Feasibility of Supporting Pointing on Large Wall Displays Using Off-the-Shelf Consumer-Grade Tracking Equipment

by

Orkhan Muradov

B.Sc. Computer Science, University of Washington, 2011

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate and Postdoctoral Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

October 2013

Abstract

Interactions between instructors and large projected displays in classrooms often lack cost-effective and easy-to-use solutions. Instead, instructors mostly use either a standard laser pointer, or a computer mouse or touchscreen on a laptop or tablet computer to interact with the content of their presentation slides on large screens. Our research investigates the possibility of using relatively inexpensive, consumer-grade tracking devices that can be easily installed in classrooms and inertial sensors that can be held in an instructor's hand to support this type of interaction. Our goal was to provide more functionality than a simple laser pointer. We compared the accuracy and ease of pointing performance using our new system to a high-end research system and to a computer mouse. We used a Fitts's law paradigm to experimentally evaluate pointing performance and accuracy. We found that our low-cost system is as accurate and as fast as the high-end research system and, somewhat surprisingly, also as accurate as a computer mouse. Because of the large size of the displays in classrooms and the small tracking volumes of most consumer-grade tracking systems, we investigated the possibility of using multiple off-the-shelf hardware units connected with each other to maximize the coverage area at the front of a classroom. We present a possible design that could be scaled to enough devices connected together to cover the front area of a classroom of almost any size. Our investigation was not conclusive. It requires additional research to prove the feasibility of our system in real classroom environments.

Preface

All research in this thesis was conducted under the supervision of Dr. Kellogg S. Booth. The Behavioral Research Ethics Board, UBC BREB Number H11-01756, provided the Ethics approval for experimentation with human subjects.

I am the primary contributor to all work reported in this thesis. The experimental design for Chapter 4 and Chapter 5 is based on similar experiments performed by Dr. Garth Shoemaker and Vasanth Rajendran. I collaborated with Junhao Shi on the work described in Chapter 4 and 5.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I want to thank my supervisor Dr. Kellogg S. Booth for his kind support, guidance and very insightful discussions in x521 room. I am fascinated with his never-ending dedication to his students at all times. It always surprised me how Dr. Booth manages to always be in touch with his students through e-mail, even on a busy schedule. I could not have finished this thesis without detailed feedback, help and assistance from Dr. Booth.

The research reported in this thesis was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under a Discovery Grant, and by the Network of Centres of Excellence Program through the Graphics, Animation and New Media NCE (GRAND). Facilities provided by the Canada Foundation for Innovation (CFI) and The University of British Columbia through the Department of Computer Science and the Institute for Computing, Information and Cognitive Systems (ICICS) were used in the research. The Department of Computer Science at UBC provided technical and administrative support during the research.

I want to thank my lab mate Junhao Shi for the huge help in my research work and all the great days we have spent together in the lab. I want to thank my UBC friends for always being here for me and helping me with the experiments.

I am very grateful to my parents, who made me who I am and their ever-lasting inspiration. I am most grateful to my wife Mariya, for all the support and belief in me.

All of this support is gratefully acknowledged.

Chapter 1: Introduction

In research reported in this thesis we investigate the question of whether instructors can use off-the-shelf consumer-grade equipment to accurately point and interact with large wall displays in a more natural manner than using a computer mouse attached to a laptop or classroom computer. This question is important because modern classrooms are often equipped with computers, large screen displays and projectors, but instructors still tend to use a laser pointer to draw the audience's attention to parts of the screen content, without being able to interact directly with the content using the pointing device. This prevents instructors from visually highlighting certain parts of the content. Instead, they have to rely on a small laser dot on the screen. In addition, presenters have to carry a small remote controller to navigate through presentation slides unless they are willing to be "tethered" to their laptop or the classroom computer. Some presenters even prefer pointing on the screen with a physical pointer or an index finger as they used to do with blackboards, but this is even less suitable for classrooms with large wall displays, because people cannot physically reach to all parts of the display, even with a large physical pointer. The research reported here attempts to bridge the gap between traditional classroom interaction that was suitable for blackboards and traditional wall displays such as maps or charts, and new computer-based interactions using mice and other devices that have been very successful for single-user interaction on personal computers but have been found to be less desirable for interactions in classrooms with large displays when the instructor wishes to roam freely at the front of the classroom while engaging the audience with material presented on one or more large display screens that serve the purpose formerly filled by blackboards, maps, and charts.

1.1 Outline of the Problem

To interact with a large screen display people often use a computer mouse, which requires a flat surface or a desk to track on. This limits the presenter to stay within an arm's reach of the computer desk and results in a lowered mobility for the speaker. Olsen and Nielsen (2001) argued that using a mouse during large screen interaction is not a perfect solution because when the presenter is facing the audience, he/she needs to turn around every time to see where the pointer is on the large display while controlling the mouse on the flat surface. The speaker needs to spend some time considering how his/her mouse movements will be mapped on the large wall display if he/she decides to face the audience while controlling the mouse. Computer displays that mirror the large screen content help mitigate this issue but still require speakers to look at the computer screen instead of focusing on the audience. On some occasions, instructors prefer to use a touch screen computer, which allows drawing on top of the projected content using a stylus pen. Such computers provide high precision but are not feasible for large screen displays (Sears & Shneiderman, 1992) unless they too are quite large. These computers usually require instructors to focus on a small touchscreen computer display, which in turn results in loosing eye contact with the audience even more so than with a regular computer, although it does solve the problem of having to turn around to face the large screen and for small touchscreens the instructor can wander around at the front of the room if there is wireless communication to the display. Some modern classrooms are equipped with more expensive and advanced equipment that can support interaction between the speaker and the large screen display. These systems will be discussed later in this thesis, but our belief is that they are still relatively expensive and thus not likely to find widespread use in the near future.

1.2 Outline of Our Research

We developed and evaluated an inexpensive prototype system that allows a presenter to interact with the content on large wall display for any size classroom using a $99 camera-based tracker and a $40 gaming motion controller. This interaction is performed by pointing at displayed slides using the gaming controller while being in the range of the camera tracker. We refer to this as mid-air pointing, interaction with a large screen display at a distance, using either low-cost or high-end advanced equipment that does not require a flat surface to track on. The goal of our research was to determine if our inexpensive system is as accurate and easy to use as the more advanced and costlier equipment that has been used in research labs for this purpose. In addition, we explored how mid-air pointing, using both expensive and inexpensive methods, compares to traditional computer mouse-based pointing. We conducted a lab experiment that simulated a classroom situation and tested whether pointing performance (time required to point at a target) and accuracy was comparable between three conditions: our inexpensive solution, an expensive commercial tracking system, and a computer mouse used in the traditional way on a desktop. We found that pointing performance (speed) and accuracy of pointing using our inexpensive system was as good as with the expensive system used in the experiment. We also found that the pointing accuracy for both the expensive and inexpensive equipment was as good as the accuracy with a mouse, but that pointing performance (speed) was not as good.

Many expensive tracking systems allow only a small portion of the room to be tracked because of their limited tracking volume. Our work also explored the possibility of using multiple camera trackers with one gaming motion controller to cover a large classroom area, to allow presenters to freely walk around the front of the room and point on a large wall screen display. We performed a pilot experiment in a real classroom to investigate this possibility and

determined that cameras need to be placed 2 meters apart to accurately track the user. We make recommendations for future studies to deploy our system with multiple tracking devices in a real classroom.

In this remainder of this chapter, we give a further introduction to the core problem statement for the research, followed by a summary of our research contributions and an outline of the rest of the thesis.

1.3 Problem Statement and Approach

The main research problem addressed in this thesis was to understand if an instructor can easily interact with a large screen display without using a laser pointer or a computer mouse or an expensive tracking system such as those used in virtual reality systems or motion capture systems that are used in the game development and special effects industries. To address this problem we needed to study the performance time and accuracy of pointing using a mid-air pointing device. Our interest was to understand if we can build a reliable system from low-cost components and if we can extend this system to larger classroom sizes. In particular, we focused on comparing a prototype low-cost system that we implemented to an advanced mid-air pointing device and a traditional computer mouse.

1.3.1 Research Questions

The high-level goal of our research is to understand how suitable our prototype system is for classroom usage and to explore interaction approaches with large wall displays that can be supported by our system.

A. Primary Questions

a. Does mid-air pointing perform as fast and as accurate as traditional pointing with a computer mouse?

b. Is low-cost tracking equipment as fast and as accurate as high-end tracking equipment during mid-air pointing?

c. How suitable is the low-cost tracking equipment in a real classroom setting in terms of accuracy and performance?

d. What is the optimal placement of low-cost equipment in a classroom setting that maximizes coverage area for pointing?

B. Secondary Questions

a. Do one-part models of pointing (Fitts and Shannon-Fitts) accurately predict pointing performance for mice, and for high-end and low-cost pointing devices?

b. Do two-part models of pointing (Welford and Shannon-Welford Laws) better predict pointing performance for mice, and for high-end and low-cost pointing devices?

1.4 Summary of Research Contributions

We summarize here the primary contributions described in our thesis. These are discussed in more detail in Chapter 7.

1. We built different systems using off-the-shelf consumer grade equipment, outlined advantages and disadvantages of each of them, and suggested areas for future research. One system used an Xbox Kinect from Microsoft and a Nintendo Wii controller, another used a Kinect and a PlayStation Move controller.

2. We experimentally evaluated the performance and accuracy of pointing using low-cost off-the-shelf equipment and compared it to a conventional computer mouse and to a high-end tracking system. Based on the results of the experiment we concluded that two-part models (such as Welford's Law) for pointing performance were not substantially better than that one-part models (such as Fitts's Law) at predicting pointing performance for both low-cost and high-end tracking devices and for a computer mouse. The high-end tracker was a Vicon system that uses infra-red cameras.
3. We performed an informal pilot study to determine how our system would work in a real classroom setting in terms of accuracy and pointing performance. Preliminary results suggest that practical pointing performance of at least 95% accuracy could be attained. In addition, we investigated the possibility of using multiple camera-based trackers in a large classroom to maximize tracked area during pointing. We present results and indicate ways for future in classroom studies using multiple camera trackers.

1.5 Overview of the Thesis

In Chapter 2, we discuss related work and the variety of devices that are used for large screen interaction, especially in classrooms. In Chapter 3 we describe variations of our implementations and their hardware and software components. In Chapter 4 we present an experiment that used both one-part and two-part predictive models to evaluate pointing performance for a mouse, a high-end system, and our low-cost system. For this experiment we used a hybrid version of the low-cost system that partially depended on the high-end system in order to calibrate the accuracy of the low-end system. In Chapter 5 we describe a pilot study that used a non-hybrid version of the low-cost equipment and compared it to the results obtained in the experiment described in Chapter 4. We also further investigated whether the position of the mouse surface in the room,

relative to the large display, had affected accuracy and performance for pointing in that earlier experiment. In Chapter 6 we present results from a test that explored the suitability of our system in real classroom settings in terms of accuracy and pointing performance. We then investigated the possibility of multiple camera-based trackers being connected to each other to maximize the tracking area at the front of a classroom. In Chapter 7 we summarize our findings and outline future research opportunities in this area.

Chapter 2: Background, Related Work, and Basic Concepts

Related work lies in the areas of mid-air pointing techniques on large displays and pointing performance research for traditional displays. We explore both of them in this section. Our work is based on previous investigations of "ray pointing" as a technique for pointing at large displays, which we summarize first. We then look at the various types of tracking systems that have been used to implement ray pointing, and then the formal models that have been developed to evaluate performance using the systems.

2.1 Ray Pointing Techniques at Large Screen Displays

Different ray-pointing techniques have been explored over time and advocated as a way for users to interact with large displays at a distance. Bolt (1980), Olsen and Nielsen (2001) characterize ray pointing as a natural interaction with large displays. In addition, unlike mouse-based pointing, it does not require any physical surface to track on (Teather & Stuerzlinger, 2008; Oh & Stuerzlinger, 2002), and it allows users to interact with the screen from a distance (Parker, Mandryk, & Inkpen, 2005; Voida, Podlaseck, Kjeldsen, & Pinhanez, 2005). Ray pointing allows multiple people to share large-screen real estate without getting in each other's way (Nacenta, Pinelle, Stuckel, & Gutwin, 2007; Vogt, Wong, Fels, & Cavens, 2003) and is more intuitive, because it consists of physical gestures similar to those performed in everyday tasks (Voida et al., 2005). As a result, ray pointing on large horizontal and vertical displays is very commonly used in research systems and commercial applications (Davis & Chen, 2002; Parker et al., 2005; Vogel & Balakrishnan, 2005). Ray pointing is becoming more and more popular in the gaming industry when using controllers such as Nintendo Wii, PlayStation Move, etc. (Jota, Nacenta, Jorge, Carpendale, & Greenberg, 2010). We review and compare side-by-side four different ray-

pointing variants: laser, arrow, image-plane and fixed-origin pointing. In all four techniques there is an origin a real or imagined ray that projects into space and eventually intersects with the display screen at some point, which is considered to be the location pointed to.

The most commonly used ray-pointing technique uses a physical laser pointer in its normal mode of operation. In laser pointing, the ray is defined by the position and direction of the device used for pointing. In this thesis we explore this technique by substituting an off-the-shelf game motion controller for an actual laser pointer. To determine the position of the controller, we use an additional tracking camera such as Microsoft Kinect device. When an actual laser pointer is used as a pointing device, the computer needs to track the intersection of the laser ray with the large screen, using with cameras and computer vision techniques. Laser pointing with physical laser pointers has been implemented for controlling cursors on large screens and analyzed many times by other researchers (Cavens, Vogt, Fels, & Meitner, 2002; Davis & Chen, 2002; Olsen & Nielsen, 2001).

One way of tracking the intersection is to place a camera in front or behind the large screen. Oh and Stuerzlinger. (2002) present performance evaluations of a computer controlled laser-pointing technique and compare it to a mouse. In Oh and Stuerzlinger's study, researchers placed a camera behind a large screen and visually tracked the ray intersection position. Their results indicate that laser pointer speed performance is approximately 75% compared to a mouse, and an increased error rate was observed. However, researchers note some issues that arise related to camera placement behind the screen, such as brightness and low resolution of the camera. Most importantly, they concluded that even though their prototype is very mobile and accessible, the implementation is very expensive, compared with the price of using mice. Olsen and Nielsen (2001) performed a similar study by detecting a laser pointer on the large screen using a camera.

He achieved substantially better than 50% recognition rates, but faced keystoning and non-linear pincushioning effects from both the camera and the projector. To resolve these issues, the camera needed to be calibrated in advance by mapping 25 points on the screen to the camera image.

Other laser pointing variants include attaching reflective markers to the hand and tracking them using vision technology. Vogel and Balakrishnan (2005) conducted an experiment by positioning passive reflective markers on the hand and tracking them using a Vicon infrared tracker. His research demonstrated high usability of such a system with a very low error rate, typically seen only in such status-quo devices as a mouse. Again, even though it is a very accurate system, it has not proven to be very practical due to the high cost of the equipment. Jota et al. (2010) performed a similar study by implementing laser pointer with an infrared-marked wand with six degrees of freedom.

All laser-pointing variants require additional enhancements to improve interaction with large screens at a distance. Unavoidable hand jitter and unintended hand's down movement while clicking on buttons are some of the main problems that all ray pointing techniques encounter. Several improvement mechanisms have been used: filtering raw data using a Kalman filter (Davis & Chen, 2002; Vogel & Balakrishnan, 2005), altering CD gain (König, Gerken, Dierdorf, & Reiterer, 2009), using bubble ray and speech recognition (Tse, Hancock, & Greenberg, 2007), copying parts of the display being pointed at to handheld device (Myers, Peck, Nichols, Kong, & Miller, 2001) and etc. Even though these improvements help in developing novel pointing interaction techniques, there are tradeoffs such as delay and lag (Pavlovych & Stuerzlinger, 2009) and inability to recognize speech (Tse et al., 2007).

In our experiments we use two types of computational filters to reduce hand jitter and smooth the sensors' raw data. Our study encountered problems with lag so we ran additional tests to improve pointing interaction.

Several researchers explored how laser pointing on one large screen performs in a group collaboration task and how it compares to using multiple mice (Vogt et al., 2004). Observations from this study suggest that using multiple mice is more favorable than using laser pointers, but laser pointers fit better in collaborative activities and information sharing. As group size increased, the presence of multiple cursors made it difficult for collaborative interaction with one display. Vogt et al. (2003) earlier described a technique to capture and differentiate multiple laser pointers simultaneously on the large screen using only one camera if the lasers were pulsed in unique temporal patterns.

Another variant of ray pointing is image-plane pointing that is based on two points in space. Many studies have been performed using a subject's eye position as one point, and the tip of the thumb or a pen pointer as second point in a VR environment (Lee, Seo, Kim, & Park, 2003; Ware & Lowther, 1997; Wingrave, Tintner, Walker, Bowman, & Hodges, 2005). Results have shown that in a 3D environment image pointing is significantly faster than laser pointing (Argelaguet & Andujar, 2009; Bowman & Hodges, 1997; Cavens et al., 2002).

Shoemaker, Tang, and Booth (2007) performed experiments similar to image-pointing studies where one of the two positions of the ray was fixed and a hand was used as a second point in space. This fixed-origin pointing method was explored using an interaction technique that makes use of a perspective projection applied to the shadow representation of the person who is pointing. The light source behind the user acts as a fixed point. Using an infrared camera

and Polhemus magnetic trackers, Shoemaker et al. (2007) were able to use a computed virtual light source that moved with the user as the user moved.

2.2 Overview of Available Tracking Systems

There are number of tracking systems on the market that are not available for individual use due to high cost. Optitrack, Optotrak, and Vicon are each high-speed motion capture systems that use infrared light to detect either specially designed reflective balls or iREDs (infra-red emitting diodes) that can be put anywhere on the body. The disadvantages of these systems are a requirement for accurate calibration, high sensitivity to vibration, bulky size, small tracking volume, and very high cost. In addition, these systems rely on line-of-sight to cameras, which is not always available.

The original Polhemus FASTRAK tracker system is another expensive tracking system that uses a nutating magnetic field for its tracking technology (von Wiegand & Pfautz, 2001). It requires installation of the magnetic field source in the room and for users to carry small trackers. However, the signal from Polhemus can become noisy as the distance between the magnetic sensor and the emitter becomes too large and it is very susceptible to interference from metal objects placed anywhere close to it. The later Polhemus Liberty LATUS system is a slightly different approach, but again relied on modulating a magnetic field to sense six-degree-of-freedom information for objects being tracked (Polhemus, 2013). Ascension Technologies Flock of Birds system uses is another magnetic-based tracker that uses a pulsed magnetic field and claims to have less susceptibility to metal objects in the tracking volume (Ascension Technology Corporation, 2013).

Handheld pointing devices such as the Cyberglove system and the Logitech air mouse can be used for pointing and interaction with displays. The Cyberglove system is a glove worn on the

hand that can track finger movements relative to the hand to detect gestures and detailed movements (Kessler, Hodges, & Walker, 1995). It can be used with other trackers, such as the Polhemus system, to detect global position and rotation data for the glove. The Logitech air mouse acts as a traditional computer mouse but can be picked up and used as a pointer in the air to point at a computer screen.

Microsoft's Kinect camera is a low-cost tracking camera equipped with an RGB camera, a Depth camera, an Infrared light emitter and a microphone array. It uses the depth sensor to track a human body at a distance. The low-cost of the device, its real-time data capability, and a powerful API toolkit make it very compelling for use in enabling applications in various research fields. As an example, the KinectFusion system reconstructs a complete 3D indoor scene using the depth sensor and creates interactions with objects (Newcombe et al., 2011). Genest, Gutwin, Tang, Kalyn, and Ivkovic (2013) developed the KinectArms system that uses Kinect's depth and RGB camera to track multiple hand gestures. The Kinect is mounted over the tabletop display and can be used to track only users' hands. In another research project multiple Kinect devices were attached on top of a cylinder and produced full holographic image of a person. It supports 360^{o} motion parallax, as a viewer walks around the cylinder (Bolton, Wang, Kim, & Vertegaal, 2012; Kim, Bolton, Girouard, Cooperstock, & Vertegaal, 2012). A successful crowd sourced commercial product "eyeCharm" allows any Kinect device to track human eye movements. It uses an additional attachment lens that lets the sensor track both eyes and enables interaction with a computer at a distance (Kickstarter, 2013).

