EVALUATING 3D POINTING TECHNIQUES

ROBERT JOHN TEATHER

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

GRADUATE PROGRAMME IN COMPUTER SCIENCE & ENGINEERING
YORK UNIVERSITY
TORONTO, ONTARIO

MAY 2013

EVALUATING 3D POINTING TECHNIQUES

by **Robert J. Teather**

A thesis submitted to the Faculty of Graduate Studies of York University in partial fulfilment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

ABSTRACT

This dissertation investigates various issues related to the empirical evaluation of 3D pointing interfaces. In this context, the term "3D pointing" is appropriated from analogous 2D pointing literature to refer to 3D point selection tasks, i.e., specifying a target in three-dimensional space. Such pointing interfaces are required for interaction with virtual 3D environments, e.g., in computer games and virtual reality. Researchers have developed and empirically evaluated many such techniques. Yet, several technical issues and human factors complicate evaluation. Moreover, results tend not to be directly comparable between experiments, as these experiments usually use different methodologies and measures.

Based on well-established methods for comparing 2D pointing interfaces this dissertation investigates different aspects of 3D pointing. The main objective of this work is to establish methods for the direct and fair comparisons between 2D and 3D pointing interfaces. This dissertation proposes and then validates an experimental paradigm for evaluating 3D interaction techniques that rely on pointing. It also investigates some technical considerations such as latency and device noise. Results show that the mouse outperforms (between 10% and 60%) other 3D input techniques in all tested conditions. Moreover, a monoscopic cursor tends to perform better than a stereo cursor when using stereo display, by as much as 30% for deep targets. Results suggest that common 3D pointing techniques are best modelled by first projecting target parameters (i.e., distance and size) to the screen plane.

Acknowledgements

There are many people I'd like to thank for helping to make this dissertation possible. First, my supervisor, Wolfgang Stuerzlinger has shaped my academic development since I began my graduate studies. I owe a great deal to him, not only for providing insight into the work itself, but for keeping me focused and making himself available to provide advice on all matters throughout the course of my studies. I'd also like to thank my supervisory committee members, Rob Allison and Scott MacKenzie. Their expertise in various aspects of this work has been immensely helpful. In particular, Rob has always been available to correct my understanding of various aspects of visual perception, and Scott has instilled in me a great appreciation of experimental methods.

I would also like to thank the examination committee members, Anne Moore, Suprakash Datta, and Doug Bowman for taking the time out of their busy schedules to review the dissertation, provide helpful suggestions on how to improve it, and attend the exam itself. Dr. Bowman's suggestions have especially been a great help in the final "polishing" phase of revising the document.

A number of colleagues and fellow students have also contributed to this work. In no particular order, I'd like to thank Steven Castellucci, Andriy Pavlovych, Ahmed Arif, Shumon Zaman, Andrew Roth, Rachel Hirsch, and Dusty Phillips for insightful conversations over the past few years. Some of these conversations were even about research!

I have also been fortunate to have received external funding throughout most of my studies, and would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Ontario Graduate Scholarship (OGS) program for providing scholarships. I would also like to thank York University for providing me the Susan Mann Dissertation Scholarship in my final year of studies.

Finally, I'd like to acknowledge the support of my family. Thanks to my parents, John and Linda Teather for encouraging words throughout the process, even though I know you don't really "get it"! Thanks also to my wife Vicky, for your constant encouragement and helping to keep things in perspective. Thanks for putting up with me through both the bad times and the good.

All of these people have shaped the work in this dissertation in some way.

Dedicated to Bronwynne
you've been a constant source of motivation since you
came into our lives on October 14, 2012

Dissemination of this Dissertation

The following chapters of this dissertation have been previously published as peer-reviewed papers:

Chapter 3: **R. J. Teather**, A. Pavlovych, W. Stuerzlinger, I. S. MacKenzie. (2009). "Effects of tracking technology, latency, and spatial jitter on object movement", *IEEE Symposium on 3D User Interfaces – 3DUI '09*, IBSN 978-142443965-2, pp. 43-50, March 2009.

Chapter 4: **R. J. Teather**, W. Stuerzlinger. (2011). "Pointing at 3D targets in a stereo head-tracked virtual environment", *IEEE Symposium on 3D User Interfaces – 3DUI '11*, ISBN 978-1-4577-0062-0, pp. 87-94, March 2011.

Chapter 5: **R. J. Teather**, W. Stuerzlinger. (2013). "Pointing at 3D target projections with one-eyed and stereo cursors", *ACM Conference on Human Factors in Computing Systems – CHI '13,* ISBN 978-145031899-0, pp. 159-168, April 2013.

Additionally, the following extended abstracts and posters are based on this work:

R. J. Teather, W. Stuerzlinger. (2012). "A system for evaluating 3D pointing techniques", Demo at the *ACM Virtual Reality Software and Technology*, Dec. 2012.

R. J. Teather, W. Stuerzlinger. (2012). "Cursors for 3D pointing", Presentation at the *ACM CHI 2012 Workshop the 3rd Dimension of CHI*, May 2012.

R. J. Teather, W. Stuerzlinger. (2012). "Investigating one-eyed and stereo cursors for 3D pointing tasks", Poster at the *IEEE Symposium on 3D User Interfaces*, pp. 167-168, March 2012.

R. J. Teather, W. Stuerzlinger. (2010). "Target pointing in 3D user interfaces", *Poster at Graphics Interface*, June 2010.

R. J. Teather, A. Pavlovych, W. Stuerzlinger. (2009). "Effects of latency and spatial jitter on 2D and 3D pointing", *Poster at the IEEE Virtual Reality Conference*, ISBN 978-142443943-0, 229-230, March 2009.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

One might speculate that reaching, grabbing and manipulating virtual objects would enable their easy and effective manipulation in virtual reality (VR). After all, this is how we interact with real objects on a regular basis. In practice, this assumption ignores numerous technical constraints and human physiological limitations. Some of these issues include the presence or absence of stereoscopic vision, a head-coupled display, or a supporting surface on which to operate an input device (i.e., passive haptic feedback). There are also technical concerns, such as latency and jitter (tracker noise). In real-world object manipulation, we are used to having various forms of immediate feedback, especially visual and tactile. Moreover, there is no latency in seeing or touching an object being manipulated.

In practice, common 3D input devices, such as trackers, wands, and gloves fare poorly in comparisons to the standard computer mouse in conceptually equivalent direct manipulation tasks, such as moving an icon. The mouse has been shown to be a good alternative to 3D input devices for constrained 3D object movement tasks in certain tasks [83-85]. This is motivated by the low latency and high precision of the mouse and builds on innovative software techniques to map 2D input to 3D operations. However, while a mouse is suitable for certain types of environments and tasks, there are situations that necessitate the use of 3D input devices. For example, in immersive virtual or

augmented reality systems, the user is often standing or even walking. In these cases, the mouse is unsuitable, as it requires a (non-portable) tracking surface. In such systems a hand-mounted tracker or 3D wand device is likely a more appropriate input device and may help enhance immersion as well. To further complicate matters, many interaction techniques will work with either class of device. For example, ray casting works with either a mouse or a 3D tracker.

On the other hand, there are situations where a system designer could use either a 3D tracker *or* a mouse as the system input device. Fish-tank virtual reality systems are a good example of this. The user is typically sitting and using a medium-sized display on a table. Here a mouse is a good choice for input. Alternatively, a 3D tracker could be used to enhance immersion. However, such choices should not be made without understanding the performance trade-off between the mouse and tracker. Similarly, although 3D selection/manipulation techniques are often informed by the choice of device, certain techniques, such as ray casting, work with both trackers and the mouse. Hence it is desirable to be able to directly compare performance between the two devices, or between different 3D interaction techniques with the same device.

It is difficult to directly compare mouse-based and tracker-based object manipulation interfaces. Few attempts have been made to do so, and no commonly accepted evaluation method exists. It is even difficult to generalize results between studies comparing only tracker-based techniques, as experimental designs vary from study to study. Consequently, it is hard to formally quantify the benefits and trade-offs of

certain techniques or devices, leaving practitioners only with general guidelines. In the absence of such quantifiable benefits, 3D input system design becomes challenging. Fitts' law has been widely used in the evaluation of 2D user interfaces based on the notion of pointing at targets, as in direct manipulation interfaces. This methodology may be similarly useful in the evaluation of 3D user interfaces, but little work has been done to investigate this hypothesis.

1.1 Motivation/Contributions

The primary objective of this dissertation is to establish methodology for the *fair and direct* comparison of 2D and 3D input devices and techniques. As discussed above, there are situations where system designers can choose between input devices. To help with this decision, one can empirically evaluate performance (e.g., in pointing tasks) using both devices. However, current methodologies would likely favour one device or technique and consequently, artificially improve its performance metrics relative to the other.

Hence, this dissertation considers only pointing tasks – fundamental motions that can be considered the "building blocks" of more complex interaction tasks, such as object movement or manipulation. Pointing tasks largely eliminate confounding factors such as user strategy, as different participants may deploy various strategies for completing a given manipulation task. A hypothesis of this work is that pointing tasks will elicit real differences between evaluated pointing techniques, and will provide a more fair comparison between 2D and 3D techniques.

A second objective of this work is to identify points of direct comparison between 2D and 3D techniques. There are two options to achieve this: either perform a 2D task with a 3D tracker input device, or perform a 3D task with a mouse. Either option requires suitable mappings of input to 2- or 3-dimensional operations. Both options are explored in this dissertation, as both are expected to be beneficial in establishing direct comparison methods.

Another objective of this work is to compare pointing performance due to commonly observed differences between the mouse and 3D tracking systems using the newly developed methodologies mentioned above. Issues such as latency, jitter, and stereo visualization are investigated. While many of these factors have been previously investigated, most have only been considered in isolation from one another, or outside the context of 2D and 3D task comparison. These individual experiments are also intended to validate the methodology established by this work.

Overall, the work presented in this dissertation is intended to contribute new tools for evaluating 3D user interfaces for designers and researchers alike. The methodologies proposed are based on an international standard of pointing device evaluation. Such standardized methodology has had a great impact on the 2D user interface domain, and may similarly benefit the 3D user interface community.

1.2 Outline

The outline of this dissertation is as follows. Chapter 2 provides an overview of literature related to the topics of 3D selection, manipulation, and pointing. Topics such as stereo

viewing are also discussed, as this is directly relevant to the research presented herein. The next three chapters detail series of studies investigating specific issues in 3D pointing/selection tasks. Chapter 3 looks at the interplay between latency and jitter common to 3D tracking technologies. Chapter 4 proposes to extend 2D pointing evaluation methods to evaluate 3D pointing tasks. It also investigates the effects of tactile feedback. Chapter 5 looks at cursor display properties, and the effect of perspective scaling on so-called "2.5" pointing tasks: situations in which 3D pointing tasks are effectively the same as 2D pointing tasks, i.e., screen-plane projected pointing. Finally, Chapter 6 concludes the dissertation with overall discussion of the results, highlighting the major contributions of this work. It finally proposes areas for future work.

Chapter 2
Background

2.1 3D Selection and Manipulation

This section addresses two central tasks for three-dimensional user interfaces, namely manipulation and selection. Manipulating objects in 3D space is a six degree of freedom (6DOF) task. There are three independent axes of movement and three axes of rotation for every object. Manipulation refers to the action of specifying the 3D pose – i.e., both position *and* orientation – of an object. According to Bowman's taxonomy [13], an object must be selected prior to manipulation. Selection, in this context, refers to the action of specifying a target object for subsequent operations. Subsequent operations are typically manipulation, but can also include altering properties, such as the colour or texture, of the object. Note that although ultimately 6DOF tasks, selection and manipulation can both be performed using 3D input devices, or, given suitable input mappings, 2D devices like the mouse. The differences between 3D and 2D selection/manipulation techniques are discussed below in Sections 2.1.1 and 2.1.6. Note that this dissertation documents specifically rigid-body manipulation. Hence operations such as object/surface deformation and object cutting/merging are beyond the scope of this dissertation.

Navigation and system control are two other primary tasks commonly required in VR systems [14]. These tasks are beyond the scope of this dissertation and are not discussed further. Finally, it is worth distinguishing between exocentric and egocentric

selection and manipulation techniques. Exocentric techniques, such as worlds-in-miniature [79], give the user a small overview of the environment and allow indirect interaction with objects via this miniature version. However, these techniques often suffer from precision problems due to the minification effect. Egocentric techniques are often similar to real-world object manipulation, requiring direct interaction with objects, sometimes even using the user's real hand.

2.1.1 6DOF Input Devices

A great deal of 3D user interface research focuses on 3D manipulation tasks using 3D input devices such as 6DOF trackers, wands, gloves and haptic devices. Some example input devices are shown in Figure 2-1. The motivation behind using 6DOF devices for 3D manipulation is that these devices afford the simultaneous positioning and orientating of virtual objects. This theoretically provides a more efficient manipulation interface compared to input devices that control fewer simultaneous DOFs. Lower-DOF devices require separate modes to independently control translation and rotation, and thus may take longer to perform the same overall manipulation task. A common belief is that 6DOF devices may also afford more "natural" or direct interaction with virtual objects, allowing users to leverage their real world object manipulation skills.

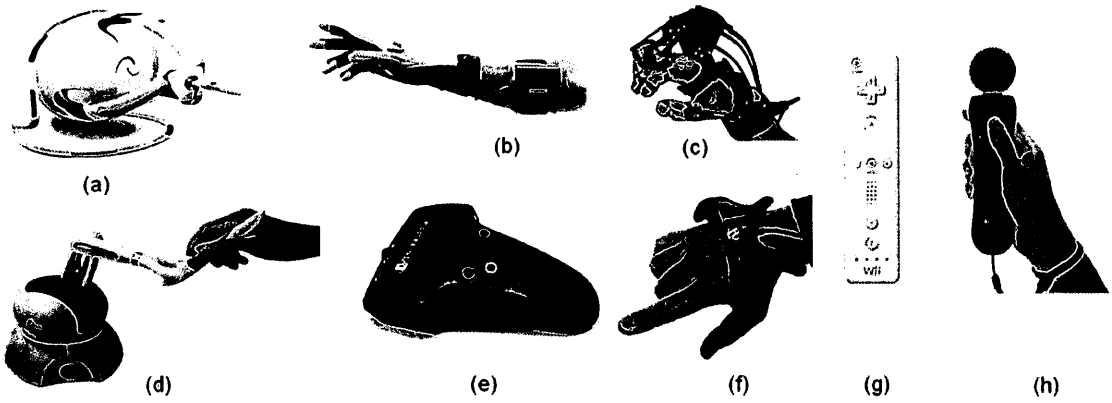

Figure 2-1: Various higher-dimensional input devices. This figure demonstrates the wide variety in input devices affording greater than 3DOF. (a) Novint Falcon (www.novint.com), a 3DOF desktop haptic device; (b) Cyberglove Systems CyberGlove II (www.cyberglove.com) which reports status of the fingers and is often used with another tracker; (c) CyberGrasp, a Cyberglove outfitted with a haptic exoskeleton; (d) Phantom Omni (www.sensable.com), a 6DOF desktop haptic device that uses a stylus as the main input device; (e) Intersense Minitrax Wand and (f) Intersense Hand Tracker (www.isense.com), a 6DOF free-space device; (g) Nintendo Wii Remote (www.nintendo.com), the first remote pointing game input device; (h) Sony Playstation Move (http://us.playstation.com/ps3/playstation-move/), a more recent 6DOF wand-styled game input device.

Many 6DOF input devices do not require a supporting surface, and thus are well-suited to virtual environment (VE) systems where the user is standing or walking such as the CAVE [20]. These input devices often use acoustic, inertial, optical, or electromagnetic tracking technologies or a combination thereof to determine the device position and orientation.

Most 6DOF selection and manipulation techniques fall roughly into two broad paradigms: ray-based techniques (and similar techniques like occlusion) and virtual hand metaphors [13, 14, 21, 65]. Each paradigm is discussed in greater detail below.

2.1.2 Virtual Hand and Depth Cursor Techniques

Virtual hand techniques almost always use a 6DOF wand or a hand-mounted tracker to control the position and orientation of a virtual hand avatar in the scene. The virtual hand represents the user's real hand in the environment, see Figure 2-2 [64]. Users can select objects by "touching" the desired object, then pressing a button on the tracker to indicate selection. To manipulate a selected object, the object is typically bound to the hand position/orientation, matching its movement and rotation until released in its new pose. These techniques are sometimes also referred to as "depth cursors".

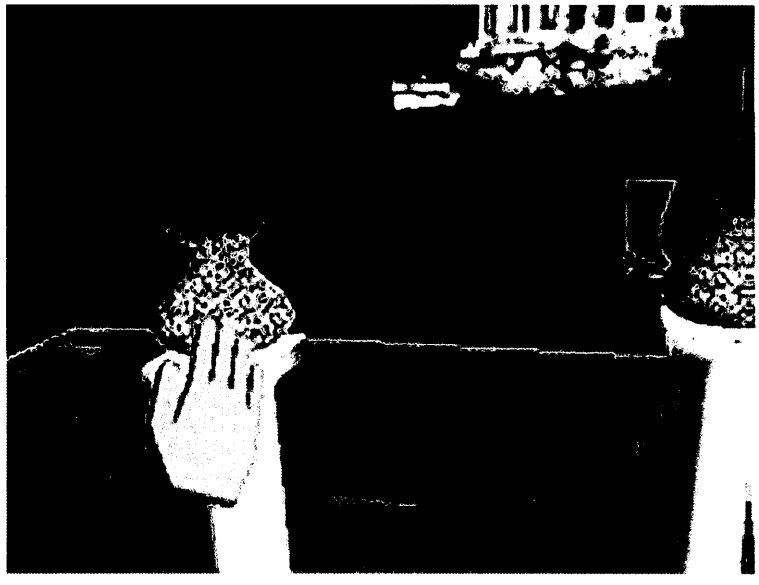

Figure 2-2: An example virtual hand technique. The hand must intersect objects to select them. This limits it to objects within the users' reach. Figure reproduced from Poupyrev et al [64].

An ideal virtual hand would mimic the user's hand, including fingers, but accurate and reliable finger motion tracking is only possible with very expensive exoskeleton systems, such as that shown in Figure 2-1(c). More recent innovations, such as the Leap

Motion (www.leapmotion.com) are very inexpensive, less than $100, but do not provide very reliable tracking of individual fingers. Non-spherical virtual hand representations are sometimes used, and they may be significantly larger than the user's hand itself to facilitate selection. A recent study compared several virtual hand shape variants, including metaphors such as cupping objects, using physical props in an augmented reality experiment and found that a "paddle" performed best in a selection task [32]. This technique allowed objects to be scooped up using a virtual paddle connected with a physical input device.

Boritz and Booth [11, 12] studied 6DOF input devices for 3D interaction, first looking at selection tasks [11]. They investigated several factors, including stereo display, head tracking, and target position. Participants had to move the cursor to one of 6 possible target locations 10cm away from the starting position along any of the positive or negative X, Y and Z axes. Target position had a significant effect on task completion time and accuracy; movement along the Z axis ("near" and "far" as it was called in the study) took longer and was less accurate than movement in the X and Y directions. However, these axis differences were smaller when using stereo vision. Boritz's second study [12] also considered the orientation of the target, requiring users to dock a cursor with a target, matching both position and orientation. Speed and accuracy both depended on the position moved to and the target orientation.

Zhai *et al.* [103] conducted a study of the *silk cursor*, a selection technique using transparency and volumetric selection for 6DOF selection tasks. They compared their

semi-transparent volumetric cursor to a wire-frame volumetric cursor under both stereo and mono display conditions. They found that in addition to significant differences by cursor type, the stereoscopic display significantly improved user speed and accuracy. Their results suggest that both (partial) occlusion and stereopsis benefit selection tasks, but using both simultaneously improves performance further. The benefits of volumetric selection have also been recognized in 2D interface design, and led to the development of area cursors [29, 41]. These effectively increase the target size, making it easier to select. Early approaches [41] used static sized area cursors, which improved accuracy, but decreased speed. The bubble cursor dynamically adjusts its size and shape to aid target selection and was demonstrated to improve selection speed and accuracy [29].

2.1.3 Ray-based Techniques

Ray-based techniques can use either 2DOF devices, like the mouse, or 3/6DOF devices, such as trackers. In this section, only the use of 6DOF trackers is discussed. The use of 2D input devices with ray-based techniques is discussed later in Section 2.1.6.

All ray-based techniques cast a virtual ray or line from the user's hand/finger or cursor. This ray is then checked for intersections with objects in the scene. Usually the object closest to the camera is selected. Ray-based selection and target disambiguation is depicted in Figure 2-3.

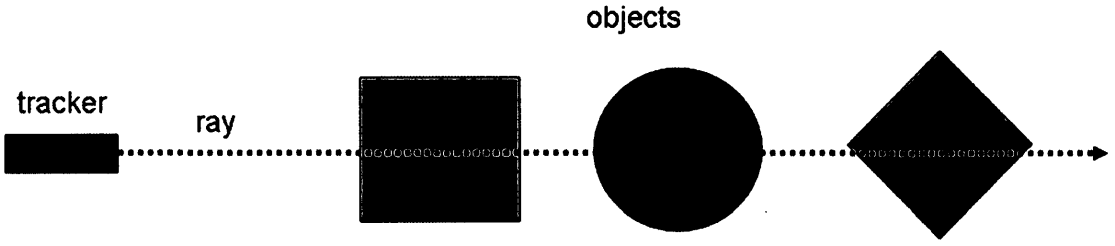

Figure 2-3: Ray selection – the ray originates at the tracker, and all objects (and/or their bounding volumes) are tested for intersections against the ray. Typically, the closest object to the ray origin is selected – in this case the green square.

Object selection is usually followed by manipulation of the selected object. A common ray-based manipulation technique is to simply fix the object at the tip of the ray once selected. Subsequent manipulation of the ray remotely moves the object until it is de-selected (e.g., by releasing a control button on the input device). With this technique, rotating the object about its centre is difficult as the center of rotation is the input device and the object is often a significant distance away. Effectively, this type of manipulation technique is akin to "skewering" the object, and then manipulating the skewer by holding the opposite end.

There is a great deal of interest in these techniques in both 2D [40, 60] and 3D [30, 44, 45, 78, 101] user interface design. In the 2D domain, these techniques are often used either to interact with large displays at a distance [40] or for collaborative systems [60].

A problem with standard ray-casting techniques is that they may perform poorly when the ray hits multiple targets. This issue commonly occurs for targets lined up in the depth direction. As the nearest object is selected by default, this may necessitate moving

the viewpoint or input device if a different target was actually intended. This can also occur for objects that are close to (or intersect) each other, or when large bounding volumes are used to improve the speed of intersection tests, see Figure 2-4.

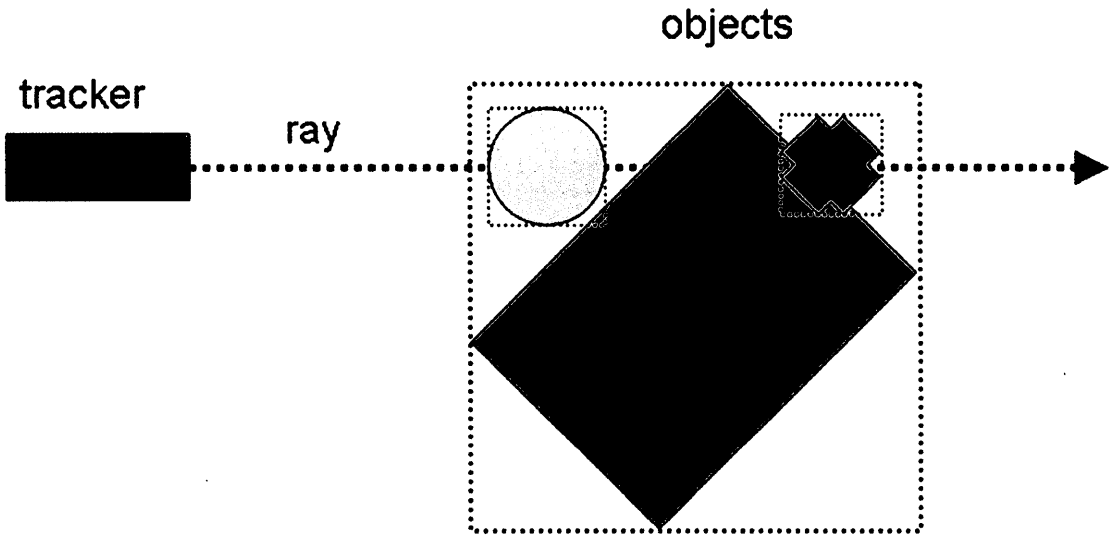

Figure 2-4: Ray disambiguation – when multiple objects or their bounding volumes (dashed boxes) are intersected by the ray, secondary measures must be used to select the specific desired object from the potential set of objects. Intersecting object bounding volumes in place of the real objects exacerbates this problem.

To address this issue, Grossman *et al.* [30] propose several extensions to the classical ray pointing metaphor. They report that a depth ray technique that allowed dynamic positioning of a cursor along the ray performed best, outperforming other techniques that required more complex disambiguation schemes.

Another commonly known drawback of ray-based techniques is the relative difficulty in selecting remote objects, as compared to close objects. Farther objects take up proportionally less screen space due to perspective, and are thus harder to select. Consequently, ray-based selection effectively has greater angular precision for close

objects. Conversely, remote objects are harder to select, as subtle device movements/rotations are amplified down the length of the ray [44].

Some work has focused on extending the "traditional" ray-casting technique to compensate for some of the observed problems. For example, Steinicke *et al.* [78] propose a dynamically bendable ray. The ray bends toward the closest objects in the scene, allowing selection even if ray does not directly hit the objects. The proposed ray also sticks to targets once hit by the ray to avoid accidental release of the object. However, no evaluation of this technique has been performed, so its potential advantages are unknown.

Another ray technique extension is cone selection [14], which allows volumetric selection of object groups within a cone emitted from the user's hand. The diameter of the cone can be dynamically adjusted to hone the selection to fewer objects as required. This is conceptually similar to the volumetric selection afforded by the silk cursor [103], which is not a ray-based technique.

2.1.4 Comparing Rays and Virtual Hands

Previous work presented taxonomies of 3D selection/manipulation techniques [13, 65], in order to characterize the fundamental components that make up 3D interaction techniques. One aspect of this work was the direct comparison between ray and virtual hand techniques. Bowman *et al.* [13] presented a fine-grained classification of techniques, breaking down each technique by its method of selection, manipulation, and de-selection, and further sub-dividing these groupings. This approach enumerates the

basic "building blocks" for 3D selection and manipulation techniques. Then one can build new ones through combining these components. Of course, some combinations make more sense than others. For example, consider a technique that requires the user to touch the object with the hand to indicate selection, but uses eye gaze to manipulate the object, and a hand gesture to de-select the object. This is likely less efficient than a technique that uses the same extremity for selection, manipulation and de-selection and buttons to indicate selection.

Results of a study [13] comparing a number of these selection/manipulation techniques indicated that ray-casting outperformed Go-Go [64]. The authors speculate this is because Go-Go (and similar virtual hand techniques) requires intersection of the user's virtual hand with the desired object, whereas ray-casting required merely pointing at it. Although ray-casting is often used with 6DOF devices, it normally only requires control of 2DOF to perform selection tasks, i.e., rotation of the input device/tracker in the yaw and pitch directions. Previous work has demonstrated that techniques requiring fewer degrees of freedom tend to outperform higher-DOF techniques [98]. These results were again confirmed by Grossman *et al.* [30] who compared selection using a relative 3DOF point cursor to ray-based techniques while investigating ray disambiguation. While the point cursor implicitly disambiguates target selection, it still underperformed relative to the ray-based techniques requiring explicit disambiguation. In particular, point cursor selection was significantly affected by target distance, unlike ray-based techniques.

Poupyrev *et al.* [65] compared selection and manipulation with 3D ray-casting and a virtual hand technique. They found no significant difference between the virtual hand and ray-casting for selection. Each technique tested had advantages and disadvantages, depending on factors such as distance to the target, object size and visual feedback. These results [65] somewhat contradict Bowman's [13]. This is likely due to the dramatic difference between the tasks used in each study, see Figure 2-5. Bowman's task required selecting a specified cube from a set and to position it between two cylinders, while varying target distance, size, and distracter densities. Poupyrev's selection task required selecting an isolated object in the environment, while his manipulation task required selecting an object then placing it on top of a second object in the environment. The difficulty in comparing these studies is exacerbated by the fact that some details (e.g., specific target sizes, distances, etc.) are not reported. Bowman's selection task is also more complex as it requires selecting the correct object from a set.

One commonly accepted explanation for the measured performance differences between ray- and hand-based techniques is the use of different muscle groups to activate these techniques. Ray-based techniques, for example, can be used by only rotating the wrist and otherwise keeping the hand immobile. Note that this requires control of (at least) 2DOF of relatively fine muscle groups, without necessarily employing larger, less agile, muscle groups (e.g., in the lower/upper arm). Conversely, virtual hand techniques can require movement at the elbow, or even the shoulder, usually necessitating 6DOF

control. This difference in the number of controlled DOFs may account for the measured differences.

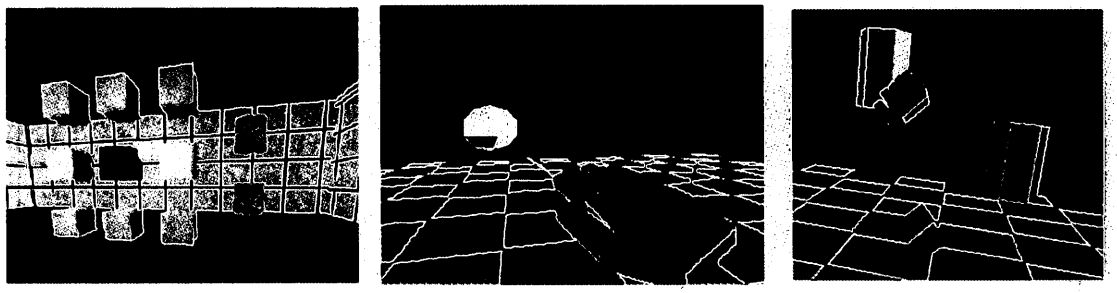

Figure 2-5: Left: Bowman's selection/manipulation testbed task. Participants would select the central (highlighted) cube, and place it between the two wooden cylinders to the right. Right: Poupyrev's selection task, ray-casting and go-go techniques.

To investigate this, a study conducted by Zhai and colleagues [104] compared the use of specific muscle groups for manipulation. The muscle groups needed to manipulate a device are an important consideration when directly comparing two different input devices. It has been suggested that the use of more dextrous muscle groups, such as those that control the fingers, can aid in 6DOF manipulation tasks. Based on this observation, it is not uncommon to see glove-based interfaces used in virtual environments (see Figure 2-1 for examples). Zhai's study compared two input devices for a 6DOF docking task: one based on a 3D tracker mounted on the palm of a glove and the other based on a 3D tracker inside a ball the user holds with their fingers. A series of analyses showed that the "FingerBall" was faster than the glove. The authors suggest that the use of fine-motor control muscle groups, such as those in the fingers, is beneficial in 6DOF manipulation tasks especially if various parts of the arm work together in unison, rather than in isolation [104].

This conclusion was supported by later work comparing muscle groups in the fingers, wrist and forearm [8]. Using these muscle groups *together* seems to result in superior performance compared to just using the fingers alone. The authors found that holding a stylus between the thumb and forefinger permitted better task performance than a sensor mounted on the fingers, wrist or forearm. The researchers isolated each muscle group using restraints. The authors conclude that certain muscle groups are likely better suited to certain types of movement tasks, and consequently, input devices that use specific muscle groups should be matched to the task at hand. These studies suggest that there may be merit to the claim that high-precision hand and finger tracking would improve virtual hand techniques immensely.

The bulk of the work described above focuses exclusively on object positioning tasks, which constitutes only one component of manipulation. Rotation tasks are also required for full 6DOF manipulation. Docking tasks require both position and orientation matching. Previous work used handheld trackers in docking tasks [12]. Results indicate that rotations about the x-axis (i.e., the axis orthogonal to the view vector and the up direction) were significantly worse than the other axes. Zhai's aforementioned experiment [104] also used a 6DOF docking task. The Fingerball was faster than the hand mounted tracker, likely due to the relative ease with which the ball could be rotated. The hand tracker required more clutching. This suggests that rotation was the dominant factor in the results; translation had a limited impact on the overall task completion time.

Note that these docking studies used isomorphic (1:1) mappings of input device rotation to virtual object rotation. Non-isomorphic mappings are also possible, and can improve performance depending on the task [67]. In particular, for large rotations, a non-isomorphic rotation technique significantly improved rotation speed, without decreasing accuracy [67].

2.1.5 Hybrid Ray/Hand Techniques

A primary advantage of ray-based techniques over standard virtual hands is that physical arm length does not limit the user when reaching for virtual objects. This limits the need for navigation. To address this, hybrid techniques have been presented to leverage this advantage of ray-casting, but including the potentially more familiar hand metaphor for up-close manipulation.

While a basic virtual hand technique uses a one-to-one mapping of hand to cursor motion, other mappings are also possible [15, 64]. For instance, virtual hands can better approximate the familiar desktop mouse interface: the input device and cursor movements are decoupled, and tracker motion maps to 3D cursor motion only in a relative rather than absolute way. An example of this is the Go-Go technique [64]. This virtual hand technique enables the user to interactively and non-linearly adjust the length of their virtual arm when manipulating an object in 3D. While it is not a ray-based technique, it is similar in that it allows remote selection of objects (followed by close manipulation).

The HOMER technique [15] is another example of a hybrid ray-casting/virtual hand technique. It uses 3D ray-casting for selection and then automatically moves the user's virtual hand to the position of the selected object. Like Go-Go [64] this effectively extends the user's arm, allowing the user to manipulate remote objects without having to physically move closer to them. This also allows a greater degree of rotational control when manipulating objects, as rotations of the hand are mapped directly to object rotation.

2.1.6 2DOF Input Devices

The research discussed so far used 3 or 6 DOF input devices. Other research suggests that 2D input devices can outperform 3D devices for certain 3D positioning tasks [9, 61, 83-85]. In general, selection and manipulation techniques based on 2D ray casting are similar to 3DOF techniques. In particular, they allow pixel-precise selection and subsequent manipulation of an object intersected by the ray, regardless of its distance, subject to the limitations discussed above. Ray casting using 2D devices is described in greater detail below.

Bowman's study [13] (discussed in greater detail in Section 2.1.4) found that selection based on ray-casting and occlusion was significantly faster than selection techniques requiring 3D hand or cursor movement. For manipulation, they found that the degrees of freedom of the manipulation task had a significant effect on task completion time. In fact, they note that this factor dominated the results, with techniques based on

2DOF movement significantly outperforming 6DOF techniques, on average. This supports earlier findings reported by Ware and Lowther [98].

One factor is that most computer users are extensively familiar with 2D input devices, in particular the mouse. Touchscreens and stylus-based interfaces are also becoming common. Practically all commercially successful 3D graphics systems (including 3D modeling packages and computer games) use a mouse-based direct manipulation interface. Clearly there are advantages to using a mouse, including user familiarity, a supporting and jitter dampening surface, high precision, and low latency. However, the use of a mouse for 3D interaction introduces the problem of mapping 2DOF mouse motions into 3 or 6DOF operations. This is typically achieved through the use of mouse-based ray-casting.

Ray-casting can also be used with 2DOF input devices to enable 3D selection. For this it suffices to use the 2D screen coordinates of the mouse cursor and to generate a ray originating at the viewpoint (the center of projection, or camera position), passing through that 2D point on the display, and into the scene. This requires an inversion of the projection process normally used in computer graphics to map the cursor position into a line, the ray, through the scene. Most graphics platforms provide support for inverting the projection matrix, effectively transforming the "un-projected" point into a line through the 3D scene. This is conceptually demonstrated in Figure 2-6.

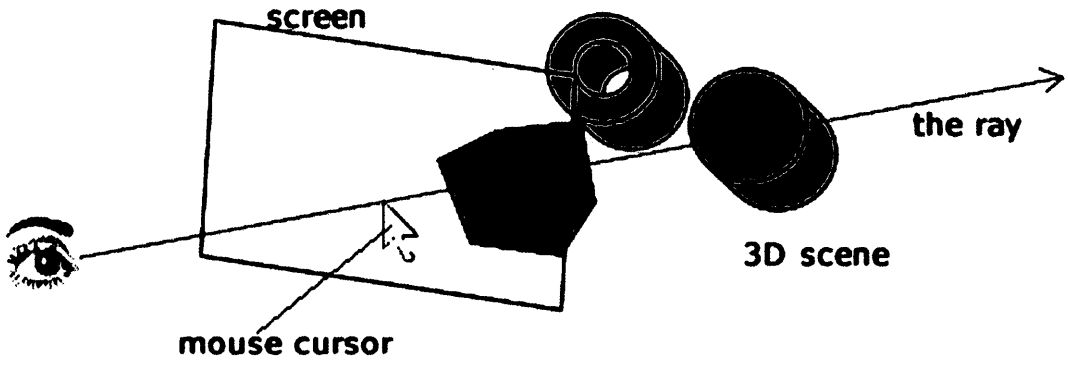

Figure 2-6: Ray casting using a mouse to select objects in 3D scene. The eye represents the centre of projection, i.e., the position of the camera in the virtual scene. Usually, the closest object intersected by the ray is selected.

Ware and Lowther [98] point out that situations where the user wishes to interact with totally occluded objects are rare and that the 2D projection of a 3D scene is fully representative of all visible objects in that scene. Ray-casting using the mouse cursor allows the user to pick any (even only partially) visible object with single pixel precision [98]. Ware and Lowther also reported that a 2D ray-casting technique using a cursor rendered only to the dominant eye in a stereo display was both faster and more accurate than a 3D selection cursor roughly corresponding to a virtual hand technique. Overall, the 2D technique offered nearly double the performance of the 3DOF technique.

Another difference between 6DOF and 2DOF ray-casting is that the former can allow selection of objects the viewer cannot actually see. Since the origin of the ray is the user's tracked hand, it is possible for them to point *around* other objects in the scene and select occluded objects. While this could be used advantageously, it can also confuse users. Consider that tracker noise and hand jitter can easily result in accidental selections

of hidden objects when the user actually meant to select the occluding object. This is not an issue with 2DOF ray-casting, as all selectable objects must be at least partially visible.

Three-dimensional manipulation with 2D input devices is less straightforward than selection, since the user has to perform either a 3DOF task when either positioning or orienting objects, or a 6DOF task when both positioning and orienting objects. However, 2D input devices, such as the mouse, only afford the simultaneous manipulation of 2DOF. Thus, 2D input must be mapped to 3D operations via software techniques. Most solutions to this problem require that users mentally translate 2D mouse movements into 3D operations, e.g., controlling one or two degrees of freedom at a time. This effectively decomposes the high-DOF manipulation task into a series of low-DOF tasks, increasing the overall cognitive overhead. Effectively, the user must focus on performing each sub-task in succession, rather than focusing on their actual goal. Examples of this strategy are 3D widgets, such as "3D handles" [19], the "skitters and jacks" technique [10], the Arcball technique for rotation [70], or the use of mode control keys.

