Behavior Changing Gestural Interface Design via Machine and Human Mutual Adaptation

Dun-Yu Hsiao

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2017

Reading Committee:

Zoran Popović, Chair

Joshua Smith

Seth Cooper

Program Authorized to Offer Degree:
Electrical Engineering

University of Washington

Abstract

Behavior Changing Gestural Interface Design via Machine and Human Mutual Adaptation

Dun-Yu Hsiao

Chair of the Supervisory Committee:
Professor Zoran Popović
Department of Computer Science and Engineering

Abstract

For centuries, the search for operating methods that are easy, intuitive and efficient has never stopped, and will most definitely go on. The current technologies allow us to interact with computers without mouse and keyboard with alternatives like gestures (visual inputs) or voice commands (audio inputs). As We manage to make communication between human and computer easier and more like that among human beings, it inevitably comes with ambiguity due to the complexity of input environments. This ambiguity exists in the latest interactive technologies like VR/AR (virtual reality/augmented reality) and drone interactions.

VR/AR equipments are nowadays quite accessible for people who want to create interactive immersive environments. However, as conventional input methods like mouse and keyboard are not included in the interaction that comes with these devices, alternatives that are as efficient and comfortable are required. Similar situations happen in drone interactions. Presently, the most common user scenario of drone is shooting fly-by footages for events or sports, in which users often could not afford to spare their hands as they are busy flying the drone.

For the devices designed for VR/AR or drone users, gestures could be an ideal interaction candidate for hardware compatibility and resemblance to how people interact with the real world. Presently, there are various gestural sensors available for gestural interfaces, and they come with fascinating capability of tracking user movements. However, there is still much space forimprovement in terms of precision. Although they may feel more intuitive than traditional methods like keyboard and mouse, they are on the other hand more ambiguous. Considering that Microsoft Hololenses are still shipped along with clickers, and that Vives work best with controllers, it's obvious that due to the limitations in current sensor technologies and human kinematic sciences, existing gestural interface designs are far from being so effective and pervasive as conventional methods of mouse and keyboard. People nowadays are not yet able to apply gestural interfaces to replace all the interactions.

Since the major purpose of gestural human-computer interaction development is to make interactions as intuitive as possible, human factor naturally plays a crucial role in the process. Hence, we propose an ideal human-computer interaction - *Mutual Adaptation*. The *Mutual Adaptation* technique comes in two parts: *Human Adaptation* and *Machine Adaptation*. *Human Adaptation* is about creating human-computer interactions to facilitate human to adapt, while *Machine Adaptation*goes the other way around by demanding machines or device to adapt to human operation. Despite the latter being much more prevailing than the former, we believe that the former comes with great potentials and worth exploring further.

In this work, we'd apply these concepts to the design of gestural human-computer interaction, and evaluate the effectiveness of various types of applications. we built our interaction prototypes around an online protein folding game, Foldit. Several user studies had been done to show the effectiveness of this approach.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table Number Page

ACKNOWLEDGMENTS

The journey of my PhD program has come with much more twists and turns than I could ever imagine. Without many people's help, I could never go this far. I sincerely thank everyone who helped me all these years, and please forgive my carelessness if I forget to include anyone who deserve the credit.

I'd like to thank my family, especially my parents and grandparents. Their full support has driven me all the way through this journey. Both of my grandfathers passed away before my graduation, and I hope they can see this in heaven. In addition, I am very lucky to have a group of uncles and aunts that give me so much encouragement as well as inspirations.

I would like to thank my mentor, Seth Cooper. After all these years, I owe him so much. Seth offered a lot of help and spent plenty of time discussing with me, working on the paper with me and guiding me through the process. There were always inspiring outcomes every time I went to him. During the hard times I had, he was always there for me to consult with even though it is never his responsibility. I thank Seth for helping me set my mindset for work. I have learned a lot from Seth, and I will always be grateful.

Also, I would like to thank my advisor Zoran Popović for giving me access to the great research environment of GRAIL. In GRAIL, I have known many outstanding people and learned about their researches. After all these years, I have learned lot a through Zoran, and for this I'm eternally grateful.

Besides, I would like to thank my co-authors, Min Sun and Christy Ballweber. Min and I chatted frequently about research together after Seth left Center for Game Science. Min is very learned in computer vision and our discussions bootstrapped and refined some of my

later work. Christy provided me some timely support for my past submissions. I also want to thank Marianne Lee for providing prompt support for my graphical needs.

In addition, I would like to thank Ching-Yi for being a supportive friend in most of my PhD years. Although not a major in STEM, Ching-Yi often enlightened me in terms of research with his creative thinking. I would also like to thank Tianshu, who encouraged me, shared my load and saw through my last two years of PhD.

I want to thank all my other committee members: Joshua Smith, Eve Riskin and Jacob Wobbrock. It was my honor to have you in my committee. Much gratitude is dedicated to Josh for being my co-advisor at the same time, and smoothing out some friction I had with Zoran, who is not affiliated to EE. I would also like to address more to thank Jacob Wobbrock for offering so more help than I could imagine. I see how Jacob takes his students seriously, and how willing he is to teach his students all what he knows.

I also want to thank Ali Farhadi for taking me several times as his TA, and it was truly my pleasure to work with him. In addition, there are several faculties in both EE and CSE who helped me in many different ways and times, including Maya Gupta, Joshua Smith, Larry Zitnick, Ming-Ting Sun, Eve Riskin, and Mari Ostendorf.

I want to thank my colleagues in Center for Game Science. Firas Khatib, Jeff Flatten, Barbara Krug, Hao Lu, Kefan Xu, Noah Snavely for offering technical or emotional support. Special thanks to Firas, who has always been so faithfully supportive.

I would like to thank the supporting staffs in CSE and EE,including Pim Lustig, Lindsay Michimoto, Brenda Larson, Elise Dorough. CSE helped me survive, and financially supported my research for several years. In my last two years, EE helped me iron out a few kinks in the process.

I want to thank Qi Shan and my colleagues in Zillow for being supportive and flexible to my needs in the last phase of my PhD program.

Last but not the least, I would like to thank some of my friends, including Jessica Yellin, Rahul Banerjee, Yvonne Chen, Travis Mandel, Eleanor ORourke, Yun-En Liu, Eric Butler, Kathleen Tuite, Chia-Fang Chung, Keyu Chen, Matthai Philipose, Ian Simon, Elizabeth Tseng, Earl Browne, Jyana Browne, Justin Lee, Roy Kuo, Pei-An Lee, Yen-Ting Kuo, David Chuang, Tim Liao and Chichi Hsiao for all the feedback, life support or editing advise. There are still many others. I thank them all.

DEDICATION

To Ching-Yi, a great friend and companion of the journey.

Chapter 1

INTRODUCTION

1.1 Motivation

For centuries, it has been part of human nature to pursue tools as intuitive and ergonomic as possible, and interactions with machines are no exception. Present technologies allow us to interact with computers, robots or drones with gestures (visual inputs) or voice commands (audio inputs) instead of traditional methods like keyboard and mouse. As We keep making communication between human and computer smarter and more adaptive to human behavior, lack of precision seems inevitable due to lack of an absolute standard. Consequent ambiguity is easy to observe in a few recently emerged interactive technologies, such like VR/AR (virtual reality/augmented reality) and drone interactions.

Figure 1.1: Gestural interaction can play an important role in the future VR/AR environment.

When it comes to VR/AR interactions, we now live with the luxury of abundance of choices available. Thanks to many major tech companies that joined the game and have produced their own VR/AR gears, various types of devices are for choice, including Oculus Rift, Google Cardboard, Samsung GearVR, HTC/Steam Vive, Microsoft Hololens and Meta

Meta 2. These devices make it much easier than before to create immersive environments, but as it's impossible to include keyboard and mouse in the interaction with these devices, alternatives are required. In terms of drone, a great number of drone startups were founded in the last 5 years, while most of the drones they developed come with cameras for fly-by footages. In the interaction with these devices, user often could hardly spare a hand to control the camera as they are fully dedicated to controlling drone movements or the activities to be recorded.

Figure 1.2: Interaction with drones, gestures can be more convenient.

Considering the actual user scenarios with VR/AR or drones, gestures become an ideal candidate as it's easily accessible and feels intuitive. Various gestural sensors such as Microsoft Kinect, LeapMotion Leap, and Playstation Move are aimed to aid people in building up gestural interfaces. When these sensing technologies work perfectly, they come with fascinating capability of tracking user operation. If these gestural sensors can all operate as expected, together with the VR/AR devices they will bring the next major evolution in human-machine interface.

However, the reality is that gestural interactions are not always as precise as we'd like them to be. They are intuitive, but could also be ambiguous sometimes. Considering the facts

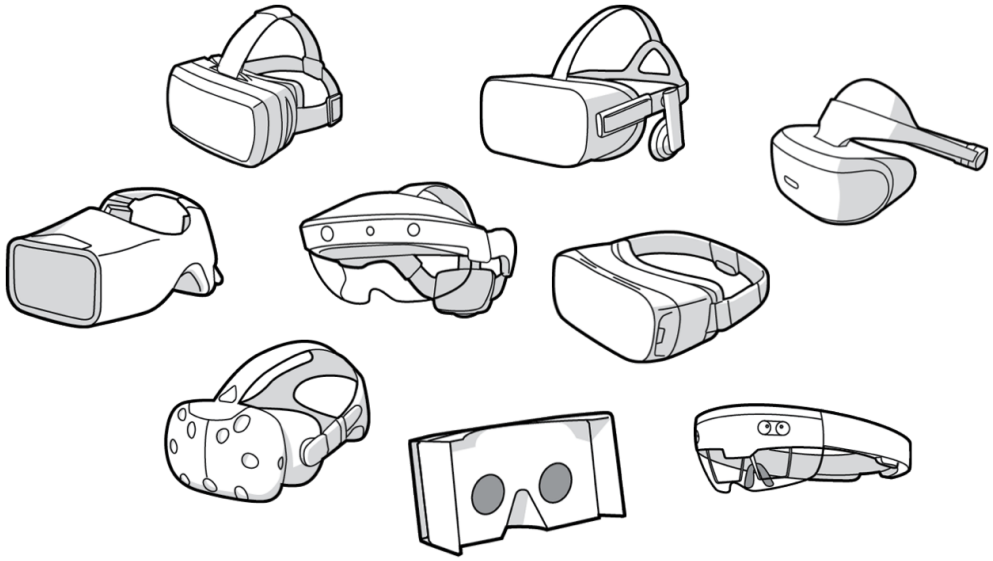


Figure 1.3: VR and AR equipment available nowadays.

that Microsoft Hololensas are still shipped with a clicker, and that Vive works best with the controller, it's obvious that due to the limitations in current sensor technologies and human kinematic sciences, existing gestural interface designs are far from being so effective and pervasive as conventional methods of mouse and keyboard are, and thus are not ready to replace existing methods completely.

Since the major purpose of human-computer interaction development is to make the interactions as intuitive as possible, it is only natural to include human factor in the solution to the challenge of ambiguity. For example, when a human is communicating with another human, if there are some ambiguity in the communication, the following strategies are often applied: one could adapt oneself to the other person, making the receiving end receive information better by leaning toward the other or putting palms to the ears to enhance the sounds; one could also make the other person adapt to oneself, asking the other person to speak louder, or repeat what he said. We believe the same tactics are applicable to human-computer interactions. While making computers more adaptive to human(*Machine Adaptation*), we should also evaluate the possibility of facilitating human to adapt in the

(a)

(b)

(c)

Figure 1.4: Human uses several different strategy to improve communication quality and reduce ambiguity. (a) Facing the person who is talking. (b) Use hands to increase signal strength in transmitting and receiving. (c) Leaning toward the person.

process (*Human Adaptation*). In this work, we propose the ideal human-computer interaction based on the combination of *Machine Adaptation* and *Human Adaptation*. We called this *Mutual Adaptation*.

Our work focuses on exploring the possibilities of creating with *Mutual Adaptation* the next generation gestural interface that is capable of tackling real world tasks. We explored *Human Adaptation* by creating dynamic gesture systems that can alter themselves to actively nudge users to adapt. Also, we explored *Machine Adaptation* through building a mobile sensing platform to proactively track user operations. Previously mentioned approaches were tested upon complex real world 3D object manipulation task. We chose to use a protein

folding game, Foldit, to be our major test environment because of the following factors: as the protein folding processing involves high 3D structure complexity, it is a task that can benefits from gestural interaction, which could provide high dimensional inputs. In addition, this task requires precision, which can be challenging with gestural interaction due to its ambiguity, while we could verify the actual effectiveness of the gestural interaction by the results. Through the experiment, we can learn to build practical gestural interface and to improve the interactions between human and computer.

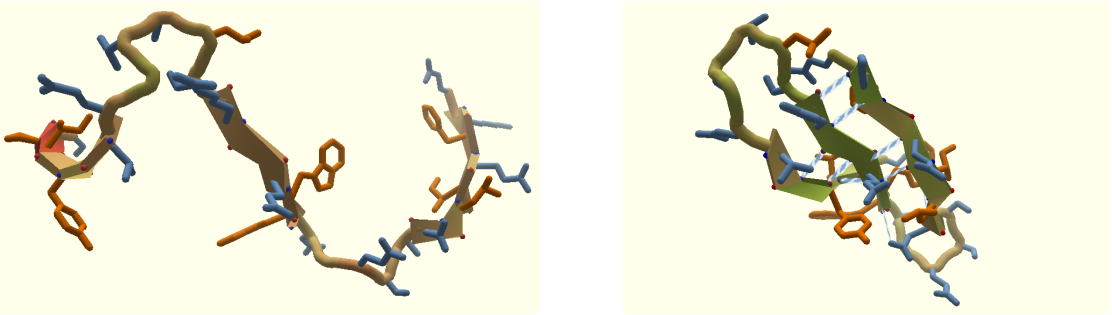
Figure 1.5: Gestural sensors available nowadays.

1.2 Gesture Interactive System with Ambiguity: Foldit Hand

1.2.1 What Foldit Is

Foldit is a game of manipulating highly complicated 3D protein structures in order to help solve protein-structure prediction problems. People who play Foldit would try to fold a

protein into a stable structure in a competitive yet collaborative gaming environment. The purpose is to reshape the proteins from long chains of atoms to more compact shapes than the originals. The most compact shape, which called the *native structure*, can guarantee the best stableness of a protein. The native structure has the lowest free energy, which means it has the most favorable set of chemical interactions. Scientists can then use the native structure to determine a protein's true function, and it could be the key of cure to many severe diseases or the materials with significant medical values.



(a) Exemplar protein start structure (b) The improved structure of the same protein

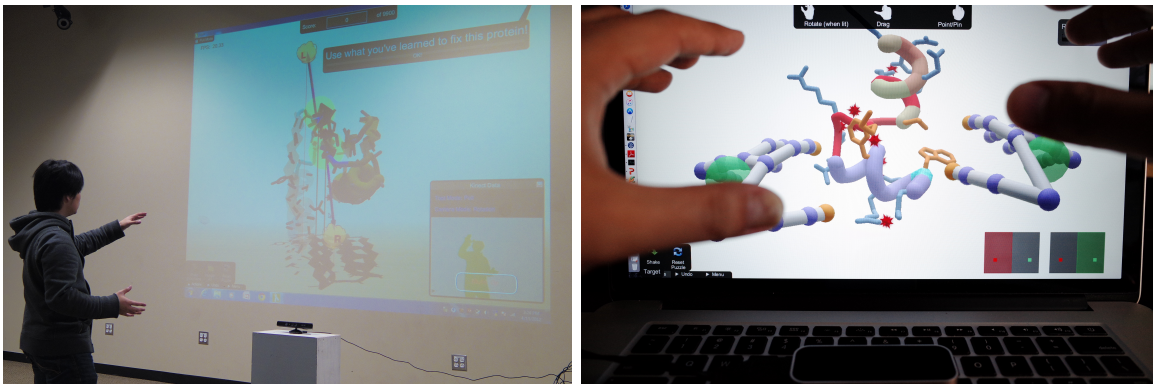
Figure 1.6: The protein start and end structure. Players of Foldit will try to improve the protein structure by finding the more compact conformation.

However, it is not easy to do protein structure predictions. As the proteins come in long chains of atoms, more specifically, amino acids, the structures of proteins are highly flexible. Each amino acid has its own degrees of freedoms, thus when they forms a long chain, the overall structure can posses extremely high inner degrees of freedoms. What makes the problem more difficult is that the amino acids come in over twenty types. The extremely high complexity is the reason why protein-structure prediction problems are very difficult to solve, even for clusters of advanced computers.

In this game, the brainpower of the gamers are made use of to solve previously unsolved scientific problems, while there has been great success in crowdsourcing complex scientific problems in the form of spatial-reasoning puzzles with worldwide collaboration. However, more efficient problem solving would require more of what people can offer, the first of which

would be human hand operation. Gestural sensors allow users to do interactions in solving proteomics problems, which are fundamentally three-dimensional ones that could not be dealt with with conventional two-dimensional input methods like mouse and keyboard.

1.2.2 Foldit Hand: Foldit with Hand Interaction



(a) (b)

Figure 1.7: The player uses his hand to manipulate the protein structure using either a (a) Microsoft Kinect or (b) LeapMotion Leap sensor.

Foldit Hand is Foldit enabled with a new 3D interface design. Players of Foldit Hand use their hands to manipulate the protein structures in air. With the interactive tools provided in Foldit Hand, players can drag, pull, bend, and mold the protein structure like what they can do to real objects. The major challenge of this interface is that the key hand-based modalities for game players to interact with structural proteomics problems are currently unknown. Determining interactions is challenging and requires more research in developing effective and intuitive methods that bring out players' full potentials in problem solving.

The goal of our experiment is to develop an effective hand-based protein folding gestural interface for Foldit. This interface will be fun, intuitive, and friendly to everyone, including new users. We will set up effective new modalities, and come up with methods with which manipulation of proteins with two hands can exceed that with mouse and keyboard. Further-

more, we will explore the benefits of operation with even more than two hands. That is, the operation that allows multiple players to collaborate simultaneously on the same structural problem. In addition, integration of hand-based folding with macros and automation would be included. We will evaluate the effectiveness of the interface by doing tests on players, and refine it over time. Ultimately, we hope to make breakthroughs with the new intuitive interactions in the game that leads to a brand new form of crowdsourcing.

1.3 Problem Statement: Interactive System with Ambiguity

There are several factors in gestural systems that can cause interactive issues. Because of its inevitable nature of ambiguity, the gesture inputs from users are often too noisy for the computer to determine what they actually mean. Furthermore, the mind model of the interaction might not be conveyed the same to the users: when a designer creates a desired gestural interaction, individual users often begin with different understandings in terms of how to perform the exact gestures. What is more, the desired interaction is often defined based on the limitations of the sensors. The interactions may hence be less intuitive, making it challenging to learn and to use. These challenges are quite observable in our test prototype.

Foldit Hand is a challenging and multidisciplinary project that involves theories of computer science, learning technology, psychology, art, and game design. In the following paragraphs, we listed a few major challenges we faced in the process of creating Foldit Hand.

1.3.1 Practical High Precision Tasks Using Ambiguous High Dimensional Input

With gestural sensors, Foldit Hand can capture human hand gestures and go beyond 2D inputs. However, this opens up a challenging issue: how to maintain precise folding operations with intrinsically noisy inputs from these sensors? A protein is a highly flexible long chain of amino acids. Each amino acid has its own degrees of freedom, thus the overall structure can possess extremely high inner degrees of freedom. It's very common for a protein to come

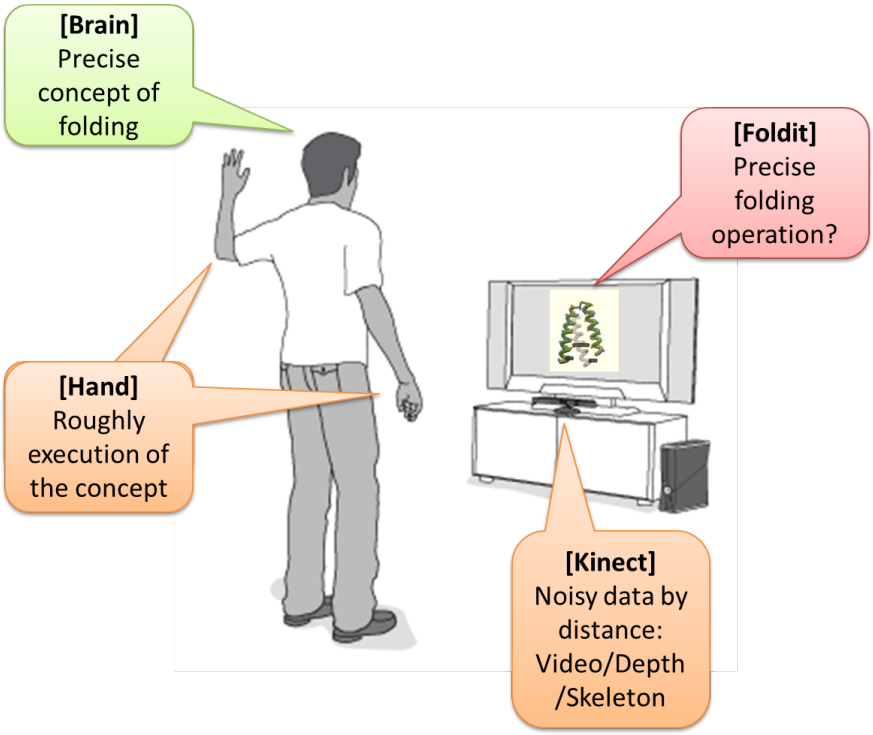


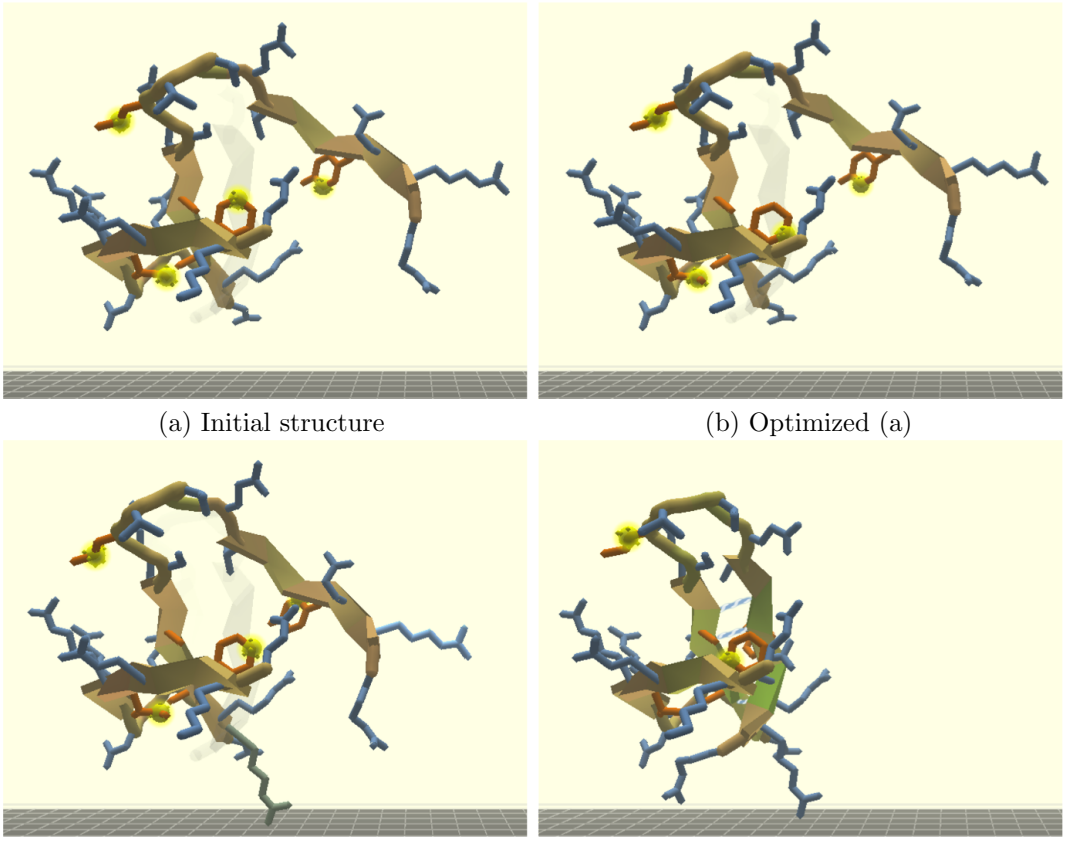
Figure 1.8: Challenges of the Foldit Hand system with an example of using Kinect.

with hundreds of degrees of freedom.

As a result of this complexity, any minor difference in the structure might prevent the structure from being an truly effective solution (as shown in Fig.1.9). In addition, the best compact conformation can vary subtly compared to other less ideal structures. Hence, to fold a protein correctly relies heavily on both 3D understanding and precise manipulation skill. Protein folders must choose the correct spot to work on and the exact spot to place the protein at.

1.3.2 Physical and Sensor Limitations Impact Input Tracking Quality

What makes the precise operation even more challenging is the tracking error. Other than the sensing noise, tracking error caused by occlusion is very common and the most severe



(a) Initial structure (b) Optimized (a)

(c) Minor tweak on (a) (d) Optimized (c)

Figure 1.9: We use protein folding as a representative precise two-handed manipulation task. Precise manipulation is required in Foldit, because even slightly different optimization standards can lead to very different results.

challenge in gestural sensors. Caused by occlusion or near-occlusion, this type of error may happen in different occasions (for example, with single or multiple users), and is very space dependent.

In Fig. 1.10, we mark the high-probability-occlusion area in red, low-probability-occlusion area in green. In addition, user could create new high-probability-occlusion region as two tracked joins are too close, for example, when two hands are close to each other. The amplitude of this noise is peak-like when it occurs: it acts like a value interchange between the affected coordinates so players will find the tracked parts jumping or lost. The occlusion

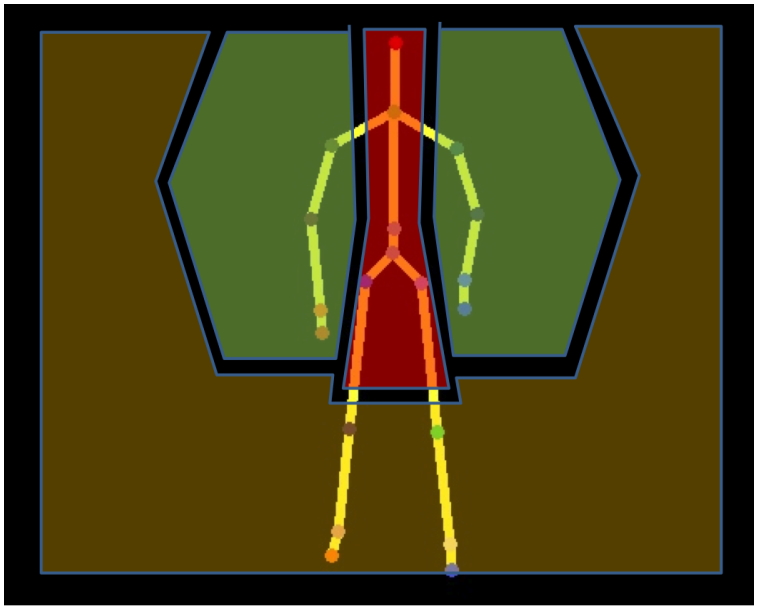


Figure 1.10: The skeleton tracked by Kinect. Colored areas indicate possible occlusion occurrence (sensor limitations). Red means high, green means low, and yellow means hard to reach intuitively by humans (physical limitations).

noise can reoccur very easily, because there are eight tracked joints in that region that can occlude one another. This will greatly reduce efficiency and lead to fatigue and even frustration to users. It is due to sensor limitation when designing gestural interactions.

The other type of limitation is the physical one. The physical limitation is created by the users when some interactive regions are not ergonomic. These regions are usually hard to reach for users. In Fig. 1.10 we mark this kind of region in yellow.

1.3.3 Interaction Design Trade-off Introduced by Limitations

When designing gestural interfaces, designers usually have to figure out a way to avoid the two mentioned limitation to keep interactions reliable. Otherwise, the disruptive tracking can severely degrade the overall interaction effectiveness. It interrupts and deteriorates any desired motion: the user has to recover from the erratic move and try again to make it right.

The error may still affect the user's following attempts as long as he/she still creates similar occlusion on the tracked parts.

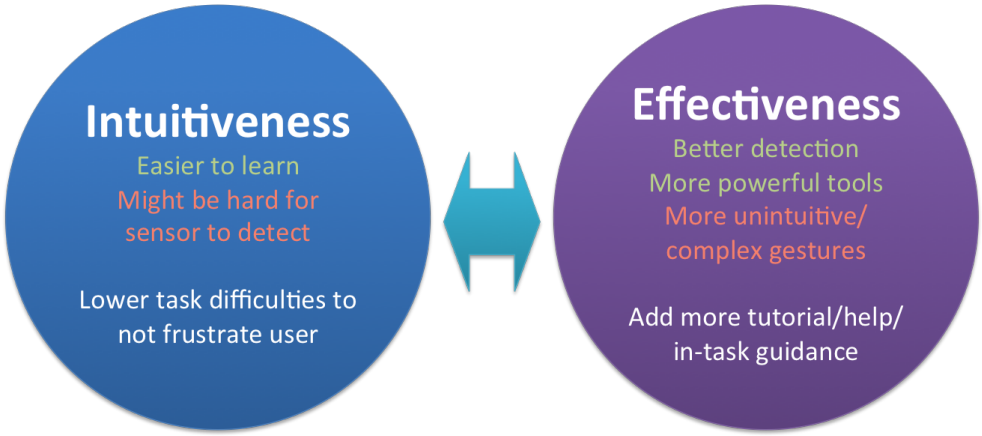


Figure 1.11: The trade-off between effectiveness and intuitiveness in designing gestural interface.

Take the normal folding gesture as an example, it is intuitive for humans to fold things right in front of their bodies. Unfortunately, as indicated in Fig. 1.10, this is not an ideal region for detection. However, if we consider the limitations and ask the users to interact in the green regions, the interaction might become less intuitive because the interaction is not a familiar metaphor anymore (as shown in Fig. 1.12). The less intuitive interaction can be difficult to learn, even with an extensive training process.

1.3.4 Learning to Gesture in the Desired Way

When a designer creates a desired gestural interaction, users often begin with different understandings of how to perform the interaction exactly. For example, when we asked users to perform a "pinch" gesture among a small group of 20 people, we didn't have any user that gestured in the same way as any other users. Furthermore, the desired interaction is often defined based on the limitations of the sensors. The interactions may hence be less intuitive, making it challenging to learn and use.

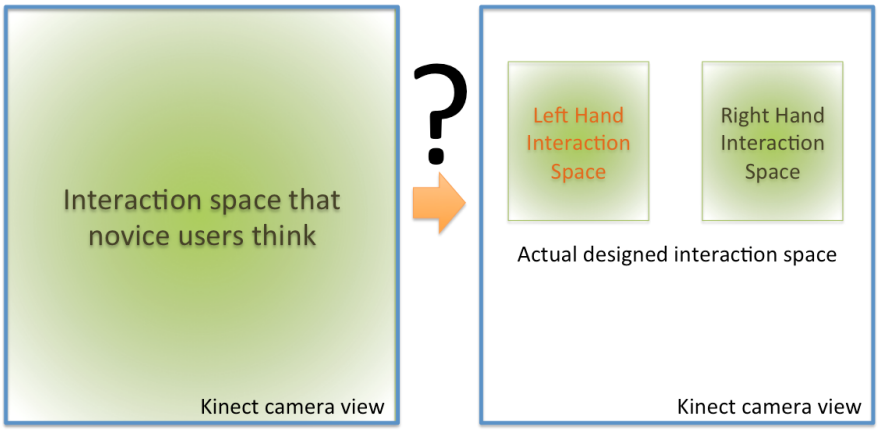
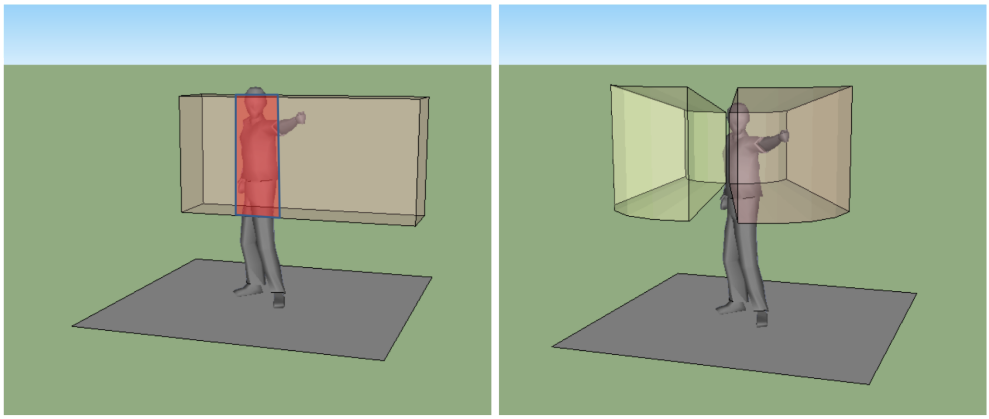


Figure 1.12: The designed interaction can be very different I terms of what users understand. In our experiment we designed a coordinate mapping space which is respective to each hand. It is challenging to make users to work in such unusual scenario with minimal training.

