Design and Evaluation of Body Proximal 3D Interaction Techniques for Large Displays

Submitted to
Graduate School of Information Science and Technology
Osaka University

July 2015

Nikolaos Katzakis

Abstract

This thesis presents the design and evaluation of a set of novel techniques for manipulating and annotating 3D graphics. These techniques are analysed in terms of human performance and efficiency. A series of four interaction techniques are presented and evaluated. All the techniques consider the manipulation of a 3D model located out of arms reach for the purposes of presentations or education but also have application in virtual reality, augmented reality etc.

The first set of techniques, Mesh-Grab and Arcball-3D leverage proprioception and general human performance when operating a pointing device (wand) from the hip area. Evaluation results point to this technique performing similarly in terms of task completion time with Scaled HOMER, a state of the art 3D manipulation technique, while at the same time being more accurate and requiring smaller motions.

Unistroke is an extension to ray-based manipulation. Unistroke allows drawing on a mesh with implicit switch to rotation for making smooth long strokes on a 3D object. Evaluation of Unistroke shows that the technique is slower and requires more strokes than a state-of-the-art 3D mesh drawing technique but is more accurate in a tracing task.

The last set of techniques free/pivot Plane-Casting explore a low cost and low fatigue off-screen means for 3D cursor translation. Evaluation of these techniques shows superior performance to a baseline wand technique as well as the users' preference towards free Plane-Casting which is used as the basis for further work. INSPECT presents a set of off-screen touch rotation techniques that allow for smooth transitions between controlling different degrees of freedom. Evaluation of INSPECT versus a state-of-the-art manipulation technique found better movement times and on-par rotation times in a docking task.

The conclusion presents a summary of our findings and directions for future improvement.

Acknowledgements

Of the many people who deserve thanks, some are particularly prominent. First and foremost my family for their unending support during the years of my graduate studies. I could not have done this without them. I also owe a great debt of gratitude to Mr. Kusuo Hatada and Mr. Masanori Hatada for their friendship and support throughout my years in Japan.

I would like to thank my actual supervisor, professor Kiyoshi Kiyokawa, for his exemplary character and for his invaluable help whenever I needed it. Professor Haruo Takemura for his financial support and for standing by me when I got in all sorts of trouble. Also to Professor Tomohiro Mashita for providing hardware resources and for his overall support and companionship during my study years. Professor Masahiro Hori of Kansai University provided me with mentorship during the start of my graduate school years in Japan. He taught me what it means to be a scientist.

Special thanks go to my collaborators, particularly Kazuteru Seki who is my co-author for Chapter 3 and Rob Teather who helped me a great deal on Chapter 7. Martin Hachet for hosting me in his lab and for his help and collaboration, as well as the inspiration from his work during my early academic days. Professor Doug Bowman deserves special mention for his collaboration and his invaluable advice about my work that helped me grow as a researcher on more than one occasion.

A big thanks to Professor Yoshifumi Kitamura for his advice and support during my entrance to graduate school. Last but not least, a big thanks to Professor Takeo Igarashi. His work has been a source of great inspiration.

To my loving family for their unending support.

- Nikos

Publications

This thesis is the product of the collaborative research effort of several researchers and is based on work that has been published in the following peer-reviewed scientific publications :

Journals

1. Nicholas Katzakis, Robert Teather, Kiyoshi Kiyokawa, and Haruo Takemura, "INSPECT: Extending Plane-Casting for 6-DOF Control" Human-centric Computing and Information Sciences, Springer, Accepted for publication June 2015.

International Conferences

1. Nicholas Katzakis, Robert J. Teather, Kiyoshi Kiyokawa, and Haruo Takemura, "INSPECT: Extending Plane-Casting for 6-DOF Control" Proceedings of the IEEE Symposium on 3D User Interfaces (3DUI), pp 165-166, Mar., 2015.

2. Nicholas Katzakis, Masahiro Hori, Kiyoshi Kiyokawa, and Haruo Takemura, "Plane-Casting: 3D Cursor Control with a Smartphone", Proceedings of the 3rd Dimension of CHI (3DCHI): Touching and Designing 3D User Interfaces (3DCHI 12), *Workshop* at ACM Annual Conference on Human Factors in Computing Systems, pp. 13-21, May, 2012

3. Nicholas Katzakis, Kiyoshi Kiyokawa, and Haruo Takemura, "Plane-Casting: 3D Cursor Control with a Smartphone" Proceedings of the 11th Asia Pacific Conference on Computer Human Interaction (APCHI), pp. 199-200, Sep., 2013.

4. Nicholas Katzakis, Kazuteru Seki, Kiyoshi Kiyokawa, and Haruo Takemura, “Mesh-Grab and Arcball-3D: Ray-based 6-DOF Object Manipulation” Proceedings of the 11th ACM Asia-Pacific Conference on Computer Human Interaction (APCHI), pp. 129-136, Sep., 2013.

5. Nicholas Katzakis and Masahiro Hori, “Mobile Phones as 3-DOF Controllers: A Comparative Study” Proceedings of the 8th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC), pp. 345-349, Dec., 2009.