In commercial 3D modeling and CAD packages, the most commonly employed solution is 3D widgets [19, 80]. These handles separate the different DOFs by explicitly breaking the manipulation down into its individual components. Small arrows/handles are provided for movement along each of the three axes or the planes defined by two axes, and orientation circles/spheres for each axis of rotation, see Figure 2-7. This is usually complemented by different simultaneous orthogonal views of the same scene from

different sides. Bier's skitters and jacks technique [10] provides a similar solution, by interactively sliding the 3D cursor over objects in the scene via ray-casting, and attaching a transformation coordinate system to the object where it was positioned.

Mode control keys allow the user to change the 2DOFs the mouse is currently controlling by holding a specific key. For example, movement may default to the XZ plane, but holding the "shift" key during the movement may change the plane of movement to the XY plane instead. The limitation of these types of manipulation techniques is that users need to mentally decompose every movement into a series of 2DOF operations mapping to individual operations along the three axes of the coordinate system. This increases user interface complexity and creates the potential for mode errors. Although practice mitigates these problems, software using these strategies tends to have a steep learning curve, requiring extensive practice to master.

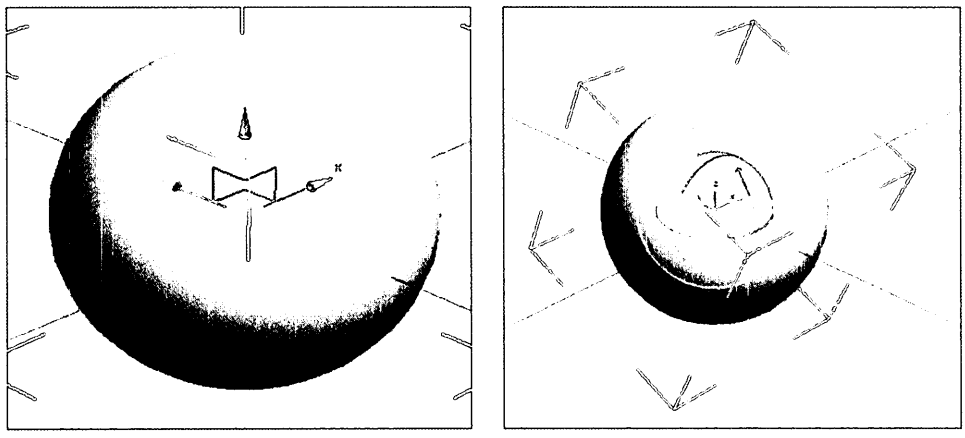

Figure 2-7: Translation and rotation 3D widgets from Autodesk's 3D Studio Max. Clicking and dragging on the arrows displayed will move the sphere along the selected axis. At most two degrees of freedom can be simultaneously manipulated in this fashion.

A different approach is to constrain the movement of objects according to physical laws such as gravity and the inability of solid objects to interpenetrate each other. Such constraints can then be used to limit object movement according to human expectations [74]. For example, chairs can be constrained to always sit on the floor, and desk lamps on top of desks. A problem with this approach is the lack of generality, as it requires object-specific constraints to be designed a priori for each available type of object in the virtual environment. As such, this type of constraint system seems more suitable for manipulating objects in games, as they typically include only a limited set of objects in a restricted environment. For systems that either allow custom object creation, or have a very large number of objects available, more general approaches are preferable.

The SESAME 3D movement technique is one such general approach, and relies solely on contact-based sliding and collision avoidance. This algorithm ensures that the object being moved remains in contact with other objects in the scene at all times [61]. Objects are selected via 2D ray casting based on the mouse cursor position. Following object selection, the user can then move the input device to simply "drag" the object across the scene, while holding the "selection/action" button down. This is inspired by the "click and drag" metaphor popularized by desktop computing. The algorithm handles depth automatically and keeps the object stable under the cursor, i.e., an object simply slides across the closest visible surface that its projection falls onto. Figure 3-1 depicts how mouse motion maps to object movement in this system. When moving the mouse forward, the selected cube first slides along the "floor" of the scene. Upon detecting

contact with the larger block in the background, the selected (moving) cube then slides up and over the front side of the stationary cube. In other words, the forward mouse movement will alternatively move the cube along the Y or Z world axes, depending on contact detection with other surfaces that constrain its movement in that direction.

Essentially, this technique reduces 3D positioning to a 2D problem, as objects can now be directly manipulated, and are moved via their 2D projection. Previous research has indicated that it is very efficient compared to other common techniques, such as 3D widgets [61]. Also, novices learn the technique very quickly, perhaps due in part to its similarity to standard 2D direct manipulation interfaces.

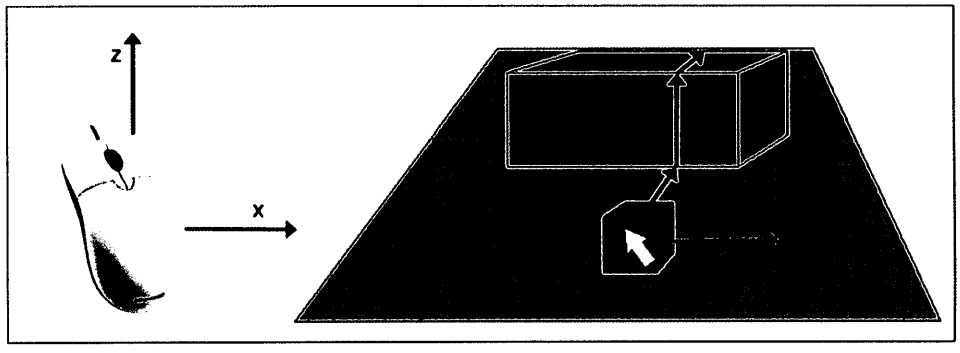

Figure 2-8 : Mouse motion to 3D movement mapping in SESAME.

This technique can be adapted for usage with 3DOF/6DOF input devices especially using ray casting. The simplest option is to ignore the third degree of positional freedom [84]. In this situation, a tracker behaves like a mouse, constrained to 2DOF movement. Figure 3-2 depicts the mapping of a 3DOF wand input device to object motion using the same 3D movement technique.

In this case, movement of the device in the XY (vertical movement plane) is mapped to 2DOF. The XZ plane (horizontal movement plane) could also be used, which effectively makes the tracker even more similar to the mouse. Results of a study investigating this last possibility revealed that a 3DOF device constrained to 2DOF operation in this fashion significantly outperforms a full 3DOF technique [84]. This was again verified in later work [85]. In fact, the last study revealed that the chosen orientation plane did not significantly affect task completion time.

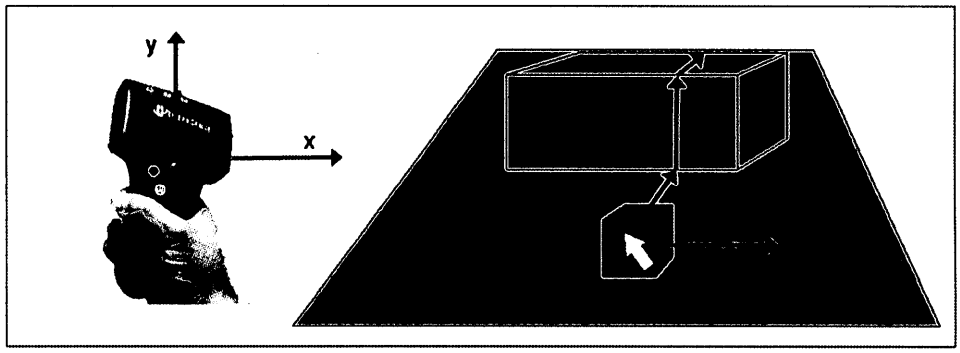

Figure 2-9 : Wand motion constrained to 2DOF operation via SESAME movement technique.

Rotating objects is also a 3DOF task. The aforementioned 3D widgets approach is also used for this in most commercial CAD/modeling systems. Like translation widgets, rotation widgets allow the user to rotate the object around one or two axes at a time. Virtual trackballs are another method of rotating objects with a mouse. The Arcball [70] is an early, yet effective, example of this. Using virtual trackballs can be conceptually thought of as rotating a trackball containing the object of interest using a single point on

its surface. This is accomplished by clicking onto part of the sphere, and dragging the mouse to a new position. The straight line formed from the mouse start and end points is treated as an arc on the surface of the sphere, and the sphere (containing the object being oriented) is rotated by the angle of this arc. Note that this accounts for only 2DOF of rotation. The third degree, rotation about the depth axis, is handled by dragging in circle motions *outside* of and around the virtual trackball. Other approaches use the mouse wheel for the third degree of freedom [71]. There are several variations on the virtual trackball approach [35]. One of these, the two-axis valuator has been empirically demonstrated to outperform the other approaches for speed [7]. This approach maps horizontal mouse movement to rotation about the "up" direction of the object, and vertical mouse movement is mapped to the vector perpendicular to the up and view vectors. This technique behaves in a very predictable way, and supporting user expectations is likely its advantage. It may be enhanced by using physical constraints (e.g., collision avoidance) [71].

2.2 Depth Cues, Stereo Graphics, and Head Tracking

This section discusses the binocular depth cues supported by stereo 3D graphics and the head motion parallax cues supported by head tracking. Used together, these two cues allow geometrically correct rendering of the scene, i.e., a stereo 3D view from the user's current viewpoint rather than a fixed camera position. For a broader overview of depth cues, see Chapter 8 of Colin Ware's *Information Visualization* [92].

2.2.1 Depth Cues – Stereopsis, Convergence, Accommodation

Many depth cues are monocular, i.e., they only require one eye to be perceived. Some of the stronger monocular cues include perspective (farther objects appear smaller) and occlusion (near objects visually block far objects) [100]. These cues are adequately simulated in computer graphics. Perspective is achieved using a frustum viewing volume and perspective transformations [33]. Occlusion, on the other hand, is typically simulated using the *z*-buffer algorithm to only render the nearest visible pixels (in the absence of transparency) [33]. Texture and illumination/shading are also depth cues that are provided in current computer graphics systems. Shadows can also be simulated, but this is more difficult and computationally expensive [33].

A different depth cue is accommodation, which refers to the flexing of the lens of the eye to focus the eye on stimuli. The lens is stretched more for far targets. This is shown in Figure 2-10 [90]. This effect is rarely simulated in modern graphical displays, as this would require the ability to detect both where the eyes are looking, as well as the focus distance of the eyes [92].

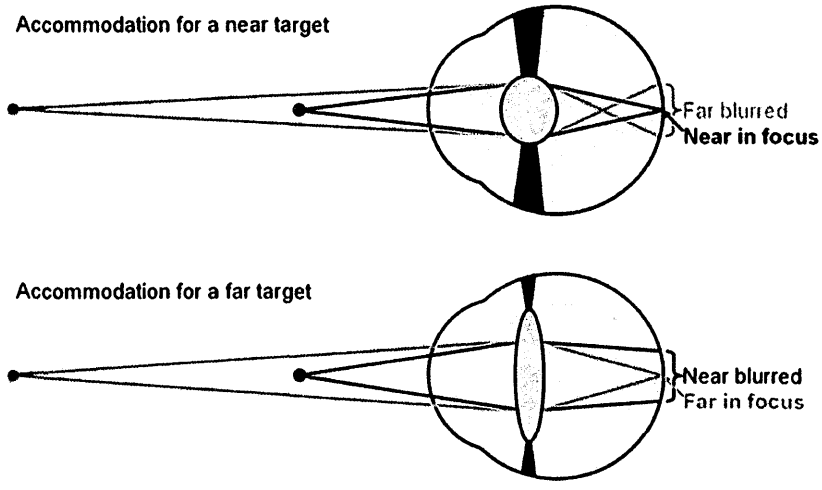

Figure 2-10: Accommodation to near and far targets. Image courtesy SAP Design Guild [90].

Of particular interest though are the binocular depth cues, those that require two eyes to be perceived. These include stereopsis and convergence [92]. Our eyes are set roughly 6 – 7 cm apart in our heads, so each eye has a slightly different view of the same scene. The image of a close-by object perceived by the left eye is shifted slightly to the right, and vice versa. Through stereopsis we can thus detect additional depth information. Convergence refers to the ability of the eyes to turn inward to cross at the perceived depth of stimuli. For very distant objects, the gaze of the eyes becomes more parallel. Figure 2-11 [90] depicts this.

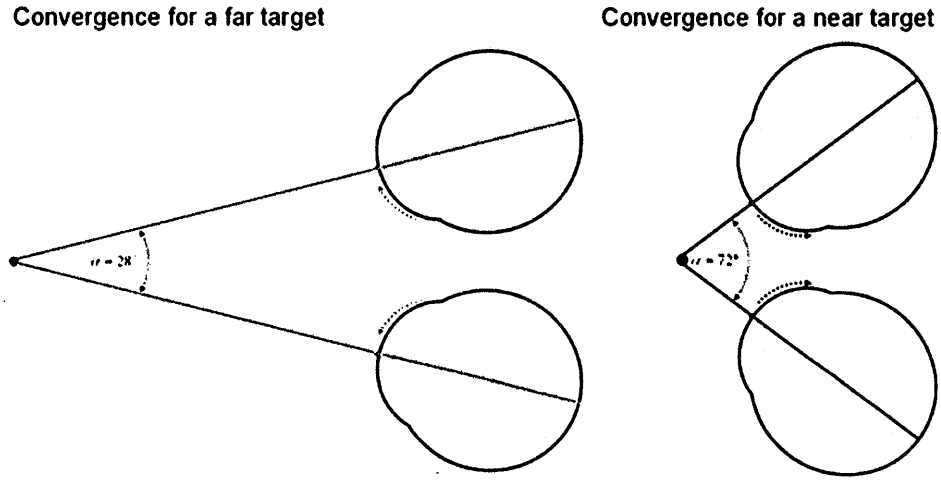

Figure 2-11: Convergence and convergence angles. Image courtesy SAP Design Guild [90].

2.2.2 Depth Cue Conflicts in Stereo Displays

When viewing objects in the real world, our eyes will converge and accommodate to the same point [92], and thus will always provide consistent depth information. In stereo graphics system, the technology yields conflicting depth information from these two cues. Our eyes converge at the true perceived depth of presented stimuli, but accommodate to a single plane, typically the display surface, see Figure 2-12, reproduced from Shibita *et al.* [69]. This can result in eyestrain, nausea, and headaches [36, 37], especially for short distances such as in small-scale VR systems or when the user is manipulating objects in arms reach.

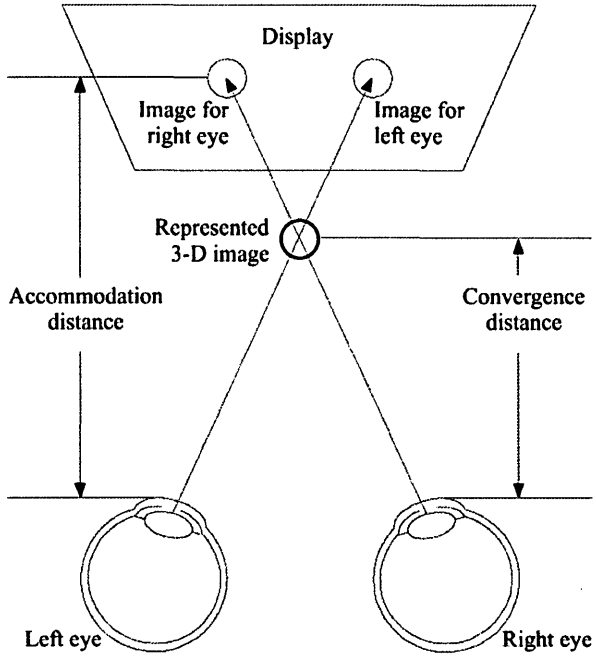

Figure 2-12: The convergence and accommodation cue conflict problem. Image courtesy Shibita et al. [69]

The convergence/accommodation cue conflict has a small negative effect on depth judgement tasks [36]. It is less clear if this also extends to interactive tasks such as 3D object manipulation, but seems likely given that these tasks are largely guided by visual perception during fine placement [102]. One option to avoid the problem altogether is to use specialized optics to adjust the accommodation depth to match the convergence depth [69]. However, this is problematic in highly dynamic systems, as the optics need to adjust quickly to the depth of the current target.

Another alternative is to use a volumetric display. These produce matching convergence and accommodation cues because they display a true 3D image composed of points of light floating in space. This is technically accomplished by either projecting into

some medium, such as a fluid [22], or by sweeping the display surface through a volume while changing the image over time [28, 30]. These systems are not without their own drawbacks, however; aside from being extremely expensive and not widely available, they prevent interaction *in* the environment due to the nature of their respective display volumes. Occlusion cues are also lost, since points of light cannot physically occlude one another. Still, they show some promise in interaction tasks possibly due to the improved depth perception they offer [31].

2.2.3 Evaluating Stereo Displays

Stereoscopic 3D rendering is commonly used in VR systems [14]. This involves displaying a slightly different image for each eye, then filtering each image to the appropriate eye, often using special glasses, or a head-mounted display with separate left and right eye displays. In 3D graphics systems, this can be exploited to make the imagery appear to extend beyond or behind the screen surface. At present, this technology is also widely used in 3D movies and was popularized recently again by James Cameron's *Avatar* and 3D capable televisions are now widely available. Stereo 3D games are also becoming common, e.g., the Nintendo *3DS*, launched on March 27, 2011 in North America (http://www.nintendo.com/3ds) and employs an autostereo display. Stereo imagery can be viewed on such displays without special filtering glasses, if the viewer's eyes are within an area usually referred to as the "sweet spot".

The objective of using stereo display technology in virtual environments is to provide a more visually rich experience. In particular, stereo systems are intended to aid

depth perception in 3D graphical systems. By exploiting these stereo depth cues in virtual reality, system designers hope to allow people to better leverage their natural vision abilities, and enhance interaction with these systems. Some researchers also argue that this improves immersion in the system. Immersion, in turn may enhance presence – the psychological sense of feeling as though you are actually *in* the virtual environment. While presence and immersion are difficult to quantify, some researchers argue that they too enhance one's ability to interact with a virtual environment [72].

2.2.4 Graph Tracing

One area that has demonstrated some clear benefits of stereo viewing is path/graph tracing [6, 31, 96, 99]. These experiments typically present a 3-dimensional graph/tree structure and participants are asked to determine if there is a path of a given length between two specific nodes. These studies have consistently shown that the presence of stereo display (and head tracking) decrease errors in this kind of task while not necessarily improving the speed at which participants perform the task.

Although not the first to conduct such a study to evaluate the importance of stereo display, Arthur *et al.* [6] were the first to conduct a stereo investigation in a fish tank VR system [94]. They found that stereo only slightly improved graph tracing speed, but significantly improved error rates in this kind of task [6], particularly when compared to a static 2D image of the graph. These results are echoed by Ware and Franck [96] and later again by Ware and Mitchell [99]. These studies also included the effects of motion parallax due to head-coupling the viewpoint; these results are discussed separately below.

Grossman and Balakrishnan's more recent study [31] also replicated the same design, but included a volumetric display. They too found that error rates were significantly lower with the stereo head-tracked condition actually outperforming the volumetric display. This may be either due to the low resolution, or to the absence of occlusion cues on the volumetric display.

2.2.5 Selection and Manipulation

Compared to graph tracing, object selection and manipulation tasks are likely more prevalent in 3D user interfaces. Consequently, it is important to understand the benefits of stereo technology in these tasks. However, there are relatively few studies that explicitly and systematically investigate the benefits of stereo in 3D selection and manipulation tasks. According to Woodworth [102], goal-directed movements (such as those used to reach out and grab an object in a virtual environment) are broken into a ballistic phase and correction phase. The ballistic phase is "pre-programmed" based on perception, but then carried out without the aid of perceptual feedback. The correction phase employs visual feedback to home into the target. Consequently, it seems likely that the improved perceptual mechanisms offered by stereo displays should improve the ease with which users can perform these tasks – at least during the correction phase of the task.

Zhai *et al.* [103] compared object selection with their semi-transparent volumetric "silk" cursor to a wireframe volumetric cursor, in both stereo and mono viewing modes. Stereo display significantly improved user speed and accuracy, especially for the

wireframe cursor. While the silk cursor was only marginally better in stereo mode than in mono, task performance with the wireframe cursor was nearly twice as fast and had roughly half the errors. Stereo display decreased error magnitude by a factor of about 2.5 relative to mono display.

Boritz and Booth [11, 12] also evaluated the benefits of stereo in 3D point selection tasks [11] and a 3D docking task [12]. Their study compared stereo to mono display, with and without head tracking. They found that stereo display significantly improved task completion time, especially for motions in depth (i.e., movements *into* or *out of* the screen). Similarly, error magnitude was significantly reduced for depth motions with stereo display. Arsenault and Ware [5] conducted Fitts' law study in a fish tank VR system and found that stereo improved target tapping/selection speed by around 33%. They did not report accuracy or throughput scores, both of which could be used to directly compare their results to more recent work. Later work [84] confirmed that stereo improved accuracy but not speed in a "drag and drop" style task.

Overall, the results of these studies suggest that stereo improves accuracy in 3D selection/manipulation style tasks. Some researchers also reported improved task completion times. This may be due to the speed/accuracy trade-off inherent in these kinds of tasks, i.e., participants did not need to spend as much time positioning/selecting the target in order to accurately hit the target.

2.2.6 Head Tracking

Many VR systems also track the user's head [5, 11, 12, 26, 46, 47, 56, 86]. Using a head tracker allows one to determine the approximate position of the viewer's eyes as offsets from the current head position. Once the eye positions are known, these can be used as the positions of the virtual cameras used to render the scene, optionally (but typically) in stereo. This effectively couples the viewpoint to the head position. In large-scale VR systems (such as CAVEs) this allows users to walk through the environment, and to display the correct view of the scene from the current vantage point. In small-scale systems, such as fish tank VR, this can create the illusion of perceptual stability – i.e., stereoscopically rendered objects appear to be suspended in front of or behind the display surface and maintain their perceived position regardless of where the user views them from. Head tracking provides motion parallax depth cue, further enhancing the perceptual richness of a virtual environment.

Several studies have been performed to determine the benefit of head-tracking in fish tank VR [11, 12, 56, 84]. The aforementioned point selection and docking studies conducted by Boritz and Booth [11, 12] did not reveal any significant effects due to the presence of head tracking. The authors reasoned that their tasks required only minimal head movement after the initial discovery of target locations; it is unlikely that head movement is required during a precise motor task. These results are confirmed in a later manipulation study as well [84]. Aresenault and Ware [5] report a significant improvement in tapping speed of about 11% due to the presence of head tracking – in

contrast, stereo improved speed by three times as much. Ware and Mitchell [99] report that stereo alone was the fastest condition in their graph tracing experiment, significantly outperforming a stereo and head-motion condition.

In general, the benefits of the extra depth cues provided by head-coupled perspective and stereoscopic graphics may be task dependent, or dependent on the user's head movement strategy. It has been suggested that tasks with a higher depth complexity would benefit more from the addition of stereo graphics and/or head-coupled perspective. This is supported by previous work in which participants were able to more quickly trace a complex graph/tree structure when provided with the extra depth cues [94]. In contrast, tasks used in other studies [11, 12, 84] were performed in simpler scenes, and required relatively few depth judgements by the users. In fact, the only depth judgements typically required were only necessary to ensure that the object being manipulated was within the extents of the target zone, i.e. along the depth axis.

2.3 Technical Issues – Input Devices and Displays

This chapter discusses several technical issues common to most VR input devices and display technologies. First, most input devices are subject to tracker jitter and latency (temporal lag). Similarly, *displays* also exhibit some latency, which in turn contributes to the overall end-to-end latency. Second, the presence of tactile/haptic feedback may improve manipulation performance. A third issue is the coupling between the input and display spaces. Many VR systems co-locate these, creating the illusion of being able to reach out and grab virtual objects with a tracked appendage.

2.3.1 Latency

End-to-end latency is the time from when the device is sampled to updates appearing on the screen. It is the sum of the latency of all parts of the system, starting from the input device, through the software and rendering system, to the display. Although hardware manufacturers may be interested in minimizing latency in a specific device it is the aggregate end-to-end latency that affects the user. When performing experimental evaluations comparing multiple input devices, care must be taken to measure the end-to-end latency, as this will at least partly account for some of the measured performance difference between devices. Ideally, if latency is not a factor being investigated, one would also match latency between conditions. Various methods exist to measure the latency in a system [54, 77]. Many of these rely on comparing the measured signal to a known ground truth, for example, the periodic motion of a pendulum.

Failure to measure this is a clear issue in experimental design. Consequently, during the 2009 IEEE Virtual Reality Conference panel "Latency in Virtual Environments" it was suggested by a prominent VR researcher, Dr. Robert van Liere, that research that fails to report the latency should be considered incomplete work in progress. Although this suggestion is somewhat controversial, it demonstrates how seriously many researchers take latency, which will always be present in devices despite advances in technology.

It is well-known that latency adversely affects human performance in both 2D pointing tasks [50, 63] and 3D tasks [23, 83, 95]. Latency has also been demonstrated to

decrease the perceptual stability of a virtual environment, and the scene appears to "swim" in front of the viewer [2]. Participants experiencing a large amount of lag (e.g., greater than 200 ms total) reported decreased perceptual stability of the virtual environment, especially during fast head movements.

MacKenzie and Ware [50] report on a 2D pointing study using a mouse with artificially added latency. The highest latency condition (225 ms) increased movement time by around 64% and error rates by around 214% relative to the base lag condition (8.3 ms). Performance degradation was especially pronounced for harder pointing tasks (i.e., those with smaller and/or farther targets), even with latency of as low as 75 ms. Ware and Balakrisnan [95] report similar findings in a 3D interpretation of a Fitts' law task. Both studies included a regression analysis to derive a predictive model of pointing performance that included latency. Both multiplied the latency by the task difficulty (index of difficulty, *ID*), but the second study [95] included a second multiplicative factor to account for an even larger measured effect of latency. It is possible that the 3D task used in this study was more sensitive to latency, especially for pointing tasks in the depth direction.

2.3.2 Jitter and Noise

Jitter is the fluctuation over time in the position of a cursor. It is caused by a combination of device signal noise and hand tremor over time. Noise can be observed by immobilizing a device while observing the reported positions; even when stationary, the reported

positions fluctuate. In addition, when held unsupported in space, the human hand shakes slightly. This hand jitter exacerbates tracking jitter in free-space tracking devices.

Small amounts of jitter seem to have limited impact on user performance. Previous work [83] demonstrated that 0.3 mm average jitter artificially introduced to both a 2D and 3D mouse-pointing task did not significantly affect user performance compared to a no-jitter condition. This (small) amount of jitter matches that present in the 3D tracking system used in the study. More extreme amounts of jitter do impact user performance though, especially for small targets [63]. Using smoothing, one can effectively trade jitter for latency, i.e., filtering eliminates jitter, but takes time and delays frames. This may be beneficial in systems with small targets, and if the cost of corrections is high [63].

Latency can also change with respect to time, and this is sometimes called latency jitter. Ellis *et al.* [24] report that people can detect very small fluctuations in lag as low as 16 ms. Hence when examining system latency, one should also ensure that latency jitter is minimized, or at least measured.

2.3.3 Physical Support, Tactile Feedback and Proprioception

One property of the mouse that is simultaneously a great advantage and a great limitation is the fact that it requires a physical surface upon which to work. Not only does this help prevent fatigue by allowing the user to rest their arm but it also steadies the hand, dampening tremor that can result in decreased movement precision. However it also makes the mouse largely unsuitable for certain types of 3D environments such as

CAVEs, since it constrains the input to locations where a flat surface is present. This problem is exacerbated in virtual environments using head-mounted displays, as the user is also unable to see the device itself [47].

The positive properties of a support surface have been recognized in the virtual reality community. An early approach to address the absence of this tactile feedback was instead to use proprioception, the sense of the position and orientation of one's body and limbs. Mine *et al.* [55] discuss the use of proprioception as a first step toward compensating for the absence of physical support surfaces and haptic feedback in most virtual environments. It allows one to tell, for example, the approximate position of one's hand relative to the rest of the body, even when the eyes are closed.

Mine *et al.* [55] proposed the use of proprioception for fixed-body position and gestural controls in a virtual environment. For example, upon selecting an object for manipulation, a user could delete it by throwing it over their shoulder – a logical mnemonic that is difficult to invoke accidentally and employs the user's proprioceptive sense. They also developed user-centred widgets that behave like tools for indirect manipulation of objects at a distance. Unlike the object-centred widgets commonly used in 3D graphics applications [19], these widgets are centered at the user's hand and are used like tools on objects in the environment. A study showed that users were able to perform 6DOF docking tasks more effectively with objects attached to their hands, and preferred widgets centred on the hand more than those floating in space. The authors reason that proprioception made these techniques easier to use than the alternatives [55].

One problem with these types of approaches is that gestural interaction requires the user to memorize specific motions in order to activate the desired operation. This is mitigated through intelligent mnemonic design, such as the delete action mentioned above.

Later research built on this idea by adding actual mobile physical support surfaces to these types of environments. Most notable among these are the personal interaction panel [81], Poupyrev's virtual notepad [66], and Lindeman *et al.*'s HARP system [46, 47]. These approaches present virtual interfaces overlaid over a real physical surface, often a pressure-sensitive tablet or "slate", which the user carries around with them. The virtual representation of the slate is registered with its real-world position, and can either feature 2D or 3D user interface widgets on it. The user typically interacts indirectly with the environment via the user interface displayed on the slate. In a sense, the idea behind these interfaces is to combine the best aspects of 2D and 3D user interfaces – a full 3D virtual environment, in which the user can navigate, coupled with and controlled by a more familiar 2D interface. These systems often use a 3D tracked input device (e.g., a stylus) to determine which UI widgets are being activated on the physical surface. Performance of a tracked stylus used in 2D pointing tasks (e.g., on a surface) is comparable to that of a mouse [86], so this design decision may yield effective interfaces to virtual environments.

Another approach [43], reminiscent of Mine's work [55], does not require the use of a tablet or secondary display. Instead, the user's non-dominant hand is tracked, and a virtual tablet is rendered registered with the hand. This is based on the premise that it is

sometimes inconvenient to carry a secondary display or other physical prop. Passive haptic feedback is provided by pressing the input device against one's own hand while interacting with widgets displayed on the virtual tablet. Note that many of the approaches described above involve a very strict separation of the 2D and 3D interface components, which may increase cognitive overhead for the user.

2.3.4 *Visual-motor co-location*

Virtual reality (VR) interfaces afford users a tightly coupled loop between input to the system, and the displayed results. The VR interaction metaphor is motivated by the assumption that the more immersive and realistic the interface, the more efficiently users will interact with the system. Ideally, users will be able to leverage existing real-world motor and cognitive skills developed through a lifetime of experience and millennia of evolution, resulting in unparalleled ease-of-use.

A common goal of VR is to create a compelling illusion of reality, wherein the user manipulates objects as in the real world. Consequently, it is often desirable in 3D user interfaces to co-locate the display and motor spaces. This allows users to effectively reach out and grab objects and manipulate them directly. The visual representation of the objects appears to occupy the same space as the user's hand (representation), and dramatically increases immersion in the environment. However, if immersion is not required, conventional input devices such as a mouse can suffice for 3D input [10, 19, 61], and can even outperform 3D devices for conceptually similar tasks [84, 85]. While

this may not qualify as a virtual reality simulation, the performance benefits of the input device may outweigh the benefits of an immersive system in these cases.

Mouse based direct manipulation is a good example of an interface where the display and input spaces are *not* co-located. Similarly, Ware's "bat" input device [97] is an early example of a 3D tracked mouse that is *not* co-located with the environment. The bat was developed on the assumption that, like a mouse, correspondence between the relative movement of the device and movement of objects is more important than direct spatial correspondence. The relative motions of the input device are used to control movement of the cursor (and selected objects). It is unclear if there are measurable benefits of co-locating the display and input spaces, and results of studies examining this effect provide slightly contradictory results.

There is some evidence in favour of co-location. Mine *et al.* [55] suggest that if objects are manipulated within arm's reach, proprioception may compensate for the absence of haptic feedback provided by virtual objects. They used a scaled-world grab that, like the Go-Go technique [64], essentially allows users to extend their virtual arm to bring remote objects close for manipulation. The rationale is that humans rarely manipulate objects at a distance, and stereopsis and head-motion parallax cues are strongest within arm's reach. They conducted a docking study comparing manipulating objects in-hand, versus at an offset distance. They found that participants were able to complete docking tasks more quickly when the manipulated object was co-located with their hand, than when it was at either a constant or variable offset distance. Arsenault and

Ware [4] found that correctly registering the virtual object position relative to the real eye position slightly improved performance in a tapping task, as did haptic feedback. Thus, they argue for correct registration of the hand in the virtual environment.

Sprague *et al.* [76] performed a similar study, but came to different conclusions. They compared three VR conditions with varying degrees of accuracy of head-coupled registration to a real pointing task with a tracked pen. They found that, while all VR conditions performed worse than reality, head registration accuracy had no effect on pointing performance. This suggests that people can quickly adapt to small mismatches between visual feedback and proprioception.

Such adaptation has been extensively studied by perception researchers using the prism adaptation paradigm. In these experiments, prisms placed in front of the eyes optically displace targets from their true position. When one reaches for these objects (or even looks at their hand) there is an initial mismatch between the visual direction of the target and its felt position [34]. However, observers quickly adapt to this distorted visual input effectively recalibrating the relationship between visual and proprioceptive space. Note, however, that temporal delay (i.e., latency) between the movement and the visual feedback degrades one's ability to adapt [34].

Groen and Wekhoven [27] examined this phenomenon in a virtual object docking task, with a VR interface using a head-mounted display and a tracked glove used to control a virtual hand interface. They were also interested if displacing the virtual hand would result in the "after-effects" reported in the prism adaptation literature. As

participants adapt to a visual prism displacement, they gradually adjust (displace) their hand position to match its perceived position. If the visual displacement is eliminated the participant will continue to displace their reach resulting in an after-effect opposite to the initial error before adaptation. Such effects are temporary and participants re-adapt to the non-distorted visual-motor relationship. In other words, participants adapt to the displaced state, then must adapt back to normal afterward. The authors found no significant differences in object movement/orientation time, or error rates between displaced (adapted) and aligned hand conditions. Furthermore, a small after-effect of displaced-hand was reported. This suggests users can rapidly adapt to displaced visual and motor frames of reference in VR.

Ware and Arsenault [93] also examined the effect of *rotating* the hand-centric frame of reference when performing virtual object rotations. Rotation of the frame of reference beyond 50° significantly degraded performance in the object rotation task. A second study also examined the effect of displacing (translating) the frame of reference, while simultaneously rotating it. They found that the preferred frame of reference also rotated in the direction of the translation. In other words, if the frame of reference was displaced to the left, it was also better to rotate it counter-clockwise to compensate.

Finally, other researchers have compared 3D interaction on and off tabletop surfaces, to assess the importance of passive haptic feedback in an environment where the display and input space are coupled [91]. Using a VR workbench, participants performed several object manipulation tasks with their hands, on the tabletop surface, above the

tabletop surface, and with the tabletop surface completely removed. They found that object positioning was significantly faster due to the support provided by the tabletop surface, but that accuracy was slightly worse.

Overall there is some evidence that input/display co-location can improve performance, but the benefits may be minor as people seem able to adapt to relatively small translation mismatches. Rotation mismatches appear harder to reconcile.

2.4 Experimental Evaluation

As discussed earlier, one drawback of previous research is the relative difficulty in generalizing study results, and the difficulty in directly comparing results between studies. This is partially due to different experimental methodologies as well as different measures used. This section details methodologies frequently used in the evaluation of 2D computing pointing devices. These include Fitts' law, ISO 9241-9, and various other measures used to better explain fundamental pointing motions. Although these methods are commonly used in evaluating 2D pointing devices, they see far less use in the evaluation of 3D devices. Adapting these tools in 3D selection/manipulation interface evaluation may prove useful, and allow fine-grained analysis of the simple motions that make up the more complex 3D tasks.

2.5 Fitts' Law

Fitts' law [25] is a model for rapid aimed movements:

$$MT = a + b \cdot \log_2(A/W + 1) \qquad (1)$$

where *MT* is movement time, *A* is the amplitude of the movement (i.e., the distance to the desired targets), and *W* is the width of a target. The log term is the Index of Difficulty (*ID*), which is commonly assigned a unit of bits:

$$MT = a + b \cdot \mathrm{ID} \qquad (2)$$

The coefficients *a* and *b* are determined empirically for a given device and interaction style (e.g., stylus on a tablet, finger on an interactive tabletop).

The interpretation of the equation is that movement tasks are more "difficult" when the targets are smaller or farther away. Fitts' law has been used to characterize the performance of pointing devices and is one of the components of the standard evaluation in accordance with ISO 9241-9 [38]. Indeed, if the movement time and determined *ID* are known, then their ratio gives the throughput of the input device in bits per second (bps).

2.6 ISO 9241-9

ISO 9241-9 [38] employs a standardized pointing task based on Fitts' law, see Figure 2-13. The standard uses *throughput* as a primary characteristic of pointing devices [6]. Throughput (*TP*) is calculated over a block of trials, and is defined in bits per second as:

$$TP = \frac{\log_2\left(\frac{A_e}{W_e} + 1\right)}{MT}, \quad \text{where} \quad W_e = 4.133 \cdot SD_x \qquad (3)$$

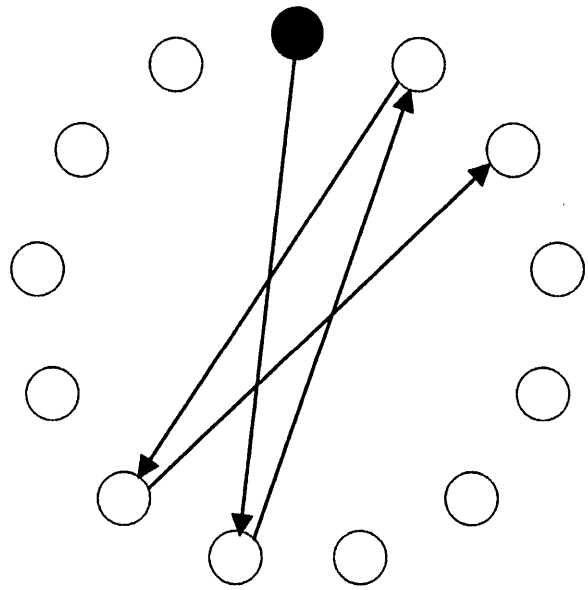

Figure 2-13: ISO 9241-9 reciprocal tapping task with thirteen targets. Participants click the highlighted target, starting with the top-most one. Targets highlight in the pattern indicated by the arrows.

Here, the log term is the *effective* index of difficulty, ID_e, and MT is the measured average movement time for a given condition. The formulation for ID_e is similar to ID in equation (1), but uses the *effective* width and amplitude in place of W and A. This accounts for the task users *actually* performed, as opposed to the task they were presented [49]. Usually larger targets are hit more frequently, and relatively closer to their centers. Smaller targets are missed more often, and with comparatively higher magnitude errors. As an illustration, Figure 2-14 depicts the distribution of hits when a task is performed repeatedly.

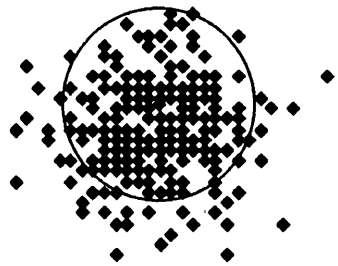

Figure 2-14. Distribution of clicks on a circular target [83] .

