2014

Evaluation of a commodity VR interaction device for gestural object manipulation in a three dimensional work environment

Frederick Victor Thompson III
Iowa State University

Evaluation of a commodity VR interaction device for gestural object manipulation in a three dimensional work environment

by

Frederick Victor Thompson III

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Co-majors: Human-Computer Interaction; Mechanical Engineering

Program of Study Committee:

Eliot Winer, Major Professor

Stephen Gilbert

Jim Oliver

Iowa State University

Ames, Iowa

2014

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Designers and engineers working in the computer-aided drafting (CAD) and computer-aided engineering (CAE) domains routinely interact with specialized computer software featuring three dimensional (3D) work environments. These professionals must manipulate virtual objects or components within this 3D work environment, but typically use traditional interaction devices with outdated technology that are more suitable for 2D work tasks. Current CAD and CAE software is designed to accommodate outdated interaction technology, but this functionality comes at the cost of efficiency in the virtual workspace. A new class of affordable interaction devices with characteristics and specifications of high-end virtual reality interaction devices is now available to consumers. These commodity VR interaction devices monitor the position and orientation of a user's hands through space to control aspects of desktop software in ways that are impossible with the traditional mouse and keyboard pair. They can be integrated with CAD or CAE software to allow gestural control of objects throughout a 3D work environment.

To evaluate the feasibility of gestural control for 3D work environments, a commercially available commodity VR interaction device was selected and integrated with specific 3D software. Gestures to control aspects of the software are developed and organized into a taxonomy. Select gestures are integrated with the software and evaluated against traditional interaction methods, using the Natural Goals Operators Methods Selection Rules Language (NGOMSL) concept. The evaluation results show that gestural interaction is efficient for object manipulation tasks, but a traditional keyboard or mouse is more efficient for

basic tool selection tasks. Estimated learning times for each input method indicate gestural control takes about 30 seconds longer to learn than traditional interaction methods.

CHAPTER 1. INTRODUCTION

This thesis focuses on the development of 3D spatial gestures for user interaction within select software 3D work environments through the use of a commodity VR interaction device. An overview of the current state of personal computer (PC) interaction, including the shortcomings of traditional interaction devices and an argument for 3D gestural control is discussed below.

Interaction with the Personal Computer

"I can make just such ones if I had tools, and I could make tools if I had tools to make them with." - Eli Whitney, inventor

Humanity's industrious accomplishments are due to our ability to develop new tools to accomplish difficult tasks. An early scene change in Stanley Kubrick's *2001: a space odyssey* serves as a succinct metaphor to describe this. An intelligent human predecessor tosses the first primitive tool upward into the sky, transitioning to an image of a manmade spacecraft. The message is clear - tool innovators, and those who adopt new tools, outclass rivals through a competitive advantage.

Arguably, the paramount tool of human kind is the PC. It has simplified the way we do many tasks while improving the quality of the resulting output. PCs have empowered individuals to perform tasks that traditionally required a substantial amount of training and specialized equipment, increasing the productivity of the

individual. Naturally, an inexpensive tool that facilitates the needs of so many should expect to see widespread adoption.

Itself a collection of many individual innovations working together in harmony, today's PC is a ubiquitous workplace tool used in nearly every industry. Over the years, numerous innovations have reduced the cost and improved the power of PCs, and today's PC user works more efficiently and solves problems of higher complexity than the user of the 1970s. Despite these advances, the user's primary method of PC interaction – the mouse and keyboard – is relatively unchanged. These default interaction devices embraced by users do perform well for certain tasks, but there is no one-size-fits-all solution for computer interaction. Recent technology advances have lead to a new interaction device class that brings the capabilities of expensive, cutting-edge Virtual Reality (VR) interaction devices to a consumer price point. These commodity VR interaction devices are significantly different the standard mouse and keyboard, and even devices like joysticks, trackballs, or touchscreens, because they monitor the motion of a user's hands through space with six degrees of freedom (DOF). These devices can facilitate natural gestural interaction with PC software, especially software with a 3D work environment, at a price that allows widespread adoption. There are known advantages to three or more DOF interaction within a 3D work environment. Access to affordable devices that allow natural gestural interaction with software may provide a benefit to engineers, designers, and even the general consumer. This thesis explores how to successfully integrate a consumer VR input device with PC engineering and design software, without the need for a full user study.

The Keyboard and Mouse

Designers and researchers have worked for years to refine the interaction between a user and his or her personal computer. Despite their efforts, most PC users interact with a traditional keyboard and mouse, an older pair of devices that were designed before 3D software workspaces were a reality.

The typical PC keyboard is a manual input device with an external design that is nearly identical to typewriters of the late 19th century. The same basic design was used in teleprinter and keypunch devices before being integrated with electronic computers in the mid 20th century [1]. The keyboard is well suited for text input, despite the fact that the common QWERTY layout for keys was designed to slow down typists to keep mechanical typewriters from binding, and it has many other special-use keys for software-specific tasks, like the arrows and the escape keys. In different software environments, the user can actuate one or more keys as a "shortcut" to quickly execute a specific software operation. In the domain of a 3D modeling program used by engineers or designers, proficient users often utilize keyboard shortcuts for operations such as tool selection or to change the camera view in the scene. Current PC users interact with the keyboard in ways that the original typewriter designer never intended.

The standard computer mouse has a much briefer history than the keyboard. The first interaction device that resembles today's mouse appeared in 1968, but it was not until the release of Xerox's Star personal computer in 1981 that consumers could purchase a PC that included a mouse [2]. The device excels at pointing tasks within a 2D graphical user interface (GUI) because it directly maps the X-Y translation of the user's hand across a work surface to the location of an on-screen cursor. A typical mouse is controlled with a single hand; it is translated across a work surface and has

three or more buttons and a scroll wheel that can be actuated by the user's fingertips. Actuation of one or more buttons triggers a context-specific software operation dependent on where the mouse pointer is located.

PC users can simultaneously interact with both a mouse and keyboard to efficiently control software. The user holds the mouse in the dominant hand for pointing tasks and uses the other hand to execute keyboard shortcuts.

3D Work Environments

Many routine tasks performed with PC software (e.g. selecting items on the computer's desktop or interacting with a word processing application) can be thought of as occurring within a two dimensional environment. The typical computer mouse is well suited for pointing operations in these 2D environments because there is a direct relationship between mouse translation across the physical work surface and cursor

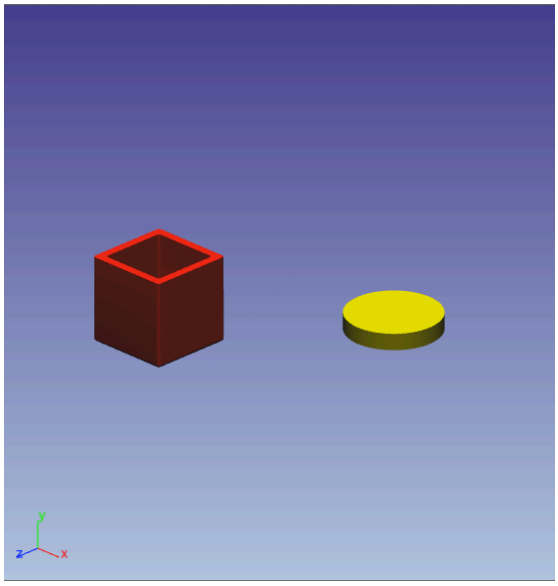

Figure 1. Isometric view of two objects positioned on the XZ plane

translation across the computer screen. A translation operation within a 2D environment, e.g. a "click and drag" action to move an icon across the desktop, is unambiguous for the user. Researchers define the disconnect between a computer user's goal and the tools and methods available to achieve the goal as the *gulf of execution* [3]. In the case of 2D mouse translation tasks, the gulf of execution is minimal. To complicate things, professionals in industries like engineering and computer graphics commonly use much more complex software with a 3D workspace. Computer-aided drafting (CAD), computer-aided engineering (CAE), and 3D computer graphics software tools all have to accommodate mouse and keyboard interaction within a 3 dimensional work environment.

User interaction in 3D environments can be very complex unless the mouse-controlled operations remain constrained to only one or two dimensions. Consider this scenario shown in Figure 1 above: assuming objects can be translated through the scene by clicking and dragging the mouse, if the user selects the yellow cylinder and translates the mouse in a given direction, where will the cylinder ultimately reside? If the mouse is pushed "up", or away from the user, the cylinder may move in the positive Y direction, the negative Z or X directions, or a combination of all three. In this case, the gulf of execution limits the user's interaction. The user's expectation for the translation direction and magnitude may not match how the software interprets the command.

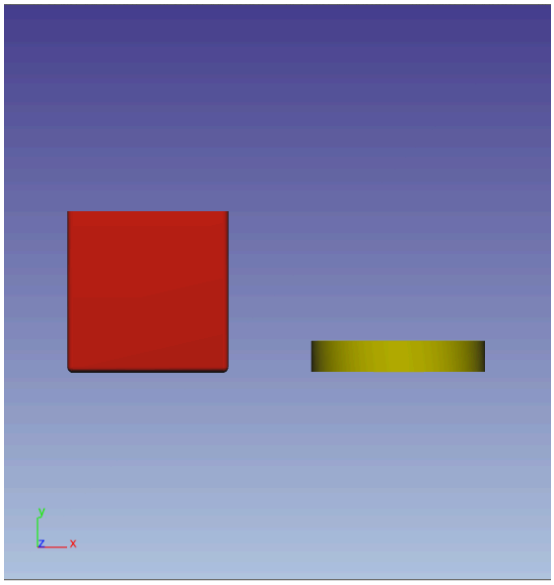

Figure 2. Orthographic view of two objects positioned on the XZ plane

Developers of 3D software use several solutions to address input ambiguity during operations like translation, rotation, or scaling of a virtual object in 3D space [4]. One common workaround is known as *view-based techniques*, where the user's view is limited to one or more orthographic scene views normal to one of three Cartesian planes, constraining the on-screen cursor to two axes of the work environment instead of three. An example of an orthographic view can be seen in Figure 2. Often, users will find that an orthogonal camera view does not reveal enough visual information about the scene, or is difficult to comprehend, and instead opt for an offset view that reveals more information, like the isometric view seen in Figure 1. In these cases, an alternate means of control is needed to accurately manipulate objects. In one solution, *controller-based techniques*, object controls are located in GUI windows or mapped to keyboard keys. Another solution, *virtual trackball techniques*, superimposes supplementary GUI elements in the scene, over the object, for the user to indicate the axis or axes on which they wish to operate. Researchers refer to a translation

manipulation element superimposed over an object in the scene as a skitter [5]. Figure 3 shows a virtual trackball superimposed on an object. Finally, *multiple-degree-of-freedom techniques* can control the virtual object through user interaction with an input device that tracks a user's hand in more than 2 axes.