6. Nicholas Katzakis and Masahiro Hori, “Mobile Devices as Multi-DOF Controllers” Proceedings of the IEEE Symposium on 3D User Interfaces (3DUI), Waltham, MA. pp. 139-140, Mar., 2010.

Contents

Chapter 1

Introduction

1.1 Interactive 3D Computer Graphics

Computer graphics technology allows for the realistic depiction of objects, with perspective projections, lighting and effects which make it possible for us to understand their surface properties, their functions and their internal structure. This is important in a number of application domains. From city planning [52] and CAD in immersive virtual reality [71], interior design [66] reality and virtual prototyping [28], to manipulating a multi-dimensional dataset for scientific data exploration [41], educational applications [67], medical training [62] and even sandtray therapy [26].

In this thesis we are mostly concerned with the following two:

- In engineering and design: Computer aided design allows for the prototyping of objects at extremely low cost compared to actual prototypes. Computer graphics also allow for the inspection of objects with significantly reduced effort. The latter is especially true for massive solid objects such as engines, buildings, vehicles/vessels etc.
- In education: Computer graphics allow for the depiction of objects isolated from their natural environment. It makes it possible to view a realistic beating heart pumping blood, and to understand its structure and functions something which would have otherwise been impossible.

Great progress has been made in the real-time realistic depiction of 3D ob-

Figure 1.1: The state of the art in real-time interactive graphics. The Brigade real-time path tracing engine [58].

jects [58], with significant increases in display resolution and realism (Figure 1.1). However, manipulation of 3D objects remains a challenge. The challenge lies in the fact that in the real world, humans use their limbs to directly apply forces to objects and change their position or rotation. Save for a few exceptions [14, 59] which are not yet ready for production work, with most types of displays that is not feasible.

Thus typically, for the control of computer graphics with a certain granularity we resort to electromechanical devices that transform forces from the limbs to values that are interpreted by the system and are used to change one degree of freedom (DOF). The most common of these devices is the mouse [13] invented by Douglas Engelbart in 1967. Despite its age, the mouse is still the most common input device for the control of computer graphics. The mouse controls two degrees of freedom and usually via UI widgets these 2-DOF are being used to control 1-DOF (Figure 1.3).

Integral control of 3D computer graphics, however, requires control of 6-DOF., i.e. if we were to manipulate objects naturally, like we do in the real world, we would need to manipulate six integral degrees of freedom.

Just like in the real world, we manipulate objects with manipulating as many DOF as possible to save time, the same is desirable in manipulations in computer graphics. Jacob et al. [34] have shown that that some DOF are better manipulated integral whereas others are better manipulated separately. Posi-

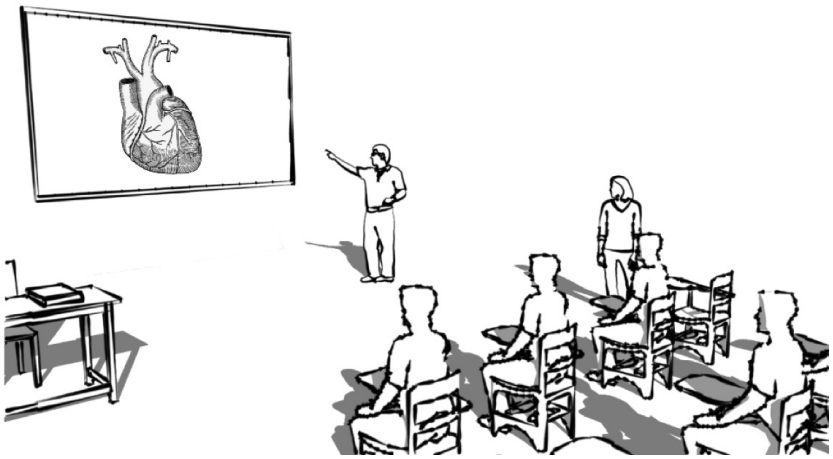


Figure 1.2: A professor presenting a 3D model of a heart in an anatomy class.

tion and rotation are attributes that are considered perceptually integral and as such we want to manipulate them together, as opposed to position and color. So *integrality* in 3D manipulation might be desirable depending on the application.

Although there are a number of devices that allow for integral control of 6-DOF for personal computers they suffer from a number of issues which make them difficult to use (more details in Chapter 2). The problem is particularly evident away from the desktop where users do not have access to such input devices. Examples of situations where there is a need to interact with computer graphics in 3D from a distance with a large display include the following:

- Education: A professor is demonstrating human anatomy by displaying 3D graphics on a large projector screen. He uses his device to rotate the model and answer questions from the students. The professor leave the podium and approach the students while still being able to interact with the model, thus making the class more engaging (Figure 1.2).

- Engineering: An engineer is showing a 3D model of her latest design to her team-mates. She rotates and translates the model, defines slicing planes to inspect the interior and discuss the design with other participants (Figure 1.4). An architect is showing a proposed building to a group of clients (Figure 1.5).

- Entertainment: A group of children are playing a game in a museum while at the same time learning about physics by interacting with objects in a sandbox-like 3D environment on a large screen.