SD_x is the standard deviation of the over/under-shoot to the target, projected onto the task axis (the vector between subsequent targets) for a given condition. The effective measures assume that movement endpoints are normally distributed around the target centre and 4.133 (±2.066) standard deviations (i.e., 96%) of clicks hit the target [39]. W_e corrects the miss rate to 4%, enabling comparison between studies with differing error rates [49]. A_e is the average movement distance for a given condition.

Throughput incorporates speed and accuracy into a single measure, and is unaffected by speed-accuracy trade-offs [51]. For example, compare a user who works quickly, but misses many targets, with a highly precise user who always hits the target – the second is effectively performing a more difficult task. Alternatively, if every hit is just outside a target, the user is effectively hitting a slightly larger target. Effective measures are computed across both hits and misses to better account for real user behaviour, and thus enable more meaningful comparison.

2.6.1 Effective Width and Effective Distance

To calculate effective measures, the actual movement vector is first projected onto the intended vector and the difference of the vector lengths is used as the deviation from the intended center. A similar approach is used for the distance: the actual movement distances are measured, and then averaged over all repetitions, thus forming the effective distance, see Figure 2-15 Finally, both effective distance and effective width, in combination with the movement time, are used to determine the throughput of a device, computed according to equation 3, above. This yields a performance measure that, as mentioned above, considers both the speed and accuracy of target acquisitions.

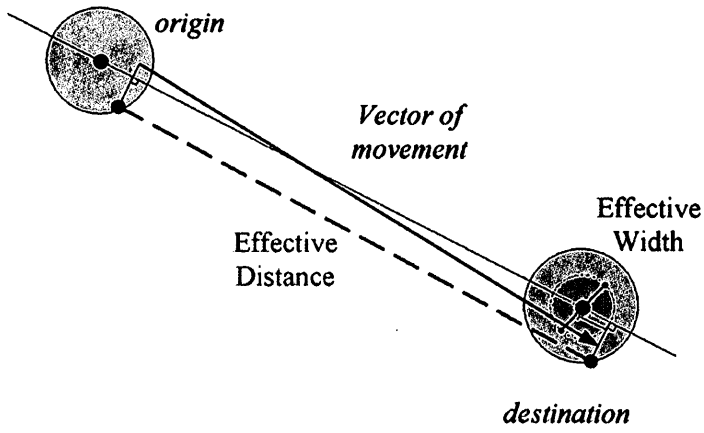

Figure 2-15: Illustration of effective width and effective distance. Note that these are averaged over multiple trials.

These effective measures are used in place of the presented target widths and amplitudes to allow seamless incorporation of differing participant strategies that favour

either speed or accuracy [49]. In essence, this approach treats more accurate clicks (i.e., clicks closer to the centre of the targets) as clicks on smaller targets, while the clicks outside of the intended targets are treated as "successful" clicks on larger targets. Hence, throughput becomes the primary characteristic of pointing device performance and accuracy. It is also the measure recommended by ISO 9241-9 to test pointing devices [38].

2.6.2 Fitts' law Extensions to 3D

Fitts' law was developed for one-dimensional aimed motions but works extremely well for 2D motions and is commonly employed in the evaluation of pointing device performance [49]. Straightforward extensions to 3D pointing generally increase the correlation between *MT* and *ID*. Note however that adding *any* extra parameter in a regression analysis will improve the fit of the model for a given dataset [62]. Thus, care must be taken to extend the model with appropriate parameters that generalize to other results as well.

Murata and Iwase [57] performed a pointing task in a 2D plane positioned vertically in front of the user. The task did not involve hitting targets at varying depths, so was not a full 3D task. From this, they derived a directional model for *ID* that incorporated the sine of the angle to the target from the x axis. The authors report a higher correlation between *MT* and their *ID* model compared to the conventional formulation. They also found that movements in upward directions took longer than those in downward directions, possibly due to gravity.

Grossman *et al.* [28] investigated pointing at trivariate 3D targets, i.e., targets varying in height, width and depth. They developed a model that considered the direction of movement as a vector through the target. However, they used only motions to targets positioned on a single "ground plane", effectively a 2D task on a plane parallel to the floor. They validated their model in an experiment using a volumetric display. These results may not extend to other VR systems since, as mentioned in Chapter 3, volumetric displays provide more complete depth cues compared to other systems. Similarly, systems that afford co-location of the display and input spaces may also require different models. Current volumetric displays do not afford this co-location. The 3D model presented by Grossman *et al.* is also inconsistent with the 2D model used by the ISO standard, preventing direct comparison between 2D and 3D pointing.

Although not necessarily a true "3D" task per se, remote pointing with ray-based techniques can be used in 3D object selection and manipulation. Recent attempts at modeling this in a manner similar to Fitts' law have resulted in angular models based on the observation that wrist rotation is more commonly employed in ray-based pointing than hand movement [44]. The model presented by Kopper et al. [44] favours the use of "angular" width and distance parameters. Effectively, this results in targets closer to the user as being easier to hit (effectively larger), due to increased ray precision near the ray origin. While the model was validated for 2D target selection tasks on large screen displays, it theoretically will also work for ray-based selection in virtual environments.

2.6.3 Motion Analysis

While the ISO standard provides a reliable methodology by which to rank input devices/techniques in absolute pointing performance, several authors [48, 52, 59, 73] point out that it does not explain *why* devices perform differently. This is arguably more valuable to know, as it can inform the design of future devices and techniques. Several researchers have proposed methods to answer these questions by investigating characteristics of the pointer/cursor motion during the movement task [48, 52, 59]. The objective of motion analysis is to provide additional measures to help explain the relative differences in performance (typically throughput) observed in pointing studies. Ultimately, these measures complement existing approaches such as analyzing speed, accuracy and throughput.

One approach suggested by Liu [48], Nieuwenhuizen [59] and their colleagues is based on a two-component model of movement proposed in 1899 by Woodworth [102]. According to Woodworth, goal-directed movements such as moving one's hand to hit a target are broken into two phases: a ballistic phase and a correction phase. During the ballistic phase, no sensory input is processed. The motion is "pre-programmed" and a quick movement gets the hand rough vicinity of the target. The correction phase accounts for inaccuracies during the ballistic phase, and effectively homes the hand toward the target using sensory input to verify success.

Liu *et al.* [48] conducted a study comparing aimed movements in reality and virtual reality. They split overall pointing tasks into ballistic and correction phases

according to the criteria proposed by Meyer [53]. According to these criteria, sub-movements are defined by pauses during which the cursor moves no faster than 0.05 times the peak velocity. Any sub-movement that contributes more than 25% to the overall path length is considered to be part of the ballistic phase. Their study used a fish-tank VR system and a wooden replica of the virtual scene for comparing movement toward real targets to virtual targets. Results of the study indicated that overall, the same movements were significantly slower (about 2.5 times) in the VR system than in reality. The division of these overall movements into phases revealed that ballistic movements were about 70% longer in virtual reality, but the correction movements were around 6 times longer. The authors posit two possible explanations for this: either the ballistic movements in VR resulted in a greater distance from the target (necessitating a longer correction phase), or conflicting depth cues made it more difficult to perform the correction phase. These explanations may be related.

Nieuwenhuizen *et al.* [59] extended this work and proposed a suite of new measures to investigate differences between real and virtual pointing using this model. In addition to commonly used measures such as movement duration and target misses, these included path length, average speed, path efficiency (ratio of path length to optimal path length), number of sub-movements, correction distance (distance to the target at the start of the correction phase), and pause time (mean duration of pauses in the correction phase). They conducted two experiments. The first of these examined differences in pointing at real targets (wooden cylinders) to performing the same task in a fish tank

virtual reality system. The second experiment looked only at virtual pointing, but accounted for previous experience with VR systems by comparing the performance of novice participants to experienced users. The results of both studies were broken down into the ballistic and correction phases, and then analyzed using the aforementioned new measures. Of particular note, overall movement was found to be slower in VR than in reality. Both the ballistic phase and correction phase took longer in VR, but for different reasons. The ballistic phase was longer due to slower average movement in the VR system. However, the correction phase was longer due to lengthier pauses between corrective movements, rather than speed differences. User practice significantly improved speed, path length, and efficiency in the ballistic phase, but not the correction phase. The correction phase benefitted from shorter pauses with more experienced users. The authors conclude that pointing facilitation techniques should focus on the correction phase, as this improves more slowly with practice. This is supported by Liu *et al.*'s earlier results [48]. It is possible that increased duration of these pauses may be due to latency in the tracking system, as this should affect only the correction phase.

Slocum *et al.* [73] also examined the two-component model for motions, and looked at peak movement velocity and proportion of distance travelled during the ballistic phase. Their study showed a significant correlation between these measures and throughput, suggesting that high efficiency pointing devices support fast and precise ballistic movements, rather than relying on correction movements.

While the aforementioned studies demonstrate that there is clearly value in examining the individual movement phases that make up an overall pointing task, one issue with these measures is the somewhat arbitrary nature of the parsing criteria. Although the authors filter the movement data to avoid detection of spurious movements (e.g., tracker jitter) as sub-movements, changing the parsing criteria (i.e., definitions of pauses) can potentially change the results. An alternative approach proposed by MacKenzie, Kauppinen and Silfverberg [52] uses objective measures to characterize movement tasks.

Some of their measures include target re-entries, task axis crossings, movement direction changes (both parallel and orthogonal to the movement direction), and movement variability. Target re-entries are a count of how many times (after the first) the cursor enters the target "hot spot". Such a measure might be extremely helpful in evaluating effects of feedback mechanisms, as they give an objective measure of how easily hit a target is [52]. Task axis crossings indicate the number of times the cursor path crosses the ideal path from one target to the next. This is related to the other scores, as the optimal path between two targets is a straight line, although this typically does not have to be followed when pointing at targets. Direction changes indicate how often the cursor movement changes direction either parallel to the movement direction or orthogonal to it. Similarly, movement error is the standard deviation of distances from the task axis to sampled points along the path. This gives an indication of how close to the ideal path the cursor was moved.

In addition to measuring differences due to device characteristics, these measures have also been used in research on motor impairments [42]. These may also be useful in evaluating 3D input technology that has several unique characteristics. For example, systems with high jitter will likely demonstrate higher movement variability. Similarly, target re-entries can help indicate problems due to imperfect (or missing) feedback mechanisms. The absence of haptic feedback, for example, has been shown to increase target re-entries, as participants have to "hunt" for the correct depth of a target in the presence of imperfect stereo depth cues [82]. These measures have not yet been examined in detail in 3D pointing tasks. Some, like movement variability and target re-entries, have obvious and natural extensions in 3D. However, other extensions such as task axis crossings are less obvious. Some such 3D metrics have been proposed [18], but these are not linked to performance.

2.7 Summary

Overall, there has been a great deal of work on 3D selection and manipulation interfaces under a variety of conditions. Such conditions include stereo and head-tracking, issues which are investigated further in this dissertation. A drawback of previous work is that different studies are not comparable. This dissertation thus proposes to use Fitts' law and the ISO 9241-9 standard to improve comparability between techniques and experiments. By investigating fundamental pointing motions, rather than more complex 3D tasks, the confounding influence of user strategy is mitigated. This also improves the directness

with which one can compare techniques – pointing motions are inherently more similar than more complex 3D manipulation techniques.

Chapter 3
Evaluating Latency and Jitter

This chapter includes two experiments investigating the effects of latency and jitter on human performance with 3D input devices. The first experiment employed Fitts' law, a well-established model of pointing device performance. Fitts' law is inherently 1-dimensional and works well with 2D pointing. However, it does not apply well to the modeling of 3D movements. Consequently, the first study, Experiment 3-1, uses only 2D pointing tasks using both a 3D tracker and mouse under a variety of latency/jitter conditions. The mouse is included as an exemplary low-latency, low-jitter condition, and latency and jitter are artificially added in some conditions to match those of the tracker.

To compare motions captured by the mouse optical sensor to those captured by a 3D tracking system, a tracker was physically mounted on the mouse using a rigid wooden frame. This effectively constrains the tracker to 2D operation, which as an added benefit seems to present a fair point of comparison between the mouse and tracker; the tracker effectively behaves like a mouse. The second experiment, Experiment 3-2, examined the effects of latency and jitter on 3D positioning. It used a subset of the conditions from the previous study.

3.1 Research Questions

The primary objective of the research presented in this chapter was to assess if latency or jitter have an impact on pointing performance in both 2D and 3D tasks. The goal of

Experiment 3-1 was to determine, all else being equal, the effects of latency and jitter, and to quantify the differences in device performance. In other words, which has a stronger impact on human performance: latency or jitter? The goal of Experiment 3-2 was to determine if 3D task performance using 2D input devices can be predicted by 2D models of performance such as Fitts' law.

3.2 Characterizing System Latency and Jitter

End-to-end system latency and jitter were characterized for both the mouse and the 3D tracking system. So-called "latency-jitter" was also considered, i.e., the amount of *change* in latency from one point in time to another. To measure this, both the mouse and tracker update frequency were examined. The tracker updates at 120 Hz [58] and the mouse at 125 Hz. A histogram of these times showed that more than 99.5% of the updates happen within 8 – 11 ms of the previous sample, which is in line with these reports. Almost all of the remaining samples follow within 5 – 8 ms. Hence, latency jitter is minimal in these experiments, which instead focus only on latency and spatial jitter.

3.3 Characterizing Latency

A variation of Mine's method was used to characterize the lag of both the mouse and the tracker [54].

3.3.1 Equipment Setup

NaturalPoint's *Optitrack* [58], a camera-based optical 3D tracking system was used as the three-dimensional tracker. This system uses digital video cameras linked to the computer via USB. The cameras perform an on-board image threshold operation (i.e., before

transmission), thus reducing both bandwidth demands and processing requirements on the host system. This setup used three NaturalPoint *Flex:C120* cameras mounted on a rigid metal frame, shown in Figure 3-1.

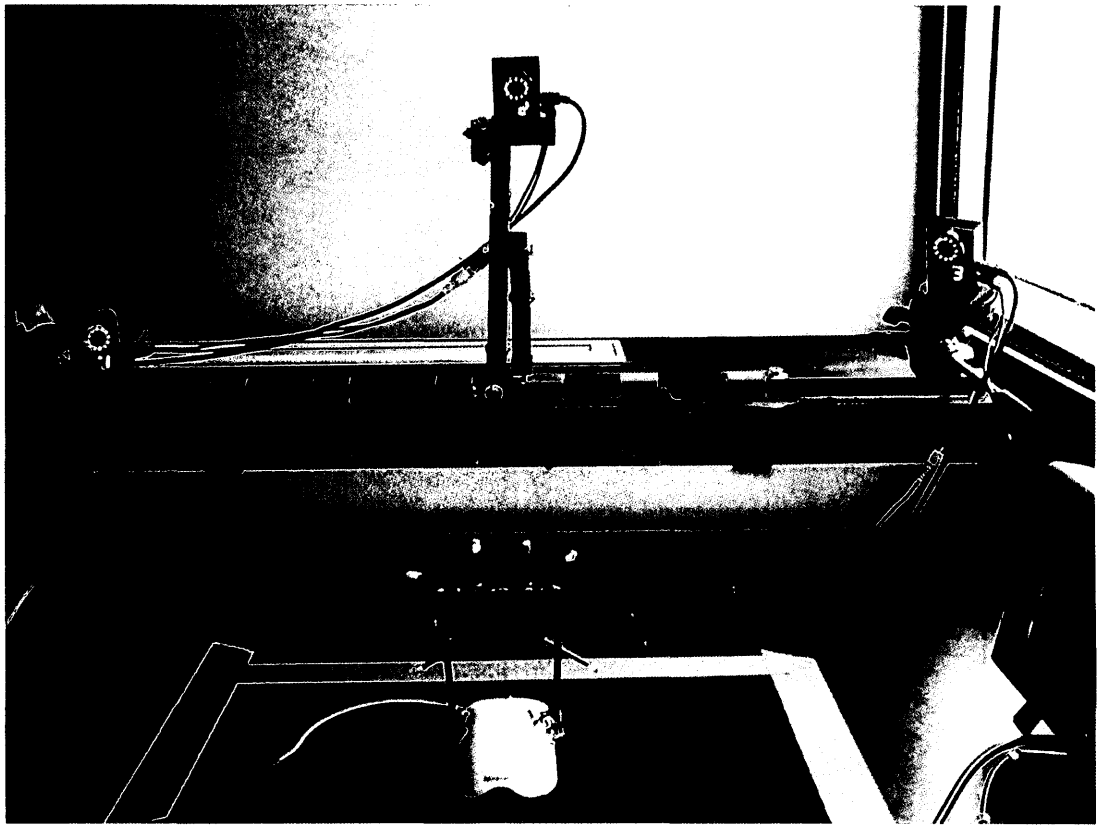

Figure 3-1. NaturalPoint cameras mounted on metal frame. The tracked mouse is also visible in this figure.

The cameras also contain infrared illuminators. Coupled with the cameras' ability to be synchronized and logically organized into an array, this creates an object tracking solution capable of recognizing emissive or retro-reflective clusters of dots on existing input devices. The NaturalPoint *Point Cloud* and *Rigid Body Toolkit* software then perform calibration and real-time 6DOF motion capture of *rigid bodies* within the

overlapping fields-of-view of the cameras. In these experiments, the tracked rigid body consisted of six markers mounted above the mouse.

For the latency measurement, a physical pendulum was suspended in front of the display (Figure 3-2a). The pendulum arm was a rigid metal rod, and the pendulum head was made of hard Styrofoam. Tracking markers placed near the center of the pendulum defined a tracked body. The tracking system cameras were positioned to cover the working area from multiple angles.

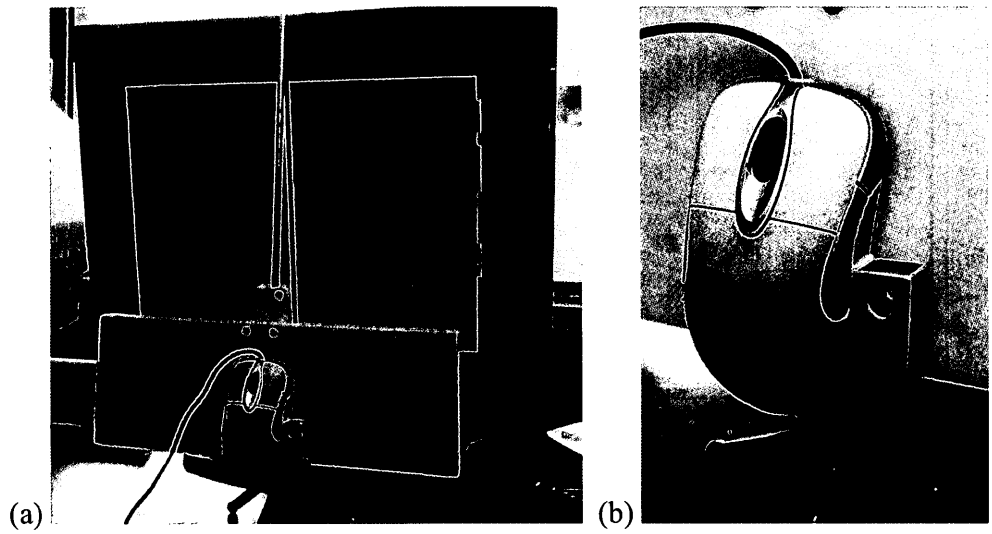

Figure 3-2. (a) Pendulum setup in front of display. (b) Mouse affixed to tripod used in mouse latency measure.

The mouse latency was measured with the same hardware configuration. A Microsoft optical mouse was affixed to a tripod positioned in front of the pendulum. The optical sensor of the mouse was pointed toward the pendulum, approximately 0.5 mm away from the Styrofoam surface. This distance was sufficient to allow the mouse to

sense the pendulum movement, but without rubbing against it and thus reducing its movement due to friction.

A 21″ CRT display was used, with a resolution of 800 x 600 pixels and a 120 Hz screen refresh rate. A digital camera was used to record the experiment at a frame rate of 60 Hz. The optical tracking system was positioned in front of the display, pointed toward it and the pendulum. The digital camera was positioned immediately behind the tracking system

3.3.2 Software Setup

The software drew two lines on the screen. The origin of the lines was registered off-screen with the pivot point of the pendulum (about 5 cm above the monitor). In the resting position, the ends of the lines were positioned directly behind the pendulum, near the center of the screen. As the pendulum swung, the ends of the two lines moved in accordance to its motion, as perceived by the mouse or the tracking system. The line origins remained stationary.

3.3.3 Procedure

The pendulum was extended by hand and released. It then freely oscillated at approximately 0.8 Hz. Depending on which device was being measured, the motion of the pendulum was detected either by the retro-reflective markers placed on it (via the cameras) or by the mouse optical sensor brushing against the Styrofoam surface. The movement of the lines displayed on the screen corresponded to the detected motions. Movement of both the pendulum and of the lines was recorded with a digital video

camera. Latency was calculated from the time delay between the pendulum motion and the line motion on the screen.

3.3.4 Analysis and Results

Approximately two minutes of video were recorded with the digital camera. This video was analyzed manually after the experiment to derive the end-to-end latency for each device.

Peaks of pendulum movement were examined. When the pendulum reached the peak of its movement in one direction or the other, the frame number and its time were noted. When it began to swing back the other way, the mouse and tracker lines would swing back as well, but after a short delay due to tracking latency. These delays were recorded.

As the camera was only recording at 60 Hz, a total of 10 measurements were performed for each device. This improves the precision of the measurement. Ultimately, the average delay of the mouse relative to the pendulum was 35 ± 2 ms, and the average delay of the tracker was approximately 40 ms larger, or 73 ± 4 ms. These two values were used as the latency measures of the respective devices.

3.4 Characterizing Tracker Jitter

Another potentially critical difference between the mouse and the tracking system is the spatial jitter in position measures. When controlling a cursor, the tracking system exhibits noticeably more jitter than the mouse, which is virtually jitter-free. This too was

measured, to account for differences in the experimental devices and to compensate in some conditions.

Note that, although the optical sensor of the mouse may be subject to some jitter, this appears to be filtered in the mouse hardware. While the technical details in each specific implementation may differ, typical optical mouse sensors are, in essence, low-resolution miniature video cameras taking images at a rate of several thousand per second [1]. Since a desktop pointing device only requires about a hundred updates per second, the 10:1 or greater excess of frames is likely used to smooth the device operation via averaging or some other filtering technique.

Note also that jitter due to hand tremor is not an issue in these experiments as resting the mouse on a physical surface largely eliminates it. This is because hand tremor, like any other mechanical oscillation, depends on friction, as well as mass, rigidity, and external disturbances. Friction *dampens*, or reduces the magnitude of the oscillations. Unlike "free-space" 3D input devices, the tracker used in these experiments was constrained to the surface by affixing it to the mouse. Hence, most hand jitter was eliminated for the tracker as well, leaving only *device* jitter. Similarly, the mouse is assumed to have no noticeable jitter of either kind (hand tremor, or device noise).

3.4.1 Equipment Setup

Tracker spatial jitter was characterized using predictable, repeatable motions. The differences between the camera-observed motions and the expected motions, were used to derive the amount of jitter. Three specific conditions were measured: (1) circular

movement of the rigid body in the horizontal plane, (2) circular movement in the vertical plane facing the cameras, and (3) linear movement along two perpendicular axes on the horizontal plane.

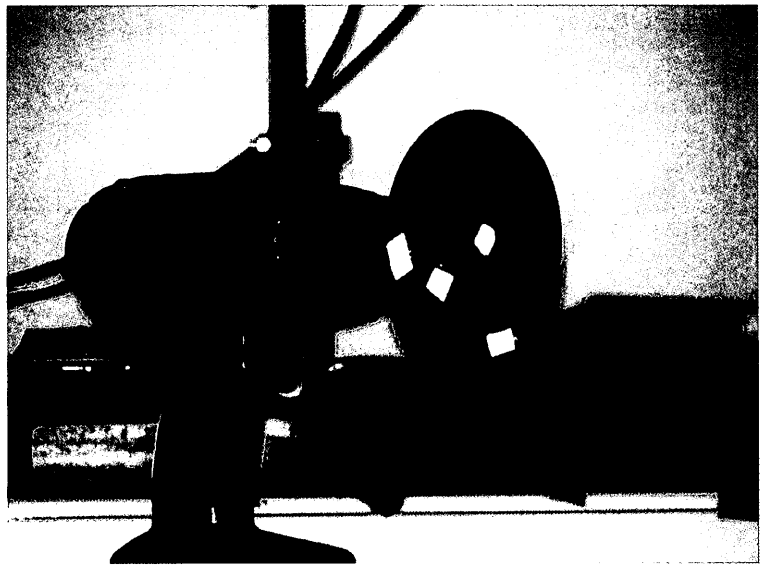

Figure 3-3. Rigid body mounted on a drill.

In the first case, reflective markers were attached to the turntable of a gramophone record player. The lowest available speed of 16 2/3 rpm (0.28 s^{-1}) was selected. In the second case, a cordless power drill was used. The reflective markers were glued to a surface of a compact disc, with the disc clamped to a metal bolt and mounted into the chuck of the drill. The speed was adjusted to the lowest possible, approximately 0.5 s^{-1}. Figure 3-3 shows this condition. For the last condition, the tracking markers were mounted on a wheeled platform, which was moved by hand along a rail during the experiment, at a speed of ~1 m/s.

3.4.2 *Analysis and Results*

The recorded motions included circular movement, which resulted in regular, sinusoidal, changes of the coordinates in the rotational platform conditions. The predictable component of this motion can be subtracted from the signal by applying a high-pass filter. Doing so leaves only the jitter, i.e., the fast-changing component of motion. Root mean square (RMS) jitter values are visualized in Figure 3-4.

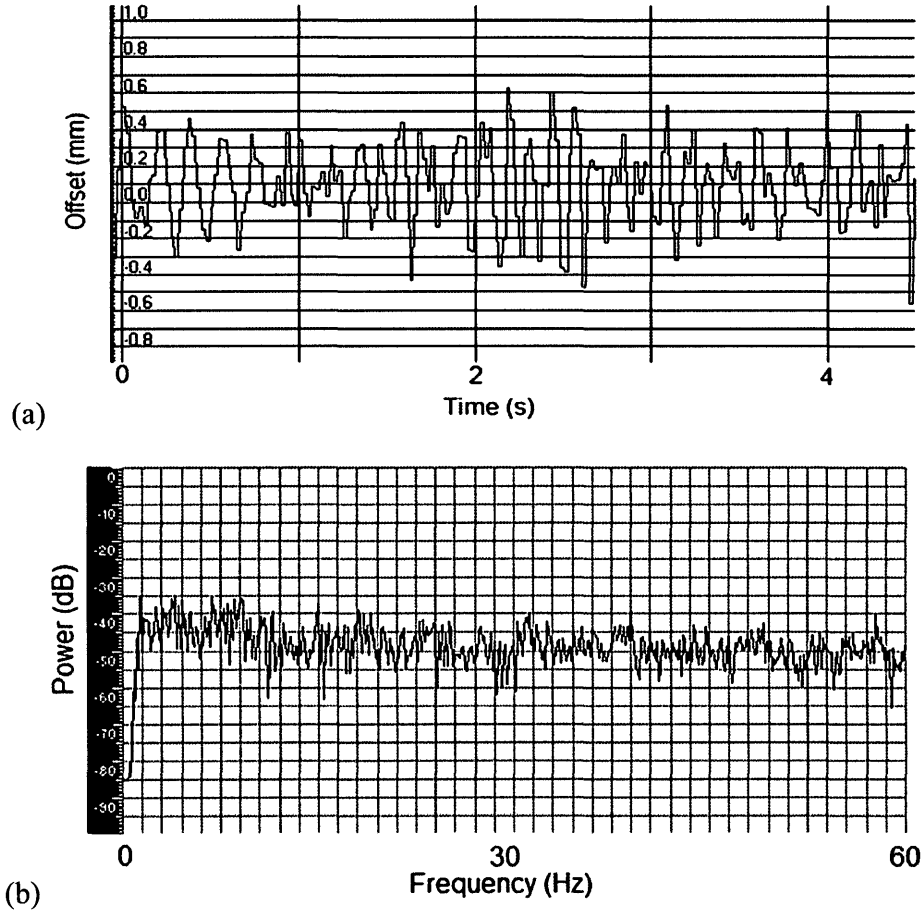

Figure 3-4. Spatial jitter of the Optitrack tracker. (a): fragment (~1 s long) of jitter displacement in mm; (b): FFT of the recorded data, logarithmic response in dB, frequencies (linearly) from 0 to 60 Hz, low frequency regular motion filtered out.

The tracker jitter mostly resembled white noise in all three examined motions, with 0.3 mm RMS. There were occasional spikes ("outliers") in the measurements. While such spikes have little effect on the frequency content and overall strength, they may have detrimental effects on performance due to their short duration, especially during high precision tasks.

3.5 Experiment 3-1 (2D Pointing)

This experiment used the ISO 9241-9 standard to evaluated 2D pointing performance of the devices.

3.5.1 Participants

Fourteen students (aged 18 to 30; mean 27.2 years) were recruited to participate in the study. Eight were male. All used the mouse with their right hand during normal computing. Participants were paid $10 upon completion of the study, which took about 1 hour. Additional participant demographic information can be found in the Appendix.

3.5.2 Apparatus

The computer was an AMD Athlon with a 64-bit CPU, running at 3 GHz, with 1 GB of RAM and a PCI-Express graphics controller. As discussed earlier, a Microsoft optical mouse was augmented with a set of retro-reflective markers and was used in all conditions (see Figure 3-5). Some conditions used the mouse optical sensor, while others used the tracker instead. Mouse acceleration was disabled in the Windows OS, and gain

was set to be 1:1 for better comparability with the tracker. While tracker jitter was carefully measured as described above, the mouse was assumed to have no jitter, as none was noticeable.

Figure 3-5. Mouse with optical tracking markers mounted.

The software was written in C# and used NaturalPoint's tracking API to enable the capture of the motion of the rigid body mounted on the mouse. The software implemented the 2D pointing task described in ISO 9241-9 [38] (see Figure 2-13). The software presented 13 targets in a circle. Upon clicking the first highlighted target (at the top) the timer starts and the opposite (bottom-left) target is highlighted, directing the participant to select it. The next target is on the opposite side, to the immediate right of the first target, and so on until all targets are clicked. The software logged target sizes, distances between targets, the times to click between targets, errors, and screen

coordinates of click events. It also performed the effective width calculation as described in Section 2.6.

3.5.3 Procedure

After signing an informed consent form, participants were seated at the computer display. The tracked mouse was positioned initially at the origin of the tracked region (the bottom left corner of the taped square in Figure 3-1).

Participants were given a brief introduction to the system, and allowed to try the system and find the most comfortable seating position. The difference between the various input modalities were explained to them. After that, they were instructed to click on the highlighted targets as quickly and accurately as possible.

3.5.4 Design

This experiment had one independent variable, input modality, with seven levels. Five of these used the mouse, and two used the tracking system.

In addition to the baseline mouse condition, M, the mouse-based input modalities involved artificially adding latency and/or spatial jitter. Two of these had increased latency only. One, ML, had latency that matched that of the tracker. The other, M225, had 225 ms of latency. This high latency condition was introduced to correlate results with previous work [50]. A fourth mouse-based condition, MJ, included increased spatial jitter to match the tracker RMS value, but did not include additional latency beyond that already present in the mouse conditions. The final mouse-based condition, MT, was a

"tracker emulation" mode, where both latency and jitter matched the tracker. The jitter in the MJ and MT modalities was calculated based on mouse sensitivity (mm/pixel), and randomly generated to match the measured tracker jitter. The tracker-based conditions used either relative movement, TR (subject to clutching, like a mouse), or absolute movement, TA (tracked in the air if clutched). These are summarized in Table 3-1.

Input Modality	*Name*	*Approx. Total Latency (ms)*	*RMS Jitter (mm)*	*Movement Mode*
M	Mouse	35	–	Relative
ML	Mouse + 40 ms latency	75	–	Relative
MJ	Mouse + jitter	35	0.3	Relative
M225	Mouse + 190 ms latency	225	–	Relative
MT	Mouse + 40 ms latency + jitter	75	0.3	Relative
TR	Tracker, relative	75	0.3	Relative
TA	Tracker, absolute	75	0.3	Absolute

Table 3-1. Summary of input techniques used in Experiment 3-1.

The input modality ordering was determined by a Latin square within each block. Additionally, half the participants used all devices in the reverse order to complete the counterbalancing.

All devices were tested under three target amplitudes (320, 450 and 640 pixels) and three target widths (12, 25 and 64 pixels). These conditions represented nine *IDs*, and were randomly ordered (without replacement) within a block. Note that *ID* was not analyzed as an independent variable, but rather was varied to ensure a realistic range of

task difficulties for the purpose of the throughput computation. Hence the only factor of interest in the following analysis was the input modality.

Each participant completed two blocks of trials. Hence, each participant completed 7 input modalities × 9 *ID*s × 2 blocks, for a total of 126 rounds (target circles). Across all 14 participants and 12 recorded target clicks (trials) per round, this gave a total of 126 × 14 × 12 = 21,168 trials. Note that the first target click in each round was not recorded, as it started the timer; this allowed participants to take breaks between rounds.

The dependent variable was device throughput (in bits per second), calculated as described earlier. Movement time and error are excluded here, as they are not statistically analyzed.

3.6 Results & Discussion

3.6.1 Throughput

Results were analysed using one-way ANOVA. There was a significant main effect for input modality on throughput, ($F_{6,84} = 38.8$, $p < .0001$). Figure 3-6 shows the throughput of the seven input modalities. The throughput of the baseline mouse condition is similar to that reported in previous work [75]. This helps validate the experimental design.

A Tukey-Kramer post-hoc analysis ($p < .05$) revealed three groupings of modalities, with no evidence of statistical difference in throughput within each group. The M and MJ conditions were the most efficient, and the M225 condition the least efficient; the rest are approximately equal. Throughput scores for all conditions are depicted in Figure 3-6.

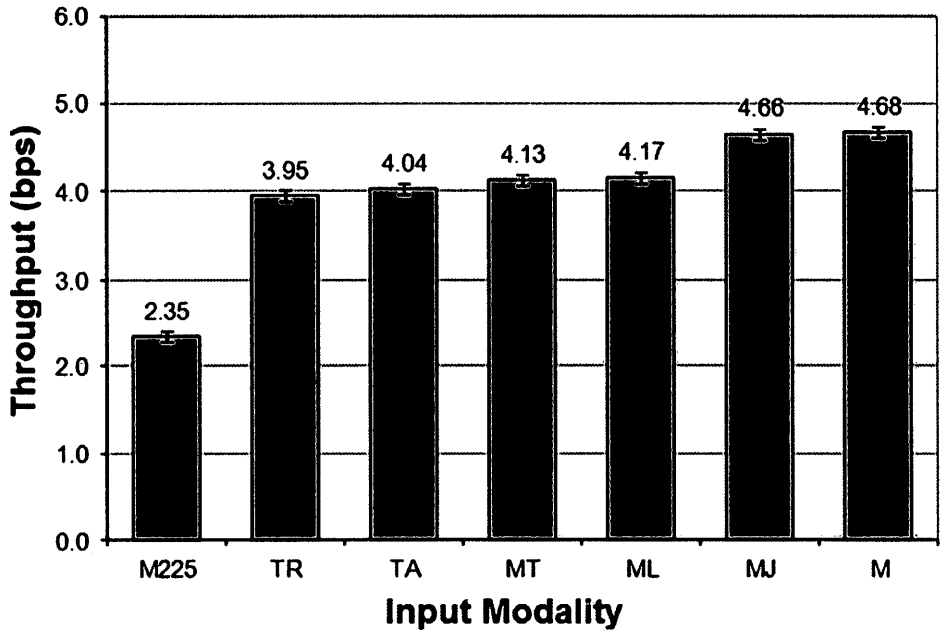

Figure 3-6. Throughput for all conditions, higher is better. Error bars represent ±1 std. error. Bars are ordered to highlight groupings.

Movement Time and Positional Error

Average movement time for the M and MJ modalities was around 990 ms. The middle group of modalities (ML, MT, TA and TR) had an average movement time of 1145 ms. Finally, the average movement time for the M225 modality was 1945 ms. The mean positional error (i.e., how far from the target centre the cursor was) across all conditions was about 6 pixels. Note that movement times are not statistically analyzed. These values are provided solely for comparison with the following experiment, which cannot be analysed in terms of throughput.

Latency

Comparing the TR, TA, MT and ML modalities – all with approximately the same lag – to the mouse indicates that the relative performance cost of 40 ms latency is around 15%. The M225 condition had the worst performance, with about 50% lower throughput. For varying ID_e, these results are similar to those observed by MacKenzie and Ware [50], see Figure 3-7. ID_e was computed using equation 3 in Section 2.6 as $\log_2(A_e/W_e + 1)$, i.e., using effective values for W and A.

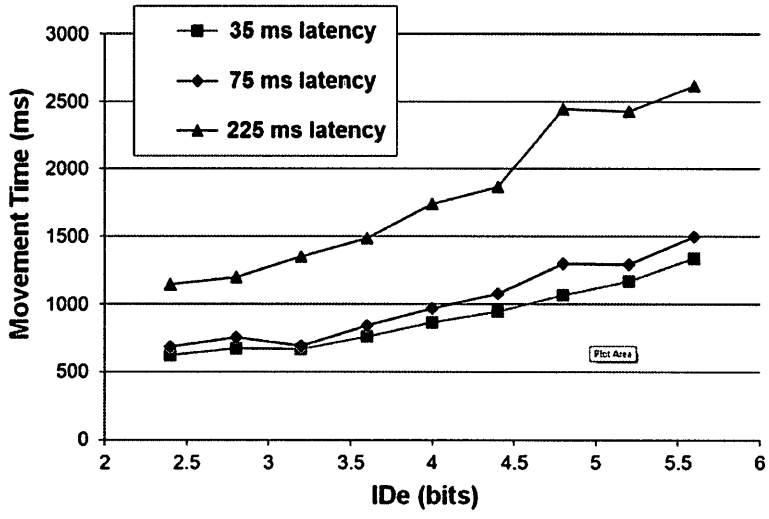

Figure 3-7. Movement Time as a function of ID_e for each latency condition.

Jitter

Examining the conditions with and without jitter, it appears spatial jitter alone did not have a significant effect on throughput. The MJ condition, with extra jitter, but no additional latency, was not significantly different than the mouse (M) condition. Like the mouse condition, it too was significantly better than both the ML and MT modalities.

Absolute vs. Relative Tracker Movement

According to the post-hoc analysis, throughput scores for the TA and TR modalities were not significantly different ($p > .05$). This can also be seen in Figure 3-6. It is possible that one reason no difference was found was that the speed of cursor control (CD gain) in all conditions was high enough to eliminate clutching. This includes the mouse conditions, as the mouse gain was set to match the tracker. Moreover, as participants tended not to lift the device during the experiment, the difference between these conditions should not be noticeable.

Similarly, the post-hoc analysis also revealed that throughput scores for the MT, TR and TA modalities were not significantly different ($p > 0.05$).