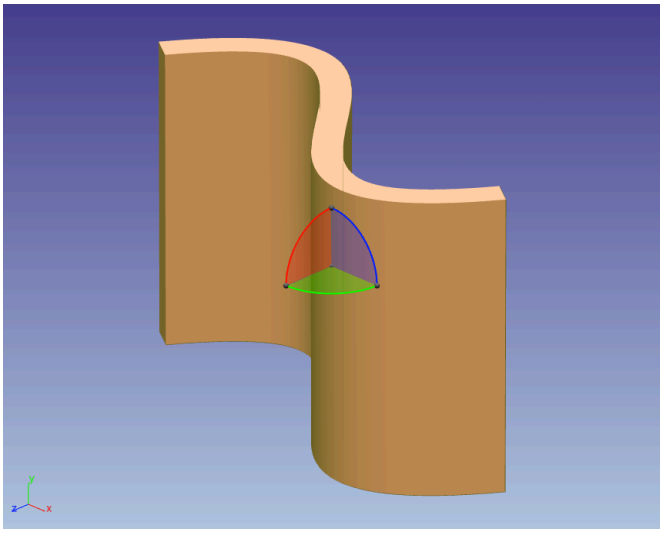

Figure 3. Virtual trackball manipulation in a 3D workspace

Ultimately, the first three solutions facilitate mouse interaction at the cost of efficiency. Mouse users have no means to manipulate an object across 3 axes in a 3D environment with the simplicity of dragging a file from the desktop into a folder. Multiple-degree-of-freedom techniques address this shortcoming and simplify user interaction.

Commodity Virtual Reality

Previous attempts to bring VR devices to the general consumer have been unsuccessful, largely due to hardware constraints and a poor understanding of user needs. One such attempt, Nintendo's 1990's-era home video game console *Virtual Boy* (seen in Figure 4), promised consumers an immersive stereoscopic experience on a

Figure 4. Nintendo's Virtual Boy HMD video game console

portable head mounted display (HMD). Instead, it left users with symptoms of cyber sickness due to hardware and software shortcomings and was discontinued after less than a year on the market [6]. Similarly, the *Power Glove* interaction device developed by Mattel for Nintendo's *Nintendo Entertainment System* in the 1980's promised video gamers hand tracking and gestural control, but was quickly rejected by consumers for it's imprecise control.

Commodity VR devices have simply provided an inadequate experience for everyday tasks such as interfacing with productivity software on a desktop computer, and instead are relegated to niche uses at best. At worst, the device is an entertainment

novelty that is incompatible with any other hardware or software without extensive modification and technical expertise. Presently, the availability, cost, and functionality of hardware is no longer a limitation and a new generation of commodity VR devices has emerged on the consumer market. For the purpose of this research, a commodity VR interaction device is defined as:

1. A device that allows natural user interaction through gestures and movements in 3D-space.
2. A device that is marketed to and priced for consumers, instead of researchers or industrial clients.
3. A device that is no more complicated to connect to a PC than a typical keyboard, mouse, or computer display.

Modern commodity VR interaction devices like the Nintendo Wii Remote [7], LEAP Motion Controller [8], Microsoft Kinect [9], and Sixense Razer Hydra [10], seen in Figure 5, are capable devices with an enthusiastic group of researchers and VR hobbyists researching the devices and developing new uses. Unlike previous commodity VR devices, these new ones are designed to easily connect to PCs and have support from the manufacturer to integrate the devices with other software. The commercial success of these devices and their use in academic research proves that commodity VR hardware is mature, however little development has been done to integrate these devices with the software commonly used by industry professionals.

Figure 5. Currently available commodity VR input devices: (Top left) Nintendo Wii Remote, (Top right) Microsoft Kinect, (Bottom left) LEAP Motion Controller, and (Bottom right) Razer Hydra.

Industry professionals and general consumers may not presently use low cost VR devices, but users are aware of VR and are capable of adapting to new computer interaction methods [11]. Public exposure to gestural and other non-traditional interaction at an affordable price has steadily increased in recent years. The first generation iPhone and second generation MacBook Pro, released by Apple Computers in 2007 and 2008 respectively, brought multi-touch gestural interaction into the homes and pockets of millions of users for the first time [12]. Sales of smartphones, many of which have a touchscreen interface, has increased to the point that over half of all American adults owned one in 2013 [13]. The strong smartphone market in the late 2000's eventually led the way to an emerging market of touchscreen tablet computers with operating systems designed for touch interaction. These tablets have an interface and workflow that is separate from the mouse and keyboard interaction of traditional PCs. Users adapted to new touchscreen-only interfaces specifically designed to accomplish goals in a computer environment lacking a keyboard and mouse [14]. In the entertainment domain, each of the three major video game console manufacturers,

Nintendo, Microsoft, and Sony, have brought novel VR interaction devices to the living room for use with their respective home video game consoles: the Wii [7], Xbox 360 [15], and PlayStation 3 [16].

Exposure outside of consumer devices is also increasing VR device awareness. Elon Musk, the founder and CEO of Space X, released a highly publicized promotional video titled "The Future of Design" in the fall of 2013, demonstrating engineering design with commodity VR devices [17]. Many online commenters compared Musk's demonstration to the futuristic technology seen in the movies *Iron Man* (2008) and *Minority Report* (2002), both of which prominently featured natural gestural computer interaction. In short, the general consumer is aware of new interaction methods and is comfortable interacting with devices without tactile button presses and pointing methods traditionally used with PCs.

The remainder of this thesis is structured as follows:

- Chapter 2 provides a review of research on 3D interaction from literature and highlights three current commodity virtual reality devices, how they have been used for research and hobbyists, and justifies the selection of a commodity VR device as a platform for testing of gestural interaction.
- Chapter 3 discusses the process of integrating the selected commodity VR device with software, the development of a gestural taxonomy, and the operations within software that receive gestural control.
- Chapter 4 provides an evaluation of the implemented gestural interaction compared to traditional mouse and keyboard interaction in a 3D work

environment through the Natural Goals, Operators, Methods, and Selection Rules Language (NGOMSL) concept.

- Chapter 5 is a discussion of the results, conclusions, and future work.

CHAPTER 2. REVIEW OF LITERATURE AND COMMODITY INTERACTION DEVICES

Literature Review

INTRODUCTION

As mentioned before, the disconnect between a computer user's goal and the tools and methods available to achieve the goal is known as the *gulf of execution*. In an attempt to minimize the gulf and ease user interaction, imaginative designers have developed a wide variety of manual pointing and locating devices of differing modality. Despite their outward differences, all input devices are naturally constrained to a set of common movements and actions defined by human physiology. A typical PC mouse and a high-tech VR wand rely on similar physiological abilities, but the user manipulates a mouse across a desktop surface and a wand through space. Jacob asserts that VR interaction devices have an advantage over the traditional mouse and keyboard because they utilize a user's "pre-existing abilities and expectations" of the real world rather than relying on "trained behaviors" for software interaction [18].

Why do we interact with the outdated mouse and keyboard when advantageous interaction devices are available? Bill Buxton succinctly explained the divide between our own physiology and our chosen devices for PC interaction with an imaginative description of the misconceptions future anthropologists would hold after discovering a hidden cache of functional computer hardware and software from the 1980s [19]. In Buxton's words:

> "My best guess is that we would be pictured as having a well-developed eye, a long right arm, uniform-length fingers and a 'low-fi' ear. But the dominating characteristic would be the prevalence of our visual system over our poorly developed manual dexterity. Obviously, such conclusions do not accurately describe humans of the twentieth century."

His main argument is that existing computer interface devices do not fully utilize our inherent dexterous and sensory abilities and may actually hinder our ability to interact with PCs. A 2D pointing device like a mouse is adequate or even superior for certain tasks, like interacting with the user interface (UI) in a word processing program or other general desktop productivity software, but it lacks a degree of freedom when interacting with a 3D interface.

JUSTIFYING THE NEED FOR 3+ DOF INTERACTION

Many research groups have worked to identify the best uses for PC interaction devices other than the mouse and keyboard, ranging in complexity from a trackball to custom-built VR controllers. Beaton et al. compared a 3D trackball, a traditional 2D mouse, and a special 3D thumbwheel controller[11] with a 3D pointing task, and found that mouse users had a higher positioning error and took longer to complete the task [20].

A number of researchers built their own devices to prove the merits of three or more DOF interaction. Djaajadiningrat et al. performed a comparison of varying user DOF during a physical sphere rotation task representative of an action typically found in

[1] The thumbwheel device used linear rotary knobs, similar in functionality to the scroll wheel on a modern mouse, but with three separate scroll wheels aligned to the 3 Cartesian axes.

3D modeling software, concluding that desktop interaction devices which offer fewer than three degrees of simultaneous control are less efficient than three DOF alternatives [21]. A user restricted to rotational input along a single orthogonal axis while attempting to rotate an object to a given orientation took longer, required more rotation actions, and was less comfortable than a user who was able to freely rotate in multiple axes without restriction.

Jones designed and fabricated a low-cost gimbal-mount six DOF desktop interaction device intended for rotation and translation in 3D objects with the non-dominant hand, leaving the dominant hand free to control a traditional mouse for object selection [22]. The device was designed for use in a fixed location on the desktop surface, so the user may rest an elbow during interaction. It was compared against a traditional 2D mouse for 3D object manipulation tasks. The researchers found that the device worked well for object translation, however users were reported to have encountered problems during rotation, either from interaction issues or mechanical problems of the device. An important takeaway of the study is that users typically find rotation tasks with parametric control (where displacement from the origin controls the rate of rotation) to be more difficult than incremental control (where the displacement from the origin controls the absolute rotation value), while either parametric or incremental control are equally suited for translation tasks.

Fröloch and Plate built their own three DOF controller to prove the benefit of maintaining a common coordinate system between input device and a 3D environment [23]. It was a cube with an embedded six DOF tracker that is intersected by three orthogonal rods, which are pushed and pulled to control motion along the three axes. Users found the device to be easy to learn because of the inherent proprioceptive cues.

Despite the technical shortcomings and low commercial success of previous commodity VR hardware described in the first chapter, researchers have found ways to bring the virtual reality experience to an affordable point through modification of existing commodity hardware [24], [25]. Some researchers have worked to integrate VR functionality with the workflow of engineers and designers without regard to cost or feasibility [26].

Summary of Literature Review

Professionals who work within 3D work environments may benefit from gestural interaction. The current body of research indicates that a traditional mouse and keyboard pair cannot perform specific tasks in a virtual 3D space with the accuracy, precision, and speed of devices that register input in three dimensions. Professionals need a device that allows gestural control of object manipulation operations like translation and rotation to execute designs quickly and accurately.

Commodity VR Interaction Device Review

Introduction

There are many commodity VR interaction devices available to consumers today, with more devices currently in development. VR interaction devices accomplish their novel interaction method by tracking the movement of the user through space through a camera system or handheld sensor. Despite the similar capabilities of devices in this domain, the devices often use dissimilar technologies with intrinsic advantages and disadvantages. Different use-cases necessitate different input methods. While one situation may benefit from a handheld controller tracked through 3D-space, another

may benefit from a camera system that can simultaneously track whole body movements of several users.

Many commodity VR input devices are available today, but three devices were explored for the purpose of this discussion: the Nintendo Wii Remote, the Microsoft Kinect, and the LEAP Motion Controller. These devices were selected because of their superior capabilities, their flexibility within a research environment, and because they are representative of the spectrum of devices available to consumers. The surveyed devices are similar in that they can track a user's hand movement with six DOF and they are suitable for use in an office environment.

The Wii Remote

The Wii Remote, introduced in 2006, is a wireless handheld motion controller designed for interaction with Nintendo's home video game console, the Wii. Users operate the device (shown in Figure 6) in a single hand, and it has a roughly rectangular in shape that is similar to a typical television remote control. It has a number of buttons for user interaction, an audio speaker, a vibratory motor, four LED lights for user

Figure 6. Nintendo's Wii Remote

feedback, and can communicate with a PC over a standard Bluetooth wireless protocol. However, the most novel feature is the controller's motion tracking capability. To detect the motion of the user's hand, the controller has an onboard three DOF accelerometer unit and an optical IR sensor, which tracks the motion of the controller relative to a static IR LED light bar. The accelerometer and camera systems work together; the accelerometers measure general motion but are susceptible to measurement error and drift, while the IR sensor augments the detection of more precise tasks. In a standard use, where users manipulate the device like a VR wand, the IR light bar is located above or below the display. The IR sensor detects it's own position relative to the static IR light bar to accurately detect the translation and rotation of the controller. Researchers have developed novel uses for the controller in this traditional configuration, as well as a reversed configuration where the controller acts as a stationary IR camera to detect the dynamic movement of the IR light bar.