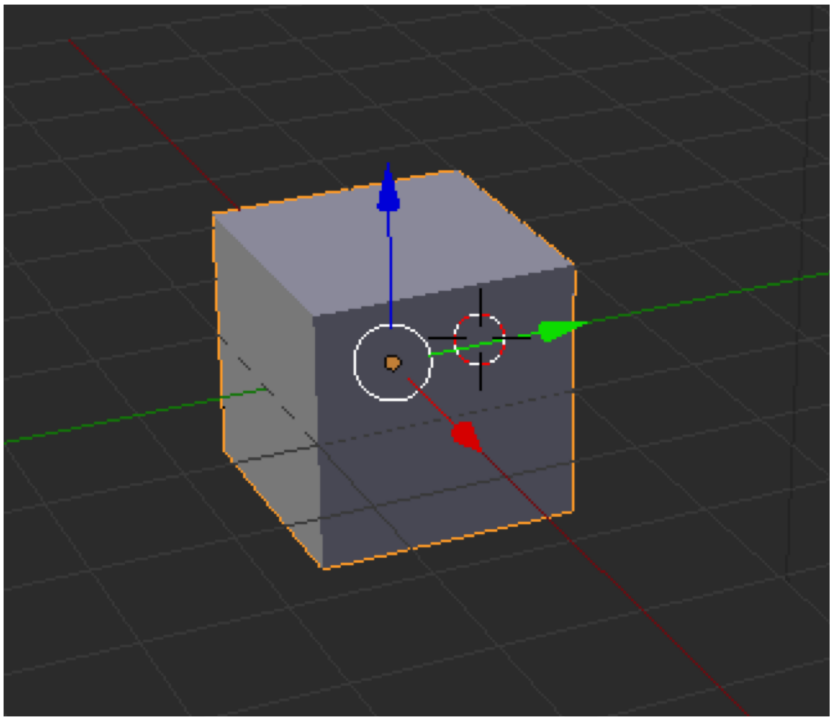


Figure 1.3: Typical usage in a CAD application (Blender [6]). The mouse clicks on the RGB coloured arrows that represent each axis to control 1-DOF with both DOF.

The design goals of an interface to be used in the aforementioned situations include the following:

Eyes-Free: In a presentation, the gaze of the presenter guides the attention of the audience. When the presenter looks at the audience, the audience know they must focus on his face and his speech. It is therefore important that the presenter is free to look at the display and the audience while interacting and not be forced to look at the device.

Off-Screen: Users would need to maintain a distance from the display so as to not obstruct the view for others.

Without a desk surface or cables: Users should be able to move around, approach the display with the controller in their hand (to show an area or point to a feature with their hand).

Without complicated instrumentation or expensive hardware: Having few hardware requirements lowers the barrier for entry, allowing classrooms and meeting rooms equipped with a projector to make more out of their existing

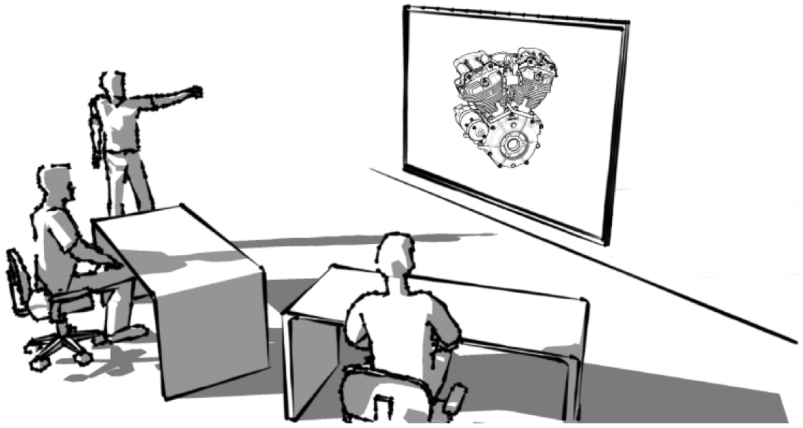


Figure 1.4: An engineer is demonstrating a new engine design to team members.

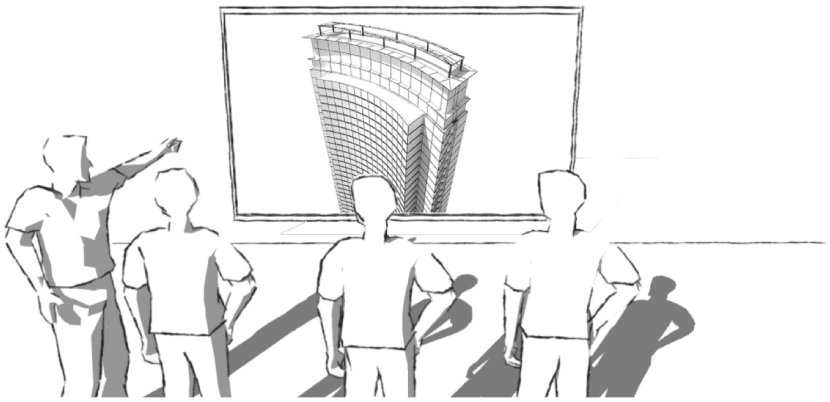


Figure 1.5: An architect is showing a proposed building to a group of clients.

setup.