1.3.5 Learning to Interact in 3D

What makes the learning problem more difficult is that monitors, mice and keyboards have been the major human-computer interfaces for decades, while humans are quite well-trained to use them and thus not so experienced in alternatives. Compared with these methods, however, the gestural interaction is still new to the general public, so it is impossible that users could begin doing it (for example, in a 3D-to-3D mapping interface) effectively and precisely right away. What is more, from the survey we had for existing Kinect games, most of them did not at first actually use the 3D capability nor adopt the distance/depth measuring that Kinect provides in the game. Instead, they are still using it as a 2D interface without the additional gesture feature. Research also shows that people often find it inherently difficult to understand 3D spaces and to perform actions in free space. Dealing with the learning difficulties for gestural interfaces is an inevitable challenge we should face.



(a) Naive mapping (b) Designed mapping

Figure 1.13: Different hand coordinate mapping schemes. (a) Naive intuitive mapping can easily come with occlusion close to the red region due to shared space among tracked objects. (b) Our mapping eliminates the shared space while retaining the freedom for hands to move. However, it is not so intuitive and hard to master.

1.4 Proposed Methods to Address Gestural Interaction Challenges

In order to address challenges that mentioned above, we established a few guidelines that described as follows:

1.4.1 Interactive Tools Based on Limitation to Reduce Input Ambiguity

In order to address input ambiguity in gestural interaction system, we create interactive tools to be able to provide powerful functions that are suitable for gestures. The idea here is to optimize effectiveness by gauging the sensor's detection limitations and the user's physical limitations. The priority is to achieve good tracking quality.

1.4.2 Human Adaptation Breaks Design Trade-off and Expedite Users Adapt

Since the priority of the gestural interaction design is to enhance tracking precision, it may come with some sacrifice of intuitiveness in the interaction. We then use the advantage

of *Human Adaptation* to deal with the learning issue. *Human Adaptation* actively nudges the user's operating behavior into what is desired quickly. This can vastly improve user experience without always making an interaction intuitive with the design trade-off between effectiveness and intuitiveness.

1.4.3 Machine Adaptation to Optimize State for Human Adaptation

Considering the existence of the user's physical limitation, sometimes the system has to figure out what's the optimal way for different user to interact with the system. We apply *Machine Adaptation* for the system to automatically figure out the optimal interaction for the current user, and then use *Human Adaptation* again to nudge the user into that operation state.

1.4.4 Visualization Optimization to Help Interact in 3D

Based on the observation, people are not yet used to high dimension inputs while the output visualization is still on a monitor screen. In often times people are still applying 2D mindset even in a full 3D interaction. We discovered this and changed some of our interaction more 2D-like and worked great with users. We also added some visual cues for people to understand the interaction environment is 3D.

Based upon these guidelines, we created the test gesture environment to validate our proposed approach. In the next section we will provide more implementation details of the system.

1.5 Test Environment Setup

1.5.1 Interactive Tools

We designed Foldit Hand that provides the following tools to facilitate two-handed manipulation process. The two-handed interaction state machine is shown in Fig. 1.14. We designed

this state machine to allow seamless switching between using one hand and two hands.

- **Grab:** Drag to reposition one component in 3D.

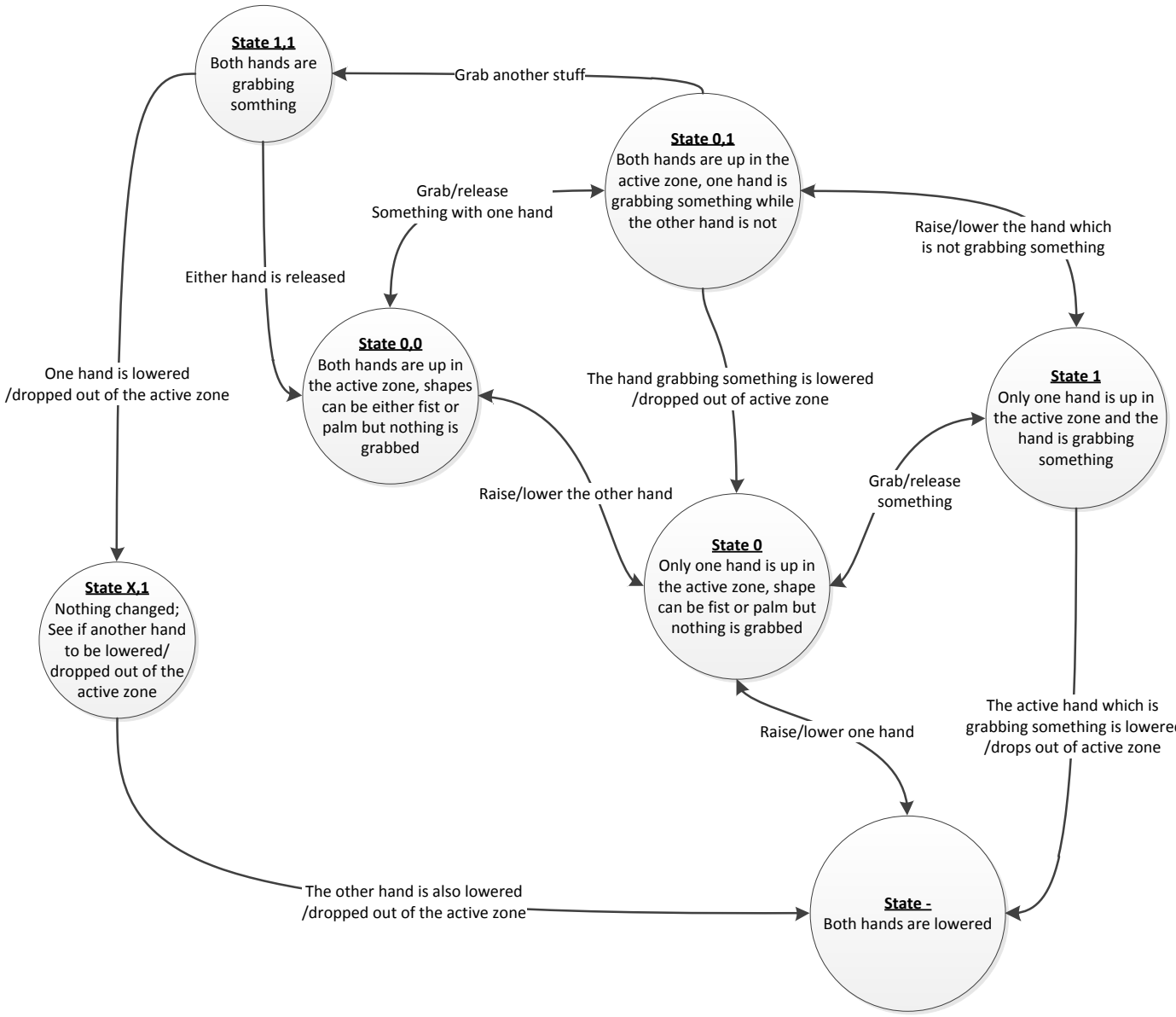


Figure 1.14: State diagram for our two hand control scheme.

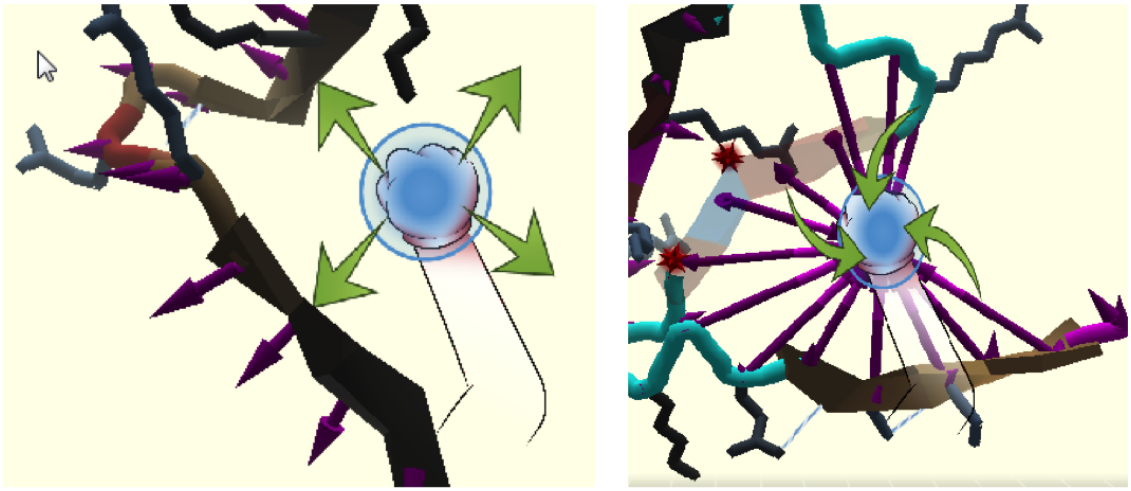


Figure 1.15: Force field tools. Left: repel. Right: attract.

- **Force Field: Repel/Attract:** A more powerful tool that allows forces acting on multiple nearby components. User can use this tool to easily split or merge parts of the structure.

- **Twist:** Apply two different torques onto two component.

- **Freeze:** Lock the movability of the component. This is useful for finer adjustment of the 3D structure.

- **Voice commands.** Voice commands to facilitate some frequent actions such as undo/redo or invoke structure optimization operations like shake/wiggle.

1.5.2 Assist 3D Object Selecting by (2+1)D Adjustment

Furthermore, we refined the object-selecting scheme. Originally this is designed as a full 3D nearest neighbor search, and the players have to put their hand at the exact 3D position to be able to select the desired object. Although having the visual guidances such as the drop shadows and the guidelines, people still made lots of mistakes when trying to select an object

by reaching with the wrong distance/depth. Understanding it is difficult to change people's familiarness of 2D selecting, we modified the object-selecting scheme to make it look like a 2D selection, users can still select any 3D objects under this scheme. The selecting scheme is first do a 2D-wise closest point search with some tolerance range. If there are more than one selectable object found in the 2D search, it will then compare them distance/depth-wise closest to the player's hand cursor. In this way, it reduces the frequency of players use the unfamiliar 3D-3D mapping and makes the selection process looks like their familiar 2D scheme. We found people can easily adapt to this kind of space mapping to do effective selecting while maintaining the ability to navigate in 3D.

1.5.3 Probabilistic Selection & Selectable Object Reduction

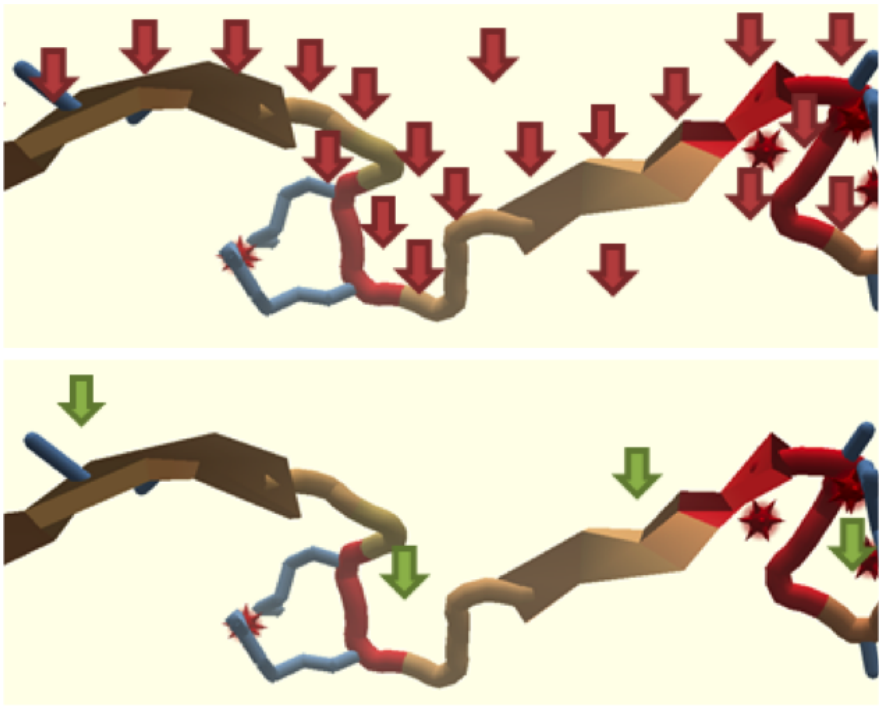


Figure 1.16: Reducing selectable objects for the interface (indicated by arrows, arrows do not appear in game). Top: before. Bottom: now.

For more easily to select and object, we also added a probability estimation over object selection. When the user's cursor is near a selectable object, we increase the probability of being selected on that component; otherwise, the probability will drop quickly overtime. This is to reduce the possibility of mistakenly select on a different object when executing the selection during motion.

In addition, seeing it's difficult for players to select objects on screen precisely, we reduced the number/density of the selectable/interactable objects on screen by grouping them based on the secondary structure. It reduces the number of selectables on screen from hundreds to lower than dozens, and hence effectively decrease the hassle and chance of selecting the unwanted object. To further eliminate interactable objects on screen, we used the player's body to control the camera instead of demanding the player to drag the background (shown in Fig.1.16).

Because using the hand as a cursor is less precise than a mouse, allowing the background interactable will increase the difficulty to select objects correctly when the player only focuses on the protein structure manipulation. In our further integration, the camera control is now controlled by the player's body, which reduces the job for hand and is proved to increase usability.

1.5.4 Adaptive Selection for 3D Object Manipulation

To Improve the object-selecting scheme, we made the selection more adaptive and similar to how people react to the objects in the real world. For example, when humans want to do a finer manipulation over an object, they apply multiple torques onto it to change the orientation or twist it. Similar scenarios apply to our system: if the user uses both of his hands to select the same secondary structure (a helix or a sheet), the user's hands will automatically drag on the ends of that structure to reorient it; otherwise, if the user's hands land on different secondary structures, the system will freeze the structures in between to allow bending/twisting/folding motions (Fig.1.17).

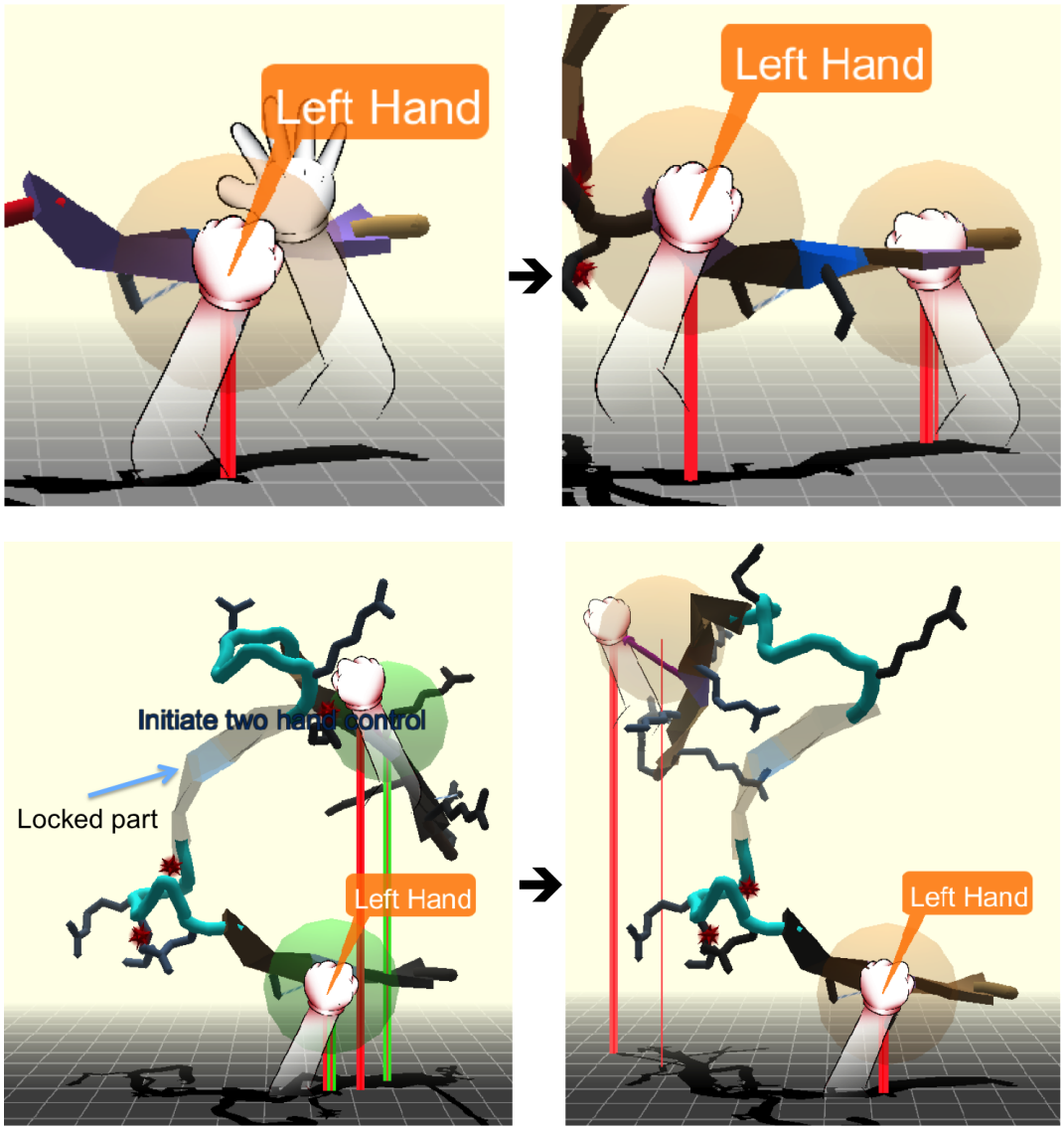


Figure 1.17: Adaptive two-hand interaction. Top row: when both hands are grabbing the same secondary structure, both ends of that structure will be grabbed. Bottom row: when the two hands are grabbing two distant structures, anything in between will be locked to allow bending on both sides.

1.5.5 Visual Guidance

To help users understand how to interact with our environments, we provide visualizations like shadow, grid plane, and cursor position indicator. They are meant to enhance the feeling

of 3D, while some tools are accessible from pie menus activated by hand gestures (Fig. 1.18).

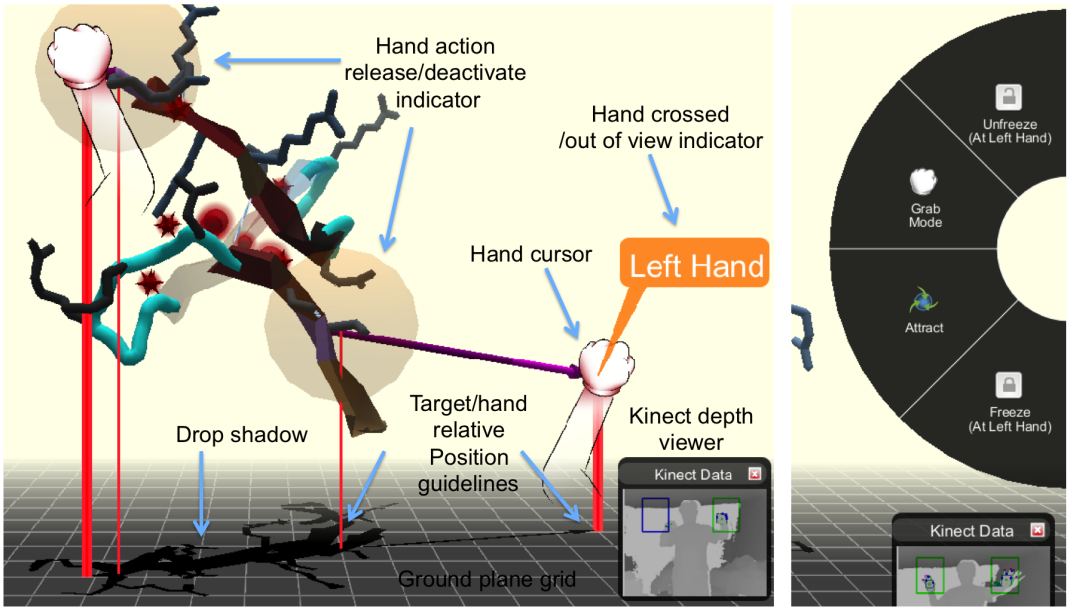


Figure 1.18: Left: Foldit Hand game screenshot, showing the visualizations that help player fold effectively in 3D environments. Right: menu example for Hand players.

1.5.6 Improving Gestural Interface Performance through Human and Machine Adaptation

In the following chapters, we will describe our experience in designing Foldit Hand with an emphasis on two parts: *Human Adaptation* and *Machine Adaptation*. Either of them can further improve the overall performance of the interaction to overcome the mentioned challenges.

Human Adaptation is about how we can effortlessly lead user to our desired interaction without explicit training. This is done by manipulating the sole feedback system in the gestural interface–Visualization (Fig. 1.19 & Fig. 1.20). As the visualization of the environment changes in an imperceivable way, the user's originally intuitive interaction is gently nudged into what we desired to gain better effectiveness. This will remove the trade-off dilemma in designing with balance between effectiveness and intuitiveness. *Human Adaptation* opens

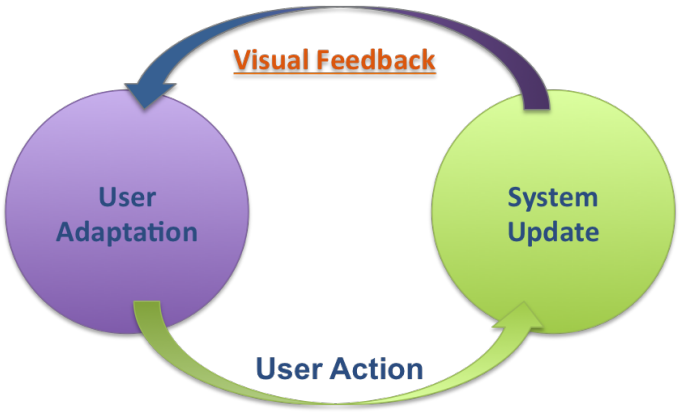


Figure 1.19: Manipulating visual feedback to facilitate human adaptation.

the path that leads to the best of both.

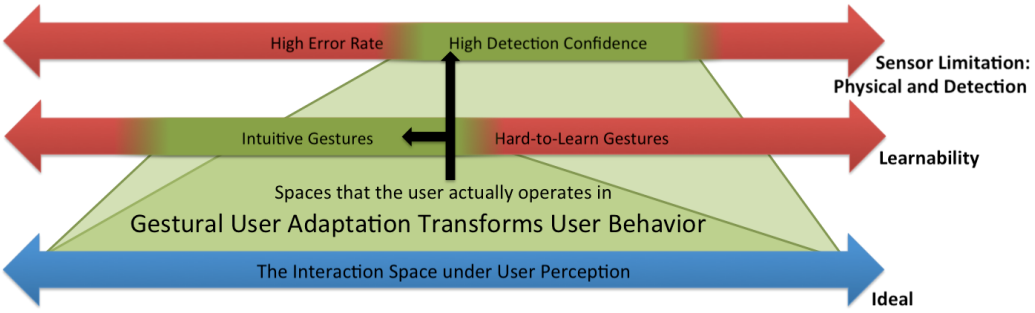


Figure 1.20: Human adaptation via dynamic input mapping shifts user interactions from the intuitive interaction space gently toward the effective interaction space (high detection confidence) as the designer's intended model.

Machine Adaptation, on the other hand, is about how the interface can adapt to the users. If the sensor can move proactively on platforms such as a robot, robot arm or even a drone, the interface can adjust itself to gain better performance by aligning better with the user's operation.

We will present our preliminary results in the next two chapters about *Human Adaptation* and *Machine Adaptation*, and we'll show how both of them can improve user performance.

Chapter 2

HUMAN ADAPTATION: ACTIVE NUDGING–A DESIGN APPROACH FOR EXPEDITING USER ADAPTATION

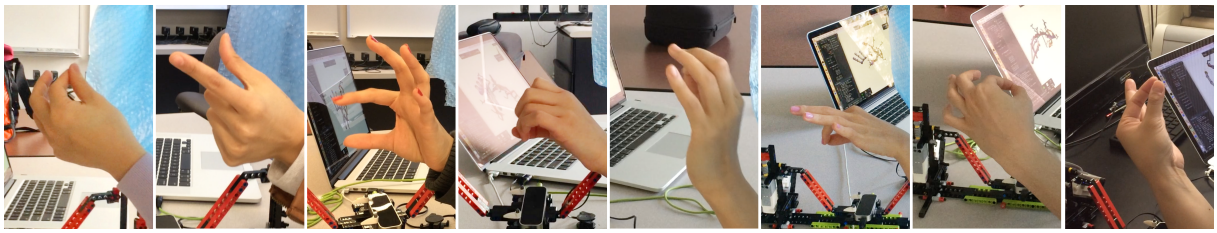
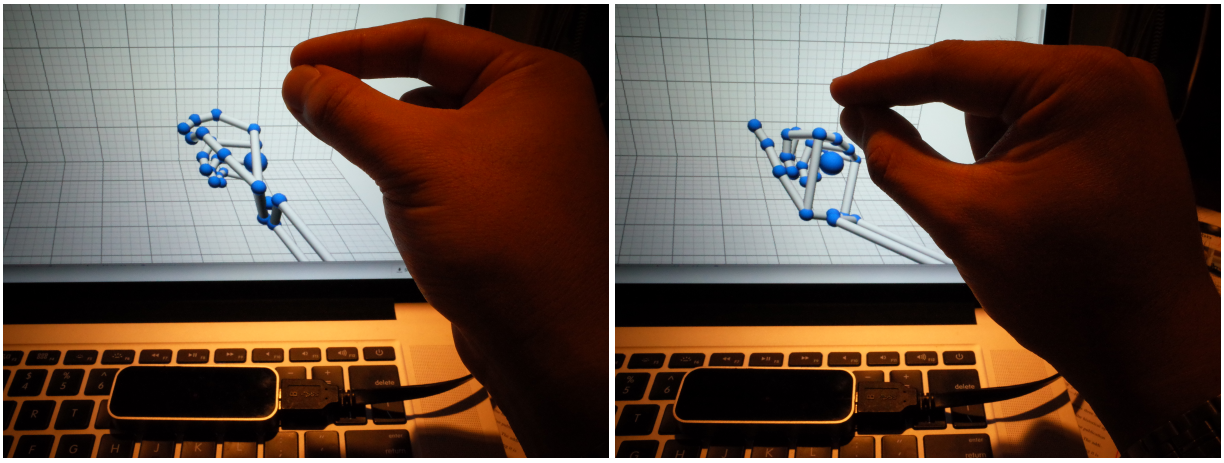


Figure 2.1: Pinch gesture variations among test subjects as the subjects were told and showed to perform a "Pinch" gesture.

While gestural input sensors are becoming more prevalent, it is still challenging to create gestural interfaces that are both robust and intuitive. When a designer creates a desired gestural interaction, users often begin with different understandings of how to exactly perform the interaction. To address this challenge, in this paper we propose the interaction design approach called *active nudging*. The *active nudging* changes the relationship between the users' actual input and the visualized input over time as they interact with the interface. Changing this relationship implicitly directs users to perform the desired interaction eventually; no explicit instruction or additional guidance are used. We demonstrate the effectiveness of this approach by applying it to interactions with the Leap and the Kinect sensors, for which we desire to nudge users to high detection confidence areas of the interaction space, which reduce occlusions and other ambiguities. We perform a user study that shows that the active nudging design approach allows users to perform tasks in less time with fewer errors than without it.

2.1 Introduction

In the past few years, virtual/augmented reality (VR/AR) technologies and devices have changed human-machine interactions for good. Seeing humans' proficiency in touch gestures widely applied to phones, tablets and computers, camera-based gesture controls devices including Microsoft Kinect [22], Leap Motion Leap controller [21], Playstation Move [31] and Nintendo Wii Remotes [24], have become promising partners for VR/AR environments.



(a) (b)

Figure 2.2: Occlusion can be destructive to tracking in camera-based gesture systems. A slight variation with occlusion can invalidate tracking completely. In our experiments, though users were told and shown how to perform the desired pinch gesture, many still came up with their own versions, even within a relatively small subject group of 20.

However, the fact that many VR/AR devices still come with controllers indicates that a robust hand-recognition solution is not yet available. A major challenge of this situation lies in operating limitations between users and camera-based systems. These limitations come in two categories: physical and system limitations. A good example of a former is that it's impossible to operate in touch regions where the user cannot reach. The latter, on the other hand, is determined by system capability of detecting user inputs. When the system is operated beyond its limitations, it might lead to spurious or even dropped frames,

compromising recognition. Camera-based system limitations are usually caused by occlusions from camera-user relative position either between users or within the user.

In designing an effective camera-based gesture system, a solution for the aforementioned operating limitations often involves defining a parameter space of working interaction and restricting gestures to within this space. Consequently, however, restricted operating gestures could become not so intuitive and hard to learn, while users might end up with various versions of gestures instead of the ones amenable to recognition by the system. Thus, a discrepancy arises between the intuitive gestures and those recognized by the system effectively. The result would be a trade-off between *intuitiveness* (gestures that the user prefers) and *recognizability* (gestures that the system can sense and recognize).

The tension between intuitiveness and recognizability is illustrated in the example in Figure 2.2, in which a pinch gesture loses track when the fingers touch one another. The tracked skeleton on screen could not track the physical hand pose due to occlusion. To avoid this problem, the designer may expect the user to do a pinch *without touching their fingers*. However, this is a less intuitive gesture. In the following paragraphs, we would demonstrate how our solution encourages the user without forcing him or her to do gesture in ways more recognizable to the system, in this case, by not quite touching the fingers during a pinch.

Our technique is called *Active Nudging*, a solution that allows users to gesture as they prefer without being explicitly instructed, and yet improves recognition by encouraging subtle changes to users' gesture behavior. The basic idea is *in addition to adapting gesture systems to users, the system should also adapt (nudge) users to comply with the system to perform more and more effectively*. This goal is achieved by dynamically changing the mapping between the parameters representing users' raw physical input and the input that is actually fed to the system. For example, the mapping of relative distance between physical fingers and visualized ones is continuously altered to lead the users to make more and more recognizable gestures to the system.

In the previous example of pinch gesture, users may start with the intuitive way of pinch-

ing with fingers touching each other, but with our solution they will end up pinching with fingers apart without noticing their gesture has changed. Therefore, applying Active Nudging on existing interaction methods could reduce time of training on effective gesture or intuitiveness-recognizability trade-off. In every operation, it gradually transforms various users' gestures into similar ones that are relatively effective. with this mechanism, we can build a gesture system that is both effective and intuitive, because users will interact with it as the designer expects without having to follow instructions against their intuition.

in this paper we'd explore the potentials of Active Nudging in two studies in camera-based gesture systems with Kinect and Leap. Twenty university students, mostly non-gamers, were involved in each. The results confirm the theory that Active Nudging could improve overall user effectiveness in camera-based gesture systems. We found that Active Nudging helped users to perform the desired gesture 26% quicker and with 65% fewer retries (error corrections). Furthermore, users spent 21% more time operating the system using the gesture closer to that expected by the system.

Following are our main achievements. First, we provide a formal definition of the concept of Active Nudging; Second, we implement example applications to demonstrate our concept; Third, we present empirical results validating the benefits of Active Nudging. This work demonstrates the potentials of Active Nudging in camera-based gesture systems; we believe the same concept could be applied to other recognition-based interactive systems.

2.2 Related Work

There are numerous articles about learning and adaptive guidance for interactive gesture-based systems. However, most are about generating adaptive guides to help users perform effectively (e.g., [3, 11, 15, 29] and several other work mentioned below). Unlike them, this work does not focus on training users to use an interface effectively. Instead, it is more about implicitly encouraging users into making optimized *physical* gestures for the interface's *virtual* environment. The purpose of Active Nudging is not to generate more helpful visual guidance,

and there is no explicit guidance in our approach at all.

