EVALUATING 3D INTERACTION TECHNIQUES

ROBERT J. TEATHER

QUALIFYING EXAM REPORT
SUPERVISOR: WOLFGANG STUERZLINGER

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING,
YORK UNIVERSITY
TORONTO, ONTARIO

MAY, 2011
ABSTRACT

One of the greatest hurdles to widespread adoption of 3D user interfaces is the relative lack of easy-to-use techniques for object selection and manipulation. While virtual reality (VR) systems commonly employ 3D trackers as the primary input device for object selection and manipulation, previous work has demonstrated that the desktop mouse can outperform these devices in conceptually identical tasks. However, direct comparison of these devices is difficult and, in general, evaluation of 3D selection and manipulation interfaces is complicated by a number of technical issues and human factors. These include the presence or absence of stereo 3D display, head-tracking, tactile feedback, latency and jitter. This report documents existing methods for comparing input devices, and the effects of each of the aforementioned issues. It also discusses alternative and standardized approaches to the evaluation of 3D pointing interfaces. The objective is to support direct and fair comparison between 3D and 2D input devices (in particular the mouse), for 3D selection and manipulation tasks.
Contents

CHAPTER 1 Introduction ...................................................................................................... 1
  1.1 Outline .................................................................................................................. 2

CHAPTER 2 3D Selection and Manipulation ........................................................................ 4
  2.1 6DOF Input Devices ............................................................................................. 5
    2.1.1 Virtual Hand and Depth Cursor Techniques ............................................. 7
    2.1.2 Ray-based Techniques .............................................................................. 10
    2.1.3 Hybrid Ray/Hand Techniques .................................................................. 14
    2.1.4 Comparing Rays and Virtual Hands ....................................................... 15
  2.2 3D Selection and Manipulation with 2DOF Input Devices .................................. 20
  2.3 Summary ............................................................................................................. 28

CHAPTER 3 Depth Cues, Stereo Graphics, and Head Tracking ......................................... 30
  3.1 Depth Cues – Stereopsis, Convergence, Accommodation .................................... 30
    3.1.1 Depth Cue Conflicts in Stereo Displays ................................................... 34
  3.2 Evaluating Stereo Displays ................................................................................ 36
    3.2.1 Graph Tracing ........................................................................................... 37
    3.2.2 Selection and Manipulation ...................................................................... 37
  3.3 Head Tracking ...................................................................................................... 39
  3.4 Summary ............................................................................................................. 41

CHAPTER 4 Technical Issues – Input Devices and Displays ........................................... 42
Figures

Figure 2-1: Various 3-6DOF input devices................................................................. 6
Figure 2-2: An example virtual hand technique....................................................... 8
Figure 2-3: Ray selection......................................................................................... 10
Figure 2-4: Ray disambiguation ............................................................................. 12
Figure 2-5: Bowman’s selection/manipulation testbed task .................................... 17
Figure 2-6: Mouse ray casting ............................................................................... 22
Figure 2-7: Translation and rotation 3D widgets ..................................................... 24
Figure 2-8: SESAME mouse motion................................................................. 26
Figure 2-9: SESAME wand motion ........................................................................ 27
Figure 3-1: Accommodation to near and far targets ............................................. 32
Figure 3-2: Convergence and convergence angles ............................................... 33
Figure 3-3: Convergence and accommodation conflict .......................................... 34
Figure 5-1: ISO 9241-9 reciprocal tapping task ................................................... 57
Figure 5-2: Distribution of clicks on a circular target ............................................. 58
Figure 5-3: Illustration of effective width and effective distance......................... 59
Chapter 1
Introduction

To this day, true virtual reality (VR) experiences remain largely inaccessible to most people, and see little use outside of lab and industrial settings, amusement parks, or technological derivatives such as video games. This is a far cry from a common belief of the 1990s: that VR would become a dominant computing paradigm employed in many aspects of day-to-day life. Although there are a variety of reasons for this, one major consideration is the relative inefficiency of such interfaces when compared to the popular desktop computer metaphor.

One might speculate that reaching, grabbing and manipulating virtual objects would enable their easy and effective manipulation in virtual environments. 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 stereo vision, head-coupled display, or a supporting surface on which to operate an input device (i.e., passive haptic feedback). Latency and jitter (tracker noise) are also concerns, as both factors are measurably worse in 3D tracking devices than in 2D devices such as the computer mouse. In practice, common VR 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 domains [76-78]. This 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. Thus, system designers must not only choose devices but also software techniques that maximize the efficiency of the user for a given task. In this case, the focus is taken off hardware and placed on software instead.

