53	0.94	0.93	1.05

The same 14 participants who did the previous experiment were invited. 4 of them are female. The average age is 30.2 (std.=3.98). Only 4 of them had shape modeling experience before.

Figure 5-10 Selected works of participants.

All of the participants successfully completed the modeling task. According to the subjective evaluation in Table 5-3, the gesture-based interaction is overall easy to learn and remember. However, in comparison to the previous user study, the average scores of gesture-based interaction decrease in all of the five aspects, among which the mental load significantly increased ($p < 0.05$) and comfortability significantly decreased ($p < 0.05$). Indeed, in the current modeling environment where more operations are involved, the difficulty of learning and remembering is accordingly increased. The prolonged interaction time introduced more arm fatigue to participants.

Although most participants agreed that the constrained interaction techniques designed based on the steering-wheel-like widget were useful, there are still half of the participants who could not make the model as precisely as they want. Selected works of participants are shown in Figure 5-10. Participants gave negative feedbacks on the problems of jump release and abrupt motion. They cannot easily release the object after rotation of wrist – the releasing movement of fingers is occluded by the palm. Although the jump release problem was alleviated to some extent algorithmically, it could not be completely eliminated due to unintentional displacement of hands during releasing. The second problem of accidental motion happens when the device misinterpret a closing gesture when the hand is in the border of the tracking area, where the quality of tracking significantly decrease. Despite that some of the participants gradually learned how to adjust their hand postures and positions to minimize occlusion during interaction, the desired perfection of interaction still could not be achieved.

5.5 Summary

In this chapter, it was investigated whether gesture-based interaction with optical hand tracking by Leap Motion controller could be used as naturally and efficiently as real-life interaction for virtual assembling and deforming tasks. A set of essential interaction techniques were designed for the basic operations with an emphasis paid on the precision of interaction. The virtual hands were avoided to be displayed in order to make users concentrate more on the objects being moved rather than hand tracking. A set of real-life interaction tasks involving manipulating and deforming virtual and real objects were

designed. Paying attention to efficiency, the users were requested in the study to perform the same operations on virtual and real objects and compared the time spent on the tasks.

According to first user study, gesture-based interaction showed more potential than collision-based methods in achieving higher efficiency with optical tracking, since less time was used to perform the gestures than colliding with virtual objects. The gesture-based interaction techniques were further explored in a shape modeling task where additional constrained interaction techniques were provided to enhance the precision of interaction. However, the idea was also not very successful since the problems of jump release and optical occlusion cannot be solved in principle.

By comparison with collision-based interaction, gesture-based interaction involves using hands in a quite similar natural way, which thus makes it still suffer from the problems of optical occlusion and intrinsic problems of human hand motion. The user studies showed negative results when trying to solve the problems algorithmically or using constraining techniques. The results lead to the conclusion that common natural interaction techniques cannot allow virtual interaction to be as efficient and as natural as real-life interaction when using optical hand tracking. The only solution remained is to design some kind of interaction technique specially for optical tracking that can completely allow the observed problems. Considering what can be modified on collision-based interaction is very limited since it intrinsically mimics real-life interaction behaviors, it is considered to design a special gesture-based interaction technique and the idea is further explored in the next chapter.

Chapter 6 Making shapes like playing music

According to the results obtained from the previous two chapters, it was concluded that gesture-based interaction was more promising for achieving higher efficiency of interaction. However, common natural gestures cannot get rid of the problems of occlusion and jump release when using optical tracking. Therefore, it is considered to come up with a new gesture specially designed for optical tracking to completely solve the observed problems. This part of work was partially published in [127].

6.1 Principles of Interaction

With the goal to completely eliminate the occlusion problem, a new gesture-based interaction technique was proposed to use hands in a way similar to how it is done when playing the Theremin – an electronic musical instrument controlled without physical contact by hands of the performer. Theremin consists of two metal antennas that sense the relative position of the Thereminist's hands and controls pitch of the notes with one hand, and volume with the other (Figure 6-1(a)).

When playing the Theremin, the hands never make any occlusion and the precision of the instrument can be tuned to each individual player including the range of the notes and volumes to be produced. It was then hypothesized that if fine music could be played by moving hands in the mid-air without any physical support, we then should be able to make fine shapes in the virtual space by moving hands in the mid-air as well. Therefore, such a kind of bimanual interaction is considered, which separates taking and moving between two hands (Figure 6-1(b)), specifically:

- The hand tracking space is divided into two unequal and non-intersecting zones: one bigger zone for tracking the dominant hand and another smaller zone for tracking the non-dominant hand.
- The dominant hand is only tracked for changing position and orientation. This can be the tracking position of the whole hand or individual fingers.
- The non-dominant hand is tracked for triggering two events – taking and releasing – as well as for continuous changing of the precision of operation. Depending on the tracking device capability, it can be done either by moving the hand up and down, as the Thereminist is doing to change the volume, or by closing/opening fingers.
- It is believed that higher efficiency can be achieved if the virtual hands are hidden since observation of the motion of the virtual objects/instruments controlled by the hands is more essential for the users than the ability to see the simulated hand. However, whenever needed, a simplified not obstructing virtual representations of the fingertips or palms still can be shown.

(a)

(b)

Figure 6-1 (a) Playing the Theremin (Moog Etherwave Standard.) (b) Theremin-style gestures proposed for 3D modeling in mid-air.

In the proposed way of interaction, both hands are still performing natural interaction ways of taking and motion. Two hands have separated motion spaces so that they will not occlude each other, as they do not interfere when the Theremin is being played. If the hands will be facing the sensor, the finger motion can be reliably tracked. There will be no jump release either because the release event, triggered by non-dominant hand, will not affect the object motion controlled by the dominant hand. Moreover, an ability to continuously change the precision of operation with another hand will allow us to solve the problem of hand jitter since the precision can be increased gradually before releasing the object.

6.2 Design of the Actual Gestures and Interaction using Leap Motion controller

The proposed bimanual interaction way allows any actual operations to be implemented on different tracking devices, while the minimal requirement is to be able to track two hands as two 3D moving points, just as in the case of Intel RealSense SR300. In the following, it is discussed how a set of basic operations – taking, positioning, releasing and deformation – can be designed with Leap Motion controller. It is also considered how these operations can be alternatively implemented on other devices such as Intel RealSense SR300 and uSens Fingo.

6.2.1 Motion space

The best tracking area (with a width of 40 cm and a height between 10 cm and 40 cm above) of Leap Motion controller is used as a space for hand motion. As shown in Figure 6-2, the whole motion space is ergonomically divided into two non-intersecting areas for two hands. The non-dominant hand has a narrower motion range, since all the involved motions are only opening and closing fingers and moving the arm vertically. The dominant hand is assigned with a larger motion for controlling the actual object/cursor motion. Since there are no any visible boundaries in the physical tracking space, visual feedback is provided to keep the hands within their tracking areas. Thus, if the dominant hand moves outside its motion area, the respective 3D cursor or object controlled by the hand will stop following it until it returns back to the tracking area. For the non-dominant hand, there is also a visual indicator of the strength of grasping which works only when the hand is located within its tracking area.

In the actual interaction, motions of two hands are kept within their respective motion areas without causing bimanual occlusion, while the arms can rest on the desk to reduce fatigue.

Figure 6-2 The interaction space of two hands.

6.2.2 Taking

The operation of taking an object is commonly performed by closing fingers towards the thumb until firm contact between the hand and the object is established. As discussed in Section 5.2.1, common taking ways can be recognized by tracking whether the minimum distance between the thumb and fingers (defined as D_s in Eq. 5.1) is below a threshold.

The major problem of tracking this gesture when using Leap Motion controller is that finger motions are inevitably occluded by palm when the whole hand is moving around (as observed in the user studies in Chapter 5). The Theremine-style interaction improves the tracking by allowing the palm of non-dominant hand to face down to the device during interaction, so that motions of fingers can be clearly visible to the device. The proposed

interaction technique can also be used with uSens Fingo, when the palm faces the camera. For devices that cannot track finger motion, e.g., Intel RealSense SR300, the user can opt for moving the non-dominant hand up and down instead while the taking strength is measured by the height of the non-dominant hand.

6.2.3 Positioning

After the non-dominant hand is closed, positioning of the object has to be controlled by motion of the dominant hand. In the following part of this section, a set of interaction techniques are designed, including 6-DOF relocation, translation and rotation based on the Theremin-style bimanual interaction.

6-DOF relocation

In real-life interaction, relocation of an object with 6 DOFs can be simultaneously controlled by free hand motion which consists of 3-DOF wrist rotation and 3-DOF whole hand translation. Similarly, relocation of the virtual object can be directly computed from translation and wrist rotation of the dominant hand. Specifically, suppose transformation of the object was W_{obj}^{n-1} in the previous frame $n-1$ and the output transformation of the dominant hand in the previous frame $n-1$ and current frame n are W_{out}^{n-1} and W_{out}^n , respectively (the details on how W_{out}^n is calculated from the actual tracked hand transformation W_{hand}^n are illustrated in Section 6.2.4). Transformation of the object in the current frame n is then calculated by

$$W_{obj}^n = W_{out}^n (W_{out}^{n-1})^{-1} W_{obj}^{n-1}. \quad (6.1)$$

The 6-DOF relocation technique allows the object to be quickly manipulated to the vicinity of the target place, while additional low-DOF translation and rotation techniques are also necessary in precise modeling for fine adjustment.