Johnny Lee is the pioneer of Wii Remote integration with the PC. He brought attention to the use of use of this device for serious human-computer interaction (HCI) applications with a demonstration of how to use the device as a basis for a low cost digital whiteboard, a head tracker for fish tank VR visualization, and a natural gesture PC interaction device that recognizes the user's fingertips [27]. Others built on Lee's work to find new uses for the Wii Remote. Lin et al. used the Wii Remote as a camera to track infrared markers affixed to the user's hands for gestural input [28]. Interaction was limited to basic 2D productivity tasks such as progressing through a slideshow or resizing an image, but users generally considered gestural input to be valuable once they learned how to perform the interactions.

Others have successfully replaced expensive VR hardware with a Wii Remote. Pavlik and Vance used the Wii Remote to track the head position of a user equipped with a head-worn IR emitter in an immersive stereoscopic 3D environment [29]. Their work built on Lee's original "head tracker" demonstration from 2008, but is compatible with immersive CAVE implementations in addition to desktop PCs. Zhu et al. used the Wii Remote as an IR tracker camera to record motion capture data for authoring 3D model animations [30]. Such systems normally cost thousands of dollars, but Zhu's Wii Remote camera system was able to accurately track IR markers and for a fraction of the price.

In addition to developing new uses for the device, researchers have compared the performance of the Wii Remote to other pointing devices. Ardito et al. compared the performance of a Wii Remote used as a wand against both a standard PC mouse and keyboard pair and a typical two-joystick game pad for translation and rotation tasks in a 3D environment [31]. The team found that users of the Wii Remote completed tasks slower and with more errors than users of the two other devices. Additionally, users rated their experience with the Wii Remote as dissatisfactory and considered the device difficult to use. Gallo et al. compared the performance of a Wii Remote used as a wand and a typical mouse and keyboard for two-axis and three-axis rotation tasks with 3D medical data [32]. Wii Remote users could simultaneously control object rotation along one or two axes, depending on condition. The team found that a mouse and keyboard outperformed the Wii Remote in both task completion time and accuracy, and the time difference between two-axis and three-axis rotation tasks indicates that two-axis tasks are easier to control regardless of input method.

The evaluations mentioned above used the Wii Remote as a 2D pointer for selection with roll/pitch/yaw control for 3D rotation tasks. The device was operated like a wand, but in practice behaved like a standard mouse with an additional degree of freedom (roll). Overall, the findings indicate that wand-style input devices are not appropriate for 3D manipulation tasks in physical environments that are able to accommodate a keyboard and mouse.

The Kinect

Microsoft's Kinect is a depth-sensing camera system used for gestural interaction with video games on Microsoft's Xbox line of home video game consoles and Windows PCs. In practice, the device can detect spatial gestures performed with the user's hands, track the movement of people and objects through a room-sized volume, and detect voice commands. Although it first premiered in 2010 as an interaction device for the Xbox 360, Microsoft released a developer-friendly version with an updated SDK for PC developers in 2012. An advanced version of the device was released for the newer Xbox One home video game console in 2013, and Microsoft went on to release a developer-friendly version of this updated Kinect in 2014. Both the 2012 and 2014 developer versions of the Kinect have an IR depth sensor as well as a standard RGB camera, a 3 DOF inertial measurement unit (IMU) to detect device movement, a microphone array for audial input, and use a wired USB connection to communicate with software on a PC. The Kinect excels at skeletal modeling in a room-sized volume, but has a shallow minimum viewing distance that makes it unreliable at desktop workstations. A Kinect can be seen in Figure 7.

Figure 7. Microsoft's Kinect Motion Controller (2012 version)

Microsoft supports research and development for the Kinect through official software libraries. In contrast to the Wii Remote, which was used as both a handheld wand and as a stationary IR tracking camera, the Kinect is most frequently used in the intended hardware configuration; fixed to a stationary position where it detects spatial gestures of one or more users.

Researchers have developed novel uses for the spatial tracking and gestural interaction afforded by the Kinect. Dave et al. used a Kinect to control a voxel-based virtual clay modeling system at a desktop environment [33]. The system played to the strengths of the Kinect and used whole-body gestural recognition to control operations like object selection and manipulation. Gallo et al. successfully used gestural commands with the Kinect to control the viewpoint of 3D medical images [34]. Santos et al. successfully used the device to control objects in an augmented reality scene without having to calibrate for skin color, a shortcoming of traditional 2D computer-vision based interaction techniques [35].

In addition to developing new uses for the device, researchers have compared the effectiveness of the Kinect against other input devices. Tilak Dutta compared the Kinect to an expensive motion capture IR camera system manufactured by Vicon, and found

that the Kinect can perform adequately as an IR marker tracker at 1 to 3 meter distances [36]. Francese et al. compared gestural control on the Kinect and Wii Remote for navigation within a 3D navigable environment viewed on a projector screen [37]. Users preferred the natural gestures afforded by the Kinect and felt they were more transparent and less intrusive than the wand-style control of the Wii Remote. Juhnke evaluated the Kinect against a typical mouse and keyboard for windowing tasks with medical imaging data [38]. The task required medical students to manipulate a two-handled 1D slider bar to reveal a specific feature of the medical image. In the Kinect condition, the user must translate his or her hands across a 1D axis in front of the Kinect, with each hand directly mapped to the position of a single slider bar handle. Mouse users could only control one handle at a time, due to the nature of mouse interaction, while Kinect users could manipulate both handles simultaneously. Participants were able to accurately reveal the correct density more slowly with the Kinect than with the mouse, and favored the mouse for small and precise adjustments. The author speculates the Kinect's performance is due to its inability to accurately distinguish user hands at a close viewpoint.

The Kinect appears to be a capable device that affords a gestural interaction experience superior to the Wii Remote. It has successfully sold to consumers and researchers, and has become a fundamental classroom tool to give students experience with gestural interaction technology [39]. Despite it's success, it performs inadequately at desktop workstations because it is designed for use at ranges of 20 inches to 13 feet. It is also known to unreliably detect users when multiple users are within the view volume. The Kinect is an undesirable input device for professionals seated at a work

computer, but is a capable interaction device for a single individual if used from a distance and paired with a large display or a projector screen.

The LEAP Motion Controller

The LEAP Motion Controller is an optical (shown in Figure 8) six DOF tracker that specializes in tracking a user's hands, fingers, or tools within a relatively small volume. The device is relatively new, as public sales only started in 2013, but developers have already shared home made software and video demonstrations of its capability. Its functionality is similar to the Kinect's, but it is not used for whole-body gestural interaction because of its small view volume. Instead, it is designed for use at desktop workstations and has an operational range of 1 inch up to 2 feet. The manufacturer supports researchers and developers with official libraries to integrate LEAP control with existing software.

Figure 8. The LEAP Motion Controller

Research with the LEAP is less plentiful than the previous two devices because of its relative new arrival on the consumer market. Developers have not "hacked" the device to improve its capability because the manufacturer frequently updates the software libraries with performance improvements. The majority of published research

has assessed the LEAP's capabilities and performance for gestural control in specific use-cases.

Weichert et al. evaluated the accuracy of the LEAP and found that it can reliably and accurately track a human hand or tool in a static posture or through a 3D trajectory in space. It can be adapted to control a variety of operations and is more accurate and precise than competing products of a similar cost [40].

Mauser and Burgert used the LEAP to advance through medical imaging data slides and control medical instruments, which necessitated both 2D and 3D gestural input [41]. Users were able to successfully control the systems with the LEAP, however the team learned that the device readily detects unintended gestures within the view volume, causing unintentional execution of software operations. The team implemented "lock" and "unlock" gestures to allow or disallow recognition to solve this problem.

The LEAP Motion Controller and the Kinect share several common benefits: both can track the movement of hands and tools through space to allow gestural control of PC software, and both have official software libraries from the manufacturer. The LEAP has the potential to excel at desktop interaction because it is more reliable than other commodity VR devices and can operate within the physical space of a desktop workstation.

Summary and Conclusion of Literature and Device Review

There are many additional commodity VR devices available today that accommodate gestural interaction. Despite the common availability and advantages of these devices, most desktop workers still perform software interaction with a keyboard and mouse. Skilled workers who regularly interact with 3D software, like designers and

engineers, use interaction technology dating back to the 1960s. They are not equipped with modern commodity VR interaction devices, despite the potential advantages in time and accuracy for routine tasks 3D work environments.

The LEAP Motion 3D controller is a promising platform to test VR interaction with PC software because it does not have the shortcomings of the other surveyed commodity VR devices. It has a focused view volume and has a shallow minimum depth, making it appropriate for interaction at a desk in an office environment. The manufacturer supports the device through software libraries that allow integration of the device with commercial software. These libraries have a robust skeletal model to track human fingers and hands, allowing precise gesture detection. Finally, a LEAP user does not need to grip a controller or use exaggerated whole-body gestures, minimizing user fatigue. For these reasons, the LEAP Motion Controller was selected as a platform to explore gestural interaction in 3D a work environment.

Research Issues

Two research issues have been identified, based on the current state of research in desktop VR interaction:

1. **How can gestural interaction be evaluated without conducting a full user study?**

   User studies require time and resources that may not be available to software developers looking to integrate commodity VR interaction with their software in an expedited time frame. Interaction evaluation without a user study allows rapid development and deployment in a market with continuously improving

interaction devices.

2. **What type of gestures should be used for object manipulation in 3D work environments?**

Unlike the interaction modality afforded by the keyboard and mouse, spatial gestures are open-ended and constrained only by hardware, software, and user limitations. Existing research indicates three or more DOF interaction is beneficial for 3D object manipulation. Stakeholders looking to implement gestural interaction in software with commodity VR devices must select appropriate gestures for the task.

CHAPTER 3. METHODS

Overview

A software platform to evaluate gestural interaction in 3D work environments with a LEAP Motion Controller is needed. The selected platform is a CAD-like conceptual design software developed by researchers at Iowa State University, called the Advanced Systems Design Suite (ASDS) [42]. Engineers and designers use ASDS to quickly visualize and assess design concepts with imported CAD geometry and a library of primitive shapes early in the design process, when exact dimensions and materials are undecided or unknown. It was selected because it has a concise set of design manipulation and assessment tools to simplify user interaction within the 3D work environment, and because the source code is readily accessible to researchers at Iowa State University's Virtual Reality Applications Center, which eases the creation and evaluation of gestural control with the LEAP Motion Controller.

Official C++ libraries supplied by LEAP Motion were integrated with the ASDS source code [43]. The gesture recognition and skeleton modeling capabilities of this library were utilized to simplify the integration of gestural interaction with ASDS. The final gestural interaction developed for ASDS utilized a subset of a gestural taxonomy specifically developed for desktop interaction in 3D work environments. This taxonomy is based on the findings of other researchers, human physiology, and the limitations of an optical hand tracker. Select gestures from this taxonomy were integrated with ASDS.

Design Assumptions

Several assumptions of the LEAP Motion Controller and the ASDS were made during planning and development.