Leap Motion is another commercial motion-tracking device that can be put on the table in front of a computer screen. It uses two cameras and three infrared LEDs to track fingers or any pointing accessories such as a pen. It has a precision of up to 0.01 mm and can be used to track

hand gestures to up to 1 meter in distance. Its low cost and small size have attracted many companies and customers. It was not yet being shipped when our research was starting, but the product now seems to be available, at least for developers (Leap Motion, 2013).

The recent announcement of the new Kinect sensor for the XBOX One gaming console shows significant hardware improvements, as well as enhanced body part tracking technique. Some enhancements include improved skeleton tracking with a rotation data of each body part, operation in the dark and a higher resolution HD camera (Slashgear, 2013).

2.3 One-Part Models of Pointing Performance

There is a rich history of modeling human performance in pointing tasks. The traditional one-part model for pointing performance, known as Fitt's Law (Fitts 1954), has been widely used to predict movement time (*MT*) for pointing in a variety of situations and is the defacto standard for many of the studies reported in the literature.

$$\boldsymbol{MT = a + b\, log_2 \left(\frac{A}{W}\right)} \qquad \textbf{(Equation 2.1)}$$

When using one-part models, the Fitts's law defines movement time (*MT*) as depending on the ratio of movement amplitude (*A*) and target width (*W*). The ($log_2 \frac{A}{W}$) part of the equation is the Index of Difficulty (*ID*). In the Fitts formulation individual values of *A* and *W* are not of importance, only their ratio matters. This is why we characterize Fitts's law as being a one-part model.

Soukoreff and MacKenzie (2004) proposed an alternative one-part model formulation of the Fitts's law, where the index of difficulty is based on an information-theoretic interpretation of Fitts's law, inspired by Shannon:

$$\boldsymbol{MT = a + b\, log_2 \left(\frac{A}{W} + 1\right)} \qquad \textbf{(Equation 2.2)}$$

During the regression analysis, the Shannon-Fitts formulation produces slightly higher R^2 values. In our experiment we will be using the Fitts (Equation 2.1) and the Shannon-Fitts (Equation 2.2) formulations for the one-part models. There have been many variants of the basic Fitts's law model, but all retain the essential one-part reliance on the ratio of A and W.

2.4 Two-Part Models of Pointing Performance

Unlike the one-part models that depend only on the ratio of *A* and *W*, the two-part models for pointing performance allow individual contributions of each *A* and *W* values. Welford first introduced a two-part model that accounts for deviations from an improved version of Fitts's original experiment (Welford, 1968):

$$\boldsymbol{MT = a + b_1 log_2(A) - b_2 log_2(W)} \qquad \textbf{(Equation 2.3)}$$

The Welford's two-part model has been largely overlooked in the HCI literature, although there has been work that suggests that *A* and *W* need to be separable in modeling movement time, especially for large screen displays (Shoemaker, Tsukitani, Kitamura, & Booth, 2012; Wobbrock, Cutrell, Harada, & Mackenzie, 2008). Our experiment results are consistent with these findings.

To combine aspects of the Shannon-Fitts formulation, Shoemaker et al. (2012) introduced a variation of the Welford's two-part model.

$$\boldsymbol{MT = a + b_1 log_2(A + W) - b_2 log_2(W)} \qquad \textbf{(Equation 2.4)}$$

Similar to the Shannon-Fitts formulation (Equation 2.1), the Shannon-Welford model maps the noise and the signal to the distance between movement amplitude A and target width W. However, the signal and the noise are partitioned into separable components, as with the Welford's formulation. In our experiment analysis we use both the Welford (Equation 2.3) and the Shannon-Welford (Equation 2.4) formulations and compare them side-by-side.

2.5 Comparing One-Part and Two-Part Models Statistically

We are interested in determining whether our experiment results are consistent with the findings of other researchers. It has been noted that the two-part models will always outperform the one-part models in the modeling movement time because of an additional degree of freedom (Soukoreff & Mackenzie, 2004). Thus, one of the goals of our research is to determine how one-part and two-part models compare to each other and if they are significantly different in a statistical sense.

For this purpose we use an F-test to compare the two models. This test provides a means of hypothesis testing, and helps to determine if the higher degree of freedom model performs significantly better than the lower degree of freedom model, or if is simply provides a better fit because of its extra degree of freedom. The drawback of the F-test is that it compares only nested model and works only if the greater degree-of-freedom model can be made the same as the lower degree-of-freedom model using an additional constraint. In our case, equations 2.1 and 2.3, and equations 2.2 and 2.4 become equivalent if $b_1 = b_2$. So we can use the F-test to determine whether Equation 2.3 is a better model than Equation 2.1, and we can similarly determine whether Equation 2.2 is a better model than Equation 2.4, but the F-test cannot compare Equation 2.1 against Equation 2.2, or Equation 2.3 against 2.4.

2.6 Effective Width

The width, W, that is used in all of the equations is the apparent width of the target, but it is not always clear how this affects actual pointing actions. The common sense view is that the small the target is, the harder it will be to point to it, but as *W* gets very large or very small, this assumption begins to be questionable. Welford (1968) introduced the effective width W_e that is derived from the distribution of tap positions on the target as an alternate way to characterize the difficult of hitting a target. Unlike the predefined target width W, the effective width is based on how users actually point and how successful they are. Adjustment of a target width by using effective width aligns well with the information-theoretic nature of Fitts's law and other similar models. These width adjustment calculations are based on a 4% nominal error rate. For example, if a nominal error rate of 4% is observed, no adjustment is required, hence $W = W_e$. When the error rate is different, the effective width is calculated in a way that makes the error rate become 4%. Figure 2.1 shows a method of computing the effective width, in effect increasing the effective width until a high error rate is reduced to 4%, or decreasing the effective width until a low error rate rises to 4%.

$$\boldsymbol{W_e = 4.133\sigma} \qquad \textbf{(Equation 2.5)}$$

The effective width can be calculated in two different ways: using the standard deviation of the taps or using the error rate.

The coefficient 4.133 in Equation 2.5 is derived from the probability distribution of a normal distribution $P[Z \geq 2.066] = 0.02$, thus 0.04 is $2.066 \times 2 = 4.133$. For example, if a 2% error was observed on a block of trials with the target width being 5cm, then the effective width can be calculated as $W_e = 2.066 / 2.326 \times 5 = 4.45$ (Figure 2.1)

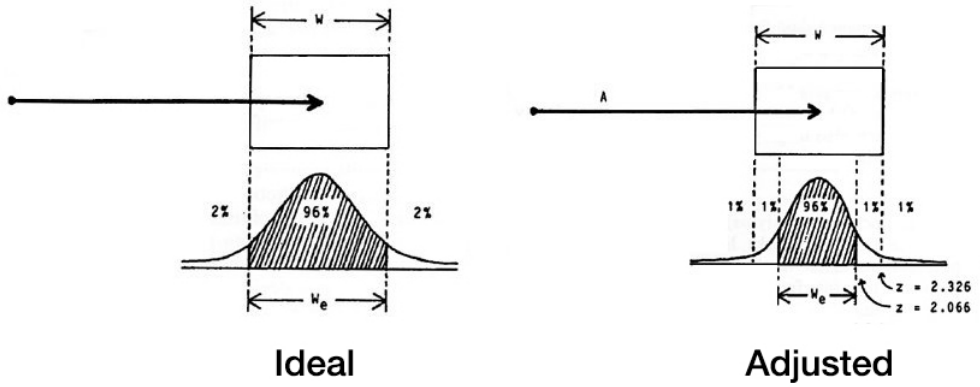


Figure 2.1: Ideal target width when adjustment is not needed and adjusted target width (Mackenzie, 1992).

Similarly, the effective width can be calculated using the observed error rate for a given condition:

$$W_e = \begin{cases} W \times \frac{2.066}{z\left(1-\frac{Perror}{2}\right)} & if\ p_{error} > 0.0049 \\ W \times 0.5089 & \text{otherwise} \end{cases} \qquad \textbf{(Equation 2.6)}$$

2.7 Throughput

Instead of using Fitts's law to model movement time, it can be used to calculate what is called the throughput of the system by measuring several movement times, and then determining how different conditions affect coefficients in the Fitts's law formulation. The throughput combines both the speed and the accuracy from Fitts's law, and is recognized as an ISO standard measure. Throughput has been used by many researchers to compare different conditions and to evaluate

pointing devices (Douglas, Kirkpatrick, & Mackenzie, 1999; MacKenzie, Kauppinen, & Silfverberg, 2001; MacKenzie & Oniszczak, 1998; Soukoreff & Mackenzie, 2004; Oh & Stuerzlinger, 2002).

The throughput is measured in bits per second (bps) and is calculated as:

$$\boldsymbol{Throughput} = \frac{\boldsymbol{ID_e}}{\boldsymbol{MT}} \qquad \textbf{(Equation 2.7)}$$

Where ID_e is the effective index of difficulty, which is defined similarly to the regular index of difficulty, ID, but using the effective width W_e instead of the regular width W:

$$\boldsymbol{ID_e} = \boldsymbol{log_2}(\mathbf{1} + \frac{\boldsymbol{D}}{\boldsymbol{W_e}}) \qquad \textbf{(Equation 2.8)}$$

The throughput is a complete measure containing the speed and the accuracy of a movement performance. In our experiment we first calculate the throughput for each effective index of difficulty and then calculate the grand mean throughput. Finally, the grand mean throughput is used to compare pointing devices with each other.

2.8 Calculation of Pointer Position on Large Display

To calculate the pointer position on a large screen we need a set of coordinates in 6 degrees of freedom: the hand position and the hand rotation. We also need to be able to convert a real distance into display pixels. Thus, it is important to know the exact height, width, and resolution of the screen in order to know the conversion ratio. Furthermore it is important to know the position of a tracking device (i.e., the Kinect) relative to the screen. In our experiments, the tracking device was always positioned in the middle of the screen. Using simple algebra we can determine the *x* and *y* position on the screen in pixel coordinates using the following two equation:

$$x = \left(Tracker_x + Hand_x + Hand_z \times \tan(Hand_{yaw}) \times ppm\right) \quad \textbf{(Equation 2.9)}$$

$$y = Screen_h - \left(Tracker_y + Hand_y + Hand_z \times \tan(Hand_{pitch}) \times ppm\right) \quad \textbf{(Equation 2.10)}$$

Equation 2.9 and equation 2.10 represent the calculation of the *x* and *y* coordinates of a pointer in pixels. The *Tracker* variable is the device position in meters relative to the left bottom corner of the screen, the *Hand* variable is a set of 6 degrees of freedom coordinates, the $Screen_h$ variable represents screen height in pixels and the *ppm* value is the number of pixels in one meter. The hand position coordinates are calculated relative to the absolute position of the tracking device.

Chapter 3: Hardware and Software for Pointing

In this chapter we describe the hardware and software that we developed to support our prototypes for pointing at large-screen displays. We first explain how the final system works and its interaction with the hardware. Then, we explain our initial attempts to build the system and the problems that we encountered throughout the process.

The system had a number of hardware and software components. Because this was an experimental prototype, additional components were used that would be too expensive for a production system in a classroom. The proof-of-concept prototype we developed would need to be refined to use current or future low-cost hardware. The purpose of our prototype was to determine the suitability of current low-cost hardware. The more expensive hardware served as a "ground truth" to measure the adequacy of the low-cost hardware. In particular, the high-end Vicon tracking system would not be used in an actual classroom situation. It was used in the prototype only to detect and sometimes correct for errors in the low-cost tracking hardware.

3.1 Final Hardware Configuration

In our research we used a large glass screen of 5.4m x 2.8m in size. The glass screen was back-projected using a 4 x 3 array of 800 × 600 resolution projectors that had around 150 pixels of overlap between projects on the screen. The glass screen had a resulting resolution of 2720 × 1480 pixels. Three out of twelve projectors did not work due to bulb malfunction, but these were positioned in the corners of the 4 x 3 array comprising the screen so they did not interfere with our experiments. Our experiments used only the central area of the glass screen, not the dark areas in the three corners caused by the broken projectors.

The display was connected to a Windows 7 x64 computer with an 8-core Intel Processor, 6 GB RAM memory, and dual NVIDIA GeForce GTX 260 graphics cards.

A Microsoft Kinect device was used to track a person's hand position. The Kinect device was placed on top of a stool positioned 1.2 meters from the ground and 45 cm in front of the screen. The Kinect device was placed on the stool because the glass screen went from the floor up, so there was no other place to mount the Kinect. According to the Kinect manual it should be placed 0.6 m to 1.8 meters above the ground (Kinect Sensor Placement, 2013). Unlike our experimental setup, most of the screens in real classrooms are positioned at some distance from the ground so the chalk or pen trays that are usually available become the perfect place to attach a Kinect device.

The *Microsoft Kinect for XBOX 360* device is priced at $99 in the online Microsoft Store; the device is equipped with an RGB camera, a depth camera and a microphone array (Microsoft Store, 2013). It also contains an internal motor that changes the angle of the camera from –27° to +27° degrees. The latest *Microsoft Kinect for Windows* device is priced at $149 on Microsoft Store website and is identical to *Microsoft Kinect XBOX 360* device in terms of the hardware. It uses slightly different firmware that allows the Windows version of the device to switch between near and far depth camera views. No other differences between the two devices have been reported. The far mode ranges from 0.8 to 4 meters, while the near mode ranges from 0.4 to 3 meters. As a result, *the Kinect for Windows* has improved depth sensor range from 0.5 m to 4m. Additional third party lens can be used to boost the range.

The Kinect device provides a 43° vertical by 57° horizontal field of view. The depth camera has a maximum resolution of 640 × 480 and runs at 30 frames per second. The Microsoft Kinect RGB camera has a maximum resolution of 1280 x 960 pixels that runs at 12 frames per second

and a default resolution of 640 × 480 that runs at 30 frames per second. For our purposes, 12 fps was too slow so we used the 640 × 480 resolution at 30 frames per second (Kinect SDK, 2013). For all our experiments we used a *Kinect for XBOX 360* device; we will refer to it as simply the "Kinect" from now on (Figure 3.1). The Kinect was directly connected to the computer through a USB connection.

A *PlayStation Move* controller was used to get rotation and position data for a person's hands (Figure 3.2). In addition, we used it as a remote controller for tapping on targets and for manual calibration of the system. PlayStation Move was wirelessly connected to the computer though a Bluetooth connection. It is priced at around $40 in most stores. Alternatively, we also used WiiMote Controller with MotionPlus attachment to get rotation data during pilot studies (Figure 3.3). The PlayStation Move controller was tracked using a Vicon motion capture system to obtain its position (Figure 3.4).

Figure 3.1: Microsoft Kinect for XBOX 360.

Figure 3.2: PlayStation Move controller with reflective markers for Vicon

Figure 3.3: WiiMote controller with MotionPlus attachment.

Figure 3.4: Vicon infrared tracker camera.

Five Vicon infrared cameras were placed around the position where a person would stand. Vicon cameras tracked reflected markers that were placed on the tip of the PlayStation Move controller. Vicon infrared cameras have a resolution of 1600 × 1280 and run at 100 frames per second. Cameras were connected to the computer through wires.

All of the above-mentioned devices were used during experiments for our system to point at a large wall display. Vicon motion trackers represented a high-end device system, while Kinect with a Playstation Move controller was used for a low-end system. Each of these input systems was used one at a time because we needed to compare them with each other.

An i>Clicker base station and an i>Clicker clicker were used to wirelessly control the main software program. This allowed us to switch the input device that was being used by the software (Figure 3.5). The i>Clicker base station was connected to a computer through a USB connection.

The base station communicates with clickers using one of 25 pairs of frequencies. We used the frequency pair whose code is AA in our experiments to retrieve data from the clicker.

Figure 3.5: i>Clicker clicker with i>Clicker base station.

3.2 Final Software Configuration

The main Interaction Manager software is a C# program with an interface for user interaction. It acts as a hub for communication with drivers and subsequent devices. The experimental software shows the real-time data received from the Kinect, the PlayStation Move controller, the i>Clicker base station, and the Vicon Tracker system. It also has functionality to connect and disconnect from each service by restarting the various drivers and the communication protocol (Figure 3.6). The Interaction Manager lets users conduct Fitts's Law experiments and run demos that illustrate how the system can be used in real classroom situations.

The first demo allows users to draw on top of a Microsoft PowerPoint presentation by simply pointing at the slide and clicking a button on the PlayStation Move controller. Users can

select different pen colors and pen thicknesses and they can easily navigate through the slideshow by clicking on predefined buttons on the PlayStation Move controller.

The second demo represents an example of a "memory game" that consists of a 6 × 6 matrix of rectangular buttons that are displayed on the screen. Each button has a label and a color, which are initially hidden from the user. A user has to point at buttons and guess the pairs that are identical. Pointing at a button and "clicking" on it reveals its label and color, but if the two consecutively connected buttons do not have the same label and color, both revert to having their labels and color hidden.

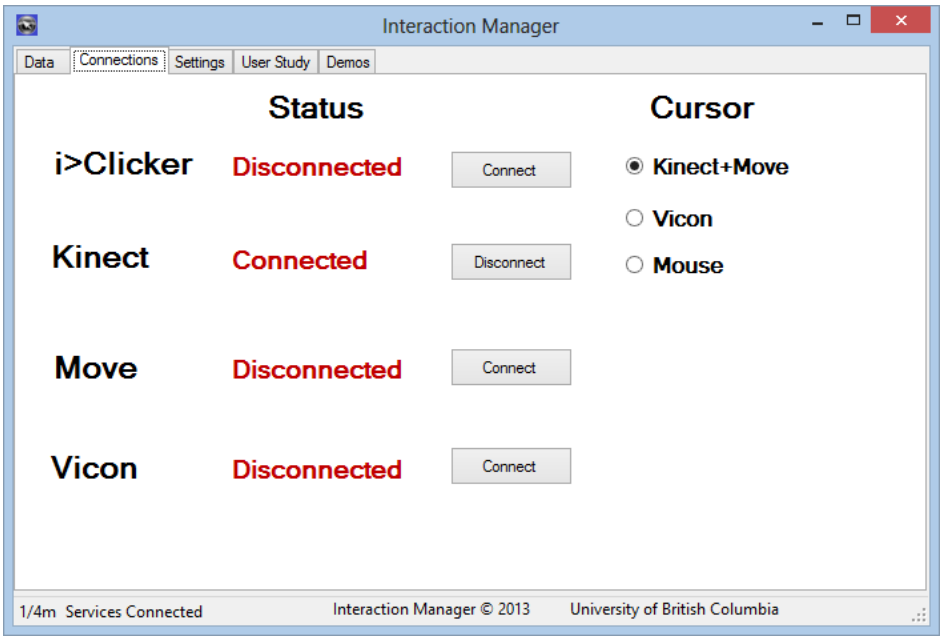


Figure 3.6: Interaction Manager UI.

The Interaction Manager software runs as a separate process that communicates with the hardware drivers over socket connections that retrieves new data as soon as it becomes available (Figure 3.7). All of the communication uses the same protocol, which means that other devices can connect to the Interaction Manager if suitable drivers are written. For example, we have written a driver for a Nintendo Wii motion controller with a Motion Plus attachment that can be

used instead of a PlayStation Move Controller by adding its connection protocol to the driver. We used Motion Plus attachment's gyroscope sensor to calculate the hand's rotation data because the base Wii controller's accelerometer sensor was not sufficiently accurate.