No big arm/hand gestures: An interface that is to be used on a daily basis and/or for many hours has to avoid large hand gestures which are bound to induce fatigue [65] and, in rare cases, even cause physical injuries to bystanders.

Simplicity: Unlike technology enthusiasts, domain experts or educators often do not have the patience or motivation to learn a new, complicated interface.

Accuracy: If the 3D model is detailed, the interface should allow the presenter to bring it closer and make fine adjustments to position and rotation.
Among the aforementioned design goals, this thesis presents a set of interaction techniques with the following three in mind:

Proximal to the torso - Small Motions: Avoid fatigue and allow for collaboration.

Simple Hardware Setup: Lower Cost, Easier Setup.

Integrality: Ability to manipulate many degrees of freedom simultaneously, having the potential for becoming proficient.

The first set of presented techniques, Mesh-Grab and Arcball-3D, employ a wand and an emitted ray to manipulate a 3D object. The second set of techniques Plane-Casting and INSPECT attempt to further simplify the setup while at the same time exploring off-screen touch input. Finally, we present a technique for annotating 3D objects that explores the boundaries between the indirect and direct interaction design space.

This thesis does not consider selection of a 3D object, it assumes manipulation of an already selected object and selection is beyond the scope of this work.

1.2 Organisation of the thesis

In addition to this introduction, a related work section and the conclusion chapter, the main body of the thesis is organised in five chapters.

1.2.1 Mesh-Grab and Arcball 3D

Chapters 3 and 4 present the design and evaluation of Mesh-Grab and Arcball 3D. These are 6-DOF manipulation techniques designed with low-fatigue in mind. Mesh-Grab and Arcball 3D can be performed without needing to extend the arm and thus have a much smaller spatial footprint than their state-of-the-art counterpart. We discuss the evaluation of the techniques and the implications of the results.

1.2.2 Unistroke

Chapter 5 presents Unistroke, a technique for drawing long strokes on 3D models. The technique was conceived for annotating a 3D mesh during a presentation on a large display that only provides a single input or on a handheld tablet. Unistroke is to be performed either with a wand or with a tablet pc held

with the off-hand while the user annotates with his dominant arm. In such a situation the presenter would not have access to a large number or modes as one would on a desktop CAD application. It is therefore important to allow for intelligent implicit mode changes that allow the user to better annotate meshes in this limited context. We present the features of unistroke and present results from a pilot-study.

1.2.3 Plane-Casting and INSPECT

Chapters 6 and 7 present Plane-Casting and INSPECT, a 6-DOF extension of Plane-Casting. To address the tracking requirements of wand techniques and the line-of-sight limitations we implemented this technique using a smartphone which has neither of these requirements. Plane-Casting is an indirect-touch 3-DOF translation method that is suitable for smartphones. INSPECT allows for full 6-DOF object manipulation using a series of explicit mode changes. Results from a pilot study show that with Plane-Casting/INSPECT users did not access all the available modes but that the techniques perform slightly better than a baseline wand technique and were preferred overall by the users.

Chapter 2

Related Work

3D Manipulation has been studied in a number of usage contexts. Listed in order of broad chronological appearance: Desktop computing, Virtual Reality using a head mounted display, Immersive Large Displays/Cave systems and finally Tabletop/Tablet computing where touch is utilised. We discuss related work with the state-of-the-art in each of these contexts and give an overview of the remaining problems for each one:

2.1 Desktop Computing

For desktop computing 3D manipulations are typically performed with the mouse. For translation, since the mouse can only control 2-DOF, UI widgets are used for controlling the Z axis and for constraining motion to a certain axis [5]. Keyboard shortcuts are also commonly used (in software like Blender 3D [6]) for switching modes or constraining axes. For rotation, in addition to UI widgets, mapping the 2D motion of the mouse to 3D rotation is used extensively [63]. The major drawback of widget-based manipulations is the lack of combined rotation and translation. To address this, some new desktop devices have been proposed [32, 17, 1]. These devices offer integral 6-DOF manipulations. One of these devices, the globefish [17] is shown in figure 2.1.

A drawback of these devices is the inability to "undo" a motion in the exact same path it was performed, and the lack of an easy way to switch modes (to lock to certain axes etc.) while the long homing time [31] makes it tedious to switch from device to keyboard/mouse. Accessing different modes is a point

Figure 2.1: The Globefish [17]. An integral 6-DOF desktop controller.

that will be discussed further in this thesis. The design of these devices is also prone to accidental input when the hand accidentally brushes up against the device or when the desk surface is shaken.

2.2 Immersive Virtual Reality

In the Immersive VR domain typically a wand is used for 3D manipulation with Scaled HOMER [73] being the latest in a long series of techniques [66, 53, 7, 56, 55]. Scaled HOMER is an extension to the classic hand-centerer manipulation, HOMER technique [7].

In Scaled HOMER, a variable gain function is applied to the input depending of the distance of the object to the user. Wilkes et al. reported an improvement over the standard HOMER technique in a docking task.

A ray extended from the wand is used for object selection in most immersive VR techniques. A common problem, however, is that they require accurate 6-DOF tracking of the wand. As such they need complicated instrumentation to set up with either magnetic trackers (Polhemus [54]) or optical tracking (Optitrack [50]). Magnetic trackers are particularly susceptible to interference from the environment while optical tracking systems depend on line-of-sight to the wand which might accidentally be occluded during interaction, especially in a collaborative setting. They also tend to induce fatigue [29], as the user must keep their hands suspended in mid-air for extended periods of time.