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 these virtually always use custom experimental designs that vary from study to study. Consequently, it is hard to formally quantify the benefits of certain techniques or devices, leaving practitioners only with general guidelines. In the absence of such quantifiable benefits, VR 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 Outline
The outline of this report is as follows. Chapter 2 documents general 3D object selection and manipulation, and previously developed techniques for these tasks. Chapter 3
discusses two technologies frequently used in virtual reality systems: stereo graphics and head tracking, and will focus in particular on work that evaluates the advantages of these technologies. It also includes a discussion of the depth cues supported by these technologies. Chapter 4 examines system-specific issues that can affect user performance when using these devices, including latency, jitter, haptic feedback, and visual-motor co-location. Chapter 5 investigates experimental methods commonly employed in 2D user interface evaluation. It also discusses the prospect of applying these in the 3D user interface domain. Finally, Chapter 6 concludes the report and discusses potential avenues for future work.
Chapter 2
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 the object. According to Bowman’s taxonomy [11], 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. Selection and manipulation tasks can both be performed using either 2D or 3D input devices, given suitable input mappings. The differences between 3D and 2D selection/manipulation techniques are discussed below in Sections 2.1 and 2.2. Note that this report documents specifically rigid-body manipulation. In other words, object/surface deformation and object cutting/merging are beyond the scope of this document.

Navigation and system control are two other primary tasks commonly required in VR systems [12]. Navigation is further subdivided into travel, the act of actually moving through the virtual environment, and way-finding, the cognitive element of navigation, i.e., determining how to reach the intended destination. System control refers to
operations that perform special, typically indirect operations, such as changing the system mode, or activating alternative features of the system. Navigation and system control tasks are beyond the scope of this report and are not discussed here, except in situations where system control is achieved by the direct manipulation of objects in the environment. Finally, it is worth distinguishing between exocentric and egocentric selection and manipulation techniques. Exocentric techniques, such as worlds-in-miniature [72], 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 of the used perspective. Egocentric techniques are generally more similar to real-world object manipulation, requiring direct interaction with objects, sometimes even using the user’s real hand. Only egocentric selection and manipulation are discussed further in this report.

2.1 6DOF Input Devices

A great deal of VR research focuses on 3D manipulation tasks using 3D input devices such as 6DOF trackers, wands, gloves and haptic devices. Some exemplary 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. A common belief is that such devices may also afford more “natural” interaction with virtual objects, allowing users to leverage their real world object manipulation skills.
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. VE systems based on head-mounted displays, CAVEs [16], or similar immersive projective displays are typical examples. In these types of systems, commercially available 3D tracking systems such as those provided by Intersense (see Figure 2-1 (d) and (e), www.intersense.com) or Polhemus (www.polhemus.com) are commonly used. These tracking systems often use acoustic, inertial, optical, or electromagnetic tracking technologies or a combination thereof to determine the device position and orientation. If the primary tracking technology is only capable of position tracking, gyroscopic sensors
are often used to determine device orientation. The tracked devices themselves tend to be small and fairly mobile, and are thus attractive solutions for systems where the user is standing or walking. However, user locomotion is often limited by the presence of cables, necessitating a “cable wrangler” to follow users around and ensure that they do not trip over cables, or pull expensive equipment off shelves. Cable-free solutions are now also available, but are more expensive. Another common limitation is the limited physical floor size of most VE systems, i.e., the real space the user can move around in.

Most 6DOF selection and manipulation techniques fall roughly into two broad paradigms: ray-based techniques (and similar techniques like occlusion) and virtual hand metaphors [11, 12, 17, 60]. Each paradigm is discussed in greater detail below.

2.1.1 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. Figure 2-2 depicts an example of a virtual hand [59]. Users can select objects by intersecting their virtual hand with 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” or volume selection.
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). 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 Fitts’ law styled study [28]. This technique allowed objects to be scooped up using a virtual paddle connected with a physical input device.
Boritz and Booth [9, 10] conducted a series of studies on 6DOF input devices for 3D interaction. They initially studied the use of 6DOF input devices for selection tasks [9]. In this study, they compared stereoscopic to monoscopic display, with and without head tracking, as well as different target positions. Their experimental task involved moving 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. They found that 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, interaction effects with the stereoscopic display mode showed that these differences were significantly lessened when users were provided with the additional depth cue of stereo vision. Boritz’s second study [10] also considered the orientation of the target, requiring users to dock a cursor with a target, matching both position and orientation. Again, it was found that differences existed depending on the position moved to, but this was further complicated by interactions with the target orientation.

Zhai et al. [93] 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, as well as stereo to mono graphics. 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 are beneficial in depth perception, but
using both simultaneously provides an even stronger depth cue. The benefits of volumetric selection have also been recognized in 2D interface design, and led to the development of area cursors [25, 37]. These effectively increase the target size, making it easier to select. Early approaches [37] 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 [25].