The Interaction Manager constantly checks for new data coming from drivers over the socket connections. As soon as new raw data arrives, the software passes the new data through a low-pass filter algorithm to smooth the raw data, calculates the new position being pointed at on the screen and updates the cursor that is displayed on the screen. Without a low-pass filter the cursor becomes very jittery, making it impossible to point easily. The alpha $\boldsymbol{\alpha}$ coefficient in the low-pass filter controls the smoothness of the data. The low alpha value $\boldsymbol{\alpha}$ results in high lag and slow reaction time, but the cursor moves smoothly and deliberately. High alpha coefficient values twitch too much for comfort. The $\alpha = 0.4$ value was selected for the user study in order to minimize the lag and maximize smoothness of the cursor movement.

The Kinect communicates with the Interaction Manager through a driver that we wrote using the Kinect SDK 1.6 (Kinect SDK, 2013). Our driver detects the user who stands closest to the camera, and retrieves a hand position from the skeleton interface provided by the SDK. Initially, the driver changed the angle of the camera dynamically, adjusting to a user's height, to make sure that she/he fits in the frame. Further tests showed that the Kinect API does not compensate for the difference in angles and outputs data that depends on the angle of the camera. For example, the driver will output different hand positions for the same user when the camera angle changes from 10° to 15° degrees. For the sake of consistency, we fixed Kinect's camera angle at 10° degrees during our user study because the distance from the position of the camera to the user was always constant during our experiments. For future research and in a classroom

application of our system, the positions of a user's hand can be adjusted using Kinect's angle to make sure that they are always in the same world coordinate system.

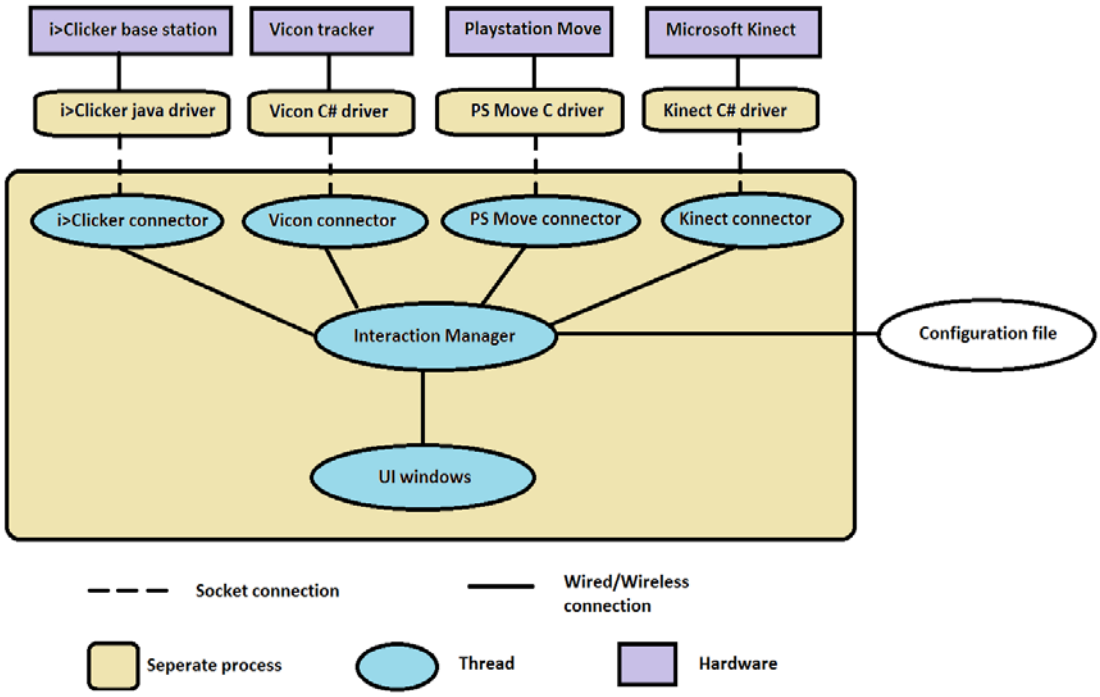


Figure 3.7: Interaction Manager connection scheme.

The Vicon infrared motion tracking system communicates with our software over a socket using the Vicon IQ program. Being an optical motion tracking system, the Vicon system sometimes loses track of reflected markers, placed on the tip of the Playstation Move motion controller. When this occurs, the cursor on the screen will stop moving for a very short time until the Vicon cameras start to again track the markers on the motion controller.

The system was very sensitive to any vibrations that occurred in the laboratory, because the Vicon cameras are mounted on tripods that were not screwed into the floor. Because of this, the cameras needed to be recalibrated every once in a while to ensure the accuracy of the motion capture. These drawbacks are unavoidable limitations of optical tracking systems.

The MoveFramework Library written in C was used to communicate with the PlayStation Move device and to detect rotation data when there were button click events (Moveframework, 2012). All gyroscope-based inertial sensors suffer from the yaw drift problem, especially low quality sensors. The MoveFramework Library uses IMU and AHRS algorithms that are based on the Kalman filter to implement sensor fusion. Using the magnetometer sensor, these filter algorithms compensate for the gyroscope bias that results in a continual drift; it also cleans noisy data from the magnetometer sensor (Madgwick, Harrison, & Vaidyanathan, 2010; Madgwick, Harrison, & Vaidyanathan, 2011). The drift resulting from the gyroscope bias was not completely eliminated by the IMU and AHRS algorithms, but it was significantly smaller than the drift coming from a Nintendo Wii remote controller, whose driver did not have a Kalman filter implementation. Throughout the development we came up with multiple strategies that helped reduce or even eliminate gyroscope drift. We used automatic and manual calibrations during the experiment, when the drift became unavoidably large.

The WiiMoteLib Library (WiiMoteLib, 2009) was used to retrieve all rotation and click events from the WiiMote controller. WiiMouse software (Wii Remote Mouse Application, 2013) was used to establish the connection with the WiiMote controller through the Bluetooth, because the default Windows 7 Bluetooth pairing functionality did not work with the controller.

i>Clicker Java based drivers, written by our laboratory, were used to communicate with the i>Clicker base station. The driver uses its AA pair of frequencies for the communication. In our system it receives clicks from only one clicker. By changing the software's settings file, a user can set any specific clicker id to be used by the program. In the presentation demo, clickers are used to switch modes, advance slides, manually calibrate inertial controllers and exit the software.

During our user study, all of the above mentioned drivers queried the associated hardware devices 100 times a second and sent the data right away to the Interaction Manager through a socket. The main software used a configuration text file to read default configurations (Figure 3.7). The configuration file allows users to modify the relative position of the Kinect device to the screen, adjust presentation screen size in meters, change clicker id and assign different settings for the user study.

3.3 Initial Prototypes for Large Screen Interaction System

Prior to building the final software we made several attempts to create a pointing system using low-cost devices. We wanted to discover all of the advantages and disadvantages of each low-cost device that was used and what needed to be done to make the final software better. We built three prototypes starting from the most basic one:

1) Screen selection in Multipresenter using Microsoft Kinect
2) Kinect based pointing system
3) Kinect based pointing system using Wiimote controller.

The screen selection prototype allowed users to select one of the three wall-size screens simply by pointing at it while standing anywhere within Kinect camera's range. Kinect based pointing prototypes let users point at the desired screen and freely move the computer cursor while standing within range.

3.3.1 Screen Selection in MultiPresenter Using Microsoft Kinect

The first attempt at using the Microsoft Kinect device was to integrate it with existing MultiPresenter software, that shows presentation slides across large output displays (Lanir, Booth, & Tang, 2008). Using this software, a presenter has to use a computer mouse to interact

with a computer, because of the lack of gesture or pointing control. Multipresenter does not have the capability of letting the user select desired display and send input without using of a computer mouse.

Our first program was designed using the Model-View-Controller design pattern, where the Interaction Manager became a model that sends output to the MultiPresenter, while receiving input from the an i>Clicker controller. The main software communicates with the MultiPresenter and i>Clicker drivers over the socket, because each of them are running on separate processes. With this software, a user can stand anywhere within the range of Kinect and point at the desired screen (Figure 3.8). An i>Clicker remote lets users send an input to the selected screen. In our demo, a menu popped up in the middle of the selected screen with five button choices, after which the clicked choice would be highlighted.

We ran our software on 3 displays: two 2.6m × 2.5m projector screens and a laptop computer with the display size of 21cm × 21cm. This pilot experiment showed accurate pointing results of the screen selection while the user was in the range of the Kinect device, which was positioned between two large projector displays. We used user's spine and the right hand out of 20 available joints in the skeleton object, provided by the Microsoft SDK. The user basically moves his/her whole arm to select the screen while facing directly at the Kinect device. By knowing the real size of the displays and their resolution we can use simple geometry to calculate where the user is pointing, while standing anywhere in the room within device range (Section 2.8).

Despite the accuracy of our experiment results, pointing using the whole arm was not intuitive. While pointing at the objects, users tend to use their whole body, twist wrists, elbows

and point using an index finger. Moreover, we wanted to have more control over a computer display, be able to manipulate a mouse pointer, and interact with objects on each screen.

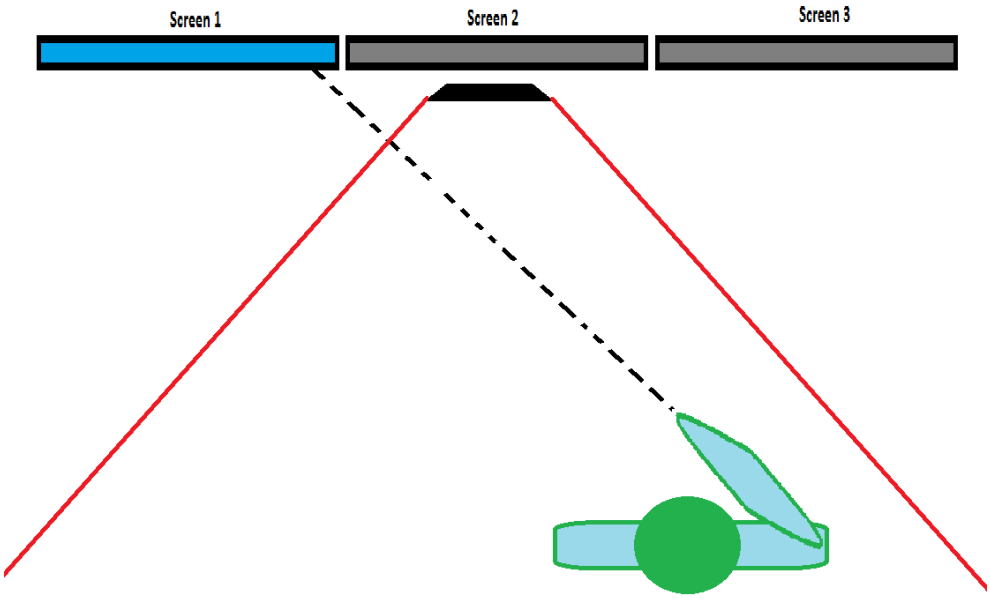


Figure 3.8: User selects first screen using whole arm in MultiPresentor software.

3.3.2 Kinect Based Pointing System

We have developed our next piece of software with an idea to control a mouse pointer using just the Kinect device. The system uses hand and wrist joints from the available skeleton object to calculate the position and rotation of the hand in 6° degrees of freedom (Section 2.8). Unlike the previous system, using hand and wrist joints together was not effective and resulted in very jittery cursor movement. Kinect SDK is not very accurate at distinguishing hand joints at a long distance and often confuses them with other parts of the body during active movement. We tried to smooth the movement using low-pass filter, but the raw data was initially inaccurate.

Extensive testing showed that inaccuracy of pointing resulted from noisy hand joint data, while wrist joint provided accurate raw data. It is most likely because Kinect needs to detect all five fingers in order to determine that the object is a hand.

3.3.3 Kinect Based Pointing System with WiiMote Controller and MotionPlus

We used a WiiMote controller's internal inertial sensors to determine hand rotation. As mentioned previously, we used the WiiMote Library to communicate wirelessly with the controller. The WiiMote Library works only with old versions of controllers, which include only an accelerometer sensor inside, when new WiiMote controllers are also equipped with gyroscope sensors. The accelerometer data was used to compare the changes of hand's position obtained through the Kinect. An average difference was 2.62 cm with a standard deviation of 4.35 cm. However the raw data from accelerometer sensor was not sufficient to calculate the rotation of the controller. To tackle this problem we attached Motion Plus attachment that has an internal gyroscope sensor, which provided yaw, pitch and roll coordinates of the controller.

To reduce pointer's jitter and noise, we used a Kalman-based IMU and AHRS algorithms to combine raw data from accelerometer and gyroscope chips using sensor fusion method (Madgwick et al., 2010; Madgwick et al., 2011). The filtered rotation data coming from the WiiMote controller and the raw position data from the Kinect device are passed through the low-pass filter before calculating the final pointer position. As mentioned previously, we used the low-pass filter with alpha = 0.4 to maximize smoothness of the pointer movement and minimize the lag from the filter.

The pointing performance was significantly improved after adding the WiiMote controller with Motion Plus attachment to the Kinect system, compared to a solely Kinect based system, described in Section 3.3.2. We observed smooth pointing and interaction with displays, however

a constant pointer drift on x-axis was unavoidable. This is a common problem with all low-cost gyroscope sensors that results in a drift yaw position. We could not determine what factors affect the drift magnitude and its sign.

3.3.4 Drift

To avoid the drift problem we came up with the following solutions to try out:

1) Use manual calibration when the drift is noticed.

2) Automatically calibrate the WiiMote using WiiMote Infrared Camera.

3) Automatically calibrate the WiiMote using Kinect's raw data.

4) Use other controllers that have magnetometer in addition to accelerometer and gyroscope.

5) Use Vicon tracker's rotation data to detect a yaw drift and calibrate the controller.

Manual calibration is simple and very effective. When the cursor drifts or appears at some distance on a x-axis from where it was expected, the user needs to point the controller perpendicularly to the screen, and tap on either WiiMote or i>Clicker remote to fix the cursor position. Manual calibration does not work when high magnitude drift is present, because the system becomes unusable and the user cannot perform any work.

We came up with an idea of using the WiiMote Controller's infrared camera to detect infrared light that is emitted by the Kinect device. Using the WiiMote Library we were able to capture this event instantly and because we know the exact position of the Kinect camera from the settings file, WiiMote controller could in theory recalibrate automatically. Unfortunately, tests showed that the WiiMote remote needed to be directly pointed at the Kinect device and this solution was not practical.

The next solution uses hand rotation data, calculated from the hand and wrist joints' raw positions, and then compares it to the WiiMote controller's yaw position. If this difference is

greater than a predefined constant delta value, then the WiiMote needs to be recalibrated using the rotation data obtained from the Kinect device. During our tests we used delta value of 5° degrees, but this solution was impractical either. As shown previously in Section 3.3.2, the hand position, obtained from Kinect, is very inaccurate and unreliable. Tests showed that the difference between WiiMote and Kinect's yaw positions averaged at 6.4° degrees with standard deviation of 14.82° degrees. This is because raw data is very noisy and at times Kinect loses track of a real hand position.

A PlayStation Move controller is another inertial controller on the market with the addition of a magnetometer sensor and a LED feedback light mechanism. We used the MoveFramework library to write the driver that interacts with the device's internal sensors. The addition of magnetometer sensor's raw data helped to fix the yaw drift problem, but the raw data coming from the magnetometer was very noisy and the sensor itself had a high bias. Moreover, compass's noise strength varied from room to room depending on number of surrounding electronics and required to be calibrated. For example, we observed highly jitter movement of the pointer in the laboratory where we held our user studies. The MoveFramework library provides calibration software that helps to reduce magnetometer's bias. As a result, the yaw drift coming from the gyroscope was not as strong anymore, as it was observed with the WiiMote controller.

A Vicon tracker provides highly accurate rotation data that can be used to compare to inertial sensors' raw data, and dynamically calibrate if a drift is observed. The main software compares the difference between yaw rotation from a PlayStation Move and a Vicon tracker, and checks if it is greater than a delta constant. If the difference exceeds a predefined constant delta value, PlayStation Move adjusts its yaw rotation to Vicon tracker's value. This solution was

highly effective during our tests when delta 10° degrees. From now on we will be referring to this device type as a Hybrid Kinect because it uses Kinect's and PlayStation Move controller's data to calculate pointer position, as well as Vicon tracker's yaw data to detect the drift.

Chapter 4: Experimental Evaluation

This chapter describes an experimental study that was conducted to evaluate the suitability of the two mid-air pointing systems described in Chapter 3 for use in a classroom situation. The study compared the high-end Vicon-based system with the low-end Kinect-based system and also compared both against a traditional mouse-based implementation in order to have a baseline for comparing the two systems. The experimental data suggested that the two systems for mid-air pointing had somewhat comparable performance, but they were not as good as a traditional mouse-based implementation, even though the mouse-based implementation did not perform as well as often reported in the literature in terms of measured throughput. As we will explain, there were two problems with the study that reduced its validity: the Kinect-based system was not independent of the Vicon-based implementation and the physical setup for the mouse implementation may have compromised participants' performance in that condition.

We begin with a description of our hypotheses, then we describe the experimental design and procedure, present an analysis of the data from the study, and follow with a discussion and interpretation of the results. As we explain later in this chapter, the results were not conclusive regarding our hypotheses, due to the two problems that we encountered, so we conducted a follow-up study in order to further test the hypotheses. That study is described in Chapter 5.

4.1 Hypotheses

Our primary goal in conducting the experiment was to compare the performance of the high-end Vicon-based system with the lower-cost Kinect and PlayStation Move-based system that we implemented, and to compare both with traditional mouse-based interaction. We looked at both speed and accuracy of pointing. We developed five hypotheses to be tested in the experiment.

The first three hypotheses are related to our primary goals of comparing the three systems to each other. When comparing the two non-mouse systems with each other we had separate hypotheses for speed and accuracy.

H1 Participants using the systems based on mid-air pointing (the Kinect with PlayStation Move and the Vicon tracker) will perform as quickly and as accurately as with the mouse while pointing on large screen displays similar to those used in classrooms.

H2a Participants using the system based on low-cost tracking equipment (the Kinect and PlayStation Move) will perform as quickly as with the system based on expensive tracking equipment (the Vicon tracker) while pointing.

H2b Participants using the system based on low-cost tracking equipment (the Kinect with PlayStation Move) will perform as accurately as with the system based on expensive tracking equipment (the Vicon tracker) while pointing.

We had two hypotheses about the theoretical models that should be used for the types of large-screen pointing we are investigating. The hypotheses are based on previous work by Shoemaker et al. (2012).

H3 One–part formulations (such as Fitts and Shannon-Fitts) will not accurately model the pointing time performance for all conditions.

H4 Two-part formulations (such as Welford and Shannon-Welford) will accurately model the pointing time performance for all conditions.

4.2 Methods

In the experimental study we used three device types for pointing: high-end *Vicon* tracking system, low-end *Hybrid Kinect* with PlayStation Move that depended on Vicon's data, and a

computer *mouse*. Participants were asked to use each of these devices to tap on targets that appeared on the large wall-size display.

4.2.1 Participants

For this experiment we recruited 24 participants though e-mail, 15 were male and 9 were female. Participants' ages ranged from 23 to 36 years. All participants were asked to use their dominant hand, regardless of whether it was their right or left hand. All but one of the participants used computers more than 8 hours a day; 11 participants wear glasses on regular basis. For their time, participants were compensated with $10.

4.2.2 Apparatus

In our experimental study we used our custom made software described in section 3.2.We used three device types (Vicon, Hybrid Kinect and mouse) and they are described in section 3.1. Hybrid Kinect is based on the data from Kinect for XBOX 360, PlayStation Move controller and Vicon. During pilot studies we used PlayStation Move controller's sensors to reduce yaw drift problem. We also used manual calibration technique as an alternative method. Pilot studies showed that these solutions were not sufficient to eliminate the drift, as its magnitude varied from one experiment to another. Due to an unknown cause of the yaw drift's strength change, we decided to use a hybrid Kinect device as a solution to minimize the chance of having malfunctions during the user study.

While using the Move Controller participants were asked to stand 3 meters away from the center of the screen. Participants sat at the table during mouse pointing block of the experiment, which was positioned 1.7 meters away and 3 meters to the left from the center of the screen. This is 3.4 meters absolute distance from the center of the large screen. This design decision was

made to simulate a real lecture mockup when the speaker uses the mouse, positioned to the side of the large screen.