1. The 3D work environment in the ASDS is assumed to represent a typical 3D work environment encountered in other engineering software tools like Dassault Systèmes' SolidWorks, as well as non-engineering 3D graphics software like AutoDesk's Maya.
2. The LEAP Motion Controller's tracking accuracy and method is representative of other commodity interaction devices.
3. Gestures developed for use with the LEAP Motion Controller could also be designed for other devices.

Design Issues

Research on gestural interaction with comparable devices guided the development of the gestural taxonomy, as well as the selection process to decide the gesture and associated software capability it would control. An understanding of the best way to orient and position the LEAP Motion Controller is needed prior to gesture development.

Device use and orientation

LEAP Motion intends their device to rest flat on a stationary desktop surface, with an approximately semi-spherical view volume centered above the device. Despite the manufacturer's intent, the device can be oriented in any position to suit a user's needs. Researchers Han and Gold explored different orientations of the LEAP Motion

Controller for 3D "tapping" tasks and found that angled and inverted orientations did not detect fingers as accurately as the intended upright orientation [44]. The conclusion is that developed gestures must be salient when observed from below.

Gestural controls and the associated ergonomics

Many people regularly interact with touchscreen interfaces on smartphones and tablets that necessitate novel touch gestures. Gestural interaction with touch screen devices has been successfully used for object control in 3D work environments [45], but the necessary considerations differ from spatial gestures: touch gestures are constrained to a relatively small 2D space and provide tactile feedback, while spatial gestures are through a larger 3D space without tactile feedback.

Nielsen et al. argue for a human-centric approach when designing gestures [46]. Gestures should be designed to the physiological and cognitive limitations of the user rather than the technical limitations of the interaction device. Existing implementations that have not addressed human concerns leave users fatigued with an effect known as "gorilla arm syndrome" [47]. With this in mind, ensuring the gestures are intuitive, easy to use, and not harmful to the user was of primary importance during taxonomy development. Hinckley et al. were early explorers of free-space gestural input and developed a framework to understand these concerns [48]. Several of their suggestions guided the development of the gesture taxonomy:

1. Arbitrary gestures for operations like translation, rotation, and scale can be difficult to use, so the manipulation of virtual objects should directly map to a user's manipulation of a tangible object.

2. Object manipulation should occur in a spatially relative position, instead of absolute position.
3. The user should be able to interact with two hands to improve task efficiency. The second hand can perform spatial gestures or interact with a different device, like a mouse.
4. Gestural controls should operate on similar attributes of virtual objects (e.g. translation, rotation, and scale instead of translation, rotation, and color).
5. Interaction should consider ergonomics to avoid injury and fatigue.

Other research guided how to register the user's end effector. Zhai et al. performed a comparison of six DOF input devices with and without finger manipulation, and found that input devices controlled by the small muscle groups in the fingers perform better than devices that use larger muscle groups in the arm [49]. Additionally, an experiment performed by Djaajadiningrat et al. revealed that user comfort varied depending on the number of fingers allowed during three DOF sphere rotation, concluding that desktop VR input devices should register three fingertips to maximize user comfort [50].

The gestures must be easy for users to learn and execute. While interacting with 3D work environments, traditional mouse and keyboard users experience the benefit of direct input mapping and tactile feedback. The operation of a mouse and keyboard with typical desktop software is intuitive, as physically moving the mouse in a direction will translate the on-screen cursor an amount in the same direction through a position-to-position mapping. Additionally, a key press on the mouse or keyboard results in audial and tactile user feedback. Each possible method of input afforded by a keyboard or

mouse is visible to the user, with the exception of multi-key shortcuts on the keyboard or non-traditional mice with a touch-sensitive surface.

In contrast, spatial gestures are actions that a user must memorize and recall. Novice users cannot rely on visual cues to recall gestures the way that novice keyboard users can "hunt and peck" for the correct key. The common use of keyboard shortcuts indicates that users are capable of recalling ambiguous commands to execute actions within a "windows, icons, menus, pointer" (WIMP) GUI, but spatial gestures do not provide the user with the tactile feedback provided by interaction with physical hardware. Unreliable gestural interaction may leave the user unsure of the software's status when a command is attempted but no result occurs, although this can be mitigated with audial or visual feedback cues. If the software is unreliable and does not provide feedback, a user may wonder: "Did I execute the command correctly? Is there something wrong with my device? Is my computer just slow?". In short, the gestures must be memorable and provide visual, audial, or tactile feedback to the user upon successful execution. Research [51] suggests that care must be given to ensure a user understands the semiotics of spatial gestures and how individual gestures relate to one another.

The findings from the design issues outlined in the previous section guided the development of the gesture taxonomy shown in Table 1 below. This taxonomy outlines every feasible motion that can be utilized as a gesture for desktop interaction with a commodity VR device similar to the LEAP Motion Controller. Care was taken to select gestures and actions that are unlikely to cause injury through repetition while being salient to optical trackers. Current tracking software for commodity optical tracking devices cannot reliably interpret gestures that contort the user's hands into complex

configurations or occlude multiple digits, so a preference is given to gestures that use the extension or flexion of one or more fingers. This minimizes self-occlusion and ambiguity to ensure reliable recognition by the tracking software.

Gestures from the taxonomy can be implemented to control a variety of software functions with an optical tracking input device. The specific taxonomy actions and postures were developed with consideration for user ergonomics, but were not validated through a user study. Designers can create novel gestures by combining the listed actions together, resulting in a unique gesture vocabulary that is suitable for a given application. However, individual actions should be selected from different segments to avoid complex or awkward gestures. For example, an action that uses two fingers extended and swept horizontally with the wrist, elbow, and shoulder is simpler than a gesture that extends one finger, then two other fingers, while swept horizontally in the same way.

Not every action in the taxonomy was included in the designed implementation. Instead, specific motions and postures from the taxonomy were combined and implemented to execute specific software operations that may benefit from spatial gestures and six-DOF interaction. Actions included in the designed implementation are designated with bold text.

Table 1. Gestural taxonomy

Segment	*Sub-Segment*	*Action*	*Note*
Fingers	1 Finger point (index)	Extension	
		Flexion to Extension	Close to open
		Extension to Flexion	Open to close
		Modulate extension to slight flexion	Vertical wag (similar to the action needed to click a mouse)
		Horizontal adduction/abduction	Horizontal "wag"
		Clockwise circumduction	Move around in circle
		Counterclockwise circumduction	Move around in circle
	1 Finger point (thumb)	Extension	
		Flexion to Extension	Close to open
		Extension to Flexion	Open to close
		Clockwise circumduction	Move around in circle
		Counterclockwise circumduction	Move around in circle
		Horizontal adduction/abduction	Horizontal "wag"
	2 Finger point (index and middle)	**Extension**	
		Flexion to Extension	Close to open
		Extension to Flexion	Open to close
		Modulate extension to slight flexion	Vertical wag (similar to the action needed to click a mouse)
	3 Finger point (index, middle, and ring)	**Extension**	
		Flexion to Extension	Close to open
		Extension to Flexion	Open to close
		Modulate extension to slight flexion	Vertical wag (similar to the action needed to click a mouse)
	3 Finger point (index, middle, and thumb)	**Extension**	
		Flexion to Extension	Close to open
		Extension to Flexion	Open to close
		Modulate extension to slight flexion	Vertical wag (similar to the action needed to click a mouse)
	4 Finger point (all fingers except thumb)	**Extension**	
		Flexion to Extension	Close to open
		Extension to Flexion	Open to close
		Modulate extension to slight flexion	Vertical wag (similar to the action needed to click a mouse)
	5 Finger point (all fingers and thumb)	Extension	
		Flexion to Extension	Close to open
		Extension to Flexion	Open to close
		Bring together	
		Fan apart	
	Thumb + 1 additional digit	Tip pinch	Pinch thumb tip to the tip of any finger
	Thumb + 2 additional digits	Tip pinch	Pinch thumb tip to the tips of any two adjacent fingers

Table 1 continued. Gestural taxonomy

Segment	*Action*	*Note*
Wrist	Flexion	Bend forwards
	Extension	Bend backwards
	Ulnar deviation	Bend towards little finger
	Radial deviation	Bend towards thumb
	Flexion to extension	
	Extension to flexion	
	Ulnar to radial deviation	
	Radial to ulnar deviation	
Forearm	**Pronation**	Palm facing down over desk
	Supination	Palm facing upwards
	Pronation to supination	
	Supination to pronation	
Elbow	**Flexion to extension**	Open arm
	Extension to flexion	Close arm
Shoulder	**Extension**	Lower arm down
	Flexion	Lift arm up
	Abduction	Move arm out of plane
	Adduction	Move arm into plane
	Lateral rotation	Cyclist "stop" signal
	Medial rotation	Cyclist "right turn" signal

Design Approach

Integrating the LEAP Motion Controller with ASDS was possible with LEAP Motion's official software development kit and access to the ASDS source code. Consideration of which software operation should receive gestural control, as well as which gestures to use, was of critical importance. The design implements the LEAP Motion Controller in a standard configuration, meaning the device rests on a stationary desk surface and detects spatial gestures that occur above it. The developed gestures use movements from the user's elbow, wrist, and fingers to reduce user fatigue and utilize inherent fine motor control.

TOOLS IN ASDS

A toolbar at the top of the main ASDS window contains 13 icons that serve as shortcuts to commonly used commands found within several pull-down menus as shown in Figure 9. The toolbar separates the icons of similar tools by proximity into three major icon families: file management, design component management, and concept manipulation and assessment.

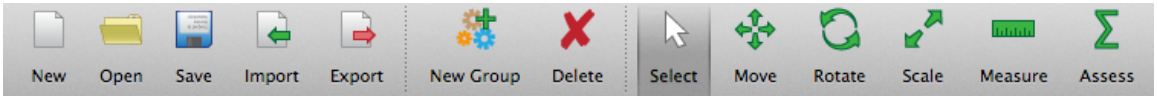

Figure 9. The ASDS Toolbar

File management tools allow the user to perform actions like opening, closing, and saving designs. The design component management tools allow a user to create groups of components for hierarchical categorization, or delete components. The third shortcut family, concept manipulation and assessment, is used to move, scale, and measure design components or an entire design. Of the three separate groups of shortcut icons, ASDS users most frequently interact with the concept manipulation and assessment family, so it was the focus for gestural interaction. Details of the concept manipulation and assessment tools appear in Appendix A.

From the concept manipulation and assessment family, the move, rotate, and scale tools are used frequently during design creation and modification, while the measure and assess tools are used to verify whether a design fits within given constraints. The select tool is used to indicate which component or components are to be affected by the other tools, as suggested by its name. Ultimately, the move, rotate, and scale tools received gestural interaction because of their frequent use, precise

capability, and their control of similar object attributes. The background research suggests that users will benefit the most from gestural interaction for these three tools.

Shortcut gestures

This implementation aims to streamline the workflow by enabling a unique "shortcut gesture" for each tool. With traditional mouse and keyboard interaction, a user can select a tool to use in two ways: by moving the mouse pointer to a tool's icon and clicking the mouse button, or by executing a keyboard shortcut. Through the developed gestural interaction, the user can select one of these three tools by expressing the associated shortcut gesture within the view volume of a LEAP Motion Controller. The shortcut gesture accomplishes the same result as traditional tool selection methods. The tool becomes active and the user can now perform the operation associated with the tool on the object. The shortcut gestures behave as follows:

Select the *move* tool – User extends two fingers and swiftly moves the extended fingers horizontally from right to left across the view volume, shown to the right in Figure 10.