Translation

Similar to 6-DOF relocation, translation of an object can be directly mapped from translation of the dominant hand. Suppose the position of the object in the previous frame $n-1$ is $\overrightarrow{P_{obj}^{n-1}}$, and the displayed position of the dominant hand (indicated by a 3D cursor) at the previous frame $n-1$ and current frame n are $\overrightarrow{P_{disp}^{n-1}}$ and $\overrightarrow{P_{disp}^n}$, respectively (how the displayed position of the dominant is computed from actual tracked position of the dominant hand is illustrated in Section 6.2.4). The new position of the object $\overrightarrow{P_{obj}^n}$ be calculated by

$$\overrightarrow{P_{obj}^n} = \overrightarrow{P_{obj}^{n-1}} + (\overrightarrow{P_{disp}^n} - \overrightarrow{P_{disp}^{n-1}}). \quad (6.2)$$

The method can be further used for translating an object in a 2D plane or a 1D line when the displacement of the cursor controlled by the dominant hand ($\overrightarrow{P_{disp}^n} - \overrightarrow{P_{disp}^{n-1}}$) is mapped onto the defined plane or line.

$$\overrightarrow{P_{obj}^n} = \overrightarrow{P_{obj}^{n-1}} + (\overrightarrow{P_{disp}^n} - \overrightarrow{P_{disp}^{n-1}})|_{line\ or\ plane} \quad (6.3)$$

Rotation

Similar to the previous techniques, 3D rotation of the object can be directly mapped from rotation of the dominant hand. However, initial tests showed that the method could not be ergonomically used because the rotation range of the wrist is so limited that the users

always tended to over-stretch their wrists during interaction. It is therefore considered to design a rotation technique which takes the 3D displacement of the dominant hand instead as input. Suppose the object is centered at $\vec{P}_c$. The direction of the line that connects $\vec{P}_c$ and the cursor $\vec{P}_{disp}^n$ controlled by the dominant hand at frame n is

$$\vec{N}_{line}^n = \frac{\vec{P}_c P_{disp}^n}{\|\vec{P}_c P_{disp}^n\|}. \quad (6.4)$$

Rotation of the object in the current frame n is calculated from the direction change of $\vec{N}_{line}^n$ from $\vec{N}_{line}^{n-1}$. The rotation axis and angle can be thus calculated by

$$\vec{N}_{rot}^n = \vec{N}_{line}^{n-1} \times \vec{N}_{line}^n \quad (6.5a)$$

and

$$\theta_{rot}^n = \cos^{-1}(\vec{N}_{line}^{n-1} \cdot \vec{N}_{line}^n). \quad (6.5b)$$

In order to make the interaction technique more intuitive to users, the line that connects the center of the object and the 3D cursor controlled by the dominant hand is displayed. Every time after the non-dominant hand is closed, the object seems plausibly to be rotated by dragging the line that is skewed in to the object with the dominant hand.

For the case of 1-DOF rotation around a fixed axis, the rotation angle θ_{rot}^n is mapped from the displacement of the cursor motion on a principle axis, e.g., the x -axis. Suppose x_n is x -coordinate of $\vec{P}_{disp}^n$ at frame n , the rotation angle around a fixed axis can be obtained by

$$\theta_{rot}^n = C(x_n - x_{n-1}), \quad (6.6)$$

where C is a constant that defines to which extent the hand displacement is mapped to the rotation angle.

6.2.4 Releasing

As the inverse operation of taking, releasing is commonly performed by opening fingers until the object being taken is no longer in contact with the hand. However, the precision of the common releasing way is limited by the problems of hand tremor and jump release. Although the problem of jump release can be minimized when the object is released by the other hand in the Theremin-style interaction, the inevitable hand tremor is still a bottleneck for precise modeling.

In order to filter the hand tremor and to increase the overall precision of modeling, the previously fixed 1:1 coordinate mapping of actual dominant hand motion to the cursor motion in the modeling space becomes variable which is continuously controlled by the non-dominant hand. Specifically, a scale factor α is calculated from the taking strength D_s by the following formula.

$$\alpha = \begin{cases} 1 - \frac{D_s - D_{min}}{D_{max} - D_{min}} & D_{min} < D_s \leq D_{max} \\ 1 & D_s \leq D_{min} \end{cases} \quad (6.7)$$

In Eq. (6.7), D_{max} and D_{min} are two thresholds set for the taking strength D_s . The actual output transformation W_{out}^n and displayed cursor position $\overrightarrow{P_{disp}^n}$ are then updated by adding the relative transformation of the actual hand motion scaled by α .

$$W_{out}^n = \begin{cases} \mathcal{F}_{slerp}(W_{out}^{n-1}, W_{hand}^n (W_{hand}^{n-1})^{-1} W_{out}^{n-1}, \alpha) & D_s \leq D_{max} \\ W_{hand}^n & D_s > D_{max} \end{cases} \quad (6.8)$$

$$\overrightarrow{P_{disp}^n} = \begin{cases} \overrightarrow{P_{disp}^{n-1}} + \alpha (\overrightarrow{P_{hand}^n} - \overrightarrow{P_{hand}^{n-1}}) & D_s \leq D_{max} \\ \overrightarrow{P_{hand}^n} & D_s > D_{max} \end{cases} \quad (6.9)$$

According to Eq. (6.7), the largest value of α is obtained when the non-dominant hand is firmly closed ($D_s \leq D_{min}$), so that the original tracked displacement of hand motion is used for the displacement of the cursor/object. When releasing the object by slowly opening or moving up the non-dominant hand ($D_s \rightarrow D_{max}$, $D_s \geq D_{min}$), α is linearly decreasing towards zero so that big displacements of the hand result in small displacements of the cursor/object. When the non-dominant hand is fully open ($D_s > D_{max}$), motion of the object is no longer controlled by the dominant hand. W_{out}^n and $\overrightarrow{P_{disp}^n}$ are reset to the tracked motion of the dominant hand.

The values of D_{max} and D_{min} are manually set for different hand sizes in the beginning of an experiment, while they are automatically calibrated during interaction to adapt to user behaviors. Details on the adaptive algorithm are further illustrated in Section 6.2.7.

6.2.5 Deformation

In this section, a set of interaction techniques are designed for deformations commonly used in shape modeling applications, including twisting, inflating, dragging, and bending. In comparison to previous operations, they have more parameters to be interactively controlled. The challenge is how to design the interaction techniques based on the Theremin-style way where the non-dominant hand is used to control activation\deactivation of the operations and the dominant hand is used for adjusting the parameters of deformation.

For this set of deformation operations, it is firstly discussed how they can be mathematically defined and controlled by hands when the object is aligned with the Y -axis and bounded in the range of (y_{min}, y_{max}) . As it can be seen in Figure 6-3, the axis that the object is aligned with and the bounds are visualized by a 3D widget which consists of an

arrow and two round plates centered on the arrow. Finally, it is considered how an arbitrary part of the object can be deformed by relocating the widget and changing the deformation functions accordingly.

Twisting

In real-life interaction, twisting is performed by holding the object firmly with two hands while rotating either or both wrists in opposite directions. In the virtual interaction, the twisting deformation start from either of the range (y_{min}, y_{max}) and propagate to the other end. The deformation is mathematically defined by

$$\theta = \begin{cases} \theta_{top} & y \geq y_{max} \\ \frac{y_{max}-y}{y_{max}-y_{min}}\theta_{top} + \frac{y-y_{min}}{y_{max}-y_{min}}\theta_{bottom} & y_{min} < y < y_{max} \\ \theta_{bottom} & y \leq y_{min} \end{cases} \quad (6.10a)$$

$$\begin{cases} X = x\cos(\theta) - z\sin(\theta) \\ Y = y \\ Z = x\sin(\theta) + z\cos(\theta) \end{cases} \quad (6.10b)$$

In Eq. (6.10 a), θ_{bottom} and θ_{top} are the twisting angles at the two ends of (y_{min}, y_{max}) . The whole space is divided into three parts along y -axis by y_{min} and y_{max} . The space inside (y_{min}, y_{max}) is twisted by an angle that is a linear combination of θ_{top} and θ_{bottom} , while the space outside of the deformation range is rotated by θ_{top} or θ_{bottom} .

The twisting angles θ_{bottom} and θ_{top} can be interactively adjusted in a way similar to the axis-constrained rotation technique designed in Section 6.2.3. Specifically, the horizontal displacement of the cursor controlled by the dominant hand is mapped to the increment of the twisting angle (See Eq. (6.11)). In this way, the twisting angle is controlled by moving the dominant hand horizontally, when the non-dominant hand is closed.

$$\theta_{top/bottom}^n \begin{cases} \theta_{top/bottom}^{n-1} + C \cdot (x_n - x_{n-1}) & D_s \leq D_{max} \\ \theta_{top/bottom}^{n-1} & D_s > D_{max} \end{cases} \quad (6.11)$$

Figure 6-3 The interaction techniques for (a) twisting, (b) inflating, (c) dragging, and (d) bending.

In the actual interaction, only one of the angles θ_{top} and θ_{bottom} is activated for adjustment, which is indicated by the highlighted plate, as it can be seen in Figure 6-3 (a). Either end can be selected for twisting by pointing the corresponding plate with the cursor controlled by the dominant hand. Further details on how to interact with the 3D widget are illustrated in the end part of this section.

Inflating

Inflating refers to the deformation happening on an expandable structure, e.g., a balloon, when it is filled with air or gas. Consider the case of blowing a balloon made of rubber. The balloon is partially inflated in the middle part at the beginning phase, while the

deformation is gradually propagating to the end part of the object until it is fully extended. In virtual modeling, this deformation can be simulated by a function $f(y)$ defined along the y -axis which scales the shape in the direction parallel to the horizontal plane.

$$\begin{cases} X = x \cdot f(y) \\ Y = y \\ Z = z \cdot f(y) \end{cases} \quad (6.12)$$

To mimic the real-life deformation, $f(y)$ is supposed to deform the object with the largest extent β_0 at $y_0 \in (y_{min}, y_{max})$ on the y -axis, while the deformation should decay towards the bound of deformation. Therefore, $f(y)$ should meet the following conditions.

$$\begin{cases} f(y) = 1 & y \in (-\infty, y_{min}] \cup [y_{max}, +\infty) \\ f(y) = \beta_0 & y = y_0 \\ f(y_1) \leq f(y_2) & y_{min} < y_1 < y_2 < y_0 \\ f(y_1) \geq f(y_2) & y_0 < y_1 < y_2 < y_{max} \end{cases} \quad (6.13)$$

In order to make the deformation look smooth and natural, $f(y)$ is expected to be continuous in the whole domain. That is, the function should be continuous at all points in (y_{min}, y_{max}) , right continuous at y_{min} , and left continuous at y_{max} .

$$\begin{cases} \lim_{y \rightarrow y'} f(y) = f(y') & y' \in (y_{min}, y_{max}) \\ \lim_{y \rightarrow y_{min}^+} f(y) = f(y_{min}) \\ \lim_{y \rightarrow y_{max}^-} f(y) = f(y_{max}) \end{cases} \quad (6.14)$$

Any function which satisfies the conditions above can be used for $f(y)$. A quadratic function (See Eq. (6.15)) is chosen for the study.

$$f(y) = \begin{cases} 1 & y \geq y_{max} \\ \beta_0 - (\beta_0 - 1) \cdot ((y - y_0)/(y_{max} - y_0))^2 & y_0 < y < y_{max} \\ \beta_0 - (\beta_0 - 1) \cdot ((y - y_0)/(y_0 - y_{min}))^2 & y_{min} < y \leq y_0 \\ 1 & y \leq y_{min} \end{cases} \quad (6.15)$$