The room setup was essential to get accurate results from the Kinect sensor and none of the participants complained that the stool with Kinect covered a small part of the screen. As mentioned previously, Vicon system sometimes loses track of reflected markers. During the user study these occurrences were recorded and later filtered out before running the statistical analysis. We identified these occurrences when more than three consecutive frames lost track of the markers. Total of 26 occurrences were recorded during the experiment across all participants.

The PlayStation Move controller's buttons also caused additional problems during the experiment. At times the controller would detect double click when the button was clicked only once, which results in false unsuccessful taps. Another error occurred when controller was not registering a tap, which resulted in falsely timed taps. When this happened, participants spent extra time detecting that the tap was not registered. After the experiment, we filtered out 98 unsuccessful taps using the raw data, which is less than 1% of all the registered taps. In our filtering phase, the tap that took less than 0.3 seconds counted as a false unsuccessful tap. The tap that was not registered for more than 3 seconds while the cursor covered the target counted as a falsely timed tap.

4.2.3 Design

A within-subject $3 \times 4 \times 4$ design was used. The independent variables were device type D (Mouse, hybrid Kinect, Vicon), target width W (15cm, 20cm, 25cm, 30cm), and movement amplitude A (60cm, 120cm, 180cm, 240cm). All variables were fully crossed and parameters were nested as follows: for each device type, all of the four target widths were used consecutively in a single block of trials, and for each of those blocks, participants performed

trials for all four target amplitudes A used consecutively. The order of device types were counter balanced among participants, each participant being randomly assigned a device type order. For each A/W pair 11 taps were performed.

In summary, the experimental design was the following:

24 participants × 3 device types ×

4 movement amplitudes (A) × 4 target widths (W) ×

11 taps = 24 participants × 528 taps = 12672 total taps.

4.2.4 Procedure

The experimental task setting is shown in the Figure 4.1. It was a multi-directional point-select task as in ISO 9241-9 (Natapov, Castellucci & Mackenzie, 2009). This task was designed and modeled closely after 1-D tasks by Shoemaker et al. (2012) and the original experiment by Fitts (1954), and 2-D tasks by MacKenzie (1999) and by Natapov et al. (2009).

The experiment was performed over approximately a two-week period. Each participant had a separate single session that lasted approximately 50 minutes, consisting of several parts. Every participant was asked to sign a consent form and fill out a pre-experimental questionnaire (Figure A.1). Each participant was then shown the experimental apparatus and the nature of the experiment and the pointing task were explained. Participants were introduced to the “Whac a Mole” game and were told that there would be three blocks of trials in the experiment, one for each device type. Each trial used a single size for all targets. The targets appeared in a circular pattern. Starting with one target, they were sequentially selected, with each subsequent target appearing on the opposite side of the circle, until all 12 targets had been selected (Figure 4.1). The first target appeared randomly on the circle. Its time performance and accuracy were not logged. This is because the previous mouse position was unknown and would not have matched

the pre-defined distance between targets. A rectangular image of wooden hammer with size approximately 10 × 10 cm replaced the traditional cursor image that indicated where a participant was pointing. As the hammer image hovered over the mole, the image became red to give users feedback that they had acquired a target.

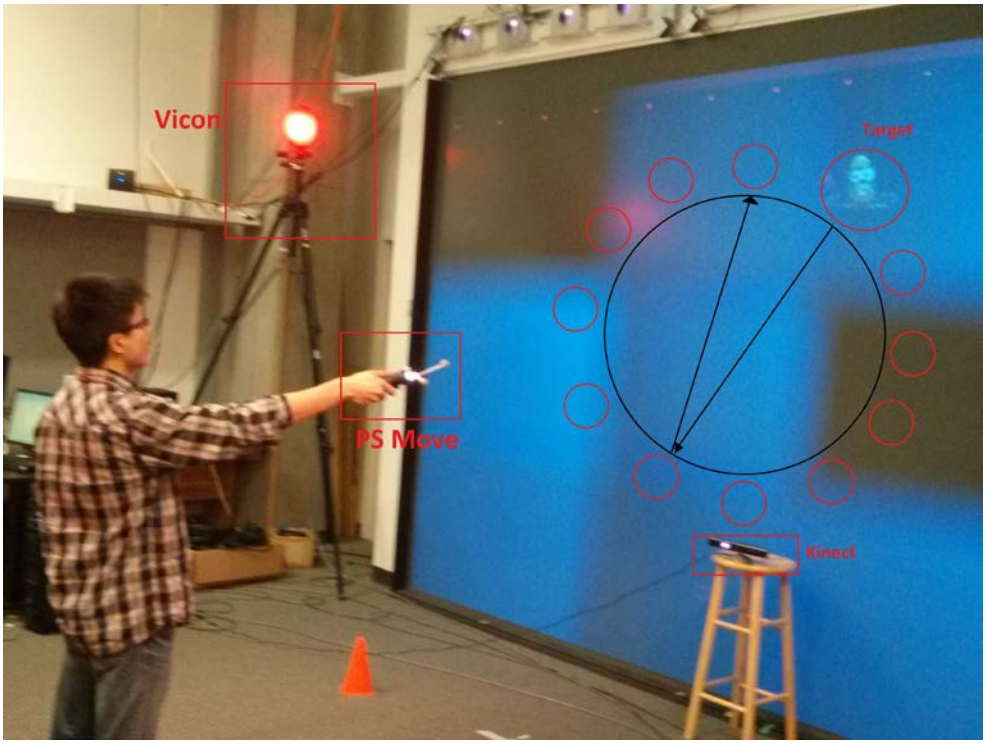


Figure 4.1: A participant pointing at a target during the pilot study.

For the mid-air pointing conditions the cursor position was computed as described in Section 2.8 for the mouse condition. Participants received 20-millisecond long vibration feedback when tapping on each target. As a participant tapped, the previous target disappeared and a new target appeared on the edge of the circle, regardless if the participant accurately clicked on the target or not. Each trial processed in sequence of 12 taps, from which 11 were recorded.

Participants were instructed to be as accurate as possible by attempting to reach a goal of 95% accuracy. As a second priority, participants were instructed to hit targets as quickly as they could while maintaining high accuracy. Each participant was asked to manually calibrate the

PlayStation Move controller before starting trials in both the Kinect and Vicon conditions. During the experiment, participants used the same calibration procedure if the cursor experienced drift.

Each participant was asked to complete a set of practice trials that lasted around 7 minutes. During the practice trials participants performed a shorter version of the experiment where each trial consisted of only two taps. The device types were presented in the same order as in the actual experiment. Participants were advised to take a short break of at least three minutes between each block of the experiment. Participants were encouraged to take longer breaks but none of them did so. Throughout the study participants did not know their score during the game, because we did not want them to get distracted.

During the experimental trials, the block for each device type, started with a pair of practice trials with a randomly chosen A / W pair to allow participants to get used to the new device type. Participants were not informed that the practice trial at the beginning of each block would not have its data analyzed.

After all three blocks were completed, participants were asked to complete a post-experiment (Figure A.2) survey that requested information from them about the overall experiment and how they felt about the different blocks. Participants used all three types of devices during the practice and the experiment, but did not know which of the two devices was being used during the non-mouse blocks of trials. Each block, representing a device type, was associated with a unique color bar on top of the game screen. Because participants did not know which of the three devices was used in some blocks, participants were asked to refer to each block by its color during the post-experiment questionnaire. Participants were reminded which color represented which block when necessary.

4.2.5 Measures

The pointing performance was measured as the time taken to tap on a target. The timing began for each A/W pair when a participant tapped on the first target, positioned randomly along the circle, and ended after the twelfth tap. Errors were measured at each tap depending if the user clicked on the target successfully or not. Exact tap positions along with distance travelled were also recorded.

4.3 Results

In this section we present the results from our experiment for movement time and error rate. In the next section we discuss these results and whether they support our hypotheses.

Before the analysis we needed to be sure that there was no learning effect in the study. We performed a within-subjects ANOVA to determine if the order of the blocks had any impact on the movement time or the error rate. Results with sphericity assumed, showed no significant effect of block on either movement time ($F(2,8446) = 0.136, p = 0.873$) or error rate ($(F2,8446) = 0.191, p = 0.826$). As a result, we included all data sets in our analysis.

Results showed that the sphrecity assumptions were not violated for the effect of device type and movement amplitude on performance time and error rate. We used a Greenhouse-Geisser correction for the rest of the effects and interactions (W, D × W, D × A, W × A, D × W × A) which did violate Mauchly's test of sphericity (Table 4.1).

Factor	Measure	Sphrecity Correction	df	F	p	Partial η2
D	MT	Sphrecity Assumed	2	638.4	.000	.708
		Greenhouse-Geisser	1.99	638.4	.000	.708
	ErrorRate	Sphrecity Assumed	2	3.312	.037	.012
		Greenhouse-Geisser	1.88	3.312	.040	.012
W	MT	Sphrecity Assumed	3	486.4	.000	.649
		Greenhouse-Geisser	2.72	486.4	.000	.649
	ErrorRate	Sphrecity Assumed	3	56.19	.000	.176
		Greenhouse-Geisser	2.33	56.19	.000	.176
A	MT	Sphrecity Assumed	3	1686.	0.00	.865
		Greenhouse-Geisser	2.93	1686.	0.00	.865
D X W	MT	Sphrecity Assumed	6	22.16	.000	.078
		Greenhouse-Geisser	4.97	22.16	.000	.078
	ErrorRate	Sphrecity Assumed	6	2.211	.039	.008
		Greenhouse-Geisser	4.56	2.211	.057	.008
D X A	MT	Sphrecity Assumed	6	10.57	.000	.039
		Greenhouse-Geisser	5.65	10.57	.000	.039
W X A	MT	Sphrecity Assumed	9	7.508	.000	.028
		Greenhouse-Geisser	7.45	7.508	.000	.028
D X W X A	MT	Sphrecity Assumed	18	3.174	.000	.012
		Greenhouse-Geisser	14.3	3.174	.000	.012

Table 4.1: Significant ANOVA results for movement time (MT) and error rate.

4.3.1 ANOVA for Movement Time

Significant main effects of *D*, *W* and *A* were found. Significant interaction effects of $D \times W$, $D \times A$ and $D \times W \times A$ were also found. Significant results for movement time are shown in Table 4.1. Detailed significant results along with the effect size and a pairwise comparison table of interactions are presented in Table A.2 and Table A.3.

Device	Fitts			Welford					F-test		
	a	b	R2	a	b1	b2	k	R2	F	p	sig?
Mouse	519.9	200.2	0.867	-188.	225.0	130.0	0.577	0.905	5.132	0.004	Yes
Vicon	661.0	233.3	0.957	866.6	226.1	253.7	1.121	0.96	0.829	0.378	No
Hybrid Kinect	624.2	270.7	0.973	437.5	277.3	253.2	0.913	0.975	0.838	0.376	No

Table 4.2: Modeling movement time (ms) using Fitts and Welford formulations based on actual width *W*.

Device	Shannon-Fitts			Shannon-Welford					F-test		
	a	b	R2	a	b1	b2	k	R2	F	p	sig?
Mouse	356.9	239.5	0.87	-413.	271.2	170.8	0.629	0.912	6.115	0.027	Yes
Vicon	468.2	280.0	0.967	633.6	273.2	294.8	1.078	0.969	0.645	0.434	No
Hybrid Kinect	404.8	323.5	0.975	167.2	333.3	302.3	0.907	0.977	1.374	0.262	No

Table 4.3: Modeling movement time (ms) using Shannon-Fitts and Shannon-Welford formulations based on actual width *W*.

Device	Fitts			Welford					F-test		
	a	b	R2	a	b1	b2	k	R2	F	p	sig?
Mouse	440.1	224.3	0.923	180.5	229.8	193.2	0.840	0.927	0.649	0.434	No
Vicon	604.0	239.3	0.904	1419	220.5	336.2	1.524	0.93	4.769	0.047	Yes
Hybrid Kinect	515.4	290.7	0.907	409.0	292.8	277.5	0.947	0.907	0.059	0.811	No

Table 4.4: Modeling movement time (ms) using Fitts and Welford formulations based on expected width W_e.

Device	Shannon-Fitts			Shannon-Welford					F-test		
	a	b	R2	a	b1	b2	k	R2	F	p	sig?
Mouse	253.7	270.0	0.938	5.728	276.3	240.4	0.869	0.942	0.792	0.389	No
Vicon	418.2	283.3	0.915	1223.	261.7	373.8	1.428	0.939	5.145	0.040	Yes
Hybrid Kinect	294.1	342.9	0.911	201.7	345.0	331.9	0.962	0.912	0.045	0.834	No

Table 4.5: Modeling movement time (ms) using Shannon-Fitts and Shannon-Welford formulations based on expected width W_e.

A linear regression was performed using the data obtained from the experiment (Table A.1). A similar regression analysis was used in earlier studies (Fitts, 1954; Soukoreff & Mackenzie, 2004). Regression constants based on actual width W were calculated using Fitts's model ($MT = a + b\ log_2 (\frac{A}{W})$), Shannon-Fitts model ($MT = a + b\ log_2 (\frac{A}{W} + 1)$), Welford two-part ($MT = a + b_1\ log_2(A) - b_2\ log_2(W)$), and Shannon-Welford model ($MT = a + b_1 \log_2(A + W) - b_2 \log_2(W)$). These results are shown in Table 4.2 and Table 4.3. Another analysis of results was performed using effective width W_e computed using both methods as described in section 2.6 (Zhai, Kong, & Ren, 2004). In our calculations we used W_e computed using the error rate. Effective width calculation is based on the spread of data and represents what participant actually

performed (Soukoreff & Mackenzie, 2004). Results based on effective width W_e are summarized in Table 4.3 and Table 4.4.

Moreover, we ran F-tests using both actual and expected width to determine if two-part model formulations significantly better describe the data than the one-part model formulation. As described in section 2.5 we perform F-test to compare pairs of nested models. The F-test that we use is *F(3 - 2, n – 2),* where 3 is the number of parameters in the two part model, 2 is the number of parameters in the one part model, and n is the number of sample points. F-test results using actual width are presented in Table 4.2 and Table 4.3, and results using expected width are summarized in Table 4.4 and Table 4.5.

Two-part models, using actual width W in calculations, showed significantly better fit than one-part models while using Mouse device. When expected width W_e was used, two-part models characterized data significantly better than one-part models while using Vicon device type. Figure 4.2 and Figure 4.3 summarize R^2 values as they vary for all devices across one-part and two-part models. For both one-part and two-part models, using Vicon and hybrid Kinect device types, fit is steeply reduced while using expected width in calculations. However, whilst using mouse, device performance is slightly increased when using expected width.

4.3.2 ANOVA for Error Rate

The mean error rate for all devices was found to be 2.36%. Significant main effects of *D, W* and somewhat significant effect of $D \times W$ interaction were found. These results are summarized in Table A.2 and Table A.*3*

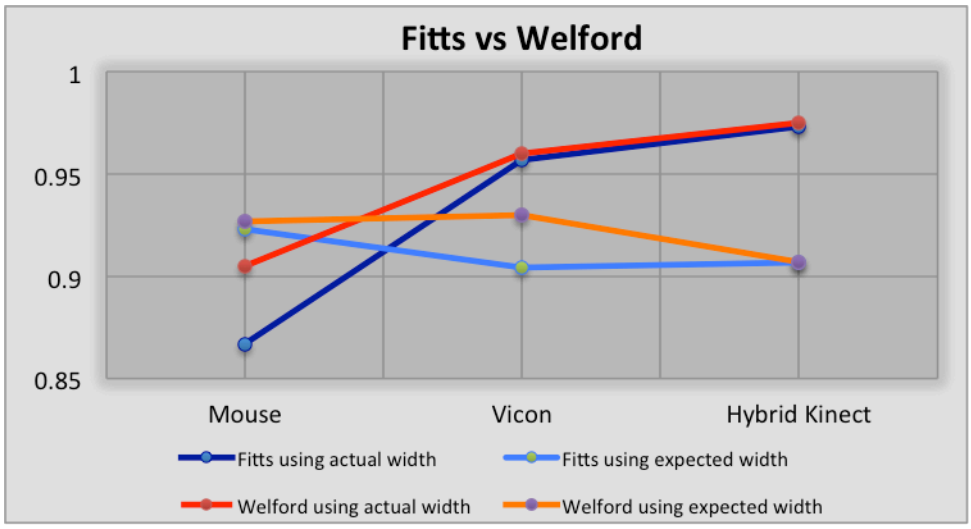


Figure 4.2: Regression R^2 values as they vary for each device for Fitts and Welford formulations.

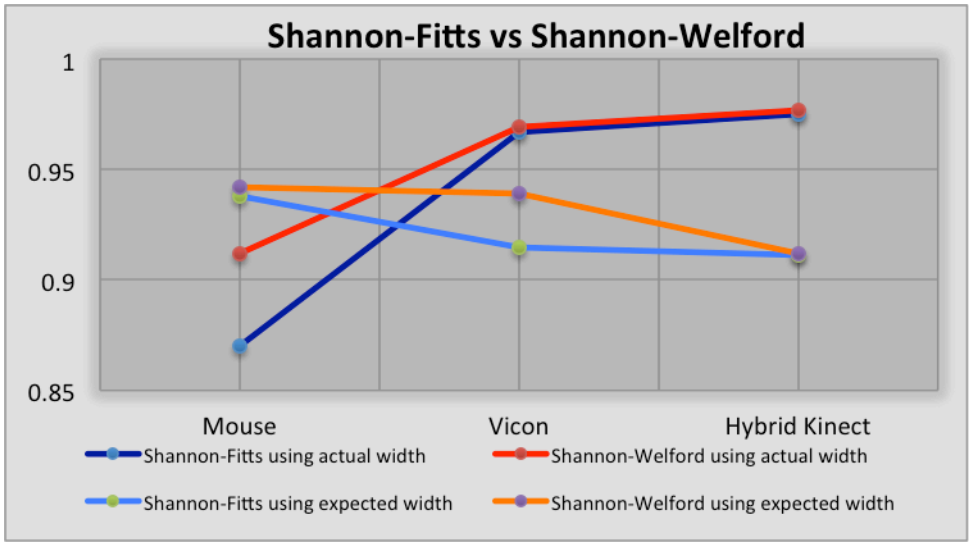


Figure 4.3: Regression R^2 values as they vary for each device for Shannon-Fitts and Shannon-Welford formulations.

4.4 Discussion

Although the data make it clear that participants were able to point fairly quickly and accurately using the two systems we developed, neither performance nor accuracy was comparable to the mouse condition, which served as a control condition in the study. We first discuss how all three devices compare to each other by performance time and accuracy, compare how one-part and

two-part models accurately predict the performance and then we look at some of the shortcomings of the study as it was conducted.

4.4.1 Performance Time

Significant effects of *A (F(2.931, 2.953) = 1686.045, p < 0.0005)* and *W (F(2.721, 2.331) = 486.43, p < 0.0005)* on movement time (*MT*) were expected and are consistent with results obtained by other researchers (Casiez, Vogel, Balakrishnan, & Cockburn, 2008; Shoemaker et al., 2012). The finding, that both the movement amplitude and the target width impact *MT*, is fundamental to any discussion on the pointing performance time.

The finding of the significant impact of device type *D* on performance time *MT (F(1.995, 1.6884) = 638.404, p < 0.0005)* was somewhat surprising and the plot is shown in Figure 4.4. The mouse was significantly faster than the Vicon and Kinect for all *A* and *W*. This is probably due to the fact that all users are accustomed to using mice rather than pointing in the air. It is worth noting that unlike other device types used in our experiment, a mouse requires a desk and a hard surface on which to track. Furthermore, in most classroom set-ups a mouse is positioned to the side of the large screen on the computer desk and this setup restricts the instructor to move freely around a classroom. However, the PlayStation Move controller requires to be held in hand while in use. Some participants used different strategies to reduce the tension in their wrist while pointing in the air.