2.1.2 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 in Section 2.2.

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. Better alternatives have been presented and are discussed below in Section 2.1.3.

Ray techniques have been more widely adopted than virtual hands. Possible reasons for this are discussed below in Section 2.1.4. There is a great deal of interest in these techniques in both 2D [36, 55] and 3D [26, 40, 41, 71, 91] user interface design. In the 2D domain, these techniques are often used either to interact with large displays at a distance [36] or for collaborative systems [55].

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. This is depicted in Figure 2-4 below.
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. [26] 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. There are two reasons for this. First, farther objects take up proportionally less screen space due to perspective, and are thus harder to select. Second, ray-based selection of close-by features effectively increases angular precision when rotating the 6DOF input device. Conversely, remote
objects are harder to select, as subtle device movements/rotations are amplified down the length of the ray [40].

Some work has focused on extending the “traditional” ray-casting technique to compensate for some of the observed problems. For example, Steinicke et al. [71] 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 advantages are questionable.

Another ray technique extension is cone selection [12], 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 [93], which is not a ray-based technique.

Finally, ray casting has not only been used for direct manipulation and object selection, but also for system control. Kunert et al. [41] examined system control in the context of direct interaction with on-screen widgets, and found that ray-casting was well-suited to this, if the widgets were displayed sufficiently large. Subsequent discussion is limited to object selection and manipulation tasks.
2.1.3 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 [13, 59]. In this sense, virtual hands can be implemented more similar to the familiar desktop mouse interface. There, the input device and cursor movements are decoupled, and tracker motion maps to 3D cursor motion only in a relative way, similar to the 2D mouse. An example of this is the Go-Go technique [59]. This virtual hand technique allows 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 [13] 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 [59] 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.

### 2.1.4 Comparing Rays and Virtual Hands

Previous work presented taxonomies of 3D selection/manipulation techniques [11, 60], 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. [11] 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 [11] comparing a number of these selection/manipulation techniques indicated that ray-casting outperformed Go-Go [59]. 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 lateral and longitudinal directions. Previous work has demonstrated that techniques requiring fewer degrees of freedom tend to outperform higher-DOF techniques [88]. These results were again confirmed by Grossman et al. [26] 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. [60] 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 [60] somewhat contradict Bowman’s [11]. 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.

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.

Different evaluation scenarios are currently common practice in virtual reality studies and it is rare to find a study precisely replicated in subsequent work. Consequently, it becomes difficult to accurately quantify the benefits of the specific factors evaluated, leaving researchers and practitioners only with general guidelines such as “haptic feedback improves selection performance” or “ray-casting usually outperforms virtual hands”. Unfortunately, due to this variation between studies, more specific claims are difficult to make.

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 only 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 usually require movement at the elbow, or even the shoulder, necessitating control of 6DOF. This difference in the number of controlled DOFs may account for the measured differences.

To investigate this, a study conducted by Zhai and colleagues [94] 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 muscles, namely 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” outperformed 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 [94].

This conclusion was supported by later work comparing muscle groups in the fingers, wrist and forearm [6]. Using these muscle groups together seems to result in superior performance compared to just using the fingers alone. In particular, the authors of this later study found that holding a stylus between the thumb and forefinger permitted better task performance than the fingers, wrist and forearm movements tested in their experiment. 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 a handheld tracked object and a virtual replica for a docking task [10]. Results of this study 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 [94] also used a 6DOF docking task. The Fingerball device outperformed the hand mounted tracker, likely due to the relative ease with which the ball could be rotated. The hand-mounted tracker required more clutching, resulting in worse task completion times. This clearly indicates that rotation was the dominating factor in the results, and that translation times had a limited impact on task completion time. However, the authors did not break down the analysis by translation and rotation times.

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 [62]. In particular, for large rotations, a non-
isomorphic rotation technique significantly improved rotation speed, without decreasing accuracy [62].

### 2.2 2DOF Input Devices

The above-mentioned studies were based on 3 or 6 DOF input devices. But some research suggests that 2DOF selection can, in fact, outperform 3DOF selection [11, 88]. In general, selection and manipulation techniques based on 2D ray casting behave similarly to the techniques described above. In particular, they allow pixel-precise selection and subsequent manipulation of any 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 [11] (discussed in greater detail earlier) 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 motions significantly outperforming 6DOF techniques, on average. This supports earlier findings reported by Ware and Lowther [88].

Moreover, 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. In spite of this, there is evidence that 2D input devices can outperform 3D devices for certain 3D positioning tasks [7, 56, 76-78]. This is typically achieved through the use of mouse-based ray-casting.