Engineered interactions. Most gesture systems must strike a balance between effectiveness and intuitiveness, determining the best possible gestures for recognition. Charade by Baudel et al. [6] was one of the early work that explored free-hand interfaces. Nevertheless, in Charade, the trade-off of intuitiveness for recognizability was quite obvious. Users had to learn from a long list of rather contrived gestures that were recognizable to the system but hard to perform. Akers et al. [2] used gesture brainstorming and gesture log analysis to figure out the interaction parameter space most suitable for a user. Song et al. [30] proposed a handlebar metaphor for 3D virtual object manipulation with Kinect. Their system may come with satisfying controllability, but could also cause much fatigue. On the other hand, Active Nudging can reduce the inevitable trade-off of intuitiveness for recognizability by making users do effective gesture without feeling strained.

Increased effectiveness using additional hardware. To reduce the occlusion that comes with intuitive gestures, many solutions have been developed with multiple-cameras. Offline ones include the research of Zhao et al. [42], Ballan et al. [4], and Wang et al. [38]. On the other hand, Sridhar et al. [32] used five RGB cameras and a time-of-flight sensor to track a user's hand at $\sim$10 Hz. Wang et al. [1, 36] also developed vision-based systems to track up to six degrees of freedom of each hand in realtime. There were also wearable solutions proposed to mitigate failures caused by occlusion. For example, Wang et al. [37] proposed vision-based systems using color gloves. However, the detection in his system could still fail if specific hand postures are not used. In addition, Kim et al. [20] proposed that the user wears a low power depth camera on the wrist. Similarly, Colaço et al. [10] developed a low-power, head mounted 3D gesture sensing solution. Harrison et al. [16] also proposed wearing a RGBD hand gesture recognition device on the shoulder. As these solutions require complex setups, none has been adopted by many.

Input mappings. Some other researchers, on the other hand, focused on how a system can be more adaptive to users. Casiez et al. [9] examined the control-display (C-D) gain

for mouse in pointing task, but the solution is limited to the mouse while the mapping in Active Nudging is not limited to a single 2D input. Frees et al. [12] proposed PRISM to dynamically change the C-D gain to improve 3D positioning, while our research comes with broader generalization of the C-D gain and focuses on gestures and interaction rather than just mouse motion. Owen et al. [25] provided an analysis of the effectiveness of one-handed and two-handed scenarios. However, they failed to find any significant difference between two of their proposed mappings. In contrast, our result shows far more difference in effectiveness between different mappings.

Redirected walking. In addition, Razzaque et al. [28] presented redirected walking in manipulating the rotating angle of the view. Suma et al. [33] showed redirected walking in physical change of motion by manipulating the virtual environment. Sun et al. [34] created a manifold that mapped the original 2D coordinates into a higher dimension. However, it takes a long time to do the environment calculation that is compromised by visual distortion, while the mapping is fixed instead of being dynamic. In short, redirected walking work focuses on physical interaction in a limited space, while our work aims at optimizing the physical interaction that makes virtual interaction effective. The user might "feel" the mapping is distorted but he doesn't need to see it—a closed loop in the virtual environment doesn't have to be a closed loop in reality. Despite Hsiao et al. [18] had some exploratory work in dynamic mapping for two-handed interaction, Active Nudging is more generalized and formalized. In addition, our approach could be applied to other scenarios.

Learnability. In other researches, the focus was mostly on how to improve the learnability of interactions. Nacenta et al. [23] tested and evaluated the memorability of pre-defined and user-defined interactions. Rachovides et al. [27] aimed to reduce the overhead of learning the system by encouraging the use of gestures associated with everyday activities. Polacek et al. [26] provided observations of Kinect interaction about the comfortable area for hands. Wobbrock et al. [40] elicited gestures by showing users a portrayal of the effects of a gesture first. Instead of finding more intuitive ways to help users learn gestures, the goal of Active

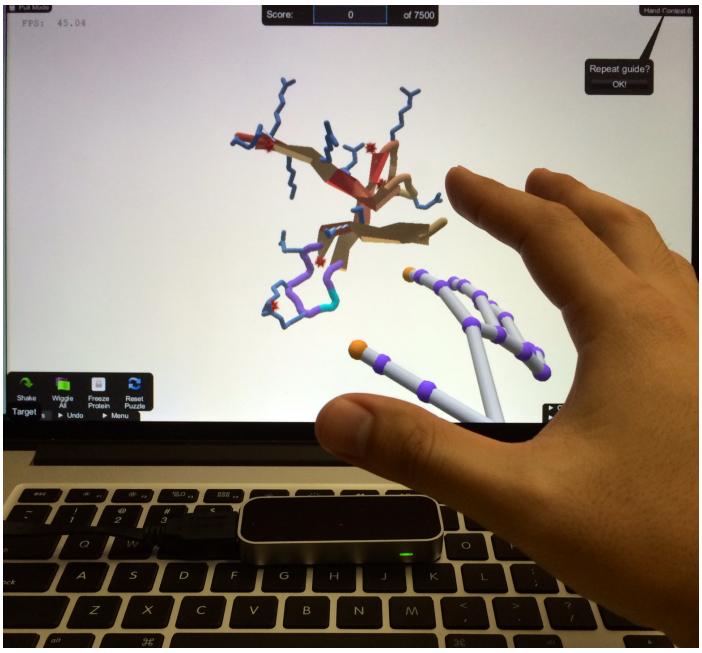
Nudging is to skip the actual training and gradually adapt users to the system.

Additional guidance and training. Still, in other researches, the explicit guides are introduced. Gustafson et al. [15] mentioned in their research the training for people to use an imaginary interface that adapts to the user's palm. The system proposed by Igarashi et al. [19] contained similar suggestions for 3D drawings. Anderson et al. [3] evaluated various types of non-gesture visual guides in their research. In addition, there has been interest in explicit training for users to perform effective gestures to the system. Octopocus by Bau and MacKay [5], GestureBar by Bragdon et al. [8], TouchGhosts from Vanacken et al. [35], LightGuide by Sodhi et al. [29] and ShadowGuides by Freeman et al. [11] all provided visual guides to train users to do expected gestures. Bragdon et al. [7] proposed a system that makes learning gestures fun. In contrast, the method proposed in this work does not make training explicit or display additional guides, but rather achieves the effects of training without actual training.

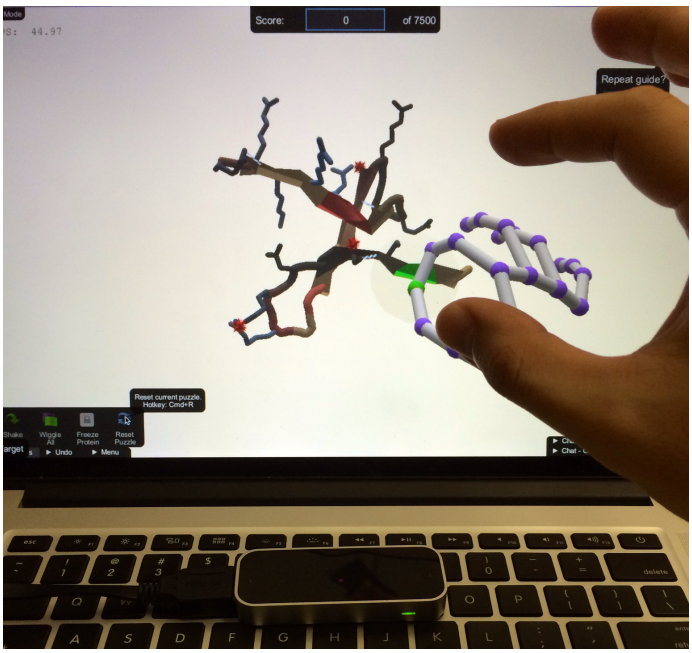
Adaptive interfaces. Additionally, there is some more related research in adaptive interfaces. Zamborlin et al. [41] created a system that learns from interaction for performance improvement. Gallacher et al. [14] proposed an adaptive system that learns user preferences. However, these projects were still quite different from this work, because with Active Nudging, the system does not simply learn to adapt to its users, but is more about changing users' behavior gradually to adapt to the system.

2.3 Examples of Active Nudging

In the introduction, it is demonstrated that a pinch gesture tracking can be not so effective with the fingers touching each other (Figure 2.2), yet gesture becomes not so intuitive otherwise. In this case, a system running with Active Nudging would allow users to begin with intuitive operations while dynamically changing the mapping between the fingers' physical distance and the corresponding visualization to nudge the user towards to aim of pinching without touching fingers (Figure 4.4).



(a)



(b)

Figure 2.3: Example of the Active Nudging for the pinch gesture activation scenario. (a) Finger distance mapping unchanged initially. (b) Pinched without touching fingers after Active Nudging. Note the hand model on screen (b) is already pinched but the real hand is not.

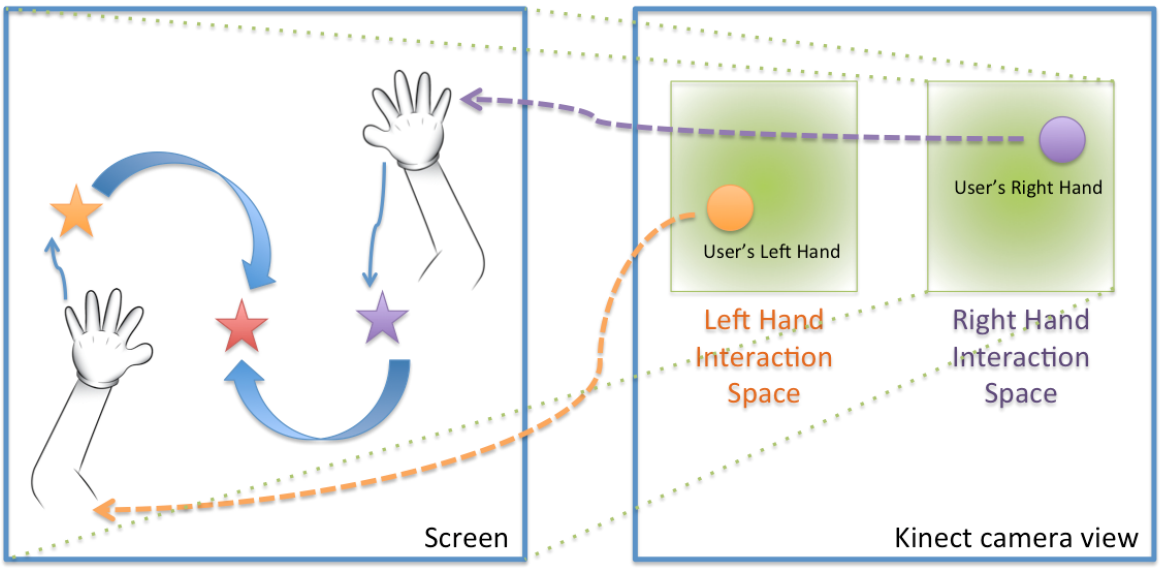


Figure 2.4: An example of Active Nudging applied to two-handed navigation. Users were asked to navigate with both hands on specific locations on the screen (simplified as stars in this figure) in order. Both hand cursors are mapped to the whole screen with the bounding boxes (green areas).

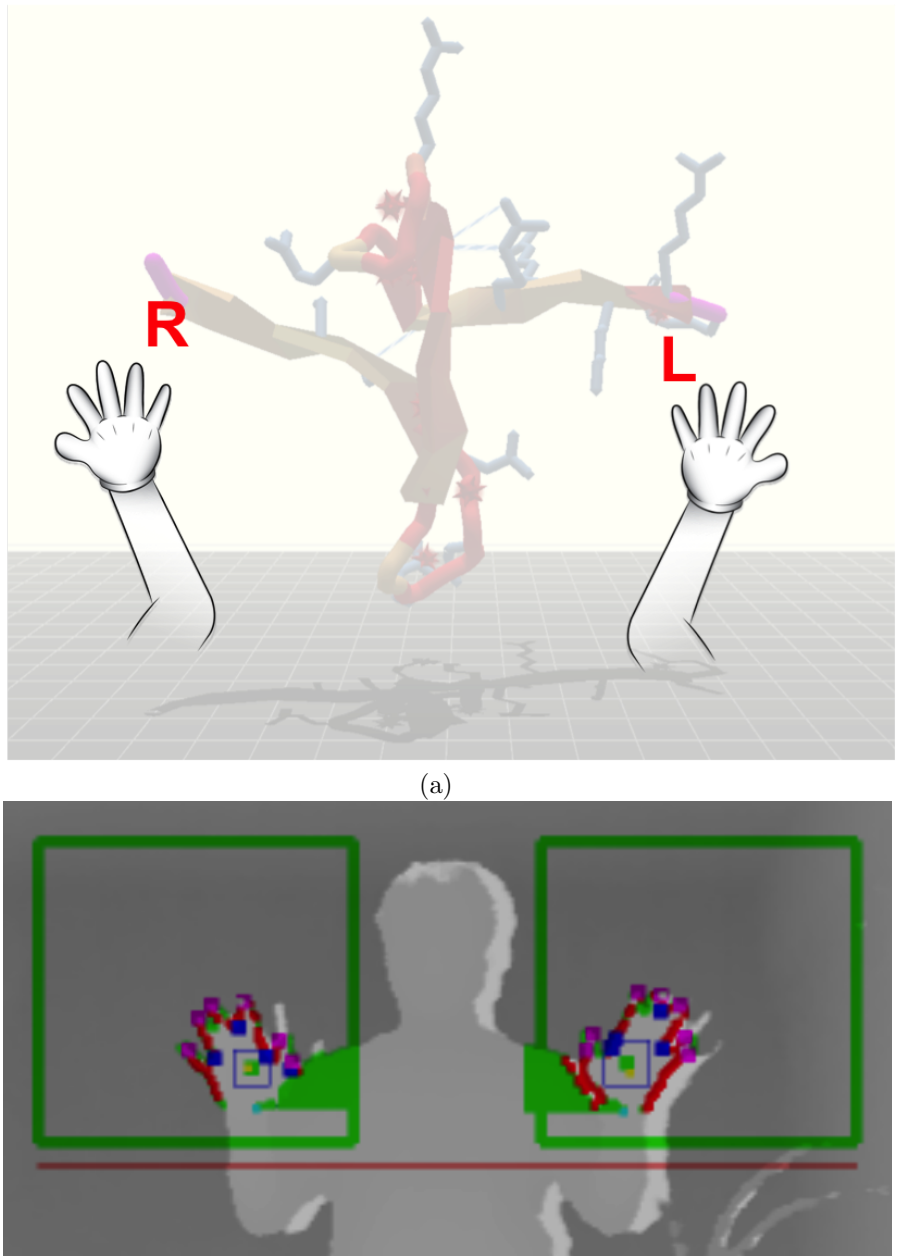
Another example of applicable gesture interaction for Active Nudging is two-handed navigation, in which users place both hand as cursors on the screen. The most ideal operation area for two-handed navigation lies around user's shoulders except the area right in front of user's body, because gesture tracking in this area comes with few errors and processing efforts. However, most users find stretching arms forwards much more intuitive than placing hands in certain area on both sides of their bodies. In the earlier experiments performed for this research, it's found extremely difficult to train users to perform effective gestures for the system. Moreover, it caused some confusion in the process as users did not see their hand cursors where they expect (Figure 2.5) to see.

However, in later experiments with Active Nudging applied, instead of directly asking users to do gesture as expected, users were first allowed to operate gestures in front of their bodies as they saw intuitive, but were later gradually nudged toward operating gestures in

the ideal operation area around their shoulders on two sides of their bodies. The point of this experiment lies in that the users could hardly detect the change of their behavior. This is done by dynamically changing the mapping relationship between the user's physical inputs and the inputs actually fed to the system.

Active Nudging is a solution that enables users to adapt to an interaction system quickly and effortlessly. It mends the gap between the user and the interaction system with no observable traces, and a new system could be easily introduced with it integrated. Its potentials of further application to various fields are quite promising. For example, a rehab gesture system in which users start with limited movements but are gradually nudged to move in specific ways to rebuild their muscles without even realizing it. Another example of extensive application would be VR redirected walking, in which users won't run into one another when walking in the same room with Active Nudging introduced in multi-user management.

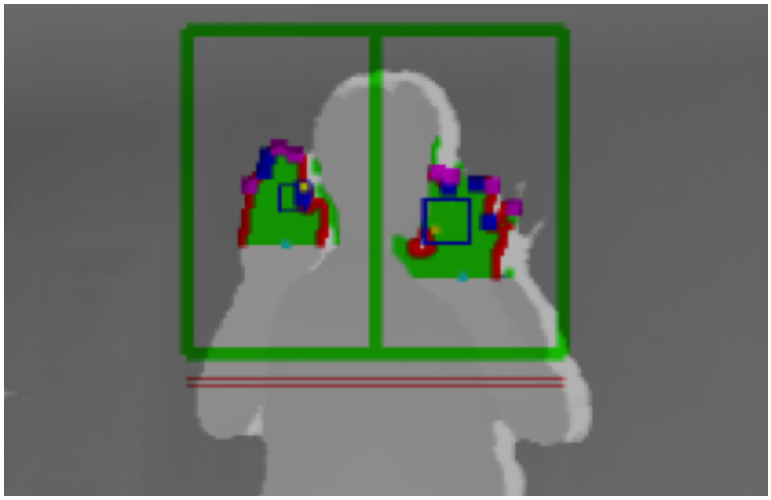
To prove the theory of Active Nudging, we implemented it to the two gesture systems mentioned, Kinect and Leap, having built two-handed navigation system with the former, while pinch gesture activation system with the latter.



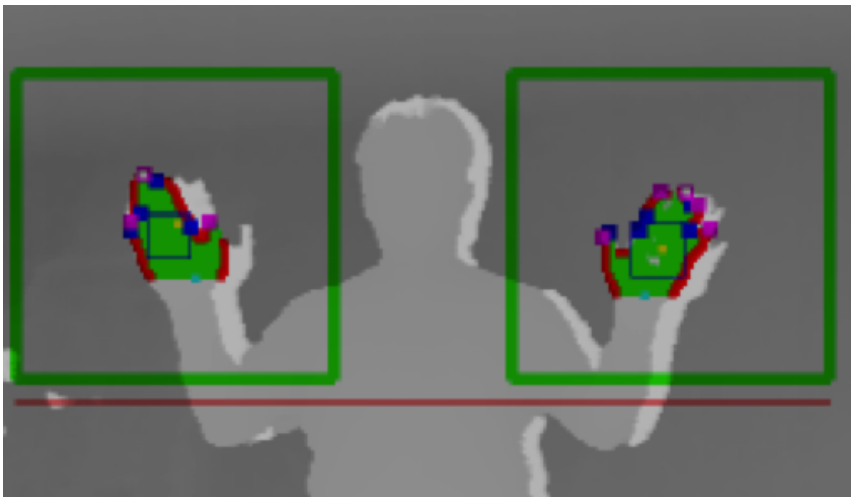
(a)

(b)

Figure 2.5: Confusion caused by a fixed mapping for the engineered interaction (hand coordination mapping bounding boxes are shown in green), as the hands on screen (a) sometimes seem not so intuitive compared with their tracked physical hands (b).



(a)



(b)

Figure 2.6: Example of the Active Nudging for the Two-handed navigation scenario. (a) Initial mapping. (b) Goal mapping (Engineered interaction). Dynamic mapping of Active Nudging shifts user's operation from more intuitive (a) to more recognizable (b) without actual training.

2.4 Formalization

2.4.1 Interaction System State Types

To go further into the formalization of Active Nudging, it's necessary to introduce the concept of interaction system S, the parameter space Θ, and the interaction state (a point in the parameter space) of the system at certain time frame t is $\boldsymbol{\theta}_t$.

When a user interacts with a gesture system S, the system first captures the raw user inputs into parameters for the system. These parameters form a parameter space Θ, and $\boldsymbol{\theta}$ represent the interaction state (as a parameter set) for the user's input $\mathbf{x}$—such as, descriptions of the physical location of the user's specific part(s) of body (the distance between the thumb and index finger). These parameters are interpreted with mapping function f, into visualizations and high level meanings in the interaction (for example, the distance between thumb and index finger being zero is mapped to a pinched gesture). The mapping function itself can further be controlled by a set of parameters $\boldsymbol{\theta}$. Based on characteristics of mapping function, interaction system states come in three types:

Natural Interaction State (static). This is the most basic and straightforward state of an interaction system, as the mapping function in this type of interaction state is static, and does not change with time. The system also tends to "borrow" metaphors from physical world in mapping users' inputs, so the results are usually quite intuitive with little confusion. For example, in a Kinect car racing game, users would be asked to pose as if they are holding a steering wheel, while the gesture is mapped to the in-game "steering wheel." We call this state of interaction system the *Natural* interaction state, denoted $\boldsymbol{\theta}_I$ (as shown in Figure 2.7). Ideally, this could be the system capable of perfect recognition, but in fact current sensor technologies can not yet guarantee that. One of the most common causes of degradation in this system type is occlusion, as some gestures can be very challenging for detection.

Engineered Interaction State (static). Despite the ideal of allowing users to perform intuitive operation, interaction designers often have to create a less intuitive interaction for

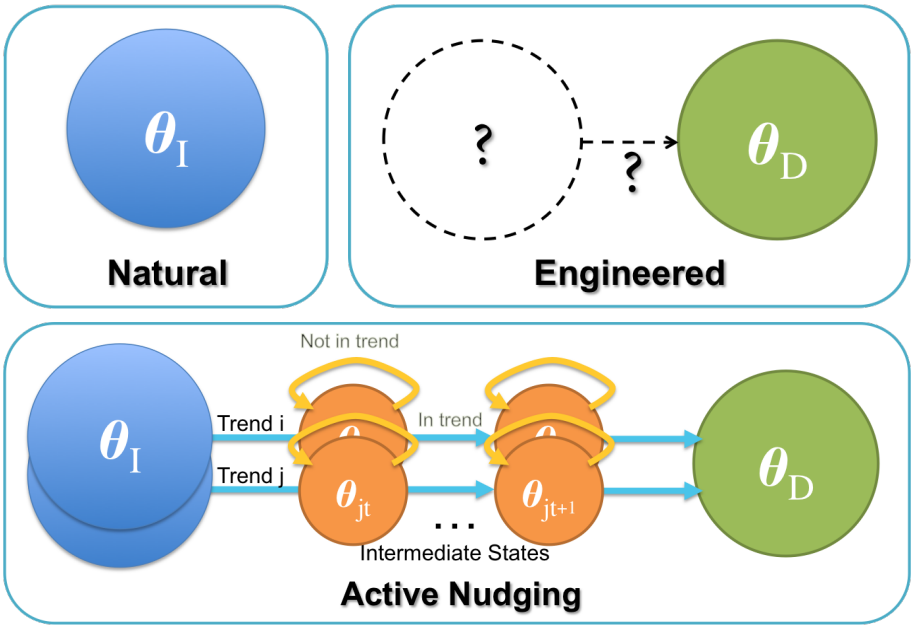


Figure 2.7: Diagrams for different kinds of interaction system states.

reliable user input detection. This type of interaction, *Engineered* interaction state, denoted $\boldsymbol{\theta}_D$ (as shown in Figure 2.7), is designed with effectiveness as primary consideration so that sensor limitations could be overcome. However, on the other hand, it could be less intuitive with correct gestures harder to perform than in natural interaction state, though mapping in both is static and doesn't change with time. To make the expected gestures less difficult to learn, designers would still sometimes use physical metaphors or provide in-depth tutorials to help users to learn the system.

Active Nudging (dynamic). Considering the strength and weakness in the preceding interaction states, *Active Nudging*, a most ideal solution, is proposed in this research. It dynamically adjusts mapping—between user's raw input and the one fed to the system—over time as the user interacts with the interface, and "nudge" the users into doing more and more effective operation. It acts as a hybrid of Natural and Engineered interaction States. By allowing intuitive but less effective input gestures at the beginning and gently nudges the user towards more effective gestures over time (see Figure 2.7), *Active Nudging* comes with

both benefits of the two other states.

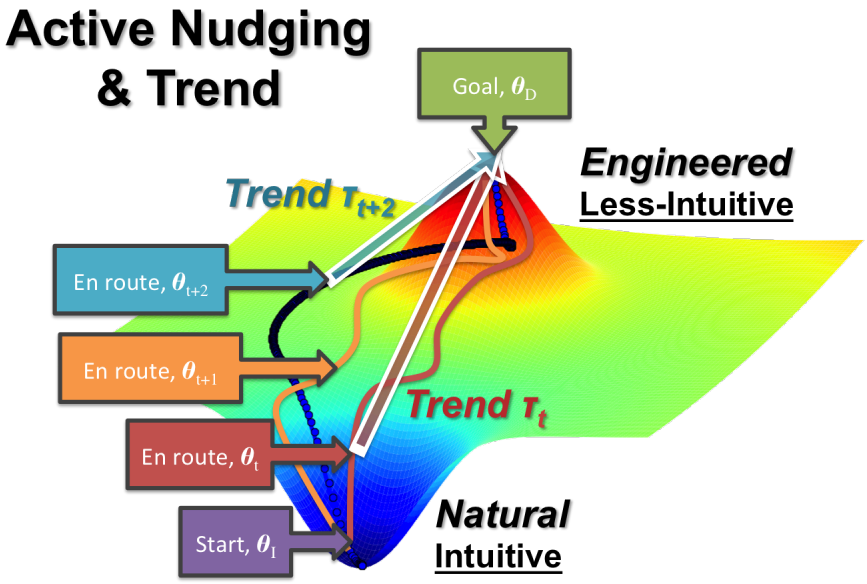


Figure 2.8: Concept visualization of the interaction state migration with Active Nudging.

2.4.2 Adopting Active Nudging in Existing Interaction Systems

There are four steps of adopting Active Nudging in an existing interaction system:

- **Define the interaction system $S = (f, \boldsymbol{\theta})$.** Find the parameters space Θ for the interaction state $\boldsymbol{\theta}$, and come up with the mapping function f that maps the parameters. In the previous pinch example, some of the parameters are 3D coordinate results of tracked fingers.

- **Figure out desired goal state $\boldsymbol{\theta}_D$.** Find out what defines the goal states and perhaps the criteria or constraints. In the pinch example, this is the distance between the fingers we want users to pinch eventually and the best palm facing angle for tracking.

- **Set trend τ.** Find out the essential parameters related to the major changes toward the goal state. In the pinch example, the finger distance and relative palm angle toward camera are the two essential parameters. These two parameters form the trend vector.

- **Find nudging function.** When the user is interacting in-trend, we can use a portion of the parameter vector to adjust the state and use the rest to feed into the mapping function for the system input.

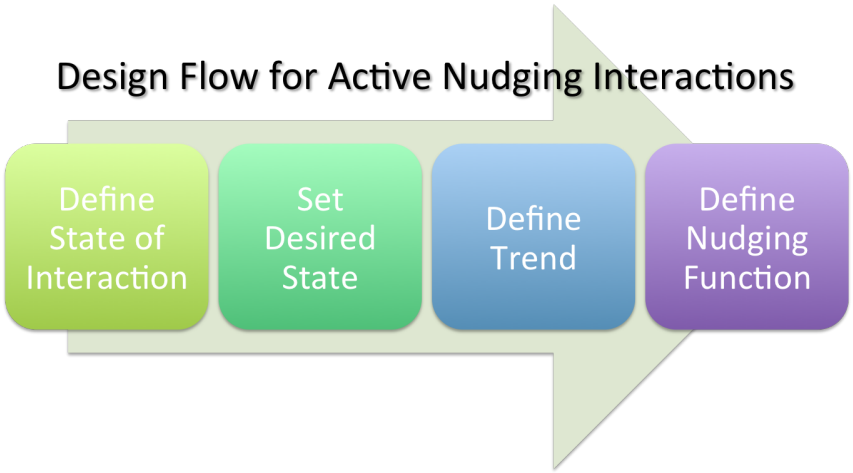


Figure 2.9: Design flow for Active Nudging Interaction.

When a designer wants to introduce Active Nudging into an existing system, some of the previous efforts done can be reused. The desired interaction and the respective parameters can be borrowed from the existing target interaction. The initial natural interaction state will be straightforward, by using a physical metaphor. The major work will be designing the morph of the mapping function.

2.4.3 Defining the Trend

A key element in Active Nudging is the concept of *trend*, which is used to determine if a user's gestures are moving closer to the desired interaction state or not. If so, the parameters of

Table 2.1: Summary of symbol meanings used in the paper.

Symbol	**Meaning**
$S = (f, \boldsymbol{\theta})$	Interaction system
f	Mapping function
$\boldsymbol{\theta}$	Set of parameters, interaction state in general
$\boldsymbol{\theta}_I$	Natural interaction state
$\boldsymbol{\theta}_D$	Engineered interaction (desired goal) state
t	Time stamp
$\boldsymbol{\tau}$	Trend vector
$\mathbf{e}$	Selection vector
α	Nudging coefficient
$\mathbf{x}$	User's physical inputs
$\mathbf{y}$	User's mapped inputs
ϕ	Polarity for trend is positively or negatively correlated to the inputs, $\phi = \pm 1$
λ	Scaling factor to keep state changing unnoticeable, $0 < \lambda < 1$
l	Attenuation for large user inputs, $0 < l < 1$

mapping would be updated accordingly. *Quasi-trend* $\boldsymbol{\delta}$ is the vector in the parameter space from the user's current interaction state toward the desired one, while *trend* $\boldsymbol{\tau}$ is its *vector subset*. This subset is selected based on whether those parameters are essential to the desired operation state. We call those parameters *essential parameters*.

If we consider the parameters as a high dimensional space, every interaction state becomes a coordinate. Consequently, designing of interaction system would consist of figuring out a most effective region for interaction in parameter space. If we simplify the operating ease of the interaction state into a single dimension of height in such space, intuitive interactions are like downward slides that are easy to get to; less intuitive ones, on the other hand, are like upward climbs that are harder to master. This is demonstrated in Figure 2.8.

When the user moves in the parameter space with positive alignment to the trend vector, it's called interacting *in-trend*, and the system will change its interaction state accordingly. For example, if the desired interaction state includes one hand staying in a certain orientation,

the angular velocity towards that orientation would become one of the essential parameters. When the angular velocity toward the desired state is positive, it means the user is acting in-trend. If the interaction state was initially $\boldsymbol{\theta}_t$, now it becomes $\boldsymbol{\theta}_{t+1}$ which is closer to $\boldsymbol{\theta}_D$ than $\boldsymbol{\theta}_t$. Eventually, the user will climb uphill to the desired interaction state (see Figure 2.8).

2.4.4 Defining Parameters

The goal of Active Nudging is to improve the efficiency of a gesture system by adapting users to operating desired gestures quickly and effortlessly. We achieve this through dynamically changing interaction state to make users adapt without being aware of any changes. State mapping function f between a set of the user's raw input $\mathbf{x}_t \in \mathbb{X}$ and and visualized one in the interaction system $\mathbf{y}_t \in \mathbb{Y}$function is changed dynamically for a given time t.

$$\mathbf{y}_t = f(\mathbf{x}_t, \boldsymbol{\theta}_t)$$

where $\boldsymbol{\theta}_t \in \Theta$ is a set of parameters that control the mapping and may change over time. Additionally, we define the *initial state parameters* $\boldsymbol{\theta}_I$ and *desired state parameters* $\boldsymbol{\theta}_D$, along with the *quasi-trend* $\boldsymbol{\delta}_t$, which is the vector points from the current state parameter set to the desired state parameter set, and α_t, the *nudging coefficient*, which will relate the change in the user's inputs to the update of the parameters:

$$\boldsymbol{\delta}_t = \boldsymbol{\theta}_D - \boldsymbol{\theta}_t$$
$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t \boldsymbol{\delta}_t$$

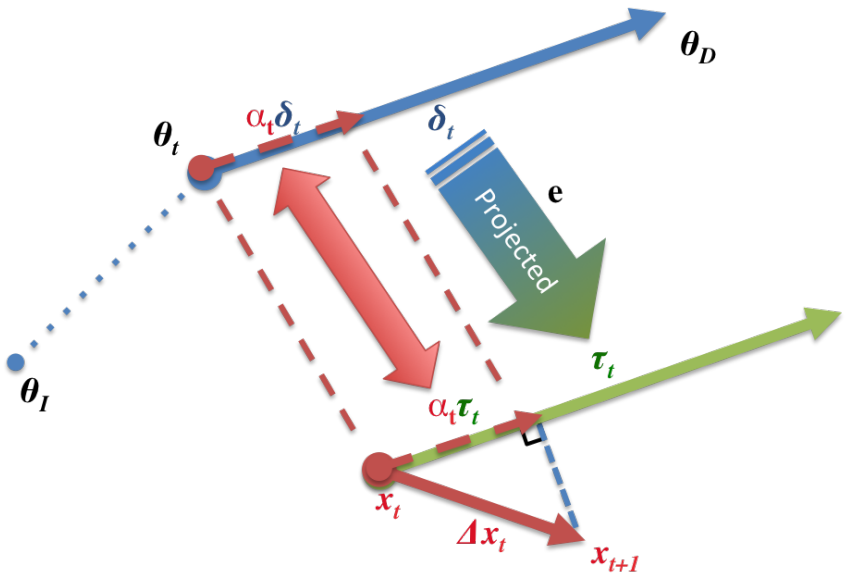


Figure 2.10: Active Nudging interaction state $\boldsymbol{\theta}_t$ moves toward the desired state $\boldsymbol{\theta}_D$ when the user's inputs $\mathbf{x}$ is acting *in-trend*.