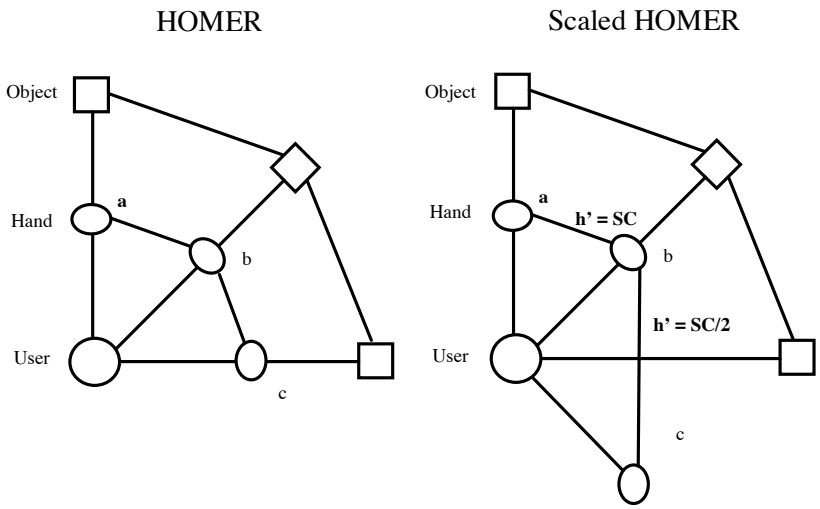


Figure 2.2: The classic HOMER technique on the left, Scaled HOMER on the right. [73]. On Scaled HOMER, when the arm slows down, the objects position is further scaled down to enhance precision.

2.3 Large Displays/Cave

3D interaction in front of a large display is not much different to immersive VR with an HMD and some of the techniques used in immersive VR could be applied to large displays. However, the major difference with HMD's and CAVE systems is that interaction in front of a large display often involves more than one user (e.g. a presenter and an audience, or two collaborators) and that makes it problematic to track the viewpoint. If the viewpoint cannot be tracked, implementation of ray-based techniques [39] becomes problematic because the ray does not look like it is emanating from the wand. Navidget [21], an alternative to ray techniques for large display interaction, uses 2D input on a tablet to position the camera in a 3D environment. Although this is similar to manipulating an object for inspection, their technique does not directly support object manipulation so it cannot be applied to collaborative scenarios (more than one users could not control the viewpoint simultaneously). The authors reported good usability for both novice and expert users based on questionnaires. Fröhlich et al. [16] presented the cubic mouse, a box with three perpendicular rods passing through its center. The authors report positive reactions from participants, yet the device form factor makes it difficult to relax the non-dominant arm in a presentation as the rods can be accidentally pressed against the presenter's body and thus induce accidental input.

Song et al. used a Kinect to track the user's limbs in front of a large display and proposed a handlebar metaphor [65] for 3D object manipulation. Their users, however, complained about fatigue. In an attempt to address fatigue in large display interaction, our earlier work [39] proposed a set of techniques that allow manipulation by holding the wand at hip height, yet that work, like other ray-based work is hard to implement in a situation where the viewpoint is not being tracked or when it is necessary to interact from the skewed position of a presenter.

2.4 Tabletop/Touch

Touch surfaces share some of the problem of the mouse, being limited to two integral DOF per touch point and typically do not allow for simultaneous translation and rotation. Various attempts have been made to address 3D manipulation using *on-screen* multi-touch. Reisman et al. [60] presented a multi-touch 3D object manipulation technique that depends on a constraint solver based on the user's perspective. However, their system was not empirically evaluated, and has some drawbacks such as ambiguous or unwanted rotations. Hancock et al. [24, 25] introduced sticky tools, a technique used to support tabletop 3D object manipulation. Hancock's technique allows simultaneous translation and rotation on a subset of the available axes. Martinet et al. citemartinet proposed a 3D manipulation technique based on the separation of translation and rotation. Martinet demonstrated benefits from separating translation and rotation in 3D manipulation. Cohé et al. [11] introduced tBox, a 3D manipulation widget for touch-screens.

The authors of tBox conducted a study using a 3D object assembly task by novice users and found tBox was an effective solution. A limitation of tBox is that the user needs to reach the edge of the widget with the cylinder in order to enact translation (Figure 2.3) which might not be suitable for rapid successive manipulations.

Wilson et al. [74] used physics simulation to manipulate 3D objects with a tabletop display. The solution by Wilson et al. is one of the few solutions in on-screen touch that allows for *integrality* during manipulation. Tse [67] investigated the use of touch and tangible objects for 3D object manipulation in education (i.e. presentations). They found that participants struggled with camera positioning but appreciated touch-based object rotation.