Ray-casting is often used with 6DOF devices as described in the preceding section. However, ray-casting can also be used with 2D 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.
Previous work indicates that 2D interface devices work well for 3D interaction when ray casting is used for selection and manipulation [11, 56, 68, 88]. Ware and Lowther [88] conjecture that this is because situations where the user wishes to interact with totally occluded objects are rare. Note 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 [88]. 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 positioning 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-level manipulation task into a series of low-level 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” [15], the “skitters and jacks” technique [8], the Arcball technique for rotation [64], or the use of mode control keys.

In commercial 3D modeling and CAD packages, the most commonly employed solution is 3D widgets [15, 73]. 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 for an example of
translation and rotation widgets taken from an industry-leading 3D modeling package, Autodesk’s 3D Studio Max. This is usually complemented by different simultaneous orthogonal views of the same scene from different sides. Bier’s skitters and jacks technique [8] 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.

![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.](image)

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.

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 [68]. 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 [56]. 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.

![Figure 2-8: Mouse motion to 3D movement mapping in SESAME.](image)

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 [56]. Also, novices learn the technique very quickly, perhaps due in part to its similarity to standard 2D direct manipulation interfaces.

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 [77]. 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.

![Figure 2-9: Wand motion constrained to 2DOF operation via SESAME movement technique.](image)

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 [77]. This was again verified in later work [78]. In fact, the last study revealed that the chosen orientation plane did not significantly affect task completion time.

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 [64]...
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 [65]. There are several variations on the virtual trackball approach [31]. One of these, the two-axis valuator has been empirically demonstrated to outperform the other approaches for speed [5]. 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) [65].

2.3 Summary

Extensive research has been conducted on the development of efficient 3D selection and manipulation techniques. The majority of these techniques fall roughly into one of two paradigms, namely virtual hand/depth cursors as well as the ray-based techniques. Ray-based techniques use either 3 or 6DOF input devices such as trackers, or 2DOF devices
such as the mouse. Previous work has demonstrated that 2D ray-casting selection tends to outperform similar 3D techniques in selection and manipulation tasks, likely due to the reduced complexity of controlling only 2DOFs. The evaluation of these techniques is complicated by various issues such as the presence of stereo display, or latency differences between devices. The following chapters discuss technical and perceptual issues related to the evaluation of 3D selection/manipulation interfaces.
Chapter 3
Depth Cues, Stereo Graphics, and Head Tracking

This section discusses the so-called depth cues commonly provided in virtual reality and 3D graphics systems. Such depth cues enable us to perceive depth through a variety of means. This section is not intended to serve as a comprehensive overview of all depth cues. Instead, I focus exclusively on those that are commonly available in virtual reality systems, especially 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 perceptually 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 comprehensive overview of depth cues, the reader is encouraged to read Chapter 8 of Colin Ware’s *Information Visualization* [82].

3.1 Depth Cues – Stereopsis, Convergence, Accommodation

A number of depth cues allow us to perceive depth [82]. Many of these 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) [90]. These cues are adequately simulated in computer graphics. Perspective is achieved using a frustum viewing volume and perspective transformations. Near objects take up proportionally more space in the viewing frustum than far ones, and
after projection onto the near clipping plane and normalization, they are eventually rasterized to a proportionally larger number of pixels, and thus appear bigger than far objects [29].

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). Initially, all entries of the z-buffer are set to some maximum depth value. As each fragment is rendered, its newly computed depth value is compared against that already in the z-buffer for a given pixel location. If it is smaller (closer) than the value already in the z-buffer the pixel colour and depth values are updated overwriting the old pixel with the new one. Conversely, if its depth value is larger, these updates do not occur. In this situation, the new object is farther and thus occluded by the previously rendered pixel [29].

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 [29].

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 3-1 [80].
Accommodation is sometimes simulated in computer graphics and games (the so-called “depth of field” effect), but without knowledge of where the eyes are actually focused, this effect is unrealistic. If the eyes are tracked, and their focus could be measured, then displays could use this as an accurate depth cue [82]. However, this is extremely uncommon in modern displays, mostly due to the lack of eye tracking systems that work in general environments without attaching technology to the head.

Of particular interest though are the binocular depth cues, those that require two eyes to be perceived. These include stereopsis and convergence [82]. 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. The stereopsis depth cue allows us to utilize this fact to detect additional depth information. Objects that are farther away project less differently than
near ones. Conversely, near objects project to more similar positions on the retina. 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 3-2 [80] depicts this.