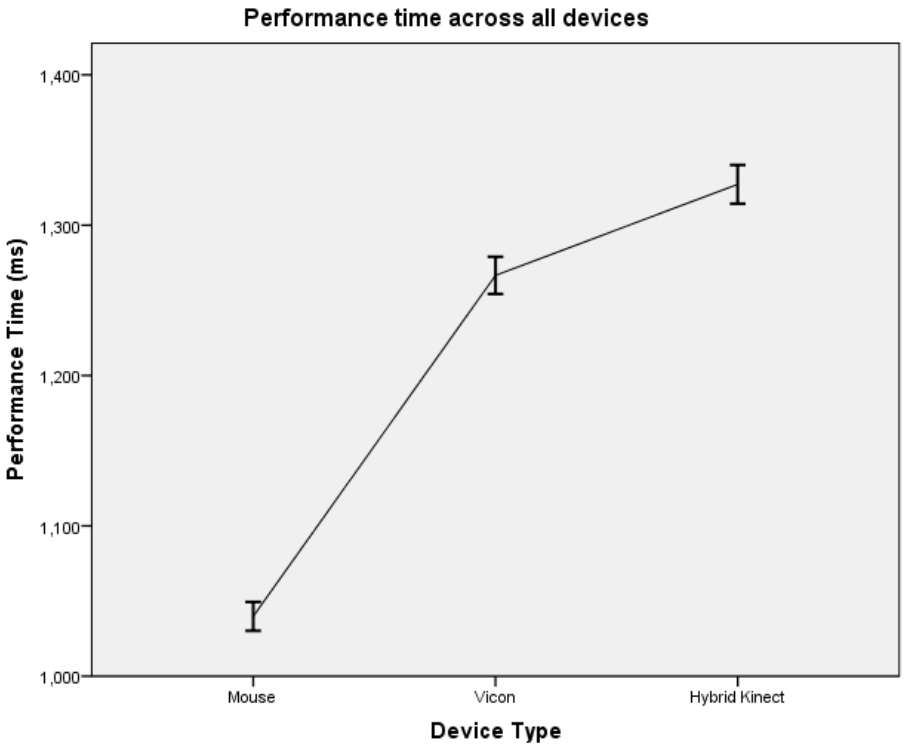


Figure 4.4: Performance time for three device types.

The pairwise comparison Table A.3 shows that the Vicon device is significantly faster (p = 0.0005) than the hybrid Kinect but the average time difference is relatively small. The average time difference between the Vicon device and the hybrid Kinect system was 60.579ms, while the pair difference between the Mouse – Vicon and the Mouse – hybrid Kinect was 226.821ms and 287.401ms respectively. Even though these results show that the low-cost Kinect system is almost as fast as the high-cost Vicon tracker, we cannot fully rely on them, because of the hybrid Kinect's dependence on the Vicon's yaw data. We ran a pilot study, described in the next chapter, to determine if results are consistent when the Kinect device is used independently from the Vicon device.

To compare three devices, we calculated the throughput *TP*, described in section 2.7, measured in bits per second, and serving as a quantitative measure to compare device types used in our experiment. The throughput was calculated from mean *TP* from each subject across all

conditions. These results are summarized in Figure 4.5. The throughput of the mouse is 2.98 bps, while hybrid Kinect's and Vicon's throughput is 2.19 and 2.26 bps respectively. Results show that the Mouse device outperformed both tracking devices. The hybrid Kinect's throughput is consistent with the results obtained by Natapov et al. (2009), where researchers used the Nintendo WiiMote controller. The Mouse was used as a baseline condition in our experiment and all participants mentioned that they are frequent mouse users. Two participants stated that pointing using the Vicon and the hybrid Kinect system was similar to Nintendo Wii's game interaction. Clearly, the mouse experience was a confounding variable. Throughput results of a mouse obtained by other researchers range from 3.5 to 4.5 bps (Soukoreff & Mackenzie, 2004) while our results showed 2.98 bps throughput. This can be explained that the desk was positioned to the side of the screen during the experiment and subjects did not have a perfect view of the screen. During the experiment two participants complained that they could not see the target well enough from that angle.

In terms of throughput, the Vicon tracker's performance is 3% better than the hybrid Kinect's and 24% worse than the Mouse's bps value. Both tracking devices' *TP* values are at high end of the 0.99 – 2.99 bps values reported for a PC's touchpad used by other researchers (Soukoreff & Mackenzie, 2004). While bps values are not as good as for a mouse, both tracking devices have respected throughput values.

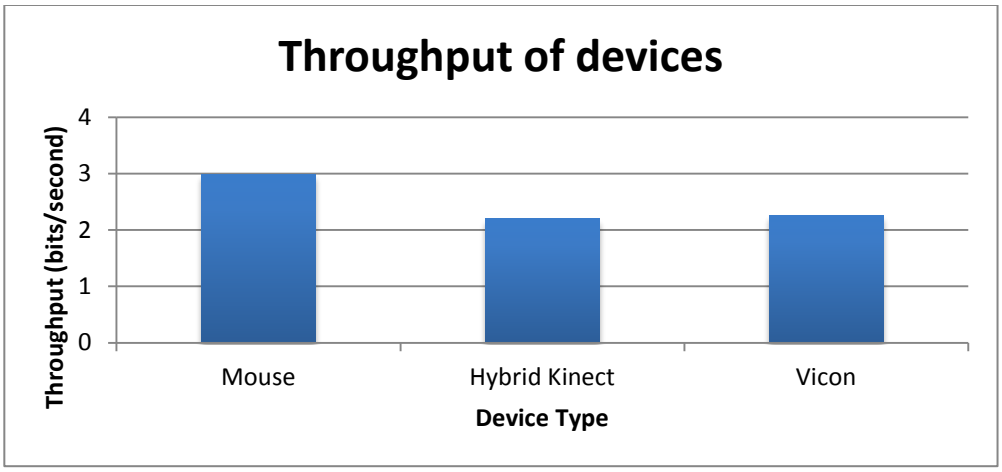


Figure 4.5: Throughput of devices measured in bits per second.

From these results we cannot yet conclude that the hypothesis H1 is rejected. To fully reject the hypothesis we will need to confirm that mouse device is more accurate than handheld tracking devices used in the experiment. To confirm this, we need to use the table surface for the mouse, positioned in front of the screen, and the non-hybrid version of Kinect system, independent from the Vicon tracker for fixing pointer drift. Similarly, we need to prove that the non-hybrid Kinect with the PlayStation Move system is as fast as the Vicon tracker.

During the experiment participants could not distinguish which device could have been a low-cost one. 7 out of 24 participants thought that the hybrid Kinect system was better than Vicon, while 6 preferred the Vicon device. Almost all participants thought that using the mouse was easier and faster. 4 participants thought that the position of the desk affected their performance while using the mouse and they preferred the desk to be positioned in front of the screen. As mentioned previously, participants were referring to each used device during trial block with a color and did not know which one was which.

Pairwise comparison table for $D \times W \times A$ interaction revealed the following noteworthy facts (Table 4.6 and Table 4.7). The Vicon system shows significant performance improvement than the hybrid Kinect for A = 240cm across targets with the target width ranging from 15 to 30

cm. In addition, the Vicon showed significant performance improvement for the amplitude A ranging from 120 cm to 180 cm for the target width $W >= 25$ cm. Moreover, the Vicon tracker performed as fast as the Mouse for targets with 20cm width and movement amplitude of 120 and 180 cm.

Measure	W (cm)	A (cm)	Device 1	Device 2	Mean Difference (device1-device2)	Std. Error	*p*
MT (ms)	15	240	Vicon	Hybrid Kinect	-103.1	36.759	0.0161
	20	240	Vicon	Hybrid Kinect	-80.52	26.326	0.0073
	25	120	Vicon	Hybrid Kinect	-53.05	20.768	0.0335
	25	180	Vicon	Hybrid Kinect	-106.7	22.506	0.0000
	25	240	Vicon	Hybrid Kinect	-95.87	24.921	0.0004
	30	120	Vicon	Hybrid Kinect	-110.8	21.695	0
	30	180	Vicon	Hybrid Kinect	-93.54	21.705	0
	30	240	Vicon	Hybrid Kinect	-127.7	22.919	0

Table 4.6: Interesting significant results obtained from device type pairwise comparison table of $D \times W \times A$ interaction effects.

Measure	Device	A (cm)	W 1 (cm)	W 2 (cm)	Mean Difference (W1 - W2)	Std. Error	*p*
MT	Mouse	60	20	25	22.757	11.463	0.2889
		120	15	20	7.8522	15.663	1
		180	20	25	41.712	18.770	0.1626
		240	15	20	41.409	23.280	0.4586
		240	15	25	-35.78	28.565	1

Table 4.7: Interesting significant results obtained from width pairwise comparison table of $D \times W \times A$ interaction effects.

In our $D \times W \times A$ analysis we observed some unexpected results that do not follow Fitts's model and are worth mentioning. The mouse performance was significantly decreased for targets with $W = 25$ cm comparing with smaller targets with $W = 15$cm at high movement amplitudes $A = 240$cm. While using a mouse, subjects did not show significant performance improvement between W ranging from 15 to 20 cm at $A = 120$ cm and $A = 240$cm. Similarly, there was no significant difference observed when the target width varied from 20 to 25cm for $A = 60$ cm and $A = 180$cm. These findings are surprising and do not follow any model. These anomalies might

be due to the fact that the table surface for the mouse device was not positioned in front of the screen.

4.4.2 Pointing Accuracy

Even though a throughput of devices is a good measure to detect how quickly subjects perform the task, it does not show how successful the task of target selection was.

The significant main effect of the actual width W on the pointing accuracy ($F(2.721, 2.221) = 56.197, p < 0.0005$) was expected because the smaller the target width, the easier to miss. The significant main effect of the device type ($F(1.884, 495.436) = 3.312, p < 0.05$) was somewhat surprising (Figure 4.6). The effect size of D on the error rate was very small (partial $\eta^2 = 0.012$). We expected the hybrid Kinect to have more misses than Vicon because of the cost, and the measured signal noise of the Vicon to be not more than 0.15mm. The interaction effect of a device and width was somewhat significant ($p = 0.057$) with a low effect size (partial $\eta^2 = 0.008$).

The pairwise comparison table (Table A.3) showed no significant difference in the error rate for the Mouse and the hybrid Kinect ($p = 0.067$) with the mean difference being 0.9%. Subjects were less accurate using Vicon device than Kinect, but the difference was not significant.

For very small targets W = 15cm, the mean error rate was 6.9% for the Mouse, 4.3% for the Vicon and 4.8% for the hybrid Kinect. Even though there was no significant difference between the Vicon and the hybrid Kinect, Figure 4.7 depicts decreased accuracy for very small targets. For very small targets, pairwise comparison table of $D \times W$ interactions revealed that the Mouse was significantly less accurate ($p = 0.033$) than the Vicon. This might be due to the fact that it was hard to see very small targets when the desk was not positioned in front of the screen. Moreover, there was no significant accuracy difference for all devices when a target was one width unit different.

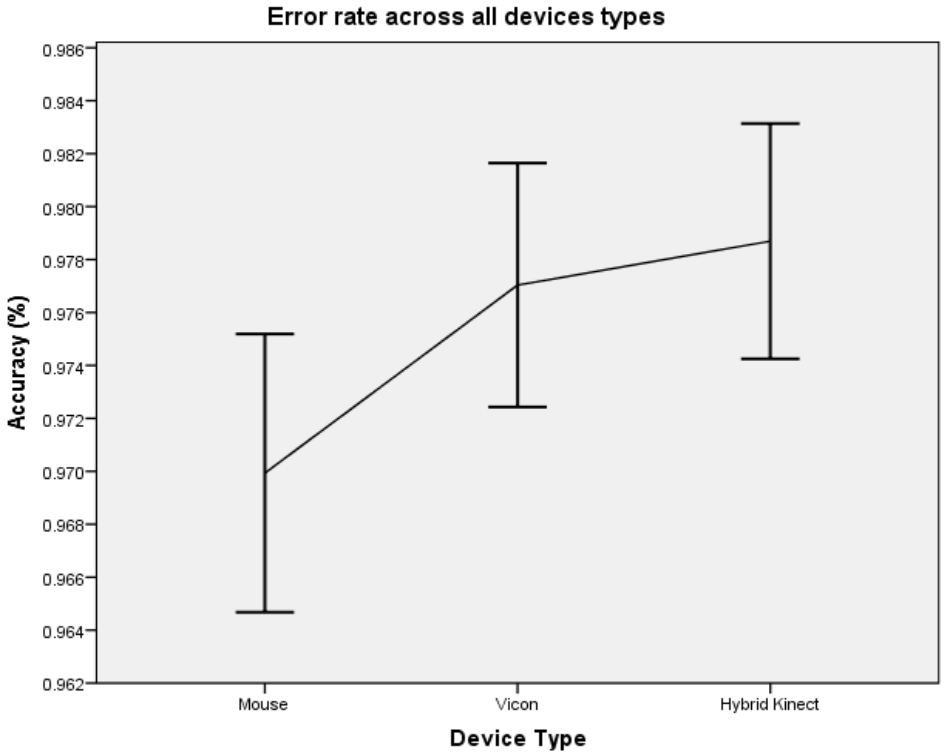


Figure 4.6: Error rate for three device types.

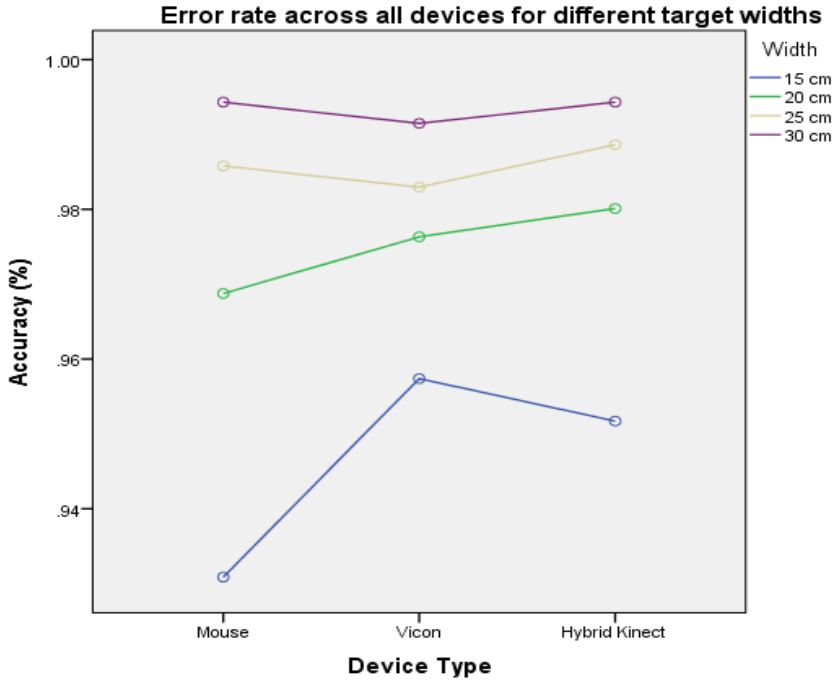


Figure 4.7: Error rate by device type and target width.

For example, error rates for targets with W = 15 cm and W = 20 cm were not significantly different for all device types.

These results are different from our expectations but make sense after running the pilot study where we changed the experiment design in order to understand results better (Chapter 5). We cannot yet reject null hypotheses until we tackle all the questions that arose during the discussion section.

4.4.3 One-Part versus Two-Part Models

Figure 4.2 and Figure 4.3 summarize regression fits for one-part and two-part models based on actual target width (W) and expected width (W_e). Even though there is no universal standard to determine an acceptable fit using models to calculate performance time, we used R^2 >= 0.9, proposed by Mackenzie (1992) as a guideline in our discussion.

For one-part models, both Fitts and Shannon-Fitts formulations accurately characterized fit for the Vicon and hybrid Kinect devices using actual target width with all R^2 values greater than Mackenzie's R^2 = 0.9 threshold. Both one-part models showed poor fit for the Mouse as a device type with R^2 less than R^2 = 0.9 threshold. The Fitts model produces R^2 = 0.867 for the Mouse, R^2 = 0.957 for the Vicon and R^2 = 0.973 for the hybrid Kinect, while Shannon-Fitts produces R^2 = 0.87 for the Mouse, R^2 = 0.967 for the Vicon, R^2 = 0.975 for the hybrid Kinect values. Therefore, both formulations produce relatively similar R^2 values although the Shannon-Fitts model is slightly better than the original Fitts formulation.

Results that used expected width W_e as a target width often more accurately represent the task and, hence, more relevant in the discussion (Soukoreff & Mackenzie, 2004). When we substituted W with W_e, all residual values for both one-part formulations produced fitting accuracy higher than Mackenzie's R^2 = 0.9 threshold. However, R^2 values for both handheld

devices decreased significantly. Fitts formulation produced $R^2 = 0.923$ for Mouse and R^2 ranged from 0.904 to 0.907 for Vicon and hybrid Kinect devices. As expected, Shannon-Fitts formulation produced slightly accurate residual values of $R^2 = 0.938$ for Mouse, $R^2 = 0.915$ for Vicon and $R^2 = 0.911$ for hybrid Kinect.

Poorer fit for the mouse can be explained by the *k* values that we calculated from two-part models (Table 4.3, Table 4.5). While using the actual target width, *k* values ranged from 0.57 to 0.63 and deviate more from 1 than *k* values that use the expected width. When using the actual width both one-part models produced very good fit for handheld devices (Vicon tracker and hybrid Kinect). However, for the Mouse device type there was an unacceptable fit. When W_e was used, the quality of fit increased for the mouse and decreased for both Vicon and hybrid Kinect devices. Nevertheless, all residual values were greater than Mackenzie's $R^2 = 0.9$ threshold.

Both two-part models (Welford and Shannon-Welford) produced an acceptable fit higher than $R^2 = 0.9$ threshold for all devices when using a target width W in calculations. Similar to one-part models, the Shannon-Welford formulation ($R^2 = 0.905$ for the mouse, $R^2 = 0.96$ for the Vicon, $R^2 = 0.975$ for the hybrid Kinect) produces slightly accurate regression fit than the Welford formulation ($R^2 = 0.912$ for the mouse, $R^2 = 0.969$ for the Vicon, $R^2 = 0.977$ for the hybrid Kinect).

When W_e values were used, regression R^2 values displayed better fit for the Mouse device with $R^2 = 0.927$ in the Welford formulation and $R^2 = 0.942$ in the Shannon-Welford formulation. R^2 values steeply decreased for both handheld devices with R^2 ranging from 0.907 to 0.93 using the Welford model and R^2 ranging from 0.912 to 0.939 using the Shannon-Welford. However, all regression values were higher than Mackenzie's $R^2 = 0.9$ threshold value. Similar to actual width calculations, Shannon-Welford produced slightly better regression fits for all three devices. It is

important to note that the regression fit steeply decreased for the hybrid Kinect when W_e values were used in calculations for both two-part models.

F-test results indicate that both two-part models produce significantly better fit than one-part models using the mouse as a pointing device and a target width W. Moreover, F-test results revealed that both two-part models produce significantly better fit than one-part models when Vicon tracker and target width W_e were used in calculations.

4.5 Conclusions

Fitts's law has been a traditional method for measuring pointing performance and creating predictive models while pointing and interacting with large displays. Low-cost tracking cameras and inertial devices have been widely used in gaming industry in recent years but have not been explored in depth for in class large screen pointing. Our first contribution compares the pointing performance and the accuracy of low-cost and high-end devices. The second contribution in this chapter consists of comparing one-part and two-part model formulations to predict the pointing performance for traditional mouse and two handheld mid-air pointing devices.

We conducted an experiment, in which we compared performance values from the Fitts-like one-part models with the Welford-like two-part models side by side. Common drifting problems in inertial sensors resulted in using a hybrid system, which slightly depended on the high-end device's data. In the experiment we used a computer mouse, placed on the table to the side of the screen, a high-end device, and a low-cost hybrid device that depends on high-end device. Experiment results showed relatively similar performance and same high accuracy of low-cost gaming hardware to high-end tracking systems. The computer mouse did not perform as we had expected and we believe this is due to the position of the desk, where mouse has been placed on

during the experiment. We also conclude that two-part models outperform one-part models in prediction of pointing performance for this experiment design.