3.6.2 Summary

The results above demonstrate that a mouse with added jitter and lag performs very similarly to the tracker having the same measured jitter and lag. Thus, when constrained to 2D operation, is appears unlikely that any additional factors beyond jitter and latency affect tracker performance. In other words, analyzing the sensing technology in isolation from all other differences between the two input devices suggests that these two factors matter most, with latency having a great impact than small amounts of jitter.

3.7 Experiment 3-2 (3D Movement)

This experiment attempts to extend the results of Experiment 3-1 to 3D object movement using a constrained 2D-3D movement mapping.

3.7.1 Participants

Twelve people participated in the experiment, with ages ranging from 19 to 30 (mean age 24 years). Six were male. Participants were paid $10 for completion of the study, which took approximately 45 minutes. Additional participant demographic information can be found in the Appendix.

3.7.2 Apparatus

The tracked mouse from the previous experiment was used. This experiment used custom 3D graphics software written in C++ with OpenGL. The software was developed for a mouse, with extensions for 3D tracking. It uses a 3D movement technique relying on mouse-based ray-casting. The technique requires only 2DOF from the input device, which is mapped to 3DOF movements. Depth is handled automatically: the software slides objects along the closest surface behind their projection as they move through the scene [61].

3.7.3 Procedure

After signing informed consent forms, participants were seated in front and to the right of the monitor, and shown how to use the system. They were shown how to use the 3D movement technique, and given a practice trial to familiarize them with the task.

The task involved moving twelve unit cubes from a circle in the centre of a plane to twelve corresponding pillars positioned in a circle at a radius of 20 units. This was designed to simulate the ISO 9241-9 task used in Exp. 3-1 in a 3D setting. The height of

the pillars varied to add a third dimension to the task. Consequently, while the distance moved in screen coordinates would be similar for each cube, the 3D distance varied more. Head-tracking was not used; the viewpoint was fixed to eliminate viewpoint control as a potential confounding factor.

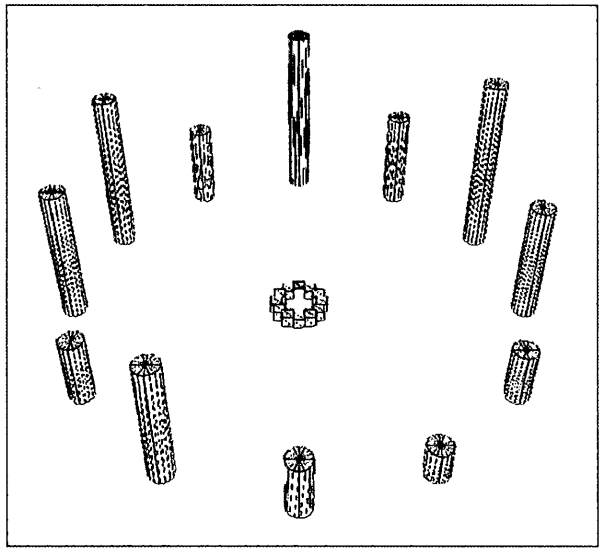

Figure 3-8. Task for Exp. 3-2 with fixed viewpoint. Participants moved each cube to the corresponding pillar on the periphery, starting with the red cube at the "noon" position. Pillar heights, diameters and positions were constant throughout the experiment.

3.7.4 Design

This study had one independent variable, input modality, with four levels. Four of the input device modalities from Experiment 3-1 were re-used. These were M, ML, MT and TA. Participants performed 10 blocks of trials. Participants completed 4 input modalities ×10 blocks. Given that there were 12 object movements per round and 12 participants, a total of 4×10×12×12 = 5760 trials were recorded.

The dependent variables were object movement time (in ms) and error. Error was measured both in screen coordinates (pixels away from ideal position) and 3D distance (units away from ideal 3D position); both error measures were included, since a small distance in screen coordinates could yield a significant change in 3D coordinates due to the nature of the sliding algorithm.

3.7.5 Results

Average Movement Time

Input modality had a significant main effect on object movement time ($F_{3,11} = 40.44$, $p < .001$). Tukey-Kramer post-hoc analysis revealed no significant difference between any of the mouse modalities; see bars M, ML and MT in Figure 3-9. However, the TA condition was significantly slower than any mouse modality, about ~30%.

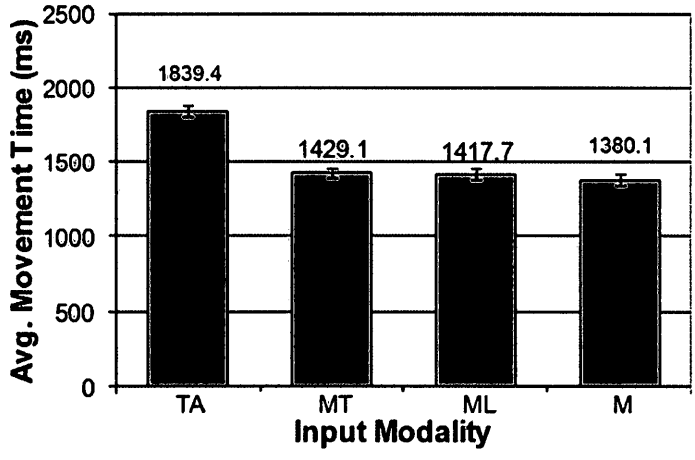

Figure 3-9. Average movement time, with standard error bars. Note this graph cannot be directly compared to Figure 9. Also, the results for the M condition are not comparable (see text).

Positioning Error

No significant difference in positioning error was found for either 2D error ($F_{3,11} = 0.56$, ns) or 3D error ($F_{3,11} = 0.96$, ns). The grand mean 2D error was 7.16 pixels, only slightly larger than in Exp. 3-1; in 3D it was 0.44 units, which corresponds to about half a cube width.

3.8 Discussion

At first glance, the results of the Exp. 3-2 appear to contradict those of Exp. 3-1, which yielded no significant differences between the tracker-like and the tracker conditions. Yet, the results of Exp. 3-2 indicate about a 30% difference between these conditions. According to an in-depth analysis, the difference is likely due to jitter "spikes" which are present in the tracker output, but not in the mouse. These spikes have much higher cost in the 3D task than the 2D task.

As previously mentioned, the tracker signal noise had comparatively large spikes in approximately 1% of the samples. While this does not affect the RMS of the tracker jitter, the performance penalty can be dramatic, especially if these spikes occur at inopportune times such as when placing an object on the target pillar. Figure 3-10 compares the magnitude of the spikes to our simulated jitter.

The performance cost of errors is higher in the 3D task than in the 2D task, since the magnitude of the 3D error can grow much more than the 2D distance moved. If a jitter spike causes the object to miss the target pillar and thus fall onto the background

plane, a lengthier correction is necessary. Effectively, a 1-pixel error in screen coordinates can map to an arbitrary drop along the corresponding 3D ray.

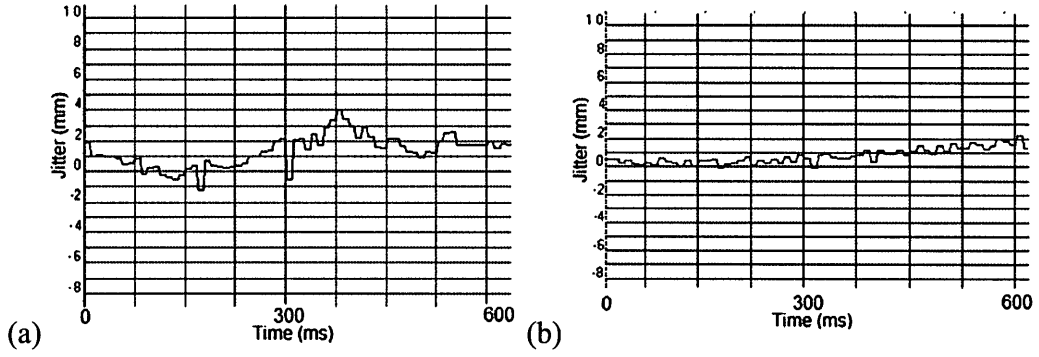

Figure 3-10. (a) Jitter spikes in tracker position outputs; (b) for comparison – the response of the mouse with the added jitter in the same area.

This is supported by the lack of significant difference in accuracy between conditions for the 3D task. This likely occurred because most errors were corrected, as suggested by the relatively low 3D error (less than half a unit of distance). Also there are strong visual cues (perspective and in some cases, occlusion) as to whether the object was in a correct position, making it easy to detect and correct errors. With the tracker, such misses appear to have happened more often, due to the jitter spikes mentioned above. The correction time contributed to the observed differences in the movement time. However, due to the corrections, there was no significant difference in accuracy. In contrast, corrections were not possible in Exp. 3-1, where each trial concluded upon clicking (whether it was a hit or miss).

Analysis of the 3D motion paths also supports this. Most errors occurred on pillars around the back of the circle. Due to the perspective distortion, errors in that

region also required the largest correction. Examination of the motion paths indicated that after making such errors, participants moved the object around to the front of the pillar and slid it up the front again, incurring a relatively large time penalty. Effectively, participants were trading speed for accuracy; while throughput accounts for this in the 2D study, the measures used in the 3D study do not.

Another issue with Experiment 3-2 was that the mouse condition was not significantly different from the mouse with latency, as in Experiment 3-1. Analysis of precise timing data revealed that the base mouse condition (and only that) suffered from higher than expected latency as well as latency jitter. On average, the base mouse condition had 15 ms extra latency, which partly explains the performance drop. Moreover, latencies exhibited a roughly bimodal distribution around 12 ms (70%) and 24 ms (30%). This likely explains the remaining performance loss. The problem was traced to timing limitations of the underlying software framework used in Exp. 3-2. However, the data from the mouse condition with latency and also with jitter are correct and directly comparable to the results of Exp. 3-1.

While the mouse motions used in Exp. 3-1 are limited to only 2 degrees-of-freedom, they may still generalize to 3D pointing if one considers pointing at target projections. This is discussed at length in Chapter 5. However, the results may not generalize to 3/6DOF input. In Exp. 3-2, the task was more characteristic of VR system usage. However, the software mapped 2D motions to 3D motions, and hence the results may not be directly applicable to full, unconstrained 3D movements. Consequently, while

these results better explain previous results [85], they do not fully explain the tradeoff between latency and jitter in 3D. This issue was addressed in later work that extended the research presented in this chapter [63].

3.9 Summary

This chapter presented two studies examining the effects of device characteristics on both 2D pointing tasks, and constrained 3D object movement tasks. In particular, the effects of latency and spatial jitter were investigated. The results indicate that for low levels of jitter, latency has a much stronger effect on performance. Exp. 3-2 also indicates that erratic jitter has significant performance cost.

One concern about these studies is that direct comparison between the results of both studies is impossible as they used a different experimental paradigm. Although the design of the 3D study simulated the 2D study to an extent, calculation of throughput was not possible. There are two main reasons for this. First, the task difficulty parameters were not comparable between the two studies. While the first study used consistent combinations of distance and size yielding a specific set of *ID*s, the second study used 3D placement, and hence the distances and target sizes were subject to perspective scaling. This issue is investigated further in Chapter 5. Second, there were inherent differences in the task itself – Experiment 3-1 used an "abstract" pointing task, while Experiment 3-2 used an object positioning task. . Hence it is only possible to consider relative differences between conditions and not absolute differences. As discussed above, the results appear slightly different for similar conditions.

From an experimental design perspective, this is somewhat limiting, as one would prefer to be able to directly compare study results. Moreover, the overall goal of this dissertation is the direct comparison of 2D and 3D input techniques. Hence, it is desirable to have a consistent task to use in both cases. The work presented in the next chapter was designed to address these limitations. In particular, the next chapter documents the development of a fish tank VR system that includes a variety of 2D and 3D pointing conditions, with the overall pointing task based on the ISO 9241-9 standard [38]. It includes three experiments ranging from a complete 2D task, to a planar 3D pointing task, and finally a true 3D pointing task.

Chapter 4
3D Target Pointing

This chapter presents three experiments on 3D pointing in a new fish tank virtual reality system. The system [88] was developed to provide a consistent testbed in which to evaluate both 2D and 3D pointing techniques under a variety of conditions. For example, the system allows comparison of target pointing under different stereoscopic stimuli, head-tracking, and target depth conditions. The system uses a 3D extension of the ISO 9241-9 task [38], described earlier.

4.1 Motivation

ISO 9241-9 is widely used in 2D pointing research as it can allow *direct* comparison *between* studies. There is currently no such standard for 3D interfaces. Using a standard highlights the benefits (and pitfalls) of 3D technology with consistency and may enable direct comparison with 2D devices. Motivated by the issues with Experiment 3-2, the main objective of this research was to determine how well the standard can be adapted to 3D pointing evaluation. To this end, these studies include situations where 2D and 3D pointing tasks are directly comparable, i.e., the task is the same. The first experiment (Experiment 4-1 in Section 4.5) presented in this chapter used planar movements between targets displayed at the screen surface, where one would expect that stereo conflicts would be minimal or nonexistent. The second study (Experiment 4-2, Section 4.6) used the same task, but target circles were stereoscopically displayed at different heights

parallel to the display. This was intended to determine how robust the adapted standard task was with respect to changes in the sensory stimuli used. In other words, does the addition of stereoscopic display impact the consistency of results? The third study (Experiment 4-3, Section 4.7) used movement between targets at different heights, thus investigating the difference between 2D and 3D motions in isolation from other factors.

Naturally, consistent with the overall objective of this dissertation, a secondary goal was to compare exemplary 2D and 3D pointing techniques using a well-understood method. Thus mouse pointing was included both as one such exemplary 2D pointing technique, but more importantly as a known benchmark of (2D) pointing performance. According to a previous survey [75] of numerous ISO 9241-9 studies, a mouse affords pointing performance (throughput) of around 3.5–4.5 bits per second. The results presented in this chapter match this, which helps validate the adapted 3D methodology against other work.

Techniques based on ray casting and the "virtual hand" metaphor (using a tracked stylus) were selected as archetypical 3D pointing techniques. The rationale for selecting each technique is explained in detail in Section 4.2. Although most of the *relative* differences of the examined factors/techniques are established by previous work, this work entails the first standard-based comparison that enables characterization of these differences in a more *absolute* sense, and more importantly, in direct comparison with 2D mouse pointing.

4.2 Research Questions

Aside from establishing the methodology (as outlined in the preceding section) there are two main research questions addressed by this chapter. The first is to establish the difference in pointing performance between the mouse, a virtual hand style technique, and a ray-based technique. All three experiments presented in this chapter include a comparison between several mouse and tracker-based techniques. The second question is the importance of tactile feedback for virtual hand type techniques. This is addressed in Exp. 4-2 and Exp. 4-3. Both experiments include conditions with targets co-located with the display surface (affording tactile feedback), allowing comparison with conditions where the targets are positioned in space.

4.3 Pointing/Selection Techniques

Many 3D pointing/selection techniques are used in VR systems. As discussed earlier, these are usually classified roughly into two categories: virtual hand (depth cursor) and ray-based techniques. The studies presented in this chapter use several such techniques:

Pen Touch (PT)

This technique used a tracked stylus and displayed a 1 mm diameter cursor (the "virtual pen tip") co-located with the stylus tip. The virtual tip was tested for intersections with targets. This technique was intended to be representative of depth cursor techniques, or those that require intersection of the hand (representation) with targets [13, 15], i.e., the so-called virtual hand techniques. The user's actual hand was not used in order to ensure

consistency with other work that used actual input devices, styli in particular. Note that if used on a 2D display surface, this technique also simulates pen-based systems [46] and tablets. Hence results for this technique can also be compared directly to known results for the stylus in 2D [75]. Finally, the technique is also sensitive to the effects of the tactile feedback afforded by the screen, i.e., when performing a pseudo-3D task first *at* and then *above* the display surface.

Pen Ray (PR)

This technique also used the same stylus, but rather than requiring intersection of the pen tip with targets, a ray was cast from the pen tip into the scene. This ray visually appeared to extend from the stylus tip, and a small sphere was displayed where the ray intersected the scene. Overall, the effect is similar to a laser pointer. This technique is representative of ray-casting techniques commonly used in 3D user interfaces [13, 65], and other remote pointing applications. Note that it may require up to 6DOF for control, hence based on previous work [98] it was expected to be slower than lower-DOF techniques. Consequently, it likely would also yield worse pointing throughput; the primary reason for including this technique was to establish a throughput score.

Mouse Cursor (MC)

This technique used a mouse-controlled 3D cursor that moved in the screen plane. A ray from the dominant eye was cast through the cursor position to determine which target is hit. This technique represents the system cursor used in "non-VR" 3D graphics software,

such as games and CAD, and also allows comparison with 2D work [49, 75, 83]. The cursor was displayed as a 1 mm sphere stereoscopically displayed at the screen surface to ensure visual consistency across techniques. Note that this technique only works for targets presented in the plane of the display (i.e., at 0 cm "height"), as higher targets occlude the cursor. The floating cursor and sliding cursor techniques (see below) was designed to address this limitation. For targets *inside* the volume of the display (i.e., stereoscopically presented behind the screen), this technique is also subject to diplopia (double-vision). This effect is explored in detail in the experiments presented in the following chapter.

Floating Cursor (FC)

This technique was intended to address the mouse cursor occlusion issue for targets above/in front of the display. Instead of displaying a cursor in the screen plane, the mouse-controlled FC "floats" in a plane parallel to the display slightly above the "highest" targets. In the experiments presented here, this was 8.5 cm above the screen. To address the issue of diplopia, the floating cursor was rendered only to the dominant eye, hence stereo depth discrimination is impossible. Previous work [98] found that a similar 2DOF "one-eyed" cursor outperformed 3D cursors (similar to PT) in a Fitts' study that did not use the ISO standard. This technique was included to compare both against the mouse cursor (MC) and against other 3D pointing techniques.

Sliding Cursor (SC)

The SC technique represents the "depth cursors" sometimes used in games and mouse-controlled 3D graphics systems [10]. SC uses the position of both the system cursor and the eye to compute the position of a 3D cursor in the scene. A ray is cast from the eye through the screen-plane position of the system cursor. The system cursor is not displayed – instead, the 3D cursor is displayed where the ray intersects the scene. Thus, this cursor slides along the visible scene geometry, and enables 3DOF cursor control with only 2DOF input. This technique was intended as a compromise between 2D and 3D pointing techniques and has not been evaluated previously in a classic pointing experiment.

With the exception of the one-eyed floating cursor, all technique visualizations (both rays and cursors) were rendered in stereo. The MC condition was included as a benchmark and for external validation. Based on previous research [75] it was expected to perform in the range of 3.5 to 4.5 bps This also enables ranking the techniques in a 2D task (similar to Experiment 3-1), a constrained 3D task, and finally in a full 3D task. All pointing techniques used in Experiments 4-1, 4-2, and 4-3 are depicted in Figure 4-1.

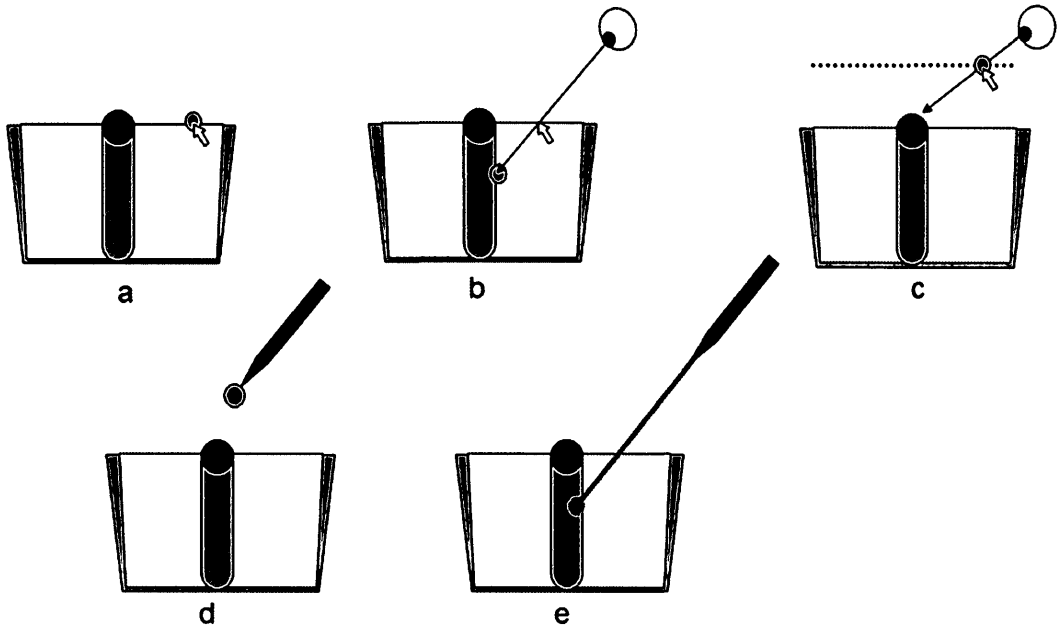

Figure 4-1. Pointing techniques used in Chapter 4 studies. a) Mouse Cursor; b) Sliding Cursor; c) Floating Cursor; d) Pen Touch; e) Pen Ray

The FC and MC techniques require only 2DOF control, while the SC and pen techniques require between 2 and 6DOF, due to the 3D tracker. The PT technique requires the user to control 3DOF of translation simultaneously, whereas all other techniques require, at minimum, accuracy in 2 dimensions. The PR condition is somewhat unusual here, in the sense that while it technically only requires 2DOF of rotation to control the position of the cursor, translation of the pen also affects the cursor position, hence its use ranges between 2DOF and 6DOF control. For the mouse and ray-based techniques, there is no way to specify target depth to select occluded targets; both types of techniques always selected the closest target to the viewpoint. While some researchers have proposed techniques that allow selection of occluded targets (e.g., [30])

no such technique was included here, as head-tracking makes it easy to see an occluded target by moving one's head. Moreover, due to the target layout used, targets rarely, if ever, occluded one another in these experiments.

4.4 System Design Issues

Several additional factors were also considered in proposing a (pseudo-) 3D extension of the ISO 9241-9 task. First, round targets were used to ensure there was only a single *w* (size) parameter. However, both disks and spheres are reasonable choices for round 3D targets. A small pilot study revealed no significant difference for different target types. Ultimately, spherical targets were used for two reasons. First, they are the more natural 3D extension of 2D circles. Second, spheres are also less likely to cause distortion of the target size parameter if viewed off-axis (e.g., when using head-tracking). A disk, on the other hand, viewed off-axis can become visually much smaller, which may affect pointing task difficulty (e.g., if trying to line a cursor up with the target projection).

A second related consideration was the placement of targets. The pilot study mentioned above included two different target placement conditions. The first used targets floating in space and the second used targets anchored onto the tops of cylinders. In the first case, only stereo depth cues were available to help participants detect the actual depth of the targets. In the second case, the cylinders were anchored in the scene, and provided additional feedback. Most pilot participants found it extremely difficult to determine the actual target depth using the pen touch technique to select floating targets. Essentially, the behaviour observed during the pilot amounted to trying to line up the

stylus tip with the target, and gradually moving the stylus into the scene until the target highlighted. This unnatural behaviour was undesired, and would bias the experiment against the pen-touch condition. The other techniques were unaffected, as ultimately they only involved selecting the target via its projection. Consequently, to afford a more fair comparison, target spheres were centered at the midpoint of the top of cylinders, instead of floating in space, as.

A third factor that was considered was how to indicate target selection. A button was mounted on the side of the stylus to indicate selection with the pen-based techniques. Without this capability, trials could only end upon successful intersection with the target, i.e., it would be impossible to miss targets. This would drastically influence the selection time distribution [82] and produce implausible throughput scores. Using a button was the least problematic option, as other alternatives, such as using the non-dominant hand, would complicate the task or introduce issues with bimanual task division. However, this button introduces the potential for "wiggle" in the pen tip upon the button press, the so-called "Heisenberg effect" which commonly affects 3D trackers [16]. However, this did not seem to affect the techniques presented here, likely because the button was positioned near the tip of the stylus where the thumb and forefinger would naturally grip it.

A final consideration relates to the choice of cursor "position" for the purpose of distance calculations (e.g., for effective width) when using ray techniques. When using a 2D mouse-based technique, the choice is obvious: the position of the cursor in the screen plane is used to calculate the distance to the target. However, ray techniques permit

several reasonable choices of the "cursor" position for distance calculations. These include the point where the ray intersects the scene itself, the intersection point with the plane in which the targets reside, and shortest distance between the ray and the target. Note that in screen-projected space, all of these points would correspond to the same 2D point on the screen, hence using screen-project measures may be the ideal solution (this is explored further in the next chapter). For the purpose of these experiments, all ray techniques used the shortest distance between the ray and the target for computing effective width. Short of using screen-projected measures, the other options would either artificially inflate the cursor-target distance, or would be impractical to use with non-planar targets. The cursor-based techniques (i.e., MC and PT), the actual cursor position was used instead. Note that for the purpose of the w_e computation, these distances are always projected onto the task axis.

4.5 Methodology

Since all three experiments were very similar, they all used the same general procedure and identical apparatus. Consequently, the general procedure, apparatus, and participant information are first explained to avoid repeating these sections later for each experiment. Specific details and differences for each experiment are outlined in later sections.

4.5.1 Participants

There were 12 different paid participants in each experiment. For Exp. 4-1, eight were male (aged 22 to 28, mean 24 years), for Exp. 4-2, seven were male (aged 20 to 29, mean

24 years), and for Exp. 4-3, seven were male (aged 19 to 31, mean 24 years). All were students, and were recruited via posters or a web page. Most were non-technical students with little gaming experience, see Appendix for details. This population was targeted as past experience [84, 85] suggests that a technical background or gaming experience can result in unusually high performance levels in 3D selection/manipulation tasks. All had normal or corrected vision, and could see stereo imagery. This was screened by asking participants to measure the height of a stereo target displayed target with a ruler. If participants accurately (within 2 cm) measured the height of the target, their stereo vision was deemed acceptable to participate in the study. All were right-handed.

4.5.2 Apparatus

All studies were conducted on a 3 GHz PC with an NVidia QuadroFX 3400 graphics card. A 22" CRT monitor was used, at 800 x 600 resolution and a 120 Hz refresh rate. The stereo display used Stereographics *CrystalEyes* shutter glasses and emitter. A NaturalPoint *OptiTrack* system with five cameras was used for tracking both the head and a 12 cm long stylus. The stylus had a single button, connected to the computer via a re-engineered USB mouse. The display was positioned on its back, rather than upright, as this was a more natural configuration for the pen-touch condition. The whole setup can be seen in Figure 4-1.

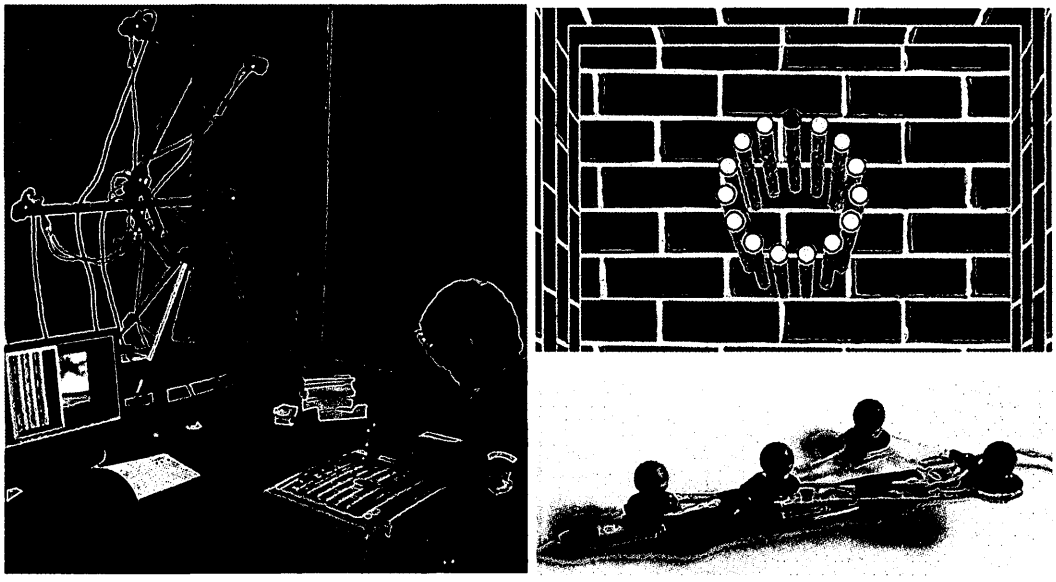

Figure 4-2 (Left). Experimental setup. (Bottom Right) The tracked stylus. (Top Right) Stereoscopically displayed scene with the target cylinder that extend to or above the screen surface.

The equipment was carefully calibrated to (approximately) 1 mm accuracy. Tracker noise was around 1 mm RMS. The system update rate was verified to be 120 Hz and measured the mean end-to-end tracker latency at 63 ms. Mouse latency was around 35 ms. The apparatus was calibrated such that displayed objects could be accurately measured with a physical object, e.g., a ruler. Thus it was possible to line up physical features (e.g., the real pen tip) with features on the image.

Software

The same software system was used in all three experiments described in this chapter. There were minor differences in the presentation of stimuli between experiments. The software presented the inside of a 10 cm deep box matching the display size. Target

cylinders were arranged along the circumference of a circle at the bottom of the box (Figure 4-1, top right). The cylinders and box were textured to enhance depth perception. Target spheres were displayed centered at the midpoints of the cylinder tops and appeared at specified heights at or above the screen surface. Note that although technically possible, targets below (behind) the screen surface were not used in these experiments as they would be impossible to hit with the PT technique. Targets highlighted red when intersected by the cursor to improve feedback. In Exp. 4-1, the mono display mode rendered the same image for both eyes; all other experiments used stereo display exclusively. Quad-buffering and off-axis frustum rendering were used for the stereoscopic head-coupled display. A 1 mm diameter sphere depicted the 3D cursor. This was co-located with the tip of the physical pen in the PT technique. The cursor was rendered semi-transparent to provide clear feedback when it intersected a target [103].

Target diameters and distances were always identical within a single circle of targets. However, cylinder diameters and distance varied *between* target circles. For Exp. 4-1 and 4-2, all cylinders in a given trial round were of equal height. and cylinder heights varied between target circles in Exp. 4-2. In Exp. 4-3, heights varied between individual targets within a target circle. Target height was measured from the surface of the screen. The reason for each of these design decisions can be found in the motivation section above.

The software logged movement times and if targets were hit or missed. Cursor and head movement trails were also logged.

4.5.3 Procedure

Participants were seated at the display and given 24 to 36 practice trials with each technique to familiarize them with the system, the task, and the techniques. Participants were instructed to select each blue highlighted target as quickly and accurately as possible – a standard instruction for Fitts' law experiments, designed to lower result variability. With the PT technique, this meant intersecting the (virtual) pen tip with the target. With the PR technique, they had to intersect the ray with the target. The mouse-based techniques required moving the corresponding cursor to/over the target. Pressing the device button indicated selection and ended the trial. The next target would then highlight regardless if the previous one was hit or missed. The target sequence is shown in Figure 2-13. Timing started after the first click in each target circle, allowing participants to take breaks as necessary. Participants wore shutter glasses in all conditions, and these were always enabled to ensure lighting was consistent.

4.6 Experiment 4-1

This experiment was intended to establish baseline 2D throughput scores for both the pen- and mouse-based techniques. Targets were only displayed at the screen surface, i.e., at a height of 0 cm. This experiment included stereo mode as a factor to rule out differences due to stereo display in the subsequent experiments, which included targets at, and above, the display surface. The expectation was that the mouse cursor in mono display mode should perform similarly to how it would in a 2D ISO 9241-9 study, validating the design of the experiment against the 2D pointing literature.

4.6.1 Design

The experiment had four independent variables with the following levels:

Target Distance:	5, 9, or 18 cm
Target Size:	0.65, 0.85, or 1.05 cm
Stereo Mode:	mono or stereo
Technique:	PT, PR, MC or FC

The experiment used a 3×3×2×4 within-subjects design. Target distance and size were chosen randomly (without replacement) for each target circle. Technique was counterbalanced according to a Latin square. To counterbalance stereo mode, half of the participants completed all stereo trials first, followed by all mono trials. The remaining participants completed these in the opposite order.

Each target circle contained 13 targets, yielding 12 recorded target clicks per circle. Thus, over 12 participants, a total of 10368 trials were logged. A total of 95 trials (~1%) were dropped as outliers (scores more than three standard deviations from the mean), leaving 10273 recorded trials. The nine combinations of size and distance resulted in nine unique IDs ranging from 2.5 to 4.8 bits. The dependent variables were movement time (ms), error rate (missed target percentage), and throughput (bits/second), calculated with Equation (2).

4.6.2 Results

Movement Time

Movement time is the average time (in milliseconds) required to hit targets in a given condition. There was a significant effect for technique on movement time ($F_{3,11} = 60.7$,

$p < .0001$). A post-hoc Tukey-Kramer test revealed that PR was significantly slower than the other techniques, ($p < 0.05$), see Figure 4-2.

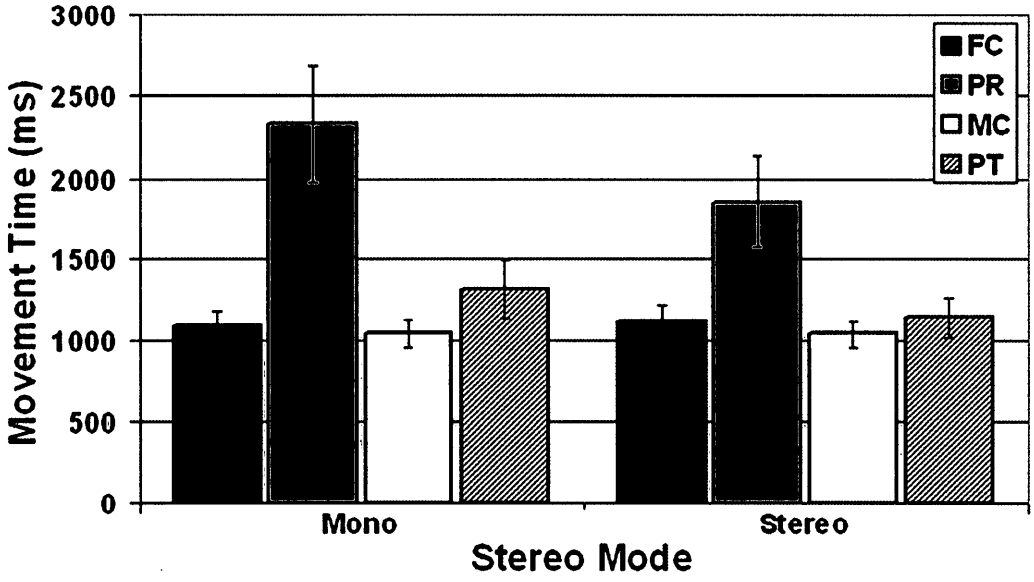

Figure 4-3. Exp. 4-1. movement times by technique and stereo mode. Error bars indicate ±1standard error.

Stereo also had a significant effect ($F_{1,11} = 10.3$, $p < .01$) on movement time, and slightly decreased movement time on average. There was also a significant interaction effect between technique and stereo ($F_{3,33} = 8.4$, $p < .001$). The pen techniques benefitted more from the stereo display mode than the mouse. In particular, PR was about 25% faster with stereo. Figure 4-2 illustrates these results.

Figure 4-3 shows the regression of movement time on *ID*. Most techniques (especially the mouse techniques, as expected) show fairly high positive correlations between movement time and *ID*.

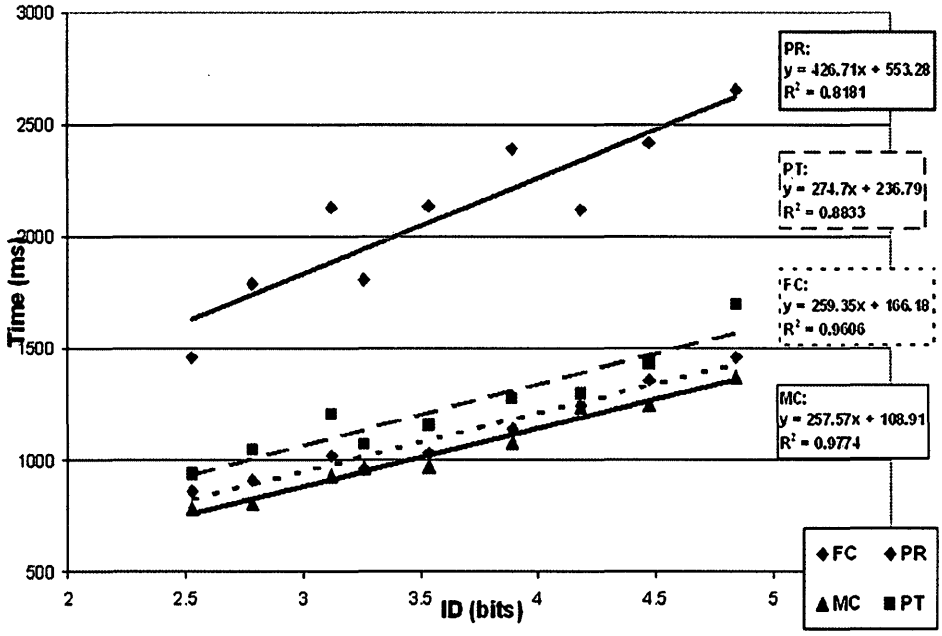

Figure 4-4. Regression of movement time onto ID for Exp. 4-1. Note that each regression line includes both stereo and mono display mode data for each technique.

Error Rate

Technique had a significant effect on error rate ($F_{3,11} = 11.9, p < .0001$). Posthoc analysis indicated that PR had significantly more errors than any of the other three techniques. The error rates were 2.8% for MC, 4.3% for FC, 8.1% for PT, and 13.6% for PR.

Throughput

There was a significant main effect for technique on throughput ($F_{3,11} = 65.4$, $p < .0001$). Consistent with previous work in 2D pointing [75], throughput for the mouse cursor technique was 3.81 bps (*SD* 0.76). Throughput scores are shown in Figure 4-4. Stereo did not have a significant effect on throughput ($F_{1,11} = 3.66$, $p > .05$), nor was there a

significant interaction effect between technique and stereo ($F_{3,33} = 0.77$, ns). The absence of these effects on throughput, despite finding such effects in movement time, may indicate that longer movement times were made up for in accuracy. Even if the raw error rate was higher, low magnitude misses have little impact on w_e, and hence throughput.

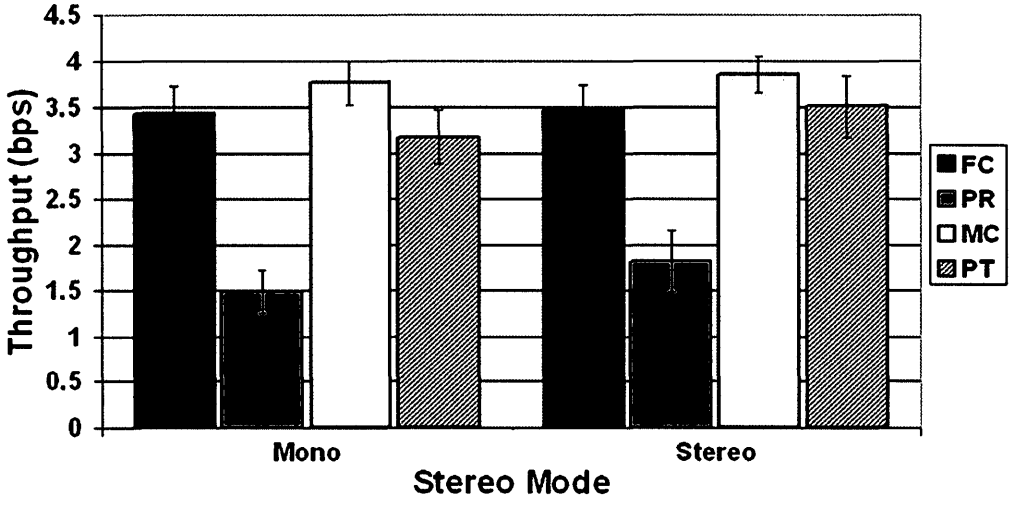

Figure 4-5. Exp. 4-1 throughput results by technique and stereo mode. Error bars indicate ±1standard error.