![Convergence diagram]

*Figure 3-2: Convergence and convergence angles. Image courtesy SAP Design Guild [80].*

Note that the stereo cues mentioned above are most effective at close range. This can be observed by viewing a near point (e.g., in arm’s reach) and closing each eye in succession. The point will appear to move dramatically when changing eyes. Contrast this with a distant point, which will appear to move much less. Similarly, the eyes converge to extremes for very close objects, and are nearly parallel for distant objects. Consequently, stereo depth cues may be beneficial in close range VR tasks such as selecting and manipulating objects [9-12, 77]. However, this benefit may not extend to other 3D tasks, especially navigation tasks such as travel and way-finding.
3.1.1 Depth Cue Conflicts in Stereo Displays

One issue in the use of stereo technology is that it introduces cue conflicts between the convergence and accommodation cues. In particular, our eyes will converge at the true perceived depth of presented stimuli, but will accommodate to a single plane, typically the display surface. This is depicted in Figure 3-3, reproduced from Shibita et al. [63].

![Figure 3-3: The convergence and accommodation cue conflict problem. Image courtesy Shibita et al. [63]](image)

When viewing objects in the real world, our eyes will converge and accommodate to the same point [82], and thus will always provide consistent depth information. In stereo graphics system, the technology yields conflicting depth information from these two cues. This can result in eyestrain, nausea, and headaches [32, 33], especially for short
distances such as in small-scale VR systems or when the user is manipulating objects in arms reach. This may be less pronounced in 3D film because the relative distance to the screen is much larger, and the eyes need not converge as much.

The convergence/accommodation cue conflict is known to have a negative impact on depth judgement tasks [32]. 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 [92]. One option to avoid the problem altogether is to use specialized optics to adjust the accommodation depth to match the convergence depth [63]. 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 [18], or by spinning a display surface quickly which shows a the image corresponding to a slightly different viewpoint at each step [24, 26]. 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 [27].
3.2 Evaluating Stereo Displays

Stereoscopic 3D rendering is commonly used in VR systems [12]. This involves displaying a slightly different image for each eye, then filtering each image to the appropriate eye, often using special glasses. 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 on the horizon and the first stereo-capable game console, the Nintendo 3DS, launched on March 27, 2011 in North America (http://www.nintendo.com/3ds). The 3DS uses a parallax barrier to provide an autostereo display – a display on which stereo imagery can be observed 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 perceptually 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, and may enhance presence – the 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 [66].
3.2.1 Graph Tracing

One area that has demonstrated some clear benefits of stereo viewing is path/graph tracing [4, 27, 86, 89]. 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. [4] were the first to conduct a stereo investigation in a fish tank VR system [84]. They found that stereo only slightly improved graph tracing speed, but significantly improved error rates in this kind of task [4], particularly when compared to a static 2D image of the graph. These results are echoed by Ware and Franck [86] and later again by Ware and Mitchell [89]. 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 [27] 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 due to the aforementioned loss of occlusion cues on the volumetric display.

3.2.2 Selection and Manipulation

While somewhat different from 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 [92], 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” and carried out without the aid of perceptual feedback, but 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. [93] evaluated object selection with a comparison of 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. Error magnitude decreased by a factor of about 2.5 globally.

Boritz and Booth [9, 10] also evaluated the benefits of stereo in 3D point selection tasks [9] and a 3D docking task [10]. Their study compared stereos to mono display, with and without head tracking. They found that the presence of stereo display significantly improved task completion time, especially for depth motions (i.e., movements into or out of the screen). Similarly, error magnitude was significantly reduced for depth motions in
the presence of stereo display. Arsenault and Ware [3] conducted a study based on Fitts’ law 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 [77] 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.

3.3 Head Tracking

Many VR systems also track the user’s head [3, 9, 10, 22, 42, 43, 52, 79]. 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 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., stereo 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 [9, 10, 52, 77]. The aforementioned point selection and docking studies conducted by Boritz and Booth [9, 10] did not reveal any significant effects due to the presence of head tracking. The authors reason that their tasks required only minimal head movement after the initial discovery of target locations. These results are confirmed in a later manipulation study as well [77]. Aresenault and Ware [3] report a significant improvement in tapping speed of about 11% due to the presence of head tracking – to contrast, stereo improved speed by three times as much. Ware and Mitchell [89] report that stereo alone was the fastest condition in their graph tracing experiment, 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. 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 [84]. In contrast, tasks investigated in other studies [9, 10, 77] were performed in simpler scenes, and required relatively few depth judgements by the users. In fact, the only depth judgements required were typically only necessary to
ensure that the object being manipulated was within the extents of the target zone, i.e. along the depth axis.

3.4 Summary

Stereo 3D graphics and head tracking are often used in virtual reality systems and 3D user interfaces. Both of these afford additional cues that enhance depth perception. This is important, since unlike 2D interfaces, objects of interest may be positioned along the depth axis. This requires an improved understanding of the spatial relationship between objects, and the ability to accurately select the object with a technique that supports depth motion (either a ray or virtual hand technique). Both of these display techniques seem to have some benefit, depending on the nature of the task at hand. In perceptual tasks such as graph tracing, they have clear benefits. For selection and manipulation tasks, there are conflicting reports.