The inflating deformation has two parameters to be interactively controlled by hands –parameter β_0 that defines the extent of deformation and the deformation center y_0 . A

direct way to control the parameters is mapping the displacements of the dominant hand along x -axis and y -axis to the increment of β_0 and y_0 respectively. In order to better visualize the parameters, a controlling point $\vec{P} = (x_p, y_p, 0)$ is displayed whose coordinates are associated with β_0 and y_0 by

$$\begin{cases} \beta_0 = 2^{x_p} \\ y_0 = y_p \end{cases} \quad (6.16)$$

Motion of the controlling point is interactively controlled by motion of the dominant hand. Suppose the position of the controlling point at frame $n-1$ is $\vec{P}^{n-1}$ and the cursor positions controlled by the dominant hand at frame $n-1$ and n are $\vec{P}_{disp}^{n-1}$ and $\vec{P}_{disp}^n$, respectively. $\vec{P}^n$ is updated by adding the displacement of the cursor on the xy -plane to $\vec{P}^{n-1}$ when the non-dominant hand is closed.

$$\vec{P}^n = \begin{cases} \vec{P}^{n-1} + (\vec{P}_{disp}^n - \vec{P}_{disp}^{n-1})|_{xy} & D_s \leq D_{max} \\ \vec{P}^{n-1} & D_s > D_{max} \end{cases} \quad (6.17)$$

The controlling point is initially placed at origin so that the object remains undeformed ($\beta_0 = 1, y_0 = 0$). The object is inflated ($\beta_0 > 1$) when the controlling point $\vec{P}$ is relocated to the positive side of x -axis and it is deflated ($\beta_0 < 1$) when $\vec{P}$ is moved to the negative side of x -axis. The extent of deformation is reflected by the length of the bar that connects the controlling point and the axis that the object is aligned with, as it can be seen in Figure 6-3 (b).

Dragging

An elastic object can be partially deformed by firmly picking or holding its part while moving the whole hand away, when the object is globally fixed by the other hand or through

some additional forces. In 3D modeling, deformation of an object when it is dragged can be simulated by displacing the shape locally with a function $f(y)$ defined along the y -axis by

$$\begin{cases} X = x + d_x \cdot f(y) \\ Y = y \\ Z = z + d_z \cdot f(y) \end{cases} \quad (6.18)$$

The object is supposed to be dragged by $(d_x, 0, d_z)$ at y_0 where it is picked, and the deformation should decay towards the ends of the deformation range (y_{min}, y_{max}) . Therefore, $f(y)$ should meet the following conditions.

$$\begin{cases} f(y) = 0 & y \in (-\infty, y_{min}] \cup [y_{max}, +\infty) \\ f(y) = 1 & y = y_0 \\ f(y_1) \leq f(y_2) & y_{min} < y_1 < y_2 < y_0 \\ f(y_1) \geq f(y_2) & y_0 < y_1 < y_2 < y_{max} \end{cases} \quad (6.19)$$

In order to make the deformation look smooth and natural, $f(y)$ is expected to be continuous in the whole domain by meeting the conditions defined by Eq. (6.14). A quadratic function Eq. (6.20) that satisfies the conditions is used for the experiment.

$$f(y) = \begin{cases} 0 & y \geq y_{max} \\ 1 - ((y - y_0)/(y_{max} - y_0))^2 & y_0 < y < y_{max} \\ 1 - ((y - y_0)/(y_0 - y_{min}))^2 & y_{min} < y \leq y_0 \\ 0 & y \leq y_{min} \end{cases} \quad (6.20)$$

Interactive dragging can be achieved by controlling motion of the point $\vec{P} = (x_p, y_p, z_p)$ on the object being dragged with 3D motion of the dominant hand, as it can be seen in Figure 6-3 (c).

$$\begin{cases} d_x = x_p \\ y_0 = y_p \\ d_z = z_p \end{cases} \quad (6.21)$$

$$\vec{P}_n = \begin{cases} \vec{P}_{n-1} + (\vec{P}_{disp}^n - \vec{P}_{disp}^{n-1}) & D_s \leq D_{max} \\ \vec{P}_{n-1} & D_s > D_{max} \end{cases} \quad (6.22)$$

In Eq. (6.22), $\overrightarrow{P^n}$ and $\overrightarrow{P^{n-1}}$ are the positions of the controlling point $\vec{P}$ at current frame n and previous frame $n-1$, respectively.

Bending

Bending can be simulated by curving the middle part of the object bounded in (y_{min}, y_{max}) and rotating the two ends accordingly around y_{min} and y_{max} . The mathematical definition proposed in [113] is used for the test. Specifically, D_{bend} is defined by

$$\hat{y} = \begin{cases} y_{min}, & y \leq y_{min} \\ y, & y_{min} < y < y_{max} \\ y_{max}, & y \geq y_{max} \end{cases} \quad (6.23a)$$

$$\theta = k(\hat{y} - y_0) \quad (6.23b)$$

$$X=x \quad (6.23c)$$

$$Y = \begin{cases} -\sin(\theta) * \left(z - \frac{1}{k}\right) + y_0, & y_{min} < y < y_{max} \\ -\sin(\theta) * \left(z - \frac{1}{k}\right) + y_0 + \cos(\theta) * (y - y_{min}), & y \leq y_{min} \\ -\sin(\theta) * \left(z - \frac{1}{k}\right) + y_0 + \cos(\theta) * (y - y_{max}), & y \geq y_{max} \end{cases} \quad (6.23d)$$

$$Z = \begin{cases} \cos(\theta) * \left(z - \frac{1}{k}\right) + \frac{1}{k}, & y_{min} < y < y_{max} \\ \cos(\theta) * \left(z - \frac{1}{k}\right) + \frac{1}{k} + \sin(\theta) * (y - y_{min}), & y \leq y_{min} \\ \cos(\theta) * \left(z - \frac{1}{k}\right) + \frac{1}{k} + \sin(\theta) * (y - y_{max}), & y \geq y_{max} \end{cases} \quad (6.23e)$$

Since D_{bend} defined by Eq. (6.23) only deforms the object in a fixed direction of $\overrightarrow{N_{bend}} = (0, 0, 1)$, the first task is thus to find D_{bend}^* , which bends the object in any direction vertical to the axis that the object is aligned on. Bending the object in an arbitrary direction $\overrightarrow{N_{bend}^*}$ can be achieved by: 1) rotating the object by R until $\overrightarrow{N_{bend}^*}$ collides with

$\overrightarrow{N_{bend}}$, 2) deforming the object by D_{bend} , and 3) rotating the object back by R^{-1} . Therefore, D_{bend}^* is defined by

$$D_{bend}^* = R^{-1}(\overrightarrow{N_{rot}}, \theta_{rot}) D_{bend} R(\overrightarrow{N_{rot}}, \theta_{rot}), \quad (6.24)$$

where $R(\overrightarrow{N_{rot}}, \theta_{rot})$ denotes the matrix, which rotates the object around $\overrightarrow{N_{rot}}$ by θ_{rot} .

$\overrightarrow{N_{rot}}$ and θ_{rot} can be obtained by

$$\overrightarrow{N_{rot}} = \frac{\overrightarrow{N_{bend}^*} \times \overrightarrow{N_{bend}}}{\|\overrightarrow{N_{bend}^*} \times \overrightarrow{N_{bend}}\|} \quad (6.25a)$$

and

$$\theta_{rot} = \cos^{-1} \left(\frac{\overrightarrow{N_{bend}^*} \cdot \overrightarrow{N_{bend}}}{\|\overrightarrow{N_{bend}^*} \cdot \overrightarrow{N_{bend}}\|} \right). \quad (6.25b)$$

When virtual interaction is done with hands, the direction and extent of bending are expected to be simultaneously controlled in a way similar to that implemented for dragging – moving the hand away from the object increases the extent of bending while the position of the hand relative to the axis controls the direction of bending. In order to make the deformation smooth and continuous, while sudden change of bending direction and extent are prevented, the deformation parameters are mapped from the position of the controlling point $\vec{P} = (x_p, y_p, z_p)$, which is continuously moved by the dominant hand by Eq. (6.22). Specifically, suppose $\vec{P'}$ is the projection of $\vec{P}$ on the central axis. The deformation extent and direction can be visualized by the length and direction of $\overrightarrow{P'P}$, as it can be seen in Figure 6-3(d). The parameters y_0 , k , and $\overrightarrow{N_{bend}}$ that control the deformation are defined by

$$\begin{cases} y_0 = y_p \\ \vec{P'} = \vec{P} - (\vec{P} \cdot \vec{N_d}) \vec{N_d} \\ k = C \|\overrightarrow{P'P}\| \\ \overrightarrow{N_{bend}} = \frac{\overrightarrow{P'P}}{\|\overrightarrow{P'P}\|} \end{cases}, \quad (6.26)$$

where C is a constant that scales $\|\overrightarrow{P'P}\|$.

Axis and range adjustment

The methods proposed in the previous parts of this section discussed how the object can be deformed in the range (y_{min}, y_{max}) when it is aligned with the y -axis. However, the capability of the object deformation on a specific local part is also necessary in most scenarios of shape modeling. Therefore, here it is considered how arbitrary deformations can be achieved by interactive relocation of the 3D widget and modifying the functions accordingly.

The deformation range (l_{min}, l_{max}) can be adjusted by interacting with the two plates of the 3D widget. Either plate can be selected by pointing it with the cursor controlled by the dominant hand, which is implemented by tracking the collision between the plate and the ray casted from the viewing point while penetrating the cursor controlled by the hand. After the non-dominant hand is closed, the deformation range (l_{min}, l_{max}) can be adjusted by moving the selected plate by along the axis. The value of $l_{min/max}^n$ in current frame n is updated by

$$l_{min/max}^n = \begin{cases} l_{min/max}^{n-1} + (\overrightarrow{P_{disp}^n} - \overrightarrow{P_{disp}^{n-1}}) \cdot \overrightarrow{N_d} & D_s \leq D_{max} \\ l_{min/max}^{n-1} & D_s > D_{max} \end{cases} \quad (6.27)$$

The whole widget can be translated, rotated, and relocated with 6 DOFs using the techniques proposed in Section 6.2.3, as it can be seen in Figure 6-3 (c). These positioning techniques can be activated in separate modes, which are accessible from the menu displayed the right side of the window. Suppose the widget is relocated from the position $\overrightarrow{P_d} = (0, 0, 0)$ and the direction $\overrightarrow{N_d} = (0, 1, 0)$ to a new position $\overrightarrow{P_d^*}$ with the direction $\overrightarrow{N_d^*}$.