In the next chapter we describe a revised version of our experiment that corrects for some of the problems we encountered and provides evidence that better supports some of the hypotheses, although the fundamental difference between performance with traditional mouse-based interface and performance with either of the mid-air techniques remains.

Chapter 5: An Exploratory Follow-up Study

To answer questions that arose during the experiment described in the previous chapter, we decided to slightly modify the design and run a pilot follow-up study. The pilot study was similar to the experiment described in Chapter 4, except that we made several changes to the apparatus and the procedure to address some of the problems that arose (these are described in section 4.4). We were interested to see if correcting these problems would provide results that better supported our hypotheses.

Findings from this follow-up study suggest that the lower accuracy of the mouse that we observed in the main experiment was due to the position of the desk. We found that the mouse is more accurate when the desk is positioned in front of the screen. This means that instructors may be likely to make more errors by using a classroom computer and mouse positioned to the side of the large presentation screen rather than pointing using handheld tracking devices. Results from the pilot study also suggest that the dependency of the Hybrid Kinect on the Vicon's data negatively affected the speed of pointing.

5.1 Methods

In the pilot follow-up study we used three device types: Vicon, non-hybrid Kinect and a mouse. We will refer to the non-hybrid Kinect as simply the Kinect in the remainder of this chapter. Subjects were asked to use each of these devices to tap on targets that appeared on the large wall-size display.

5.1.1 Participants

Six members from our research laboratory participated in the follow-up experiment. They were not formal participants because they were part of the research team and thus were considered to be co-experimenters in the informal pilot study. All participants were familiar with the experiment and two of them had participated in the main experiment. Five of the participants were males and one was a female.

5.1.2 Apparatus

In the follow-up study the mouse was placed on the table surface, positioned 3 meters in front of the large screen. This is the same position where participants stood with the PlayStation Move controller during the main experiment. The table was moved because we wanted to know if the position affects the accuracy and performance during pointing when using a mouse. Even though throughput of the mouse was higher than the Vicon and hybrid Kinect's throughput in the main study, it was lower than the values obtained by other researchers (Natapov et al., 2009). Furthermore, 4 participants from previous study mentioned that they would have pointed more accurately if they had had a direct view at the large screen.

Secondly, unlike in the main experiment, the Kinect camera with PlayStation Move did not depend on the Vicon tracker's rotation raw values that were previously used to fix yaw drift problem. Instead, we asked participants to manually calibrate the PlayStation Move controller when it was needed. The manual calibration technique is described in section 3.3.3. We wanted to use fully standalone low-cost tracking system to determine if the drift problem would decrease the pointing performance and accuracy. The non-hybrid Kinect for XBOX 360 with a PlayStation Move system is referred to as simply the Kinect device in our discussion. All other

configurations, including the Vicon tracker, were unchanged and were exactly the same as in the main experiment described in Chapter 4. The pilot study was conducted in the same laboratory under the same conditions except for the changes already noted.

5.1.3 Design

A within-subjects design was used. All independent variables were identical to the ones used in the main experiment and were fully crossed. The order of device types were counter-balanced among participants and each subject was randomly assigned to each device order. The design was as following:

6 members of the research team × 3 device types ×

4 movement amplitudes (A) × 4 target widths (W) ×

11 taps = 6 participants × 528 taps = 3168 total taps.

5.1.4 Procedure

The overall experimental procedure was exactly the same as in the main experiment that was described in the previous chapter.

5.2 Results

We performed an ANOVA identical to the one described in Chapter 4. All main effects and interactions are shown in Table B.1, Table B.2, Table B.3 and Table B.4. In addition, we performed the same regression analysis to determine how the position of the computer mouse and independent Kinect system affects prediction models.

5.3 Discussion

We discuss only results that are different from the ones obtained from the main experiment, and describe how these new findings support our preliminary hypotheses.

We first examine whether one-part and two-part model formulations support our hypotheses. We then look at whether the significant main and interaction effects of movement time and error rate support our hypotheses.

H1 The system based on mid-air pointing equipment (i.e. a Kinect with PlayStation Move or a Vicon tracker) will perform as fast and as accurately as the mouse for pointing on large screen displays.

Somewhat supported.

H2a The system based on low-cost tracking equipment (i.e. a Kinect and PlayStation Move) will perform as fast as the system based on expensive tracking equipment (i.e. a Vicon tracker) while pointing.

Supported.

H2b The system based on low-cost tracking equipment (i.e. Kinect with PlayStation Move) will perform as accurately as the system based on expensive tracking equipment (i.e. Vicon tracker) while pointing.

Supported.

H3 One–part formulations (i.e. Fitts and Shannon-Fitts) will not accurately model the pointing time performance for all devices.

Not supported.

H4 Two-part formulations (i.e. Welford and Shannon-Welford) will accurately model the pointing time performance for all devices.

Supported.

Consistent with the main experiment's results, the computer mouse was significantly faster than the Vicon tracker and the non-hybrid Kinect with PlayStation Move controller. However, the mouse performance was greatly improved when the table surface for the mouse was placed in front of the large screen (Figure 5.1). In addition, the mouse's pointing accuracy was improved (Figure 5.2), but unlike the findings in the main experiment, there was no significant difference in accuracy observed among all three devices ($F(1.683, 93.320) = 1.720$, $p = 0.188$, partial eta squared = 0.026).

Unexpected results were observed during the main experiment and we wanted to know if they were caused by the position of the mouse. When the table surface for the mouse was positioned in front of the screen, no abnormalities were observed and the pointing performance followed the traditional Fitts model. However, similar to the results discussed in Chapter 4, there was no significant difference observed in the pointing performance when the target width W ranged from 20 cm to 25 cm for amplitude A = 60 cm and A = 240 cm.

To compare the pointing time performance of devices, the throughput was calculated for all three devices and results are shown in Figure 5.3. Unlike the results obtained in the Chapter 4, the throughput of the mouse in the pilot follow-up study is consistent with the findings by other researchers (Natapov et al., 2009; Soukoreff & Mackenzie, 2004).

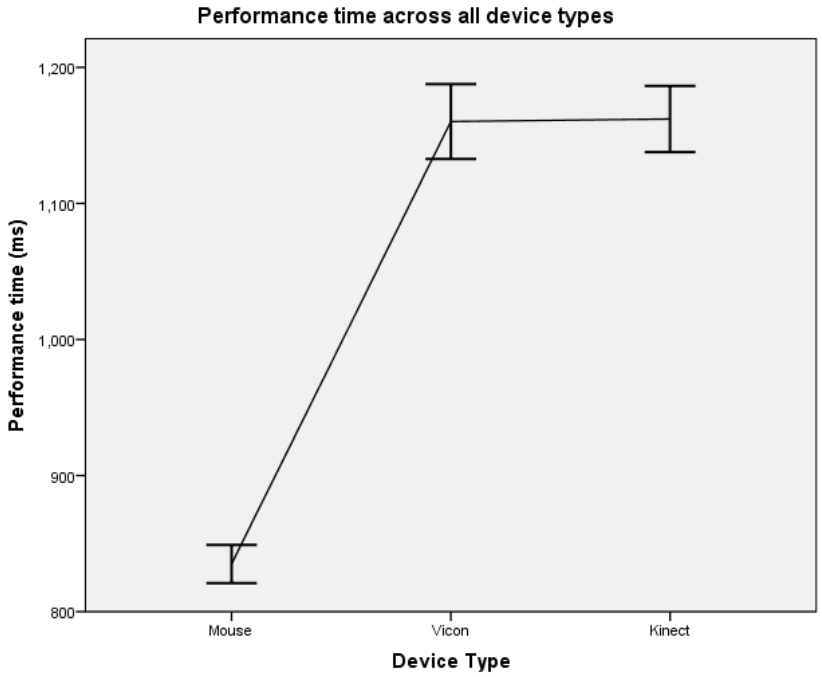


Figure 5.1: Performance time for three device types.

To summarize, the Mouse performed faster than did the handheld devices when the table was positioned in front of the large display. Even though the mouse has a higher throughput and a significantly higher pointing performance ($p = 0$) than the Vicon tracker and Kinect, it is not significantly more accurate. Hence, the hypothesis **H1 is partially supported**.

Results indicate no significant difference in the pointing performance between the Vicon tracker and the non-hybrid Kinect with PlayStation Move device. In fact, the mean difference in pointing performance reduced from 60.579 ms (Table A.3) to 1.85 ms (Table B.42). Thus, we can conclude that the non-hybrid Kinect system is no worse than the Vicon system in pointing performance (speed) and hypothesis **H2A is fully supported**.

As a result, the use of Vicon's rotation data to correct hybrid Kinect's yaw drift problem, negatively affected the pointing performance during the main experiment. It is important to note that none of the users manually calibrated the PlayStation Move controller during the pilot. Even though the pointer did drift to the right or to the left on the x-axis, it did not cause any

inconvenience during the pointing. During the pilot study none of the users noticed that the drift even existed.

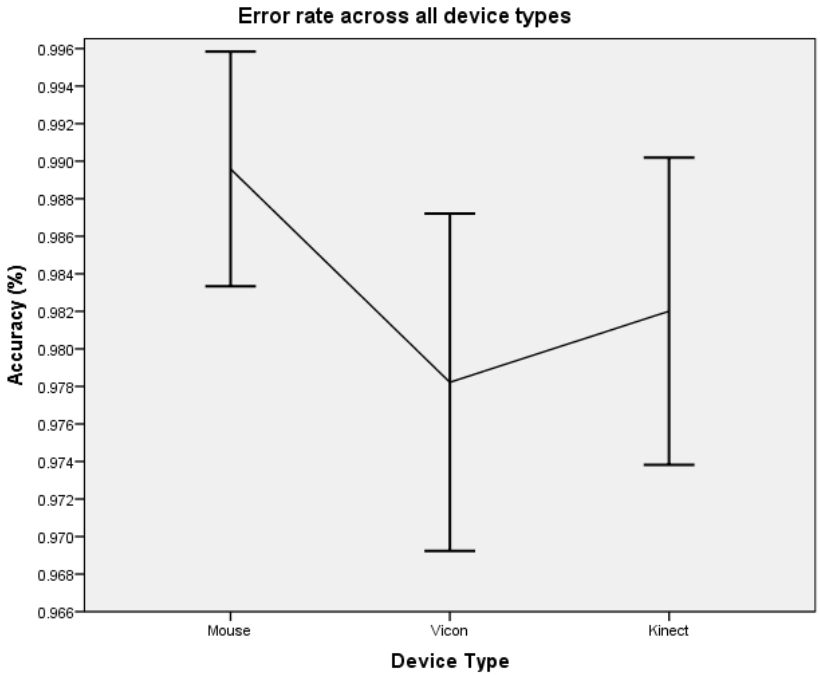


Figure 5.2: Error rate for three device types.

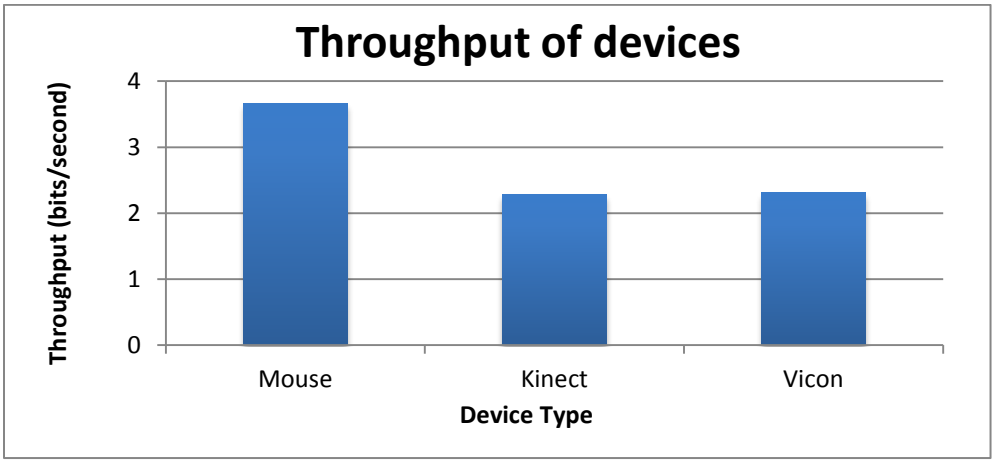


Figure 5.1: Throughput of the Mouse, Kinect and Vicon devices in bits per second.

Moreover, consistent with the main experiment, there was no significant difference in the pointing accuracy between Kinect system and Vicon tracker (Table B.1). Hence, we can conclude that the hypothesis **H2B is fully supported**.

Figure 5.4 and Figure 5.5 summarize R^2 values as they vary for all devices across one-part and two-part models. Unlike the regression values obtained in the previous experiment for the Mouse using actual width (R^2 = 0.867 using the Fitts formulation, R^2 = 0.905 using the Welford formulation), the center position of the table surface showed much better fit (R^2 = 0.945 using the Fitts formulation, R^2 = 0.946 using the Welford formulation). Moreover, when the expected width was used in the calculation, it slightly reduced the fit instead of improving it, but still higher than Mackenzie's R^2 = 0.9 threshold value.

The non-hybrid version of the Kinect with PlayStation Move produced better fit than hybrid Kinect in the main experiment when the expected width was used, both greater than R^2 = 0.9 Mackenzie's threshold value (R^2 = 0.907 using Fitts for the hybrid Kinect, R^2 = 0.938 using Fitts for the non-hybrid Kinect).

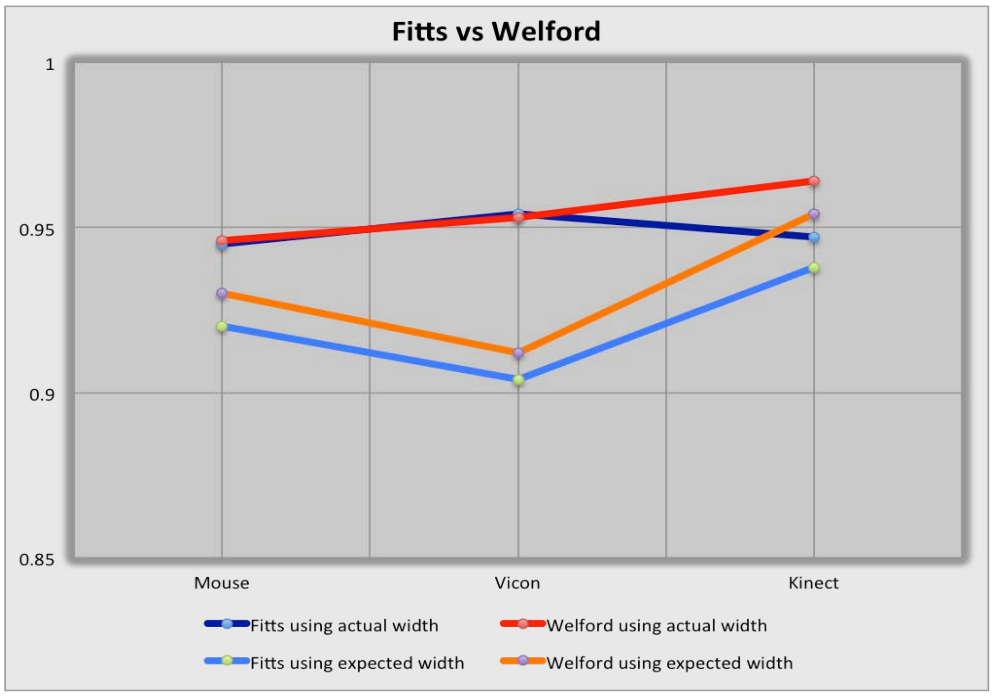


Figure 5.2: Regression R^2 values as they vary for each device for Fitts and Welford formulations.

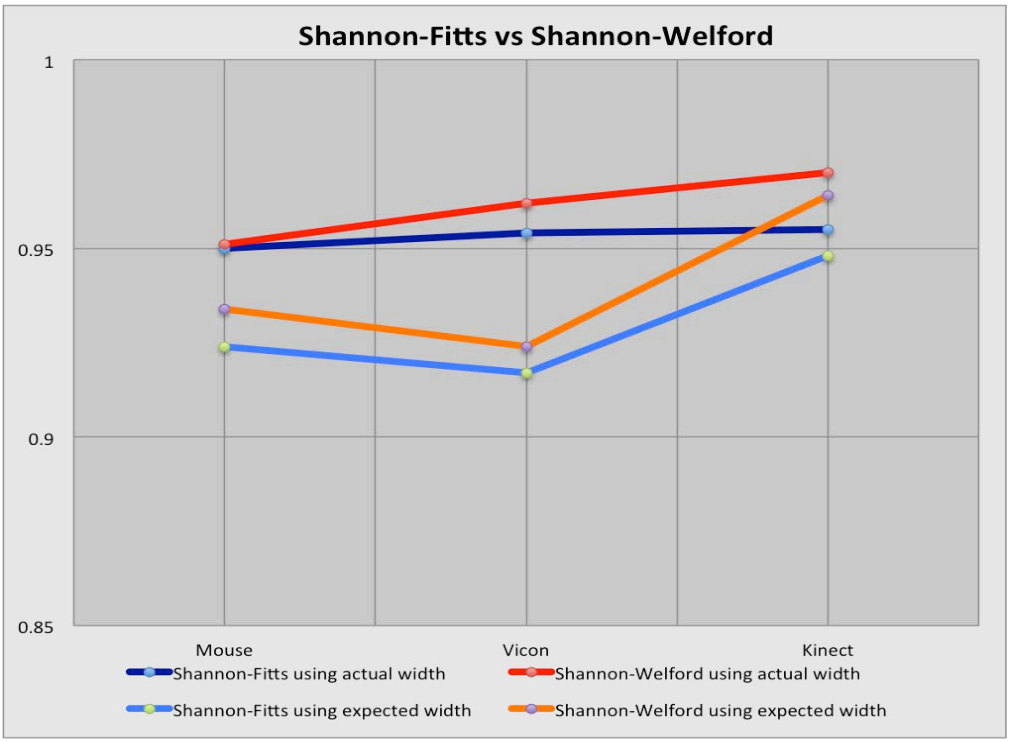


Figure 5.3: Regression R2 values as they vary for each device for Shannon-Fitts and Shannon-Welford formulations.

To summarize, both one-part and two-part models, using either W or W_e, produced a better fit for the Mouse device when the table surface was positioned in front of the large screen instead of when it was positioned to the side, as described in the main experiment in the Chapter 4. In addition, the non-hybrid Kinect produced higher residual values than the ones obtained for the hybrid Kinect. As expected, residual values for the Vicon tracker system did not change because the design parameters were not changed. In fact, residual values for all three devices, obtained in the pilot using the Fitts, Shannon-Fitts, Welford and Shannon-Welford models, are higher than the Mackenzie's $R^2 = 0.9$ threshold (Figure 5.4 and Figure 5.5). Hence, both one-part and two-part models do accurately model pointing performance for all three devices, regardless if actual or expected width is used. Using these results, we can conclude that the hypothesis **H3 is fully rejected** and hypothesis **H4 is supported**.

5.4 Conclusions

In this chapter, we looked more closely at the problems in the main experiment, described in Chapter 4, slightly changed the experimental apparatus, and ran a follow-up study. The change to the apparatus was fully isolating the Kinect with PlayStation Move system from Vicon tracker (non-hybrid Kinect) and positioning the table surface for the Mouse in front of the large wall display. Our results indicate that the position of the mouse greatly affects the accuracy and performance but it is not significantly more accurate than the non-hybrid Kinect system. Moreover, we can conclude that the PlayStation Move's yaw drift does not affect the pointing time performance and accuracy, and can be manually calibrated at any time. The regression analysis for both one-part and two-part models produced a good fit, higher than the Mackenzie's $R^2 > 0.9$ threshold value, used in Chapter 4.