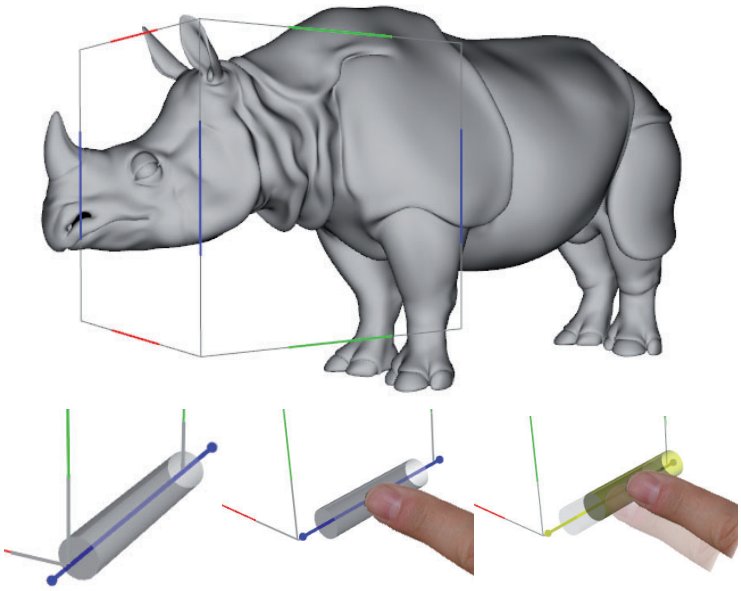


Figure 2.3: tBox[11]. A sliding widget appears when a colored segment is touched. The translation starts after the slider collides with the other edges of the box. When the projected size of an edge is too small, the translation slider is adapted, so the gesture performed before starting a translation is similar for any situations.

The aforementioned 3D interaction techniques that utilise *on-screen* touch are all limited by display size. When the display exceeds a certain size threshold, touch input starts to become cumbersome. The user is required to cover a large area with physical movements and parts of the display area are out of arm's reach. This is the case with large tiled displays, or those using projectors. In addition, physically approaching the display limits the user's activity to a very small area while in collaborative systems the interacting user obstructs the view for the rest of the group.

Off-screem touch is ergonomically superior to on-screen touch while at the same time overcoming the aforementioned problems with collaboration. Conversely, with off-screen touch there is no cursor for selection [70]. In an attempt to overcome this DOF limitation Ohnishi et al. [48] used two touch pads resting on a desktop 3D selection and annotation scenario. Finally, Wigdor et al. [72] proposed a set of techniques that employ shaped touches to control gain, overcome occlusion avoidance, and manage separation of constraints in a 2D task. Their approach, however novel, requires a tabletop and would be hard to implement on a large screen interaction scenario.

Despite many contributions having been made by earlier researchers, there is no widely accepted standard for 3D model manipulation on a large display. Our work, in an attempt to address the outstanding issues presents a number of solutions in the following chapters.

2.5 Drawing on a 3D Mesh

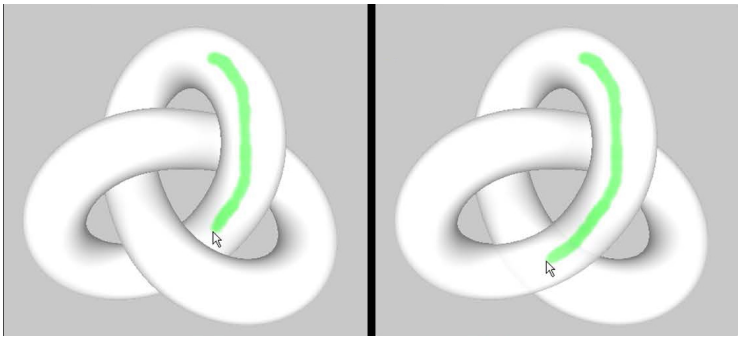


Figure 2.4: Layerpaint [18]. Region in the front becomes transparent to allow for the stroke to continue uninterrupted.

Chapter 5 examines making long strokes on a 3D mesh. Fu et al. introduced layerpaint [18] , a system that automatically re-orders layers in a 3D mesh to allow for long strokes (Figure 2.4). Layerpaint works by segmenting the mesh into regions then popping up the appropriate occluded region to ensure a smooth stroke. That solution, however, is only suitable for a certain type of mesh and might not work well for meshes which are not smooth. The same limitation applies to the work by Ortega et al. [51]. Their solution to making a long stroke on a 3D mesh is to automatically move the camera around the 3D model so that the system is able to keep the draw point under the mouse (Figure 2.5).

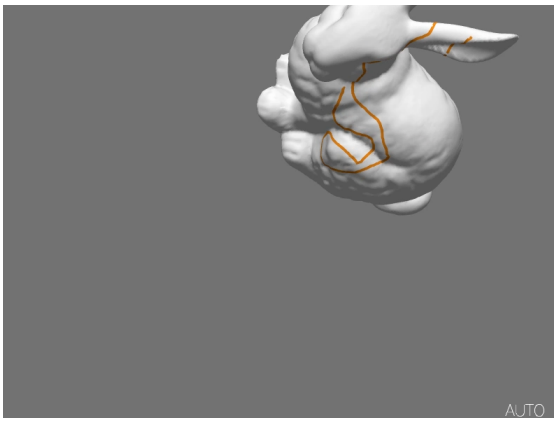


Figure 2.5: ACCD [51]. Depending on the direction of the stroke the model might drift off-screen (model is moving towards the top-right of the figure in this case) and the user manually needs to bring it back to the center to continue drawing.