4.7 Experiment 4-2

This experiment used targets that were stereoscopically presented at or above the display surface. Unlike the previous experiment, stereo display was always used, since selecting targets displayed above the screen without stereo was nearly impossible. Target heights were fixed *within* a target circle, but varied *between* circles. In other words, selections were always made at the same depth, but multiple different depth conditions were included in the experiment. A 0 cm condition was included to enable comparison with the results of Experiment 4-1, as this directly corresponds to the task used in that experiment.

Initially, a 2 cm target height condition was also considered. This condition was ultimately rejected after pilot study participants were observed resting their hands on the screen surface with PT for such targets. This distorted the data as only the 0 cm height condition was intended to afford tactile feedback. This "cheating" was impossible at 4 cm and higher.

The main objective of this study was to investigate pointing in the absence of tactile feedback. A secondary objective was to investigate vergence/accommodation conflicts (discussed in Sections 2.2.1 and 2.2.2). This cue mismatch has negative effects in depth perception experiments [36] and thus may also affect pointing tasks.

4.7.1 Design

The experiment used the following independent variables:

Target Distance:	5, 9, or 18 cm
Target Size:	0.65, 0.85, or 1.05 cm
Target Height:	0, 4, 6 or 8 cm above the display
Technique:	PT, PR, FC or SC

The experiment used a 3×3×4×4 within-subjects design. Target distance, diameter and height were chosen randomly (without replacement) for each target circle. Technique was counterbalanced according to a Latin square. Each target circle contained 13 targets, yielding 12 recorded target clicks per circle. Over all 12 participants, 20736 trials were logged. A total of 284 trials (≈1.2%) were dropped as outliers, leaving 20452 recorded trials. The same set of *ID*s as in Exp. 1 was used. The dependent variables were movement time (ms), error rate (missed target percentage), and throughput (bits per second).

4.7.2 Results

Movement Time

There were significant main effects for technique ($F_{3,11} = 23.5$, $p < .0001$) and target height ($F_{3,11} = 21.5$, $p < .0001$) on movement time. Movement time generally increased with height. This is especially evident in the significant interaction between technique and height ($F_{9,33} = 7.78$, $p < .0001$), see Figure 4-5.

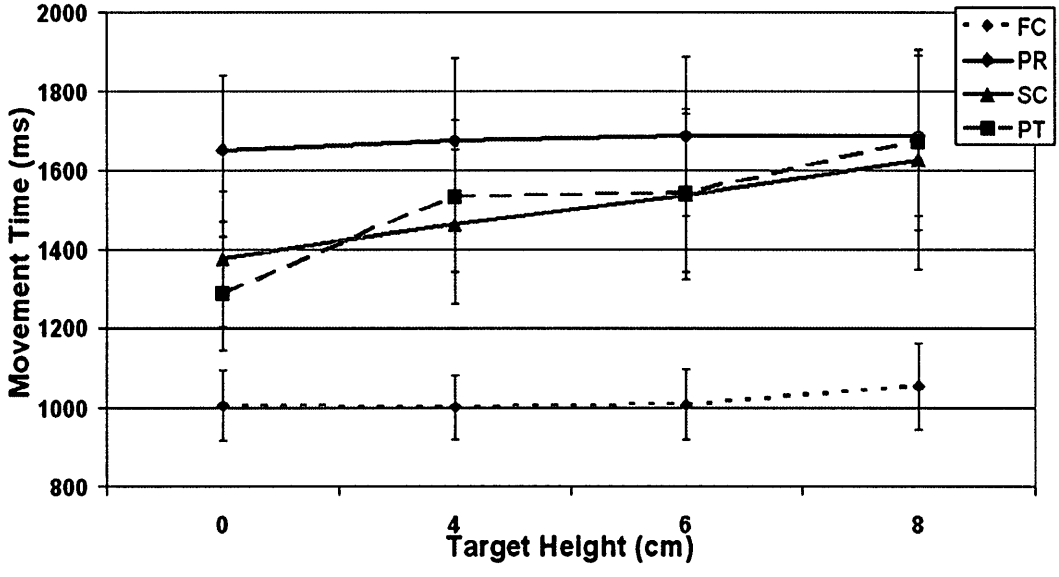

Figure 4-6. Exp. 4-2 movement times by target height and technique. Error bars indicate ±1standard error.

The regression of *MT* on *ID* is shown in Figure 4-6 and indicates strong correlations, especially for the PR and FC techniques.

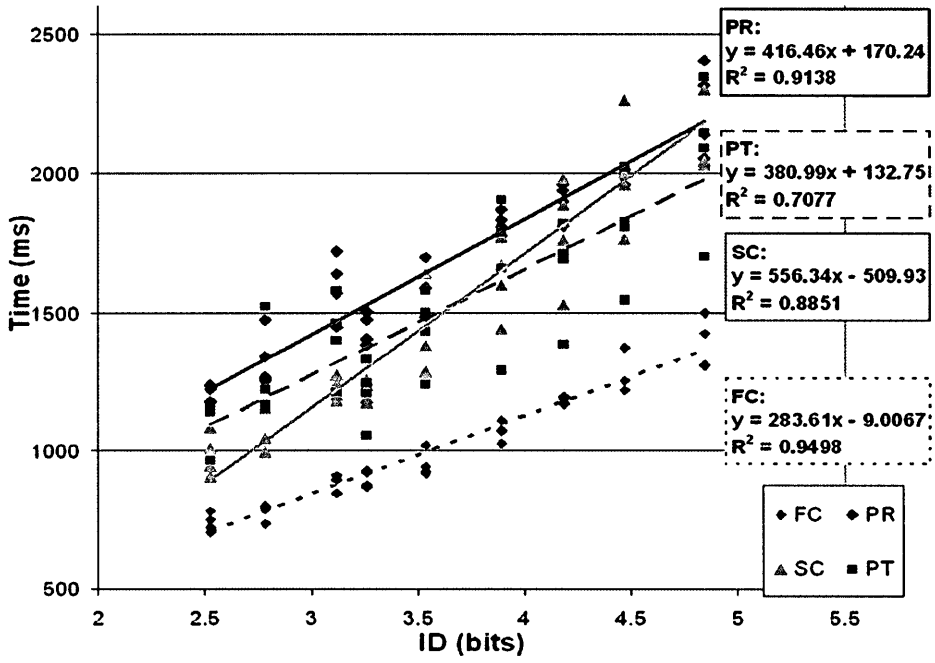

Figure 4-7. Regression of movement time onto ID for Exp. 4-2. Each ID includes one data point for each of the four height conditions.

Error Rate

Technique had a significant effect on error rate ($F_{3,11} = 18.4$, $p < .0001$), while height alone did not ($F_{3,11} = 0.38$, ns). The mean error rate for each condition was 5.1% for FC, 8.5% for SC, 13.5 for PT, and 16.8% for PR. There was a significant interaction between height and technique ($F_{3,33} = 9.73$, $p < .0001$). PT had far fewer errors when the targets were at the screen surface, likely due to the absence of stereo conflicts, or the presence of tactile feedback from the screen surface.

Throughput

There was a significant main effect for technique on throughput ($F_{1,11} = 51.0$, $p < .0001$). A Tukey-Kramer test ($p < .05$) revealed that throughput for the floating cursor was significantly higher than all others, followed by the sliding cursor, and then both pen-based modes, see Figure 4-7. The FC throughput was 3.65 bps, consistent with 2D mouse pointing throughput scores. There was a significant interaction between technique and target height ($F_{9,33} = 5.56$, $p < .0001$). Throughput for PT was significantly higher for targets *at* the screen surface (3.3 bps, in line with Exp. 1). Again, this is likely due to tactile feedback or the absence of stereo cue conflicts. Figure 4-7 shows throughput for MC, PT and FC from Exp. 4-1, as single data points for the 0 cm target height. Although MC was excluded from Exp. 4-2, these are provided for reference. PT and FC performed nearly identically in both studies at 0 cm height. Performance of FC was comparable to MC (within 1 *SE*) in Exp. 4-1, regardless of target height.

4.8 Experiment 4-3

Experiment 4-3 also presented targets above the display. Unlike Exp. 4-2, target heights varied *within* target circles, necessitating true 3D motions. However, targets were still arranged in a circle, to keep the task and target placement relatively simple. The alternative of a spherical arrangement [68] would involve larger numbers of targets and increase the likelihood of confounding target occlusion issues. Consequently, to improve cross-experiment comparisons, the same planar task was used, but the plane was angled

so that it was no longer parallel to the display. This target arrangement then requires true depth motions.

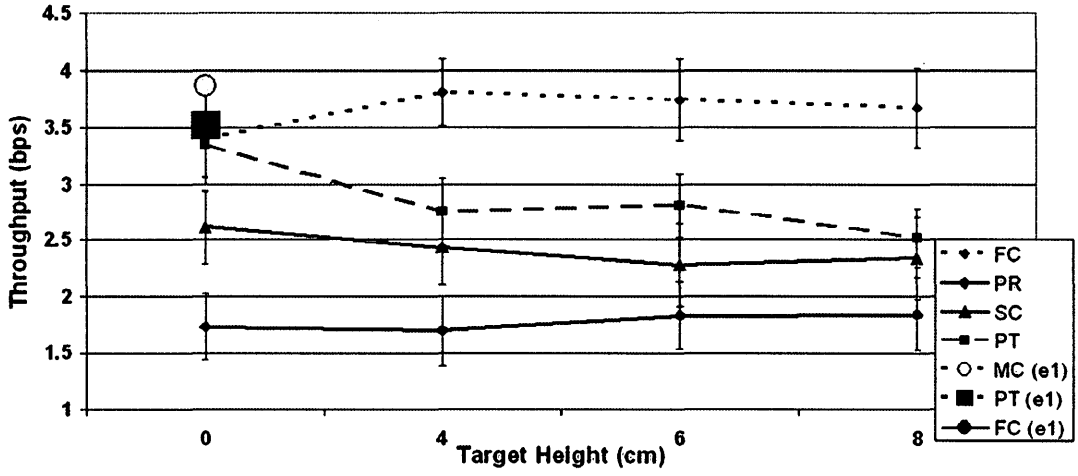

Figure 4-8. Exp. 4-2 throughput scores. Error bars indicate ±1standard error. Exp. 4-1 MC, PT and FC scores are included for reference.

4.8.1 Apparatus

Target heights were determined as follows. First, one target in the circle was randomly set to 8 cm high. The next target following this one in the sequence was was set to 0 cm. The heights of all other target were determined according to a sinusoid between these extremes, effectively arranging targets in a randomly oriented "diagonal" plane.

4.8.2 Design

The experiment used the following independent variables:

Target Distance:	5, 9, or 18 cm between cylinder centres
Target Size:	0.65, 0.85, or 1.05 cm
Technique:	PT, PR, FC or SC

Overall, the experiment used a 3×3×4×4 within-subjects design. Target distance and diameter were chosen randomly (without replacement) for each target circle. Technique

was counterbalanced according to a Latin square. Each target circle contained 13 targets, yielding 12 recorded target clicks per circle. For all 12 participants, a total of 20736 trials were logged. A total of 279 trials (~1.3%) were dropped as outliers, leaving 20457 trials for the analysis.

Target distance indicates the distance between the cylinders in the circle. Here, the Euclidean 3D distance between consecutive target spheres was used in the computation of *ID*. This was intended to better reflect the increased task difficulty due to mixed height targets for the PT technique. Hence, the range of *ID*s is slightly larger than in Exp. 4-1 and 4-2, and there are also a larger number of intermediate *ID* values. However, this is likely not a good representative of task difficulty with the essentially 2D mouse pointing techniques. A better alternative is to use the screen space target size and distance parameters, and is presented in Chapter 5 of this dissertation.

The dependent variables were movement time (ms), error rate (missed target percentage), and throughput (bits per second).

4.8.3 Results

Results were analyzed by height difference between consecutive targets. These were binned into 9 groups: ±8, ±6, ±4, ±2 and 0 cm height difference. A negative height difference indicates that the second target was lower than the first, i.e., required movement *into* the scene. The 0 cm bin contains movements (approximately) parallel to the display surface. This is comparable to the 4 cm conditions in Exp. 2.

Movement Time

Both technique ($F_{3,11} = 25.5$, $p < .0001$) and height difference ($F_{3,11} = 8.1$, $p < .0001$) had significant main effects on movement time, see Figure 4-8. A Tukey-Kramer test revealed that FC was significantly faster than all other techniques. Also, pointing tasks with small depth components (i.e., 0 to 4 cm difference) were significantly faster than those with larger depth components. The regression of movement time onto *ID* is presented in Figure 4-9. FC still shows a strong correlation between movement time and *ID*. The other techniques (especially PT) show lower correlations than in the first two experiments.

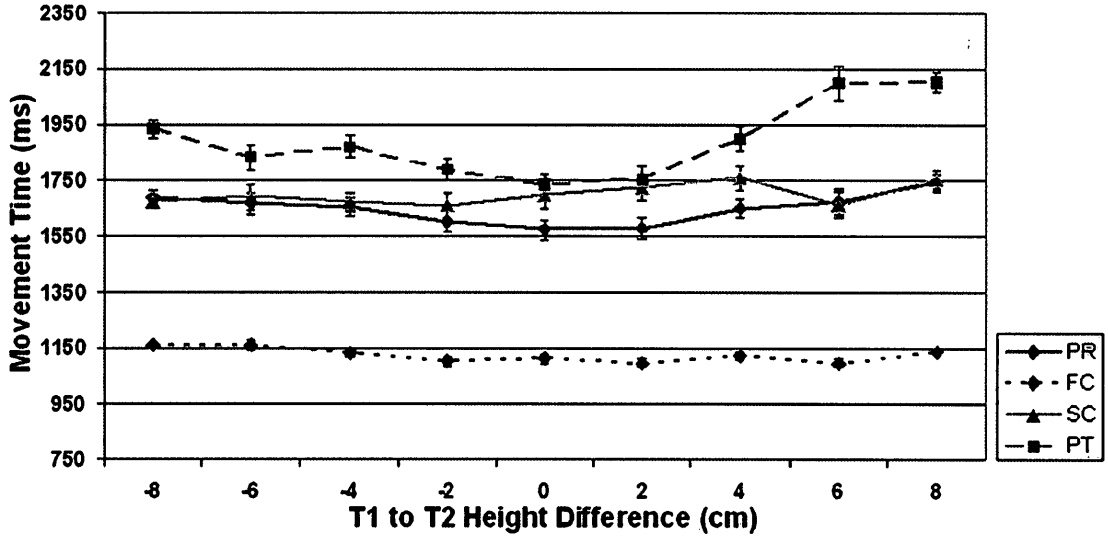

Figure 4-9. Movement time by technique and target height difference for Exp. 4-3. Error bars indicate ±1 standard error.

Error Rate

The error rates were 7.1% for FC, 12.7% for SC, 19.9% for PT and 20.6% for PR. There was a significant main effect for technique on error rate ($F_{3,11} = 11.9$, $p < .0001$). The

higher error rates likely reflect increased task difficulty due to varying target heights. There was a significant interaction between technique and height difference ($F_{3,11} = 4.31$, $p < .0001$), with the PT technique becoming significantly worse for larger height differences, while the rest remained fairly consistent.

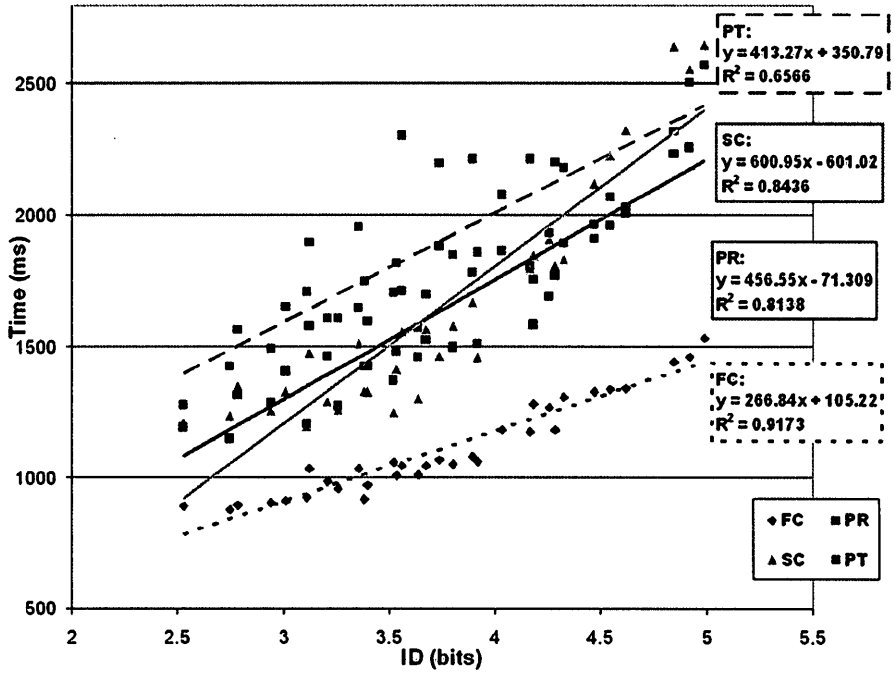

Figure 4-10. Regression of MT *onto* ID *for Exp. 4-3. Note that there are additional* ID *datapoints as the target distance varied more.*

Throughput

There was a significant main effect for technique on throughput ($F_{3,11} = 20.6$, $p < .0001$). The throughput scores are summarized for each technique in Figure 4-10. Average throughput scores for 4 cm height for each technique from Exp. 4-2 are included here for comparison. These scores are quite close to those found in Exp. 4-3.

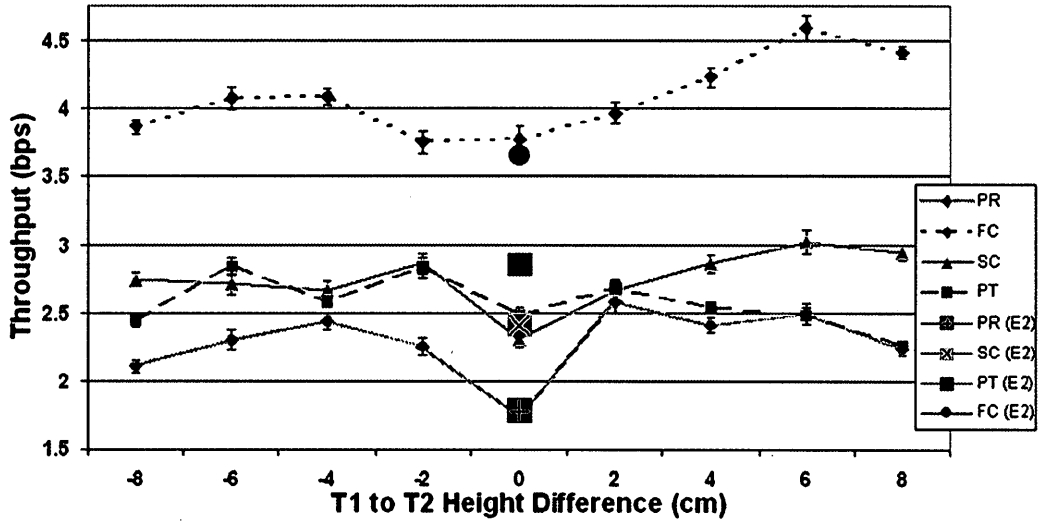

Figure 4-11. Exp. 4-3 throughput by technique and height difference. Error bars show ±1SE. Marks for throughput scores from Exp. 2 are included for reference (for the 4 cm height difference only).

4.9 Discussion

This section discusses the results outlined from the three experiments above. Subjective questionnaire results for the three experiments can be found in the Appendix. These subjective results are not statistically analyzed.

Throughput and Technique Performance

The consistency of the throughput scores reported in these experiments illustrates what is arguably the greatest strength of this measure. Since each experiment included at least one identical condition to the previous one, direct comparison between studies is possible, despite varying error rates. Throughput for the mouse techniques was consistent across all three studies. Moreover, the throughput for the mouse cursor in Exp. 4-1 is consistent with other reports [75]. Similar conditions between experiments showed highly

similar throughput scores. Overall, this suggests both internal and external validation of the experimental methodology.

The natural progression of experiments demonstrated several important and novel results. First, mouse-based pointing throughput is generally unaffected by target depth, regardless of stereo cue conflicts, *if* all targets are at the same depth. If target depths vary there may be an effect due to perspective, which alters the perceived target size. Higher targets are closer to the viewer, and thus are slightly larger (~10%) on the screen. This may make them easier to hit. Conversely, downward motions fare worse, as lower targets are smaller. Since FC is a 2DOF technique, the "correct" target size would be its 2D projection as seen from the current eye position. However, Experiment 4-3 did not exhibit sufficiently strong trends in the data to verify this. This may be due to the limited depth range (maximum 8 cm) used in the experiment. Note that this only affects mouse-based techniques, as hit testing for the PT and PR techniques is performed in 3D motor space. The idea of using screen-space projections of targets for calculating *ID* and throughput is investigated in detail in the following chapter.

The FC technique outperformed both 3D techniques in Exp. 4-2 and 4-3, and had performance comparable to the mouse cursor in Exp. 4-1. A partial explanation is the influence of the number of DOF on pointing task difficulty [98]. It is likely that stereo cue conflicts impacted performance of the 3D-based pointing techniques, as did the absence of tactile feedback in Exp. 4-2 and 4-3. It is unlikely that participant familiarity

with the mouse had a major impact, as the comparatively unfamiliar PT technique performed *as well as* the mouse for targets displayed at the screen surface.

The sliding cursor (SC) performance was on par with the pen techniques. One issue was the added effect of tracker noise and latency, since the cursor position was computed via a ray cast from the eye/head position. Jitter is "amplified" along the length of this ray. Since the targets were small (0.65-1.05 cm) even a little tracker jitter may affect performance. This was also observed as an issue in the experiment. Moreover, some participants resorted to sliding the cursor up the fronts of target pillars, even though this was unnecessary. This same behaviour was noted in Experiment 3-2, presented in the previous chapter. As SC used the eye-ray scene intersection the cursor was partially controlled by the head as well. This too may have affected performance.

The PR technique was worst overall. One issue is that PR was susceptible to tracker jitter amplified down the ray (similar to SC). Yet, PR actually performed worse than SC, perhaps due to the potential for extra degrees of freedom to control the pen. Although the pen only required 2DOF rotations to control, positional control was also possible. The presence of a supporting (and thus jitter dampening) surface also helps SC relative to PR. For a discussion of the effects of technology, see below. Although these results only apply *directly* to fish-tank VR systems, the effect of jitter is likely even more pronounced in larger VR systems due to larger pointing distances, assuming comparable target sizes. Of course, larger targets are possible in a larger display, which may also reduce task difficulty.

Another possibility as to why PR performed so poorly relates to how far away the targets were in the experiments. The targets are much closer than one would normally "remotely" point at – in that sense, the direct touch condition (PT) is a far more natural technique. In general, and especially in large-scale VR systems, remote pointing is reserved for targets that are out of reach. Hence the use of a remote pointing technique for such close targets is somewhat unnatural. This is a limitation of the system and methodology used: for consistency with the other techniques, close-ranged targets are used, even though such a condition is unlikely to occur in a "real-world" application where one would use direct touch for close targets, and remote pointing for far targets.

The absence of tactile feedback and presence of stereo conflicts likely both impacted the PT technique. However, these experiments do not provide sufficient data to identify which effect is stronger. This is because the fish tank VR system used cannot produce a condition *with* tactile feedback but *without* stereo conflicts, or vice versa. Consequently, both factors are investigated together and their combined effect is reported.

Finally, results presented in the previous chapter indicated that the mouse-based technique with 35 ms of latency had 15% higher throughput than and a 3D tracker with 75 ms of latency. This difference was directly attributed to latency. Comparing the latency measurements of these experiments (35 ms and 63 ms) with the previous results, the maximum difference due to latency should be less than 15%. If a lower latency tracking system was used for tracking the stylus in Experiment 4-1, PT may perform

more similarly to the mouse. However, even with a low-latency tracker, PR is extremely unlikely to reach the performance level of the other techniques, as the differences are much larger than 15%. Hence, the difference with PR is likely attributable to factors *other* than latency alone. In Exp. 4-2 and 4-3, the differences between FC and the other techniques in the above-screen conditions were also larger than 15%. Consequently, these results are unlikely to change dramatically with a low-latency tracker.

Modeling 3D Pointing with Fitts' Law

Exp. 4-1 demonstrated strong correlations between *MT* and *ID*. For all techniques, the model explains over 80% of the variability. The mouse-based techniques show the highest correlations, at a level consistent with other (2D) Fitts' law studies, despite stereo rendering. Correlations were also consistently high across all *three* studies for the mouse-based techniques. The addition of a target height factor does not seem to weaken the predictive capabilities of Fitts' law for these techniques within the range of motions evaluated.

However, the correlation between *MT* and *ID* for the PT technique was much worse in Exp. 4-2 and 4-3 compared to the first. The 2D Fitts' law formulation did not model these 3D pointing motions well. This is likely due to differences in pointing strategies and either stereo cue conflicts and/or the absence of tactile feedback. Without tactile feedback, participants had to rely on imperfect stereo depth cues to determine the correct target height. This is consistent with observations during Exp. 4-2: participants had a harder time hitting targets displayed above the screen with PT than at the screen,

and resorted to "homing" motions, effectively searching for the right height. While in Exp. 4-2, participants could "discover" the correct height once per target circle, this was impossible in Exp. 4-3, where participants had to constantly adjust to changing heights. This made the task even harder, and participants frequently resorted to "homing motions". Clearly, this homing behaviour is inconsistent with a typical rapid aimed movement as modeled by Fitts' law.

4.10 Summary

This chapter presented a series of three experiments replicating the 2D pointing task prescribed by ISO 9241-9 in a fish-tank VR system using both a 3D tracked stylus and a mouse as input devices. The results of these studies indicate that 3D pointing performance degrades when targets are displayed stereoscopically above the screen for 3D techniques, but *not* for 2D techniques. Pointing motions *at or parallel to* the surface of the screen are well-modeled using the 2D formulation of Fitts' law for most techniques. Simply using the Euclidean 3D distance rather than 2D distance into account seems to predict 3D motions sufficiently well for 3DOF interaction techniques within the investigated range of motions. However, as discussed at length in the next chapter, for screen-plane 2DOF techniques (i.e., mouse-based techniques) a better approach is to use screen-space parameters, by projecting the pointing task into 2D on the screen plane.

More importantly, this chapter presents a more refined methodology for comparing 2D and 3D pointing devices than that presented previously. This was established by first (Exp. 4-1) using a task that was directly comparable to a 2D pointing

condition. Despite rendering targets as 3D geometry, and using stereo display in some conditions, all targets were displayed in the screen plane, and the task was effectively no different than any other 2D ISO 9241-9 task. It is thus perhaps somewhat unsurprising that the results were fairly consistent with 2D pointing studies. Experiment 4-2 extended this methodology by including both a directly comparable condition to Experiment 4-1, but also 3D pointing conditions by moving the targets into the space above the display. Although directly comparable to Experiment 4-1, the results are still somewhat artificial in the sense that planar movements in 3D are unrealistic. Experiment 4-3 addresses this concern by using a truly 3D task, necessitating movement in all three axes.

This methodology and apparatus is employed in the experiments presented in the following chapter. These studies directly extend this work, in particular, by looking at the issue of screen-plane projections of targets. These are expected to provide a more appropriate representation of target size when using 2D (screen-plane) techniques.

Chapter 5
Stereo Cursors & "2.5D" Pointing

This chapter investigates the interplay between pointing device, technique, and stereo cursors when selecting perspective-scaled 3D targets. Since a primary goal of this research is to develop better methods to directly compare 2D and 3D pointing, both mouse and remote pointing are included in these experiments. Both devices are used with both a screen-plane (2D) pointing technique and a depth-cursor (3D) pointing technique. Although the mouse works with many desktop 3D systems [83], it is impractical in immersive VR systems, and 3D trackers are frequently used (see e.g., [30, 45, 89]). This work further refines the methodology of the previous chapter, and "bridges the gap" between these types of systems by comparing both classes of device and technique.

A mouse cursor with ray-casting affords selection of 3D objects via their screen-space projections. However, projections of far objects are smaller due to perspective, and such objects may be harder to hit with the mouse or remote pointing. Therefore, the effect of perspective due to target depth must also be considered. The first experiment of this chapter uses constant depth between targets, much like Experiment 4-2. The second experiment uses varying target depth. The goal of this work is to model the effect of perspective by extending the 2D formulation of Fitts' law [25] and the ISO 9241-9 standard [38], rather than developing a 3D model. This is more appropriate in such

"2.5D" or projected pointing tasks, as the pointing techniques studied here do not actually require precision in depth.

This chapter also extends the work of the previous chapter [87], by more finely splitting apart certain factors. This includes stereo cursor issues, primarily the depth at which the *cursor* should be displayed. Simply displaying a stereo cursor in the screen plane yields vergence-accomodation conflicts and cause diplopia when trying to select objects at different depths. In contrast, a one-eyed (mono) cursor, first suggested by Ware *et al.* [98], eliminates stereo cue conflicts by displaying the cursor only to the dominant eye. It is thus also immune to diplopia.

In summary, the contributions of this chapter are:

- A comparison of one-eyed and stereo cursors, extending the work of Ware [98] with a more robust experimental paradigm. This shows that one-eyed cursors improve performance with screen cursors, but hinder ray-based techniques.
- A novel screen-plane ray technique that outperforms standard ray-casting and may be more adaptable to immersive VR/AR systems than mouse pointing.
- Evidence that 2D projected Fitts' law parameters are more appropriate than 3D extensions when using screen-plane techniques.
- Evidence that consistent target depth does not affect performance with screen-plane cursors

5.1 Motivation

A drawback of ray techniques is the relative difficulty in selecting remote objects [44]. Far objects take up proportionally less screen space due to perspective. However, in a static scene, far targets are also closer together on the screen. Thus, according to Fitts' law [25], pointing at objects at the same visual depth from the viewer projected onto a screen should be unaffected by object depth, since both width and distance parameters scale by the same factor. There is some evidence to support this presented in the previous chapter (see Experiment 2), but the main objective of the research presented in this chapter is to verify this hypothesis.

Formally, the motivating hypothesis of this work is that selecting targets presented in the same depth plane yields constant performance, regardless of the depth of that plane (and assuming the target plane is orthogonal to the view direction). Here *ID*, which depends on the ratio between *D* and W, is unaffected by target depth as both parameters are scaled by the same factor due to perspective. This can be trivially seen by including a "perspective scale" factor, *p*, in the Fitts' law equation:

$$MT = a + b \cdot \left(\frac{D \cdot p}{W \cdot p} + 1 \right) \rightarrow MT = a + b \cdot \left(\frac{D}{W} + 1 \right) \qquad (4)$$

The scale factor, *p*, cancels out, leaving the standard form of Fitts' law. Perspective scaling of target size is depicted in Figure 5-1, but note that the same scaling applies to target distance as well.

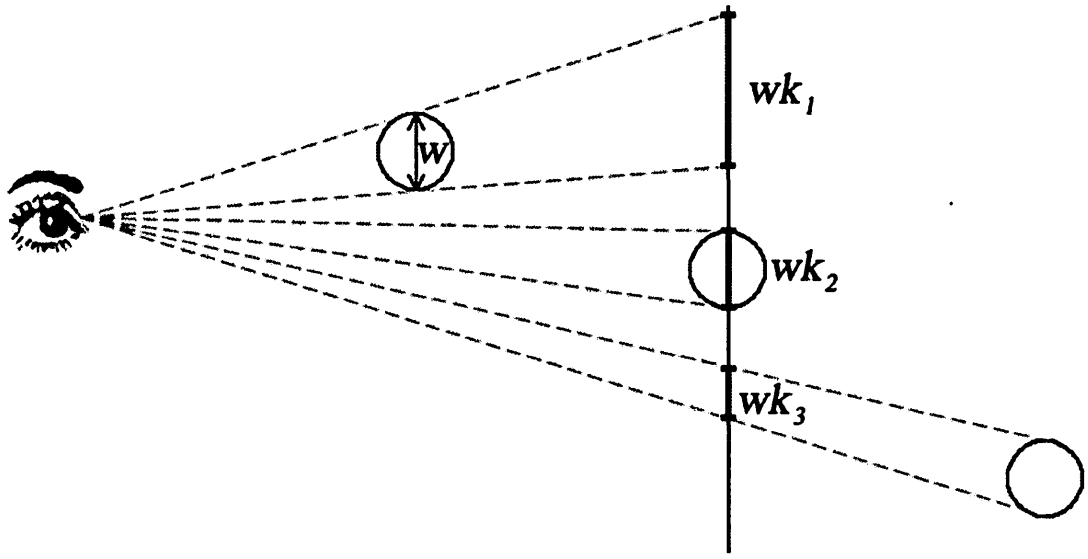

Figure 5-1. Three targets (circles) positioned at three distinct distances from the eye are all of size w. They project to different sizes (wk_1, wk_2, and wk_3) onto the display surface (depicted as the black line), depending on their distance to the display. The same argument applies to target distance.

Consequently, one would expect that screen-plane pointing techniques, such as the mouse, are not affected by target depth, assuming a one-eyed cursor is used to avoid diplopia.

This does not apply to targets presented at different depths nor when head-tracking is used since both affect how targets project to the screen. In a head-tracked system, ID would constantly change for screen-plane conditions as each head motion could affect the target size and distance. In both cases, the *D* and *W* parameters scale by different factors, and *ID* will subsequently change from what was presented. For small head movements or targets that are far away this change may be insignificant, though. Nevertheless, head-tracking is not used in these experiments to avoid this potential confound.

The presence of the aforementioned stereo cue conflicts and diplopia in particular may complicate this issue when pointing at stereo targets, even if using screen-plane techniques. Ware and Lowther [98] report that a "one-eyed" (monoscopic) cursor outperformed a stereo cursor in 3D selection tasks with a 3DOF tracker; a major consideration here is that the one-eyed cursor eliminates diplopia. However, there are large differences between these two techniques and their study did not account for differences in degrees-of-freedom or input device. The stereo cursor required matching the position in all three dimensions. Their one-eyed cursor ignored tracker depth and moved the mono-rendered cursor in the screen plane, effectively pointing at object screen projections. Thus, the current experiments expand on this by comparing cursor rendering style across both 2 and 3DOF techniques.

5.2 Research Questions

There are two primary research questions addressed by the work presented in this chapter. The first question relates to the use of stereo cursors in the screen plane. As discussed above, stereo display yields a double-vision effect (diplopia) when the cursor falls on the projection of a stereo target displayed at a different depth. How strongly this affects performance is unclear. The second issue addressed by this chapter relates to the appropriateness of using screen-space parameters for modeling 2DOF pointing tasks at 3D targets, in the presence of perspective distortion.

5.3 Pointing Techniques

While the current work is largely focused on cursor properties, the effect of input devices is also considered, as the two are not independent. Thus, two different cursor modes are used with each device. The first uses a screen plane cursor and the second a sliding cursor [86]. The first experiment included all four combinations depicted in Figure 5-2.

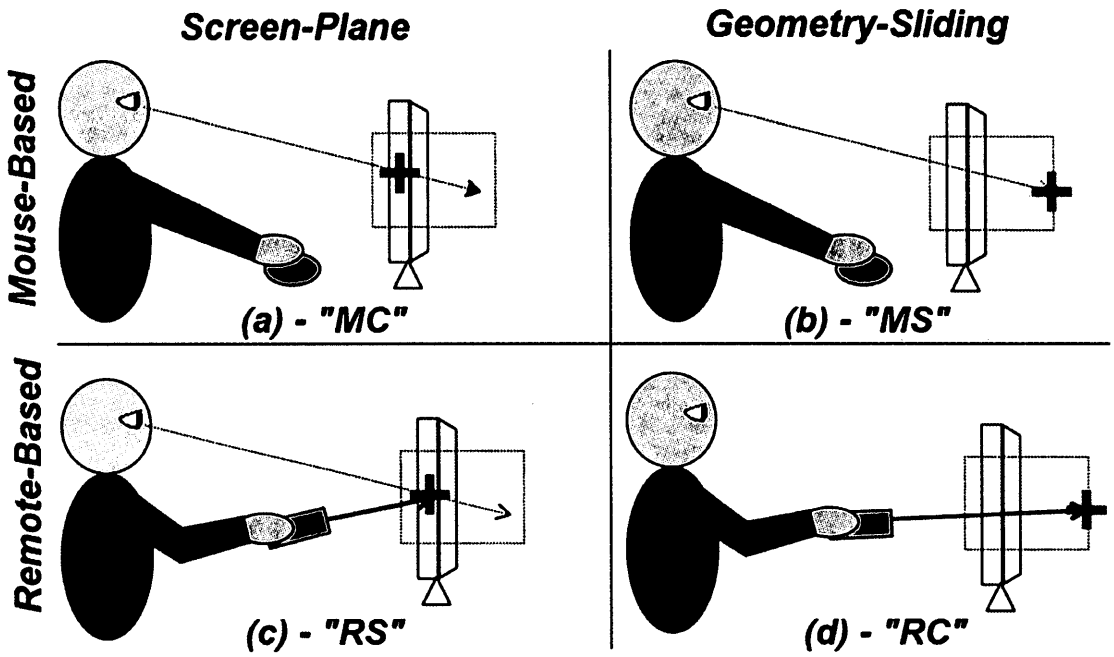

Figure 5-2. The four 3D pointing techniques used in Experiment 5-1. Note that Experiment 5-2 only used the screen plane techniques on the left of the figure: mouse cursor and ray-screen.

The first mouse technique, refered to as MC (mouse cursor), (Figure 5-2a) displays a cursor in the screen plane and uses the eye-cursor ray for selection. This represents typical 3D selection techniques with the mouse. The sliding mouse cursor, or MS (mouse slide), (Figure 5-2b) instead displays the cursor where the (same) selection

ray intersects the *scene*. Thus, the cursor slides across the geometry. The novel "ray-screen" technique, RS, (Figure 5-2c) displays a screen cursor where the device ray intersects the screen, but does not use this ray for selection. Instead, the ray from the eye through this cursor is used for selection. This effectively affords selection of object *projections* via a user-controlled cursor on the screen, similar to mouse pointing. This is different from the technique developed by Argelaguet et al. [3], which used solely input device rotation to control the cursor. While RS is somewhat similar to zoomable interfaces, it also affords off-axis pointing and uses an implicit zoom control (as a function of perspective scaling). The final technique, RC, (Figure 5-2d) is traditional ray casting: a device-centric ray that requires users to point the device directly at the 3D target position, which is a form of sliding cursor.