The pilot follow-up study showed fast and accurate pointing using low-cost off-the-shelf hardware equipment. However, this equipment has been used under laboratory conditions and it is yet to be discussed in Chapter 6 whether the same performance can be obtained while using the system in real life classroom, and finding out how practical it is.

Chapter 6: Pointing at Large Screen Displays in a Classroom Setting

We conducted a test to determine the maximum tracking coverage area during pointing interaction and to determine how well our system performs in a real classroom setting. In order to maximize tracking coverage we determined the optimal placement of multiple Kinect Cameras along the wall in the classroom. This allows users to freely walk around the room and point on the large wall screen. Based on this, we make recommendations for future studies that might run our system with multiple tracking devices in a real classroom.

6.1 Tracking Coverage Test

The future work for our research includes placing multiple Kinect devices along the wall to cover larger classroom areas and to track instructors during classroom presentations. In order to determine the accuracy of the Kinect's specifications, outlined in section 2.2, we performed a test in a university classroom. The goal of this test was to understand, how close Kinect devices need to be placed to each other in order to maximize coverage area. Kinect's depth camera was configured to run at 30 frames per second at a resolution of 640 x 480 pixels.

The classroom was 8.5 meters wide and 14 meters long. The screen was 2.26 meters wide and 1.727 meters high. The *Kinect for XBOX 360* was positioned on the chalkboard, 0.88 meters beneath the screen. In this pilot study we used non-hybrid version of Kinect system that was used in Chapter 5. The test consisted of blocks of trials at 110 different positions in the room. A person who is highly familiar with the task was positioned in 0.5-meter increments away from the Kinect, either to the right or left relative to the center, where Kinect device was positioned. During each trial we performed a pointing task replicating our main experiment. The task involved tapping on targets of widths *W* with movement amplitude *A*. We refer to a position as

out of range, being defined as 15 unsuccessful consecutive taps or a total loss of a cursor control. Out of 110 tested positions 84 were out of range. We performed 175 taps at each position that was in range.

6.2 Test Results

Table 6.1 represents the pointing performance time of our system where Kinect-center is the position of Kinect device. Table 6.2 shows the pointing accuracy at all positions. Positions marked as N/A are out of range positions.

We clearly see a trapezoidal pattern, showing the angular view of the depth sensor. Almost all in-range positions have at least 95% accuracy, meaning that if the subject is in depth sensor's range, then the system will be at least 95% accurate. At 2.5 - 3 meters away from the device, the pointing performance time is slightly increased; however the accuracy is still the same as at other positions. Surprisingly, the depth sensor operates from 1 to 4.5 meters away when the subject stands in front of the device. However, it operates only from 1 to 3.5 meters when subjects stand up to 1 meter to the either side of the device.

	Position relative to Kinect (m)										
	To the left					Center		To the right			
Distance (m)	2	1.75	1.5	1.25	1	0	1	1.25	1.5	1.75	2
0.5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1	N/A	N/A	N/A	N/A	964.7	947.2	958.3	N/A	N/A	N/A	N/A
1.5	N/A	N/A	N/A	N/A	891.2	892.2	893.3	N/A	N/A	N/A	N/A
2	N/A	N/A	N/A	N/A	865.5	836.7	838.3	N/A	N/A	N/A	N/A
2.5	N/A	N/A	N/A	832.6	826.7	816.3	818.3	834.4	N/A	N/A	N/A
3	N/A	N/A	810.6	802.0	833.9	768.8	826.3	806.5	799.8	N/A	N/A
3.5	N/A	N/A	N/A	N/A	1614.	845.4	1430.	N/A	N/A	N/A	N/A
4	N/A	N/A	N/A	N/A	N/A	889.6	N/A	N/A	N/A	N/A	N/A
4.5	N/A	N/A	N/A	N/A	N/A	948.2	N/A	N/A	N/A	N/A	N/A
5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 6.1: Pointing performance (MT) results at different positions.

	Position relative to Kinect (m)										
	To the left				Center			To the right			
Accuracy (%)	2	1.75	1.5	1.25	1	0	1	1.25	1.5	1.75	2
0.5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1	N/A	N/A	N/A	N/A	0.977	0.977	0.975	N/A	N/A	N/A	N/A
1.5	N/A	N/A	N/A	N/A	0.954	0.971	0.958	N/A	N/A	N/A	N/A
2	N/A	N/A	N/A	N/A	0.965	0.988	0.973	N/A	N/A	N/A	N/A
2.5	N/A	N/A	N/A	1	0.988	0.982	0.984	0.992	N/A	N/A	N/A
3	N/A	N/A	0.988	0.988	1	0.994	0.994	0.991	0.976	N/A	N/A
3.5	N/A	N/A	N/A	N/A	0.822	0.942	0.846	N/A	N/A	N/A	N/A
4	N/A	N/A	N/A	N/A	N/A	0.937	N/A	N/A	N/A	N/A	N/A
4.5	N/A	N/A	N/A	N/A	N/A	0.923	N/A	N/A	N/A	N/A	N/A
5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 6.2: Pointing accuracy (1 - error rate) results at different positions.

This means that we have a rectangular area that covers all positions from 1 to 3.5 meters away and up to 1 meter to the either side from the device location, resulting in $5m^2$ coverage when using single *Kinect for XBOX 360*. It is important to note that the *Kinect for Windows* works at 0.5 meters away from the device in a near-range mode and results in $6m^2$ coverage per device.

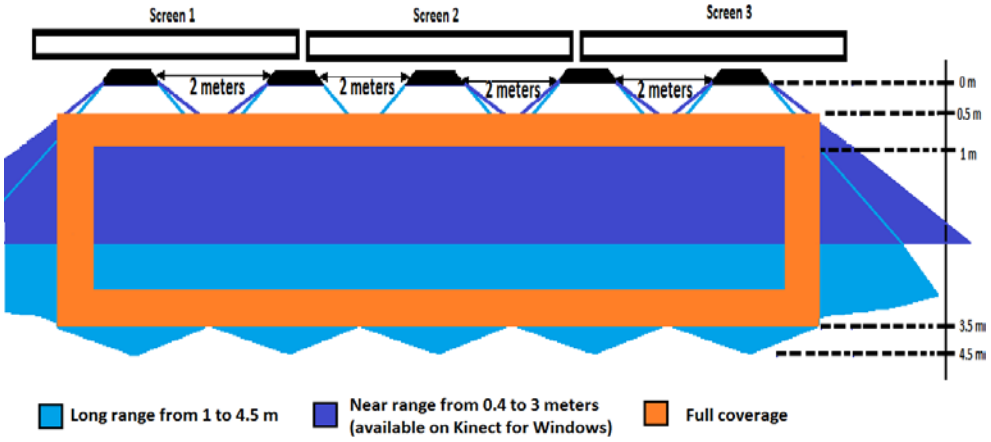


Figure 6.1: Proposed full classroom coverage with 3 large displays and 5 Kinect for Windows devices.

Test results show that if we were to put multiple Kinect devices side by side along the large screen display to maximize larger area, we would need to put them 2 meters apart from each

other to have full coverage from 0.5 to 3.5 meters away from the screen. This coverage ensures practical pointing performance and at least 95% accuracy during pointing at large screen display in classroom setting. We get coverage from 0.5 to 3 meters using the near-mode and 1 to 3.5 meters using the long-range mode. We suspect that results from 1 to 3 meters obtained from the near-mode will be less accurate than the ones obtained from the long-range mode. However, it is yet to be researched further. Our experiment shows that Kinect's depth camera does not operate from 3.5 to 4 meters when the subject stands 1 m to the either side from the device position. Figure 6.1 depicts suggested *Kinect for Windows* positions to support coverage from 0.5 to 3.5 meters with the use of 5 Kinect devices positioned 2 meters apart from each other.

Chapter 7: Conclusions

In this thesis, we have explored the pointing performance (speed) and accuracy of pointing using low-cost off-the-shelf equipment and compared it to a conventional computer mouse and a high-end tracking device (i.e. Vicon tracker). We have built different systems using off-the-shelf equipment, tested the feasibility of pointing using our system in a laboratory and classroom settings, and researched ways to maximize pointing coverage using multiple tracking devices.

7.1 Research Contributions

The following is the summary of research contributions outlined in this thesis:

1. We have built different systems using off-the-shelf consumer grade equipment, outlined advantages and disadvantages of each of them and indicated ways for future research.
2. We have explored the pointing performance and accuracy of pointing using low-cost off-the-shelf equipment and compared it to a conventional computer mouse and high-end tracking devices (i.e. Vicon tracker). Through the experiment we have concluded that two-part models (such as Welford's Law) of pointing performance were not substantially better than that one-part models (such as Fitts's Law) at accurately predicting pointing performance for low-cost, high-end devices and a computer mouse.
3. We performed a test to determine how our system is suitable in a real classroom setting in terms of accuracy and pointing performance. Results showed practical pointing performance and at least 95% accuracy during pointing. In addition, we also explored the possibility of using multiple camera trackers in a large classroom to maximize tracked area during pointing. We present results and indicate ways for future in classroom studies using multiple camera trackers.

7.2 Future Work

In Chapter 3 we provide multiple solutions to reduce or fully eliminate the inertial controllers' yaw drift problem and we showed advantages and disadvantages of each solution during pilot studies. In Chapter 5 we run a study that uses manual calibration technique to fix the yaw drift problem in a laboratory setting but it would be worthwhile to investigate if other solutions can significantly reduce or fully eliminate yaw drift. We investigate the possibility of using multiple Kinect devices, connected sequentially two meters apart from each other, in a classroom to maximize coverage area for tracking.

Future work should investigate the feasibility of pointing in classroom settings while using multiple Kinect devices connected sequentially. It is also yet unknown to us how the underlying implementation for the system will look like that can handle multiple tracking devices connected together.

The implementation of our pointing system includes basic support for painting and interacting with PowerPoint slides. The addition of gestures to our implementation should be performed and further analyzed.

In many applications with the large screen interaction, a user has to interact with a screen and the audience at the same time. It is yet to be researched how sophisticated interaction with large displays, that includes pointing, drawing and gesture control, can affect user's ability to continue interacting with the audience in the same way. In addition, using the same equipment, the audience should be able to interact with the presenter, as well as with the large screen display. For example, during classroom lectures, students should be able to give feedback to the instructor to notify if they are falling behind or do not understand certain parts of the lecture. Moreover, students should also be able to point at the large screen and be able to navigate

through slides to show part of the lecture that they are unable to understand. There has been research suggesting the possibility of presenter – large group collaboration for peer instructions (Reinhardt et al., 2012). It is therefore important to understand how one improvement in the interaction (i.e. user with a large screen display) could affect the performance of the other interaction (i.e. user with a large audience).

References

Argelaguet, F. and Andujar, C. (2009). Visual feedback techniques for virtual pointing on stereoscopic displays. *VRST '09 Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*, pp. 163--170.

Ascension-tech.com (2013). *Ascension Technology Corporation*. [online] Retrieved from: http://www.ascension-tech.com/realtime/RTflockofBIRDS.php [Accessed: 21 Oct 2013].

Bolt, R. (1980). “Put-that-there”: Voice and gesture at the graphics interface. *SIGGRAPH '80 Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, 14 (3), pp. 262-270.

Bolton, J., Wang, P., Kim, K. and Vertegaal, R. (2012). BodiPod: interacting with 3d human anatomy via a 360° cylindrical display. *CHI EA '12 CHI '12 Extended Abstracts on Human Factors in Computing Systems*, pp. 1039--1042.

Bowman, D. and Hodges, L. (1997). An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. *I3D '97 Proceedings of the 1997 symposium on Interactive 3D graphics*, pp. 35--.

Burns, C. (2013). *New Kinect for Windows borrows Xbox One's updated sensor*. [online] Retrieved from: http://www.slashgear.com/new-kinect-for-windows-borrows-xbox-ones-updated-sensor-23283355/ [Accessed: 12 Dec 2012].

Casiez, G., Vogel, D., Balakrishnan, R. and Cockburn, A. (2008). The impact of control-display gain on user performance in pointing tasks. *Human--Computer Interaction*, 23 (3), pp. 215--250.

Cavens, D., Vogt, F., Fels, S. and Meitner, M. (2002). Interacting with the big screen: pointers to ponder.*CHI EA '02 CHI '02 Extended Abstracts on Human Factors in Computing Systems*, pp. 678--679.

Code.google.com (2013). *moveframework - Framework for Playstation Move on PC (Windows only, C++ and C# support) - Google Project Hosting*. [online] Retrieved from: https://code.google.com/p/moveframework/ [Accessed: 14 Dec 2012].

Davis, J. and Chen, X. (2002). Lumipoint: Multi-user laser-based interaction on large tiled displays. *Displays*, 23 (5), pp. 205--211.

Douglas, S., Kirkpatrick, A. and Mackenzie, I. (1999). Testing pointing device performance and user assessment with the ISO 9241, Part 9 standard. *CHI '99 Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 215--222.

Fitt, P. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47 (6), pp. 381--391.

Genest, A., Gutwin, C., Tang, A., Kalyn, M. and Ivkovic, Z. (2013). KinectArms: a toolkit for capturing and displaying arm embodiments in distributed tabletop groupware. *CSCW '13 Proceedings of the 2013 conference on Computer supported cooperative work*, pp. 157--166.

Jota, R., Nacenta, M., Jorge, J., Carpendale, S. and Greenberg, S. (2010). A comparison of ray pointing techniques for very large displays. *GI '10 Proceedings of Graphics Interface*, pp. 269--276.

König, W., Gerken, J., Dierdorf, S. and Reiterer, H. (2009). Adaptive pointing: implicit gain adaptation for absolute pointing devices. *CHI EA '09 CHI '09 Extended Abstracts on Human Factors in Computing Systems*, pp. 4171--4176.

Kessler, G., Hodges, L. and Walker, N. (1995). Evaluation of the CyberGlove as a whole-hand input device. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 2 (4), pp. 263--283.

Kickstarter (2013). *NUIA eyeCharm: Kinect® to eye tracking*. [online] Retrieved from: http://www.kickstarter.com/projects/4tiitoo/nuia-eyecharm-kinect-to-eye-tracking [Accessed: 13 Apr 2013].

Kim, K., Bolton, J., Girouard, A., Cooperstock, J. and Vertegaal, R. (2012). Telehuman: effects of 3d perspective on gaze and pose estimation with a life-size cylindrical telepresence pod. *CHI '12 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2531--2540.

Kinect SDK (2013). *Kinect SDK*. [online] Retrieved from: http://www.microsoft.com/en-us/kinectforwindows/develop/developer-downloads.aspx [Accessed: 12 Apr 2013].

Kinect Sensor Placement (2013). *Kinect Sensor Placement*. [online] Retrieved from: http://support.xbox.com/en-US/xbox-360/kinect/sensor-placement [Accessed: 12 Apr 2013].

Lanir, J., Booth, K. and Tang, A. (2008). MultiPresenter: a presentation system for (very) large display surfaces. *MM '08 Proceedings of the 16th ACM international conference on Multimedia*, pp. 519--528.

Leap Motion (2013). *Leap Motion*. [online] Retrieved from: https://www.leapmotion.com/ [Accessed: 10 Jan 2012].

Lee, S., Seo, J., Kim, G. and Park, C. (2003). Evaluation of pointing techniques for ray casting selection in virtual environments. *Third International Conference on Virtual Reality and Its Application in Industry*, pp. 38--44.

Mackenzie, I. (1992). Fitts' law as a research and design tool in human-computer interaction. *Human-computer interaction*, 7 (1), pp. 91--139.

Mackenzie, I. and Oniszczak, A. (1998). A comparison of three selection techniques for touchpads. *CHI '98 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 336--343.

Mackenzie, I., Kauppinen, T. and Silfverberg, M. (2001). Accuracy measures for evaluating computer pointing devices. *CHI '01 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 9--16.

Madgwick, S., Harrison, A. and Vaidyanathan, R. (2011). Estimation of IMU and MARG orientation using a gradient descent algorithm. *Rehabilitation Robotics (ICORR) IEEE International Conference*, pp. 1--7.

Madgwick, S., Harrison, A. and Vaidyanathan, R. (2010). An Efficient Orientation Filter for InertialMeasurement Units (IMUs) and MagneticAngular Rate and Gravity (MARG) SensorArrays.*Department of Mechanical Engineering*.

Microsoft Store (2013). *Microsoft Store*. [online] Retrieved from: http://www.microsoftstore.com/store/msusa/en_US/pdp/productID.253169000 [Accessed: 12 Apr 2013].

Msdn.microsoft.com (2013). *MSDN – the Microsoft Developer Network*. [online] Retrieved from: http://msdn.microsoft.com/ [Accessed: 12 Apr 2013].

Myers, B., Peck, C., Nichols, J., Kong, D. and Miller, R. (2001). Interacting at a distance using semantic snarfing. *UbiComp '01 Proceedings of the 3rd international conference on Ubiquitous Computing*, pp. 305--314.

Nacenta, M., Pinelle, D., Stuckel, D. and Gutwin, C. (2007). The effects of interaction technique on coordination in tabletop groupware. *GI '07 Proceedings of Graphics Interface*, pp. 191--198.

Natapov, D., Castellucci, S. and Mackenzie, I. (2009). ISO 9241-9 evaluation of video game controllers.*GI '09 Proceedings of Graphics Interface 2009*, pp. 223--230.

Newcombe, R., Davison, A., Izadi, S., Kohli, P., Hilliges, O., Shotton, J., Molyneaux, D., Hodges, S., Kim, D. and Fitzgibbon, A. (2011). KinectFusion: Real-time dense surface mapping and tracking.*IEEE ISMAR*, pp. 127--136.

Oh, J. and Stuerzlinger, W. (2002). Laser pointers as collaborative pointing devices. 2002 pp. 141--149.

Olsen Jr, D. and Nielsen, T. (2001). Laser pointer interaction. *CHI '01 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 17--22.

Parker, K., Mandryk, R. and Inkpen, K. (2005). Tractor- Beam: seamless integration of local and remote pointing for tabletop displays.. *GI '05 Proceedings of Graphics Interface*, pp. 33-40. [Accessed: 20 Sep 2013].

Pavlovych, A. and Stuerzlinger, W. (2009). The tradeoff between spatial jitter and latency in pointing tasks.*EICS '09 Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pp. 187--196.

Polhemus.com (2013).*Polhemus: LIBERTY LATUS Motion Tracker--Wireless, 6 DOF Motion Tracking System*. [online] Retrieved from: http://www.polhemus.com/?page=Motion_Liberty_Latus [Accessed: 20 Oct 2013].

Reinhardt, W., Sievers, M., Magenheim, J., Kundish, D., Herrmann, P., Beutner, M. and Zoyke, A. (2012). PINGO: Peer Instruction for Very Large Groups.. *Lecture Notes in Computer Science*, 7563 pp. 507-512.

Sears, A. and Shneiderman, B. (1991). High precision touchscreens: design strategies and comparisons with a mouse. *International Journal of Man-Machine Studies*, 34 (4), pp. 593--613.

Shoemaker, G., Tang, A. and Booth, K. (2007). Shadow reaching: a new perspective on interaction for large displays.*UIST '07 Proceedings of the 20th annual ACM symposium on User interface software and technology*, pp. 53--56.

Shoemaker, G., Tsukitani, T., Kitamura, Y. and Booth, K. (2012). Two-Part Models Capture the Impact of Gain on Pointing Performance. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 19 (4), p. 28.

Soukoreff, R. and Mackenzie, I. (2004). Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in HCI. *International Journal of Human-Computer Studies*, 61 (6), pp. 751--789.

Teather, R. and Stuerzlinger, W. (2008). Assessing the effects of orientation and device on (constrained) 3D movement techniques. *3D User Interfaces*, pp. 43--50.

Tse, E., Hancock, M. and Greenberg, S. (2007). Speech-filtered bubble ray: improving target acquisition on display walls. *ICMI '07 Proceedings of the 9th international conference on Multimodal interfaces*, pp. 307--314.

Vogel, D. and Balakrishnan, R. (2005). Distant freehand pointing and clicking on very large, high resolution displays. *UIST '05 Proceedings of the 18th annual ACM symposium on User interface software and technology*, pp. 33--42.