Figure 10. The shortcut gestures for the move tool

Select the *rotate* tool – User extends three fingers and swiftly moves the extended fingers horizontally from right to left across the view volume, shown to the right in Figure 11.

Figure 11. The shortcut gestures for the rotate tool

Select the *scale* tool – User extends four fingers and swiftly moves the extended fingers horizontally from right to left across the view volume, shown to the right in Figure 12.

Figure 12. The shortcut gestures for the scale tool

Manipulation Gestures

The gesture integration goes beyond simple tool selection. After a user selects a tool, he or she can execute a separate manipulation gesture to perform the tool's operation. With traditional mouse and keyboard interaction, a user interacts with a skitter to accomplish the following: translate in one or two dimensions, rotate in one dimension, or scale in one, two, or three dimensions.

In this implementation, the LEAP Motion Controller measures the translation and rotation of a user's palm throughout the view volume and maps the position and

orientation of the hand to the position, rotation, and scale of the object, depending on which tool is active. The relations between hand position and orientation and the object position, orientation, and scale have a relative mapping, rather than an absolute mapping. In practice, this means that the difference between the start and end conditions of the user's hand dictates the magnitude and direction of translation, rotation, or scale operations that occurs on the virtual object.

The object manipulation gestures behave as follows:

Translation – When the *move* tool is active, object translation in three axes is relatively mapped to the position of the user's hand with three or more extended fingers in view volume.

Rotation – When the *rotate* tool is active, object rotation in three axes is relatively mapped to the orientation of the user's hand with three extended fingers in view volume.

Scale – When the *scale* tool is active, object scale is relatively mapped to the position of the user's hand with variable axes of control. Three extended fingers translated in any direction indicates a scale in three dimensions. A single extended finger translated in any direction indicates a scale in the single gestured direction.

An issue with gestural interaction [48] is that spatial manipulation requires a means of "clutching", or turning on and off, the relation between hand position and object manipulation. Clutching allows the user to specify when the software must track their hand, and when they are simply repositioning it or removing it from the view volume.

Without clutching, a user who intends to move his or her hand out of the view volume after completing an operation will continue to manipulate the object until the hand is no longer seen by the input device. This will surely frustrate a user attempting to accurately control objects. Clutching also accommodates relative mapping between hand position and object position in cases where the view volume cannot accommodate the size of the 3D work environment. The user can move his or her hand to the edge of the view volume, engage the clutch to disable the link between hand and object, move his or her hand back into the view volume, disengage the clutch, and continue translation. Clutching and manipulation gestures appear above in Figure 13.

Figure 13. Manipulation gesture with clutch engaged (left) and disengaged (right)

In this implementation, users engage or disengage the clutch by opening or closing their fingers, affecting the amount of "grip" they express. Grip is readily detectible by the LEAP Motion Controller and can modulate without affecting in situ gesture recognition. The clutch engages (gestures do not operate on objects) by opening the palm completely flat, and disengages (gestures operate on objects) by slightly closing the hand to a resting position. In this way, a user can use a single hand to perform manipulation operations and control the clutch.

Design summary

The development of gestures and integration with the ASDS aimed to address the findings of previous researchers within the constraints of the LEAP Motion Controller. As an example scenario, a user attempting to move and rotate an object on an existing model could do the following:

1. Use traditional mouse and keyboard methods to import existing geometry and select which component to adjust.
2. Execute the shortcut gesture for the *move* tool. Engage clutch.
3. Disengage clutch. Translate his or her hand through the view volume to translate the selected object in one axis, utilizing the clutch to reposition their hand, if needed.
4. Engage clutch. Execute the shortcut gesture for the *rotate* tool.
5. Disengage clutch. Rotate his or her hand through the view volume to rotate the selected object three axes simultaneously, utilizing the clutch to reposition their hand, if needed.
6. Execute the shortcut gesture for the *move* tool. Engage clutch.
7. Disengage clutch. Translate his or her hand through the view volume to translate the selected object in two axes, utilizing the clutch to reposition their hand, if needed.
8. Execute the shortcut gesture for the *scale* tool. Engage clutch.
9. Disengage clutch. Translate his or her hand through the view volume to scale the selected object in three axes, utilizing the clutch to reposition their hand, if needed.

10. Engage the clutch and remove hand from view volume.
11. Save file and exit program.

In this manner, a user can quickly alternate between selecting tools and performing operations necessary to accomplish a design, with one hand.

Table 2 outlines the selected software tools, *move, rotate,* and *scale,* and their corresponding shortcut and manipulation gestures. The listed shortcut gesture is a gestural analog to the keyboard shortcut used to select the tool, while the manipulation gesture is the behavior the user must express to achieve the desired operation with the chosen tool. Manipulation action is a listing of which kinematic functions of the human body are required to execute the manipulation gesture and is selected from actions in the gesture taxonomy shown in Table 1. Each tool utilizes the clutch operation to enable or disable the relationship between gestures and object manipulation.

Table 2. Gestural shortcuts and operations in ASDS

Tool	Shortcut Gesture	Manipulation Gesture	Manipulation Actions
Move	Translate two fingers right-to-left	Three finger point (index, middle, and thumb), translated within view volume. Relative mapping between user's end effector and selected object.	Three finger point (index, middle, and thumb) extension. Wrist flexion/extension and ulnar/radial deviation. Forearm pronation. Elbow flexion/extension. Shoulder flexion/extension, adduction/abduction, and lateral/medial rotation.
Rotate	Translate three fingers right-to-left	Three finger point (index, middle, and thumb), rotated within view volume. Relative mapping between user's end effector and selected object.	Three finger point (index, middle, and thumb) extension. Wrist flexion/extension and ulnar/radial deviation. Forearm pronation. Elbow flexion/extension. Shoulder flexion/extension, adduction/abduction, and lateral/medial rotation.
Scale	Translate four fingers right-to-left	Three finger point (index, middle, and thumb), translated within work area controls 3-axis scale. Single finger point (index), translated along an axis within the view scales object along hand translation axis. Relative mapping between user's end effector and selected object.	One (index) or three finger point (index, middle, and thumb) extension. Wrist flexion/extension and ulnar/radial deviation. Forearm pronation. Elbow flexion/extension. Shoulder flexion/extension, adduction/abduction, and lateral/medial rotation.
Clutch			
Decrease grip (open the hand to a flat posture) to engage clutch and enable manipulation. Increase grip (close hand to a natural rest) to disengage clutch and disable manipulation.			

CHAPTER 4. RESULTS

After gestural interaction was integrated with the ASDS using the LEAP Motion Controller, a means of evaluation was needed to determine the efficiency and feasibility of the design. User study evaluation is a valuable tool, but a study requires significant resources and time to execute. Stakeholders looking to quickly understand the performance of a new design cannot afford the time and resources to conduct a full study at each design iteration. An alternate means to evaluate the performance of gestural interaction is needed.

Interaction Evaluation

EVALUATION METHODOLOGY

The Goals, Operators, Methods, and Selections (GOMS) concept is a method of interface evaluation developed by Card et al. [51] and is based on the human processor model, a method to calculate completion time for a given task. The GOMS concept separates user interaction into discrete operations for evaluation: goals (what the user wishes to accomplish), operators (actions that must occur to reach the goal), methods (sequences of operators that must occur to reach the goal), and selection rules (the process of choosing the optimal method). GOMS is a relatively accurate predictor of task completion time, and can estimate interaction efficiency and identify problematic areas of software interaction and workflow, or determine which of two or more different designs is the most effective. It is a simple way to evaluate an in-progress design and highlight areas that need additional consideration. The nature of GOMS evaluation – subdividing user goals into a set of serially executed subtasks – makes it a suitable

method to estimate a CAD user's workflow. A variant of GOMS, called Natural GOMS Language (NGOMSL), is a suitable method to evaluate the implemented gestural interaction in comparison to traditional interaction methods [52]. It differs from the traditional GOMS concept in that it predicts both execution time and learning time. Research has shown that NGOMSL requires less time than user study evaluation while providing valuable insight to highlight shortcomings and guide development effort [53]. A process for the evaluation of traditional interaction and gestural interaction in the ASDS with the NGOMSL concept was clarified by Kieras [54] and the *Handbook of Human-Computer Interaction* [55].

As mentioned, two separate interaction modes are analyzed – interaction with a traditional keyboard and mouse, and the developed gestural interaction with commodity VR hardware. The task analysis evaluates the user goals identified below in Table 3. The possible operations afforded by the implemented gestures found in Table 1 in Chapter 3 determined the user goals included in the analysis.

User goals are decomposed into a series of operators that must occur to

Table 3. User goals included in NGOMSL analysis

User Goals	
Select the *move* tool	Translate an object
Select the *rotate* tool	Rotate an object
Select the *scale* tool	Scale an object

accomplish the goal (e.g., identify the icon, move the mouse over the icon, and click the icon). Sets of operators comprise a method, or a group of actions a user must perform to achieve a goal. Each operator in a method is assigned a primitive operator from the Keystroke-Level Model, a GOMS variant that is utilized within NGOMSL analysis, to

classify the type of action. These primitive operators allow an estimate of the corresponding completion time for each operator, and thus estimate the overall completion time for a method. Kieras et al. provided time values for many common user operators, such as clicking a mouse, and moving a pointer across a screen [56].

Despite the availability of KLM values for traditional operators, no published values exist for gestural interaction. Each undocumented operator corresponding to a gesture received an estimated time value that is based on known execution times for comparable operators. Appendix B shows the KLM values used in the analysis, as well as the source of estimated values for undocumented operators.

In addition to considering execution time, the analysis attempts to estimate the time required of a novice user to learn how to perform the operators within a method. These learning times are estimated with the Pure Method Learning Time (PLMT) technique outlined by [51]. PLMT considers each operator within a method and assigns an estimated *Learning Time Parameter*, a time value that is expected for a first-time user to comprehend verbal instructions of how to perform the operator. These Learning Time Parameters are assigned based on the complexity of the operator, so routine actions that the user understands from previous experience receive a Learning Time Parameter of 0 seconds, general learning situations receive a Learning Time Parameter of 17 seconds, and rigorous procedure training receives a Learning Time Parameter of 30 seconds. One of these three values are assigned to each operator within a method, and when summed provide insight on the complexity of a method. The resulting PLMT value for a method is considered an "up front" cost for the user. Once they have spent the required time to learn the method, the learning time value is not factored into the time needed to accomplish a goal.

Evaluation Results

Tables 4, 5, and 6 respectively outline the execution and learning times for goals performed with the move, rotate, and scale tools. Users have the option to select a tool with the mouse, keyboard shortcuts, or a spatial gesture. However, users cannot manipulate objects through keyboard interaction, so only mouse and gestural input methods appear for object translation, rotation, and scale goals.

The execution and learning time performance of each tool selection method is identical across the three tool selection goals, because the tool selection operators are identical for each tool. A user selecting the *rotate* tool and a user selecting the *scale* tool through the same method (keyboard, mouse, or gesture) will execute the same general operators. A comparison of the execution and learning times for tool selection between selection methods indicates that the keyboard is the most efficient method to select a tool, at an estimated execution time of 1.68 seconds. Tool selection through mouse interaction is estimated at 2.8 seconds and tool selection through gestural interaction is estimated at 2.98 seconds. Further discussion of manipulation execution times appear below, followed by discussion of learning times.