Both depth cursor techniques (RC and MS) used a cursor displayed in the scene, which was subject to perspective scaling. Note that the actual effect of perspective scaling was small, as the cursor depth varied at most 30 cm (the overall depth of the scene). In practice, the cursor would typically be closer to the viewer, as all targets were closer.

5.4 Methodology

This section describes this chapter's two user studies investigating the effect of cursors, devices, and target depth on performance. The first study, Experiment 5-1, looks only at cursors and devices for motions between targets at equal depths. The second study,

Experiment 5-2, investigates a subset of the conditions on motions between targets at different depths.

5.5 Experiment 5-1

This study establishes a baseline for essentially 2D target selection of 3D target projections. Consequently, targets were presented at a consistent depth, i.e., in each circle of 11 targets, all targets were at the same visual depth from the viewer. All trials in a circle used the same depth. However, depth was varied *between circles* to determine if performance was constant, despite target depth.

5.5.1 Participants

Sixteen paid participants were recruited (mean age 23.1 years, *SD* 5.4). All were undergraduate students, and eight were female. All use the mouse with their right hand and have normal stereo viewing capability. Six participants had previously used 3D input devices in pointing studies.

5.5.2 Apparatus

The apparatus used for these studies was similar to that described in the previous chapter, with some updates to the hardware. This included a 3 GHz PC running Windows XP, an Nvidia *Quadro 4400*, and a 24" 120 Hz stereo LCD. The participant sat approximately 65 cm away from the display on a fixed chair. Although the system supports head-tracking, this was disabled to avoid the potential confounds discussed above. Instead, the

user sat in a fixed chair. The stereo LCD was synchronized via an RF hub with NVidia *3DVision Pro* shutter glasses. Five NaturalPoint *Optitrack S250e* cameras were used for 3D tracking. The tracked remote pointing device was calibrated to 0.7 mm RMS. End-to-end system latency was about 65 ms. No smoothing was used, as noise was already very low and the latency cost of additional filtering may outweigh the benefits [63, 83]. Mouse acceleration was disabled, and gain was set to one level lower than default, for a constant gain of 0.75 [17]. Although low gain levels may increase clutching and impact performance, this was rarely observed in the studies.

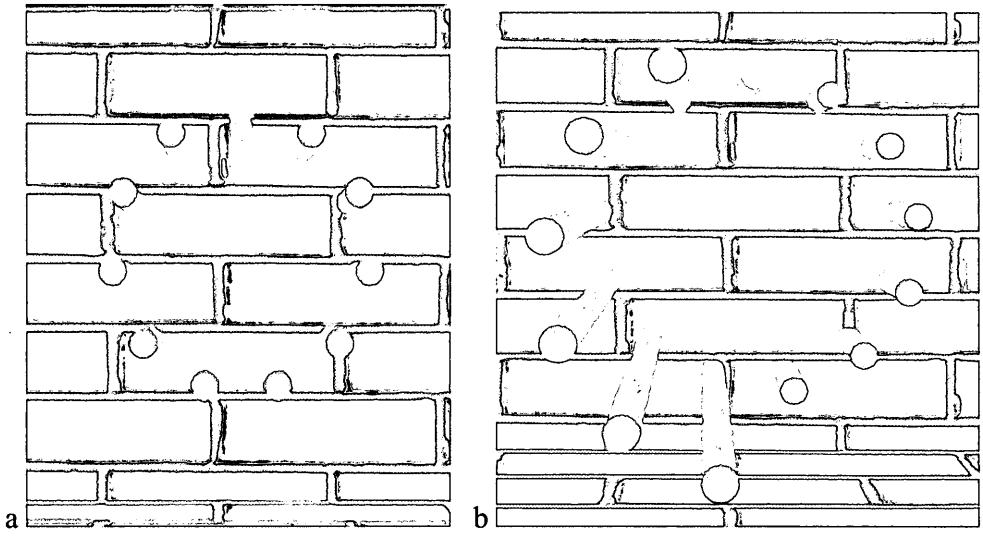

Figure 5-3 (a). Exp. 5-1 setup: the scene depicting a target circle at −20 cm depth. (b) Exp. 5-2 setup: The scene depicting targets at varying depths. Targets on the right side of the circle are at a depth of −20 cm, while targets on the right are presented at a depth of +8 cm relative to the screen.

The same fish-tank VR software that was developed for the Chapter 4 experiments was used here, with some modifications reflecting the updated hardware and

different focus of the studies. The 3D scene was a 30 cm deep box matching the display size, see Figure 5-3a.

Textures and cylinders helped facilitate spatial perception of the 3D scene. Target spheres were placed on top of cylinders arranged in a circle. The active target was highlighted in blue. Targets highlighted red when intersected by the cursor. Selection was indicated by pressing a button on the device. The cursor was always displayed as a small 3D crosshair, either at the screen plane or in the 3D scene, depending on the current condition. The center point of the 3D crosshair had to be inside the target sphere for successful selection; otherwise, the software recorded a miss. In one-eyed mode, the cursor was displayed only to the viewer's dominant eye. Eye dominance was determined by asking participants to visually line up their thumb with a remote feature. In ray mode, the 3D device ray was also displayed to improve feedback. Stereo display was active in all conditions, regardless of cursor style. Target size, distance, and depth were constant within target circles, but varied between circles. Target depth was measured relative to the screen surface; negative depth indicates a target behind the screen.

5.5.3 Procedure

Participants were first instructed on the task. To partially compensate for their lack of familiarity with remote pointing, participants were asked to perform 10–20 practice trials with the ray techniques, until they indicated that they felt comfortable with it. Participants were instructed to select the blue highlighted target as quickly and accurately as possible. The general experimental paradigm followed that of ISO 9241-9 [38]. Target order

started with the "top-most" target (highlighted in Figure 5-3) and always went across the circle. The target order was the same as that depicted in Figure 2-13.

5.5.4 Design

This study used a 2×4×4 within-subjects design. The independent variables were cursor style (one-eyed, stereo), technique (MC, MS, RS, RC), and target depth (+8, 0, −8, −20 cm). The dependent variables were movement time (ms), error rate (percentage of targets missed), and throughput (bits per second). There were 10 trials recorded per target circle. Each target circle represented a different index of difficulty, combinations of 3 distances and 2 sizes. Target distances were 7, 15, and 19 cm, while sizes were 0.9 or 1.5 cm. This yields six distinct *IDs* ranging from 2.5 to 4.5 bits, representing a typical range of pointing task difficulty. *ID* was not used as a factor in the statistical analysis, but was used only to create a range of task difficulties. Each participant completed a total of 1920 trials, for a total of 30720 recorded trials overall.

The dependent variables included movement time (ms), error rate (%), and throughput (bits per second).

5.5.5 Results & Discussion

Data were normally distributed according to a Shapiro-Wilks test at the 5% level. Results were analyzed using repeated-measures ANOVA. Individual conditions were compared with Tukey-Kramer multiple comparisons at the 5% significance level (with Bonferoni

correction). Statistical results for all independent and dependent variables are reported in Table 5-1.

Effect		Movement Time		Error Rate		Throughput	
	d.f.	F	p	F	p	F	p
(T)echnique	3, 15	62.7	***	13.5	***	103.1	***
(C)ursor	1, 15	16.9	**	3.4	.08	0.26	ns
(D)epth	3, 15	7.4	**	6.1	*	18.1	***
T×C	3, 45	46.7	***	8.7	**	52.0	***
T×D	9, 45	11.7	***	2.0	*	7.7	***
C×D	3, 45	13.4	***	2.9	*	5.3	**
T×C×D	9, 135	4.3	***	0.97	ns	4.9	***

*Table 5-1. Experiment 5-1 statistical report. Significant effects are marked * for p <. 05, ** for p < .001 and *** for p < .0001. Actual p values shown for non-significant results. Results with F values lower than 1, which cannot be significant, are indicated with ns.*

Movement time

Average movement times are shown in Figure 5-4. On average, the mouse techniques were faster than the remote pointing ones, and ray-screen was significantly faster than ray-casting. However, these must be considered in light of the interaction effects noted in Figure 5-4.

There was a significant two-way interaction between cursor style and technique. Ray-casting with the one-eyed cursor was significantly worse than all other conditions. The other conditions all *benefitted* from the one-eyed cursor, with one exception: the fastest condition overall was MC with stereo cursor at 0 cm depth. The three-way interaction effect between technique, cursor style, and target depth reveals that the screen-plane conditions (mouse and ray-screen) with the stereo cursor performed significantly worse at the -20 cm depth.

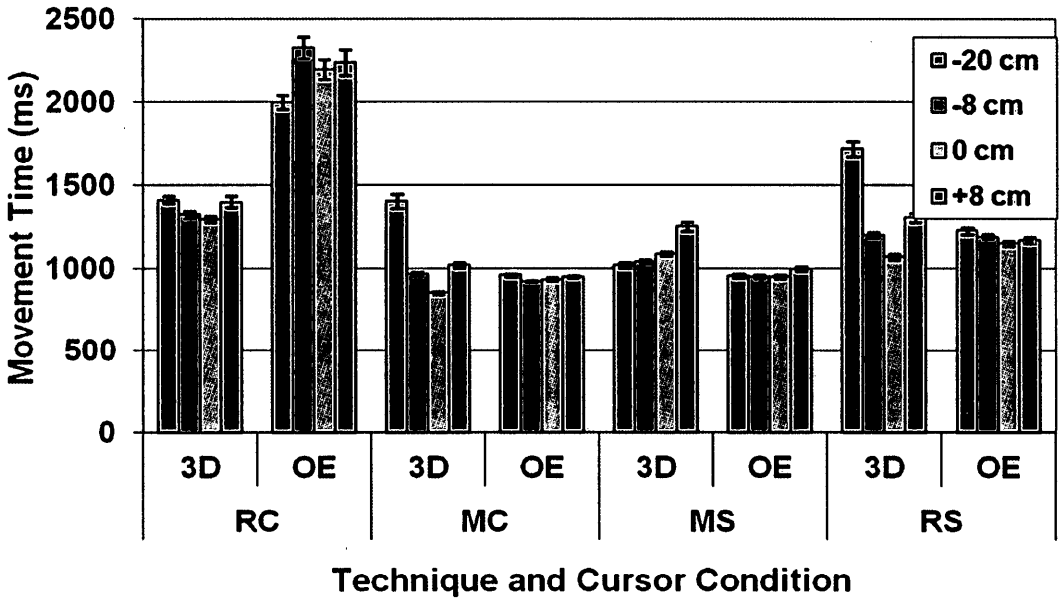

Figure 5-4. Movement time for each condition. One-eyed cursor conditions are represented with "OE", stereo cursor conditions with "3D". Error bars show ±1 standard error.

Error Rate

Error rate is the percentage of trials where the participant missed the target. See Table 5-1 for statistical results and Figure 5-5 for error rates. The mouse techniques had significantly lower error rates than the remote techniques, around 4%, consistent with 2D pointing experiments. A significant interaction between technique and cursor style revealed that the one-eyed cursor significantly increased error rates with the RC technique. This combination of conditions had the highest average error rate (over 20%), suggesting participants had a very hard time accurately selecting targets in this condition. The significant interaction effects between cursor and depth indicate that error rates were significantly higher when using a stereo cursor to select deep targets, especially for the screen-plane techniques.

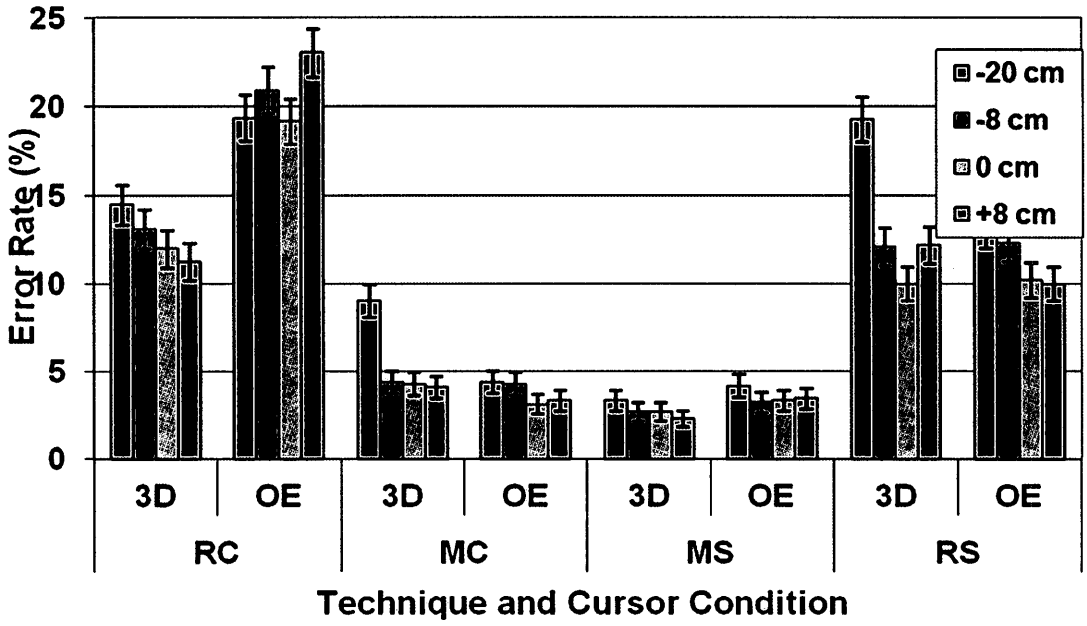

Figure 5-5. Error rates for each condition. Error bars show ±1 standard error

Throughput

Throughput was computed as described earlier, using effective width and distance, see Table 5-1 and Figure 5-6. On average, both mouse conditions were close to 4 bits per second and consistent with 2D pointing, followed by RS at around 3 bps, and finally RC at 2.5 bps. There was a significant interaction effect between technique, cursor style, and target depth. Throughput fell dramatically for targets at -20 cm depth using the stereo cursor for screen-plane pointing techniques (RS and MC). This is clearly the effect of diplopia. The one-eyed cursor hindered the RC technique, which was the worst condition overall, regardless of target depth. The RC technique with the one-eyed cursor again performed worst of all.

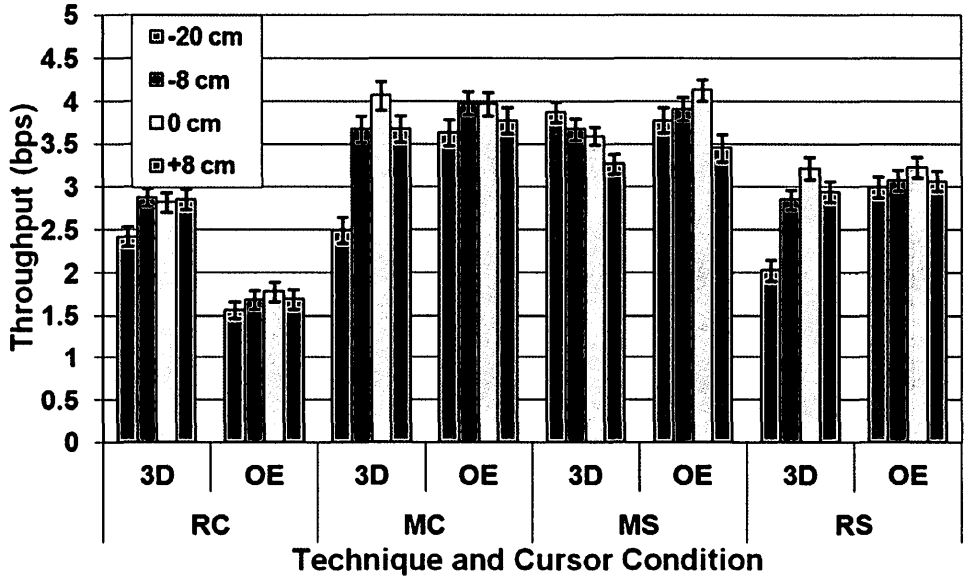

Figure 5-6. Throughput for each condition. Error bars show ±1 standard error. Higher throughput is better.

5.5.6 Modeling

Fitts' law can also be used as a predictive model, by regressing movement time on index of difficulty. This analysis was performed for each technique for both the stereo and one-eyed cursor, and is presented in Figure 5-7 and Figure 5-8. The predictive quality of the model (as expressed by the R^2 values) is very high. However, it is worth noting that the one-eyed cursor consistently improved R^2 values. The one-eyed mouse cursor conditions both show R^2 of around 0.97, indicating almost perfect prediction for both the MC and MS conditions. This makes sense, as these two techniques are arguably 2D pointing techniques, despite the fact that the task is ultimately 3D selection.

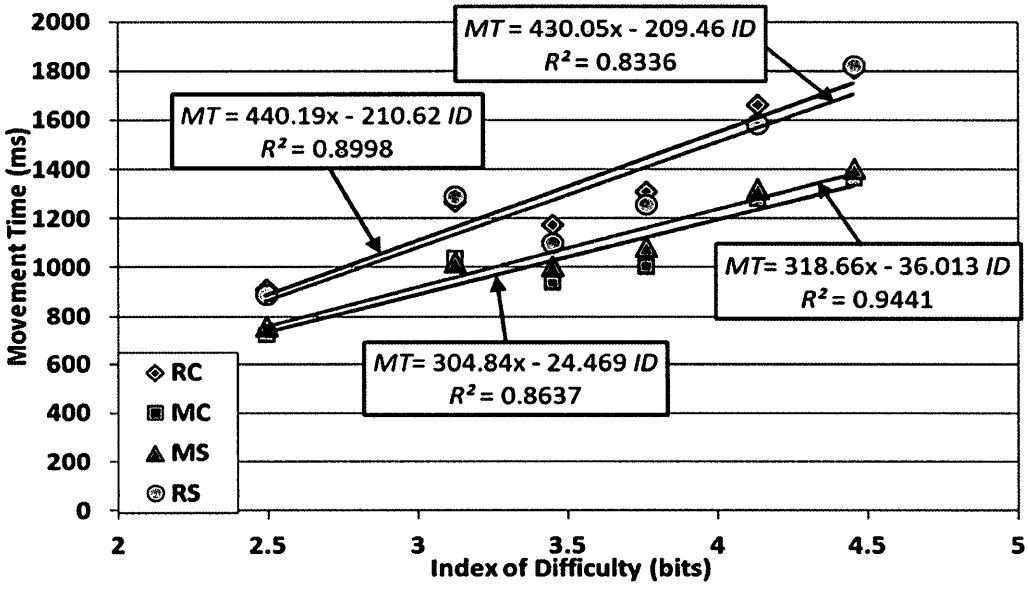

Figure 5-7. Fitts' law models for stereo cursor conditions.

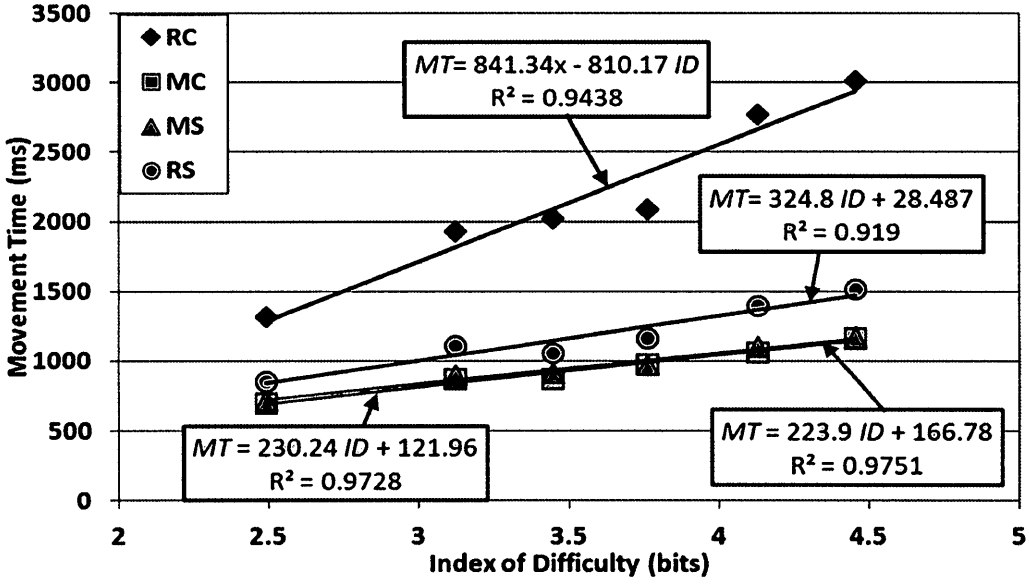

Figure 5-8. Fitts' law models for one-eyed cursor conditions.

The stereo cursor degrades the correlation, especially for the MC and RS techniques. This is likely because the effect of diplopia is strongest on deeper targets. The screen plane cursors (MC and RS) are susceptible to this, especially for the deepest (-20 cm) targets. This is also reflected in the interaction effects noted above, e.g., for throughput, which should performance was significantly worse for these conditions.

The sliding cursor was less affected by diplopia. This is likely because the cursor is in contact with the geometry – consequently, the cursor and target depths are approximately the same most of the time. Overall, this illustrates that the predictive capabilities of Fitts' law are unaffected by target depth for 3D pointing techniques that use 2DOF input and a 2D cursor visualization.

5.5.7 Discussion

Consistent with previous results [98], the one-eyed cursor improved performance, but only for certain pointing techniques. Only the mouse, mouse-slide, and ray-screen conditions benefitted, while ray-casting performed worse with the one-eyed cursor. These results also quantify the benefits of the one-eyed cursor in a more robust experimental paradigm compared to the original experiment [98].

The one-eyed cursor improved performance with mouse-based techniques by reducing the impact of target depth in these conditions. The depth effect is most noticeable in the screen-plane stereo cursor conditions. In particular, throughput peaked at 0 cm depth (i.e., at the screen surface) and fell for targets at different depths. The +8 cm and −8 cm depths exhibit similar throughput, but the −20 cm condition shows a

dramatic degradation of performance. The one-eyed cursor does not suffer from this problem, as it is immune to diplopia. See Figure 5-6.

Movement time for the mouse slide technique using the stereo cursor was significantly faster for deeper targets compared to closer ones. This seems to be related to participants sliding the cursor up the sides of the target cylinder instead of relying on it "popping" to the front. This suboptimal behaviour was also reported in Chapter 4 when using a similar sliding technique [83]. The one-eyed cursor *eliminated* this problem, and participants reported that they could not tell the difference between that condition and the one-eyed mouse (screen) condition, despite perspective scaling of the sliding cursor. The movement times for these conditions are nearly identical independent of depth, and are not significantly different ($F_{1,15} = 0.23$, ns).

The results also reveal the differences between pointing techniques. The mouse techniques performed best, but the new ray-screen technique was competitive and significantly outperformed standard ray-casting. Thus this style of image plane technique may be more appropriate than standard ray-casting for VR systems and games alike. This is similar to Argelaguet's results [3], but contradicts Jota's work [40]. However, Jota used a large non-stereo display system, while the apparatus used for the studies reported here was a smaller display and used stereo. This difference may account for the discrepancy and thus these results may not generalize to large displays (especially without stereo). The multiple interaction effects indicate that most techniques work best with a one-eyed

cursor, while some require a stereo cursor. Similarly, some techniques perform best for deeper targets, while others perform best for close targets.

The ray-casting condition's relatively poor performance may again be related to the somewhat unrealistic use of the technique. As in Chapter 4, the range of "remote" pointing was quite small (less than 100 cm); remotely pointing at targets within arm's reach may not be a very natural task. The fact that the worst condition overall was the RC technique with the one-eyed cursor may be due to the relative difficulty in determining the actual depth of the cursor, and especially how its position was related to the device. This highlights the importance of stereo display for ray-casting, but also highlights that the 2DOF techniques can be effectively used without stereo.

Finally, the one-eyed mouse cursor afforded throughput similar to a standard 2D mouse cursor. This was fairly consistent for both one-eyed mouse conditions. The one-eyed ray-screen condition was also unaffected by target depth. The movement times confirm that performance is unaffected by the perspective scaling of a scene with targets displayed *at the same depth* when using screen-plane techniques. The following study expands this investigation by looking at pointing for targets *at different depths*.

5.6 Experiment 5-2

In this study, target depth varied between subsequent targets. As a result, perspective scaling affected the projection of the targets. The objective of this experiment was to empirically measure and model the effect of perspective scaling. To keep the size of the

experiment manageable, only the best-performing mouse and ray techniques from the Exp. 5-1 were included, i.e., the mouse cursor and ray-screen conditions.

5.6.1 Participants

Twelve participants (mean age 29.4 years, *SD* 5.4) took part in the study. Nine were male, and all were right-handed.

5.6.2 Apparatus

The hardware setup was identical to that used in Study 1. However, the software was modified such that target depth varied from target to target. Each target circle was arranged such that every *other* target was at a different depth. This ensured that every subsequent target selection required moving either from a deep target to a near target, or vice versa. This can be seen in Figure 5-3b. Correspondingly, data were later split into "up" and "down" motions to analyze each separately. This design is one of the few options for accurately analyzing 3D movements with the ISO standard, which requires uninterrupted "circles" of targets. More importantly, it improves further on the methodology used for Experiment 3 of Chapter 4, as it allows analysis of discrete depth differences within a circle of targets. Since the depth differences in Chapter 4's Experiment 3 were continuous, such analysis was impractical.

5.6.3 Procedure

While the apparatus was modified, the procedure was identical to that of the previous study.

5.6.4 Design

The study used a 2×2×3×3 within-subjects design. The first two independent variables were cursor style (one-eyed, stereo) and technique (MC, RS). The remaining independent variables were all nine possible combinations of the three target depths (+8, 0, −20 cm). The dependent variables were movement time (ms), error rate (percentage of targets missed), and throughput (bits per second). There were 12 trials recorded per target circle. Each target circle represented a different index of difficulty, combinations of 3 distances and 2 sizes. Target distances, more precisely the distances between cylinders, were 7, 15, and 19 cm apart, while target sizes were 0.9 or 1.5 cm in diameter. This yielded six distinct *IDs* ranging from 2.5 to 4.5 bits, when computed according to the conventional formulation of Fitts' law (discussed further below). Thus each participant completed a total of 2592 recorded trials, for a total of 31104 trials overall.

5.6.5 Results & Discussion

Approximately 8% of all trials were dropped as outliers, if their movement times were more than three standard deviations from the grand mean time. After outlier removal, the data were normally distributed according to a Shapiro-Wilks test at the 5% level. Results were analyzed using repeated measures ANOVA and Tukey-Kramer multiple

comparisons at the 5% significance level (with Bonferonni correction). Data for each "round" of trials with different target depths were separated into two sets: upwards and downwards movements. These two data sets were treated separately from then on, including the calculation of standard deviations. Statistical reports for each independent and dependent variable combination are shown in Table 5-2.

Effect		Movement Time		Error Rate		Throughput	
	d.f.	F	p	F	p	F	p
(T)ech	1, 11	16.7	*	37.0	***	9.3	*
(C)ursor	1, 11	3.1	.11	8.8	*	1.2	.30
(D)epth	8, 11	16.9	***	17.3	***	9.4	***
T×C	1, 11	1.1	.32	3.1	.11	0.03	ns
T×D	8, 88	3.8	***	6.9	***	1.1	.39
C×D	1, 88	11.7	***	4.2	***	10.3	***
T×C×D	8, 431	1.5	.19	2.5	*	1.8	.08

*Table 5-2. Statistical results for Experiment 5-2. Significant effects are marked * for p <. 05, ** for p < .001 and *** for p < .0001. Depth represents all combinations of target depths. Note that throughput here refers to "screen-projected" throughput. This is explained in detail below. Actual p values shown for non-significant results. Results with F values lower than 1, which cannot be significant, are indicated with ns.*

Movement time

Movement times are shown in Figure 5-9 and statistical values are shown in Table 5-2. Technique had a significant main effect on movement time, while cursor style did not. However, depth combination did have a significant effect, suggesting that it was a greater source of variability. Significant interactions between technique and depth suggest that ray-screen is more strongly affected by increasing movement *into* the scene. This is likely because these targets are perspective-scaled to appear smaller while target distance stays (mostly) constant. The ray technique is also subject to greater input device noise than the mouse. An interaction between depth and cursor style suggests that stereo cursor

performance falls with deep targets, regardless if the target depths are the same (e.g., the −20 to −20 condition) or not. The slowest conditions overall were ray-screen with stereo cursor and motions involving −20 cm deep targets. Movement *out* of the scene or in front of the screen had relatively little impact on performance, regardless of technique or cursor style, see Figure 8. The fastest condition was the ray-screen/stereo cursor condition at the screen surface, i.e., when all targets were at 0 cm.

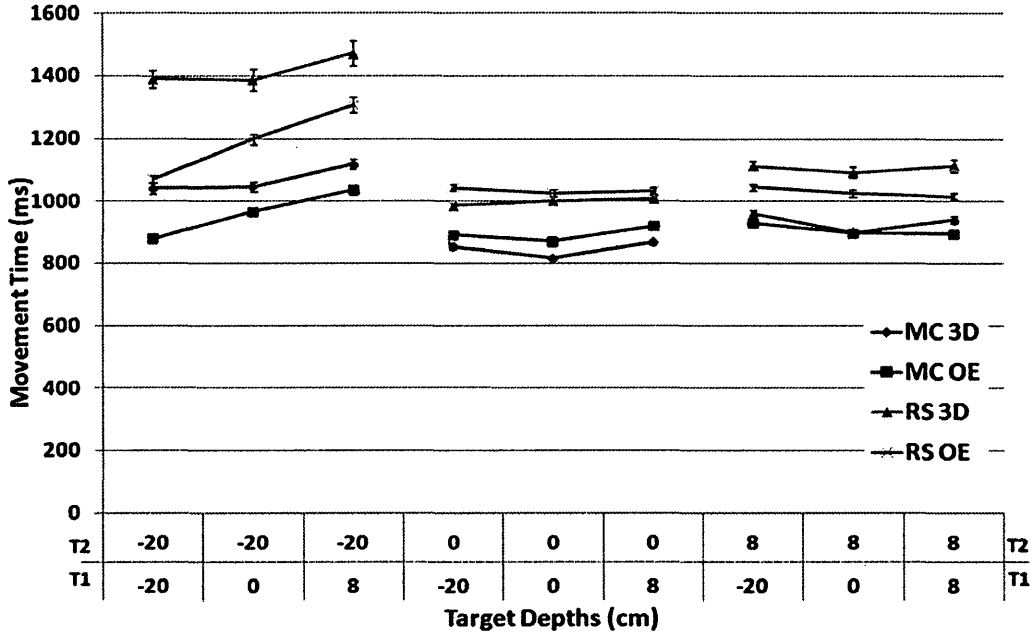

Figure 5-9. Movement time by depth combination, cursor mode, and input technique for Exp. 5-2. Error bars show ±1 standard error. T1 and T2 are the start and end depths of each trial.

Error Rate

Error rate is the average percentage of trials where participants missed the target for a given condition. Statistical results for error results can be found in Table 5-2. Error rates are summarized for each condition in Figure 5-10. Every investigated independent

variable had a significant main effect on error rate. The error rate for the one-eyed mouse cursor is around 5.5%, slightly higher than in the previous study.

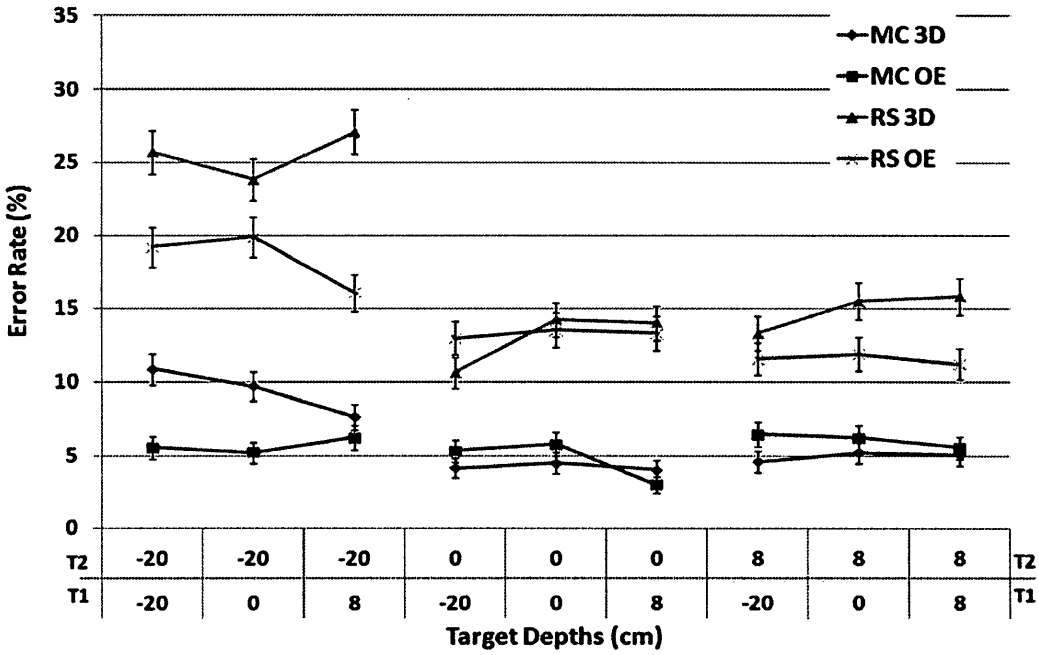

Figure 5-10. Error rates by technique, cursor style, and depth combination. Error bars show ±1 standard error. T1 and T2 are the start and end depths of each trial.

For the RS technique the average error rate is much higher than for the mouse, between 10% and 25%. This is highlighted by the significant interaction effect between the technique, depth, and cursor conditions. While the ray-screen condition is significantly worse than the mouse cursor, it is unsurprisingly far worse with a stereo cursor when pointing at deep targets. This can be seen in Figure 5-10 for any target depth ending at a −20 cm target. On the other hand, the mouse cursor error rate is essentially constant with the one-eyed cursor, regardless of the depth of the start or end target. This is further evidence that this condition is unaffected by target depth.

"Euclidean" Throughput

Initially, throughput was computed as in the previous study, and also in the previous chapter [86], by using the Euclidean 3D distance between the targets. While this works fine for targets at the same depth, it artificially inflates throughput scores for movements involving two different target depths. This inflation is more pronounced for greater depth differences. In Figure 5-11 this manifests as a "dip" in the middle, with inexplicably higher throughput scores for greater depth differences. One can see a similar "dip" in some conditions in Figure 12 in previous work [86], especially for ray-casting.

Screen-Projected Throughput

To avoid this inflation, the concept of "screen-projected" throughput is proposed. The motivation for this modified throughput score is based on the observation that most pointing techniques used in this study effectively require only 2D input. These include the mouse cursor, ray-screen, and arguably ray-casting. For these techniques, performance should be evaluated in the screen plane by first projecting the pointing task (i.e., cursor position, target position, size and distance) onto the screen plane.

To compute screen-projected throughput, the target and cursor positions are projected onto the screen plane. Effective width is then computed using the standard deviation of the *2D* distances from the projected target to the projected cursor instead of 3D distances. For simplicity, the small effect of perspective distortion of the target sphere shapes is ignored. Similarly, effective distance is computed as the 2D distance between the projected cursor position and the previously clicked projected position for each trial.

Throughput is then computed normally using these screen-space effective width and distance values. The statistical results for screen-projected throughput are shown in Table 5-2 and mean screen-projected throughput scores can be found in Figure 5-12.

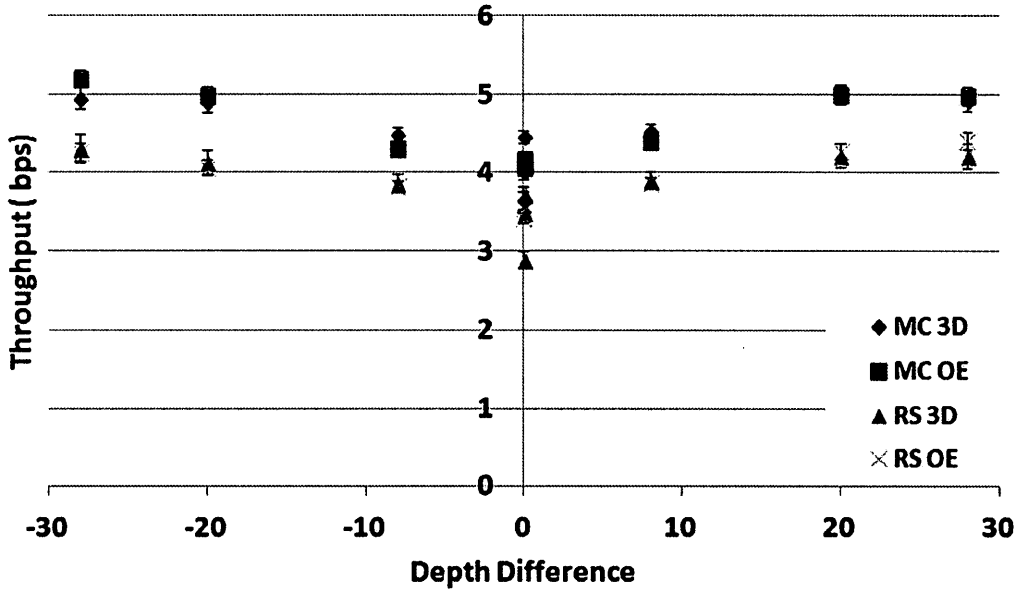

Figure 5-11. "Euclidean" throughput, illustrating the artificial inflation of throughput scores for movements with greater target depth differences. While the planar movements (i.e., depth difference of 0) throughput scores are reasonable, the extreme depth difference conditions do not make sense in this figure.

Technique had a significant main effect on the new screen-projected throughput while cursor style did not. The combination of start and end target depth did have a significant effect as well. Overall, the mouse cursor affords significantly higher throughput than the ray-screen technique. There is a significant interaction effect between cursor style and target depth combination. Pointing at deeper targets is significantly worse with the stereo cursor than with the one-eyed cursor, as expected. The end target depth of the current trial (T2) seems to matter most here. For example, throughput is

fairly consistent for all −20 cm deep targets, irrespective of the depth of the start target depth (T1).