Chapter 3

Mesh-Grab and Arcball-3D: Ray-Based 6-DOF object manipulation

3.1 Introduction

Advances in computer graphics and display hardware technologies have made it possible to render near-photorealistic graphics at interactive frame rates and to enjoy rich content from a distance on large displays and projectors. Large displays are especially important in education, engineering/CAD and science as they allow for multiple users to simultaneously view the same contents without obstructing one another. Naturally, users not only want to view the contents, but also want to interact with them.

This work focuses on remote 3D object manipulation on a large display for interactive presentations. An example could be in education, where a professor in medical school is projecting a human heart during class. In such a scenario the professor would need to rotate and translate the heart (6-DOF) to show different perspectives and answer questions from students (Figure 1.2).

In an architectural meeting the participants would need to control the position and location of the camera to inspect the building from various angles to discuss improvements. Similarly in an engineering meeting participants often need to interact with a 3D model to discuss design issues.

Some available remote interaction controllers and techniques suffer from fatigue issues or lack intuitiveness in addition to the expensive and complicated setup required. Motivated by this, we have developed two new ray-based 3D interaction techniques: **Mesh-Grab** and **Arcball-3D** (inspired by Shoemake's Arcball[63]). The proposed techniques are an extension to techniques used with a 2D pointer (like a mouse pointer[63] or a single finger [24]). The main contribution of this work is the exploration of a mode that combines rotation and translation into a single action. There is, to our knowledge, no ray-based interaction technique that accomplishes this.

As such, this study aims to answer the following questions:

- How do these 3D extensions of classic techniques perform relative to direct manipulation (with a 6-DOF tracker)?
- How well can users combine rotation and translation into a single action using the combined mode?

This study also aims to fill a gap in the literature ever since the evaluation of the first ray casting techniques [56]. Whereas direct interaction techniques have matured, notably by input scaling [73], there has been little progress in ray-based techniques. Part of the problem was that tracking a wand in 3D used to be expensive (Optitrack) or tethered (Polhemus FasTrack) but with the emergence of novel hardware (PlayStation Move, Kinect) there are new opportunities for wand-based interaction.

3.2 Related Work

There is a large body of work on manipulating objects from a distance in immersive VR [53, 55, 66, 73]. Much of that work focuses on selecting and placing remote objects. In applications where a 3D model is being demonstrated to an audience, however, the user would most likely need to interact in a limited depth, a form of FishTank VR or "*Shallow depth interaction*"[24]. Some other efforts for remote 3D object manipulation include work by Song et al. [65] that uses Microsoft Kinect to track the user's arms and manipulate objects using a handle-bar metaphor. Myers et al. looked at the performance of laser pointer-like devices and concluded that conventional interaction techniques and UI paradigms are not suitable for use with laser pointers [47] (ray-casting with a wand is similar to using a laser-pointer).

Touch-input: Cohe et al. [11] introduced tBox, a 3D manipulation widget for touch-screens and found that users could easily assemble a 3D object with it. Touch input enables users to interact with a display by removing an indirection layer, and there are quite a few solutions for 3D interaction by using multi-touch[24]. However when the display size exceeds a certain threshold, touch input ceases to be an option as the user needs to cover a large area with physical movements and in some cases the display area is out of reach (as is the case with projectors and tile-displays). In addition to the aforementioned problems, physically approaching the display to interact limits the user's activity to a very small area of the display and in the case of collaborative work, the interacting user obscures the display for the rest of the group. In an attempt to address these limitations, Katzakis et al. introduced Plane-Casting, a technique for 3D cursor control using a Smartphone [38].

Performance-wise, direct interaction has long been accepted as the most natural way to interact with an acquired 3D object [79], yet one of its problems lies in tracking (Figure 3.1). To allow for free manipulation of the object with the fingers using most currently available technologies requires a magnetic tracker. Magnetic trackers are not practical for daily use, they require an expensive, complicated setup and are prone to interference. As such they are unsuitable in an active learning or presentation environment where presenters and audience want to move around and interact with the display. With optical tracking, attaching retro-reflective markers makes for a bulky controller and occluding the retro-reflective markers during rotation often causes loss of tracking. There are some efforts underway for RGBD-based 6-DOF tracking [27] but those methods still lack robustness for practical use.

Wilkes and Bowman proposed Scaled HOMER [73]. Scaled HOMER is an extension to the classic hand-centered manipulation, HOMER technique by Bowman and Hodges [8] in which the authors combine PRISM-like[15] velocity-scaling with the, already input scaled, HOMER technique for improved performance. The authors reported an improvement over the standard HOMER technique, however, their evaluation task is, essentially, only a 2D movement task, not a 3D task. As such another goal of our study was to ascertain the performance of Scaled HOMER in a proper 3D task.