For any deformation D_d , which is one of twisting, inflating, dragging and bending defined based on $\overrightarrow{P_d}$ and $\overrightarrow{N_d}$, the task is to find D_d^* when the 3D widget is relocated to the new position $\overrightarrow{P_d^*}$ and direction $\overrightarrow{N_d^*}$. Suppose the widget is transformed by W_{wid} . D_d^* can be obtained by 1) transforming the object inversely by W_{wid}^{-1} , 2) deforming the object by D_d , and 3) transforming the object back by W_{wid} .

$$D_d^* = W_{wid} D_d W_{wid}^{-1} \quad (6.28)$$

Any W_{wid} can be calculated from

$$W_{wid} = T_{wid} R_{wid}, \quad (6.29)$$

where R_{wid} is the matrix that rotates the widget around $\overrightarrow{P_d}$ to align $\overrightarrow{N_d}$ with $\overrightarrow{N_d^*}$ and T_{wid} is the matrix that translates the object from $\overrightarrow{P_d}$ to $\overrightarrow{P_d^*}$.

6.2.6 Additional operations for interaction control

Different manipulation and deformation modes are accessible from the menu displayed on the right side of the scene. The region can be reached when the dominant hand moves to the right border of the tracking area, as shown in the Figure 6-2. When the dominant hand is in the vicinity of the menu, 2D tracking is triggered so that the cursor is always placed over the menu items regardless of occasional moves of the hand in the third dimension. The menu items can be browsed by moving the dominant hand vertically, while the precision is controlled by the non-dominant hand. In this way, the target menu item can be precisely selected while accidental selection caused by hand tremor can be prevented.

Figure 6-4 The interaction technique for scene manipulation.

During the modeling process, the scene needs to be frequently rotated in order to view the object in different directions. This operation is accessible in any interaction mode by moving up the dominant hand until the visual indicator displayed on the left bottom corner of the scene indicates its activation. After the non-dominant hand is closed, the whole scene can be rotated by moving the dominant hand in the vertical plane or dragged towards or away from the user by moving the hand away from and towards the screen, as shown in Figure 6-4. This is achieved by mapping the horizontal and vertical displacement of the cursor position controlled by the dominant hand to the rotation around the y -axis and x -axis of the scene respectively with the axis-constrained method proposed in Section 6.2.3. Displacement of the cursor along the axis vertical to the view plane is mapped to translation of the viewpoint.

6.2.7 Adaptive calibration algorithm

As discussed in the previous Section 6.2.4, D_{max} and D_{min} are two thresholds set for the taking strength D_s , which is measured by the minimum distance between the thumb and fingers. They divide the motion space into three regions $[0, D_{min}]$, (D_{min}, D_{max}) and $[D_{max}, +\infty)$, which correspond to firm taking, precision control, and releasing, respectively. In the beginning of an experiment, D_{max} and D_{min} are manually calibrated for different hand sizes to make sure that all the three regions can be easily and comfortably reached. As a user gradually acquires the motor skills for interacting with the system, he/she tends to use the consistent way he/she finds comfortable with for taking and releasing so that the values of D_s may fall in to two different small ranges in these modes. For convenience of description, suppose D_s converges to D_t^* for the taking mode and D_r^* for the releasing mode. The following two cases may happen:

- 1) D_t^* is below D_{max} ($D_t^* < D_{max}$), or
- 2) D_r^* is below D_{min} ($D_r^* > D_{min}$).

The first case introduces the problem of jump release – the object is displaced before it is completely released. The second case reduces the scale of object/cursor motion which tends to drive dominant hand out of its motion space. These problems can happen unconsciously when the user is staring at the target object/cursor instead of observing the visualizer displayed on the left bottom corner of the screen. Meanwhile, the values of D_t^* and D_r^* may also shift due to inconsistent tracking of D_s by Leap Motion when the user moves the non-dominant hand or rotates the wrist. Therefore, it is considered to design an algorithm running behind in the background to automatically calibrate the values of D_{max} and D_{min} according to changes of D_t^* and D_r^* .

During the experiment, the average of D_s is calculated for each time of taking/releasing. In order to reduce the accidental errors, the value is filtered out if the an event of taking/releasing lasts less than time T . Suppose the recoded average values of D_s in recent m times of taking and releasing are stored in $S_t = \{\overline{d_t^1}, \overline{d_t^2}, \dots, \overline{d_t^m}\}$ and $S_r = \{\overline{d_r^1}, \overline{d_r^2}, \dots, \overline{d_r^m}\}$, respectively. Ideally, the non-dominant hand should have a room for motion in $[0, D_{min}]$ and $[D_{max}, +\infty)$ so that the elements in S_t and S_r should not fall in the vicinity of D_{min} and D_{max} , respectively. The algorithm will be activated on the following conditions:

Condition I: There are more than m_1 elements of S_t in $(D_{min} - \varepsilon_t, D_{min})$.

Condition II: There are more than m_2 elements of S_r in $(D_{max}, D_{max} + \varepsilon_r)$.

The algorithm will try to increase D_{min} by Δd_t for Condition I and decrease D_{max} by Δd_r for condition II. Adjustment on D_{min} and D_{max} has to follow the following rules:

- There has to be enough motion space for precision control so that $D_{max} - D_{min} \geq \Delta D$.
- If increasing D_{min} by Δd_t will violate the first rule, the adjustment can only by executed when D_{max} can be increased by Δd_t accordingly. If decreasing D_{max} by Δd_r will violate the first rule, the adjustment can only be executed when D_{min} can be decreased by Δd_r accordingly.
- D_{max} can be increased by Δd_r only when condition II is not violated if D_{max} is adjusted to $D_{max} + \Delta d_r$. D_{min} can be decreased by Δd_r only when condition I is not violated after D_{min} is adjusted to $D_{min} - \Delta d_r$.

The details of the algorithm are described as in Figure 6-5.

```

If  $S_t$  is updated
    If  $\text{Card}(S_t)=m$  and  $\text{Count}(D_{\min} - \varepsilon_t < \overline{d}_t^l < D_{\min}|_{i=1}^m) > m_1$ 
        If  $D_{\max} - D_{\min} - \Delta d_t \geq \Delta D$ 
             $D_{\min} \leftarrow D_{\min} + \Delta d_t$ 
            Empty  $S_t$ 
        Else if  $\text{Card}(S_r)=m$  and  $\text{Count}(D_{\max} + \Delta d_t < \overline{d}_r^l < D_{\max} + \Delta d_t + \varepsilon_r|_{i=1}^m) < m_2$ 
             $D_{\min} \leftarrow D_{\min} + \Delta d_t$ 
             $D_{\max} \leftarrow D_{\max} + \Delta d_t$ 
            Empty  $S_t$ 
        End if
    End if
End if

If  $S_r$  is updated
    If  $\text{Card}(S_r)=m$  and  $\text{Count}(D_{\max} < \overline{d}_r^l < D_{\max} + \varepsilon_r|_{i=1}^m) > m_2$ 
        If  $D_{\max} - D_{\min} - \Delta d_r \geq \Delta D$ 
             $D_{\max} \leftarrow D_{\max} - \Delta d_r$ 
            Empty  $S_r$ 
        Else if  $\text{Card}(S_t)=m$  and  $\text{Count}(D_{\min} - \varepsilon_t - \Delta d_r < \overline{d}_t^l < D_{\min} - \Delta d_r|_{i=1}^m) < m_1$ 
             $D_{\min} \leftarrow D_{\min} - \Delta d_r$ 
             $D_{\min} \leftarrow D_{\min} - \Delta d_r$ 
            Empty  $S_r$ 
        End if
    End if
End if

```

Figure 6-5 The adaptive calibration algorithm.

6.3 Evaluation

Before undertaking the final user study, the previous tasks are tested with proposed interaction techniques (See Figure 6-6). It was observed that the tasks could be done more precisely and robustly than before since the occlusion problem was minimized with the new method (See the video from <https://youtu.be/sa1IjWLu03M>). After obtaining these encouraging results, it is then investigated whether the proposed mid-air interaction method could improve existing interaction of shape modeling with common 2D tracking devices.

Figure 6-6 Implementations of the Theremin-style interaction for natural interaction tasks studied in previous sections. (a) Manipulating the nut. (b) Rotating the bolt by rotating the scene. (c) and (d) are manipulating a pencil and a book.

The commercial modeling tool Autodesk Maya was used in this exercise. In Autodesk Maya, manipulation and deformation widgets were implemented for 3D tasks to be done

with a mouse. As analyzed in Section 2.2, the obstacle of using a mouse is that 3D operations have to be decomposed into a sequence of 2D operations. It is studied whether such operations can be more efficiently done with the proposed interaction technique, while the intrinsic problems of hand motion and tracking could be minimized.

6.3.1 Task

A common shape modeling case is considered which requires to /1/ Select items from a menu; /2/ Rotate the 3D scene with a work-piece object to any desired orientation; /3/ Select a tool-object and place it on, near or inside the work-piece object; /4/ Apply some CSG operations onto them (add and remove material); /5/ Perform deformations of the work-piece. The actual task (see Figure 7-7) contained the following five steps:

- In step (a), the ellipsoid had to be placed on the center of the top square face of the block while the bottom was touching the surface.
- In step (b), the three blocks had to be placed in the same horizontal plane with the ellipsoid while touching the surface. The blocks had to be individually manipulated so that the angles between each two neighbors were 120° .
- In step (c), bending had to be done at the upper half part of the first block. This required to 1) move up the deformation widget, 2) adjust the bending range and length of the line that defines the extent of deformation, and 3) to adjust the bending direction until the line controlling bending penetrated through an edge of the block.
- In step (d), the deformation widget had to be aligned with the central axis of the lower unbent part of the body block. This required to 1) align the widget with the

edge of the lower unbent part of the block, and 2) to translate the widget to the center.

- In step (e), the bounding range had to be adjusted to exactly include the unbend part of the block before twisting the object.

Figure 6-7 The shape modeling task in Autodesk Maya.