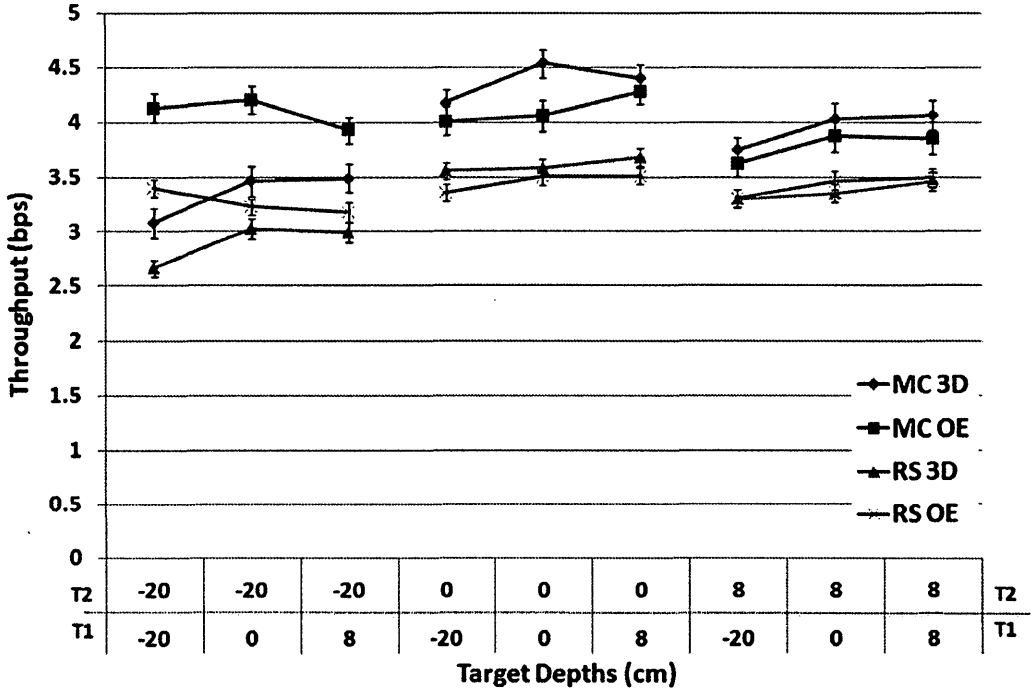

Figure 5-12. Screen-projected throughput by technique, cursor style, and depth combination. Error bars show ±1 standard error. T1 and T2 are the start and end depths of each trial.

5.6.6 Discussion

Screen plane throughput was not affected by depth with either one-eyed cursor technique (MC OE, or RS OE). To reiterate the results of the Tukey-Kramer post-hoc test, target depth does not significantly affect the mouse ($F_{6,101} = 0.96$, ns), nor ray-screen ($F_{6,101} = .85$, ns). In the absence of diplopia this suggests that perspective scaling of targets due to depth does not affect pointing performance. This makes sense and supports the argument that screen-projected throughput is an appropriate measure for such tasks.

One would expect throughput to remain constant regardless of target depth. This is similar to how the measure behaves for changing distances and sizes in 2D.

5.6.7 Perspective Scaling of Same-Depth Targets

As discussed earlier, a primary hypothesis of this work is that selecting targets subject to the same perspective scaling should yield constant performance when using screen-plane techniques. Hence, if all targets in a circle are at the same depth, then throughput should not change regardless of depth. To verify this, the same-depth conditions in User Study 2 were analyzed in greater detail, i.e., the conditions [−20 to −20], [0 to 0], and [+8 to +8 cm]. Figure 5-13 depicts the mean screen-projected throughput for each condition.

Figure 5-13 illustrates that performance for both techniques was mostly constant with the one-eyed cursor. There is at most 5% variation in throughput for the mouse and only 1% for ray-screen. Neither are significant (mouse, $F_{2,33} = 0.3$, ns; ray-screen, $F_{2,33} = 0.16$, ns). While this does not conclusively prove that depth has no effect, it indicates that the null hypothesis – that there is no difference due to depth – cannot be rejected. This is currently the best explanation for this data. Performance was much more variable with the stereo cursor for both pointing techniques. This is not unexpected, due to the stereo cue conflicts present in these cases. In particular, the -20 cm depth condition was most strongly affected by diplopia, as in the first user study of this chapter.

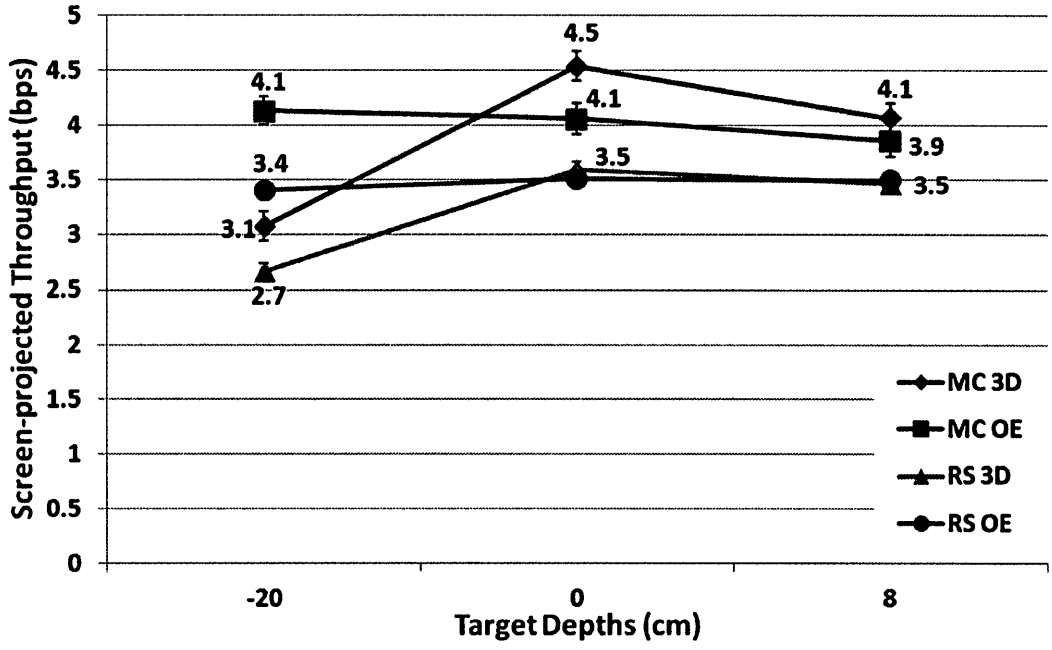

Figure 5-13. Screen-projected throughput by pointing technique and cursor style for each combination of same-depth targets. Error bars show ±1 standard error.

5.6.8 Modeling

Performance models were created for each condition, using screen-projected target size and distances. No additional parameters were incorporated into the Fitts' law models, as screen-projected ID (calculated the same was as screen-projected throughput) should be sufficient to explain the effect of perspective scaling. Figure 5-14 presents the aggregate models for each pointing technique, using both one-eyed (Figure 5-14a) and stereo (Figure 5-14b) cursor styles.

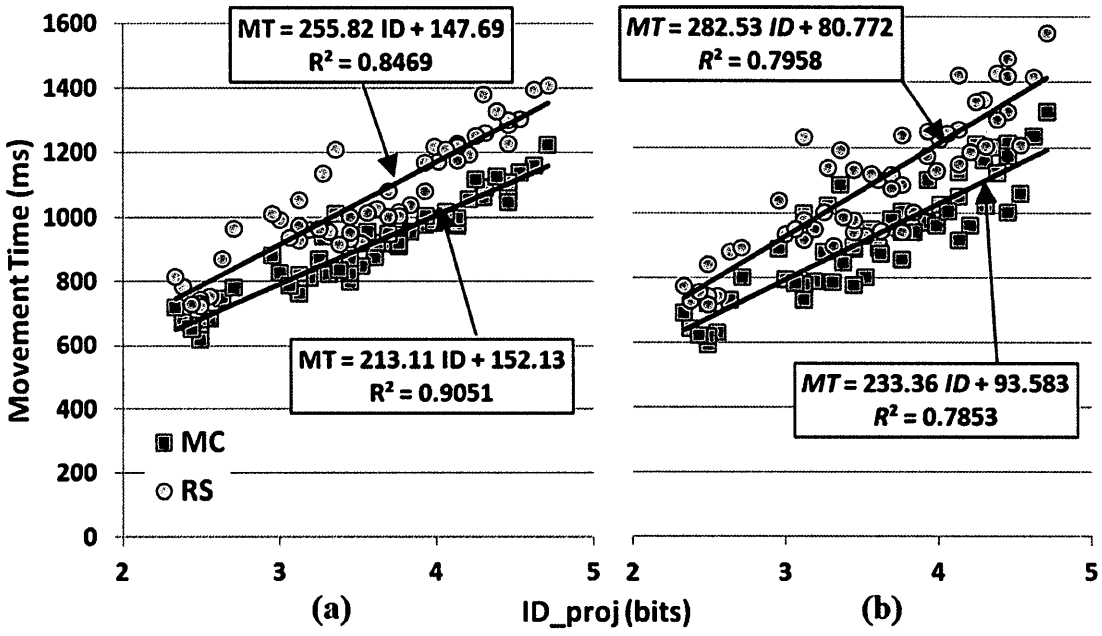

Figure 5-14. Regression models for the one-eyed (a) and stereo cursor conditions (b) using ID_proj, i.e., screen-projected ID. These models include all target depth combinations.

The models fit slightly worse than one would expect of Fitts' law, especially the stereo cursor cases. Given the aforementioned depth cue conflicts with the stereo cursors, this result is not surprising. The worse correlation exhibited by the one-eyed cursor conditions may be due to the time required to re-adjust the eyes to different depths in presence of accommodation-vergence conflicts [36]. Therefore, separate regression analyses were performed for each target depth combination. These models are summarized in Table 5-3. As expected [36], participants required more time to adjust for greater depth differences. This is visible both as higher intercepts and worse predictive qualities, R^2. The models fit very well for near-screen conditions, where depth cue conflicts are weakest.

Note that regression analyses for the stereo cursor conditions are not included, as the effect of diplopia is too strong to produce reasonable models. Moreover, it is difficult to predict the additional time required to acquire a target in the presence of both diplopia and the aforementioned accommodation-vergence conflict. This is a topic for future investigation.

Depth Diff. (cm)	Target Depths (cm)	Mouse, One Eyed Cursor			Ray-Screen, One-Eyed Cursor		
		Intercept (a)	Slope (b)	R^2	Intercept (a)	Slope (b)	R^2
-28	8, -20	254.5	198.3	.8554	435.7	199.9	.8113
-20	0, -20	253.5	189.9	.9066	188.2	257.9	.8059
-8	8, 0	77.2	230.8	.9826	52.6	268.5	.9355
0	-20, -20	166.8	198.8	.9551	154.5	252.8	.8781
0	0, 0	111.4	211.3	.9962	119.7	251.0	.9042
0	8, 8	10.3	246.59	.9826	18.4	279.4	.9601
+8	0, 8	54.7	240.3	.9886	-15.4	296.2	.9309
+20	-20, 0	185.0	204.8	.9685	208.5	242.6	.9236
+28	-20, 8	277.9	191.8	.9585	312.7	214.9	.9074

Table 5-3. Regression models between projected ID and movement time for the one-eyed cursor conditions for each distinct depth difference. Target depths indicate the starting and ending depth of a pointing task.

5.7 Motion Analysis

As discussed by MacKenzie et al. [52], throughput is useful for establishing differences between techniques, but provides little insight into *why* there are performance differences between techniques. Consequently, movements were analyzed to help better explain the observed differences in throughput.

Two types of analyses were considered here. The first considers measures proposed by MacKenzie et al. [52] which look at how frequently movement inefficiencies occur (typically a count normalized per trial). These include:

- task axis crossing (TAC): how frequently the cursor crossed the task axis, i.e., the line between two consecutive targets
- target (re-)entries (TE): how often the cursor enters (or re-enters) the target before selection occurs
- movement direction change (MDC): how often the direction of movement changes in a direction parallel to the task axis
- orthogonal direction change (ODC): how often movement direction changes in a direction orthogonal to the task axis
- movement variability (MV): variability in the movement path relative to a straight line
- movement error (ME): standard deviation of movement from the task axis
- movement offset (MO): average deviation relative to the task axis (i.e., average distance, either positive or negative)

While these measures were developed for 2D pointing analysis, a simple and reasonable extension is used here to adapt these measures for use with *projected* motion trails and targets. Hence, for all measures, the screen-plane cursor movement is used. Similarly, screen space positions/sizes of targets are used where applicable. For example, to consider "projected" task-axis crossings, one must first determine the 2D screen-plane position of two consecutive target centres. This is done using a ray-plane intersection test, using the ray from the eye through the target's 3D position. The 2D line (in the screen plane) between these two targets is then computed, and used as the task axis. Since the

techniques in this study strictly used screen-plane cursor motion, no additional projection is required to create a 2D cursor position trail. Using the 2D projected task axis, and the cursor trail, the number of crossings are then computed. Other measures are computed similarly, by first projecting the targets. Although similar analysis of 3D motions is possible [18], such measures were not created for analyzing these results, as ultimately, the task was considered to be 2D.

Another method of evaluating motion characteristics is by breaking down the movements into different phases. This approach has been used with some success in 3D movement analysis previously [48, 59]. It seems plausible that this type of analysis may explain some of the same movement inefficiency expressed by MacKenzie's measures. So this analysis has also be conducted. The same parsing criteria adopted by Liu et al. [48] and originally proposed by Meyer [53] are used to split apart the movements into the ballistic and correction phases. Specifically, the peak movement speed is first found, then "pauses" in the movement are detected as intervals where the movement speed drops to less than 0.05 times the peak velocity. These pauses delimit sub-movements in the overall motion path. Any sub-movement that contributes more than 25% to the overall path length is considered to be part of the ballistic phase. The remaining movements are considered part of the correction phase of movement. In addition to determining the overall length (in milliseconds) of these two phases, the number of ballistic and corrective sub-movements are also counted. Like the measures discussed above, these are normalized per trial.

The results of these two analyses for Experiment 5-2 are summarized in Table 5-4, which also reports significant differences between the conditions. The four main conditions are included: MC OE, MC 3D, RS OE, and RS 3D. Note that there are significant differences across almost all measures by condition. The only exception is ballistic sub-movements (BSM, row 10), which are not significantly different regardless of condition.

	MC OE		MC 3D		RS OE		RS 3D		
Variable	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**	**F**
1. TE	1.06	0.35	1.06	0.38	1.46	0.88	1.51	1.05	119.7 ***
2. TAC	1.30	1.25	1.29	1.21	1.94	1.86	1.90	1.88	35.8 ***
3. ME	0.66	0.36	0.65	0.33	0.73	0.42	0.71	0.43	3.63*
4. MV	0.50	0.29	0.49	0.27	0.56	0.34	0.55	0.41	5.73***
5. MO	-0.04	0.65	-0.03	0.62	0.02	0.72	0.01	0.71	20.3 ***
6. ODC	0.002	0.07	0.001	0.05	0.02	0.21	0.01	0.20	10.1 ***
7. MDC	0.002	0.06	0.001	0.04	0.01	0.33	0.01	0.21	5.8 ***
8. BPL	506.9	194.0	500.6	182.4	553.5	310.9	534.3	343.9	9.6 ***
9. CPL	407.7	292.0	471.3	387.5	558.9	518.7	732.8	831.2	33.7 ***
10. BSM	1.66	0.75	1.65	0.74	1.65	0.88	1.60	0.89	0.8, ns
11. CSM	1.99	2.85	2.48	4.28	3.56	5.54	4.86	9.09	21.3 ***
* p < .05, ** p < .005, *** p < .001									

Table 5-4. Mean, standard deviation, and statistical result for each accuracy measure. Accuracy measures and their units are: TE – target entries, count; TAC – task axis crossings, count; ME – movement error, cm; MV – movement variability, cm; MO – movement offset, cm; ODC – orthogonal direction changes, count; MDC – movement direction changes, count; BPL – ballistic phase length, ms; CPL – correction phase length, ms; BSM – ballistic sub-movements, count; CSM – correction sub-movements, count. Excluding the time measures, all of the above averages are normalized as ratio measures per trial.

As can be seen in Table 5-4, the ray-screen with both the one-eyed and stereo cursor generally exhibit worse scores for all measures than both mouse conditions. Tukey-Kramer posthoc tests (at the .05 level) indicate that for most measures, there are no significant differences due to the *cursor*, i.e., there is no difference for any measure between MC OE and MC 3D, or RS OE and RS 3D. Interestingly, an exception to this is both the correction phase length and number of corrective sub-movements (CPL and CSM respectively). These two metrics *are* significantly different for the ray-screen conditions due to cursor visualization – RS 3D is significantly worse than RS OE for both of these. While not significant, the CPL and CSM scores for the mouse cursor technique are also quite different. It is possible with additional participants, these differences would become more pronounced and would also be significant.

The difference observed in the correction phases makes sense, and is consistent with Woodworth's original hypothesis [102] and subsequent work investigating this in 2D [53] and 3D [48, 59] pointing. Specifically, during the ballistic phase, sensory information is not processed – the ballistic phase is "pre-programmed" and carried out in the absence of feedback. Once the correction phase of motion begins, feedback is processed in a closed loop as the user gradually advances the cursor toward the target, constantly using (visual) feedback to assess the motion. This process is clearly impacted strongly by the presence of diplopia as indicated by the significantly worse scores with RS with the stereo 3D cursor.

Naturally, device characteristics (e.g., latency and jitter) also impact performance. Overall, these appear to have a much greater impact on performance than cursor visualization. This is evident in the first eight scores reported in Table 5-4, as the mouse cursor tends to outperform the ray-screen condition in all cases (with the aforementioned exception of BPL). It is somewhat surprising that cursor visualization did not have a stronger effect, given that each condition reported represents a range of depth combinations. One might expect that all such scores would be worse for the stereo cursor conditions due to the effect of diplopia and other stereo cue conflicts. However, this only appears to be true in the measures relating to the correction phase of movement. The mouse conditions have shorter correction phases, but the presence of the stereo cursor increases the length of the correction phase for both the mouse and significantly for the ray-screen techniques.

5.8 Overall Discussion

The above motion analysis provides some insights into the reasons for the significant differences in throughput. In these experiments, it appears that the device characteristics tend to have a much stronger impact on performance, as these are significantly worse across device. The state of the cursor has less impact – most of the measures were not significantly between the one-eyed and stereo cursor states for a given input device. The exceptions to this are the correction phase length and corrective sub-movement measures, which suggest that during the sensory feedback loop, the state of the cursor *does* matter. This is likely due to the wide range of target depths used in the study, and the impact

diplopia clearly has when there is a large depth difference between a stereo cursor and target.

The models presented in Sections 5.4.6 and 5.5.8 were not directly compared to others (e.g., [28, 44, 57]) for several reasons. First, the task used in these studies is essentially 2D, as it involves pointing at 2D projections of targets on the screen plane. Murata's model [57] may be applicable, but their addition of a free parameter is not well justified. There several differences between the task used here and those used by Grossman [28] or Kopper [44]. Grossman's work used a position-controlled 3D cursor and trivariate targets. Consequently, Kopper's work is a better comparison point.

Kopper's model [44] is, in some ways, similar to the idea of projecting targets to the screen plane. It is based on the visual angle size and distance of targets, while the model presented here projects targets to their "2D" size. Although conceptually similar, there are some differences in these two approaches. First, Kopper's model relies on a novel formulation of *ID*, and is consequently incompatible with the Shannon formulation used by the ISO standard. It provides no means to compute effective target size and distance, and thus no means to produce the standardized throughput measure. As discussed earlier, a primary objective of the research presented in this dissertation is the direct comparison between 2D and 3D pointing. This is an advantage of using screen-projected size and distance over the visual angle approach.

Second, Kopper's work focused exclusively on distal (remote) pointing. It is thus unlikely that their model would work with the mouse. Hence, a direct comparison of

models between devices is not feasible. Using screen-projected parameters applies to either type of input technique.

Additionally, Kopper used neither stereo display nor varying target depths. Thus their results are not subject to the stereo cue issues or to the perspective scaling of targets observed in the respective studies. Although the correlations of the models presented here are somewhat low, the mouse model matches or exceeds the predictive capabilities of Kopper's model for ray-casting for individual depth conditions. Moreover, it is in line with what one would expect from standard (2D) Fitts' law studies.

For summaries of participant subjective questionnaire responses, please see the Appendix.

5.8.1 Implications for Designers

These results show that 3D user interface designers should be wary of using stereo cursors for selecting targets displayed away from the screen. Interestingly, both studies seem to indicate that stereo cursors offer slightly better performance for targets near the display surface. However, screen-based stereo cursors hurt performance when targets are presented away from the screen. This is likely due to diplopia and/or the accommodation-vergence conflict. Experiment 5-2 suggests that it is the disparity between the target and cursor that matters most, rather than the actual depth difference. This is also reflected in the measurably worse correction phase of movement when using a stereo cursor with mixed target depths.

This also suggests that developers of stereo 3D games should avoid screen-plane stereo cursors. Unfortunately, they are currently common practice in games. Overall, both studies indicate that the advantages of stereo cursors are minimal. But, in general, their usage can significantly *hinder* user performance in 3D pointing. Thus, system developers are encouraged to consider including a one-eyed cursor option. This leaves the decision of whether to use a stereo cursor to the user, and permits them to avoid performance degradation in stereo display systems.

The results of the motion analysis also confirm that there are still a number of characteristics of remote pointing devices that are undesirable. In practice, this is typically not a concern, as most commercially viable 3D software use a mouse as an input device. However, it does suggest that if pointing performance is a concern, designers of VR systems might consider the use of a mouse, or even a touchscreen. Although touchscreens were not studied explicitly, one might speculate that the presence of their support surface would yield similar benefits to a mouse.

Finally, there is now interest in the development of stereo touchscreen interfaces [89]. Such interfaces suffer the same problems when interacting with stereo targets far from the screen. Much like a stereo mouse cursor, a finger on a stereo touchscreen is also subject to diplopia! This work indicates how much of an impact this effect may have.

5.9 Summary

The two studies of this chapter investigate stereo cursor properties and the effect of perspective on target selection. Their results quantify the benefits of the one-eyed cursor

in a more well-refined experimental paradigm compared to previous work [98] and suggest that the one-eyed cursor is not universally beneficial. A major contribution of this chapter is that it provides evidence that consistent target depth does not affect pointing performance. While this might be expected, it appears to be a completely novel result. Exp. 5-2 identified that varying target depth affects performance, but this can be (at least partly) accounted for by using screen-plane projections of targets. Overall, mouse-based techniques tended to perform best. But the new "ray-screen" selection technique also outperforms traditional ray-casting. Consequently, VR system designers are encouraged to consider adaptation of this new technique for immersive 3D systems that use remote pointing devices.

Chapter 6
Overall Discussion & Conclusions

This dissertation investigated several issues relating to previously measured [84, 85] performance differences between 2D and 3D input devices. This previous work initially began as an investigation of why 3D manipulation tasks see relatively little use in practice. Ultimately, that work established that there are clear and significant performance differences between 2D and 3D input devices when performing conceptually equivalent selection and manipulation tasks. Those studies used a somewhat coarse methodology, however, and consequently it is difficult to target specific aspects of 3D manipulation tasks for improvement.

The approach used in this dissertation dramatically improves on the previous work in this area. This involved the systematic comparison of simple 2D and 3D pointing motions that make up more complex selection and manipulation tasks. The main advantage of this approach is that such simple motions have little to no element of user strategy, unlike more complex manipulation tasks. This is a desirable characteristic of such evaluations, as it elicits only differences due to the investigated technical (e.g., latency) or perceptual (e.g., stereo cue conflicts) limitations of a given condition. Previous work [85] yielded fewer meaningful results, despite large differences in the explored conditions.

Since a primary goal of this work was to directly compare 2D and 3D pointing motions, a side-effect of this investigation was to explore improved experimental methods for comparing the two. This appears to be a novel contribution in and of itself, as there appears to be very little work done on the direct comparison of 2D and 3D pointing motions. For reasons outlined earlier, this involved the use and modification of the ISO 9241-9 standard [38] and Fitts' law [25]. In particular, situations where 2D and 3D pointing motions were *directly* comparable were investigated, such as when pointing at targets displayed in the screen plane. The experiments of Chapter 4 used this situation to establish a baseline of pointing performance, then investigating true 3D pointing motions. The experiments of Chapter 5 built on this by matching the cursor degrees-of-freedom between 2D and 3D pointing devices, and investigating screen-space pointing.

The subsequent sub-sections of this chapter summarize the main findings and contributions of each of the preceding chapters.

6.1 Three-Dimensional and Screen-Space Pointing

This dissertation makes several important contributions to the area of 3D pointing device evaluation – the methodologies proposed in each chapter are, in themselves, contributions. As discussed earlier, a primary motivation for this work is the *fair* and *direct* comparison of 3D and 2D pointing techniques/devices. Previous work [82, 84, 85] arguably does not accomplish this. For example, comparing 2D pointing using a front-face "depth sliding" algorithm [84] reduces the dimensionality of a 3D manipulation task to 2DOF; comparing this to a 3DOF manipulation condition is not really fair. Similarly,

the confounding effect of user strategy found in previous studies [85] limits the directness of any comparison between conditions.

Each chapter in this dissertation proposes a (slightly) different approach for the direct and fair comparison of 2D and 3D pointing. In all cases, these approaches employ modified versions of the ISO 9241-9 standard method [38] for pointing device evaluation. Motivated by Fitts' law, this investigates simple pointing motions rather than complex manipulation tasks.

Experiment 3-1 proposed a useful methodology for comparing 2D and 3D pointing. This is based on the observation that in order to directly compare the two, one must perform either a 3D or 2D pointing task with both techniques. Since one cannot use the mouse for a direct 3DOF pointing task, one logical alternative used by the experiments of Chapter 3 is to constrain the tracker to 2D operation by mounting it on the mouse. The advantage of this approach is that one rules out differences due to hand tremor and the absence of a supporting surface required by the mouse. In effect, only differences in the *sensor*, such as latency and jitter, are measured.

The other possibility is to perform an actual 3D pointing task with both devices. This is the approach used by Chapters 4 and 5. This approach is somewhat less direct than mounting a tracker on a mouse. Hence a wider range of pointing techniques is used with each device in these chapters, to provide a gradient of "directness" between full 3D and 2D pointing. As discussed earlier, this necessitates techniques like ray-casting to map 2D mouse input to 3D pointing operations. Chapter 4 investigates several possibilities for

mouse cursor placement when using mouse ray casting (e.g., screen plane, floating, or sliding along the end of the selection ray). Arguably, the sliding cursor proposed here is most similar to remote pointing with a tracker-based ray technique (e.g., the pen-ray technique evaluated in this chapter), while the mouse-cursor technique is most similar to standard 2D pointing. Using a gradient of techniques like this allows varying degrees of direct comparison between 2D and 3D pointing.

Chapter 5 further examines this issue by investigating only screen-space techniques – effectively, inverting the ideas of Chapter 4, which attempts to make mouse-based pointing techniques more like 3D techniques. In this case, operation of a tracker is limited to fewer degrees-of-freedom, by intersecting the device ray with the screen and using a cursor in the screen plane. This new ray-screen technique is compared to similar 2DOF mouse-based pointing techniques, and a standard ray.

Overall, the experimental methods proposed in these chapters provide researchers with a set of tools for comparing 2D and 3D pointing technique. More importantly, these methods are also validated; each included a mouse pointing technique that was directly comparable to standard 2D pointing and each used throughput, computed according to the ISO 9241-9 standard. As discussed earlier, a primary consideration in the use of this measure is that it tends to be more comparable between studies using the same conditions. The throughput scores reported in these studies for the standard 2D mouse pointing techniques are quite consistent between the studies presented in this dissertation. More importantly, they are also quite consistent with the 2D pointing literature on the

whole. Thus, the available evidence suggests that these methods are reliable for comparing 2D and 3D pointing. Consequently, a recommendation that results from this work is that researchers consider the use of similar methods for the evaluation of 3D pointing techniques, especially when attempting to compare these directly to 2D techniques.

6.2 Technical Issues and Pointing

Using the methodologies described above, this dissertation also investigated several factors thought to influence performance in 3D pointing tasks. These include latency, jitter, tactile feedback.

Chapter 3 investigated the effects of latency and jitter on pointing tasks. This was directly motivated by the (lack of) results of previous work [85] which did not take these factors into consideration when comparing mouse-based and tracker-based 3D manipulation.

The results of Experiments 3-1 and 3-2 suggest that latency has a greater impact on performance than small amounts of jitter. For the measured ~35 ms difference in latency and 0.4 mm (peak-to-peak) jitter between the mouse and tracker used in the study, the latency yielded a 15% difference in performance, while the jitter had no measurable effect.

A major contribution of this work is the result that designers must consider the *tradeoff* between latency and jitter. This launched a separate series of experiments [63] further investigating the tradeoff. One can reduced jitter by smoothing noisy input with a

filtering technique, but such techniques introduce additional latency. Similarly, one can use cursor motion prediction to effectively reduce latency, but inaccuracies in the prediction will introduce additional latency. Hence one can trade latency for jitter, and vice versa. However, these results indicate that there is some threshold where latency has a greater impact on performance. Subsequent results [63] indicate that larger amounts of jitter also affect performance.

Consequently, 3D interactive system designers concerned with pointing device performance are encouraged to first measure *both* the baseline latency and jitter present in their input device. They can then determine the tradeoff of latency and jitter by simply filtering the jitter (e.g., with several size of moving windows), re-measuring latency and jitter, and observing if performance increases or decreases. A small pilot study with a few participants should be sufficient to detect these differences reliably, and will help designers determine a comfortable performance threshold on the spectrum between high latency/low jitter and high jitter/low latency.

The experiments of Chapter 4 partly look at the importance of tactile feedback. Tactile feedback, especially in the form of a supporting surface, is another factor which helps explain why the mouse tends to outperform 3D trackers held in free space. Experiments 4-1 and 4-2 include conditions requiring target selection *at* and *in front of* the screen surface. Results indicate that performance in a stylus-based pointing condition fell dramatically in the absence of tactile feedback, i.e., trying to select targets without

being able to feel them. Unfortunately, without using a haptic system to simulate the sense of touch, this is a difficult problem to solve

6.3 Cursor Visualization, Stereo Display, and Target Depth

This section discusses three related issues in 3D pointing. VR systems frequently use stereo display, and this is becoming more common in games as well. The work presented in this dissertation used stereo display almost exclusively, but under the consideration that it introduces certain issues such as depth cue conflicts and diplopia. These factors were the subject of the investigation outlined in Chapter 5. The one-eyed cursor was used to avoid these issues, and was evaluated in Chapter 5. Target depth is also investigated throughout several of the experiments, and is discussed in this section

Somewhat surprisingly, even in the presence of stereo display, visual feedback mechanisms are insufficient. This is especially evident in the results of Experiments 4-2 and 4-3. Despite consistently highlighting the targets when touched by the stylus, participants generally resorted to a "hunting" type task – gradually pushing the stylus toward targets until it highlighted. This behaviour was measurably worse when targets were displayed at varying depths, as reflected by the throughput scores of Experiments 4-3. Experiments 4-1 and 4-2 used only planar targets; hence participants seemed better able to keep the stylus at a consistent depth, even without tactile feedback. When target depth varied, as in Experiment 4-3, this behaviour was impossible, and performance was worse as a result.

This is consistent with previous findings by Ware et al. [94] that suggest that the dimensionality of the task (or degrees-of-freedom of input) are a key consideration. The studies of Chapter 5 address this consideration by using only techniques that require the same numbers of degrees of freedom. Ray-screen, and even standard remote ray-pointing are arguably 2DOF techniques, despite allowing a greater number of degrees-of-freedom from the input device; one can control either technique using only the orientation of the device, or even only the position of the device (although the latter is likely an awkward means of control).

Ultimately, ray-screen controls the 2D position of a cursor in the screen plane, while standard ray pointing controls the position of a cursor as it slides along the surfaces pointed to by the ray. Target depth is handled automatically by both techniques, albeit in different ways. With ray-screen, the only influence of depth relates to the screen-space projection of targets. This was explored in depth in Experiment 5-2, and using screen-space projected parameters in a Fitts' law model were found to produce a reasonable fit. Standard ray pointing, while in essence a 2DOF technique, may still be affected by target depth more greatly because imprecision in pointing "amplifies" down the ray [44]. Consequently, ray-screen is recommended for consideration by system designers, as it limits the input degrees-of-freedom, and appears to be less affected by depth than standard ray pointing. While it does not perform as well as mouse pointing, it may be useful in systems where a mouse is inappropriate (e.g., walking/standing VR and AR systems).

Another issue investigated by the experiments of Chapter 5 is the appearance of the cursor. This is based on early work by Ware [98] which suggested that a "one-eyed" (monoscopic) cursor can be used with a mouse and affords better performance than a 3DOF tracker. The results presented in Chapter 5 confirm this for certain cases. Certainly the presence of stereo benefitted the standard ray-condition. In all other techniques, the one-eyed cursor performed better, especially for deeper targets. This is an important, and timely result. For example, recent graphics hardware can automatically adapt games to stereo 3D, but often, the cursor is placed in the incorrect depth plane, and is by default rendered in stereo. The results of this chapter show how much this can impact performance. The greater the depth difference between a stereo cursor and a stereo target it overlaps, the worse performance becomes. The one-eyed cursor appears to be an effective solution to this problem for the most part. Other options, such as positioning the cursor at the nearest depth behind it may also be effective.

6.4 Limitations and Future Work

One aspect of using the ISO 9241-9 standard (and Fitts' law, in general) as an experimental framework is that it uses "abstract" pointing motions. As argued in earlier chapters this is often a benefit. But there are some limitations to this as well. Abstract pointing motions may not be representative of real-world use cases. Consequently, this really gives a "best-case" estimate of performance in an unrealistic task.

One could address this limitation with additional experimentation using more complex 3D tasks. This could help determine if the relative ranking of techniques

experimentally established in the preceding chapters holds in more realistic scenarios. To improve the external validity of the results, one could use additional types of VR systems for such experimentation, such as large-display systems and/or haptics. The experiments of Chapter 4, for example, would extend naturally towards a haptic system. Large display systems would also allow consideration of more complicated scenarios, such as when the user is walking, standing, or where manipulation tasks are integrated with the user motions.

6.5 Conclusion

As mentioned earlier, this dissertation evolved out of an earlier investigation of why 3D interaction is rarely used in "serious" applications. The answer does not appear to be simple; one cannot claim, for example, that the difficulty is due to latency alone. Naturally, if it was the case that a single factor was the problem, 3D tracker designers could easily address it by improving hardware. However, the results of this research indicate that the factors discussed above all play a role in the difficulty of 3D pointing tasks. Since these tasks are the "building blocks" of more complex 3D tasks, it appears as though there is no quick and easy answer to improve these interfaces. Moreover, human perception considerations, such as difficulty in handling multi-DOF tasks are less easily addressed: a fundamental principle of HCI is that while technology can be relatively easily changed, humans cannot.

There does appear to be promise in development of DOF-limiting techniques, such as ray-screen, and careful use of stereo display, e.g., in the form of one-eyed

cursors. Both of these techniques "side-step" the issues of human perceptual limitations, and reduce the impact of technological limitations (e.g., the absence of a supporting surface). Although such ideas are somewhat removed from the dream of virtual reality – to move and interact with a computer-generated environment as easily as the real-world – such departures may be necessary to improve the practicality of these systems.

References

1. Agilent, Agilent adns-3080 optical mouse sensor datasheet: Agilent Technologies, 2005.

2. Allison, R. S., Harris, L. R., Jenkin, M., Jasiobedzka, U., and Zacher, J. E., Tolerance of temporal delay in virtual environments, in *Proceedings of the Virtual Reality 2001 Conference (VR'01)*: IEEE Computer Society, 2001.

3. Argelaguet, F. and Andujar, C., Efficient 3d pointing selection in cluttered virtual environments, *Computer Graphics and Applications, IEEE, 29*, 2009, 34-43.

4. Arsenault, R. and Ware, C., Eye-hand co-ordination with force feedback, in *Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI '00*. The Hague, The Netherlands: ACM, 2000, 408-414.

5. Arsenault, R. and Ware, C., The importance of stereo and eye-coupled perspective for eye-hand coordination in fish tank VR, *Presence: Teleoperators & Virtual Environments, 13*, 2004, 549-559.

6. Arthur, K. W., Booth, K. S., and Ware, C., Evaluating 3D task performance for fish tank virtual worlds, *ACM Trans. Inf. Syst., 11*, 1993, 239-265.

7. Bade, R., Ritter, F., and Preim, B., Usability comparison of mouse-based interaction techniques for predictable 3D rotation, in *Smart graphics*, vol. 3638, *Lecture notes in computer science*, (A. Butz, B. Fisher, A. Krüger, and P. Olivier, Eds.). Springer Berlin / Heidelberg, 2005, 924-924.

8. Balakrishnan, R. and MacKenzie, I. S., Performance differences in the fingers, wrist, and forearm in computer input control, in *Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI '97*. Atlanta, Georgia, United States: ACM, 1997.

9. Bérard, F., Ip, J., Benovoy, M., El-Shimy, D., Blum, J., and Cooperstock, J., Did "Minority Report" get it wrong? Superiority of the mouse over 3d input devices in a 3D placement task, in *Human-Computer Interaction – Interact 2009*, vol. 5727, *Lecture Notes in Computer Science*, (T. Gross, J. Gulliksen, P. Kotzé, L. Oestreicher, P. Palanque, R. Prates, and M. Winckler, Eds.). Springer Berlin / Heidelberg, 2009, 400-414.

10. Bier, E., Skitters and jacks: Interactive 3D positioning tools, in *Proceedings of the 1986 Workshop on Interactive 3D Graphics*. Chapel Hill, North Carolina, United States: ACM, 1987, 183-196.

11. Boritz, J. and Booth, K. S., A study of interactive 3d point location in a computer simulated virtual environment, in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology - VRST '97*. Lausanne, Switzerland: ACM, 1997, 181-187.

12. Boritz, J. and Booth, K. S., A study of interactive 6 dof docking in a computerized virtual environment, in *Proceedings of the Virtual Reality Annual International Symposium*. IEEE Computer Society, 1998, 139-146.

13. Bowman, D., A, Johnson, D., B., and Hodges, L., F. , Testbed evaluation of virtual environment interaction techniques, in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology - VRST '99*. London, United Kingdom: ACM, 1999, 26-33.

14. Bowman, D., A. , Kruijff, E., LaViola, J., J. , and Poupyrev, I., 3D *User Interfaces: Theory and Practice*. Addison Wesley Longman Publishing Co., Inc., 2004.

15. Bowman, D. A. and Hodges, L. F., An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments, in *Proceedings of the ACM Symposium on Interactive 3D Graphics - SI3D '97*. Providence, Rhode Island, United States: ACM, 1997, 35-38.

16. Bowman, D. A., Wingrave, C. A., Campbell, J. M., Ly, V. Q., and Rhoton, C. J., Novel uses of pinch gloves for virtual environment interaction techniques, *Virtual Reality, 6*, 2002, 122 - 129.

17. Casiez, G. and Roussel, N., No more bricolage!: Methods and tools to characterize, replicate and compare pointing transfer functions, in *Proceedings of the ACM Symposium on User Interface Software and Technology - UIST '11*. Santa Barbara, California, USA: ACM, 2011, 603-614.

18. Castellucci, S. J., Teather, R. J., and Pavlovych, A., Novel metrics for 3d remote pointing, in *ACM Symposium on Spatial User Interaction - SUI '13*. New York: ACM Press, 2013, *to appear*.

19. Conner, B. D., Snibbe, S., S. , Herndon, K., P, Robbins, D., C. , Zeleznik, R., C. , and van, D., Andries Three-dimensional widgets, in *Proceedings of the*

Symposium on Interactive 3D Graphics - SI3D '92. Cambridge, Massachusetts, United States: ACM, 1992, 183-188.

20. Cruz-Neira, C., Sandin, D. J., DeFanti, T. A., Kenyon, R. V., and Hart, J. C., The CAVE: Audio visual experience automatic virtual environment, *Communications of the ACM, 35*, 1992, 64-72.