Object Translation Execution Time

The translation goals in Table 4 consider object translation in one, two, and three simultaneous axes. Mouse translation in one or two dimensions is estimated to take 5.5 seconds to execute and 34 seconds to learn. For 3D translation, mouse interaction is estimated at 11.2 seconds and 68 seconds to learn. In contrast, translation in one, two, and three axes with the LEAP Motion Controller takes 2.1 seconds to execute and 68 seconds to learn. In short, the LEAP Motion controller is more efficient for translation

goals, especially in three dimensions, through the position-to-position mapping between the user's hand and the virtual object.

The ASDS interface allows mouse users to translate an object along a 1D axis or a 2D plane with equal simplicity, but the most efficient 3D translation method is a 1D translation and a 2D translation executed in succession. In contrast, object translation with the LEAP Motion Controller can occur in all three axes simultaneously.

Table 4. Execution and learning times for *move* tool goals

Goal	Method	Execution Time (s)	Learning Time (s)
Select the *move* tool	Mouse and Keyboard	2.8	0
	Keybord shortcut	1.68	17
	LEAP Motion Controller	2.98	34
Translate an object in 1 axis	Mouse and Keyboard	5.5	34
	LEAP Motion Controller	2.1	68
Translate an object in 2 axes	Mouse and Keyboard	5.5	34
	LEAP Motion Controller	2.1	68
Translate an object in 3 axes	Mouse and Keyboard	11.2	68
	LEAP Motion Controller	2.1	68

Object rotation execution time

The rotation goals in Table 5 consider object rotation in one, two, and three simultaneous axes. Mouse rotation in one axis is estimated to take 5.5 seconds to execute and 34 seconds to learn. Unlike mouse translation, mouse rotation in two axes is higher than one axis at 11.2 seconds and 34 seconds. Mouse-controlled three-axis rotation is estimated at 16.9 seconds to execute and 34 seconds to learn, while rotation with the LEAP Motion Controller is estimated at 2.1 seconds to execute and 68 seconds to learn for 1, 2, or 3-axis rotation. Overall, the advantages of the LEAP Motion Controller are again apparent for rotation goals.

The ASDS does not allow simultaneous 2D rotation tasks with the mouse, which hinders user interaction. This limitation is common among many CAD and CAE software packages. A mouse user must execute three successive single-axis rotations to rotate an object in all three axes. As seen before in the results of the translation tasks, the LEAP Motion Controller excels because it enables simultaneous object rotation along any combination of axes.

Table 5. Execution and learning times for *rotate* tool goals

Goal	Method	Execution Time (s)	Learning Time (s)
Select the *rotate* tool	Mouse and Keyboard	2.8	0
	Keybord shortcut	1.68	17
	LEAP Motion Controller	2.98	34
Rotate an object in 1 axis	Mouse and Keyboard	5.5	34
	LEAP Motion Controller	2.1	68
Rotate an object in 2 axes	Mouse and Keyboard	11.2	34
	LEAP Motion Controller	2.1	68
Rotate an object in 3 axes	Mouse and Keyboard	16.9	34
	LEAP Motion Controller	2.1	68

Object Scaling Execution Time

The scale goals in Table 6 consider object scaling operations in one, two, and three axes. Mouse interaction allows scaling in one, two, or three simultaneous axes and is estimated at 5.5 seconds to execute and 34 seconds to learn. Gestural scaling with the LEAP Motion Controller varies between DOF, with one and three-axis operations equally efficient at 2.1 seconds to execute and 68 seconds to learn, and two-axis scaling at 4.4 seconds to execute and 68 seconds to learn. Once again, LEAP Motion Controller users experience an advantage over mouse users for execution time.

Unlike translation and rotation operations, the ASDS allows mouse users to scale objects in one, two, or three axes. The implemented LEAP Motion Controller gestures

do not allow this degree of control, so 2D scale tasks necessitate two separate 1D scaling operations. Despite this shortcoming, the relatively inefficient 2D LEAP Motion Controller scaling method is still quicker to execute than the same mouse scaling method.

Table 6. Execution and learning times for *scale* tool goals

Goal	Method	Execution Time (s)	Learning Time (s)
Select the *scale* tool	Mouse and Keyboard	2.8	0
	Keybord shortcut	1.68	17
	LEAP Motion Controller	2.98	34
Scale an object in 1 axis	Mouse and Keyboard	5.5	34
	LEAP Motion Controller	2.1	68
Scale an object in 2 axes	Mouse and Keyboard	5.5	34
	LEAP Motion Controller	4.4	68
Scale an object in 3 axes	Mouse and Keyboard	5.5	34
	LEAP Motion Controller	2.1	68

LEARNING TIME

In general, the learning time for each method tends to increase along with execution time. Methods that utilize novel operators that the user has not encountered before receive a higher value learning parameter than common operators, so methods that include routine actions with the mouse[2] are easier to learn than methods that use spatial gestures.

The Pure Method Learning Time concept considers a method's total learning time to include the learning time of any included sub-method. This approach may overstate the learning time in specific cases, since manipulations in two or three dimensions sometimes require multiple 1D manipulations to achieve the goal. A user is

[2] e.g. clicking and dragging feature across the screen.

repeating a learned action, and PLMT analysis assumes a user can implement it a second or third time in a separate goal without needing the same learning time to understand the sub-method. In scenarios like 1D and 2D object translation with a mouse, the operators needed to achieve the goal are relatively similar, but not identical. A user must identify a unique handle and understand how the mouse interacts in each case. To achieve 3D object translation with a mouse, a user must accomplish a 1D and 2D translation goal. This analysis treats each goal as a separately learned task, so users who want to accomplish 3D translation and have not yet learned 1D and 2D translation are affected by the learning time, but users who have already learned 1D and 2D translation can accomplish the corresponding 3D goal without any training. In short, the learning times listed are for a novice user learning the specified goal for the first time, regardless of whether the method utilizes sub-methods.

Overall, the difference in learning time between methods is not drastically different – the highest learning time difference between is typically around a 34 second advantage for the mouse. The LEAP Motion Controller is more difficult to learn in all cases except 3D translation. Arguably, the initial time investment to learn the gestural interaction methods is worth the saved execution time. Users executing many operations over the course of a workday will save time with gestural interaction. An analysis of the example scenario outlined in the Design Summary of Chapter 3 is shown in Table 7 below.

Table 7. Example execution and learning time analysis

Goal	Mouse		LEAP Motion Controller	
	Execution Time (s)	Learning Time (s)	Execution Time (s)	Learning Time (s)
Import geometry and select component	-	-	-	-
Select move tool	2.8	0	2.98	34
Translate object in 1D	5.5	34	2.1	68
Select rotate tool	2.8	0	2.98	34
Rotate object in 3D	16.9	34	2.1	34*
Select move tool	2.8	0	2.98	0
Translate object in 2D	5.5	34	2.1	0
Select scale tool	2.8	0	2.98	34
Scale object in 3D	5.5	34	2.1	34*
Save file and exit program	-	-	-	-
Totals	44.6	136	20.32	306

* Indicates a task that has a partial learning time reduction due to the previously learned "clutching" gesture

This example highlights the saved learning time for tasks that utilize previously learned methods. 3D mouse rotation necessitates three separate 1D rotations, so the learning value for the goal is simply the time needed to learn a single 1D rotation. Similarly, LEAP Motion Controller users already know the second move shortcut, so they do not need to learn it again. 2D translation is a unique operation with a mouse, but identical to the previous 1D translation for LEAP users, so mouse users must learn a new task while LEAP users can move straight into execution. Most importantly, once

these gestures have been learned, LEAP Motion Controller users experience a significant time savings to the mouse interaction.

The NGOMSL concept was used to evaluate the gestural interaction with the LEAP Motion Controller in comparison to a typical mouse and keyboard for tool selection and object manipulation goals in the ASDS.

Overall, the NGOMSL execution and learning time analysis indicates that gestural interaction is quicker than the mouse for object manipulation in the 3D work environment, but is slightly slower than the mouse and the keyboard shortcuts for tool selection tasks. The LEAP Motion controller initially takes longer to learn than the mouse or keyboard, but the shorter execution time for manipulation goals with gestural interaction justifies its use.

Gestural interaction has the greatest advantage when implemented for rotation tasks because it allows three-axis object rotation with a single action, while mouse users must execute three separate one-axis rotation actions to achieve the same effect.

Gestural interaction is also beneficial for 3D translation tasks, again because mouse users must execute multiple actions to accomplish 3D object translation, however the advantage is less apparent for translation tasks than for rotation tasks because ASDS allows two-axis translation in a single mouse operation.

Object scaling is quicker with gestural interaction than with mouse interaction, but mouse is arguably more efficient for 2D manipulation. The ASDS interface allows mouse users to execute 2D scale manipulations with a single method, but gesture-controlled scale manipulations in two axes require two separate 1D manipulations. Aside from this, 1D and 3D scale manipulations with a mouse or LEAP Motion Controller are equally efficient, and only require a single method.

CHAPTER 5. DISCUSSION

The implemented gestures and selected development platforms are simply a starting point to assess gestural control in 3D work environments through a fast and effective evaluation method. Evaluation with the NGOMSL concept ultimately reveals that the gestures can adequately manipulate objects within software like the ASDS.

A discussion of the stated research issues with respect to the final implementation and results is outlined below, followed by a discussion of technology adoption requirements, shortcomings of this implementation, and future work.

Results and Research Issues

After establishing the need for gestural interaction in 3D work environments within PC software used by engineers and designers, two research issues were specified in Chapter 2: **1)** How can gestural interaction be evaluated without conducting a full user study? **2)** What types of gestures should be used for object manipulation in 3D work environments? Discussion of the results in regard to the research issues, and the case for gestural control in 3D work environments, appear below.

How can gestural interaction be evaluated without a user study?

The NGOMSL concept provides a clear approach to evaluate and compare different interaction devices and information architectures. It allows system designers to quickly identify inefficient operations and estimate execution and learning time for new input modes. NGOMSL also allows comparison of two or more gesture sets, allowing designers to quickly find the most efficient gestures for software operations. The resources saved by using NGOMSL analysis instead of a traditional user study can

be reinvested into the interaction design, improving the final solution and expediting the overall development process.

What types of gestures should be used?

The specific gestures integrated with software in this implementation were selected based on previous interaction research, the limitations of the LEAP Motion Controller, and the specific needs of the ASDS software. Existing research shows that gestural control should utilize small muscle movements of a user's hand rather than large arm motions. Implemented gestures focused on movements from the user's elbow, wrist, and fingers to reduce user fatigue and take advantage of a user's fine motor control. Informal testing found that these gesture types were readily detectable by the LEAP Motion Controller, and the system rarely detected a gesture incorrectly. The performance of implemented gestures suggests that developers can create a gesture set suitable for their needs from individual members of the gesture taxonomy in Table 1. The taxonomy supports many unique interaction gestures, so the number of gestures available for a given application is limited only by the user's memory.

Ensuring Adoption of Commodity VR Interaction

Many novel interaction technologies have been developed and marketed to industry professionals and consumers, but most computer interaction still occurs with a mouse and keyboard. High quality and capable commodity VR interaction devices can now be affordably manufactured, but device manufacturers must address users' needs in order to sell devices. To ensure the adoption of commodity VR devices by the engineers

and designers who will benefit from gestural interaction, several conditions must be met [57]:

1. Commodity VR devices must provide some advantage to a user, possibly an economic advantage due to efficient use of time or facilitation of the output of a higher quality product than can be designed with traditional tools.
2. These commodity VR devices must be compatible with existing software and the physical work area.
3. User interaction with new commodity VR input devices must match the level of comfort and safety found with a traditional mouse and keyboard.
4. The commodity VR devices cannot be difficult to operate.

Satisfaction of these conditions ensures control and manipulation of 3D work environments has a high probability of adoption.