The virtual model had to be viewed from different directions by rotating the scene in order to eliminate the misalignments. The size of the model displayed on the screen could also be adjusted by dragging the scene towards and away from the user. In Autodesk Maya, the scene could be manipulated by dragging the mouse while the “ALT” key is pressed. In the mid-air interaction, the virtual scene could be manipulated at any time when moving up the non-dominant hand, as illustrated in Section 6.3. In the mid-air interaction, taking was performed by moving the fingers towards the thumb. The distance between the thumb tip and the index fingertip D was measured for activating events. The maximum and minimum

thresholds D_{max} and D_{min} (which were illustrated in Section 6.2.4) could be adjusted for different hand sizes in the experiment, and they fell into the range between 3 cm and 7 cm.

6.3.2 Apparatus

The apparatus included a desktop computer and a Leap Motion controller. The desktop computer was Alienware Auroro R5 with the Intel® Core™ i7-6700 Processor (4-Cores, 8MB Cache, Turbo Boost 2.0, up to 4.0GHz) and 32GB (4X8GB) DDR4 2133MHz SDRAM Memory. The Graphic card was NVIDIA® GeForce® GTX 1080 with 8GB GDDR5X. The monitor was 24 inches with a resolution of 1920x1200. A Leap Motion controller was connected to the desktop though a USB port. It was placed on the table facing up in front of the screen.

(a)

(b)

Figure 6-8 The setup for the user study for the (a) mid-air group and (b) Maya group.

6.3.3 Procedure

24 participants (18 male, 6 female) were invited. The average age was 25.9 ($sd = 3.7$). All of them were students from the university or colleagues in the research lab. None of them had an experience with Leap Motion controller or Autodesk Maya. They were evenly divided into two groups: to use Autodesk Maya and to use mid-air interaction. The user study setups for the two groups are shown in Figure 6-8. The maximum and minimum

thresholds of taking strength D_{max} and D_{min} were calibrated for the mid-air group. Participants were required to open and close their fingers several times until they could comfortably reach the three ranges of taking separated by the two thresholds. The objective and requirements were illustrated to the participants in the beginning. The participants were allowed to practice the task and then to perform it. The experimenter was sitting next to the participants to take videos and to give instructions. Finally, the participants were asked to fill in a questionnaire based on a 5-point Likert scale, where 1 = strongly disagree and 5 = strong agree.

- Q1: It is easy to learn and acquire the interaction skills;
- Q2: It is easy to remember the interaction techniques;
- Q3: I feel comfortable during interaction;
- Q4: The system is easy to control;
- Q5: I can do the tasks as precisely as I want.

The user study took around 30 minutes.

6.3.4 Results

All the participants successfully completed the task, and 24 pieces of videos were collected and analyzed. The results showed that the mid-air interaction (297.3 sec, $sd = 66.7$) was significantly faster than the interaction with Autodesk Maya (519 sec, $sd = 149.8$) for the task ($p \approx 0$) (Also see video in <https://youtu.be/6h296T2daUk>). By breaking down the time into individual steps, it was found that for steps (a) - (d), mid-air interaction was significantly faster than that using Maya ($p \approx 0.05$ for (b) and $p \approx 0$ for the others), but there was no significant difference for step (e). The average time and standard

error of each step when using mid-air interaction and Autodesk Maya can be seen in Figure 6-9.

Figure 6-9 Average time spent on each step of the task.

The initial proposition, explaining the time difference between the two kinds of interaction, was that mid-air interaction allowed direct manipulations or deformations in 3D space instead of separating them into a sequence of 2D operations. In order to verify it, the number of manipulations or deformation operations involved in the two interaction paradigms was counted. One operation begins with clicking the mouse button (or closing the non-dominant hand), and ends with releasing the button (or opening the non-dominant hand). The counted operations included manipulation of the scene and interacting with the scene objects. Menu selection was excluded from the counting, since they were inevitably involved but not relevant to the major interest of this user study. It was found that the average number of operations when using mid-air interaction (46, $sd = 7.6$) was significantly less ($p \approx 0$) than that when using Autodesk Maya (102.2, $sd = 20.4$). The

breakdown of the average number of operations on each step is shown in Figure 6-10. Furthermore, it was found that there was a significant positive relationship between the interaction time and operations for both interactions using Autodesk Maya ($r = 0.69$, $F_{(1,N=10)} = 9.01$, $p < 0.05$) and mid-air interaction ($r = 0.60$, $F_{(1,N=10)} = 5.58$, $p < 0.05$). Finally, the total time was divided by the total number of operations. It was found that significantly less time was needed ($p < 0.05$) for performing one operation when using Autodesk Maya (5.05, $sd = 0.76$) than that using Mid-air interaction (6.8, $sd = 2.8$).

Figure 6-10 Average number of operations on each step.

By observation of the user interaction, translation (step a), placing (step b), and alignment (step d) were indeed performed more efficiently with mid-air interaction, where decomposed sequence of 2D operations when using a mouse could be performed as integrated 3D translation or 6D positioning. For the task of bending, the designed interaction technique allowed the bending position, extent and direction to be controlled

simultaneously with 3 DOFs. In comparison, these operations had to be done with translation, rotation and bending widgets respectively when using Autodesk Maya. It further complicated the task when the user had to switch between different widgets frequently in the process. Too many interactive arrows or lines were displayed, which confused the user and resulted in a higher rate of errors. However, the twisting task only consisted of several operations such as adjusting the bounding range and rotating one end of the shape around its central axis. Therefore, similar number of operations and time were involved in both interaction paradigms, as it can be observed in the Figure 6-10.

The results of participants' subjective evaluations of the interaction techniques showed that there was no significant difference between mid-air interaction and using a mouse in aspects of learnability, mental load, comfortableness, controllability and precision, as shown in Figure 6-11. The proposed interaction technique was easy to learn (3.75, $sd = 0.97$) and remember (3.67, $sd = 0.89$). Most participants could acquire and remember the interaction technique easily, since it only involved natural ways of grasping and motion of hands. Compared to interaction using a mouse, mid-air interaction was still a new interaction way to the users which required some practice.

Figure 6-11 Results of the user questionnaire.

The results also showed that the proposed interaction technique was as controllable and precise as using a mouse (Q4 and Q5). By separating functions among two hands, the proposed interaction way effectively solved the problem of jump release and minimized occlusions. The ability of continuous changing precision of interaction allowed the participants to place objects precisely despite of hand tremor. By observation of the user interaction, their performance indeed improved significantly in comparison to the previous user studies. The participants made fewer errors when they were performing precision-controlled interaction with the mid-air interaction way. It was even more preferable than using a mouse for some 3D tasks. As reflected by some of the participants, they could not use a mouse precisely because it always failed to map the way how they wanted to interact in 3D mentally. In contrast, the mid-air interaction allowed them to work on the tasks in a more direct way, which was exactly how they planned the interaction with their hands mentally.

Finally, in aspects of comfort (Q3), a prolonged mid-air interaction caused arm fatigue, which made it less comfortable than using a mouse. However, some of the participants could also gradually learn to rest their arms on the desk, as it was actually designed, when they became familiar with the mid-air interaction.

Therefore, the obtained results proved our initial hypothesis that interaction in 3D modeling can be improved by using optical sensors. However, the naturalness had to be compromised to some extent to make sure hand motion can be reliably tracked so that the problems of hand tremor and jump release could be solved.

6.4 Discussion

By review of previous works, the problems of hand tremor and jump release were considered to be solved in three major ways including 1) providing additional visual feedbacks for users to make them grasp lightly [128], 2) optimization of the algorithm for releasing [129-131], and 3) designing additional algorithms for enhancing the stability, e.g., to dynamically change the precision according to the hand motion speed [132]. Although these relevant techniques may solve the problems to some extent when other devices such as glove-based systems are used, the problem of optical occlusion cannot be solved in principle in the algorithm level. This has been proved in the user studies conducted in Section 4.10, 5.3 and 5.4. In contrast to previous works, the bimanual interaction techniques proposed in this chapter can effectively minimize the observed problems. It has to be admitted that separating natural grasping motion among two hands is a compromise due to the poor tracking quality of Leap Motion controller. However, it was reflected in the user subjective evaluation that the technique was not very difficult to learn since two hands were

still performing natural motions. Functions were separated among two hands in accordance to the natural asymmetric way of using our hands in real life [133].

As for comparison with common interaction in shape modeling, the proposed interaction involved less number of operations compared to using the standard mouse-based interaction. However, on average it took a longer time to accomplish one operation than using a mouse, which made it not much advantageous when doing low DOF tasks. This was also confirmed by Burno et al. [134], who found that compared to mouse, gesture-based interaction was less efficient and with a higher error rate for 2D selection tasks. Therefore, according to these observations, mid-air interaction is more promising to be used for 3D modeling scenarios where arbitrary 3D interaction consists of the major part of interaction.

6.5 Summary

In this chapter, it was investigated whether 3D tasks could be performed as natural as in real life and as efficient as in common 2D-tracking based interaction with optical sensors. Inspired by the way of playing the Theremin, a new gesture-based interaction technique was proposed which separated functions among two hands: the dominant hand was used for tracking position and orientation while the non-dominant hand was used to trigger events and to continuously control precision. Based on the interaction way, a set of techniques were designed for the basic operations of taking, positioning, releasing and deformation.

A user study was conducted to compare the proposed interaction technique with the common 2D-tracking based interaction. According to the subjective evaluation, no significant differences were observed between mid-air interaction and using a standard mouse in aspects of learnability, mental load, comfort, controllability and precision. The

specific modeling task could even be more quickly accomplished with the proposed interaction technique. The obtained results proved usefulness of existing optical sensors in precise shape modeling and virtual assembling tasks.

Chapter 7 Conclusion

The thesis investigated whether interaction in 3D modeling can be improved by using affordable optical sensors, so that 3D tasks can be performed in a way similar to interaction in real life and as efficiently as using common 2D-tracking based interaction while still minimizing the intrinsic problems of precise hand manipulations. In order to explore this idea, both natural collision and gesture-based interaction were implemented and compared with real-life interaction. Leap motion controller was selected as the major testing device for the research, since it showed better performance over other devices.