When mapping starts as the user starts interacting with the system, $\boldsymbol{\theta}_t$ is set to $\boldsymbol{\theta}_I$; during the interaction, if the user ace in-trend, the state $\boldsymbol{\theta}_t$ is updated until it becomes $\boldsymbol{\theta}_D$. A subset of the parameters $\boldsymbol{\theta}$ is related to the user's input $\mathbf{x}$. So we have *trend* $\boldsymbol{\tau} \in \mathbb{X}$:

$$\boldsymbol{\tau}_t = \mathbf{e} \odot \boldsymbol{\delta}_t$$

where $\mathbf{e}$ is the selection vector to select a subset of the parameters that represent the trend and $\odot$ represents the element-wise production. In addition, the change of the user's input $\triangle\mathbf{x}_t = \mathbf{x}_{t+1} - \mathbf{x}_t$ that is scalar projected onto $\boldsymbol{\tau}_t$ is $\triangle\mathbf{x}_t \cdot \boldsymbol{\tau}_t/|\boldsymbol{\tau}_t|$, where '$\cdot$' denotes dot product. We use this to calculate the α_t–determining the proportion between the projected $\triangle\mathbf{x}_t$ and $\boldsymbol{\tau}_t$ as shown in Fig. 2.10. Hence,

$$\alpha_t \propto \triangle \mathbf{x}_t \cdot \boldsymbol{\tau}_t / |\boldsymbol{\tau}_t|^2$$

when $\alpha_t \in [0, 1]$. Therefore, the full representation of α_t is:

$$\alpha_t = min(0, max(1, \gamma(\triangle \mathbf{x}_t \cdot \boldsymbol{\tau}_t / |\boldsymbol{\tau}_t|^2)))$$

$$\gamma = \begin{cases} \phi\lambda l & \text{if } |\triangle \mathbf{x}_t| > |\boldsymbol{\tau}_t| \\ \phi\lambda & \text{if } |\triangle \mathbf{x}_t| \leq |\boldsymbol{\tau}_t| \end{cases}$$

where ϕ, l, and λ are tuning parameters set by the designer. $\phi = \pm 1$ allows the designer to decide if the trend is positively or negatively correlated to the inputs—that is, if the mapping should be updated when $\triangle \mathbf{x}_t$ points in the same direction as $\boldsymbol{\tau}_t$ or the opposite. l is an scaling factor applied to prevent updating of mapping from finishing in one frame when $|\triangle \mathbf{x}_t| > |\boldsymbol{\tau}_t|$. λ is a constant scaling factor applied to prevent the new mapping from reversing or eliminating the change in the user's input. If we look at the relationship between the virtual inputs under a static mapping $\mathbf{y}'_{t+1} = f(\mathbf{x}_t, \boldsymbol{\theta}_t)$ and under a dynamic mapping $\mathbf{y}_{t+1} = f(\mathbf{x}_t, \boldsymbol{\theta}_{t+1})$, we want to ensure that, for each element i of $\mathbf{y}$:

$$sgn(\mathbf{y}'_{t+1,i} - \mathbf{y}_{t,i}) = sgn(\mathbf{y}_{t+1,i} - \mathbf{y}_{t,i})$$
$$|\mathbf{y}'_{t+1,i} - \mathbf{y}_{t,i}| > |\mathbf{y}_{t+1,i} - \mathbf{y}_{t,i}| > 0$$

given that $|\mathbf{x}_{t+1} - \mathbf{x}_t| > 0$. As mentioned before, part of the user input vector change aligning with the trend is made use of to change nudge state and the rest of the vector. We apply the attenuation factor $\lambda \in (0, 1)$ to ensure these constraints.

Note that if $\phi(\triangle \mathbf{x}_t \cdot \boldsymbol{\tau}_t) \leq 0$ —that is, if the user's motion is not in the desired trend direction—the mapping will not be updated.

For a given application, we define the state with user inputs $\mathbf{x}$, the mapping function f, $\mathbf{y}$, and parameters $\boldsymbol{\theta}$, their spaces $\mathbb{X}$, $\mathbb{Y}$ and Θ, the $\boldsymbol{\theta}_I$ and desired $\boldsymbol{\theta}_D$, the selection vector $\mathbf{e}$ for trend, the tuning parameters λ, ϕ, and l, and α_t, the *nudging coefficient* for the nudging function.

In the following sections, the two scenarios of Active Nudging application will be presented. These two scenarios were chosen because they are fundamental building blocks for any gesture systems, but they are not robust enough as solved problems are.

2.4.5 Formalization: Two-handed Screen Navigation

In this scenario, the goal of application of Active Nudging is to gradually move users' hands from right in front of their bodies (intuitive but very ineffective) to the space around their shoulders on the two side of their bodies (less intuitive but very effective).

As mentioned earlier, Kinect sensor is very susceptible to occlusion: If the user puts both hands together in front of the center of his body, one hand will occlude the other, and both hands will occlude the body, causing ambiguity in detection. To minimize such occlusion, a more effective way to operate with both hands is having two relative coordinate mappings (one for each hand, shown as green boxes in Figure 2.4).

Since the main feedback in this system is visualization, we alter the coordinate mapping function $f(\mathbf{x}, \boldsymbol{\theta}_D)$ for both hand cursors on screen. Initially, the coordinate mapping function $f(\mathbf{x}, \boldsymbol{\theta}_I)$ is determined by the initial location of the user's hands, when first appearing to maintain its relative location on screen.

When the user's hand moves with a vector $\triangle X$ pointing toward idea area for operation, i.e., coherent to the dynamic mapping trend $\boldsymbol{\tau}$, the hand movement appearing on screen is reduced to $\triangle X' = k \triangle X$, where $k = \lambda * l < 1$ to slightly morph $f(\mathbf{x}, \boldsymbol{\theta}_t)$ toward $f(\mathbf{x}, \boldsymbol{\theta}_D)$.

After a few iterations $f(\mathbf{x}, \boldsymbol{\theta}_I)$ will become $f(\mathbf{x}, \boldsymbol{\theta}_D)$, and the user is led to perform the desired operation.

Following is the setup for the rest of the system based on the rules provided.

$$Mapping\ \ function: \quad f(\mathbf{x}_t, \boldsymbol{\theta}_t) = M_\theta \mathbf{x}_t + \mathbf{b}_\theta$$
$$State\ \ parameters: \quad \mathbb{X} = \mathbb{Y} = \mathbb{R}^3, \Theta = \mathbb{R}^4$$
$$Natural/Engineered\ \ states: \quad \boldsymbol{\theta}_I = [\mathbf{p}_i, d_i], \boldsymbol{\theta}_D = [\mathbf{p}_d, d_d]$$
$$Essential\ \ parameter\ \ selector: \quad \mathbf{e} = [1, 1, 1, 0]$$

Mapping function is defined for each hand separately; following is the definition of the mapping for the right hand.

$$M_\theta = \begin{bmatrix} W_x/B_x & 0 & 0 \\ 0 & W_y/B_y & 0 \\ 0 & 0 & W_z/B_z \end{bmatrix}$$

$$B_x = \begin{cases} d_i * (2/3) & \text{if } d_i/2 \leq d_d * 2/3 \\ d_d & \text{otherwise} \end{cases}$$

$$B_y = 3/5 * B_x, \ \ B_z = B_x$$

where $[W_x, W_y, W_z]$ is the projected constant bounding box size in the in-game virtual space on screen, and $[B_x, B_y, B_z]$ is the dynamic bounding box in the sensor space that will nudge the user towards the desired operation. We assume d_d is the width of the desired bounding box. The width is scaled to the height of a man, and designed with consideration of making the system adapt to people with different height.

$$\mathbf{b}_\theta = \begin{bmatrix} W_x(1/2 - p_{t,x}/B_x) \\ W_y(1/2 - p_{t,y}/B_y) \\ W_z(1/2 - p_{t,z}/B_z) \end{bmatrix}$$

$$\mathbf{p}_t = [p_{t,x}, p_{t,y}, p_{t,z}]$$

$$\mathbf{p}_i = \begin{cases} [x_{i,x} - 1/6 * B_x, x_{i,y}, x_{i,z}] & \text{if it is for left hand} \\ [x_{i,x} + 1/6 * B_x, x_{i,y}, x_{i,z}] & \text{otherwise} \end{cases}$$

where $\mathbf{x}_i = [x_{i,x}, x_{i,y}, x_{i,z}]$, and $[p_{t,x}, p_{t,y}, p_{t,z}]$ is the center of the dynamic bounding box in the camera space at time t. $\mathbf{x}_t$ and $\mathbf{y}_t$ are the physical and on-screen 3D coordinates of the user's hand. $\mathbf{p}_i \in \mathbb{R}^3$ is the 3D coordinates of the initial bounding box center and $d_i \in \mathbb{R}^1$ is the distance between the user's hands. $\mathbf{p}_i$ and d_i are only updated when the user's hand enters active detection region from inactive region. $\mathbf{p}_d \in \mathbb{R}^3$ are the 3D coordinates of desired bounding box center and $d_d \in \mathbb{R}^1$ is the width of the desired bounding box.

Given a $\boldsymbol{\theta} = [\mathbf{p}, d]$, M_θ is a matrix that scales the width of the mapping based on d, and $\mathbf{b}_{\boldsymbol{\theta}}$ is an offset vector that moves the center of the mapping based on $\mathbf{p}$. We picked λ as 0.2 in our implementation to ensure the change will be retained and only be partially used for nudging. It was empirically decided, as ideally any number between 0 to 1 should be acceptable. Taking 0.2 in this case was due to consideration for overall system sensibility. Since the $\triangle\mathbf{x}_t$ is positively correlated to the $\boldsymbol{\tau}_t$ we have $\phi = 1$. We set $l = 1$ because the sensor's frame rate is much higher compared with that of the user's movement, so $|\triangle\mathbf{x}_t| < |\boldsymbol{\tau}_t|$ in general.

2.4.6 Formalization: Pinch Gesture Activation Application

In this example, the goal of application of Active Nudging is to turn the users from performing pinch gesture with fingers touching (intuitive but not so effective) to that without fingers touching (effective but not so intuitive). Theoretically, in pinch gesture detection, a system

that recognizes all possible variations of the gesture would be ideal, as it feels intuitive. However, due to occlusion, Leap cannot always provide the reliable detection as expected. Thus, to achieve optimal system efficiency, it's necessary to ask users to perform the gesture in a not so intuitive way (not touching fingers) that more likely comes with valid detection by the system.

Assuming the most ideal distance confidence for detection is around T, and the confidence will drop to unusable when $0 < d < T$. We map the physical distance of the fingers $[T, d_{max}]$ to that on the screen $[0, d'_{max}]$. The mapping can be linear or non-linear.

To achieve the goal, we define $\mathbf{x}_t$ as the difference between distance from the user's thumb to forefinger and the *desired minimum thumb-forefinger distance* d_d. On the other hand, $\mathbf{y}_t$ is the visualized thumb-forefinger distance. The assumed maximal value of thumb-forefinger distance detected with Leap is defined as d_{max}, while minimal value reliably detectable with Leap, d_d. In this application, since some vectors come with only one element, they are used as scalars.

$$Mapping\ \ function:$$

$$f(x_t, \theta_t) = \begin{cases} k_{max}(\frac{x_t + d_d}{d_{max}}) & \text{if } x_t + d_d > d_s \\ k_{max}(\frac{x_t + d_d - \theta_t}{d_s - \theta_t})(\frac{d_s}{d_{max}}) & \text{if } x_t + d_d \leq d_s \end{cases}$$

$$State\ \ parameters: \ \ \mathbb{X} = \mathbb{Y} = \Theta = \mathbb{R}^1$$

$$Natural/Engineered\ \ states: \ \ \boldsymbol{\theta}_I = [0], \boldsymbol{\theta}_D = [d_d]$$

$$Essential\ \ parameter\ \ selector: \ \ \mathbf{e} = [1]$$

where d_s is the threshold for the piece-wise linear function that $0 < d_s < d_{max}$. $\lambda = 0.3$ and $l = 0.25$ are empirically picked in the implementation. Since the $\triangle \mathbf{x}_t$ is negatively correlated to the $\boldsymbol{\tau}_t$, ϕ is set to -1. It's very likelt that $|\triangle \mathbf{x}_t| > |\boldsymbol{\tau}_t|$ occurs, because the

user's movement can be large under the sensor's frame rate.

The mapping begins as a direct mapping from the physical input distance to the virtual input distance, but over time, the visualized fingers will appear closer to each other than the actual ones, even when the user's fingers are not really touching. The non-linear behavior of f makes the visualized thumb-forefinger distance decrease more rapidly than users' fingers do. This would create a feeling of "snap" motion when pinch gesture is done without fingers touching each other(as shown earlier in Figure 4.4).

Table 2.2: Summary of Applications for User Study.

Application	**Two-handed Navigation**	**Pinch Gesture Activation**
Sensor	Kinect	Leap
Restrictions	Prone to error when the hand is overlapped with other body parts	Prone to error when any two fingers are touched or occluded
Goal	Teach user that each hand owns its coordinate mapping	Teach user to pinch without touching fingers
Human Adaptation	Nudging screen coordinate mapping function	Nudging distance between on-screen fingers
Evaluation	**Time:** Total time for task completion **Quality:** Percentage of time that user operates both hands in the intended region	**Time:** Total time for task completion **Quality:** Time percentage of finger distance of the pinch gesture compared to the intended distance **Count:** Total effort for task completion

In both applications presented we let users to know about the tasks (two-handed navigation/pinch) which are the baseline mental models.

2.5 Evaluation Method

In order to discover whether Active Nudging can improve the effectiveness of existing gesture interactions, we ran a study that applies Active Nudging in two fundamental gesture interaction, examples of which were mentioned earlier: two-handed navigation and gesture activation.

2.5.1 Participants

Data for the aforementioned examples were collected from 20 participants recruited from Facebook, of which 8 were male and 12 were female, aging from 18 to 30. Among them there was one heavy gamer, three casual ones, while the rest were non-gamers; none of them were Kinect or Leap users.

2.5.2 Apparatus

Our experiment was conducted with an 15" Macbook Pro with Retina display released in late 2012 along with sensors from the Microsoft Kinect and Leapmotion Leap. We had participants use exactly the same set of devices to ensure environment consistency. The software test bed was an application we wrote in C++, which ran full-screen (shown in Figure 4.4 & 2.11). The software showed the manipulable abstract 3D protein structure and user's hand visualization (either hand cursors or hand skeletons) depending on which sensor was used. The software presented trials, and came with log files containing all the trial data.

2.5.3 Procedure

In our pilot studies, we discovered that the performance of our gesture system with Natural interaction was always far worse than that with Engineered interaction or Active Nudging. Therefore, we compared Active Nudging to the built-in Engineered recognizer (with or without instruction/training) to learn how efficient Active Nudging is.

- *Task for Kinect Two-handed Screen Navigation*

  In this application, participants were asked to perform object manipulation with both hands around the screen with Kinect, and they were allowed to navigate with both hands moving freely (either simultaneously or one-at-a-time). A 3D protein structure

in the center of the screen is displayed along with the hand cursors. The locations we asked participants to move to is shown in Figure 2.11:

- Join & Spread: The participant spreads both hands from two locations towards a joint one, and then back to the original ones. This operation is repeated 5 times.
- Exchange: The participant exchanges the position of two hands. This operation is repeated 5 times.

Each task began with participants standing still with their hands naturally drooping, while we started timing right after they raised their hands. The segments on which the participant's hands hovered would be highlighted, and the highlight moves with the participants' h hands. The timestamps were logged once the participant hovered on the correct segments.

- *Task for Leap Pinch Gesture Activation*

In this application, participants used a Leap Motion Controller to perform a pinch gesture, and were asked to pinch on different objects as quickly as possible until 20 successful pinches were performed. Any object can be chosen to be pinched at as long

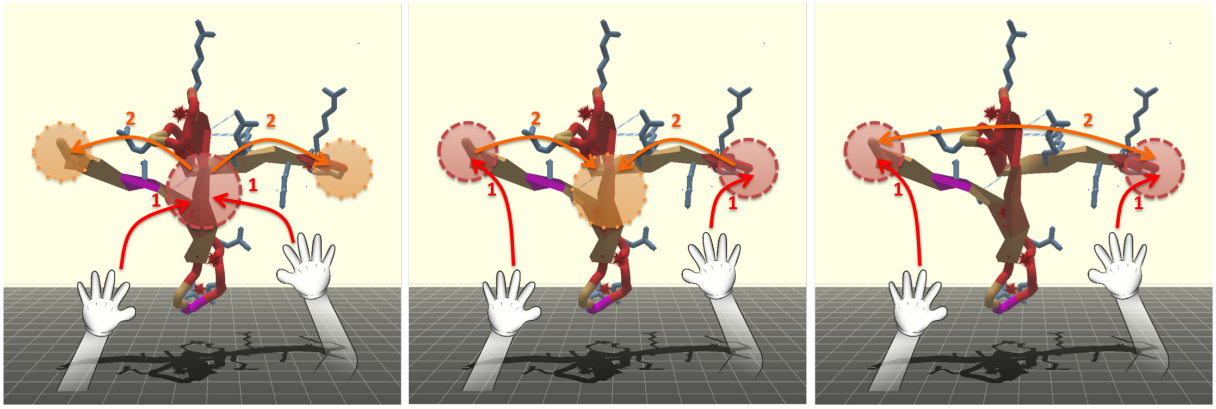


Figure 2.11: Illustration of Kinect tasks: (Left) Spread, (Middle) Join, and (Right) Exchange.

as it is not the current one. An object will be highlighted when a cursor hovers on it, and participants were supervised to ensure hovering is done on the objects before the user makes the pinch gesture. To provide a total number of attempted pinches, participants were also asked to count out loud with each pinch gesture done under supervision.

Each participant performed tasks under three conditions: Active Nudging without instruction (Nudge), Engineered Interaction without instruction (Engineered), and Engineered Interaction with instruction (Instruction). Active Nudging and the other conditions were partially counterbalanced to reduce the chances of possible ordering or learning effects, as Engineered Interaction without instruction (Engineered) condition always came before the Engineered Interaction with instruction (Instruction) (that is, each participant was assigned the conditions in either Nudge $\rightarrow$ Engineered $\rightarrow$ Instruction, Engineered $\rightarrow$ Nudge $\rightarrow$ Instruction or Engineered $\rightarrow$ Instruction $\rightarrow$ Nudge order). After all three conditions were completed, the first condition, whichever one it was, was repeated and the best record was chosen for data.

At the beginning of each condition, participants had up to 5 minutes to practice each task to make sure they fully understood what to do and were capable of completing it, although most participants did not need the full 5 minutes. In the conditions with no instructions, participants were told to practice the tasks and informed the former instructions are no longer valid. In the conditions with instructions, participants were additionally given explicit instructions about how to perform gestures in the desired way. In the Kinect scenario, participants were told they should try spreading their hands apart (instead of making them closer) when they see their hands appear at place they don't expect. In the Leap scenario, participants were told to perform the pinch gesture without touching their index finger and thumb. Participants then performed each task (in a random order) five times with the best performance data recorded.

The main reason for the partially counterbalanced first three conditions is that we want to ensure a "clean" Engineered Interaction without instruction condition (Engineered) exists every time even though we informed participants that the given instructions are valid. Although in of terms learning effects, Engineered Interaction with instruction (Instruction) might seem favorable, there is ample evidence supporting that the Active Nudging condition outperforms both competitors.

In addition, as we repeat the first condition in the experiment, considering the first three and the last three conditions being put together can help reduce the partial counterbalancing, since the only missing condition combination is Instruction → Engineered → Nudge. We also choose the best records from repeated conditions, this will again give us stronger support if Active Nudging condition outperforms both since Engineered conditions (with and without instructions) are more likely to get repeated.

For the Active Nudging condition, the mappings described earlier in Formalization Section were used. For the Engineered Interaction, since it is static, $\boldsymbol{\theta}_t = \boldsymbol{\theta}_D$ and $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t$ were used.

2.5.4 Design and Analysis

The study compared our three Gesture System Types (Engineered, Instruction, and Nudge) in two tasks on Microsoft Kinect (Join & Spread, Exchange) and one task on Leap (20 successful consecutive pinch and grabs). For the Kinect tasks, participants did a total of 800 trials, which consisted of combinations of three conditions (with the first repeated) and two tasks. For the Leap task, participants did a total of 400 trials. Each participant did as many pinch and grab gestures as required to perform 20 successful gestures. Data were analyzed using a non-parametric Friedman test [13]. Pairwise contrasts were performed with Wilcoxon signed-rank tests [39].

The hypotheses about the three different conditions were:

H1. **Active Nudging comes with comparative or better performance than both Engineered interactions even with instruction.** It's hypothesized that a gesture system with Active Nudging applied allows participants to achieve task goals within less time and with fewer tries/retries than that with Engineered Interaction conditions with or without instructions. To evaluate this, we measure *completion time and number of tries*: the total number of tries until task completion serve as overall performance of each condition per task.

H2. **Active Nudging changes users' operating behavior, make them gesture as desired.** It's hypothesized that a gesture system with Active Nudging applied should make participants spend more percentage(defined as *quality*) of the total operation time on the desired gestures. To evaluate this, we measure *quality*: the percentage of total task duration time when the participant is operating as in the interaction space we expected.

Based on our hypothesis, we believed that the Active Nudging condition (Nudge) is more ideal than the Engineered interaction condition (Engineered), and is better than or comparative to the Engineered interaction condition with instruction (Instruction).

2.6 Results

Valid evidence supporting H1 and H2 could be found from the results of the experiments performed for this research.

A summary of results is given in Figure 2.12 and 2.13. The error bars show standard deviations. Pair-wise difference comparison is shown in the link following the bar graph. When the result shows statistical significance, the link is green and marked with *. *** (Significant, $p < .001$), ** (Significant, $p < .005$), * (Significant, $p < .05$).

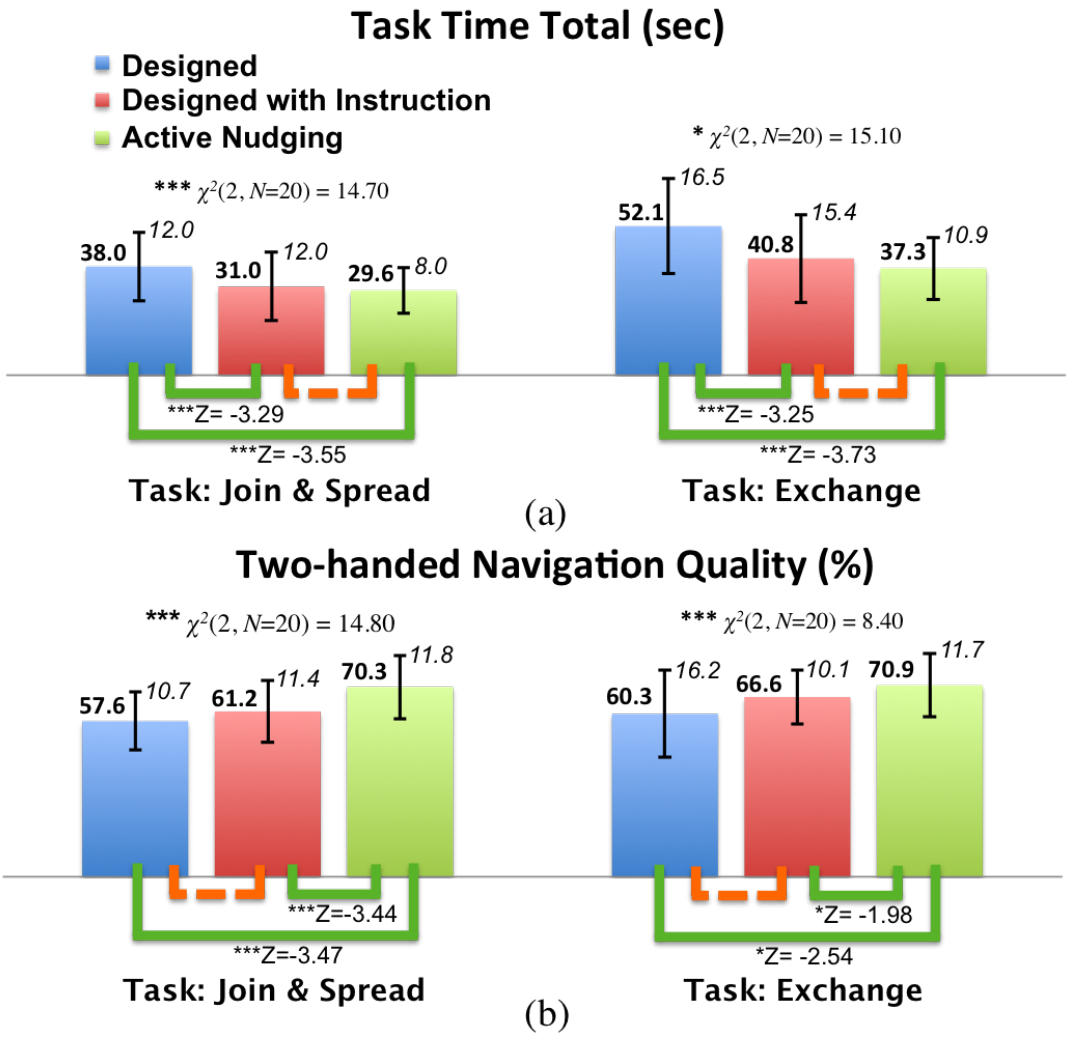


Figure 2.12: (a) Two-handed navigation time for each task; the lesser the better. (b) Two-handed navigation quality for each task; the higher the better. Quality is the percentage of time that user operates both hands in the intended region.

2.6.1 Evaluating Two-handed Screen Navigation

Time of Task: Join & Spread. There was statistically significant difference among the three conditions. Post hoc tests further revealed that there were statistically significant differences between the Engineered condition and the Instruction/Nudge condition.

Time of Task: Exchange. There was statistically significant difference among the three conditions. Post hoc tests further revealed that there were statistically significant differences between the Engineered condition and the Instruction/Nudge condition.

Statistical details are provided in Figure 2.12. As shown in Figure 2.12, the subjects in both Nudge and Instruction conditions were able to complete the tasks faster than those in the Engineered condition with statistical significance. This supports H1.

Quality of Task: Join & Spread. Post hoc tests revealed that the subjects in the Nudge condition took significantly less time than those in the Engineered/Instruction condition.

Quality of Task: Exchange. There was statistically significant difference among the three conditions. Post hoc tests further revealed that there were statistically significant differences between the Nudge condition and both the Engineered/Instruction condition.

Quality-wise, subjects in Nudge condition gestured significantly closer as desired than those in the Engineered/Instruction conditions. This result supports H2.

2.6.2 Evaluating Pinch Gesture Activation

Total Pinch Count. There was statistically significant difference among the total number of pinches across the three conditions. Post hoc tests further revealed that there were statistically significant differences between the Nudge condition and the Engineered/Instruction condition.

Repeated Pinch Count. There was statistically significant difference among the number of repeated pinches across the three conditions. Post hoc tests further revealed that there were statistically significant difference between all three 2-way comparisons.

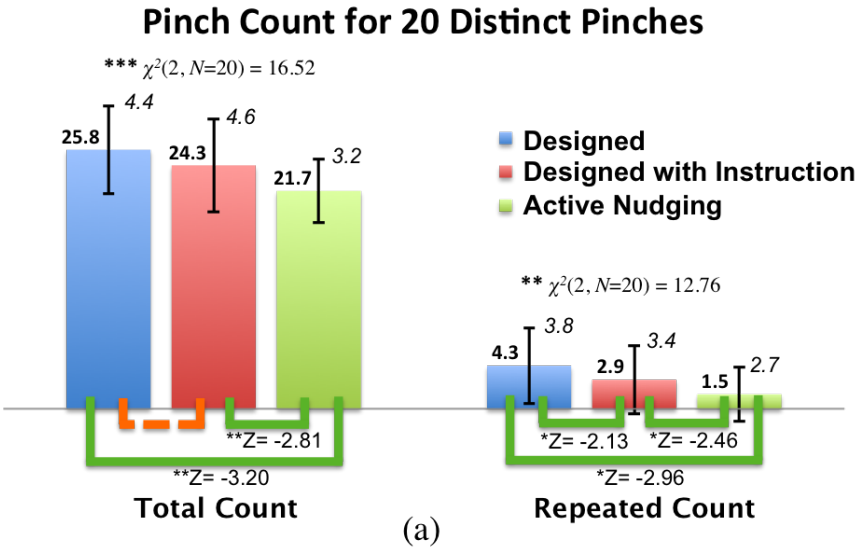
Time Finish First 20 Pinches. Post hoc tests revealed that subjects in Nudge condition took statistically significantly less time than those in both Engineered conditions and Instruction condition.

Time Spent to Finish 20 Distinct Pinches. There was statistically significant difference among the total time spent across the three conditions. Post hoc tests further revealed that there was a statistically significant difference between the Nudge condition and the Engineered/Instruction condition.

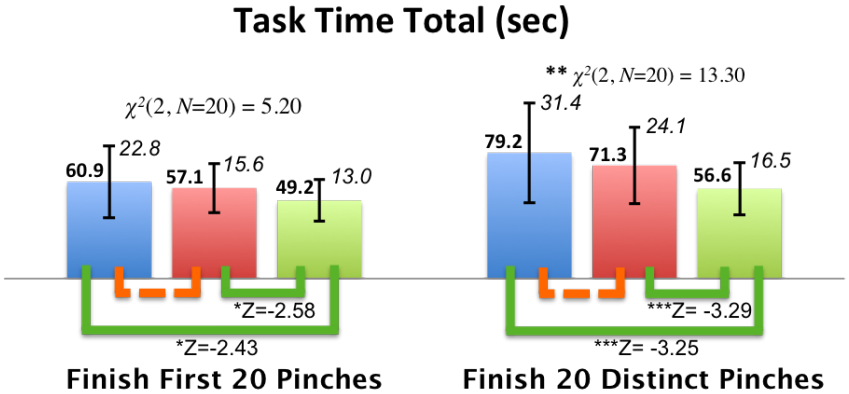
Statistical details are provided in Figure 2.13. As shown in Figure 2.13, subjects in Nudge condition subjects were able to complete the tasks faster with lower error rate than those in both Engineered and Instruction conditions with significant difference. These results support H1.

Pinch Quality. There was statistically significant difference among the pinch quality of the three conditions. Operating quality-wise, there was statistically significant difference between the Nudge condition and the Engineered/Instruction condition.

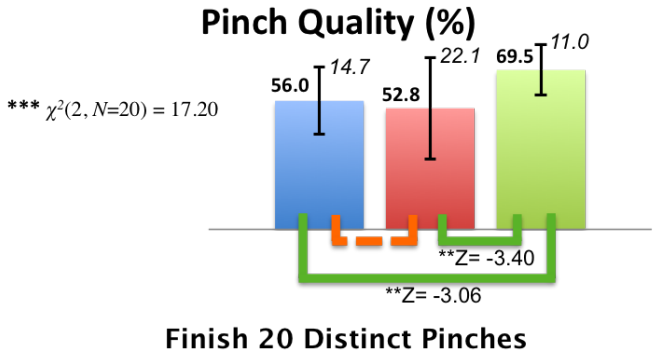
It's concluded that subjects in Nudge condition operate pinch gestures with desired distance between fingers more often than those in the Engineered/Instruction conditions. This result supports H2.



(a)



(b)



(c)

Figure 2.13: (a) Pinch count for the task; the fewer the better. **Total Count** is the total number of pinches necessary to perform 20 successful pinches; **Repeated Count** is the number of unsuccessful pinches. (b) Time total for the tasks; the lesser the better. **Finish First 20 Pinches** is the time to execute 20 pinches, regardless of whether they were successful or not; **Finish 20 Distinct Pinches** is the time to execute 20 successful pinches. (c) Pinch quality for the task; the higher the better. Quality is the average finger distance of the pinch gesture comparing to the intended distance.