Vogt, F., Wong, J., Fels, S. and Cavens, D. (2003). Tracking multiple laser pointers for large screen interaction. *Ext. Abstracts UIST*, pp. 95--96.

Vogt, F., Wong, J., Po, B., Argue, R., Fels, S. and Booth, K. (2004). Exploring collaboration with group pointer interaction. *Proc. Computer Graphics International*, pp. 636--639.

Voida, S., Podlaseck, M., Kjeldsen, R. and Pinhanez, C. (2005). A study on the manipulation of 2D objects in a projector/camera-based augmented reality environment. *CHI '05 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 611--620.

Von Wiegand, T. and Pfautz, J. (2001). The "Hand of Death," a distributed haptic interface for signalling exposure to virtual enemy fire. *Presence-Connect*.

Ware, C. and Lowther, K. (1997). Selection using a one-eyed cursor in a fish tank VR environment. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 4 (4), pp. 309--322.

Welford, A. (1968). Fundamentals of skill.. *Methuen*.

Wii Remote mouse Application (2013). *Wii Remote mouse Application*. [online] Retrieved from: http://home.exetel.com.au/amurgshere/wiimouse.phtml [Accessed: 12 Feb 2012].

Wiimotelib.codeplex.com (2013). *A library for using a Nintendo Wii Remote (Wiimote) from .NET*. [online] Retrieved from: http://wiimotelib.codeplex.com/ [Accessed: 15 Nov 2012].

Wingrave, C., Tintner, R., Walker, B., Bowman, D. and Hodges, L. (2005). Exploring individual differences in raybased selection: Strategies and traits. *Virtual Reality, 2005. Proceedings. VR 2005. IEEE*, pp. 163--170.

Wobbrock, J., Cutrell, E., Harada, S. and Mackenzie, I. (2008). An error model for pointing based on Fitts' law. pp. 1613--1622.

Zhai, S., Kong, J. and Ren, X. (2004). Speed--accuracy tradeoff in Fitts' law tasks—on the equivalency of actual and nominal pointing precision. *International journal of human-computer studies*, 61 (6), pp. 823--856..

Appendices

Appendix A

Pointing experimental data and questionnaire

Table A.1: Raw performance results from the study described in Chapter 4. Movement time and error rate were calculated by averaging over all subjects for each combination D, A and W.

Independent			Observed		
Device	A(cm)	W (cm)	MT (ms)	W_e(cm)	Error(%)
Mouse	60	15	876.26	17.223	0.0719
Mouse	60	20	816.19	18.138	0.0227
Mouse	60	25	793.44	19.341	0.0075
Mouse	60	30	724.05	30	0
Mouse	120	15	1041.6	16.270	0.0568
Mouse	120	20	1033.8	17.607	0.0189
Mouse	120	25	968.89	21.266	0.0151
Mouse	120	30	909.19	24.485	0.0113
Mouse	180	15	1218.1	16.756	0.0643
Mouse	180	20	1072.5	21.357	0.0530

Continued on next page

Table A.1 – continued from previous page

Independent			Observed		
Device	A (cm)	W(cm)	MT (ms)	W_e (cm)	Error(%)
Mouse	180	25	1030.8	23.280	0.0265
Mouse	180	30	1013.5	24.485	0.0113
Mouse	240	15	1335.5	17.896	0.0833
Mouse	240	20	1294.1	19.075	0.0303
Mouse	240	25	1371.3	19.341	0.0075
Mouse	240	30	1137.0	30	0
Vicon	60	15	1115.1	14.926	0.0378
Vicon	60	20	1079.4	18.138	0.0227
Vicon	60	25	981.79	25	0.0026
Vicon	60	30	902.35	23.209	0.02075
Vicon	120	15	1390.2	14.306	0.0303
Vicon	120	20	1240.5	19.075	0.0303
Vicon	120	25	1208.4	25	0.0022
Vicon	120	15	1061.9	30	0.0007
Vicon	180	15	1504.8	16.270	0.0568
Vicon	180	20	1372.1	17.012	0.0151
Vicon	180	25	1246.1	19.341	0.0075
Vicon	180	30	1183.9	24.485	0.0113

Continued on next page

Table A.1 – continued from previous page

Independent			Observed		
Device	A (cm)	W(cm)	MT (ms)	W_e (cm)	Error(%)
Vicon	240	15	1691.7	15.491	0.0454
Vicon	240	20	1515.8	18.624	0.0265
Vicon	240	25	1471.7	20.404	0.0113
Vicon	240	30	1299.8	30	0.0045
Hybrid Kinect	60	15	1183.3	14.926	0.0378
Hybrid Kinect	60	20	1033.8	17.012	0.0151
Hybrid Kinect	60	25	976.87	25	0.0037
Hybrid Kinect	60	30	815.31	24.485	0.0113
Hybrid Kinect	120	15	1454.4	14.926	0.0378
Hybrid Kinect	120	20	1276.9	17.012	0.0151
Hybrid Kinect	120	25	1261.5	25	0.0037
Hybrid Kinect	120	30	1172.8	30	0
Hybrid Kinect	180	15	1584.8	15.491	0.0454
Hybrid Kinect	180	20	1403.3	17.607	0.0189
Hybrid Kinect	180	25	1352.9	25	0.0037
Hybrid Kinect	180	30	1277.4	23.209	0.0075

Continued on next page

Table A.1 – continued from previous page

Independent			Observed		
Device	A (cm)	W(cm)	MT (ms)	W_e (cm)	Error(%)
Hybrid Kinect	240	15	1794.8	17.223	0.0719
Hybrid Kinect	240	20	1596.3	19.075	0.0303
Hybrid Kinect	240	25	1567.6	24.374	0.0340
Hybrid Kinect	240	30	1427.5	30	0.0037

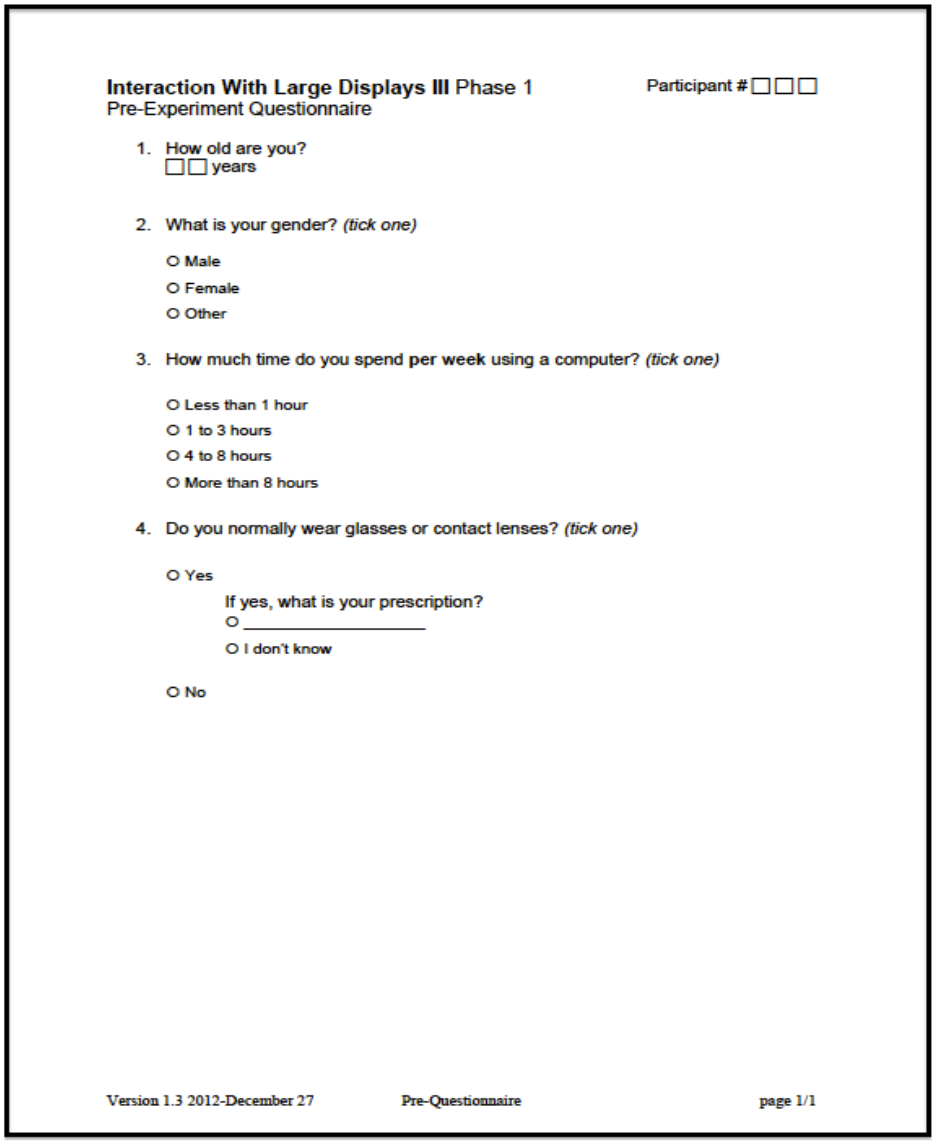
Table A.2: ANOVA results for movement time and error rate.

Factor	Measure	Corr.	df	F	*p*	Partial η2
Dt	MT	-	2	638.4	0	0.708
		GG	1.995	638.4	0	0.708
	Error Rate	-	2	3.312	0.037	0.012
		GG	1.883	3.312	0.040	0.012
Width	MT	-	3	486.4	0	0.649
		GG	2.721	486.4	0	0.649
	Error Rate	-	3	56.19	0	0.176
		GG	2.331	56.19	0	0.176
A	MT	-	3	1686.	0	0.865
		GG	2.930	1686.	0	0.865
	Error Rate	-	3	2.221	0.084	0.008
		GG	2.953	2.221	0.085	0.008
Dt * Width	MT	-	6	22.16	0	0.077
		GG	4.978	22.16	0	0.077
	Error Rate	-	6	2.211	0.039	0.008
		GG	4.567	2.211	0.056	0.008
Dt * A	MT	-	6	10.57	0	0.038
		GG	5.655	10.57	0	0.038
	Error Rate	-	6	1.760	0.103	0.006
		GG	5.686	1.760	0.107	0.006
Width * A	MT	-	9	7.508	0	0.027
		GG	7.455	7.508	0	0.027
	Error Rate	-	9	1.000	0.436	0.003
		GG	6.706	1.000	0.427	0.003
Dt * Width * A	MT	-	18	3.174	0	0.011
		GG	14.36	3.174	0	0.011
	Error Rate	-	18	1.014	0.439	0.003
		GG	11.93	1.014	0.432	0.003

GG = Greenhouse-Geisser;

Table A.3: Pairwise comparison table showing effect of device type (D) on movement time (MT).

Measure	Device1	Device2	Mean Difference (Device1-Device2)	Std. Error	*p*	Lower Bound	Upper Bound
MT	Mouse	Vicon	-226.8	8.4579	0	-247.2	-206.4
		Hybrid Kinect	-287.4	8.6707	0	-308.2	-266.5
	Vicon	Mouse	226.82	8.4579	0	206.44	247.20
		Hybrid Kinect	-60.57	8.3067	0	-80.59	-40.56
	Hybrid Kinect	Mouse	287.40	8.6707	0	266.50	308.29
		Vicon	60.579	8.3067	0	40.565	80.593



Interaction With Large Displays III Phase 1 Participant #☐☐☐
Pre-Experiment Questionnaire

1. How old are you?
   ☐☐ years

2. What is your gender? *(tick one)*

   O Male
   O Female
   O Other

3. How much time do you spend **per week** using a computer? *(tick one)*

   O Less than 1 hour
   O 1 to 3 hours
   O 4 to 8 hours
   O More than 8 hours

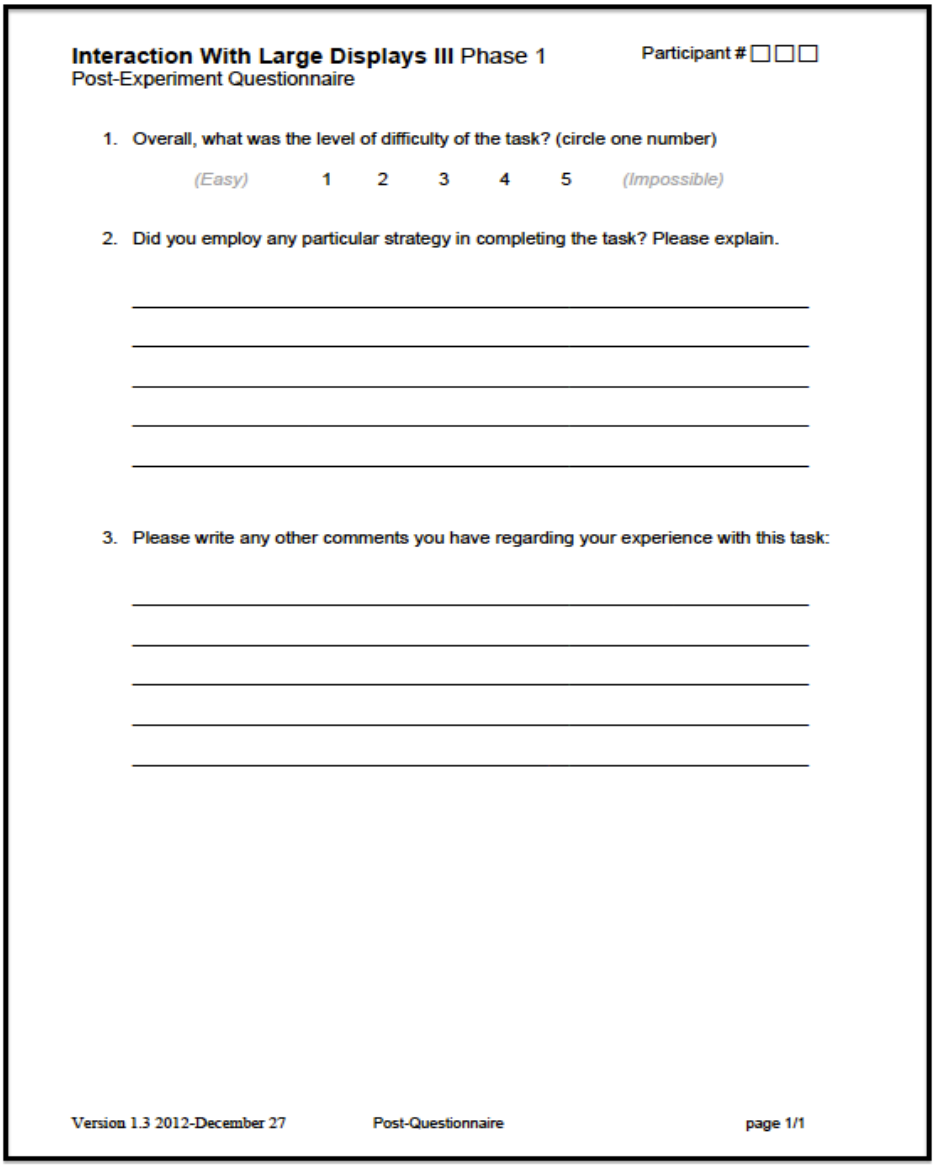
4. Do you normally wear glasses or contact lenses? *(tick one)*

   O Yes
      If yes, what is your prescription?
      O _____
      O I don't know

   O No

Version 1.3 2012-December 27 Pre-Questionnaire page 1/1

Figure A.1: Pre-experiment questionnaire for the pointing study.



Interaction With Large Displays III Phase 1
Post-Experiment Questionnaire

Participant # ☐☐☐

1. Overall, what was the level of difficulty of the task? (circle one number)

   (Easy) 1 2 3 4 5 *(Impossible)*

2. Did you employ any particular strategy in completing the task? Please explain.

   _____
   _____
   _____
   _____
   _____

3. Please write any other comments you have regarding your experience with this task:

   _____
   _____
   _____
   _____
   _____

Version 1.3 2012-December 27 Post-Questionnaire page 1/1

Figure A.2: Post-experiment questionnaire for the pointing study.

Appendix B

Table B.1: ANOVA results for movement time and error rate.

Factor	Measure	Corr.	df	F	*p*	Partial η2
Device	MT	-	2	140.1	0	0.683
		GG	1.435	140.1	0	0.683
	ErrorRate	-	2	1.720	0.183	0.025
		GG	1.682	1.720	0.188	0.025
Width	MT	-	3	230.1	0	0.779
		GG	1.283	230.1	0	0.779
	ErrorRate	-	3	7.283	0.000	0.100
		GG	2.132	7.283	0.000	0.100
A	MT	-	3	557.4	0	0.895
		GG	1.691	557.4	0	0.895
	ErrorRate	-	3	1.011	0.388	0.015
		GG	2.714	1.011	0.383	0.015
Device * Width	MT	-	6	20.79	0	0.242
		GG	3.449	20.79	0	0.242
	ErrorRate	-	6	0.975	0.441	0.014
		GG	4.308	0.975	0.425	0.014
Device * A	MT	-	6	10.16	0	0.135
		GG	3.403	10.16	0	0.135
	ErrorRate	-	6	1.305	0.253	0.019
		GG	5.008	1.305	0.261	0.019
Width * A	MT	-	9	6.487	0	0.090
		GG	4.388	6.487	0	0.090
	ErrorRate	-	9	1.487	0.148	0.022
		GG	5.406	1.487	0.188	0.022
Device * Width * A	MT	-	18	4.208	0	0.060
		GG	4.318	4.208	0.001	0.060
	ErrorRate	-	18	0.664	0.848	0.010
		GG	10.29	0.664	0.762	0.010

Table B.2: Performance time (MT) comparison table for device type (D).

Measure	Device1	Device2	Mean Difference (Device1-Device2)	Std. Error	*p*	Lower Bound	Upper Bound
MT	Mouse	Vicon	-325.1	28.577	0	-395.3	-254.9
		Kinect	-327.0	17.220	0	-369.3	-284.6
	Vicon	Mouse	325.16	28.577	0	254.93	395.38
		Kinect	-1.850	20.099	1	-51.24	47.544
	Kinect	Mouse	327.01	17.220	0	284.69	369.32
		Vicon	1.8503	20.099	1	-47.54	51.244

Table B.3: Performance time (MT) comparison table for target width (W).

Measure	Width1 (cm)	Width2 (cm)	Mean Difference (Width1-Width2)	Std. Error	*p*	Lower Bound	Upper Bound
MT	15	20	125.55	8.4119	0	102.66	148.44
		25	213.78	14.905	0	173.21	254.34
		30	277.36	16.207	0	233.25	321.46
	20	15	-125.5	8.4119	0	-148.4	-102.6
		25	88.230	9.0997	0	63.465	112.99
		30	151.80	9.5210	0	125.89	177.71
	25	15	-213.7	14.905	0	-254.3	-173.2
		20	-88.23	9.0997	0	-112.9	-63.46
		30	63.577	4.8743	0	50.312	76.841
	30	15	-277.3	16.207	0	-321.4	-233.2
		20	-151.8	9.5210	0	-177.7	-125.8
		25	-63.57	4.8743	0	-76.84	-50.31

Table B.4: Performance time (MT) comparison table for movement amplitude (A).

Measure	A1 (cm)	A2 (cm)	Mean Difference (A1 - A2)	Std. Error	*p*	Lower Bound	Upper Bound
MT	60	120	-195.1	6.7217	0	-213.4	-176.8
		180	-310.9	11.519	0	-342.3	-279.6
		240	-481.9	17.289	0	-528.9	-434.8
	120	60	195.12	6.7217	0	176.83	213.41
		180	-115.8	8.6157	0	-139.3	-92.41
		240	-286.8	13.795	0	-324.3	-249.2
	180	60	310.98	11.519	0	279.63	342.33
		120	115.86	8.6157	0	92.414	139.30
		240	-170.9	11.879	0	-203.2	-138.6
	240	60	481.92	17.289	0	434.87	528.98
		120	286.80	13.795	0	249.25	324.34
		180	170.94	11.879	0	138.61	203.27