The results showed that using common collision and gesture-based interaction techniques did not allow the desired objective to be achieved. While gesture-based interaction showed better efficiency, it still was not completely free of the problems of occlusion and jump release. Finally, a Theremin-style bimanual gesture-based interaction technique for optical sensors was proposed which minimized the problems of occlusion and jump release. The technique separates function among two hands and assigns separated motion spaces to them. It was proved that it is more precise and robust than previous common techniques when used with optical tracking. In the user study on comparing the technique with the common 2D-tracking based interaction, the subjective evaluation suggested the interaction technique could be used as controllably and precisely as common 2D-tracking based interaction. It was also found that the technique can outperform 2D-tracking based interaction for some 3D modeling tasks. The results proved usefulness of existing optical sensors in precise 3D modeling tasks. It also provides designers with insights on how to use affordable optical hand tracking to do precise virtual interaction tasks.

In addition, incremental research novelty has also been achieved with the proposed nature collision and gesture-based interaction techniques. The collision-based techniques allow hand interaction to be more precisely and reliably done than using the standard interaction engine provided by Leap Motion. The gesture-based interaction techniques can recognize common gestures with variations by only a few algorithms while the precision was controlled dynamically during releasing the objects. The proposed techniques can be used for implementing natural user interfaces on different platforms. They can be more reliably used with a further hand tracking system with better precision.

The results of the thesis were published in 5 conference papers and 1 journal paper:

- C1. Jian Cui and Alexei Sourin, “Interactive Shape Modeling Using Leap Motion Controller.” SIGGRAPH Asia 2017 Technical Briefs, ACM, 2017.
- C2. Jian Cui, Arjan Kuijper, Dieter W. Fellner, and Alexei Sourin, “Understanding People's Mental Models of Mid-Air Interaction for Virtual Assembly and Shape Modeling.” In Proceedings of the 29th International Conference on Computer Animation and Social Agents (CASA '16), ACM, 2016.
- C3. Jian Cui, Dieter W Fellner, Arjan Kuijper, and Alexei Sourin, “Mid-Air Gestures for Virtual Modeling with Leap Motion,” in Distributed, Ambient and Pervasive Interactions: 4th International Conference, DAPI 2016, Held as Part of HCI International, 2016.
- C4. Jian Cui, Arjan Kuijper, Dieter W. Fellner, and Alexei Sourin, “Exploration of natural free-hand interaction for shape modeling using leap motion controller.” 2016 International Conference on Cyberworlds (CW), 2016.

- C5. Jian Cui and Alexei Sourin, "Feasibility Study on Free Hand Geometric Modelling Using Leap Motion in VRML/X3D," 2014 International Conference on Cyberworlds (CW), 2014.
- J1. Jian Cui, Alexei Sourin, "Mid-air Interaction with Optical Tracking for 3D Modeling." Computers & Graphics, Volume 74, Pages 1-11, 2018, ISSN 0097-8493, <https://doi.org/10.1016/j.cag.2018.04.004>.

Chapter 8 Future Works

The thesis investigated whether virtual interaction with optical hand tracking can be performed as natural as real-life interaction and as efficient as using 2D tracking devices for 3D modeling. Natural interaction ways were initially designed and implemented with both collision-based and gesture-based interaction techniques. However, the results showed that natural hand motion without any constraints can never be reliably tracked by one optical sensor due to inevitable occlusion. Therefore, a gesture-based interaction technique which adapts to the tracking capabilities of optical sensors was proposed and it finally allowed the objective to be achieved. Following the idea, future works can be continued in two directions.

In the first direction, it is considered to design such a special interaction technique that can be reliably tracked by optical sensors for collision-based interaction. Although the results from the thesis suggested that collision-based interaction was less efficient than gesture-based interaction for basic manipulation and deformation tasks, it can compensate the capabilities of gesture-based interaction in simulating freeform deformation with whole hand input such as clay modeling. In the second direction, the quality of hand tracking is considered to be improved by using multiple optical sensors to minimize occlusion. It is expected that hand motion involved in natural real-life interaction ways can be finally reliably tracked in the improved tracking condition. These future directions are further discussed in the following sections.

8.1 Separating DOFs of the virtual hand among two hands

Collision-based methods define interaction by implementing collision between the virtual objects and virtual hands. The major problem is that the motion of a virtual hand cannot be precisely controlled with one hand, because the finger motion cannot be reliably tracked when the whole hand is moving. Inspired by the Theremin-style bimanual interaction technique, collision-based interaction should also be more reliably tracked when motion of the virtual hand is separated and controlled by two hands. Specifically, the global motion of the virtual hand is controlled by the dominant hand, while the local finger motion is controlled by the non-dominant hand. In the actual interaction, the non-dominant hand can face to the device all the time so that finger motion will be reliably tracked with minimal occlusion.

The proposed interaction technique can be more reliably tracked by optical sensors than common collision-based interaction ways. Actually, the idea was also explored for application of multi-finger haptic interfaces [135], where the haptic feedbacks of one virtual hand were separated among fingers and whole palm and applied on two hands. Their successful user studies motivate research in this direction. Indeed, controlling the motion of one virtual hand with two real hands is a case that users never experienced in real life. It requires users to develop some new hand motor skills to master such an interaction way. Therefore, research should be conducted on investigating whether the virtual hand motion in proposed interaction way can be controlled precisely in the mid-air environment without any force feedbacks.

8.2 Multiple sensors setup

Efforts should be made in the direction to explore the best tracking capability that can be achieved with optical systems. Since occlusion is the major problem and it cannot be solved in principle with one optical sensor, a direct solution is to set up a system with multiple affordable optical sensors, which are located in different positions and orientations. This idea was initially explored in [136], where a system with two Leap Motion controllers was set up in order to track hand motion for the task of robotic manipulation. However, they only implemented a grasping gesture, which was too simple to reflect many dexterous ways of how hands are used in real-life interaction tasks. Therefore, natural interaction ways should be classified and it should be understood to which extent these natural interaction ways can be reliably tracked if multiple optical sensors are available.

The following tasks need to be considered in order to explore this idea:

- Select a platform and make the hand tracking data from different sensors available on the platform.
- Investigate how to set up two or multiple sensors to get the best tracking quality for different gestures.
- Design data fusion algorithms and output the finally skeleton hand model by using data from different sensors.

The first step has been achieved with Leap Motion controllers. Because the current SDK of the device only allows one controller to be used on one computer, the initial setup included two computers which were connected by Leap Motion controllers respectively. Connection between two computers was implemented with BS Collaborate package, which allowed data communication between two computers through local area networks.

For the second step, different setups should be explored for different interaction purposes. A simple way that has been explored is to line up the two sensors on the table with some distance to enlarge the tracking area. However, it is yet still difficult to answer the questions on *how should the sensors be set to minimize two-hand occlusion? How many sensors are enough to totally eliminate occlusion? What is the best setup with a given number of sensors to track hand motion in a specific interaction task?*

As for the third step, the data fusion only needs to be implemented for finger motion data since motion of palm can already be reliably tracked when hand is moving. The initial idea is to vote the data sent from each sensor according to the predication on whether the data is actually tracked by the sensor or interpolated algorithmically for occlusion. Specifically, the prediction can be done by casting rays from the sensor to the finger and calculating to which extent the rays are blocked in the middle by palm or other fingers.

Based on the future improvements on the tracking capability of using optical sensors, real-life natural hand motion ways, which cannot be reliably tracked now, should be re-evaluated.

References

- [1] Taylor, J., et al., Articulated distance fields for ultra-fast tracking of hands interacting. *ACM Trans. Graph.*, 2017. 36(6): p. 1-12.
- [2] Ge, L., et al. Robust 3D Hand Pose Estimation in Single Depth Images: From Single-View CNN to Multi-View CNNs. in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [3] Markussen, A., M.R. Jakobsen, and K. Hornbæk. Vulture: a mid-air word-gesture keyboard. in *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*. 2014. ACM.
- [4] Nancel, M., et al., Mid-air pan-and-zoom on wall-sized displays, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2011, ACM: Vancouver, BC, Canada. p. 177-186.
- [5] Ataş, M., Hand Tremor Based Biometric Recognition Using Leap Motion Device. *IEEE Access*, 2017. 5: p. 23320-23326.
- [6] Leap Motion Gallery. Available from: <https://gallery.leapmotion.com>. Date accessed 28. Jan. 2019.
- [7] Feix, T., et al., The GRASP Taxonomy of Human Grasp Types. *IEEE Transactions on Human-Machine Systems*, 2016. 46(1): p. 66-77.
- [8] Corey, S. Definition of Gross & Fine Motor Skills. Available from: <http://sciencing.com/definition-gross-fine-motor-skills-6778426.html>. Date accessed 28. Jan. 2019.
- [9] Bullock, I.M., T. Feix, and A.M. Dollar. Human precision manipulation workspace: Effects of object size and number of fingers used. in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2015.
- [10] Gloumakov, Y., et al. Object stability during human precision fingertip manipulation. in *2016 IEEE Haptics Symposium (HAPTICS)*. 2016.
- [11] Sturman, M.M., D.E. Vaillancourt, and D.M. Corcos, Effects of Aging on the Regularity of Physiological Tremor. *Journal of Neurophysiology*, 2005. 93(6): p. 3064-3074.
- [12] Burkhard, P.R., J.W. Langston, and J.W. Tetrud, Voluntarily simulated tremor in normal subjects. *Neurophysiologie Clinique/Clinical Neurophysiology*, 2002. 32(2): p. 119-126.
- [13] Bachratý, M. and M. Žalman, 2D Position Measurement with optical laser mouse sensor. 2010.
- [14] Conner, B.D., et al., Three-dimensional widgets, in *Proceedings of the 1992 symposium on Interactive 3D graphics*. 1992, ACM: Cambridge, Massachusetts, USA. p. 183-188.
- [15] Stuerzlinger, Y.J.Z.D.S.W. Comparison of multiple 3D rotation methods. in *Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS)*. 19-21 Sept. 2011. Ottawa, ON, Canada: IEEE.
- [16] Oh, J.-Y. and W. Stuerzlinger, Moving objects with 2D input devices in CAD systems and Desktop Virtual Environments, in *Proceedings of Graphics Interface 2005*. 2005, Canadian Human-Computer Communications Society: Victoria, British Columbia. p. 195-202.
- [17] Teather, R.J. and W. Stuerzlinger, Guidelines for 3D positioning techniques, in *Proceedings of the 2007 conference on Future Play*. 2007, ACM: Toronto, Canada. p. 61-68.
- [18] Foxlin, G.W.a.E., Motion tracking: no silver bullet, but a respectable arsenal. *IEEE Computer Graphics and Applications*, 2002. 22(6): p. 24-38.
- [19] Wikipedia. Wii Remote. 2006 Nov. 19. Available from: https://en.wikipedia.org/wiki/Wii_Remote. Date accessed 28. Jan. 2019.
- [20] Sinclair, B. Sony reveals what makes PlayStation Move tick. 2010 Mar 11. Available from: <https://web.archive.org/web/20110817023855/http://gdc.gamespot.com/story/6253435/sony-reveals-what-makes-playstation-move-tick>. Date accessed 28. Jan. 2019.