2.7 Discussion

The analysis made in this work indicates statistical evidence shows that Active Nudging condition out-performed the other conditions in most of the 2-way comparisons. In addition, collected data support both of our hypotheses. More statistical details are enlisted in Figure 2.12 and 2.13.

Data collected from the two-handed navigation application indicate that operations in Active Nudging condition converged to the desired gestures after around 10 seconds of continuous movement on average, while all subjects in Active Nudging condition test cases converged before the task was finished. However, in the case that the user didn't act in alignment with the trend and the mapping wasn't updated, it could take much longer time for the operation to converge with significantly high error rate.

From Figure 2.12, it's clear that Active Nudging condition come with better quality, and takes comparable or less time for task completion than Engineered condition. The same conclusion could be established from the results of the pinch gesture activation application enlisted in in Figure 2.13.

Six participants in the Kinect example under Active Nudging condition mentioned that they noticed the hand cursors sometimes crossed each other even when the physical hands were not. However, on the other hand, they said they didn't feel confused this happened (as compared to under both Engineered conditions).

In addition, despite two out of twenty subjects preferred other conditions over the ActiveNudging one, they still did better in the Active Nudging condition than in the other ones. This indicates Active Nudging indeed enhances the efficiency of the system in gesture recognition.

2.8 Future Work

The major purpose of this work is to increase gesture system efficiency by adapting users away from ambiguous inputs or occlusions. Future work could include examining effectiveness of this method in creating other types of desired interactions in scenarios like reducing fatigue caused in operation or input orientation. Other scenarios include: guiding multiple users interacting in a small area by dynamically altering the virtual distance between their bodies and thus changing rotation of the user skeleton, so they would end up moving their bodies further away from others' and wouldn't collide into one another.

Also, there are potentially applicable cases in terms of taking specific users into account for adaptation. This could involve recognizing once a user has learned the desired interaction state (after using the system for some time) and transforming to it right away.

However, the current limitation of our approach is that the manual work to identify the desired target parameters. One area of future work should be to enable automatic input recognition for problematic operations (such as physical limitations) and adjust the target mapping to avoid the situations. This concept also comes with great potentials in future AR/VR, in which visual feedback plays a significant role. As virtual reality systems become more prevailing, new possibilities for Active Nudging may emerge, and this approach may be even more effective when the users cannot see their bodies directly, based on the user's sense of proprioception.

2.9 Conclusion

In this work, we present *Active Nudging* can facilitate gesture interactions, formalize a unifying formulation, and demonstrate with two different types of applications. Unlike conventional approaches, in which either the system has to adapt to users, a system with *Active Nudging* applied encourages user gestures into desire ones without any explicit instructions or training. With this approach, users' physical gestures are changed with better performance

in the virtual interface, and the users' behaviors were shifted elegantly without explicit explanation or instruction.

Bibliography

[1] 3GearSystems. 2012. 3GearSystems. http://www.threegear.com. (2012).

[2] David L. Akers. 2007. Observation-based design methods for gestural user interfaces. In *CHI '07 extended abstracts on Human factors in computing systems - CHI '07*. ACM Press, New York, New York, USA, 1625. DOI:http://dx.doi.org/10.1145/1240866.1240868

[3] Fraser Anderson and Walter F. Bischof. 2013. Learning and performance with gesture guides. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*. ACM Press, New York, New York, USA, 1109. DOI:http://dx.doi.org/10.1145/2470654.2466143

[4] Luca Ballan, Aparna Taneja, Jürgen Gall, Luc Van Gool, and Marc Pollefeys. 2012. Motion capture of hands in action using discriminative salient points. In *ECCV*. Springer.

[5] Olivier Bau and Wendy E. Mackay. 2008. OctoPocus: A Dynamic Guide for Learning Gesture-based Command Sets. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology (UIST '08)*. ACM, New York, NY, USA, 37–46. DOI:http://dx.doi.org/10.1145/1449715.1449724

[6] Thomas Baudel and Michel Beaudouin-Lafon. 1993. Charade: Remote Control of Objects Using Free-hand Gestures. *Commun. ACM* 36, 7 (July 1993), 28–35. DOI:http://dx.doi.org/10.1145/159544.159562

[7] Andrew Bragdon, Arman Uguray, Daniel Wigdor, Stylianos Anagnostopoulos, Robert Zeleznik, and Rutledge Feman. 2010. Gesture Play: Motivating Online Gesture Learning with Fun, Positive Reinforcement and Physical Metaphors. In *ACM International Conference on Interactive Tabletops and Surfaces (ITS '10)*. ACM, New York, NY, USA, 39–48. DOI:http://dx.doi.org/10.1145/1936652.1936661

[8] Andrew Bragdon, Robert Zeleznik, Brian Williamson, Timothy Miller, and Joseph J. LaViola, Jr. 2009. GestureBar: Improving the Approachability of Gesture-based Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 2269–2278. DOI:http://dx.doi.org/10.1145/1518701.1519050

[9] Gry Casiez, Daniel Vogel, Ravin Balakrishnan, and Andy Cockburn. 2008. The Impact of Control-Display Gain on User Performance in Pointing Tasks. *HUMANCOMPUTER INTERACTION* 23, 3 (2008), 215–250.

[10] Andrea Colaço, Ahmed Kirmani, Hye Soo Yang, Nan-Wei Gong, Chris Schmandt, and Vivek K. Goyal. 2013. Mime: Compact, Low Power 3D Gesture Sensing for Interaction with Head Mounted Displays. In *UIST*.

[11] Dustin Freeman, Hrvoje Benko, Meredith Ringel Morris, and Daniel Wigdor. 2009. ShadowGuides: Visualizations for In-situ Learning of Multi-touch and Whole-hand Gestures. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '09)*. ACM, New York, NY, USA, 165–172. DOI:http://dx.doi.org/10.1145/1731903.1731935

[12] Scott Frees and G. Drew Kessler. 2005. Precise and Rapid Interaction Through Scaled Manipulation in Immersive Virtual Environments. In *Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality (VR '05)*. IEEE Computer Society, Washington, DC, USA, 99–106. DOI:http://dx.doi.org/10.1109/VR.2005.60

[13] Milton Friedman. 1937. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *J. Amer. Statist. Assoc.* 32, 200 (1937), 675–701. http://www.jstor.org/stable/2279372

[14] Sarah Gallacher, Eliza Papadopoulou, Nick K. Taylor, and M. Howard Williams. 2013. Learning user preferences for adaptive pervasive environments: An incremental and temporal approach. *ACM Transactions on Autonomous and Adaptive Systems* 8, 1 (April 2013), 1–26. DOI:http://dx.doi.org/10.1145/2451248.2451253

[15] Sean G. Gustafson, Bernhard Rabe, and Patrick M. Baudisch. 2013. Understanding Palm-Based Imaginary Interfaces: The Role of Visual and Tactile Cues when Browsing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*. ACM Press, New York, New York, USA, 889. DOI:http://dx.doi.org/10.1145/2470654.2466114

[16] Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. 2011. OmniTouch: Wearable Multitouch Interaction Everywhere. In *UIST*.

[17] Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics* 6, 2 (1979), 65–70.

[18] Dun-Yu Hsiao, Seth Cooper, Christy Ballweber, and Zoran Popović. 2014. User Behavior Transformation through Dynamic Input Mappings. In *Proceedings of the Ninth International Conference on the Foundations of Digital Games (FDG '14)*.

[19] Takeo Igarashi and John F. Hughes. 2001. A suggestive interface for 3D drawing. In *Proceedings of the 14th annual ACM symposium on User interface software and technology - UIST '01*. ACM Press, New York, New York, USA, 173. DOI:http://dx.doi.org/10.1145/502348.502379

[20] David Kim, Otmar Hilliges, Shahram Izadi, Alex D Butler, Jiawen Chen, Iason Oikonomidis, and Patrick Olivier. 2012. Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 167–176.

[21] LeapMotion. 2013. LeapMotion. (2013). https://leapmotion.com/about.

[22] Microsoft. 2011. Kinect. (2011). http://www.microsoft.com/en-us/kinectforwindows/.

[23] Miguel A. Nacenta, Yemliha Kamber, Yizhou Qiang, and Per Ola Kristensson. 2013. Memorability of pre-designed and user-defined gesture sets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*. ACM Press, New York, New York, USA, 1099. DOI:http://dx.doi.org/10.1145/2470654.2466142

[24] Nintendo. 2006. Wii. (2006). https://wii.com.

[25] Russell Owen, Gordon Kurtenbach, George Fitzmaurice, Thomas Baudel, and Bill Buxton. 2005. When it gets more difficult, use both hands: exploring bimanual curve manipulation. In *Proceedings of Graphics Interface 2005 (GI '05)*. Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 17–24. http://dl.acm.org/citation.cfm?id=1089508.1089512

[26] Ondrej Polacek, Martin Klima, Adam J. Sporka, Pavel Zak, Michal Hradis, Pavel Zemcik, and Vaclav Prochazka. 2012. A comparative study on distant free-hand pointing. In *Proceedings of the 10th European conference on Interactive tv and video (EuroiTV '12)*. ACM, New York, NY, USA, 139–142. DOI:http://dx.doi.org/10.1145/2325616.2325644

[27] Dorothy Rachovides, James Walkerdine, and Peter Phillips. 2007. The conductor interaction method. *ACM Transactions on Multimedia Computing, Communications, and Applications* 3, 4 (Dec. 2007), 1–23. DOI:http://dx.doi.org/10.1145/1314303.1314312

[28] Sharif Razzaque, Zachariah Kohn, and Mary C Whitton. 2001. Redirected Walking. (2001).

[29] Rajinder Sodhi, Hrvoje Benko, and Andrew Wilson. 2012. LightGuide: Projected Visualizations for Hand Movement Guidance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 179–188. DOI:http://dx.doi.org/10.1145/2207676.2207702

[30] Peng Song, Wooi Boon Goh, William Hutama, Chi-Wing Fu, and Xiaopei Liu. 2012. A handle bar metaphor for virtual object manipulation with mid-air interaction. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*. ACM, 1297–1306.

[31] Sony. 2010. SonyMove. http://us.playstation.com/ps3/playstation-move/. (2010).

[32] Srinath Sridhar, Antti Oulasvirta, and Christian Theobalt. 2013. Interactive Markerless Articulated Hand Motion Tracking Using RGB and Depth Data. In *ICCV*.

[33] Evan A. Suma, Seth Clark, Samantha Finkelstein, Zachary Wartell, David Krum, and Mark Bolas. 2011. Leveraging Change Blindness for Redirection in Virtual Environments. In *IEEE Virtual Reality*. 159–166. http://people.ict.usc.edu/~suma/papers/suma-vr2011.pdf (19% acceptance rate).

[34] Qi Sun, Li-Yi Wei, and Arie Kaufman. 2016. Mapping Virtual and Physical Reality. *ACM Trans. Graph.* 35, 4, Article 64 (July 2016), 12 pages. DOI:http://dx.doi.org/10.1145/2897824.2925883

[35] D. Vanacken, A. Demeure, K. Luyten, and Karin Coninx. 2008. Ghosts in the interface: Meta-user interface visualizations as guides for multi-touch interaction. In *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*. 81–84. DOI:http://dx.doi.org/10.1109/TABLETOP.2008.4660187

[36] Robert Wang, Sylvain Paris, and Jovan Popović. 2011. 6D hands: markerless hand-tracking for computer aided design. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 549–558.

[37] Robert Y. Wang and Jovan Popović. 2009. Real-time Hand-tracking with a Color Glove. In *ACM SIGGRAPH 2009 Papers (SIGGRAPH '09)*. ACM, New York, NY, USA, Article 63, 8 pages. DOI:http://dx.doi.org/10.1145/1576246.1531369

[38] Yangang Wang, Jianyuan Min, Jianjie Zhang, Yebin Liu, Feng Xu, Qionghai Dai, and Jinxiang Chai. 2013. Video-based hand manipulation capture through composite motion control. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 43.

[39] Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin* 1, 6 (12 1945), 80–83.

[40] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. 2009. User-defined Gestures for Surface Computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1083–1092. DOI: http://dx.doi.org/10.1145/1518701.1518866

[41] Bruno Zamborlin, Frederic Bevilacqua, Marco Gillies, and Mark D'inverno. 2014. Fluid gesture interaction design: Applications of continuous recognition for the design of modern gestural interfaces. *ACM Transactions on Interactive Intelligent Systems* 3, 4 (Jan. 2014), 1–30. DOI:http://dx.doi.org/10.1145/2543921

[42] Wenping Zhao, Jinxiang Chai, and Ying-Qing Xu. 2012. Combining marker-based mocap and RGB-D camera for acquiring high-fidelity hand motion data. In *Proceedings of the ACM SIGGRAPH/eurographics symposium on computer animation*. Eurographics Association, 33–42.

Chapter 3

MACHINE ADAPTATION: PROACTIVE SENSING FOR IMPROVING HAND POSE ESTIMATION

We propose a novel sensing technique called *proactive sensing*. Proactive sensing is a mechanism that continually repositions a camera-based sensor to improve hand pose estimation, and a scheme that effectively learns how to move the sensor to improve pose estimation confidence while requiring no ground truth hand poses. We demonstrate this concept with a low-cost rapid swing arm system built around the state-of-the-art commercial sensing system Leap Motion. The results of our user study show that proactive sensing helps estimate user hand poses with higher confidence compared to both *static* and *random* sensing. We further present an online model update for Leap that enhances system performance.

3.1 Introduction

User interface with bare-hand-in-air gesture operation has become quite popular and gained massive attention in both academia and industry. In particular, real-time, fine-grained 3D hand pose estimation is crucial for a variety of applications, such as immersive virtual reality, assistive technologies, robotics, home automation, and gaming.

However, the design of efficient real-time 3D hand pose estimation is extremely challenging, as many elements should be taken into consideration, such as numerous degrees of hand movement due to large number of joints, different hand shapes, sizes, and texture of possible covering materials (e.g., gloves). Earlier systems that gained success have to go with hands with gloves or markers, and were cumbersome and inaccurate. More recent works focus on camera-based systems that reduce the demand of mandatory hand augmentation, and

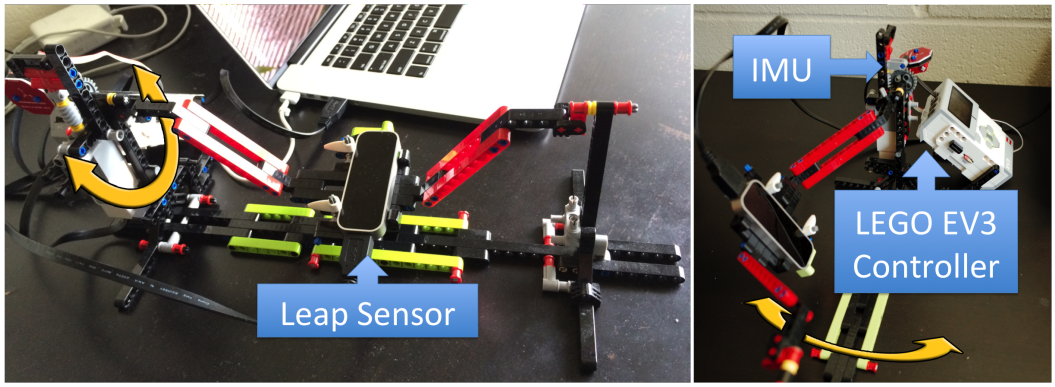


Figure 3.1: The swing arm setup for proactive sensing. Left: at 30°. Right: at −60°. The arm can swing around one axis, as shown in yellow arrows.

hence allow a more natural user interaction compared with the predecessors. Nevertheless, most modern systems still struggle with high estimation failure rate due to the ambiguity of finger locations in certain gestures and self-occlusion among different parts of the hand, and such challenges are quite common even in commercial systems circulating in the market. Currently, many approaches address these issues by setting constraint the operation environments. For instance, they mostly support only front-facing camera scenarios or require multiple cameras to reduce the inaccuracycaused by occlusions.

In this work, we present a novel system called *proactive sensing* (Fig. 3.1), which addresses the issues mentioned by allowing a camera-based sensor with a single *viewpoint* to keep moving to find a good sensing position. There may be multiple cameras in the sensor, but they share a similar, single viewpoint (as with Leap Motion or Kinect systems) and are thus susceptible to similar occlusions. This approach allows users to move their hand freely during interactions and does not require a cumbersome setting with multiple viewpoints. Our approach was inspired by the observation that different hand poses can be robustly estimated under different viewpoints. However, instead of setting up a multiple viewpoint system to capture all viewpoints at all times, we propose to learn a user's operating habits and dynamically predict the most ideal viewpoint for estimation of the user's coning hand

poses.

Our prototype system consists of a circular moving swing arm that allows the sensor to move and access any viewpoint. We show that an existing commercial sensor can benefit from our moving sensor system. In particular, we evaluated 17 users on the task of playing the protein folding game *Foldit* [3]. Our proposed system consistently improved 3D hand pose estimation confidence across different users compared to both a state-of-the-art static sensor solution and a random moving sensor solution. Our system also has the ability to adapt to the habits of each specific user, so that the more they use it, the more robust the system becomes.

3.2 Related Work

Many of the various real-time hand pose estimation methods recently been proposed are summarized in the following paragraph.

RGB image. Hand pose estimation using monocular RGB images has been a challenge (see Erol *et al.* [5] for a summary). Many of the early works (e.g., Wu *et al.* [23] and de La Gorce *et al.* [4]) operated offline processing of recorded sequences, while the work of Heap and Hogg [8], a deformable model, is an exception, which produces estimation of hand poses at $\sim$10 Hz. However, there were many challenges such as complex poses, background changes, and occlusions. Recently, several relatively more real-time systems have been proposed. Song *et al.* [18] proposed an efficient multi-stage random forest based hand gesture recognition system on mobile devices, and later *et al.* [17] further proposed a system of directly mapping 2D color images to 3D hand positions and gestures.

Depth image. With the development of consumer depth cameras such as Kinect, a number of new methods have been proposed for reliably estimating hand poses in real-time. Oikonomidis *et al.* [14] presented a generative method for hand pose estimation at 15 Hz on a GPU. However, it comes with some limitations such as it requires an initial hand pose, and

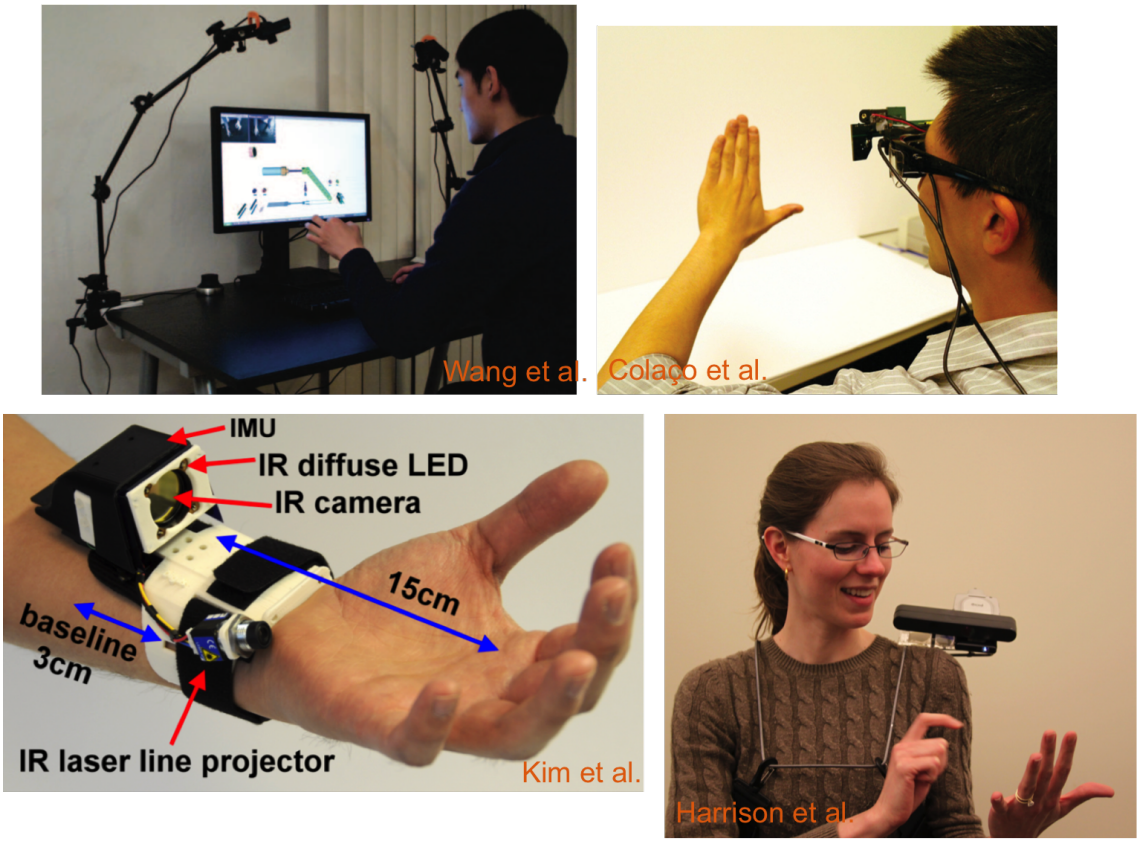


Figure 3.2: Static gesture systems.

cannot recover from estimation failures. Qian *et al.* [15] removed these limitations, buthis system still requires clear caption of fingertips to start. Keskin *et al.* [9] proposed a novel framework for real-time hand pose estimation on a CPU. Sharp *et al.* [16] proposed a system with an enhanced reinitializer to handle estimation failure more reliably, and used temporal information to achieve smooth processing and accurate results. Sridhar *et al.* [19] further achieved 50 Hz using a CPU only implementation.

Multiple viewpoints. To obtain reliable and high-quality hand pose estimation results, many methods have been based on data captured with multiple-camera rigs. Most of the works (e.g., Wang *et al.* [21], Zhao *et al.* [24], and Ballan *et al.* [1]) operates offline. One of the exceptions is Sridhar *et al.* [20], which comes with a rig with five RGB cameras and a

time-of-flight sensor to estimate a user's hand at ∼10 Hz. Neverthless, the setup is expensive and complicated, and thus not so feasible in general use scenarios.

Wearable solutions. To mitigate failures due to occlusion, Kim *et al.* [10] proposed that the user wears a low power depth camera on the wrist. Similarly, Colaço *et al.* [2] explored a low-power, head mounted 3D gesture sensing solution. Harrison *et al.* [7] also proposed wearing a RGBD hand gesture recognition system, but on the shoulder. However, these methods are designed based on certain assumptions—such as not including holding objects or restricting hand positions—that prevent users from using their hands freely.

In this work, we allow a single-viewpoint sensor to proactively search for an ideal sensing position according to the user's behavior. Essentially, our low-cost single viewpoint solution shares the advantages an expensive multiple-viewpoint solution.

3.3 System Overview

Our proactive sensing platform and the underlying algorithm used to automatically control the sensor's movement is described in this session. It is demonstrated that our proactive sensing system can improve hand pose estimation performance of a state-of-the-art Leap Motion sensor [11]. To prove this concept, we built a sensor platform with one degree of freedom.

3.3.1 Swing Arm

We used LEGO bricks and a MINDSTORMS EV3 Intelligent Brick to construct a swing arm, adding one degree of freedom to the sensor (between $-90°$ to $90°$ with top speed of $25°$ per second, shown in Fig. 3.1). An inertial measurement unit (IMU) was attached to the sensor to measure the position of the swing arm. The EV3 Intelligent Brick was used as an interface to drive the motor and read the data from the attached IMU.

When the sensor moves, the position of the hand in the sensor coordinate system changes.

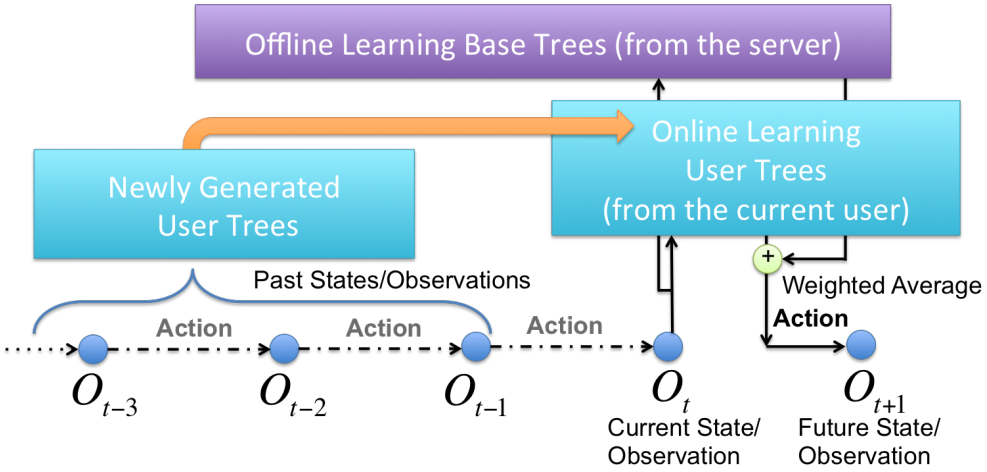


Figure 3.3: Online action learning and prediction.

We converted from sensor coordinates to world coordinates, and used the IMU mounted on the swing arm to calculate the initial transformation between the sensor and world coordinates. To mitigate sensor error and drift, we updated the transformation using Iterative Closest Point (ICP) to align the current hand pose to the initial hand pose. We found the IMU and ICP worked well in our case, though more sophisticated methods such as those of those of Newcombe *et al.* [13] could be applied.

3.3.2 Action Learning

Our proactive sensing system determines an action to enhance performance in estimating a user's hand pose. We take a learning approach to learn a function $f : \mathbb{O} \rightarrow \mathbb{A}$, where $\mathbb{O}$ is the set of possible observations and $\mathbb{A}$ is the set of possible actions. Our system learns to improve estimation *confidence*. Confidence is provided by Leap Motion at each timestamp, as a single score from 0 (worst) to 1 (best) indicating confidence in the current pose estimate. We chose confidence as it does not require ground truth and is highly correlated with accuracy (discussed below).

Approximate kNN Classification

k-Nearest Neighbor (kNN) is a widely used algorithm with desirable properties for several reasons. First, kNN can be significantly sped up during testing by incorporating Approximate Nearest Neighbor (ANN) search [12]. Second, training a KD-tree for ANN search is also efficient. We train multiple KD-trees in an online fashion utilizing data collected on individual behavior.

We take a supervised approach to train the action function f. Our training data includes:

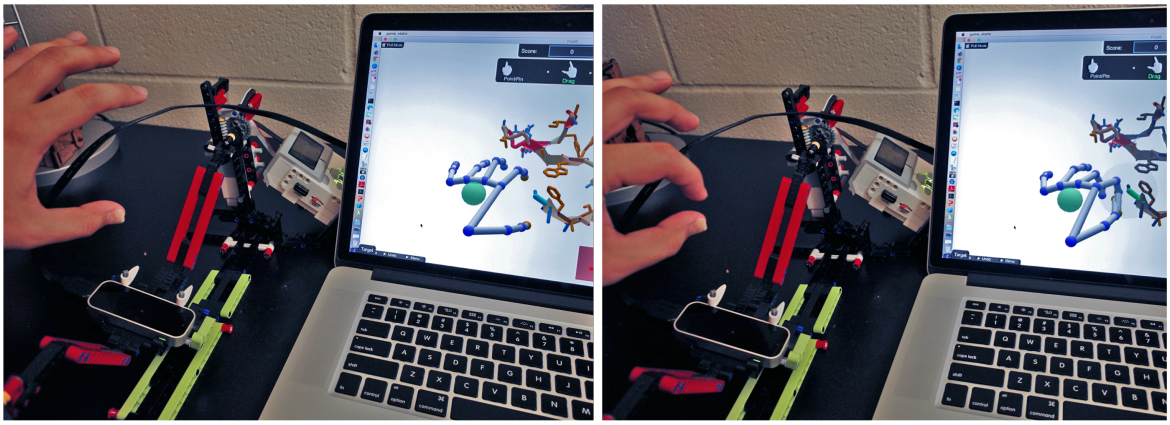
- Observations $\mathbb{O}$ containing features from the hand pose estimation system. We start with 84 features: hand type (left/right), swing arm angle, confidence, palm position (3), velocity (3), and orientation (3), and positions of the pose (24×3). This is reduced to 10 features as described below.
- The best action $a \in \mathbb{A}$ where $\mathbb{A} = \{left, right, still\}$.

Offline Training

We first generated training data in an offline stage and trained a number of KD-trees, called *base trees*, to predict actions for all users when first using our system.

The training data set was built by sampling 12 sets of gestures, both static (7 different hand postures from fully open to fully closed) and dynamic (grab in the air using 5 different hand postures). Each gesture was performed while the swing arm swung through the whole range 10-20 times.

To pre-compute the best action for each training entry, first we used kNN to find neighbors of the entry (within 10 swing degrees). Each neighbor voted for the action toward higher weighted confidence (weighted by confidence and distance). At runtime, if the current confidence is above 0.93, the action *still* is used; otherwise, the same kNN lookup is used, using the pre-computed actions for each neighbor, to compute the action taken. For classification,



(a) Open (b) Closed

Figure 3.4: Task overview. Subjects can modify the structure freely with any hand poses they want to pinch.

we searched among all available trees to get the three closest training samples for each tree, and accumulated the votes for each action as the prediction score.

Online Training

Since the confidence of the hand pose estimation is continuously observed, we proposed to collect training data when each user interacts with the system, and train a set of new KD-trees, called *user trees*, in an online fashion (every 2,000 samples). The scores of the predicted actions from both base and user trees are fused by weighted averaging of the scores into *combined trees*.

The block diagram showing the online training procedure and action prediction fusion is shown in Fig. 3.3.

Dimensionality Reduction for Speed-up

By collecting training data at 60 fps and each observation with84 dimensions, our data and KD-trees took considerable amount of memory (10 MB per 2,000 samples). We used several

dimensionality reduction techniques to reduce the feature dimensions from 84 to 10. First, we used feature selection techniques to select the top few features (i.e., $1 \sim 4$ features). We simply exhaustively tested different combinations with cross validation, and found the three best features, which were the hand type (left/right), sensor angle, and the tracking confidence from the sensor. For the remaining 81 features, we applied PCA to reduce dimensions to 7, such that the final total dimensions were $3 + 7 = 10$. After dimension reduction, it took 10KB per 2,000 samples, and the precomputation time was reduced from about 30 minutes to around 20 seconds for a small dataset ($\sim$6.7MB). On a 2.3 GHz Intel Core i7 laptop, a new classification tree can usually be trained in 30 seconds, and the query among 500 trees can be done faster than 30 Hz.

3.4 User Study

We carried out a user study to examine the effect of different types of sensing on pose estimation confidence, and conducted a within-subjects experiment with 17 subjects recruited randomly from college, most of whom were novices at using the Leap Motion sensor.

We used three different *sensing types* to utilize the degrees of freedom of the swing arm:

- `PROACTIVE`: The swing arm moves the sensor with the algorithm we purposed, which produces the most ideal action estimation based on current the observation.
- `STATIC` (baseline 1): The swing arm was always static, which is the most common sensing type adopted in many state-of-the-art systems (e.g., [11, 16]).
- `RANDOM` (baseline 2): The swing arm moved to a random angle from time to time. This was to confirm that moving the sensor meaningfully is important (rather than randomly moving the sensor, which might occasionally avoid occlusion to achieve better confidence).

The task given to subjects was to use a pinch gesture to do 3D object manipulation

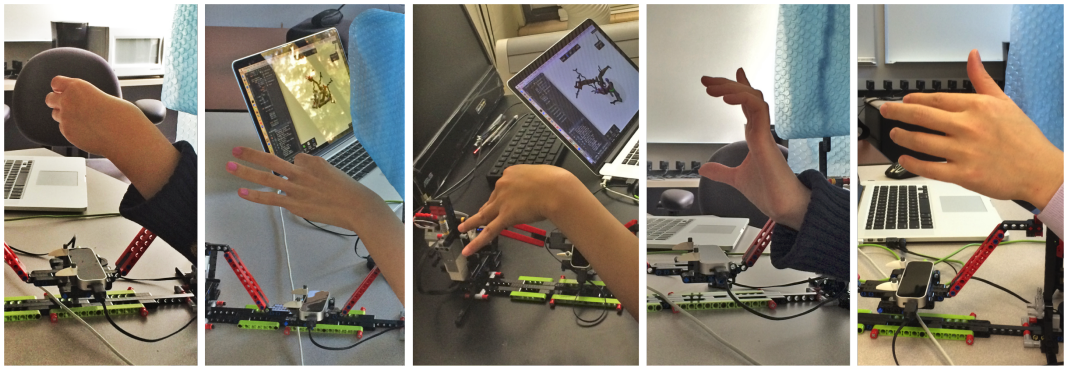


Figure 3.5: Pinch gesture variations among test subjects.