There is, however, some evidence that well designed or limited DOF techniques can reduce or eliminate the need for these additional depth cues. Furthermore, without special optical correction, all existing stereo displays introduce cue conflicts between convergence and accommodation (focus). These conflicts hinder depth perception, although the effects of this on 3D selection/manipulation are not yet known.
Chapter 4
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 noise (jitter) and latency (temporal lag). Similarly, most 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. However, co-location is not always used and may not even benefit manipulation tasks. These issues complicate the evaluation of 3D user interface because they vary wildly between input/display devices and thus confound experimental results. Consequently, each of these issues is discussed below.

A number of 3D tracking technologies exist today. Typically, such a tracking system is required for a 3D input device. Foxlin [22] provides a thorough overview of the available types of tracking technologies. Although Foxlin argues that one should choose a specific tracking technology based on needs, most tracking technologies have several shortcomings in common that affect user performance. Specifically, most tend to suffer from increased latency, spatial jitter, and potentially also temporal jitter.
4.1 Latency

End-to-end latency, or lag, 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. Various methods exist to measure the latency in a system [50, 70]. 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 confound 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 in the community, it demonstrates how seriously many researchers take latency, which is still present in devices despite advances in technology.

It is well-known that latency adversely affects human performance in both 2D pointing tasks [46, 58] and 3D tasks [19, 76, 85]. 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 [1]. 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 [46] 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, even with latency of as low as 75 ms. Ware and Balakrisnan [85] 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 [85] 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.

More recent work [76] included MacKenzie and Ware’s 225 ms latency condition. The authors report that a 40 ms latency difference accounted for about a 15% performance drop in terms of pointing throughput. This corresponds to the measured latency difference between a mouse and a 3D tracking system used in the study. The 225 ms latency condition accounted for a 50% performance drop.
4.2 Jitter

Spatial jitter is caused by a combination of hand tremor and noise in the device signal. Noise can be observed by immobilizing a device while observing the reported positions; even 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 [76] 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 a 3D tracking system used in the study. More extreme amounts of jitter do impact user performance though, especially for small targets [58]. 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 [58].

Temporal jitter, or latency jitter, is the change in latency with respect to time. Ellis et al. [20] report that people can detect very small fluctuations in lag as low as 16 ms. Hence when examining system lag, one should also ensure that latency jitter is minimized, or at least measured.
4.3 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 jitter 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 tabletop surface or similar is present. This problem is exacerbated in virtual environments using head-mounted displays, as the user is also unable to see the device itself [43].

The positive properties of a support surface have been recognized in the virtual reality community. Mine et al. [51] 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. Proprioception is the sense of the position and orientation of one’s body and limbs. 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. [51] 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 [15], 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 [51]. 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 [74], Poupyrev’s virtual notepad [61], and Lindeman et al.’s HARP system [42, 43]. 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 [79], so this design decision may yield effective interfaces to virtual environments. Another similar idea is to utilize a secondary interaction surface such as a tablet PC [14]. Not only does this provide a physical support surface, but also a familiar interface displayed on a secondary touch-sensitive screen.

Another approach [39], reminiscent of Mine’s work [51], 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 however that most of these approaches still involve a very strict separation of the 2D and 3D interface components, which may increase cognitive overhead for the user.

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 [81]. 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. Note that this study co-located the input and motor spaces. To contrast, when using a mouse, or bat (“flying mouse”) [87], the control space is disjoint from the display space. Users do not operate directly on virtual
objects (e.g., with their hands), but instead manipulate the device to indirectly control objects in the environment.

4.4 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 [8, 15, 56], and can even outperform 3D devices for conceptually similar tasks [77, 78].

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 [87] 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). Consequently, 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. [51] 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 [59], 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. The study conducted by Arsenault and Ware [2] was intended to determine if correctly registering the virtual object position relative to the real eye position improved performance in a tapping task. Their results indicate that this did improve performance slightly, as did haptic feedback. Thus, they argue for correct registration of the hand in the virtual environment.
Sprague et al. [69] 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 [30]. However, observers quickly adapt to this distorted visual input over repeated trials 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 [30].

Groen and Wekhoven [23] 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 [83] 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.

In summary, while there is some evidence that input/display co-location can improve performance, the benefits may be somewhat minimal given that people seem readily able to adapt to mismatches.

4.5 Summary

A number of factors complicate the evaluation of 3D selection and manipulation tasks. Two system specific issues are latency and jitter. Failure to at least measure these can clearly confound the results of studies comparing input devices with varying levels of
latency and jitter. The presence of tactile feedback and visual/motor co-location are system design considerations that may also influence the performance of selection and manipulation techniques. Visual/motor co-location does not appear to dramatically improve performance in selection/manipulation tasks, as humans can rapidly adapt to non co-located interfaces.