The evaluation in Chapter 4 validates conditions one and three. A commodity VR device and gestural control provides a benefit to users in the form of improved execution time over mouse interaction, and the gestural interaction is not significantly difficult to learn. Additionally, the work outlined in Chapter 3 provides a strategy for rapid development and evaluation of gestural interaction.

The responsibility for the second condition, compatibility with existing software, rests on the software manufacturers. The businesses that use engineering design software cannot implement commodity VR device interaction in software without access to the software source code. Software developers must partner with device manufacturers to integrate a commodity VR device and gestural control within an application.

The third condition, concerning ergonomics, was addressed in the development of the gestural taxonomy. Developers must use gestures that do not fatigue or strain users, and the gestures in the taxonomy satisfy this requirement. Additional research of gestural control ergonomics will benefit a future gestural taxonomy.

The fourth condition is the responsibility of the device manufacturer and the software developer. Most novice users would have little trouble connecting a new mouse or keyboard to a PC, and a new interaction device must match this level of accessibility. Additionally, the interaction gestures must be designed with suggestions from Chapter 3. Finally, since gestural interaction does not provide the tactile feedback provided by traditional input devices, the software interface should augment interaction with feedback in another sense, so the user understands the interaction status at all times.

Design Limitations

The work outlined in this thesis is not without several shortcomings and limitations:

1. The NGOMSL analysis does not evaluate device ergonomics. Although the gestures included in the gestural taxonomy were developed based on existing research with user ergonomics in mind, user testing is needed to fully understand the impact of gestural interaction on users.
2. The implemented gestures were chosen with consideration for the LEAP Motion Controller's limitations, the dimensions of a typical office work area, and the ergonomic and mental burden placed on the user. Future commodity VR devices may track user's hands differently or with more accuracy, facilitating gestural

interaction for goals in ways that are quicker or easier to learn. NGOMSL analysis can be used to determine if a new interaction device is quicker and easier to learn than the LEAP Motion Controller.

3. The NGOMSL analysis does not consider different decisions that a user could make while working to achieve a goal, instead, it is assumed that users will choose the most effective means to achieve a goal. Object manipulation goals in two or three axes sometimes require several sub-methods, and in a typical use-case a user can decide which combination of sub-methods to use (e.g. 3D translation with a mouse can be accomplished with one 1D translation and one 2D translation, or three 1D translations). Ultimately, this assumption means that the results only show the most efficient execution and learning time for each input device.

Future Work

User testing should be conducted to assess the validity of the implemented gestural interaction with 3D work environments through a commodity VR device. The assessment outlined in Chapters 3 and 4 informs us that gestural control can perform some tasks quicker than other input devices, but significant knowledge gaps remain. Specifically, user testing to evaluate the ergonomics should be a primary focus. The device usage and implemented gestures were chosen with ergonomics and comfort in mind, but the NGOMSL analysis does not validate this. Additionally, future user studies should consider a quantitative evaluation of the estimated execution and learning times to confirm the NGOMSL analysis results.

Successful integration of commodity VR interaction devices for software control can only occur with an understanding of best practices for gestural interaction and support from the software owners. The NGOMSL evaluation determined that gestural control performs well for general object manipulation tasks, but is not ideal for tool selection tasks. The performance of tool selection gestures reveals that gestures are not an ideal replacement for a simple button on a toolbar, suggesting that software developers looking to implement gestural control with commodity VR devices should focus on user tasks that necessitate interaction in three or more axes. Despite this finding, the integration and evaluation process outlined in this work can also be applied to software without 3D work environments. Designers should consider which software operations may benefit from gestural control, and then validate the choices through NGOMSL evaluation.

REFERENCES

[1] P. A. David, "Clio and the Economics of QWERTY," *The American economic review*, pp. 332–337, 1985.

[2] B. A. Myers, "A brief history of human-computer interaction technology," *interactions*, vol. 5, no. 2, pp. 44–54, 1998.

[3] D. A. Norman and S. W. Draper, *User Centered System Design; New Perspectives on Human-Computer Interaction*. 1986, p. 544.

[4] K. Henriksen, J. Sporring, and K. Hornbæk, "Virtual Trackballs Revisited," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, pp. 206–216, 2004.

[5] C. Hand, "A Survey of 3D Interaction Techniques," *Computer Graphics Forum*, vol. 16, no. 5, pp. 269–281, 1997.

[6] S. Boyer, "A Virtual Failure: Evaluating the Success of Nintendo's Virtual Boy," *The Velvet Light Trap*, vol. 64, pp. 23 – 33, 2009.

[7] Nintendo, "Nintendo." [Online]. Available: http://www.nintendo.com. [Accessed: 10-Jun-2014].

[8] LEAP Motion, "LEAP Motion Controller." [Online]. Available: http://www.leapmotion.com. [Accessed: 10-Jun-2014].

[9] Microsoft, "Kinect for Windows." [Online]. Available: http://www.microsoft.com/en-us/kinectforwindows/. [Accessed: 10-Jun-2014].

[10] Sixense, "Razer Hydra." [Online]. Available: http://sixense.com/razerhydra. [Accessed: 10-Jun-2014].

[11] L. Garber, "Gestural Technology: Moving Interfaces in a New Direction [Technology News]," *Computer*, vol. 46, no. 10, IEEE, pp. 22–25, 2013.

[12] B. Miller, "Touchscreen Technology: A Total Hit!," 2012. [Online]. Available: http://www.adtouch.com/touchscreen-technology/.

[13] A. Smith, "Smartphone ownership – 2013 update," Pew Research Center's Internet & American Life Project, Washington, D.C., 2013.

[14] D. E. Kieras, D. Meyer, and J. Ballas, "Towards demystification of direct manipulation: Cognitive modeling charts the gulf of execution," in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 2001, no. 3, pp. 128–135.

[15] Microsoft, "Xbox 360." [Online]. Available: http://www.xbox.com/en-US/xbox-360?xr=shellnav. [Accessed: 10-Jun-2014].

[16] Sony, "Playstation 3." [Online]. Available: http://www.playstation.com/en-us/explore/ps3/. [Accessed: 10-Jun-2014].

[17] SpaceX, "The Future of Design," 2013. [Online]. Available: https://www.youtube.com/watch?v=xNqs_S-zEBY. [Accessed: 11-Jun-2014].

[18] R. J. K. Jacob, "Human-computer interaction: input devices," *ACM Computing Surveys (CSUR)*, vol. 28, no. 1, pp. 177–179, Mar. 1996.

[19] W. Buxton, "There's more to interaction than meets the eye: Some issues in manual input," *User centered system design: New perspectives on human-computer interaction*, vol. 319, p. 337, 1986.

[20] R. J. Beaton, R. J. Dehoff, N. Weiman, and P. W. Hildebrandt, "An Evaluation of Input Devices for 3-D Computer Display Workstations," *SPIE*, vol. 761, 1987.

[21] J. P. Djajadiningrat, C. J. Overbeeke, and G. J. F. Smets, "The importance of the number of degrees of freedom for rotation of objects," *Behaviour & Information Technology*, vol. 16, no. 6, pp. 337–347, 1997.

[22] P. E. Jones, "Three-dimensional input device with six degrees of freedom," vol. 9, pp. 717–729, 1999.

[23] B. Frohlich and J. Plate, "The cubic mouse: a new device for three-dimensional input," *Conference on Human Factors in Computing Systems: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, vol. 1, no. 06, pp. 526–531, 2000.

[24] R. Pausch, "Virtual reality on five dollars a day," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1991, pp. 265–270.

[25] A. Basu, C. Saupe, E. Refour, A. Raij, and K. Johnsen, "Immersive 3DUI on one dollar a day," in *2012 IEEE Symposium on 3D User Interfaces (3DUI)*, 2012, pp. 97–100.

[26] T. A. Erlemeier, "The Development of a Virtual Reality Based CAD System For Design Review," Iowa State University, Ames, Iowa, 2006.

[27] J. C. Lee, "Johnny Chung Lee > Projects > Wii," 2008. [Online]. Available: johnnylee.net/projects/wii/.

[28] J. Lin, H. Nishino, T. Kagawa, and K. Utsumiya, "A method of two-handed gesture interactions with applications based on commodity devices," *Computers and Mathematics with Applications*, vol. 63, no. 2, pp. 448–457, Jan. 2012.

[29] R. A. Pavlik and J. M. Vance, "A Modular Implementation of Wii Remote Head Tracking for Virtual Reality," in *ASME 2010 World Conference on Innovative Virtual Reality*, 2010, pp. 351–359.

[30] W. Zhu, A. M. Vader, A. Chadda, M. C. Leu, X. F. Liu, and J. B. Vance, "Wii remote–based low-cost motion capture for automated assembly simulation," *Virtual Reality*, vol. 17, no. 2, pp. 125–136, Dec. 2011.

[31] C. Ardito, P. Buono, M. F. Costabile, R. Lanzilotti, and a. L. Simeone, "Comparing low cost input devices for interacting with 3D Virtual Environments," *2009 2nd Conference on Human System Interactions*, 2009.

[32] L. Gallo, a. Minutolo, and G. De Pietro, "A user interface for VR-ready 3D medical imaging by off-the-shelf input devices," *Computers in Biology and Medicine*, vol. 40, no. 3, pp. 350–358, Mar. 2010.

[33] D. Dave, A. Chowriappa, and T. Kesavadas, "Gesture Interface for 3D CAD Modeling using Kinect," *Computer-Aided Design & Applications*, vol. 9, pp. 1–7, 2012.

[34] L. Gallo, A. P. Placitelli, and M. Ciampi, "Controller-free exploration of medical image data: Experiencing the Kinect," in *Proceedings - IEEE Symposium on Computer-Based Medical Systems*, 2011.

[35] E. S. Santos, E. A. Lamounier, and A. Cardoso, "Interaction in Augmented Reality Environments Using Kinect," *2011 XIII Symposium on Virtual Reality*, pp. 112–121, 2011.

[36] T. Dutta, "Evaluation of the Kinect^{TM} sensor for 3-D kinematic measurement in the workplace," *Applied ergonomics*, vol. 43, no. 4, pp. 645–649, 2012.

[37] R. Francese, I. Passero, and G. Tortora, "Wiimote and Kinect: Gestural User Interfaces add a Natural third dimension to HCI.," *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pp. 116–123, 2012.

[38] B. J. Juhnke, "Evaluating the Microsoft Kinect compared to the mouse as an effective interaction device for medical imaging manipulations," Iowa State University, Ames, Iowa, 2013.

[39] N. Villaroman, D. Rowe, and B. Swan, "Teaching natural user interaction using OpenNI and the Microsoft Kinect sensor," in *Proceedings of the 2011 conference on Information technology education*, 2011, pp. 227–232.

[40] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler, "Analysis of the accuracy and robustness of the Leap Motion Controller," *Sensors (Switzerland)*, vol. 13, no. 5, pp. 6380–6393, 2013.

[41] S. Mauser and O. Burgert, "Touch-free, gesture-based control of medical devices and software based on the leap motion controller.," *Studies in health technology and informatics*, vol. 196, pp. 265–70, 2014.

[42] C. Noon, R. Zhang, E. Winer, J. Oliver, B. Gilmore, and J. Duncan, "A system for rapid creation and assessment of conceptual large vehicle designs using immersive virtual reality," *Computers in Industry*, vol. 63, no. 5, pp. 500–512, 2012.

[43] LEAP Motion, "LEAP Motion Developer Portal," 2014. [Online]. Available: https://developer.leapmotion.com/. [Accessed: 10-Jun-2014].