(Fig. 3.4). Subjects manipulated a highly deformable 3D object's shape by dragging its parts to different locations. Subjects were asked to continuously manipulate the 3D structure with no provided goal. Note that any recognizable pinch gestures were allowed. As the result, we observed a large variation of pinch gestures (Fig. 3.5). All subjects were asked to use their left hand in the entire experiment. In addition, in order to eliminate users' attention from the swing arm, we presented the subjects from seeing their acting hand and the sensor by blocking the view. Also, we had the subjects listen to white noises to ensure they could not hear the sound from the motor.

For each subject, the experiment consisted of three *tree types*: base, user, and combined. Within each tree type, there were three *task iterations*. Each task iteration took 2 minutes. During each task iteration, we switched between the three *sensing types* without any interruption (i.e., 40 seconds per sensing type); the swing arm rapidly reset to 0° between each sensing type. We counterbalanced for any learning effects, and randomized the tree type order; however, if user or combined trees were to be first, subjects did an additional base trees first (to get user data to construct the other tree types), and we compared the better of the two base trees uses. The sensing type order was random in each task iteration for each subject. At the beginning of each tree type, subjects were given up to 5 minutes to practice the task to make sure they understood it fully, and were capable of completing it, although

most subjects did not need the full time.

3.4.1 Aggregate Confidence

We examined two outcomes of aggregate confidence by subject, and took the overall (mean) confidence, collected at 60 Hz, within each 40 second use of a sensing type. Then, for each sensing type, we took the *mean* across all uses as the *mean overall confidence* and the *best* across all uses as the *best overall confidence*. Since our outcome variables were not all normally distributed, we used the non-parametric Friedman Test for repeated measures, along with subsequent Wilcoxon Rank Sum Tests and the Bonferroni correction, to identify statistically significant 2-way comparisons.

Sensing Types

For best overall confidence, there was a statistically significant difference, $\chi^2(2,N=17)=17.294$, $p<.001$, among the three sensing types. In post-hoc tests, it's found that there was no statistically significant difference between `STATIC` (M = .767, SD = .121) and `RANDOM` (M = .770, SD= .095), Z=-.118, p=.906. However, there was a statistically significant difference between `STATIC` and `PROACTIVE` (M = .847, SD = .085), Z=-3.479, $p < .001$. `RANDOM` was also statistically significantly different from `PROACTIVE`, Z=-3.053, $p = .002$. Fig. 3.6 shows comparison between three sensing types across all subjects.

For mean overall confidence, we also found a statistically significant difference among the three sensing types, $\chi^2(2, N = 17) = 10.706, p = .005$. Post-hoc tests revealed that although there was no statistically significant difference between `STATIC` (M = .629, SD = .112) and `RANDOM` (M = .621, SD= .105), Z=-.544, p=.586, there was a statistically significant difference between `STATIC` and `PROACTIVE` (M = .668, SD = .104), Z=-2.959, $p = .003$, and `RANDOM` was statistically significantly different from `PROACTIVE`, Z=-2.675, $p = .007$. Fig. 3.7 shows comparison between three sensing types across all subjects.

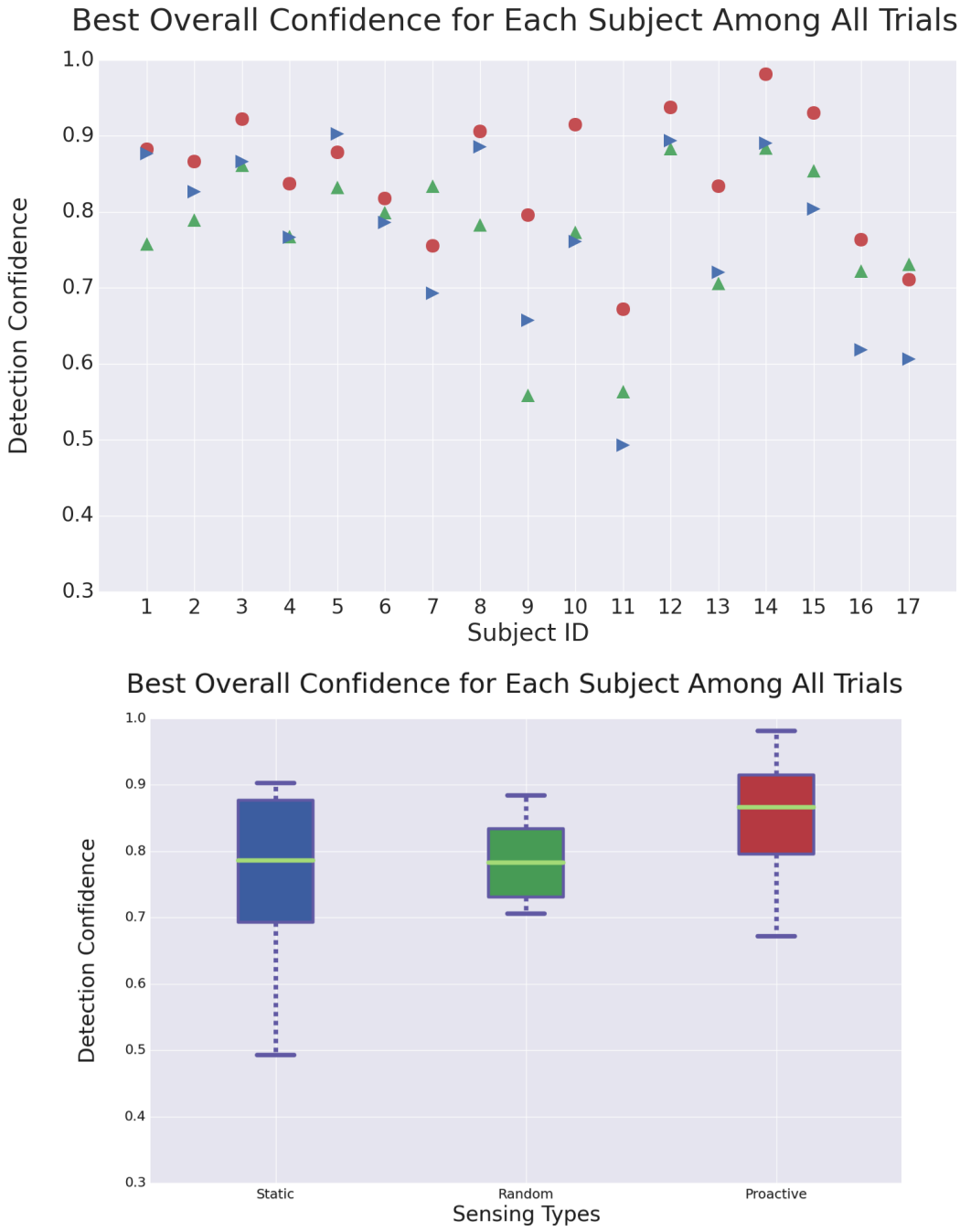


Figure 3.6: Best overall detection confidence for each subject among all task iterations.

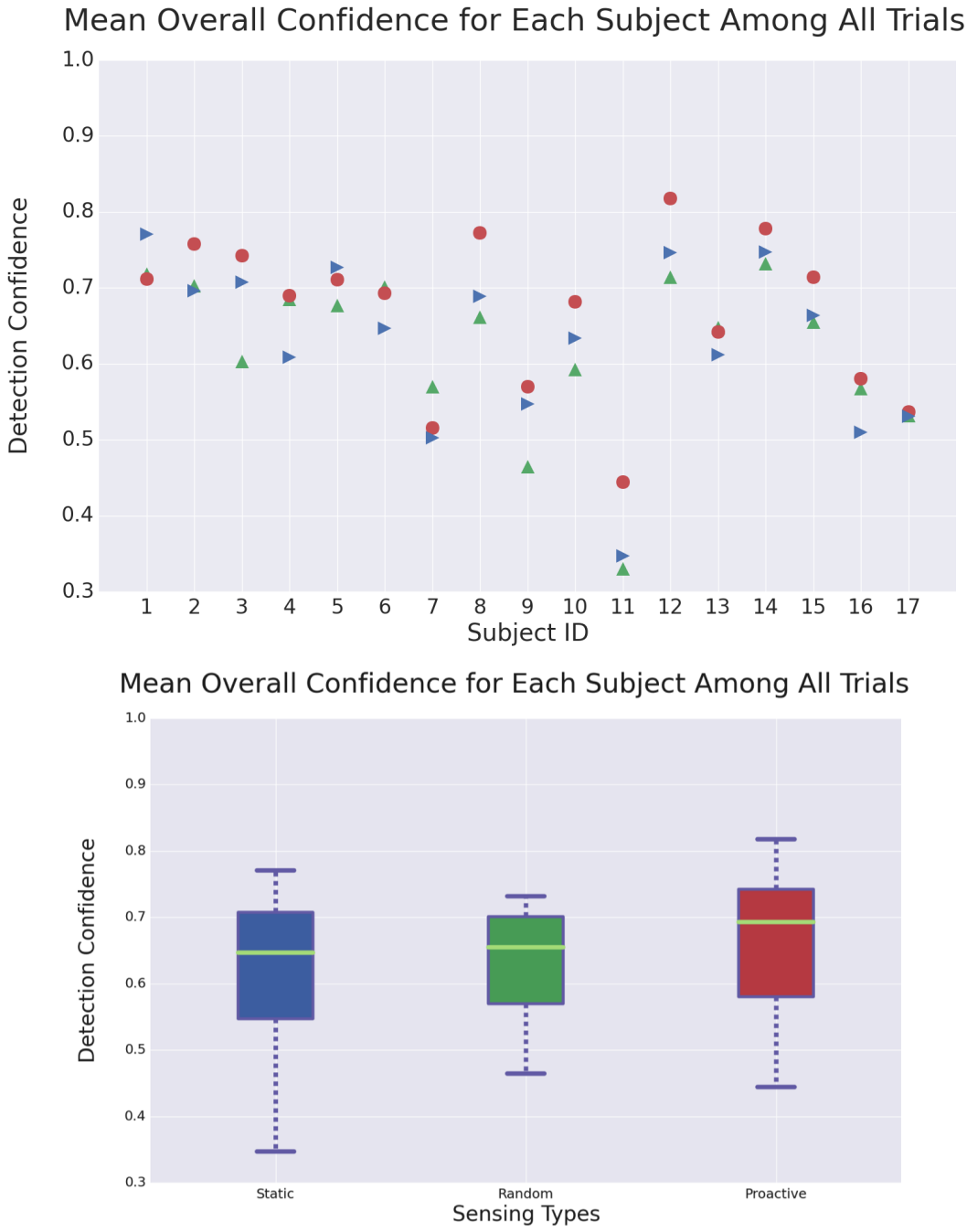


Figure 3.7: Mean overall detection confidence for each subject among all task iterations.

Tree Types

During our experiment, we captured confidence of the different tree types and averaged across all subjects. User trees obtained a 2.7% higher average, and 3.8% higher best, relative confidence; however, theses results were not statistically significant.

Confidence	Base	User	Combined
Mean overall	0.658	0.676	0.672
Best overall	0.737	0.766	0.763

3.4.2 Confidence and Accuracy Correlation

We assume confidence is highly correlated to accuracy, and hence improving confidence results in improved accuracy. To prove this theory, we gathered data with a technique similar to an existing work on the improvement of Leap Motion sensor accuracy [6, 22]. We used an artificial hand to perform known poses when the sensor moved. In the global coordinate system, we measured accuracy for each pose and compared to confidence at each time-stamp. We tested six sets of static hand poses (fully open, fully closed, and partially open, each with 0 and 45 degree palm facings). The cross-correlations (with no shifting) between confidence and accuracy for each pose were 0.939, 0.971, 0.955, 0.931, 0.924, and 0.890.

3.5 Conclusion and Future Work

In this work, we demonstrated the potential for proactive sensing aimed to improve sensing performance by dynamically positioning the sensor. The core contribution is an effective learning scheme that requires no ground truth hand poses. Also, we demonstrated a pipeline aimed to build a classification system that can learn and improve itself overtime with speed efficiency, and scalability.

Currently, our method could be used to predict the most ideal action to take for the current hand pose, while ignoring that the hand pose can change. By acting at a high frame rate, our sensor can still move to an ideal position to catch up with the movement of the hand. We believe a model that explicitly predict the movement of a hand can further improve the performance of the recognition system. Additionally, the proactive sensing algorithm could be applied to other movable platforms with high degrees of freedom. For instance, we imagine that a drone could serve as the ultimate movable platform withour algorithm,as it could learn the most ideal position to capture human postures, including hand and body poses.

Bibliography

[1] Luca Ballan, Aparna Taneja, Jürgen Gall, Luc Van Gool, and Marc Pollefeys. 2012. Motion capture of hands in action using discriminative salient points. In *ECCV*. Springer.

[2] Andrea Colaço, Ahmed Kirmani, Hye Soo Yang, Nan-Wei Gong, Chris Schmandt, and Vivek K. Goyal. 2013. Mime: Compact, Low Power 3D Gesture Sensing for Interaction with Head Mounted Displays. In *UIST*.

[3] Seth Cooper, Firas Khatib, Adrien Treuille, Janos Barbero, Jeehyung Lee, Michael Beenen, Andrew Leaver-Fay, David Baker, Zoran Popović, and Foldit Players. 2010. Predicting protein structures with a multiplayer online game. *Nature* 466, 7307 (2010), 756–760.

[4] Martin de La Gorce, David J. Fleet, and Nikos Paragios. 2011. Model-Based 3D Hand Pose Estimation from Monocular Video. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 33, 9 (Sept 2011), 1793–1805.

[5] Ali Erol, George Bebis, Mircea Nicolescu, Richard D. Boyle, and Xander Twombly. 2007. Vision-based Hand Pose Estimation: A Review. *Computer Vision and Image Understanding* 108, 1-2 (Oct. 2007), 52–73.

[6] Jože Guna, Grega Jakus, Matevž Pogačnik, Sašo Tomažič, and Jaka Sodnik. 2014. An Analysis of the Precision and Reliability of the Leap Motion Sensor and Its Suitability for Static and Dynamic Tracking. *Sensors* 14, 2 (2014), 3702.

[7] Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. 2011. OmniTouch: Wearable Multitouch Interaction Everywhere. In *UIST*.

[8] Tony Heap and David Hogg. 1996. Towards 3D hand tracking using a deformable model. In *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*. 140–145.

[9] Cem Keskin, Furkan Kıraç, Yunus Emre Kara, and Lale Akarun. 2012. Hand Pose Estimation and Hand Shape Classification Using Multi-layered Randomized Decision Forests. In *ECCV*.

[10] David Kim, Otmar Hilliges, Shahram Izadi, Alex D Butler, Jiawen Chen, Iason Oikonomidis, and Patrick Olivier. 2012. Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 167–176.

[11] Leap Motion, Inc. 2014. (2014). `http://leapmotion.com/`

[12] Marius Muja and David G. Lowe. 2014. Scalable Nearest Neighbor Algorithms for High Dimensional Data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 36 (2014).

[13] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. 2015. DynamicFusion: Reconstruction and Tracking of Non-Rigid Scenes in Real-Time. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015).

[14] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. 2011. Efficient model-based 3D tracking of hand articulations using Kinect.. In *BMVC*.

[15] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. 2014. Realtime and robust hand tracking from depth. In *CVPR*.

[16] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, Daniel Freedman, Pushmeet Kohli, Eyal Krupka, Andrew Fitzgibbon, and Shahram Izadi. 2015. Accurate, Robust, and Flexible Real-time Hand Tracking. CHI.

[17] Jie Song, Fabrizio Pece, Gábor Sörös, Marion Koelle, and Otmar Hilliges. 2015. Joint Estimation of 3D Hand Position and Gestures from Monocular Video for Mobile Interaction. In *CHI*.

[18] Jie Song, Gábor Sörös, Fabrizio Pece, Sean Ryan Fanello, Shahram Izadi, Cem Keskin, and Otmar Hilliges. 2014. In-air Gestures Around Unmodified Mobile Devices. In *UIST*.

[19] Srinath Sridhar, Franziska Mueller, Antti Oulasvirta, and Christian Theobalt. 2015. Fast and Robust Hand Tracking Using Detection-Guided Optimization. In *CVPR*.

[20] Srinath Sridhar, Antti Oulasvirta, and Christian Theobalt. 2013. Interactive Markerless Articulated Hand Motion Tracking Using RGB and Depth Data. In *ICCV*.

[21] Yangang Wang, Jianyuan Min, Jianjie Zhang, Yebin Liu, Feng Xu, Qionghai Dai, and Jinxiang Chai. 2013. Video-based hand manipulation capture through composite motion control. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 43.

[22] Frank Weichert, Daniel Bachmann, Bartholomäus Rudak, and Denis Fisseler. 2013. Analysis of the Accuracy and Robustness of the Leap Motion Controller. *Sensors (Basel, Switzerland)* 13, 5 (05 2013), 6380–6393.

[23] Ying Wu, John Y Lin, and Thomas S Huang. 2001. Capturing natural hand articulation. In *ICCV*, Vol. 2.

[24] Wenping Zhao, Jinxiang Chai, and Ying-Qing Xu. 2012. Combining marker-based mocap and RGB-D camera for acquiring high-fidelity hand motion data. In *Proceedings of the ACM SIGGRAPH/eurographics symposium on computer animation*. Eurographics Association, 33–42.

Chapter 4

MUTUAL ADAPTATION: COMBINING HUMAN ADAPTATION AND MACHINE ADAPTATION

The major consideration in conventional human-computer interaction design is how to make the computer "intelligent." That is, to make it possible for the computer to process massive information feeds in complicated scenarios, such as natural language, gestural interfaces or voice commands, make sense of it, and react accordingly as expected. On the other hand, as computing devices have been more and more portable invarious form factorsand no longer restricted to specific locations, the difference in the form of human-computer and interpersonal interaction is gradually diminished. As the result, just like in the case of interpersonal communication, ambiguity inevitably emerges in human-computer communication, and compromises the performance of the system in handling information.

However, it is possible to overcome the challenge of ambiguity, and one of the sources ofinspirations of a solution lies in interpersonal interaction. When people interact with one another, several tactics are often applied to reduce ambiguity and hence improve the efficiency of communication: one could either adapt oneself by improving information receptionlike leaning toward the other or putting palms to the ears, or ask the counterpart to adapt, such as to speak louder or to repeat what was said. Similar concepts have been demonstrated in the previous chapters in this work: the method of *Human Adaptation* which nudges the user in the interaction toward the ideal operation for system recognition, helps expedite the user adapt particularly when the ideal operation is not so intuitive or easy for training. On the other hand, the method of *Machine Adaptation*, with the ideal operation (such as viewing angle) figured out constantly, provide a target range for *Human Adaptation*.

Combining *Human Adaptation* with *Machine Adaptation*, and there comes the solution of *Mutual Adaptation*, which we believe can further eliminate the factors that compromise the efficiency of human-computer interaction. Furthermore, we will include a supporting preliminary experiment and results in this chapter.

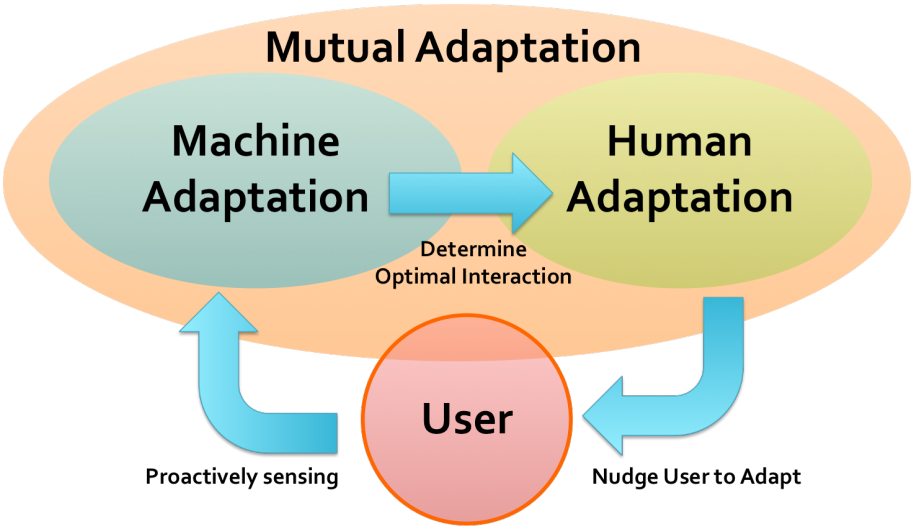


Figure 4.1: Mutual Adaptation relationship diagram.

In this chapter, we will first explore the possibility of extending existing *Human Adaptation* experiments to include *Machine Adaptation* as automatic target setter for *Human Adaptation*. Furthermore, we will apply the concept in a VR redirected walking application as another illustration of how the concept could be applied to various user scenarios.

4.1 Smart Nudging: Automatic Target Setting for Mutual Adaptation

Mutual Adaptation, the solution we propose in this chapter consists of two concepts: a system could be designed to guide the users toward ideal operation and hence adapt to the system; meanwhile, it couldalso exploit the range of ideal operation as the target which it guides the users toward. In the previous chapters, we demonstrated how a system with *Human Adaptation* utilized helps the users adapt to itself quickly. We used two gestural interaction

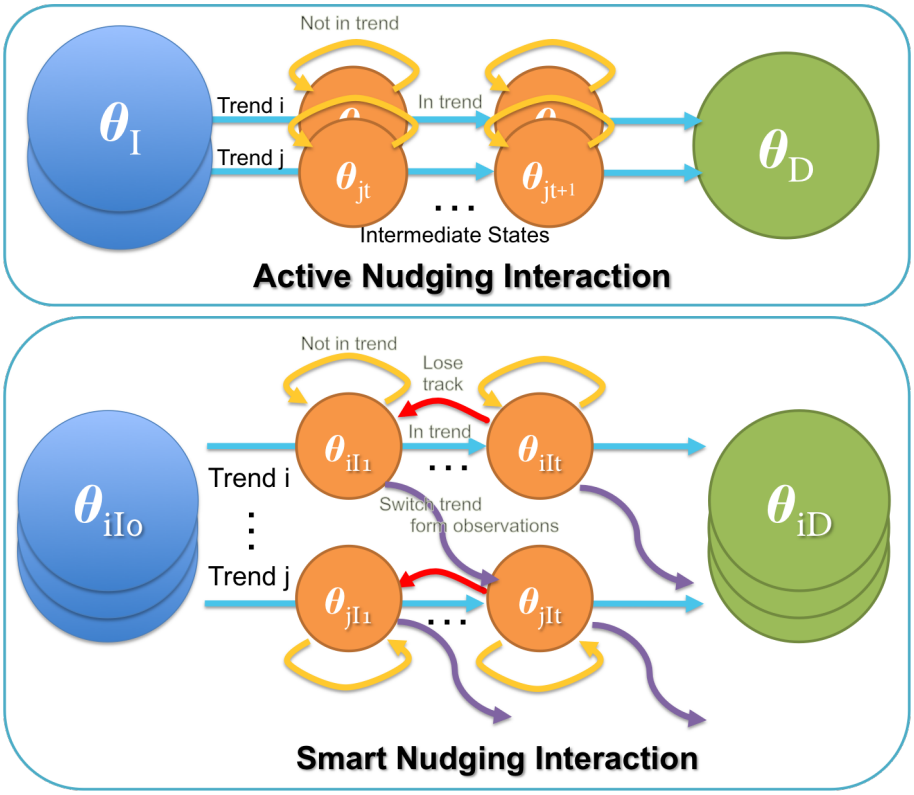


Figure 4.2: Overview of the Smart Nudging Interaction comparing to the Nudging Interaction.

scenarios–two-handed navigation and gesture activation–to show that applying this concept can effectively nudge people into operation for system recognition, even when the operation is not so intuitive. Nudging Interaction tolerates high error rate of the input, while gently push the user towards giving more and more reliably recognized input over time with the dynamically and adaptively updated mapping (see Figure 4.2).

However, although the nudging process is dynamic, these demonstrated scenarios' goal interaction states were usually predefined and static, and are potentially not so adaptive for every user, particularly in the scenarios with massive scale of user base. In addition, in termsof defining a range of ideal operation, there's always much space for improvement. Considering these factors, it would be worthwhile to automate some parts of the process of

target operation definition, as it would lead to constant improvement in the effectiveness of the range defined. Hence, we want to explore the possibility of automating the search of the target interaction state, and our first step is to create "flexible goals" by allowing goal setting rules creation in the goal finding mechanism. We call this Smart Nudging. Without a fixed handpicked goal, Smart Nudging can really work corresponding to how each of the the users interacts with the system, while some design efforts in system enhancement could be saved.

We believe that Smart Nudging Interaction exceeds its predecessor, as it learns from users inputs overtime to automatically figure out and adjust the destination interaction space ($\boldsymbol{\theta}_{iD}$). Hence, different users may end up at different goal states that fit the very user's intuition (as shown in Fig. 4.3). We will try some learning techniques and some non-linear optimization methods to see the process can be automated, and evaluate the resulting interaction states by comparing the scenarios to those with prior handpicked goals.

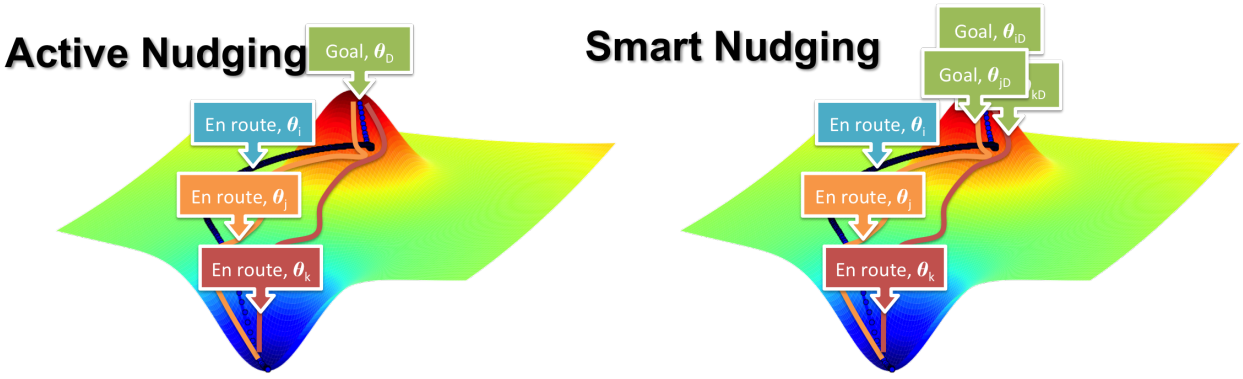


Figure 4.3: Nudging Interaction and Smart Nudging Interaction illustrated.

In the chapter of *Human Adaptation*,we seehow it manages to transform user interactive behavior toward what's ideal for the system. Furthermore, with the active platform of *Machine Adaptation* integrated, a system would be able to improve itself in terms of detecting human interaction. It is natural to combine both adaptation methods to bring about *Mutual*

Adaptation, which optimizes the interface effectiveness in human-machine interaction. In the following paragraphs, we'd like to explore the optional performance of this combined system.

Also, it is also likely that an active platform for destination interaction space definition isn't always accessible, while improving the active platform will certainly increase the maintenance cost and brings impact to scalability. Seeing this, we come up with an approach that allows *Machine Adaptation* to serve as an active platform in a system. We believe that this is feasible with*Human Adaptation* leveraged:

A system with *Human Adaptation* can guide the users to interact from interaction state $\boldsymbol{\theta}_{iI}$ toward targeted interaction state $\boldsymbol{\theta}_{iD}$,while that with *Machine Adaptation* can figure out the targeted $\boldsymbol{\theta}'_{iD}$ most ideal for the system. Hence, here comes the the question: if the $\boldsymbol{\theta}'_{iD}$ are learned and collected withcrowdsourcing the interaction, is it possible to lead the users to their $\boldsymbol{\theta}'_{iD}$ without the presence of active platform? We believe that it is indeed possible to retroactively apply the learnt model to the visualizations which gradually nudge users to their $\boldsymbol{\theta}'_{iD}$. For example, if we know that the best way of detection is to roll the sensor to the left by 30 degree, we can alter the visualization to lead the user to roll the hand to the right by 30 degrees instead of moving the sensor. This seems like an ultimate form for the adaptation method for gestural interface, and we'd like to further evaluate its performance.

As virtual reality systems become more pervasive, new potentials for Adaptation Interaction are emerging: as users cannot see their body directly, the approach we propose can be even more effective due to the fact that visualizations are the only source of operation feedback, which could even be further altered to manipulate user's sense of proprioception. This is one of the potentials of our proposed concept in AR/VR application scenarios.

In the following sections, we will demonstrate the application of *Mutual Adaptation* in two scenarios: Two-handed Navigation with Leap and VR redirected walking assistance.

4.2 Example: Two-handed Navigation with Leap

In this section, we will illustrate the details of our implementation of *Mutual Adaptation* upon the two-handed navigation example used previously in the *Human Adaptation* chapter.

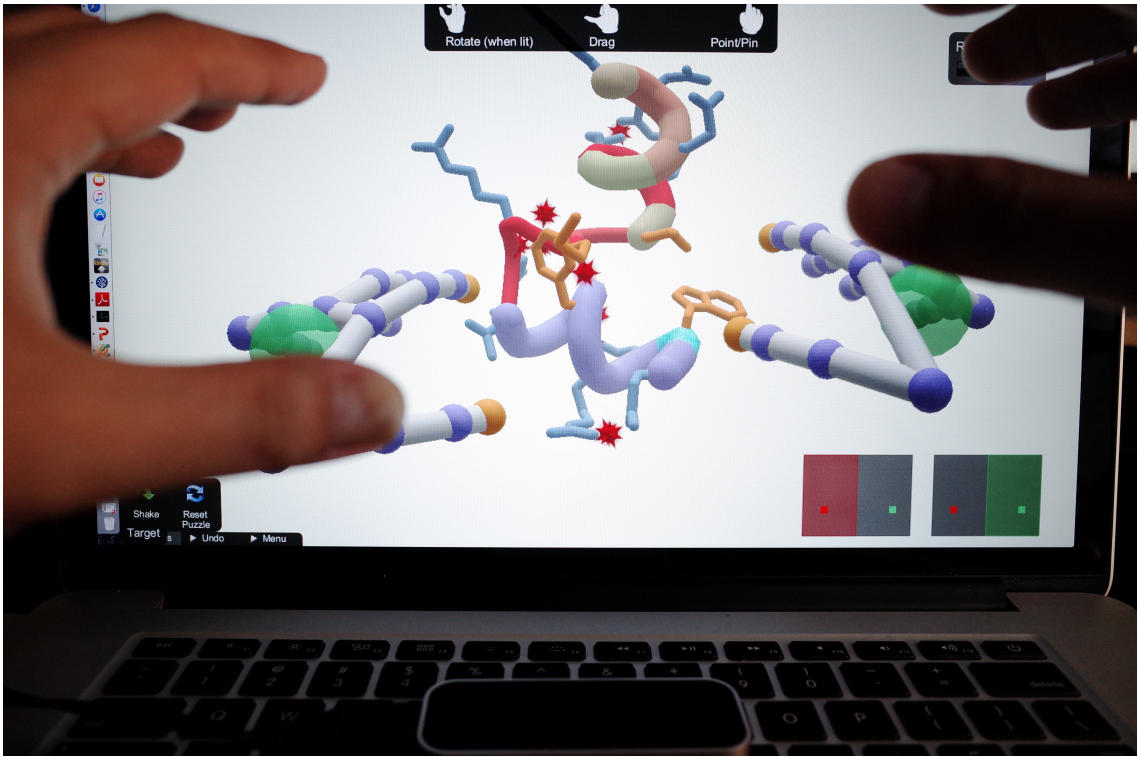


Figure 4.4: Exemplar task for Leap two-handed navigation experiment. Users were asked to navigate with both of their hands at random screen locations. Coordinate mapping is done by each of the hand bounding boxes, which is dynamically changed and mapped to the whole screen.

4.2.1 Problem Statement

We experimented *Mutual Adaptation* first within the Foldit Hand scenario with Leap sensor. Leap sensor comeswith a range of roughly 150° field-of-view width and 120° field-of-view depth (as shown in Fig. 4.5). If the user rest his/her hands on the desk, the average height

of hands is around 20-25cm (7.87-9.84 inches). Therefore, the average width of interaction area is around 35-43cm (13.78-16.93 inches) from the center to either side.