Results of previous studies clearly demonstrate that tactile (or haptic) feedback is beneficial in these kinds of tasks. This type of feedback can help account for imperfect stereo perception commonly employed in 3D user interfaces. However, haptic feedback is comparatively difficult to simulate, and requires expensive, uncommon input devices. Most of these devices afford haptic feedback only on a single point, such as a stylus. One simple alternative is to register support surfaces or props with the position of objects in the environment. This works well for interface devices (e.g., handheld menus or windows) but not for objects that can move in the environment, or whose shape does not conform to a flat surface.
Chapter 5
Experimental Evaluation

As discussed earlier, one drawback in 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. In this section, methodologies frequently used in the evaluation of 2D computing pointing devices are examined. 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. The adoption of these tools in 3D selection/manipulation interface evaluation may prove invaluable, as this would allow fine-grained analysis of the simple motions that make up the more complex 3D tasks.

5.1 Fitts’ Law

Fitts’ law [21] is a model for rapid aimed movements:

\[ MT = a + b \cdot \log_2(A/W + 1) \]  

(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:

\[ MT = a + b \cdot ID \]  

(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 [34]. 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).

5.2 ISO 9241-9

ISO 9241-9 [34] employs a standardized pointing task based on Fitts’ law, see Figure 5-1. The standard uses throughput as a primary characteristic of pointing devices [6]. Throughput ($TP$) is defined in bits per second as:

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

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 [45]. $SD_x$ 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 [35]. $W_e$ corrects the miss rate to 4%, enabling comparison between studies with differing error rates [45]. $A_e$ 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 [47]. 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. Effective measures may also make throughput less sensitive to device characteristics (e.g., device noise). This is desirable in cross-device comparisons.
Figure 5-1: 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.

5.2.1 Effective Width and Effective Distance

During the evaluation, participants are asked to click on targets of various sizes, spaced at various distances. 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. Thus, it is beneficial to take this increase or decrease of accuracy into account. As an illustration, Figure 5-2: depicts the distribution of hits when a task is performed repeatedly.
Figure 5-2: Distribution of clicks on a circular target.

It is a convention to use a sub-range of the hit data, corresponding to about 96%, as the effective width of the target [45]. This range corresponds to approximately 4.133 standard deviations of the observed coordinates of hits, relative to the intended target center. This corresponds better to the task that the user actually performed, rather than the task the user was asked to perform.

In general, a projection of the actual movement vector onto the intended vector is computed 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 5-3 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 5.3: Illustration of effective width and effective distance. Note that these are averaged over multiple movement vectors.

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 [45]. 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 [34].

5.3 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 [45]. 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 always improve the correlation [57]. Thus, it is not certain if extra factors improve a model’s predictive capabilities in an appropriate way.

Murata and Iwase [53] 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. [24] 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 [40]. The model presented by Kopper et al. [40] 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.

5.4 Motion Analysis

While the ISO standard provides a reliable methodology by which to rank input devices/techniques in absolute pointing performance, several authors [44, 48, 54, 67] point out that it does little to explain why devices perform how they do. 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 [44, 48, 54]. 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 [44], Nieuwenhuizen [54] and their colleagues is based on a two-component model of movement proposed in 1899 by Woodworth [92]. 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. [44] 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 [49]. 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. [54] 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 [44]. 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. [67] 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 [48] 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 [48]. 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 [38]. 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 [75]. 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.
5.5 Summary

Fitts’ law and ISO 9241-9 provide what may be an invaluable tool in evaluating the fundamental motions that make up higher level selection and manipulation tasks. This may enable the direct comparison of 2D and 3D input devices, e.g., using ray-casting selection. However, despite widespread adoption in 2D user interface evaluation, these are not yet widely used in the evaluation of 3D tasks. One possible reason for this is the fundamental differences between 2D and 3D pointing tasks. This may necessitate extensions to the formulation of the index of difficulty to account for the potentially harder 3D tasks.
Chapter 6
Conclusion

Overall, the objective of this report was to document several issues related to the design and evaluation of 3D selection and manipulation interfaces. Although these issues are all quite different, all must be considered in order to account for differences between 2D and 3D selection/manipulation interfaces.

The first such issue is the difference between input techniques themselves. In particular, most techniques can be classified as either ray-based (remote pointing) techniques, or virtual hands (3D cursor) techniques. These two paradigms are very different. Ray techniques allow remote pointing at targets for both selection and subsequent manipulation. Virtual hands require intersection of the hand with objects in order to select them. When using 6DOF input devices for these techniques, a virtual hand technique requires accurate intersection in three dimensions to select objects. In contrast, ray-based techniques need as few as 2DOFs to accurately select an object. These differences make the direct comparison of these techniques difficult.