- [21] Lighthouse tracking examined. 2016 May 25. Available from: <http://doc-ok.org/?p=1478>. Date accessed 28. Jan. 2019.
- [22] Borges, M., Andrew Colquhoun Symington, Brian Coltin, Trey Smith and Rodrigo Ventura. HTC: Analysis and Accuracy Improvement. 2018.
- [23] Luckett, E., A Quantitative Evaluation of the HTC Vive for Virtual Reality Research, in Computer and Information Science. 2018, The University of Mississippi.
- [24] Adrián, B., et al., Comparison of Oculus Rift and HTC Vive: Feasibility for Virtual Reality-Based Exploration, Navigation, Exergaming, and Rehabilitation. *Games for Health Journal*, 2018. 7(3): p. 151-156.
- [25] Windolf, M., N. Götzen, and M. Morlock, Systematic accuracy and precision analysis of video motion capturing systems—exemplified on the Vicon-460 system. *Journal of Biomechanics*, 2008. 41(12): p. 2776-2780.
- [26] Aurand, A.M., J.S. Dufour, and W.S. Marras, Accuracy map of an optical motion capture system with 42 or 21 cameras in a large measurement volume. *Journal of Biomechanics*, 2017. 58(Supplement C): p. 237-240.
- [27] OptiTrack. OptiTrack - Prime 41. Available from: <https://optitrack.com/products/prime-41/>. Date accessed 28. Jan. 2019.
- [28] Sun, X., et al. Cascaded hand pose regression. in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015.
- [29] Tompson, J., et al., Real-Time Continuous Pose Recovery of Human Hands Using Convolutional Networks. *ACM Trans. Graph.*, 2014. 33(5): p. 1-10.
- [30] Tkach, A., et al., Online generative model personalization for hand tracking. *ACM Trans. Graph.*, 2017. 36(6): p. 1-11.
- [31] Leap Motion. Available from: <https://www.leapmotion.com/>. Date accessed 28. Jan. 2019.
- [32] uSens Fingo. Available from: <https://www.usens.com/fingo>. Date accessed 28. Jan. 2019.
- [33] BlasterX Senz3D. Available from: <https://sg.creative.com/p/peripherals/blasterx-senz3d>.
- [34] Weichert, F., et al., Analysis of the Accuracy and Robustness of the Leap Motion Controller. *Sensors*, 2013. 13(5): p. 6380.
- [35] Guna, J., et al., An Analysis of the Precision and Reliability of the Leap Motion Sensor and Its Suitability for Static and Dynamic Tracking. *Sensors*, 2014. 14(2): p. 3702.
- [36] Hornsey, R.L. and P.B. Hibbard, Evaluation of the accuracy of the Leap Motion controller for measurements of grip aperture, in Proceedings of the 12th European Conference on Visual Media Production. 2015, ACM: London, United Kingdom. p. 1-1.
- [37] Dipietro, L., A.M. Sabatini, and P. Dario, A Survey of Glove-Based Systems and Their Applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 2008. 38(4): p. 461-482.
- [38] CyberGlove Systems. Available from: <http://www.cyberglovesystems.com/>. Date accessed 28. Jan. 2019.
- [39] HumanGLove. Available from: <http://www.hmw.it/en/humanglove.html>. Date accessed 28. Jan. 2019.
- [40] 5DT Data Gloves. Available from: <http://www.5dt.com/data-gloves/>. Date accessed 28. Jan. 2019.
- [41] PINCH Glove as VR Input Device. Available from: <https://www.evl.uic.edu/entry.php?id=1777>. Date accessed 28. Jan. 2019.
- [42] Pinch Glove. Available from: <http://www.ultimate3dheaven.com/pinchglove.html>. Date accessed 28. Jan. 2019.
- [43] VMG Lite. Available from: <http://www.virtualmotionlabs.com/vr-gloves/vr-glove/>. Date accessed 28. Jan. 2019.
- [44] Manus VR. Available from: <https://manus-vr.com/>. Date accessed 28. Jan. 2019.
- [45] Kessler, G.D., L.F. Hodges, and N. Walker, Evaluation of the CyberGlove as a whole-hand input device. *ACM Trans. Comput.-Hum. Interact.*, 1995. 2(4): p. 263-283.

- [46] Wang, Y. and M. Neff, Data-driven glove calibration for hand motion capture, in Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation. 2013, ACM: Anaheim, California. p. 15-24.
- [47] Kahlesz, F., G. Zachmann, and R. Klein. 'Visual-fidelity' dataglove calibration. in Proceedings Computer Graphics International, 2004. 2004.
- [48] Steffen, J., J. Maycock, and H. Ritter, Robust Dataglove Mapping for Recording Human Hand Postures, in Intelligent Robotics and Applications: 4th International Conference, ICIRA 2011, Aachen, Germany, December 6-8, 2011, Proceedings, Part II, S. Jeschke, H. Liu, and D. Schilberg, Editors. 2011, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 34-45.
- [49] Jin, S., Y. Li, and W. Chen. A novel dataglove calibration method. in 2010 5th International Conference on Computer Science & Education. 2010.
- [50] Te-Shun Chou, A.G., Dave Knott. Hand-Eye: A Vision-Based Approach to Data Glove Calibration. in Proceedings of Human Interface Technologies. 2000.
- [51] Borst, C. and A. Indugula, A spring model for whole-hand virtual grasping. Presence, 2006. 15(1): p. 47-61.
- [52] Vikash Kumar, E.T. MuJoCo HAPTIX: A virtual reality system for hand manipulation. in Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on. 3-5 Nov. 2015. Seoul, South Korea: IEEE.
- [53] Todorov, E., T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2012.
- [54] Talvas, A., et al., Aggregate Constraints for Virtual Manipulation with Soft Fingers. IEEE Transactions on Visualization and Computer Graphics, 2015. 21(4): p. 452-461.
- [55] Jan Jacobs, B.F. A soft hand model for physically-based manipulation of virtual objects. in Virtual Reality Conference (VR), 2011 IEEE. 2011. Singapore, Singapore.
- [56] McNeely, W.A., K.D. Puterbaugh, and J.J. Troy. Six degree-of-freedom haptic rendering using voxel sampling. in ACM SIGGRAPH 2005 Courses. 2005. ACM.
- [57] Kijimam, R. and M. Hirose. Representative Spherical Plane Method and Comsposition of Object Manipulation Methods. in Virtual Reality Annual International Symposium. 1996. IEEE Computer Society.
- [58] Boulic, R., S. Rezzonico, and D. Thalmann. Multi-finger manipulation of virtual objects. in Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST). 1996. Citeseer.
- [59] Holz, D., et al., Multi-contact grasp interaction for virtual environments. Journal of Virtual Reality and Broadcasting, 2008. 5(7): p. 1860-2037.
- [60] Sato, M., V. Savchenko, and R. Ohbuchi. 3D freeform design: interactive shape deformations by the use of CyberGlove. in Cyberworlds, 2004 International Conference on. 2004. IEEE.
- [61] Fuge, M., et al., Conceptual design and modification of freeform surfaces using dual shape representations in augmented reality environments. Computer-Aided Design, 2012. 44(10): p. 1020-1032.
- [62] Shieh, M.-D. and C.-C. Yang, Designing Product Forms Using a Virtual Hand and Deformable Models. 2006(42578): p. 823-830.
- [63] Quek, F., et al., Multimodal human discourse: gesture and speech. ACM Trans. Comput.-Hum. Interact., 2002. 9(3): p. 171-193.
- [64] Schraefel, M.K.a.m.c., A Taxonomy of Gestures in Human Computer Interactions. 2005.
- [65] Pavlovic, V.I., R. Sharma, and T.S. Huang, Visual interpretation of hand gestures for human-computer interaction: A review. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1997. 19(7): p. 677-695.
- [66] Bobick, A. and J. Davis. Real-time recognition of activity using temporal templates. in Applications of Computer Vision, 1996. WACV '96., Proceedings 3rd IEEE Workshop on. 1996.
- [67] Kyugon, C. and S.M. Dunn, Learning shape classes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1994. 16(9): p. 882-888.

- [68] Hyung-Ji, L. and C. Jae-Ho. Hand gesture recognition using orientation histogram. in TENCON 99. Proceedings of the IEEE Region 10 Conference. 1999.
- [69] Kuno, Y., et al. Vision-based human interface with user-centered frame. in Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on. 1994.
- [70] Ren, Z., J. Yuan, and Z. Zhang, Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera, in Proceedings of the 19th ACM international conference on Multimedia. 2011, ACM: Scottsdale, Arizona, USA. p. 1093-1096.
- [71] Ren, Z., J. Meng, and J. Yuan. Depth camera based hand gesture recognition and its applications in Human-Computer-Interaction. in 2011 8th International Conference on Information, Communications & Signal Processing. 2011.
- [72] Sharp, T., et al., Accurate, Robust, and Flexible Real-time Hand Tracking, in Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. 2015, ACM: Seoul, Republic of Korea. p. 3633-3642.
- [73] Tagliasacchi, A., et al., Robust articulated-ICP for real-time hand tracking, in Proceedings of the Eurographics Symposium on Geometry Processing. 2015, Eurographics Association: Graz, Austria. p. 101-114.
- [74] Tkach, A., M. Pauly, and A. Tagliasacchi, Sphere-meshes for real-time hand modeling and tracking. ACM Trans. Graph., 2016. 35(6): p. 1-11.
- [75] Sridhar, S., et al. Real-Time Joint Tracking of a Hand Manipulating an Object from RGB-D Input. in Computer Vision – ECCV 2016. 2016. Cham: Springer International Publishing.
- [76] Nickel, K. and R. Stiefelhagen. Pointing gesture recognition based on 3D-tracking of face, hands and head orientation. in Proceedings of the 5th international conference on Multimodal interfaces. 2003. ACM.
- [77] Segen, J. and S. Kumar. Gesture VR: Vision-based 3D hand interace for spatial interaction. in Proceedings of the sixth ACM international conference on Multimedia. 1998. ACM.
- [78] O'Hagan, R. and A. Zelinsky, Visual Gesture Interfaces for Virtual Environments, in Proceedings of the First Australasian User Interface Conference. 2000, IEEE Computer Society. p. 73.
- [79] Wang, R., et al., 6D hands: markerless hand-tracking for computer aided design, in Proceedings of the 24th annual ACM symposium on User interface software and technology. 2011, ACM: Santa Barbara, California, USA. p. 549-558.
- [80] Guiard, Y., Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. Journal of motor behavior, 1987. 19(4): p. 486-517.
- [81] Bier, E.A., et al., A taxonomy of see-through tools, in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 1994, ACM: Boston, Massachusetts, USA. p. 358-364.
- [82] Mapes, D.P. and J.M. Moshell, A Two-Handed Interface for Object Manipulation in Virtual Environments. Presence: Teleoperators and Virtual Environments, 1995. 4(4): p. 403-416.
- [83] Cutler, L.D., B. Fröhlich, and P. Hanrahan. Two-handed direct manipulation on the responsive workbench. in Proceedings of the 1997 symposium on Interactive 3D graphics. 1997. ACM.
- [84] Mine, M., Working in a virtual world: Interaction techniques used in the chapel hill immersive modeling program. University of North Carolina, 1996.
- [85] Hinckley, K., et al. Passive real-world interface props for neurosurgical visualization. in Proceedings of the SIGCHI conference on Human factors in computing systems. 1994. ACM.
- [86] Hinckley, K., et al., Two-handed virtual manipulation. ACM Trans. Comput.-Hum. Interact., 1998. 5(3): p. 260-302.
- [87] Llamas, I., et al., Twister: a space-warp operator for the two-handed editing of 3D shapes, in ACM SIGGRAPH 2003 Papers. 2003, ACM: San Diego, California. p. 663-668.
- [88] Song, P., et al. A handle bar metaphor for virtual object manipulation with mid-air interaction. in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 2012. ACM.