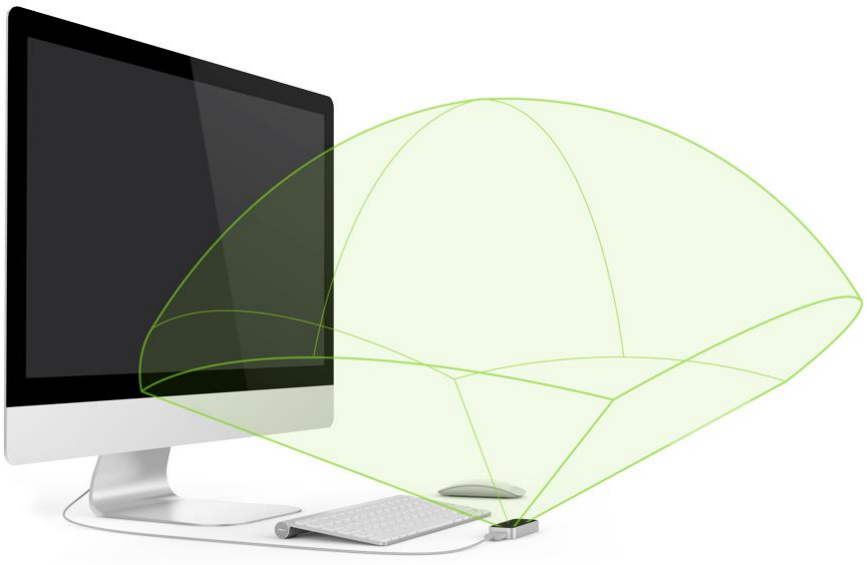


Figure 4.5: Leap sensor interaction area limitation.

In single-hand operation, this interaction area size comes with satisfying precision and ergonomics. However, it is more challenging under bimanual scenarios as the interaction area is now shared by both hands. With the presence of both hands, the interactive area dedicated for each hand is reduced significantly. Therefore, the ratio between the on-screen cursor movement and the physical hand motion becomes larger, which implies any sensor noise gets amplified accordingly.

In addition to the detection limitation of the sensor, there's another challenge due to the problem of uneven operation of the two hands as people tend to use one ofthe hands exclusively for some specific operations. For example, oftentimes people only use their right hand instead of the left one to write or to use a knife when eating, and this kind of habits lead to the difference in muscle memory and strength of the two hands. Usually, one of the hand is more frequently used, and is hence stronger and more precise than the other in terms

of operation, while individual preference varies from one person to another. Another factor of physical limitation is that the interaction areas for the user's hands are sometimes not symmetric or identical. For example, an injured or handicapped arm will usually lead to smaller and lower interaction areas than that of a healthy arm.

In order to optimize interactions for system recognition, is is necessary to set up a proactive interaction environment not compromised by these limitations, decide the best interaction space, and dynamically change the destination interactionspace to nudge users into. In the next section, we will provide the details of the guideline of set the goal interaction state for the two-handed navigation example.

4.2.2 Smart Nudging Guidelines for Two-handed Navigation with Leap

We set up the following guidelines in order to determine the optimal interaction state for the respective user:

Maximize interaction area to increase precision. To handle the detection limitation such as the noise level of the sensor, we want to maximize the interaction area whenever it is possible and permitted.

Determine user preferred interaction area. The user may use the two of his/her hands differently, and the way he/she used to use his/her hands could lead to different muscle memory or proficiency of the two hands. A user might be more comfortable moving his/her preferred hand more than the other hand, and the system should be able to perceive the preference while adjusting the interaction areas accordingly.

Determine user reachable interaction area. In some cases, people might have hard limitations of the range they can reach with hands. The system should be able to discover limitation while observing hand motion occurrence, and reduce interaction area for the respective hand.

Dynamically adjust interaction area. The optimal interaction area can vary with time. For example, if the user operates his/her hands more asynchronously and let one of the hands idle, the system may increase interactive area for the hand that has more motion, and shrink that of the other. When the user starts using his/her idled hand, the interaction area of the previously idling hand should increase and decrease that of the other.

4.2.3 Implementation Details

To implementation such system, we treat it as two dynamically changing mapping bounding box for each hand as we have in the *Human Adaptation* chapter. We use the centers of the bounding boxes along with their width and height to represent the interaction state of the system. In the *Human Adaptation* chapter, the final goal interaction state $\boldsymbol{\theta}_D$ is set to a fixed location and area. In this chapter, however, the state is dynamic and constantly adjusted based on the observation of how user interact with the system. Therefore, the system setup is very similar:

$$
\begin{aligned}
Mapping\ \ function: &\quad f(\mathbf{x}_t, \boldsymbol{\theta}_{jt}) = M_{\mathbf{j}\theta}\mathbf{x}_t + \mathbf{b}_{\mathbf{j}\theta} \\
State\ \ parameters: &\quad \mathbb{X} = \mathbb{Y} = \mathbb{R}^3, \Theta = \mathbb{R}^4 \\
Natural/Engineered\ \ states: &\quad \boldsymbol{\theta}_{jI} = [\mathbf{p}_{ji}, d_{ji}], \boldsymbol{\theta}_{jD} = [\mathbf{p}_{jd}, d_{jd}] \\
Essential\ \ parameter\ \ selector: &\quad \mathbf{e} = [1, 1, 1, 0]
\end{aligned}
$$

The main difference here is that now the goal interaction states is no longer $\boldsymbol{\theta}_D$ but replaced by a flexible $\boldsymbol{\theta}_{jD}$

$$
M_{\mathbf{j}\theta} = \begin{bmatrix} W_x/B_{jx} & 0 & 0 \\ 0 & W_y/B_{jy} & 0 \\ 0 & 0 & W_z/B_{jz} \end{bmatrix}
$$

Mapping function is defined for each hand separately; following is the definition of the mapping for the right hand in order to maximize interaction area for more precision. We use $d_{jI},\ d_{jBL},\ d_{jBR},\ d_{jPL},\ d_{jPR},\ d_{jHL},\ d_{jHR}$ to determine the max bounding box that we can find, where $d_{jPL},\ d_{jPR}$ implies the user preferred interaction area, and $d_{jHL},\ d_{jHR}$ indicates user reachable interaction area.

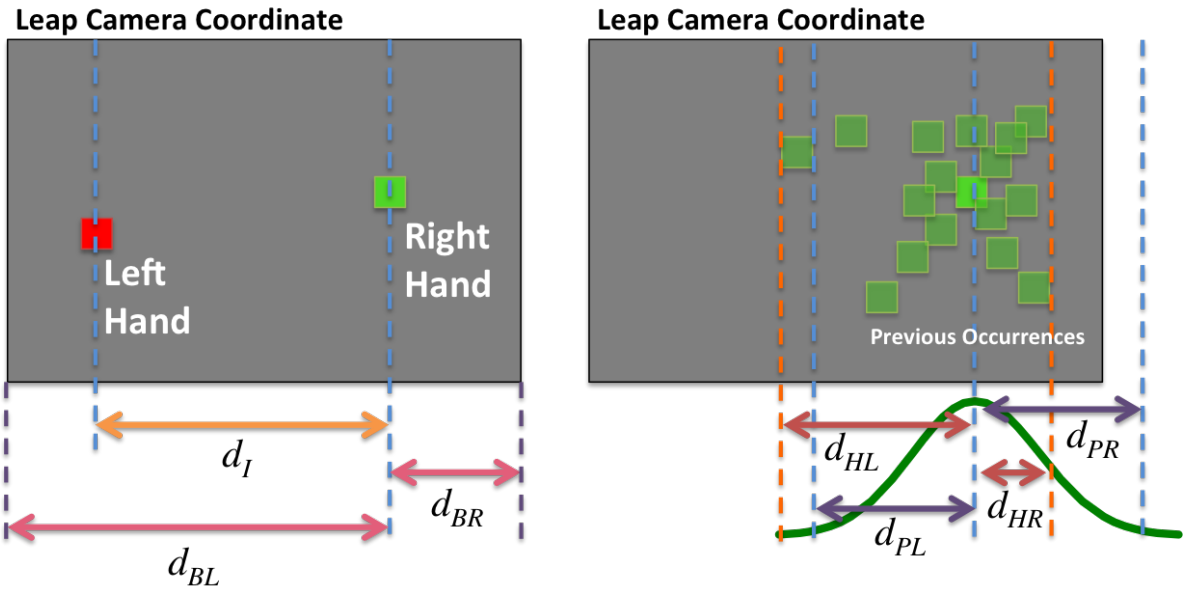


Figure 4.6: Leap sensor interaction area coordinate mapping.

$$B_{jx} = 3 * min(d_{jI}/2,\ d_{jBL}/2,\ d_{jBR},\ d_{jPL}/2,\ d_{jHL}/2,\ d_{jPR},\ d_{jHR})$$

d_{jI}**:** Distance between two hand cursor.

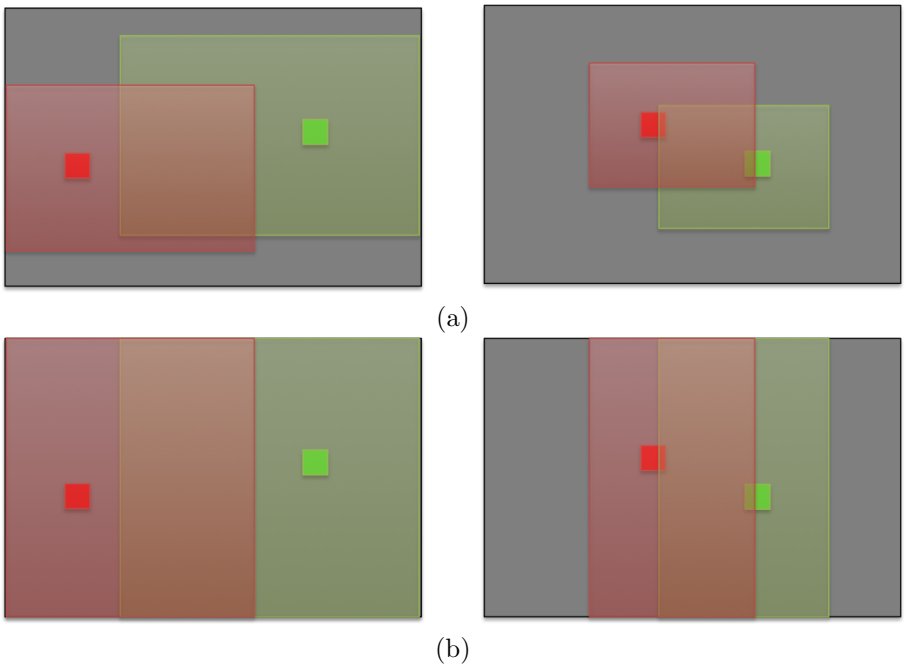
$d_{jBL},\ d_{jBR}$**:** Distance to the left or right bound of the sensor camera coordinate system.

$d_{jPL},\ d_{jPR}$**:** Distance of 1.15 times of standard deviation for 75% confidence of hand occurrence to the left or right side.

d_{jHL}, d_{jHR}**:** Distance of the hard bound of hand occurrence to the left or right side.

$$B_{jy} = 3/5 * B_{jx}, \;\; B_{jz} = B_{jx}$$

It is also possible to set $B_{jy} = W_y$ to gain a even larger interaction area. After we obtain the bounding box size, we can reciprocally calculate the centers of the bounding boxes for the current time frame based on the hand coordinates in the previous time frame.



(a)

(b)

Figure 4.7: Smart nudging initial mapping bounding box example. Bottom row shows the one with $B_{jy} = W_y$ setup.

4.2.4 Exemplar Results

Here are some results from a few example trials. The mapping bounding boxes will adjust to the user's operating behaviors.

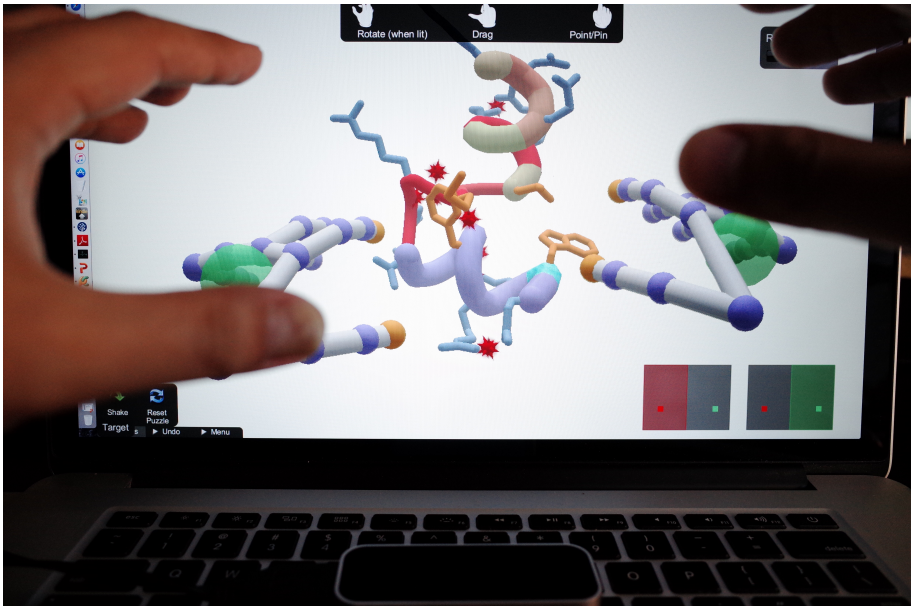


Figure 4.8: Two-handed navigation with Leap sensor when both hands' motion are large and synchronous, the system prefers each hand move within their respective bounding box.

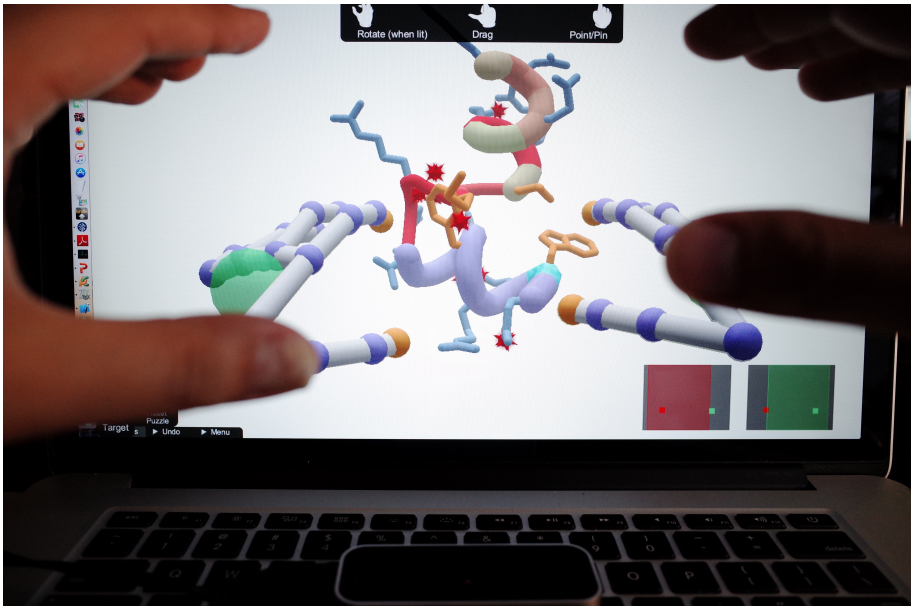


Figure 4.9: Two-handed navigation with Leap sensor when both hands' motion are small or asynchronous, which allows larger bounding boxes.

4.3 Example: VR Redirected Walking

Another example we want to present with *Mutual Adaptation* is VR redirected walking. In a VR environment [5], as users are unable to see anything in the space of actual reality, they have to completely depend on the virtual space while navigating. Despite several navigations, such as driving [21], skiing, riding [26], flying [40], swimming [13], and special actions [3, 6, 16] such as teleporting or using a portal, don't require actually moving around in the physical space, these methods can cause nausea after some periodof successive usage due to the inconsistency of sense of presence in the virtual and the physical space.

On the other hand, in certain VR setup, users are allowed to walk in the physical space in order to navigate in the virtual world. Being allowed to walk, users may have some consistent sense of presence, which leads to a comfortable and natural interaction [8, 29, 34, 40, 45].

However, because the physical space is never unlimited, walking with a straightforward coordinates mapping is not feasible when exploring a large virtual environment. Therefore, in order to overcome the physical interaction space limitation, researchers developed a technique called redirected walking. Redirected walking [27] in VR is about how we can utilize limited physical space to allow users to explore a large space in the virtual world.

Since redirected walking systems map their users' movement in the physical space into the virtual space, we can apply *Mutual Adaptation* in this mapping relationship to increase its effectiveness.

4.3.1 Problem Statement

As briefly mentioned, one of the challenges in VR redirected walking is to make good use of the limited physical space to allow the user to navigate in the virtual space without bumping into walls or obstacles. Again, there could be detection or physical limitations that decrease applicable physical space in the mapping process. For example, some areas might be invalid space for the system due to occlusion.

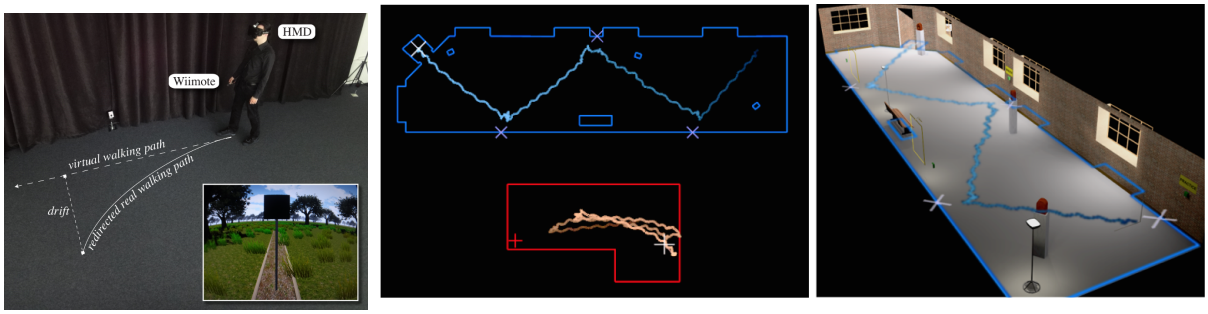


Figure 4.10: Redirected walking system examples.

In addition, the shape of the physical environment could make it challenging to navigate freely, while objects become obstacles, still or moving. Thus, how to avoid bumping into anything in space while navigating the virtual one is of crucial importance, particularly in a multi-user VR environment,

Finally, another challenge worth mentioning is how to provide proper feedback to help increase the user's sense of presence and avoid dizziness. To achievethis, one commonly applied tactic in redirected walking is to manipulate the mapping scale with rotation or translation between users physical and virtual motions. Unfortunately, human beings have limited tolerance to the mismatch between visual and vestibular cues. When the threshold is surpassed, human would notice the manipulations, [32] while the effectiveness would decrease [40].

We will try to address some of the challenges mentioned in this work.

4.3.2 Related Work

To be able navigate in the virtual environment has been one of the fundamental capability in virtual reality setups. The most straight forward and intuitive way to allow this is by mapping the physical interaction space directly into the virtual space with 1:1 ratio. However, this is in general not feasible due to the virtual environment is often larger than the limited

physical space. Hence, VR researchers have been working on various approaches in order to facilitate navigation.

Non-walking Methods

Some researchers proposed approaches to allow the user to navigate with an accelerated locomotion in the virtual environment without using the walking metaphor. They usually borrow some other metaphor from real world experience to maintain the affordance. These methods include: portal [6], flying [40]

Walking Methods

In-place Walking-like Methods. The first category of walking navigation is a compromised approach from the real walking approach: it incorporates part of the gesture motion that involved during walking or movement but the user's physical location does not actually change. Approaches in the category use motion such as head motion [40] step motion [38], heel and chest motion [12], or through capturing using gestural sensor [35].

Redirected Walking. Redirected walking is about providing locomotion in the virtual environment with real walking motion in the limited physical interaction space. The main concept of this approach is by altering the motion gain between the mapping of physical and virtual motion. There are a few different approaches in this category:

Stop/reset and redirect. The simplest implementation for redirected walking is asking the user to stop and reorient when there's no physical space available to continue walking [27,44]. Some applied with better visual cues in order to achieve the same idea [25].

Gain manipulation. Most of the variant of redirected walking were build around

the gain manipulation concept [11, 24, 31] that initially proposed by Razzaque et al [27]. Hodgson et als [15] further refined the approach with Steer-To-Center and Steer-To-Orbit [11, 14] algorithms to use rotation and curvature gains to steer the user toward either the center or an orbit with respect to the physical space. Another special aim through this method is to provide more haptic feedback using limited real world object as show in Haptic Retargeting [2].

There are some work put the gains on acceleration, using metaphors like Seven league boots and Jumper. Seven league boots apply acceleration according to the users speed [16]. Jumper will sense the user's acceleration all the time: uses real walking for short distances and when acceleration is below some threshold. If the acceleration got passed the threshold, a virtual jumps will be applied to a predicted location with relatively large distances [3].

Nescher et al. [22] generalized approach to planning and applying redirected walking techniques. It is capable of dynamically selecting suitable redirected walking techniques and also controlling their strengths.

Spatial structure manipulation. Sometimes the physical interaction space does not easily allow rotation or curvature gain to apply, so some researchers borrowed some other metaphor to reshape the virtual world in a way that navigable. Portals were used in the Arch-Explore project [6]. The portal served as a corridor that connected to another part of the virtual environment, hence the virtual environment's interaction space was reshaped to facilitate navigation. The portal was disappeared after the user entered the connected part of the virtual environment.

There are several other methods manipulate the spatial layout of the virtual environment. Vasylevska et al. proposed a procedural generation of architectural layouts that supports infinite walking through large, highly-occluded virtual environments, which we refer to as flexible spaces [41, 42]. Sums et al. proposed a spatial layout change method done through self-overlapping architecture via

pushing shared walls between adjacent rooms [36]. This approach allows more efficient use of the physical space. Sun et al. proposed using a warped mapping structure to replace the mapping between the physical and virtual space but will introduce warped visualizations [37].

Manipulation using perception flaws. A different type of approach took advantage of human perceptual flaws was proposed in [33], which known as change blindness. Change blindness represents the failure to see large changes that should be noticed easily [30]. When the user was working on a task and got distracted, the technique would change the position of some key components in the spatial layout such as the position of the door. Therefore it is possible to redirect user when the user travels from room to room without becoming noticeable by the user. Another method that extended change blindness was proposed by Bruder et al. [7]. They took the advantage of human's eye blinking to change the scene through applying intermittent gray screens to hide away the changes.

Walking with External Hardware. Since the discrepancy in human's sensing will can affect the performance of redirected walking. Some researchers in redirected walking proposed using additional hardware to alter human's sense of presence to facilitate redirected walking. Various prototypes of interface devices have been developed. Approaches include treadmills [4], carpet [10], tiles [19], hamster ball from VirtuSphere, and shoes [9, 20, 43].

Toolkits to test redirected walking. In order to more efficiently test out redirected walking algorithms, there is also a unified platform available for developing, benchmarking, and deploying [1].

Redirected walking limitations. Steinicke et al. evaluated human's sensitivity with respect to the changes of the gain applied in redirected walking [31,32]. Human is about twice as sensitive toward distance gain comparing with the rotation gain and a circular

walking path of 22 meters in radius.

4.3.3 Test Environment

Virtual Environment

To apply *Mutual Adaptation* to simulate a redirected walking scenario, Foldit Hand is no longeran ideal candidate. Instead, we created a procedurally generated maze (see Fig. 4.11) as test environment to evaluate the proposed algorithm, as we want to create a complex map that can be navigated. It will be based on first person viewing perspective so that the entire environment will be similar to a Doom-like maze (see Fig.4.12).

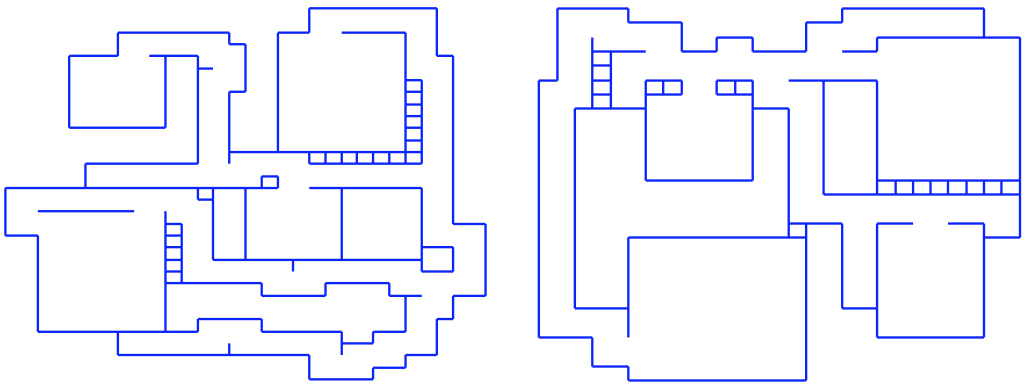


Figure 4.11: Room generation example.

Describing Interaction Space

To clearly define the accessible interaction space, we have to first describe the shape of the interaction space. It is done with a radial shape descriptor as shown in Fig. 4.13. Given the current location and looking direction, the radial shape descriptor casts rays and samples the distance from the current location to the first obstacle each ray hits. In our implementation, we used a sweep of 360° with 1° interval through combining the depth sweep of 90° in the

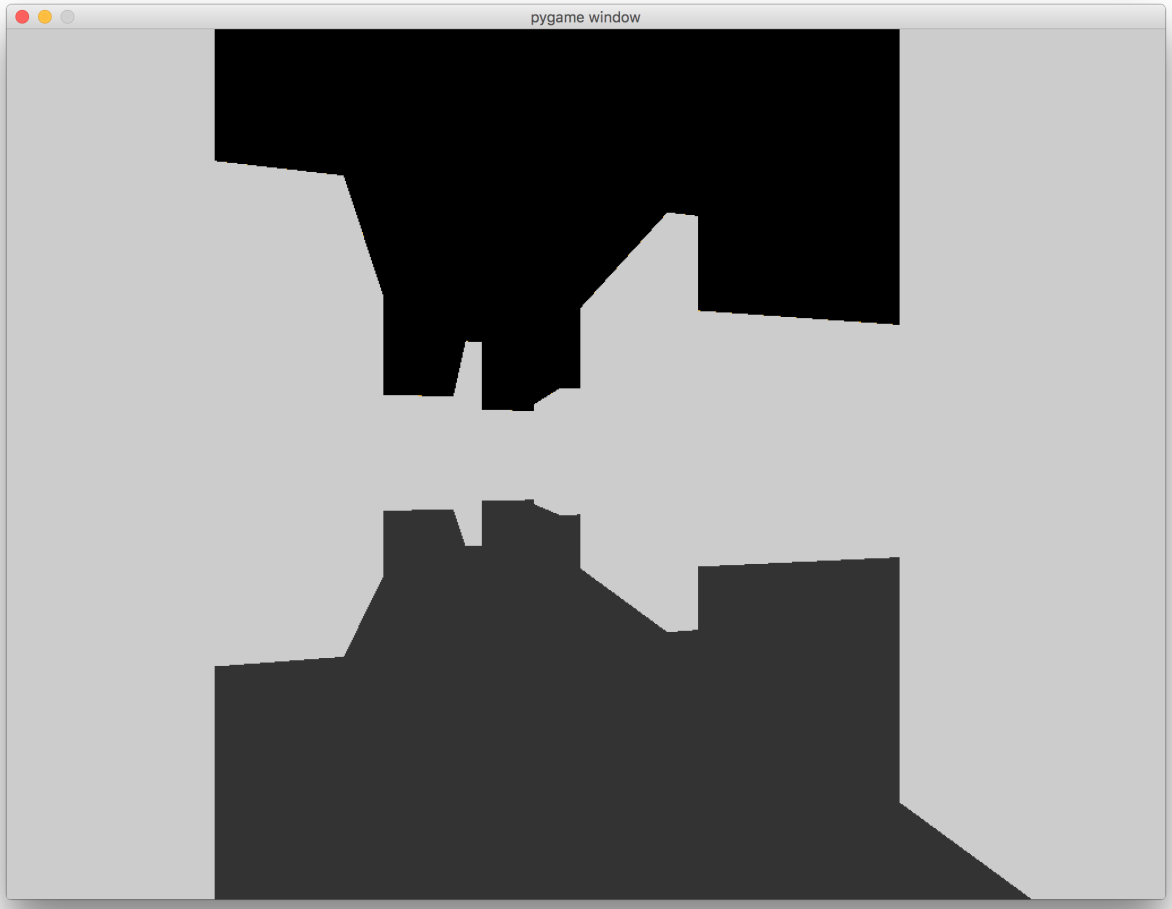


Figure 4.12: Redirected walking room.

forward, left, back and right directions.

Control Mechanism

There are a few plans of implementation to the system, including controller input, automated simulation or real walking in physical space. In the first version, the system provides navigation control through the joystick on a DualShock 4 controller as shown in Fig. 4.15. The

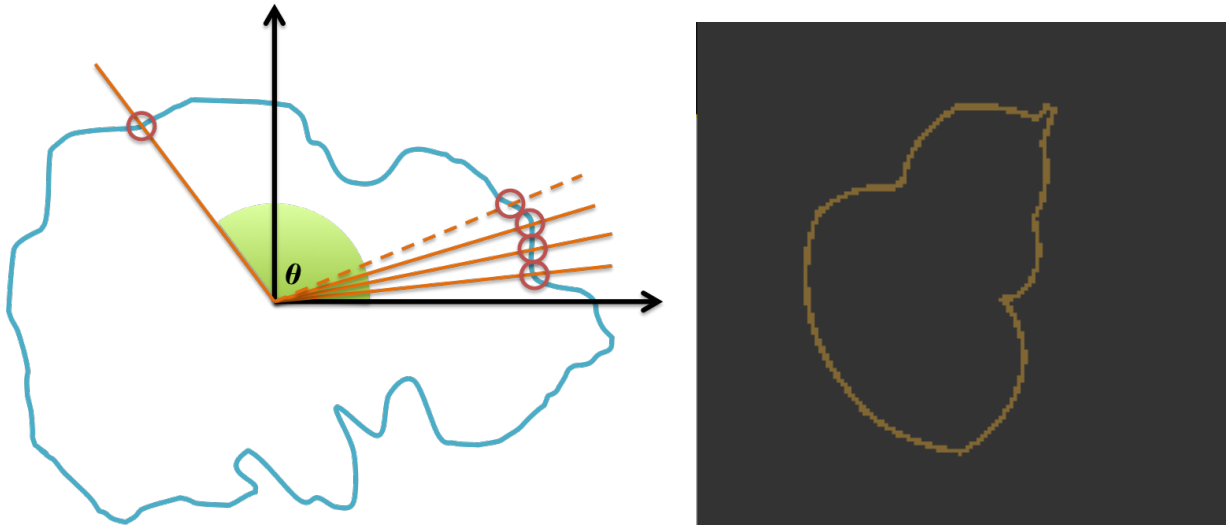


Figure 4.13: Shape descriptor to define the interaction space.

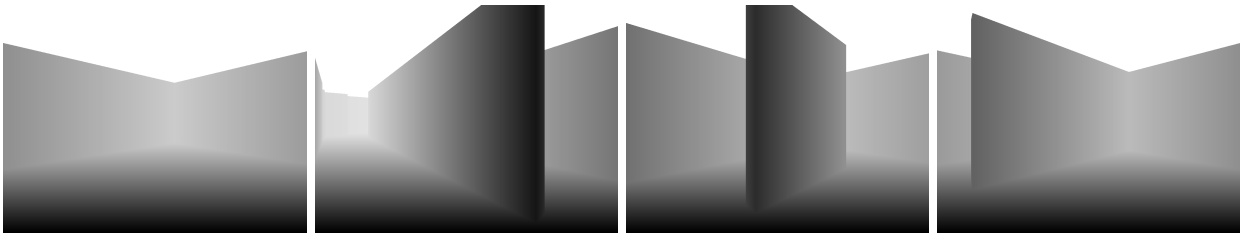


Figure 4.14: 90° FOV depth map in four directions.

user of the system can either use a VR headset or a monitor to view the virtual world. The joystick forward/back will move the user forward or backward, and the joystick left/right will rotate the user counterclockwise or clockwise.

Visualization

To help user explore the virtual environment, we created a few visualizations, from left to right:

Mini-map for Virtual Environment. The top-down view of the virtual en-

Figure 4.15: Controller for redirected walking test environment.

vironment. The current position and viewing angle are represented by the red arrow. Colored trajectory will be displayed when the user moves around.

Shape Descriptor for Virtual Environment. The top-down view of the shape descriptor for the virtual environment with respect to the current position and viewing angle. The larger the radius is, the farther the nearest obstacle is.