21. Dang, N.-T., A survey and classification of 3D pointing techniques, in *IEEE International Conference on Research, Innovation and Vision for the Future*: IEEE Press, 2007, 71 - 80.

22. Eitoku, S., Tanikawa, T., and Suzuki, Y., Display composed of water drops for filling space with materialized virtual three-dimensional objects, in *IEEE Virtual Reality Conference - VR '06*. IEEE Computer Society, 2006, 159-166.

23. Ellis, S. R., Breant, F., Manges, B., Jacoby, R., and Adelstein, B. D., Factors influencing operator interaction with virtual objects viewed via head-mounted see-through displays: Viewing conditions and rendering latency, *Virtual Reality Annual International Symposium.* IEEE Computer Society, 1997, 138-145.

24. Ellis, S. R., Young, M. J., Adelstein, B. D., and Ehrlich, S. M., Discrimination of changes of latency during voluntary hand movements of virtual objects, in *Proceedings of the Human Factors and Ergonomics Society Conference.* Houston, Texas, 1999, 1182-1186.

25. Fitts, P. M., The information capacity of the human motor system in controlling the amplitude of movement, *Journal of Experimental Psychology, 47*, 1954, 381-391.

26. Foxlin, E., Motion tracking requirements and technologies, in *Handbook of virtual environments: Design, implementation and applications*, (K. M. Stanney, Ed.). Lawrence Erlbaum, 2002, 163 - 210.

27. Groen, J. and Werkhoven, P. J., Visuomotor adaptation to virtual hand position in interactive virtual environments, *Presence: Teleoperators & Virtual Environments, 7*, 1998, 429-446.

28. Grossman, T. and Balakrishnan, R., Pointing at trivariate targets in 3D environments, in *Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI '04*. Vienna, Austria: ACM, 2004, 447-454.

29. Grossman, T. and Balakrishnan, R., The bubble cursor: Enhancing target acquisition by dynamic resizing of the cursor's activation area, in *Proceedings of*

the ACM Conference on Human Factors in Computing Systems - CHI '05. Portland, Oregon, USA: ACM, 2005, 281-290.

30. Grossman, T. and Balakrishnan, R., The design and evaluation of selection techniques for 3D volumetric displays, in *Proceedings of the ACM Symposium on User Interface Software and Technology - UIST '06.* Montreux, Switzerland: ACM, 2006, 3-12.

31. Grossman, T. and Balakrishnan, R., An evaluation of depth perception on volumetric displays, in *Proceedings of the Working Conference on Advanced Visual Interfaces - AVI '06.* Venezia, Italy: ACM, 2006.

32. Ha, T. and Woo, W., An empirical evaluation of virtual hand techniques for 3d object manipulation in a tangible augmented reality environment, in *IEEE Symposium on 3D User Interfaces - 3DUI '10.* IEEE Press, 2010, 91 - 98.

33. Hearn, D. and Baker, M. P., *Computer Graphics with OpenGL 3rd edition.* Pearson Prentice Hall, 2004.

34. Held, R., Efstathiou, A., and Greene, M., Adaptation to displaced and delayed visual feedback from the hand, *Journal of Experimental Psychology, 72,* 1966, 887 - 891.

35. Henriksen, K., Sporring, J., and Hornbaek, K., Virtual trackballs revisited, *IEEE Transactions on Visualization and Computer Graphics, 10,* IEEE Press 2004, 206-216.

36. Hoffman, D. M., Girshick, A. R., Akeley, K., and Banks, M. S., Vergence-accommodation conflicts hinder visual performance and cause visual fatigue, *Journal of Vision, 8(3),* 2008, 1-30.

37. Howarth, P. A., Potential hazards of viewing 3-d stereoscopic television, cinema and computer games: A review, *Ophthalmic and Physiological Optics, 31,* 2011, 111 - 122.

38. ISO 9241-9, Ergonomic requirements for office work with visual display terminals (vdts) - part 9: Requirements for non-keyboard input devices. International Standard, International Organization for Standardization, 2000.

39. Jagacinski, R. J. and Flach, J. M., Chapter 4 - information theory and Fitts' law, in *Control theory for humans: Quantitative approaches to modeling performance*). Lawrence Earlbaum Associates, Inc, 2003.

40. Jota, R., Nacenta, M. A., Jorge, J. A., Carpendale, S., and Greenberg, S., A comparison of ray pointing techniques for very large displays, in *Graphics Interface 2010*. Canadian Information Processing Society, 2010, 269-276.

41. Kabbash, P. and Buxton, W. A. S., The "Prince" technique: Fitts' law and selection using area cursors, in *Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI '95*. Denver, Colorado, United States: ACM Press/Addison-Wesley Publishing Co., 1995, 273-279.

42. Keates, S., Hwang, F., Langdon, P., Clarkson, P. J., and Robinson, P., Cursor measures for motion-impaired computer users, in *Proceedings of the ACM Conference on Assistive Technologies*. Edinburgh, Scotland: ACM, 2002, 135-142.

43. Kohli, L. and Whitton, M., The haptic hand: Providing user interface feedback with the non-dominant hand in virtual environments, in *Proceedings of Graphics Interface 2005*. Victoria, British Columbia: Canadian Human-Computer Communications Society, 2005, 1-8.

44. Kopper, R., Bowman, D. A., Silva, M. G., and McMahan, R. P., A human motor behavior model for distal pointing tasks, *International Journal of Human-Computer Studies, 68*, 2010, 603-615.

45. Kunert, A., Kulik, A., Lux, C., and Fröhlich, B., Facilitating system control in ray-based interaction tasks, in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology - VRST '09*. Kyoto, Japan: ACM, 2009, 183-186.

46. Lindeman, R. W., Sibert, J. L., and Hahn, J. K., Hand-held windows: Towards effective 2D interaction in immersive virtual environments, in *Proceedings of the IEEE Virtual Reality Conference - VR '99*. IEEE Press, 1999, 205-212.

47. Lindeman, R. W., Sibert, J. L., and Hahn, J. K., Towards usable VR: An empirical study of user interfaces for immersive virtual environments, in *Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI '99*. Pittsburgh, Pennsylvania, United States: ACM, 1999, 64-71.

48. Liu, L., Liere, R. v., Nieuwenhuizen, C., and Martens, J.-B., Comparing aimed movements in the real world and in virtual reality, in *Proceedings of the IEEE Virtual Reality Conference - VR '09*. Lafayette, Lousiana: IEEE, 2009, 219-222.

49. MacKenzie, I. S., Fitts' law as a research and design tool in human-computer interaction, *Human-Computer Interaction, 7*, 1992, 91-139.

50. MacKenzie, I. S. and Colin, W., Lag as a determinant of human performance in interactive systems, in *Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI '93*. Amsterdam, The Netherlands: ACM, 1993, 488-493.

51. MacKenzie, I. S. and Isokoski, P., Fitts' throughput and the speed-accuracy tradeoff, in *Proceeding of the ACM Conference on Human Factors in Computing Systems - CHI '08*. Florence, Italy: ACM, 2008, 1633-1636.

52. MacKenzie, I. S., Kauppinen, T., and Silfverberg, M., Accuracy measures for evaluating computer pointing devices, in *Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI 2001*. New York: ACM, 2001, 9 - 16.

53. Meyer, D. E., Abrams, R. A., Kornblum, S., Wright, C. E., and Smith, J. E. K. , Optimality in human motor performance: Ideal control of rapid aimed movements. *Psychological Review, 95*, 1988, 340-370.

54. Mine, M., R, *Characterization of end-to-end delays in head-mounted display systems*, Technical Report, University of North Carolina at Chapel Hill, 1993.

55. Mine, M. R., Frederick P. Brooks, J., and Sequin, C. H., Moving objects in space: Exploiting proprioception in virtual-environment interaction, in *Proceedings of the ACM Conference on Computer Graphics and Interactive Techniques*: ACM Press/Addison-Wesley Publishing Co., 1997, 19-26.

56. Mulder, J. D. and Liere, R. V., Enhancing fish tank VR, in *Proceedings of the IEEE Conference on Virtual Reality - VR '00.* IEEE Computer Society, 2000, 91-98.

57. Murata, A. and Iwase, H., Extending fitts' law to a three-dimensional pointing task, *Human Movement Science, 20*, 2001, 791-805.

58. NaturalPoint, Naturalpoint Optitrack: http://www.naturalpoint.com/optitrack/ NaturalPoint, Inc., 2008. Accessed July 4, 2013.

59. Nieuwenhuizen, K., Liu, L., Liere, R. v., and Martens, J.-B., Insights from dividing 3D goal-directed movements into meaningful phases, *IEEE Computer Graphics and Applications, 29*, 2009, 44 - 53.

60. Oh, J.-Y. and Stuerzlinger, W., Laser pointers as collaborative pointing devices, in *Proceedings of Graphics Interface '02.* Canadian Information Processing Society, 2002, 315-320.

61. Oh, J.-Y. and Stuerzlinger, W., Moving objects with 2D input devices in CAD systems and desktop virtual environments, in *Proceedings of Graphics Interface '05*. Victoria, British Columbia: Canadian Human-Computer Communications Society, 2005, 195-202.

62. Pagano, R. R., *Understanding statistics in the behavioural sciences*. Wadsworth Publishing, 2006.

63. Pavlovych, A. and Stuerzlinger, W., The tradeoff between spatial jitter and latency in pointing tasks, in *Proceedings of the ACM Symposium on Engineering Interactive Computing Systems - EICS '09*. Pittsburgh, PA, USA: ACM, 2009, 187-196.

64. Poupyrev, I., Billinghurst, M., Weghorst, S., and Ichikawa, T., The go-go interaction technique: Non-linear mapping for direct manipulation in VR, in *Proceedings of the ACM Symposium on User Interface Software and Technology - UIST '92*. Seattle, Washington, United States: ACM, 1996, 79-80.

65. Poupyrev, I., Ichikawa T., Weghorst, S., and Billinghurst, M., Egocentric object manipulation in virtual environments: Empirical evaluation of interaction techniques, in *Proceedings of Eurographics '98*, vol. 17, 1998, 41-52.

66. Poupyrev, I., Tomokazu, N., and Weghorst, S., Virtual notepad: Handwriting in immersive VR, *in Proceedings of the Virtual Reality Annual International Symposium, 1998*. IEEE Computer Society, 1998, 126-132.

67. Poupyrev, I., Weghorst, S., and Fels, S., Non-isomorphic 3D rotational techniques, in *Proceedings of the ACM Conference on Human Factors in Computing Systems*. The Hague, The Netherlands: ACM, 2000, 540-547.

68. Schmitt, B., Raynal, M., Dubois, E., and Croenne, D., A composite approach to evaluate two interaction techniques for a 3D pointing task, in *Proceedings of the IEEE Symposium on 3D User Interfaces - 3DUI '12*. IEEE Press, 2012, 159-160.

69. Shibata, T., Kawai, T., Otsuki, M., Miyake, N., Yoshihara, Y., and Iwasaki, T., Stereoscopic 3-d display with optical correction for the reduction of the discrepancy between accommodation and convergence, *Journal of the Society for Information Display, 13*, 2005, 665-671.

70. Shoemake, K., Arcball: A user interface for specifying three-dimensional orientation using a mouse, in *Proceedings of Graphics Interface '92*. Vancouver, British Columbia, Canada: Morgan Kaufmann Publishers Inc., 1992.

71. Shuralyov, D. and Stuerzlinger, W., A 3D desktop puzzle assembly system, in *Proceedings of the IEEE Symposium on 3D User Interfaces - 3DUI '11*, IEEE Press, 2011, 139-140.

72. Slater, M., Linakis, V., Usoh, M., and Kooper, R., Immersion, presence, and performance in virtual environments: An experiment with tri-dimensional chess, in *ACM Symposium on Virtual Reality Software and Technology - VRST '96*. New York: ACM, 1996, 163-172.

73. Slocum, J., Chaparro, A., McConnell, D., and Bohan, M., Comparing computer input devices using kinematic variables, in *Human Factors and Ergonomics Society 49th Annual Meeting*: Human Factors and Ergonomics Society, 2005, 711-715.

74. Smith, G., Salzman, T., and Stuerzlinger, W., 3D scene manipulation with 2D devices and constraints, in *Proceedings of Graphics Interface '01*. Ottawa, Ontario, Canada: Canadian Information Processing Society, 2001, 135-142.

75. Soukoreff, R. W. and MacKenzie, I. S., Towards a standard for pointing device evaluation, perspectives on 27 years of fitts' law research in HCI, *International Journal of Human-Computer Studies 61*, 2004, 751-789.

76. Sprague, D. W., Po, B. A., and Booth, K. S., The importance of accurate VR head registration on skilled motor performance, in *Proceedings of Graphics Interface '06*. Quebec, Canada: Canadian Information Processing Society, 2006, 131-137.

77. Steed, A., A simple method for estimating the latency of interactive, real-time graphics simulations, in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology - VRST '08*. Bordeaux, France: ACM, 2008, 123-129.

78. Steinicke, F., Ropinski, T., and Hinrichs, K., Object selection in virtual environments using an improved virtual pointer metaphor, in *Computer Vision and Graphics*, vol. 32, *Computational Imaging and Vision*, (K. Wojciechowski, B. Smolka, H. Palus, R. S. Kozera, W. Skarbek, and L. Noakes, Eds.). Springer Netherlands, 2006, 320-326.

79. Stoakley, R., Conway, M. J., and Pausch, R., Virtual reality on a WIM: Interactive worlds in miniature, in *Proceedings of the ACM Conference on Human Factors in Computing Systems*. Denver, Colorado, United States: ACM Press/Addison-Wesley Publishing Co., 1995, 265-272.

80. Strauss, P. S. and Carey, R., An object-oriented 3D graphics toolkit, *ACM Computer Graphics, 26*, 1992, 341-349.

81. Szalavári, Z. and Gervautz, M., The personal interaction panel - a two-handed interface for augmented reality, *Computer Graphics Forum, 16*, 1997, 335 - 346.

82. Teather, R. J., Allison, R. S., and Stuerzlinger, W., Evaluating visual/motor co-location in fish tank virtual reality, in *IEEE Toronto International Conference - Human Factors and Ergonomics Symposium.* IEEE Press, 2009, 624 - 629.

83. Teather, R. J., Pavlovych, A., Stuerzlinger, W., and MacKenzie, I. S., Effects of tracking technology, latency, and spatial jitter on object movement, in *Proceedings of the IEEE Symposium on 3D User Interfaces - 3DUI '09.* Lafayette, Lousiana, USA: IEEE, 2009, 43-50.

84. Teather, R. J. and Stuerzlinger, W., Guidelines for 3D positioning techniques, in *Proceedings of the ACM Conference on Future Play '07.* Toronto, Canada: ACM, 2007, 61-68.

85. Teather, R. J. and Stuerzlinger, W., Assessing the effects of orientation and device on (constrained) 3D movement techniques, in *Proceedings of the IEEE Symposium on 3D User Interfaces - 3DUI '08.* Reno, Nevada, USA: IEEE Press, 2008, 43-50.

86. Teather, R. J. and Stuerzlinger, W., Pointing at 3D targets in a stereo head-tracked virtual environment, in *Proceedings of the IEEE Symposium on 3D User Interfaces - 3DUI '11.* Worcester, MA, USA: IEEE Press, 2011, 87-94.

87. Teather, R. J. and Stuerzlinger, W., Cursors for 3D pointing, in *3DCHI Workshop*, 2012.

88. Teather, R. J. and Stuerzlinger, W., A system for evaluating 3D pointing techniques, in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology - VRST '12.* Toronto: ACM Press, 2012, 209.

89. Valkov, D., Steinicke, F., Bruder, G., and Hinrichs, K., 2D touching of 3D stereoscopic objects, in *Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI '11.* Vancouver, BC, Canada: ACM, 2011, 1353-1362.

90. Waloszek, G., Vision and visual disabilities: An introduction, vol. 2011: SAP Design Guild, 2005.

91. Wang, Y. and MacKenzie, C. L., The role of contextual haptic and visual constraints on object manipulation in virtual environments, in *Proceedings of the*

ACM Conference on Human Factors in Computing Systems - CHI '00. New York: ACM, 2000, 532-539.

92. Ware, C., *Information visualization*. Morgan Kaufmann, 2000.

93. Ware, C. and Arsenault, R., Frames of reference in virtual object rotation, in *Proceedings of the Symposium on Applied Perception in Graphics and Visualization*. Los Angeles, California: ACM, 2004, 135-141.

94. Ware, C., Arthur, K., and Booth, K. S., Fish tank virtual reality, in *Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI '93*. Amsterdam, The Netherlands: ACM, 1993, 37-42.

95. Ware, C. and Balakrishnan, R., Reaching for objects in VR displays: Lag and frame rate, *1*, 1994, 331-356.

96. Ware, C. and Franck, G., Evaluating stereo and motion cues for visualizing information nets in three dimensions, *ACM Transactions on Graphics*, *15*, 1996, 121-140.

97. Ware, C. and Jessome, D., R, Using the bat: A six-dimensional mouse for object placement, *IEEE Computer Graphics and Applications*, *8*, 1988, 65-70.

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

99. Ware, C. and Mitchell, P., Visualizing graphs in three dimensions, *ACM Transactions on Applied Perception*, *5*, 2008, 1-15.

100. Wickens, C. and Hollands, J., Spatial displays, in *Engineering psychology and human performance (3rd edition)*). Prentice Hall, 1999.

101. Wingrave, C. A., Tintner, R., Walker, B. N., Bowman, D. A., and Hodges, L. F., Exploring individual differences in raybased selection; strategies and traits, in *Proceedings of the IEEE Conference on Virtual Reality - VR '05*: IEEE Computer Society, 2005, 163-170.

102. Woodworth, R. S., The accuracy of voluntary movement, *Psychological Review*, *3*, 1899, 1 - 114.

103. Zhai, S., Buxton, W., and Milgram, P., The "silk cursor": Investigating transparency for 3D target acquisition, in *ACM Conference on Human Factors in Computing Systems - CHI '94.* ACM Press, 1994, 459 - 464.

104. Zhai, S., Milgram, P., and Buxton, W., The influence of muscle groups on performance of multiple degree-of-freedom input, in *Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI '96.* Vancouver, British Columbia, Canada: ACM, 1996.

Appendix
Subjective Questionnaire Results

Participant Demographics

Participant demographic data are provided below for each experiment. In all experiments, game playing experience was assessed with a 5-point Likert scale, using the following question:

How often do you play 3D games?

5 - Very frequently (every day)
4- Frequently (several times per week)
3- Infrequently (several times per month)
2 - Rarely (one or twice per month)
1- Never (I never play games)

Chapter 3 Experiments

Table A-1.Chapter 3 experiment participant demographics

Experiment 3-1 demographics

Participant	*Age*	*Sex*	*Game Playing*
1	29	m	2
2	26	m	1
3	18	f	1
4	30	m	4
5	29	f	1
6	26	m	4
7	29	m	3
8	32	m	1
9	23	f	1
10	25	f	1
11	30	m	2
12	29	f	1
13	28	m	2
14	27	f	1
AVG	27.2		1.8
SD	3.5		1.4

Experiment 3-2

Participant	*Age*	*Sex*	*Game Playing*
1	25	m	3
2	21	f	3
3	22	m	1
4	24	f	1
5	26	f	4
6	19	f	1
7	25	m	5
8	28	m	3
9	24	f	1
10	24	m	1
11	25	m	2
12	30	f	2
AVG	24.4		2.3
SD	2.9		1.4

Chapter 4 Experiments

Table A-2.Chapter 4 experiment participant demographics

Experiment 4-1

Participant	*Age*	*Sex*	*Game Playing*
1	25	m	1
2	25	f	2
3	26	f	1
4	22	m	2
5	24	m	1
6	22	m	2
7	22	f	2
8	28	f	2
9	22	m	2
10	28	m	2
11	22	m	4
12	22	m	2
AVG	24.0		1.9
SD	2.4		0.8

Experiment 4-2

Participant	*Age*	*Sex*	*Game Playing*
1	20	f	1
2	25	f	2
3	22	f	3
4	22	m	2
5	20	f	1
6	27	m	3
7	29	m	1
8	25	m	2
9	20	m	2
10	29	f	4
11	25	m	2
12	29	m	2
AVG	24.4		2.1
SD	3.6		0.9

Experiment 4-3

Participant	*Age*	*Sex*	*Game Playing*
1	27	m	4
2	21	f	1
3	25	m	2
4	26	m	2
5	22	f	3
6	19	f	1
7	20	f	2
8	26	m	4
9	25	f	2
10	31	m	2
11	25	m	2
12	19	m	1
AVG	23.8		2.2
SD	3.7		1.0

Chapter 5 Experiments

Table A-3.Chapter 5 experiment participant demographics

Experiment 5-1

Participant	*Age*	*Sex*	*Game Playing*
1	19	f	2
2	20	f	1
3	21	f	1
4	27	m	1
5	20	m	2
6	21	m	1
7	20	m	2
8	25	f	2
9	21	m	2
10	18	f	1
11	25	m	4
12	20	m	4
13	25	f	1
14	39	f	3
15	19	f	1
16	30	m	4
AVG	23.1		2
SD	5.4		1.2

Experiment 5-2

Participant	*Age*	*Sex*	*Game Playing*
1	31	m	3
2	20	m	2
3	31	f	4
4	25	m	5
5	26	m	4
6	32	m	1
7	31	m	1
8	29	f	1
9	25	m	1
10	27	m	2
11	39	f	2
12	38	m	1
AVG	29.4		2.3
SD	5.4		1.4

Subjective Questionnaire Scores

Participants also completed a subjective survey and ranking of conditions in each experiment. These questions were based on those recommended by the ISO 9241-9 standard. Following the actual survey questions used:

Q1. Smoothness during operation
- 5 - very smooth
- 4 - smooth
- 3 - average
- 2 - rough
- 1 - very rough

Q2. Mental effort required
- 5 - very low
- 4 - low
- 3 - average
- 2 - high
- 1 - very high

Q3. Physical effort required
- 5 - very low
- 4 - low
- 3 - average
- 2 - high
- 1 - very high

Q4. Accurate targeting was
- 5 - very easy
- 4 - easy
- 3 - average
- 2 - hard
- 1 - very hard

Q5. General comfort
- 5 - very comfortable
- 4 - comfortable
- 3 - average
- 2 - uncomfortable
- 1 - very uncomfortable

Q6. Overall, the condition was
- 5 - very easy to use
- 4 - easy to use
- 3 - average
- 2 - difficult to use
- 1 - very difficult to use

Summary data for the Chapter 4 and 5 experiments follow. Some experiments also include ranking data for each condition.

Experiment 4-1

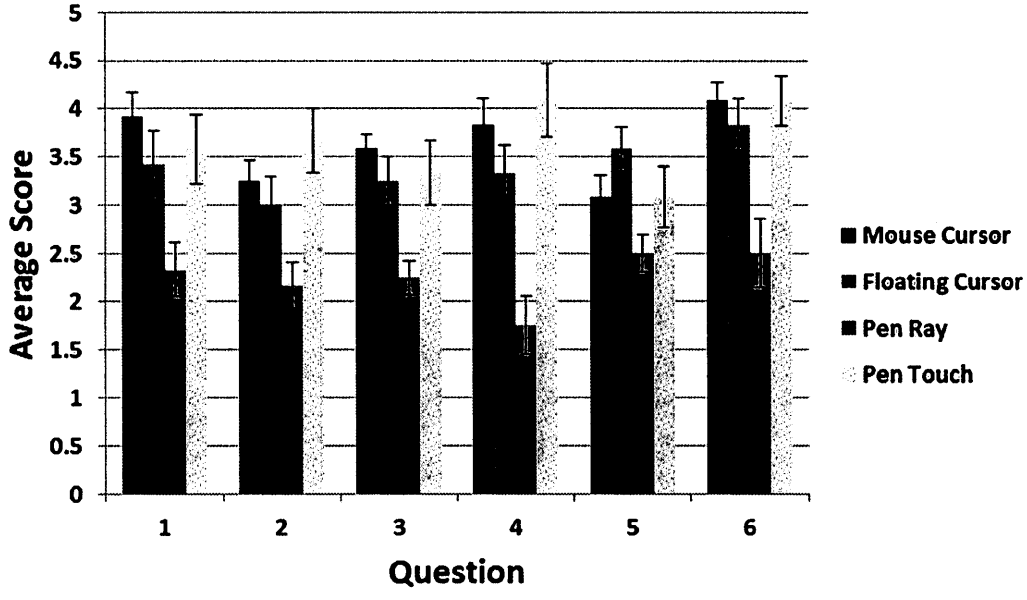

Figure A-1. Subjective Questionnaire results for Exp. 4-1, higher scores are better. Error bars show +/- 1 SE.

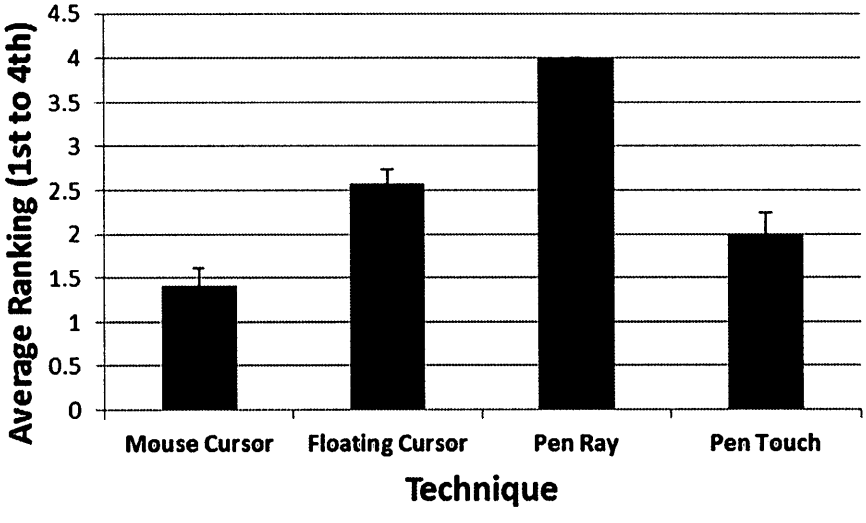

Figure A-2. Exp. 4-1 subjective ranking averages, lower scores are better. Error bars show +/- 1 SE. Note that Pen Ray has 0 variability, as all participants ranked it 4th.

Experiment 4-2

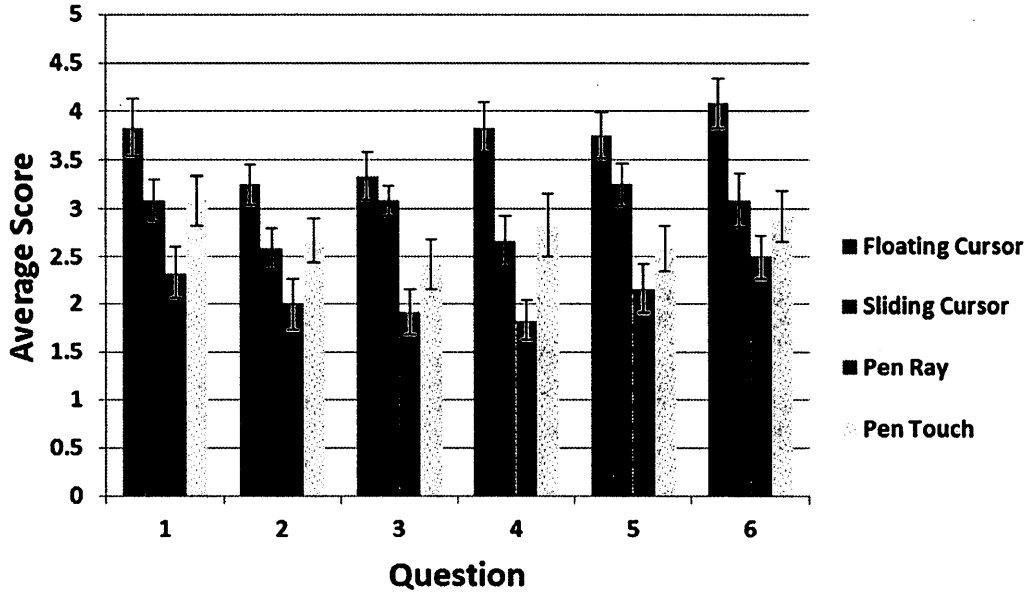

Figure A-3. Subjective Questionnaire results for Exp. 4-2, higher scores are better. Error bars show +/- 1 SE.

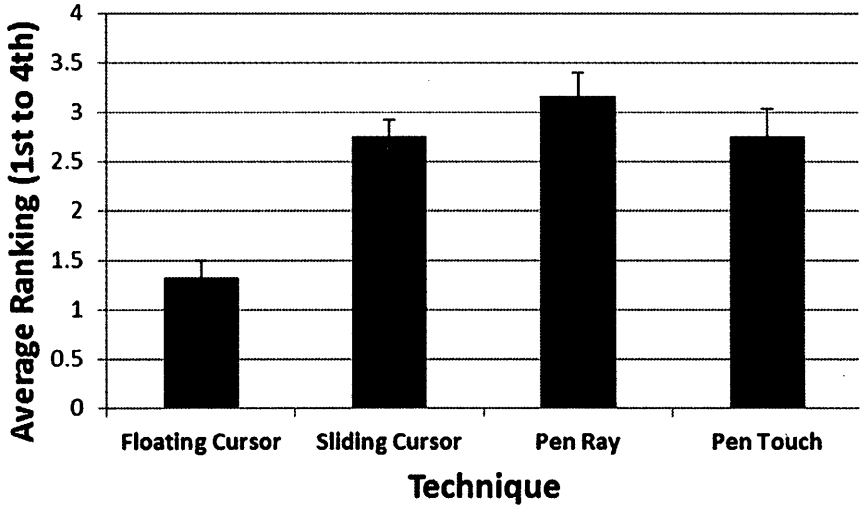

Figure A-4. Experiment 4-2 technique rankings. Note that lower is better. Error bars show +/- 1SE.

Experiment 4-3

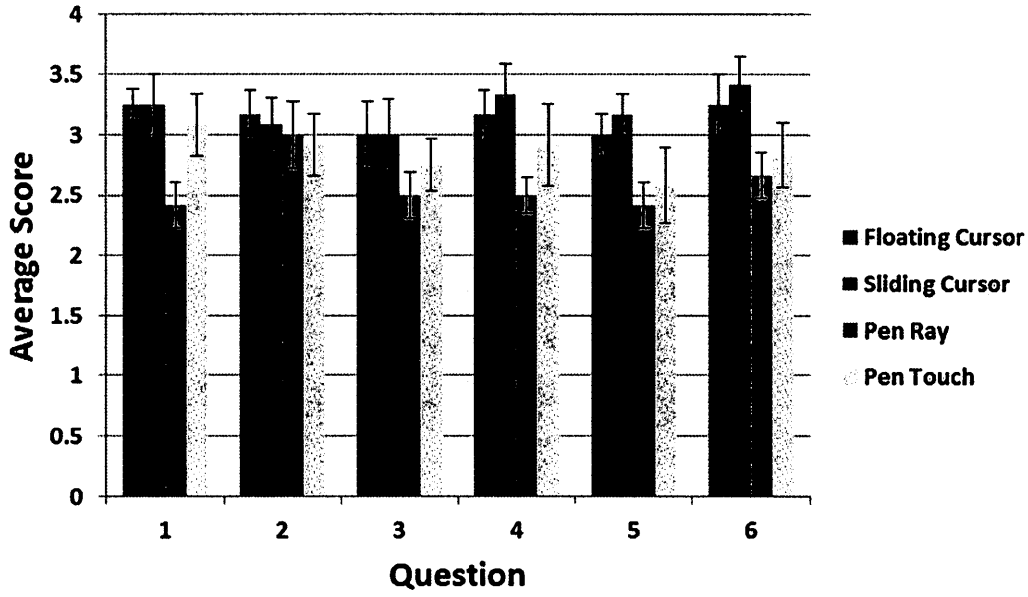

Figure A-5. Subjective Questionnaire results for Exp. 4-3, higher scores are better. Error bars show +/- 1 SE.

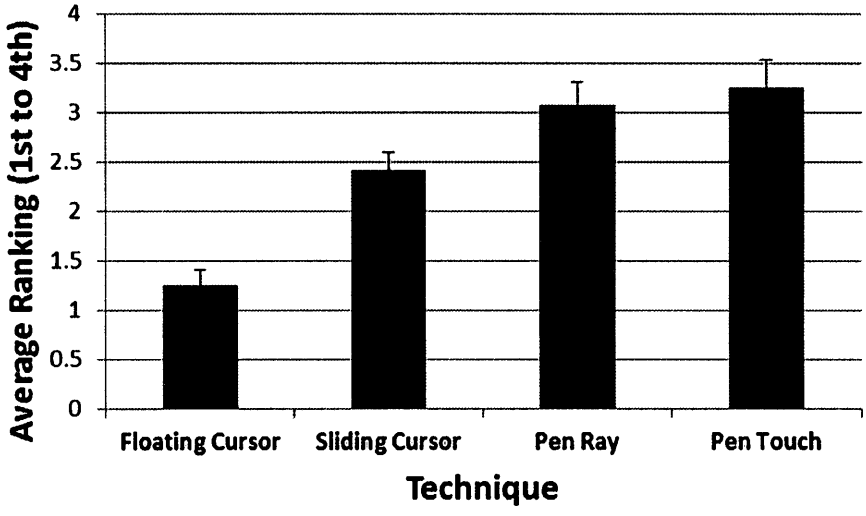

Figure A-6. Experiment 4-3 technique rankings. Note that lower is better. Error bars show +/- 1SE.

Experiment 5-1

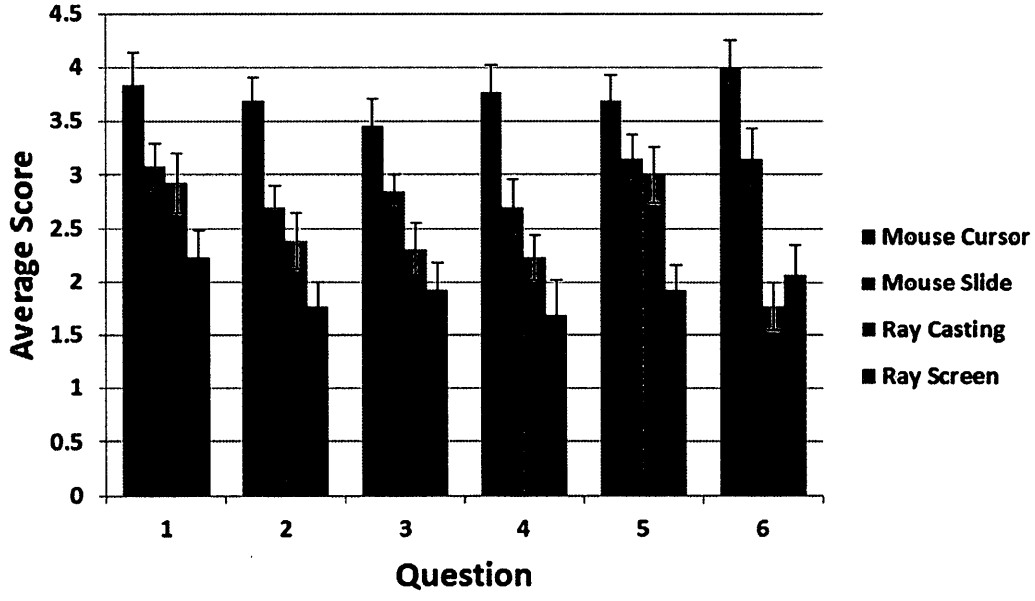

Figure A-7. Subjective Questionnaire results for Exp. 5-1, higher scores are better. Error bars show +/- 1 SE.

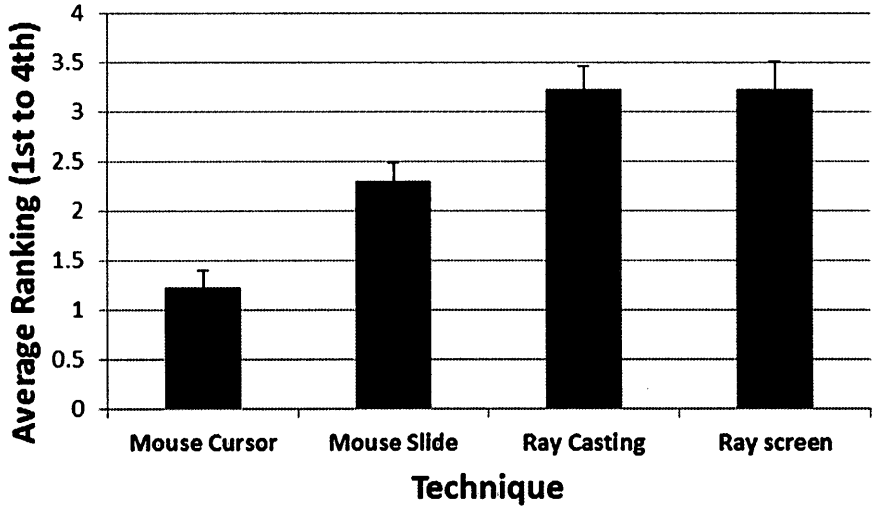

Figure A-8. Experiment 5-1 technique rankings. Note that lower is better. Error bars show +/- 1SE.

Experiment 5-2

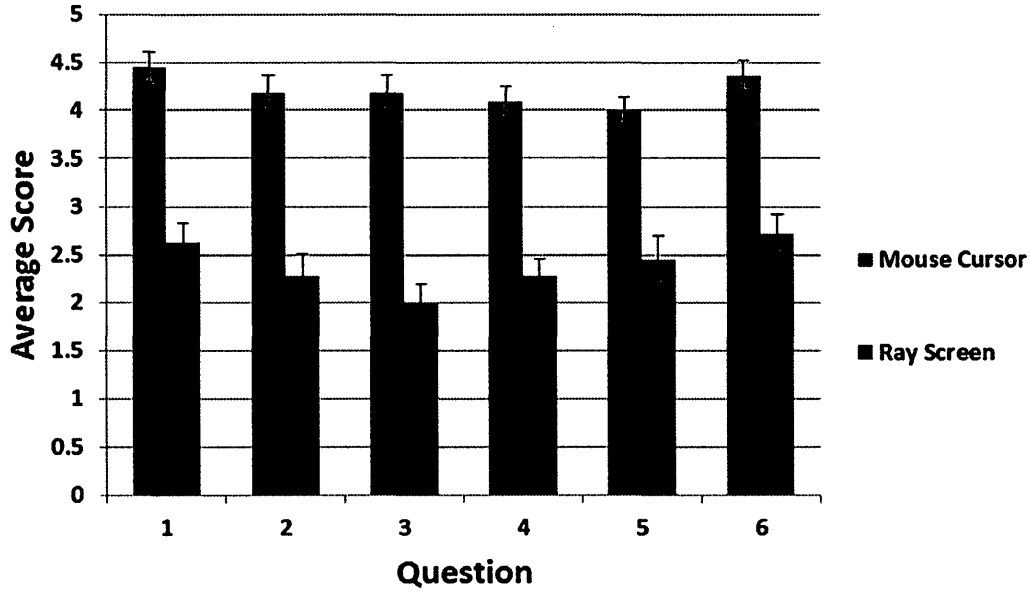

Figure A-9. Subjective Questionnaire results for Exp. 5-2, higher scores are better. Error bars show +/- 1 SE.