- [89] Wang, R., S. Paris, and J. Popović. 6D hands: markerless hand-tracking for computer aided design. in Proceedings of the 24th annual ACM symposium on User interface software and technology. 2011. ACM.
- [90] Schlattmann, M. and R. Klein, Efficient Bimanual Symmetric 3D Manipulation for Bare-Handed Interaction. JVRB - Journal of Virtual Reality and Broadcasting, 2010. 7(2010)(8).
- [91] Murugappan, S., H. Liu, and K. Ramani, Shape-It-Up: Hand gesture based creative expression of 3D shapes using intelligent generalized cylinders. Computer-Aided Design, 2013. 45(2): p. 277-287.
- [92] De Araújo, B.R., et al., Mockup Builder: 3D modeling on and above the surface. Computers & Graphics, 2013. 37(3): p. 165-178.
- [93] Kang, J., et al., Instant 3D design concept generation and visualization by real-time hand gesture recognition. Computers in Industry, 2013. 64(7): p. 785-797.
- [94] Leap Motion App Store | Sculpting. Available from: <https://apps.leapmotion.com/apps/sculpting/windows>. Date accessed 28. Jan. 2019.
- [95] Pinch Draw. Available from: <https://gallery.leapmotion.com/pinch-draw/>.
- [96] Pinch Move. Available from: <https://gallery.leapmotion.com/pinch-move/>.
- [97] Form and Function 3D. Available from: <https://apps.leapmotion.com/apps/form-and-function-3d/windows>.
- [98] Taylor, J., et al., Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. ACM Trans. Graph., 2016. 35(4): p. 1-12.
- [99] Colgan, A. How Does the Leap Motion Controller work? 2014; Available from: <http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/>.
- [100] L., N. Archived - Introducing the Intel® RealSense™ Camera SR300. 2016 22 March; Available from: <https://software.intel.com/en-us/articles/introducing-the-intel-realsense-camera-sr300>.
- [101] Augmented and Virtual Reality Tracking Technology from uSens Now Needs Only One Stereo Camera Module. Available from: <https://www.usens.com/press/one-stereo-camera/>.
- [102] uSens Makes its Color Fingo Virtual and Augmented Reality Tracking Kit Available for Pre-Order. Available from: <https://www.usens.com/press/devkit>.
- [103] Intel® RealSense™ Camera SR300 Product Datasheet. Available from: https://www.mouser.com/pdfdocs/intel_realsense_camera_sr300.pdf.
- [104] API Overview. Available from: https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Overview.html. Date accessed 28. Jan. 2019.
- [105] NVIDIA PhysX Available from: <https://developer.nvidia.com/physx-sdk>. Date accessed 28. Jan. 2019.
- [106] Bullet Physics. Available from: <http://bulletphysics.org/wordpress/>. Date accessed 28. Jan. 2019.
- [107] open dynamics engine. Available from: <http://www.ode.org>. Date accessed 28. Jan. 2019.
- [108] Rigidbody — Unity Documentation. Available from: <https://docs.unity3d.com/ScriptReference/Rigidbody.html>. Date accessed 28. Jan. 2019.
- [109] Cui, J. and A. Sourin. Feasibility Study on Free Hand Geometric Modelling Using Leap Motion in VRML/X3D. in 2014 International Conference on Cyberworlds. 2014.
- [110] Zachmann, G. and A. Rettig. Natural and robust interaction in virtual assembly simulation. in Eighth ISPE International Conference on Concurrent Engineering: Research and Applications (ISPE/CE2001). 2001. Citeseer.
- [111] Advanced Collision Detection, NVIDIA PhysX SDK 3.4.0 Documentation. Available from: <http://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/Manual/AdvancedCollisionDetection.html>. Date accessed 28. Jan. 2019.
- [112] Sato, M., V. Savchenko, and R. Ohbuchi. 3D freeform design: interactive shape deformations by the use of CyberGlove. in 2004 International Conference on Cyberworlds. 2004.

- [113] Barr, A.H., Global and local deformations of solid primitives. SIGGRAPH Comput. Graph., 1984. 18(3): p. 21-30.
- [114] Arata, H., et al. Free-form shape modeling by 3D cellular automata. in Shape Modeling and Applications, 1999. Proceedings. Shape Modeling International'99. International Conference on. 1999. IEEE.
- [115] Dewaele, G. and M.-P. Cani, Interactive global and local deformations for virtual clay. Graphical Models, 2004. 66(6): p. 352-369.
- [116] Unity Assets for Leap Motion Orion Beta. Available from: <https://developer.leapmotion.com/unity/#116>. Date accessed 28. Jan. 2019.
- [117] Blocks - Leap Motion Gallery. Available from: <https://gallery.leapmotion.com/blocks/>. Date accessed 28. Jan. 2019.
- [118] Korzen, D. DefKit — unity Asset Store. Available from: <https://assetstore.unity.com/packages/tools/physics/defkit-50767>. Date accessed 28. Jan. 2019.
- [119] Frees, S., G.D. Kessler, and E. Kay, PRISM interaction for enhancing control in immersive virtual environments. ACM Trans. Comput.-Hum. Interact., 2007. 14(1): p. 2.
- [120] Stanculescu, L., R. Chaine, and M.-P. Cani, Freestyle: Sculpting meshes with self-adaptive topology. Computers & Graphics, 2011. 35(3): p. 614-622.
- [121] FVRML/FX3D: Function-based extension of VRML and X3D (version 5.0). Available from: <http://www.ntu.edu.sg/home/assourin/FVRML.htm>. Date accessed 28. Jan. 2019.
- [122] Pasko, A., et al., Function representation in geometric modeling: concepts, implementation and applications. The Visual Computer, 1995. 11(8): p. 429-446.
- [123] Lai, D. and A. Sourin. Interactive Visualization of Mathematics in 3D Web. in 2012 International Conference on Cyberworlds. 2012.
- [124] Garbaya, S. and U. Zaldivar-Colado, The affect of contact force sensations on user performance in virtual assembly tasks. Virtual Reality, 2007. 11(4): p. 287-299.
- [125] Cui, J., et al., Mid-Air Gestures for Virtual Modeling with Leap Motion, in Distributed, Ambient and Pervasive Interactions: 4th International Conference, DAPI 2016, Held as Part of HCI International 2016, Toronto, ON, Canada, July 17-22, 2016, Proceedings, N. Streitz and P. Markopoulos, Editors. 2016, Springer International Publishing: Cham. p. 221-230.
- [126] Cui, J., A. Kuijper, and A. Sourin. Exploration of Natural Free-Hand Interaction for Shape Modeling Using Leap Motion Controller. in 2016 International Conference on Cyberworlds (CW). 2016.
- [127] Cui, J. and A. Sourin, Interactive shape modeling using leap motion controller, in SIGGRAPH Asia 2017 Technical Briefs. 2017, ACM: Bangkok, Thailand. p. 1-4.
- [128] Borst, M.P.C.W., Design and Evaluation of Visual Interpenetration Cues in Virtual Grasping. IEEE Transactions on Visualization and Computer Graphics, 2016. 22.
- [129] Prachyabrued, M. and C.W. Borst, Virtual grasp release method and evaluation. International Journal of Human-Computer Studies, 2012. 70(11): p. 828-848.
- [130] Osawa, N., Enhanced Hand Manipulation Methods for Efficient and Precise Positioning and Release of Virtual Objects. IEICE Transactions on Information and Systems, 2008. E91.D(10): p. 2503-2513.
- [131] Osawa, N., Automatic adjustments for efficient and precise positioning and release of virtual objects, in Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications. 2006, ACM: Hong Kong, China. p. 121-128.
- [132] Frees, S., G.D. Kessler, and E. Kay, PRISM interaction for enhancing control in immersive virtual environments. ACM Transactions on Computer-Human Interaction (TOCHI), 2007. 14(1): p. 2.
- [133] Guiard, Y., Asymmetric Division of Labor in Human Skilled Bimanual Action. Journal of Motor Behavior, 1987. 19(4): p. 486-517.
- [134] Burno, R.A., et al., Applying Fitts' Law to Gesture Based Computer Interactions. Procedia Manufacturing, 2015. 3: p. 4342-4349.

- [135] Achibet, M., G. Casiez, and M. Marchai. DesktopGlove: A multi-finger force feedback interface separating degrees of freedom between hands. in 2016 IEEE Symposium on 3D User Interfaces (3DUI). 2016.
- [136] Jin, H., et al., Multi-LeapMotion sensor based demonstration for robotic refine tabletop object manipulation task. CAAI Transactions on Intelligence Technology, 2016. 1(1): p. 104-113.