[44] J. Han and N. Gold, "Lessons Learned in Exploring the Leap Motion(TM) Sensor for Gesture-based Instrument Design," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2014, pp. 371–374.

[45] K. Kin, T. Miller, B. Bollensdorff, T. DeRose, B. Hartmann, and M. Agrawala, "Eden: a professional multitouch tool for constructing virtual organic environments," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2011, pp. 1343–1352.

[46] M. Nielsen, M. Störring, T. B. Moeslund, and E. Granum, "A procedure for developing intuitive and ergonomic gesture interfaces for HCI," in *Gesture-Based Communication in Human-Computer Interaction*, Springer, 2004, pp. 409–420.

[47] T. Carmody, "Why 'Gorilla Arm Syndrome'Rules Out Multitouch Notebook Displays," *Wired, Oct*, vol. 10, 2010.

[48] K. Hinckley and R. Pausch, "A survey of design issues in spatial input," *Proceedings of the 7th …*, pp. 213–222, 1994.

[49] S. Zhai, P. Milgram, and W. Buxton, "The influence of muscle groups on performance of multiple degree-of-freedom input," *Proceedings of the SIGCHI conference on Human factors in computing systems common ground - CHI '96*, pp. 308–315, 1996.

[50] J. Payne, P. Keir, J. Elgoyhen, M. McLundie, M. Naef, M. Horner, and P. Anderson, "Gameplay issues in the design of spatial 3D gestures for video games.," in *CHI'06 extended abstracts on Human factors in computing systems*, 2006, pp. 1217–1222.

[51] S. K. Card, A. Newell, and T. P. Moran, *The Psychology of Human-Computer Interaction*. Hillsdale, New Jersey, USA: L. Erlbaum Associates Inc, 1983.

[52] B. E. John and D. E. Kieras, "Using GOMS for user interface design and evaluation: Which technique?," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 3, no. 4, pp. 287–319, 1996.

[53] R. J. Gong, "Validating and refining the GOMS model methodology for software user interface design and evaluation," University of Michigan, 1993.

[54] D. Kieras, "A guide to GOMS model usability evaluation using GOMSL and GLEAN3," *University of Michigan*, no. 313, 1999.

[55] M. G. Helander, T. K. Landauer, and P. V Prabhu, *Handbook of human-computer interaction*. Elsevier, 1997.

[56] D. Kieras, "Using the keystroke-level model to estimate execution times," *University of Michigan*. Ann Arbor, Michigan, 2001.

[57] E. M. Rogers, *Diffusion of Innovations*, 5th ed., vol. 27. New York, New York, USA: Free Press, 2003, p. 551.

APPENDIX A. THE ASDS TOOLBAR

Tool Name	*Use*	*Category*
Select	Select one or more components to manipulate or assess.	Support for Concept Manipulation and Assessment
Move	Translate one or more selected components in 3D space.	Concept Manipulation
Rotate	Rotate one or more selected components in 3D space.	Concept Manipulation
Scale	Increase or reduce the size of one or more selected components in 1, 2, or 3 dimensions of space.	Concept Manipulation
Measure	Measure a bounding box around one or more selected components, or as a point-to-point virtual tape measure.	Concept Assessment
Assess	Calculate the center of gravity, wheel loading, or tipping angle of a concept.	Concept Assessment

APPENDIX B. KLM NOTATION AND VALUES

Operator	*Symbol*	*Description*	*Time Value (seconds)*
Keystroke	K	Pressing a key or button on the keyboard	0.28
Point with mouse to target on display	P	The action of moving the mouse to point the cursor to a desired place on the screen	1.1
Press or release mouse button	B	A rapid click or release, used for click and drag	0.1
Click mouse button	BB	A rapid click and release, used for clicking buttons or icons	0.2
Mental act of routine thinking or perception	M	Routine tasks like finding something on screen or recalling a tool name	1.2
Manipulation gesture	Gm	Execute an object manipulation gesture. Time value is analogous to operator P.	1.1
Engage or disengage clutch gesture	Gc	Execute the engage or disengage clutch gesture. Time value is analogous to operator B.	0.1
Shortcut gesture	Gs	Execute a tool selection shortcut gesture. Time value is analogous to operator K.	0.28

APPENDIX C. FORMULAS

The following formulas are used to calculate a method's total learning and execution time.

Pure Method Learning Time =

*Learning Time Parameter * (No. of learned NGOMSL statements)*

Execution Time =

Σ (KLM Time Values for each operator in method)

*+ (No. of operators in method * 0.1)*

APPENDIX D. FULL NGOMSL ANALYSIS RESULTS

GOAL: TOOL SELECTION

Method: **Mouse and Keyboard**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Locate icon for tool on screen	M	1.2	0
Move cursor to tool icon location	P	1.1	0
Click mouse button and release	BB	0.2	0
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		2.8	0
Method: **Keyboard Shortcut**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Recall keyboard shortcut	M	1.2	17
Execute keyboard shortcut	K	0.28	0
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		1.68	17
Method: **LEAP Motion Controller**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Recall shortcut gesture	M	1.2	17
Perform shortcut gesture	Gs	0.28	17
Visually confirm tool is selected	M	1.2	0
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		2.98	34

SUB-GOAL: IDENTIFY OBJECT MANIPULATOR

MOVE: ARROW OR PLANE. ROTATE: VIRTUAL TRACKBALL. SCALE: MANIPULATOR HANDLE

Method: **Mouse and Keyboard**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Locate manipulator on screen	M	1.2	17
Within manipulator: Identify feature corresponding to desired Goal (object translation, rotation, or scale in 1 or more axes).	M	1.2	0
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		2.6	17

GOAL: TRANSLATE OBJECT IN 1 AXIS

Method: **Mouse and Keyboard**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Accomplish Goal: Identify manipulator arrow	M	2.6	17
Point mouse to arrow on manipulator	P	1.1	0
Press and hold mouse button down	B	0.1	0
Drag object to new location	P	1.1	17
Release mouse button	B	0.1	0
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		5.5	34
Method: **LEAP Motion Controller**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Move hand into device view volume	H	0.4	17
Disengage clutch	Gc	0.1	17
Perform spatial gesture	Gm	1.1	17
Engage clutch	Gc	0.1	17
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		2.1	68

GOAL: TRANSLATE OBJECT IN 2 AXES

Method: **Mouse and Keyboard**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Accomplish Goal: Identify manipulator plane	M	2.6	17
Point mouse to plane on manipulator	P	1.1	0
Press and hold mouse button down	B	0.1	0
Drag object to new location	P	1.1	17
Release mouse button	B	0.1	0
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		5.5	34
Method: **LEAP Motion Controller**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Move hand into device view volume	H	0.4	17
Disengage clutch	Gc	0.1	17
Perform spatial gesture	Gm	1.1	17
Engage clutch	Gc	0.1	17
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		2.1	68

GOAL: TRANSLATE OBJECT IN 3 AXES

Method: **Mouse and Keyboard**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Accomplish Goal: translate an object in 1 axis	-	5.5	34
Accomplish Goal: translate an object in 2 axes	-	5.5	34
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		11.2	68
Method: **LEAP Motion Controller**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Move hand into device view volume	H	0.4	17
Disengage clutch	Gc	0.1	17
Perform spatial gesture	Gm	1.1	17
Engage clutch	Gc	0.1	17
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		2.1	68

GOAL: ROTATE OBJECT IN 1 AXIS

Method: **Mouse and Keyboard**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Accomplish Goal: Identify virtual trackball	M	2.6	17
Point mouse to rotation axis on virtual trackball	P	1.1	0
Press and hold mouse button down	B	0.1	0
Drag mouse to rotate object	P	1.1	17
Release mouse button	B	0.1	0
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		5.5	34
Method: **LEAP Motion Controller**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Move hand into device view volume	H	0.4	17
Disengage clutch	Gc	0.1	17
Perform spatial gesture	Gm	1.1	17
Engage clutch	Gc	0.1	17
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		2.1	68

GOAL: ROTATE OBJECT IN 2 AXES

Method: **Mouse and Keyboard**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Accomplish Goal: Rotate object in 1 axis	-	5.5	34
Accomplish Goal: Rotate object in 1 axis	-	5.5	0
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		11.2	34
Method: **LEAP Motion Controller**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Move hand into device view volume	H	0.4	17
Disengage clutch	Gc	0.1	17
Perform spatial gesture	Gm	1.1	17
Engage clutch	Gc	0.1	17
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		2.1	68

GOAL: ROTATE OBJECT IN 3 AXES

Method: **Mouse and Keyboard**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Accomplish Goal: Rotate object in 1 axis	-	5.5	34
Accomplish Goal: Rotate object in 2 axes	-	11.2	0*
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		16.9	34
Method: **LEAP Motion Controller**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Move hand into device view volume	H	0.4	17
Disengage clutch	Gc	0.1	17
Perform spatial gesture	Gm	1.1	17
Engage clutch	Gc	0.1	17
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		2.1	68

* = Learning time value for 2D rotation is 0 because 3D rotation requires three separate 1D rotations. 2D rotation is two separate 1D rotations.

GOAL: SCALE OBJECT IN 1 AXIS

Method: **Mouse and Keyboard**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Accomplish Goal: Identify desired manipulator handle	M	2.6	17
Point mouse to manipulator handle corresponding to scale plane	P	1.1	0
Press and hold mouse button down	B	0.1	0
Drag mouse to scale object	P	1.1	17
Release mouse button	B	0.1	0
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		5.5	34
Method: **LEAP Motion Controller**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Move hand into device view volume	H	0.4	17
Disengage clutch	Gc	0.1	17
Perform spatial gesture	Gm	1.1	17
Engage clutch	Gc	0.1	17
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		2.1	68

GOAL: SCALE OBJECT IN 2 AXES

Method: **Mouse and Keyboard**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Accomplish Goal: Identify desired manipulator handle	M	2.6	17
Point mouse to manipulator handle corresponding to scale planes	P	1.1	0
Press and hold mouse button down	B	0.1	0
Drag mouse to scale object	P	1.1	17
Release mouse button	B	0.1	0
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		5.5	34
Method: **LEAP Motion Controller**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Accomplish Goal: Scale object in 1 axis	-	2.1	68
Accomplish Goal: Scale object in 1 axis	-	2.1	0
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		4.4	68

GOAL: SCALE OBJECT IN 3 AXES

Method: **Mouse and Keyboard**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Accomplish Goal: Identify desired manipulator handle	M	2.6	17
Point mouse to manipulator handle corresponding to 3D scale	P	1.1	0
Press and hold mouse button down	B	0.1	0
Drag mouse to scale object	P	1.1	17
Release mouse button	B	0.1	0
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		5.5	34
Method: **LEAP Motion Controller**	*KLM Notation*	*Operator Time (s)*	*Learning Time (s)*
Move hand into device view volume	H	0.4	17
Disengage clutch	Gc	0.1	17
Perform spatial gesture	Gm	1.1	17
Engage clutch	Gc	0.1	17
Return with Goal accomplished	-	*Total Execution Time (s)*	*Total Learning Time (s)*
		2.1	68

ACKNOWLEDGEMENTS

This work would not have been possible without the support of my major professor Eliot Winer as well as my committee, and also my colleagues, friends, mentors, and everyone else at the Virtual Reality Applications Center. Thank you all for the guidance, and for helping me find my vocation.

Special thanks to Trevor Richardson, for helping me dig through the ASDS source code and successfully make the LEAP Motion Controller talk with Qt widgets.