Shape Descriptor for Physical Environment. The top-down view of the shape descriptor for the physical environment with respect to the current position and viewing angle. The larger the radius is, the further the nearest obstacle is.

Mini-map for Physical Environment. The top-down view of the physical environment. The current physical environment is an automatically generated simulation. When we move forward to the next test stage, it can be replaced by the map of the real space. The current position and viewing angle are represented by the red arrow. Colored trajectory will be displayed when the user moves around.

An example of the full visualization can be seen in Fig.4.16.

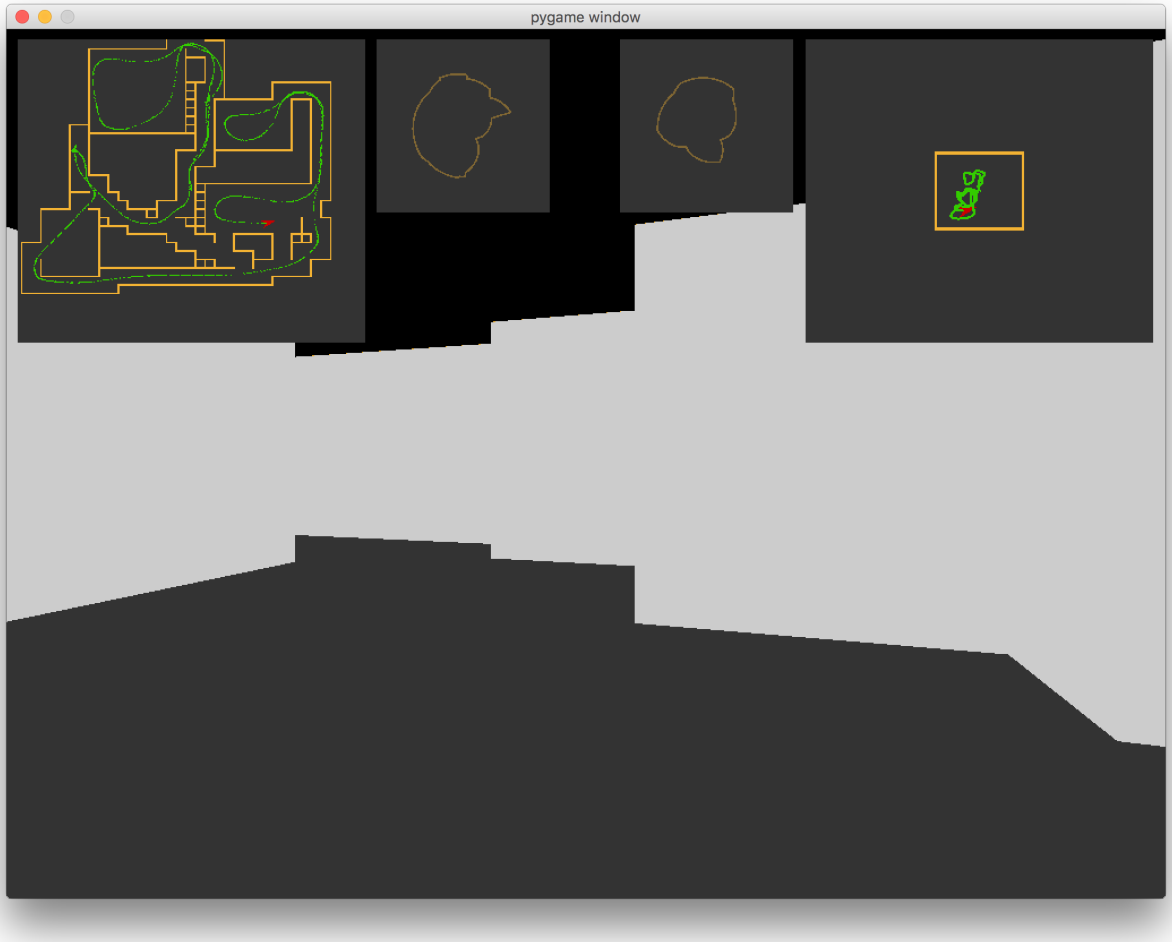


Figure 4.16: Redirected walking test environment.

4.3.4 Smart Nudging Guidelines for VR Redirected Walking

We set up the following guidelines in order to determine the optimal interaction state:

Maximize interaction area. Again, due to the signal to noise ratio, the greater gain will lead to higher noise. In order to suppress the noise level, we want the interaction area to be

as large as possible.

Figure out the user's current interaction area. The space in the direction of where the user can move into will determine the gain of the user's following motion. Hence, it is usually preferable if the user continues moving into physical areas with comparatively more space than the present one. In order to do so, it's necessary to find a way to describe the current interaction space.

Determine next optimal interaction area. Since it is favorable to continuously move the user into more spacious counterpart in the physical space, based on how the current interaction space is defined, the system will select a candidate interaction space to move into based on the detected motions.

Provide haptic feedback/sense of presence if possible. While the candidate physical interaction space to move into is determined, if the physical layout could provide haptic feedback in certain candidate interaction space, the interaction space will be preferable for the system.

Dynamically adjust interaction area to lead the user to the optimal interaction area. Use nudging technique to adjust translation and rotation gain to gradually drive the user into the desired interaction space.

4.3.5 Automatic Goal Setting: Optimize Dynamic Depth Mapping

To address the guidelines for getting updated goal interaction states over time, we implemented two different methods to determine the optimal goal state (direction to move into) at each time frame.

Optimal Shape Matching

The main concept of this method is to best match the shape of the physical space with the virtual one. For example, if the virtual space ahead of the user is a long corridor, and there's still much physical space for the user on the slight right, the system will nudge the user to slight right in order to avoid increasing motion gain. This is how the physical space is utilized to prevent collision.

The shape descriptors of the virtual and physical environment compared with theattempt to find the best shift match through cross-correlation. This method then suggests that direction (green line in Fig.4.17) for rotation gain to change.

Max Peaks Finding

The idea for this method is to suggest candidate direction to move into by finding potential angles through distance peaks, and to provide strong signal when the user is very close to a physical space limit, for example, a corner. In such cases, we have to provide a strong direction to move into, and apply a much higher rotation gain to redirect users from the corner.

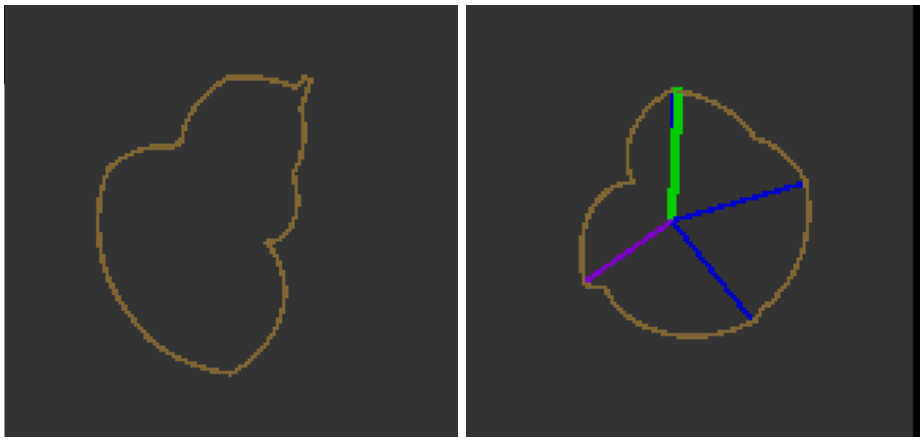


Figure 4.17: Redirected walking mapping. Left: virtual space. Right: physical space.

This method finds distance peaks for both left and right directions by sweeping $-180°$ and $+180°$. Candidate directions are shown as purple (left, $-180°$ sweep) or blue (right, $+180°$ sweep) lines in Fig.4.17.

The optimal goal state (direction to move into) is then determined through a voting among those suggested angles wit respect to the current motion, i.e., weighting more on the angle more if the user is moving toward that direction.

4.3.6 Nudging: Translation and Rotation Motion Gain

After the optimal goal state is determined, we can then apply nudging technique to put the user into the right interaction space.

Translation Gain with the Benefit of Haptic Feedback

The idea of translation gain is straightforward: to map the existing physical interaction space 100% to the virtual space. This is achieved by getting the scaling coefficients on each single angle direction. The benefit of this mapping is that the physical space in this mapping can provide haptic feedback. For example, when the user comes near an obstacle in the physical space, he/she also comes near an obstacle in the virtual space. In this scenario, consequently, the user is forced to turn around due to the visualization of obstacle he/she sees, and the rotation gain can nudge the user away from the obstacle by manipulating the user's physical direction.

Rotation Gain

Rotation gain is variable that allows the user to rotate with different magnitude in the physical space from that in the virtual one. This technique is done by applying different scale to the magnitude of the rotation depending on how far the user is from the desired direction. When the user is far from the desired direction, then the rotation gain is lowered to allow

more physical rotation to catch up. The rotation will become less sensitive consequently. Conversely, if the user is getting close to the desired direction, the rotation gain will ramp with the rotation becoming more sensitive.

Results

Here are some results from a few example runs. All the trajectories are well bounded within the physical interaction space without causing any trouble moving around in the virtual environment.

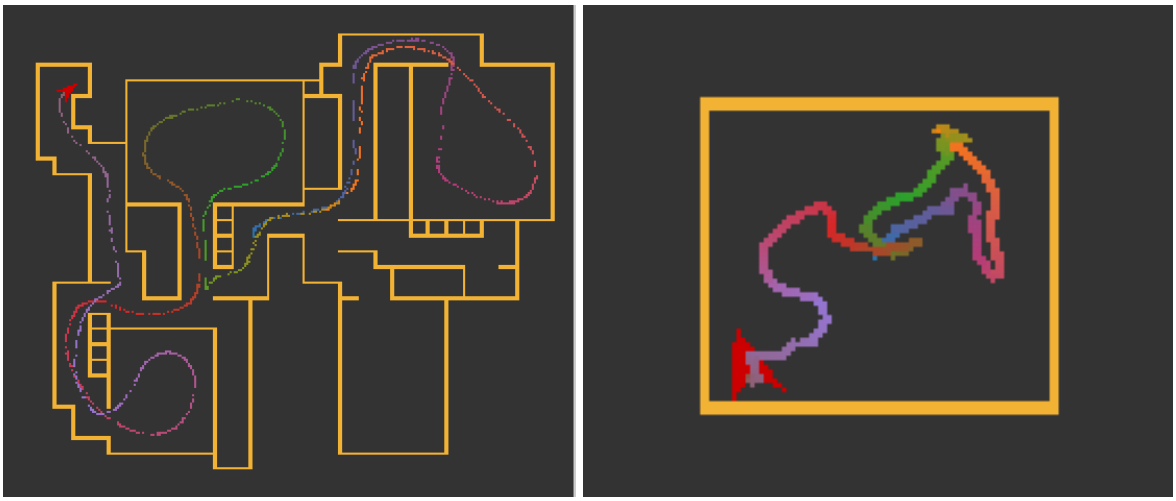


Figure 4.18: Redirected walking trajectories. Left: virtual space. Right: physical space.

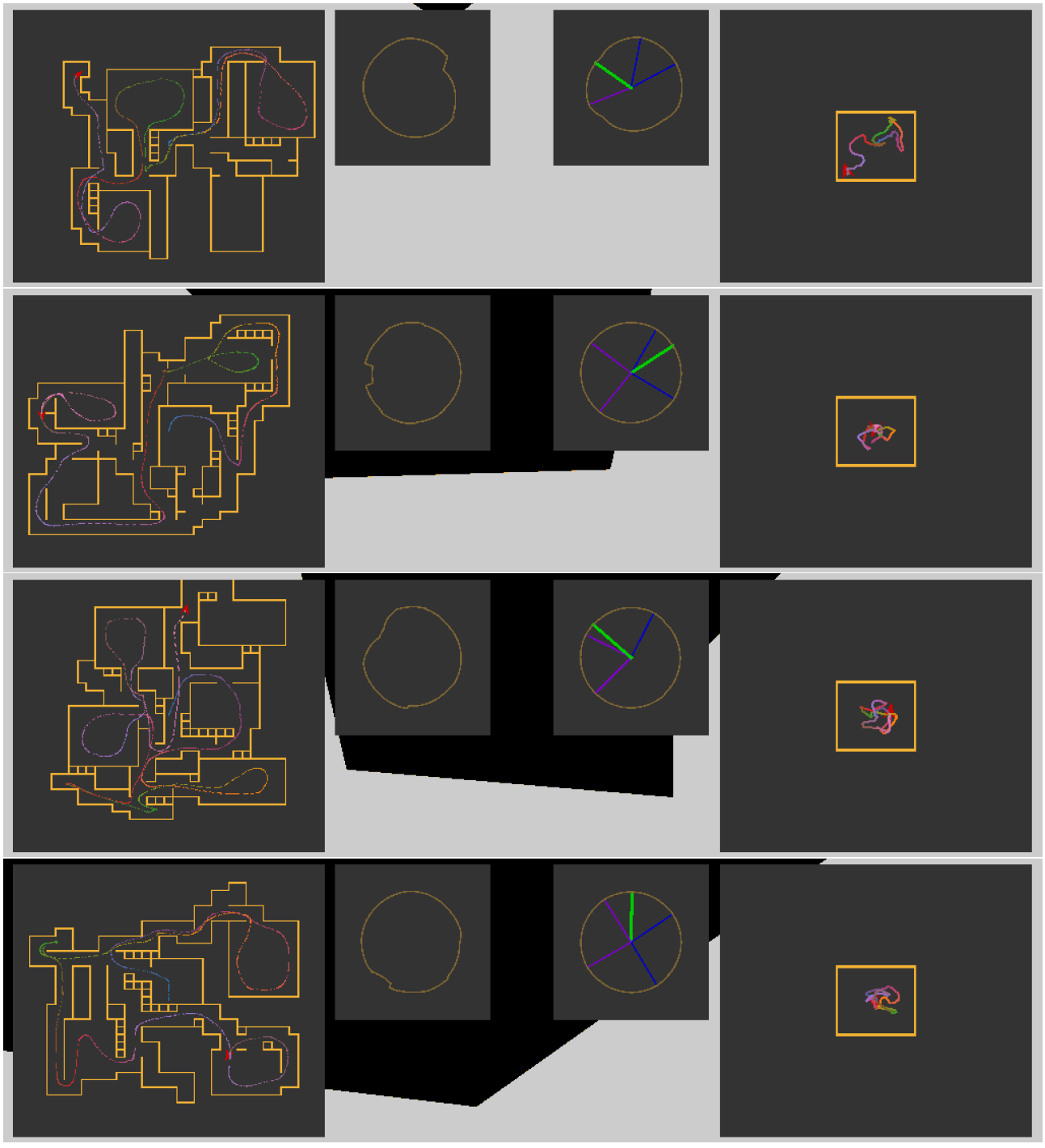


Figure 4.19: Redirected walking results.

4.4 Conclusion and Future Work

In this chapter, we applied *Mutual Adaptation* technique to two working interaction systems, one of which is the two-handed navigation scenario with the Leap sensor. The results indicatethe performance is indeed enhanced andconsequently better than the previous version with *Human Adaptation*. A flexible goal interaction state is set with respect to its current user, while the noise level us reduced.

In addition, we applied *Mutual Adaptation* techniques in a VR redirected walking scenario. With *Mutual Adaptation*, it is possible to continuously determine and nudge its user to a workable interaction space over time. Being more flexible and dynamic is the main strength for our proposed approach.

Figure 4.20: SLAM enabled VR setup.

The next step of this experiment will be putting people to use this redirected walking system in the physical space. In order to do so, we'd need information of the space around the users. The techniques of Simultaneous Localization and Mapping (SLAM) can be made use of to obtain the depth information, which is used to generate the shape descriptor around the user. Currently, we have a SLAM working setup that can be seen in Fig.4.20.

We believe that the concept of *Mutual Adaptation* will be applicable in many other application scenarios. As long as the interaction comes with some features of inter-personal communication, it is always beneficial to have the interaction designed with integration of *Mutual Adaptation*.

Bibliography

[1] Mahdi Azmandian, Timofey Grechkin, Mark Bolas, and Evan Suma. 2016a. The Redirected Walking Toolkit: A Unified Development Platform for Exploring Large Virtual Environments. In *2nd Workshop on Everyday Virtual Reality*. IEEE, Greenville, SC. http://www.adalsimeone.me/papers/WEVR2016/WEVR2016_Azmandian.pdf

[2] Mahdi Azmandian, Mark Hancock, Hrvoje Benko, Eyal Ofek, and Andy Wilson. 2016b. Haptic Retargeting: Dynamic Repurposing of Passive Haptics for Enhanced Virtual Reality Experiences. (May 2016), 1968–1979.

[3] Benjamin Bolte, Gerd Bruder, and Frank Steinicke. 2011. The Jumper Metaphor: An Effective Navigation Technique for Immersive Display Setups. In *Proceedings of the Virtual Reality International Conference (VRIC)*. 1–7. http://basilic.informatik.uni-hamburg.de/Publications/2011/BBS11a

[4] Laroussi Bouguila, Masahiro Ishii, and Makoto Sato. 2002. Virtual Locomotion System for Human-scale Virtual Environments. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '02)*. ACM, New York, NY, USA, 227–230. DOI: http://dx.doi.org/10.1145/1556262.1556299

[5] Doug A. Bowman, Ernst Kruijff, Joseph J. LaViola, and Ivan Poupyrev. 2004. *3D User Interfaces: Theory and Practice*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

[6] Gerd Bruder, Frank Steinicke, and Klaus H. Hinrichs. 2009. Arch-explore: A natural user interface for immersive architectural walkthroughs. In *In Proceedings of Symposium on 3D User Interfaces*. IEEE Press, 75–82.

[7] G. Bruder, F. Steinicke, and P. Wieland. 2011. Self-motion illusions in immersive virtual reality environments. In *2011 IEEE Virtual Reality Conference*. 39–46. DOI:http://dx.doi.org/10.1109/VR.2011.5759434

[8] Sarah S. Chance, Florence Gaunet, Andrew C. Beall, and Jack M. Loomis. 1998. Locomotion Mode Affects the Updating of Objects Encountered During Travel: The Contribution of Vestibular and Proprioceptive Inputs to Path Integration. *Presence: Teleoper. Virtual Environ.* 7, 2 (April 1998), 168–178. DOI:http://dx.doi.org/10.1162/105474698565659

[9] Cyberith. Online. Cyberith Virtualizer. (Online). http://www.cyberith.com/.

[10] Alessandro De Luca, Raffaella Mattone, Paolo Robuffo Giordano, Heinz Ulbrich, Martin Schwaiger, Michael Van Den Bergh, Esther Koller-Meier, and Luc Van Gool. 2013. Motion control of the CyberCarpet platform. *IEEE Transactions on Control Systems Technology* 21, 2 (2013), 410–427. DOI:http://dx.doi.org/10.1109/TCST.2012.2185051

[11] David Engel, Cristóbal Curio, Lili Tcheang, Betty Mohler, and Heinrich H. Bülthoff. 2008. A Psychophysically Calibrated Controller for Navigating Through Large Environments in a Limited Free-walking Space. In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology (VRST '08)*. ACM, New York, NY, USA, 157–164. DOI:http://dx.doi.org/10.1145/1450579.1450612

[12] Jeff Feasel, Mary C. Whitton, and Jeremy D. Wendt. 2008. LLCM-WIP: Low-latency, continuous-motion walking-in-place. In *In IEEE Symposium on 3D User Interfaces*. 104.

[13] Sidney Fels, Steve Yohanan, Sachiyo Takahashi, Yuichiro Kinoshita, Kenji Funahashi, Yasufumi Takama, and Grace Tzu-Pei Chen. 2005. *User Experiences with a Virtual Swimming Interface Exhibit*. Springer Berlin Heidelberg, Berlin, Heidelberg, 433–444. DOI:http://dx.doi.org/10.1007/11558651_42

[14] Tom Field and Peter Vamplew. 2004. Generalised algorithms for redirected walking in virtual environments. In *2004 International Conference on Artificial Intelligence in Science and Technology*. 1357–1366.

[15] Eric Hodgson and Eric Bachmann. 2013. Comparing Four Approaches to Generalized Redirected Walking: Simulation and Live User Data. *IEEE Transactions on Visualization and Computer Graphics* 19, 4 (April 2013), 634–643. DOI:http://dx.doi.org/10.1109/TVCG.2013.28

[16] Victoria Interrante, Brian Ries, and Lee Anderson. 2007. Seven League Boots: A New Metaphor for Augmented Locomotion through Moderately Large Scale Immersive Virtual Environments.. In *3DUI*. IEEE Computer Society, 35. http://dblp.uni-trier.de/db/conf/3dui/3dui2007.html#InterranteRA07

[17] Hiroo Iwata. 2013. *Locomotion Interfaces*. Springer New York, New York, NY, 199–219. DOI:http://dx.doi.org/10.1007/978-1-4419-8432-6_9

[18] Hiroo Iwata and Takashi Fujii. 1996. Virtual perambulator: a novel interface device for locomotion in virtual environment. In *Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium*. 60–65, 265. DOI:http://dx.doi.org/10.1109/VRAIS.1996.490511

[19] Hiroo Iwata, Hiroaki Yano, Hiroyuki Fukushima, and Haruo Noma. 2005. CirculaFloor. *IEEE Comput. Graph. Appl.* 25, 1 (Jan. 2005), 64–67. DOI:http://dx.doi.org/10.1109/MCG.2005.5

[20] Hiroo Iwata, Hiroaki Yano, and Hiroshi Tomioka. 2006. Powered Shoes. In *ACM SIGGRAPH 2006 Emerging Technologies (SIGGRAPH '06)*. ACM, New York, NY, USA, Article 28. DOI:http://dx.doi.org/10.1145/1179133.1179162

[21] H. S. Kang, M. K. Abdul Jalil, and Musa Mailah. 2004. A PC-based Driving Simulator Using Virtual Reality Technology. In *Proceedings of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry (VRCAI '04)*. ACM, New York, NY, USA, 273–277. DOI:http://dx.doi.org/10.1145/1044588.1044646

[22] Thomas Nescher, Ying-Yin Huang, and Andreas Kunz. 2014. Planning redirection techniques for optimal free walking experience using model predictive control. *2014 IEEE Symposium on 3D User Interfaces (3DUI)* 00 (2014), 111–118. DOI:http://dx.doi.org/doi.ieeecomputersociety.org/10.1109/3DUI.2014.6798851

[23] Thomas Nescher, Markus Zank, and Andreas Kunz. 2016. Simultaneous mapping and redirected walking for ad hoc free walking in virtual environments. In *2016 IEEE Virtual Reality (VR)*. 239–240. DOI:http://dx.doi.org/10.1109/VR.2016.7504742

[24] Tabitha C. Peck, Henry Fuchs, and Mary C. Whitton. 2010. Improved Redirection with Distractors: A large-scale-real-walking locomotion interface and its effect on navigation in virtual environments. In *2010 IEEE Virtual Reality Conference (VR)*. 35–38. DOI:http://dx.doi.org/10.1109/VR.2010.5444816

[25] Tabitha C. Peck, Mary C. Whitton, and Henry Fuchs. 2008. Evaluation of Reorientation Techniques for Walking in Large Virtual Environments. In *2008 IEEE Virtual Reality Conference*. 121–127. DOI:http://dx.doi.org/10.1109/VR.2008.4480761

[26] Richard Ranky, Mark Sivak, Jeffrey Lewis, Venkata Gade, Judith E. Deutsch, and Constantino Mavroidis. 2010. VRACK – virtual reality augmented cycling kit: Design and validation. In *2010 IEEE Virtual Reality Conference (VR)*. 135–138. DOI:http://dx.doi.org/10.1109/VR.2010.5444798

[27] Sharif Razzaque. 2005. Redirected Walking. *PhD Thesis at UNC Chapel Hill* (2005).

[28] Sharif Razzaque, David Swapp, Mel Slater, Mary C. Whitton, and Anthony Steed. 2002. Redirected Walking in Place. In *Proceedings of the Workshop on Virtual Environments 2002 (EGVE '02)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 123–130. http://dl.acm.org/citation.cfm?id=509709.509729

[29] Roy A. Ruddle and Simon Lessels. 2009. The Benefits of Using a Walking Interface to Navigate Virtual Environments. *ACM Trans. Comput.-Hum. Interact.* 16, 1, Article 5 (April 2009), 18 pages. DOI:http://dx.doi.org/10.1145/1502800.1502805

[30] Daniel J. Simons and Ronald A. Rensink. 2005. Change Blindness: Past, Present, and Future. *Trends in Cognitive Sciences* 9, 1 (2005), 16–20.

[31] Frank Steinicke, Gerd Bruder, Jason Jerald, Harald Frenz, and Markus Lappe. 2008. Analyses of Human Sensitivity to Redirected Walking. In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology (VRST '08)*. ACM, New York, NY, USA, 149–156. DOI:http://dx.doi.org/10.1145/1450579.1450611

[32] Frank Steinicke, Gerd Bruder, Jason Jerald, Harald Frenz, and Markus Lappe. 2010. Estimation of Detection Thresholds for Redirected Walking Techniques. *IEEE Transactions on Visualization and Computer Graphics* 16, 1 (Jan. 2010), 17–27. DOI:http://dx.doi.org/10.1109/TVCG.2009.62

[33] Evan A. Suma, Seth Clark, David Krum, Samantha Finkelstein, Mark Bolas, and Zachary Warte. 2011. Leveraging change blindness for redirection in virtual environments. In *2011 IEEE Virtual Reality Conference*. 159–166. DOI:http://dx.doi.org/10.1109/VR.2011.5759455

[34] Evan A. Suma, Samantha Finkelstein, Seth Clark, Paula Goolkasian, and Larry F. Hodges. 2010. Effects of Travel Technique and Gender on a Divided Attention Task in a Virtual Environment. In *IEEE Symposium on 3D User Interfaces*. 27–34. http://people.ict.usc.edu/~suma/papers/suma-3dui2010.pdf

[35] Evan A. Suma, David Krum, Belinda Lange, Sebastian Koenig, Albert Rizzo, and Mark Bolas. 2013. Adapting User Interfaces for Gestural Interaction with the Flexible Action and Articulated Skeleton Toolkit. *Computers & Graphics* 37, 3 (2013), 193–201. http://www.sciencedirect.com/science/article/pii/S0097849312001756

[36] Evan A. Suma, Zachary Lipps, Samantha Finkelstein, David M. Krum, and Mark Bolas. 2012. Impossible Spaces: Maximizing Natural Walking in Virtual Environments with Self-Overlapping Architecture. *IEEE Transactions on Visualization and Computer Graphics* 18, 4 (April 2012), 555–564. DOI:http://dx.doi.org/10.1109/TVCG.2012.47

[37] Qi Sun, Li-Yi Wei, and Arie Kaufman. 2016. Mapping Virtual and Physical Reality. *ACM Trans. Graph.* 35, 4, Article 64 (July 2016), 12 pages. DOI:http://dx.doi.org/10.1145/2897824.2925883

[38] Sam Tregillus and Eelke Folmer. 2016. VR-STEP: Walking-in-Place Using Inertial Sensing for Hands Free Navigation in Mobile VR Environments. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 1250–1255. DOI:http://dx.doi.org/10.1145/2858036.2858084

[39] Amy Ulinski, Zachary Wartell, Paula Goolkasian, Evan A. Suma, and Larry F. Hodges. 2009. Selection Performance Based on Classes of Bimanual Actions. In *IEEE Symposium on 3D User Interfaces*. 51–58. http://people.ict.usc.edu/~suma/papers/ulinski-3dui2009.pdf

[40] Martin Usoh, Kevin Arthur, Mary C. Whitton, Rui Bastos, Anthony Steed, Mel Slater, and Frederick P. Brooks, Jr. 1999. Walking > Walking-in-place > Flying, in Virtual Environments. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 359–364. DOI:http://dx.doi.org/10.1145/311535.311589

[41] Khrystyna Vasylevska, Hanne Kaufmann, Mark Bolas, and Evan A. Suma. 2013a. Flexible spaces: A virtual step outside of reality. In *2013 IEEE Virtual Reality (VR)*. 109–110. DOI:http://dx.doi.org/10.1109/VR.2013.6549386

[42] Khrystyna Vasylevska, Hannes Kaufmann, Mark Bolas, and Evan A. Suma. 2013b. Flexible spaces: Dynamic layout generation for infinite walking in virtual environments. In *2013 IEEE Symposium on 3D User Interfaces (3DUI)*. 39–42. DOI:http://dx.doi.org/10.1109/3DUI.2013.6550194

[43] Virtuix. Online. Virtuix Omni. (Online). http://www.virtuix.com/.

[44] Betsy Williams, Gayathri Narasimham, Bjoern Rump, Timothy P. McNamara, Thomas H. Carr, John Rieser, and Bobby Bodenheimer. 2007. Exploring Large Virtual Environments with an HMD when Physical Space is Limited. In *Proceedings of the 4th Symposium on Applied Perception in Graphics and Visualization (APGV '07)*. ACM, New York, NY, USA, 41–48. DOI:http://dx.doi.org/10.1145/1272582.1272590

[45] Catherine A. Zanbaka, Benjamin C. Lok, Sabarish V. Babu, Amy C. Ulinski, and Larry F. Hodges. 2005. Comparison of Path Visualizations and Cognitive Measures Relative to Travel Technique in a Virtual Environment. *IEEE Transactions on Visualization and Computer Graphics* 11, 6 (Nov. 2005), 694–705. DOI:http://dx.doi.org/10.1109/TVCG.2005.92

[46] Ruimin Zhang and Scott A. Kuhl. 2013. Flexible and general redirected walking for head-mounted displays. In *2013 IEEE Virtual Reality (VR)*. 127–128. DOI:http://dx.doi.org/10.1109/VR.2013.6549395

Chapter 5

CONCLUSION

In this work, we applied a few interpersonal interaction techniques to human-computer interaction to eliminate the ambiguity. With the focus on the gestural interactions, We presented how users can adapt themselves to the existing system, and the limitation as well as weakness of the methods. Following, instead of the mainstream of single-handedly making the interaction system more intelligent and adaptive, we explored the possibility of factor in human capability to adapt along with the adaptation of the gestural interaction system to increase the overall effectiveness. The results shed a light on leveraging human ability to adapt in the loop of gestural interaction design. *Human Adaptation*, the solution we proposed, nudges users in a system from intuitive operations toward more effective ones that are easy to recognize. This bridges the mental model between user and interaction designer. It reduces training/guidance and improves overall effectiveness of the interaction. Our main contributions with *Human Adaptation* are as follows. We proposed a design concept that creates dynamic interface to improve interaction effectiveness. The concept is to allow inputs with potentially high error rate (for sensors), while nudging the user towards giving better detected inputs by adaptively updating the mapping between the user's input and the virtual input in the interactive system. We presented a generalized formal definition of the Active Nudging Interaction approach, along with applications of the approach to interactions with Leap and Kinect. Furthermore, we demonstrated the results of a user study, in which it's obvious that Active Nudging Interaction can improve user performance costing less time and fewer errors than other conventional interactions. In short, a system integrated with our interface design will surely run with higher efficiency than previously.

After the success in *Human Adaptation*, which nudges the user into improve interaction, our next step is to discover the optimal interaction between the user and the gestural interaction system. Our first approach is to create a system that can proactively track the user and make the optimal sensing environments based on the user's habits. We called it *Machine Adaptation*. In our test interaction–hand pose estimation, our proposed scheme can effectively learn how to move the sensor to improve pose estimation confidence while requiring no ground truth hand poses. The results from our user study show that proactive sensing helps estimate users' hand poses with higher confidence compared with some baseline sensing. We further present an online model update to improve system performance.

Finally, we refined the idea of *Human Adaptation* and *Machine Adaptation* by combining them into *Mutual Adaptation*: discovering the optimal interaction for the the gestural system while nudging the user into it. This is very similar to how human tackle with the ambiguity in the interpersonal communication. The interaction system can not only help users adapt to itself, but also comes with the knowledge of how and where the user should adapt toward. We believe that this can tremendously improve the effectiveness in human-computer interaction compromised by ambiguity or obscurity.

A lot of potentials of future development lie in *Mutual Adaptation*: with VR/AR environment and drone become more and more prevalent, environments of controlled sensation will be very popular in the future. In VR/AR setup, in particular, the visual or audio sensing can be directly done by the robots or drones that are controlled with the computer remotely. Therefore, it would be even easier to drive users into what is expected. With such potentials,it's not hard to envision a future with computer interactions that change more dynamically with little deliberate human adaptation in the design.

We hope that you have found the the idea of interaction system we proposed interesting, and are inspired to explore the vet frontier of the coevolution of human and computer.