Differences in input devices also increase the difficulty of comparing techniques. There are obvious differences between devices such as the number of simultaneous degrees of freedom the device can control. The mouse, for example can control 2DOF at once, whereas most 3D trackers afford 6DOF of control. However, more subtle differences between devices also exist. Latency and jitter are two related factors. Other
device-dependent factors include the presence of tactile feedback and/or physical support surfaces. Many 3D input devices do not require a supporting surface like a mouse. While this is less constraining (e.g., the user can walk away from the support surface) the absence of said supporting surface may actually impair the usability of these devices.

Stereo display and head tracking is often used in 3D user interfaces to enhance the perceptual richness of a virtual environment. These technologies are widely believed to also enhance the usability of the environment, but empirical results vary. This again relates to the issue of experimental design, and the effects of these technologies may be highly task-dependent. Consequently, an “impartial” objective measurement technique would help quantify the absolute differences in performance due to stereo and head-tracking.

6.1 Open Problems

6.1.1 Selection/Manipulation Techniques

Although ray-based techniques tend to outperform virtual hand techniques, there is no generally accepted methodology to evaluate this issue. Given this lack of a standardized methodology, the comparability of previous work is somewhat limited. This is because most studies in this area tend to use complex selection and/or manipulation tasks. An alternative approach is to investigate the fundamental pointing motions that make up more complex manipulation tasks. This is a common practice in 2D user interface evaluation [45]. Chapter 5 discussed this in greater detail.
Moreover, very little work has gone into the direct comparison of 2D and 3D interfaces that perform conceptually identical tasks. For example, the direct manipulation of desktop icons in a 2D interface is an easy day-to-day task. The conceptually similar task of moving a 3D object using a 3-6DOF virtual hand technique in a 3D environment is far more challenging. Results of the research discussed above suggest that intelligent software techniques can overcome the limited degrees of freedom for devices like the mouse and, with appropriate mappings, 2D devices outperform 3D devices. However, previous studies [77, 78] on this matter use somewhat crude experimental designs. In particular, the tasks used are highly subject to effects of user strategy, i.e., two participants doing the same task may do it in a different order, potentially affecting their results. This again relates to the issue of moving toward a standardized methodology. Consequently, the development of an “optimal” experimental methodology for the evaluation of 3D selection/manipulation interfaces is a major open problem for the area. This is challenging due to the wide variety of input devices and techniques available for selecting and manipulating objects in 3D.

6.1.2 Stereo and Head-Tracking

The benefits of stereo 3D graphics and head-tracking are somewhat debateable. Previous work has suggested that the benefits may be task dependent, but it is somewhat unclear which tasks, exactly, benefit from stereo. Since stereo and head-motion parallax cues are strongest close to the viewer, selection and manipulation tasks in arm’s reach should be improved. However, this has not been consistently demonstrated in previous studies. One
potential avenue for future work is to further investigate this in the context of simple fundamental pointing motions. Moreover, it is also unclear if stereo and head-tracking benefit only 3 or 6DOF techniques, or if they also affect essentially 2DOF selection/manipulation techniques (e.g., mouse pointing with ray-casting). Consequently, the effects of stereo viewing on 3D selection using a mouse warrants further investigation.

6.1.3 Device Characteristics

There is an inherent tradeoff between latency and jitter. Jitter can be smoothed, but smoothing delays device samples and thus introduces more latency. Conversely, movement prediction can effectively reduce latency, but incorrect predictions must be corrected and introduce more jitter. A better understanding of this latency-jitter trade-off may help optimize input techniques for common 3D selection and manipulation tasks.

6.1.4 Standard Evaluation

One possible evaluation method to address the above issues is the standardized methodology prescribed by ISO 9241-9, and based on Fitts’ law. Although some researchers in the VR community are aware of these tools, they are not as widely used as in the 2D user interface domain. Consequently, a clear avenue for future work is the investigation of applying this methodology to the evaluation of 3D selection and manipulation techniques. The techniques described above all conform to the Fitts’ paradigm of rapid aimed movements. There is effectively a direct correspondence
between selection tasks and the pointing tasks modeled by Fitts’ law. Manipulation tasks are slightly more complex, but most direct manipulation should also conform to this, as in the 2D domain. This may also help elicit differences between display technologies (i.e., stereo and head-tracking) in a fair, objective way.
References


86. Ware, C. and Franck, G., Evaluating stereo and motion cues for visualizing information nets in three dimensions, *ACM Trans. Graph.*, 15, 1996, 121-140.