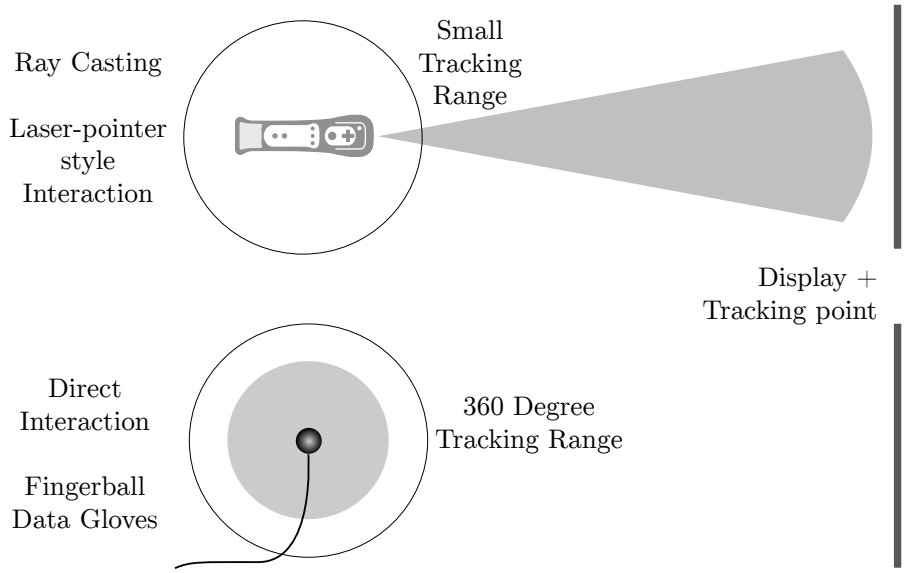


Figure 3.1: To rotate an object in 3D, direct interaction requires full range tracking whereas ray-based techniques only require a small angle which can be implemented at lower cost (e.g. IR leds + Wii-mote camera or PS Move controller).

3.3 Proposed Techniques

The design philosophy behind the techniques is to allow as much as possible for intuitive and fatigue-free interaction based on pointing. Pointing is very intuitive to humans and it is used as a means of communication from a very young age, even before the development of speech [9]. Similarly, pointing with a laser pointer-like device is also intuitive, but extending the arms in front of the torso is slow and induces fatigue over time. As such, we noticed users tend to shoot laser pointers from the hip or the abdominal level during our pilot trials. This is possible because pointing, even in the real world, is based on continuous feedback from vision and proprioception. So similar to how users, along with proprioception, see their arm and finger for feedback in natural pointing, they use proprioception and the pointer's dot as feedback when using a laser pointer.

The two proposed techniques are a combination of 3 different modes, depending on the buttons the users are pressing. There is a translation mode, a rotation mode and a combined Translation+Rotation mode.

Mesh-Grab

In the first of our two evaluated techniques, Mesh-grab, the user casts a ray from a handheld wand (Wiimote in our implementation). The ray intersects the mesh at some point. With the simultaneous press of buttons A+B (thumb and forefinger) the user can "skewer" the mesh at that point. While the buttons are pressed the mesh behaves as if the ray was a solid link that hinges on the mesh with freedom to move on one hemisphere, like a ball-point joint (Figure 3.3). Calculation of the behavior of the object is based on a constraint solver that ensures that, while the buttons are pressed, the acquired point always remains under the ray, at the same, locked, distance from the wand.

The axis of rotation $\vec{R}$ is calculated from the cross product of the two vectors (Figure 3.2). $\vec{v1}$ is a vector starting from the center of the object and ending at the intersection point while $\vec{v2}$ is the same vector following movement of the ray:

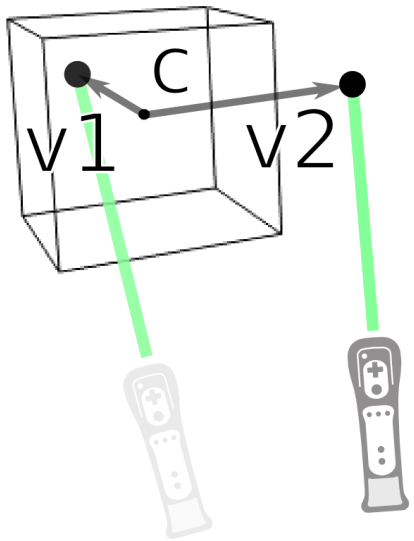


Figure 3.2: The translation and rotation algorithm of the combined mode. C is the center of the object.

$$\vec{R} = \vec{v1} \times \vec{v2}$$

and the angle θ from their dot product.

$$\theta = \arccos \vec{v1} \cdot \vec{v2}$$

Thus we calculate one transformation matrix $\boldsymbol{M_r}$, that holds that rotation θ in its rotation component. To judge whether or not translation is applied we compare if the length of $\vec{v2}$ is different than $\vec{v1}$. If it is, then we calculate a transformation matrix $\boldsymbol{M_t}$ as a unit matrix with the translation component being the vector $\vec{t}$. $\vec{t}$ is a vector who's length is the difference in magnitude of the two vectors $\vec{v1}$ and $\vec{v2}$, on the direction of $\vec{v2}$.

$$|\vec{t}| = |\vec{v2}| - |\vec{v1}|$$

The transformation $\mathbf{M}$ applied finally is the combination of the rotation and translation matrices:

$$[\boldsymbol{M}] = [\boldsymbol{M_r}] \times [\boldsymbol{M_t}]$$

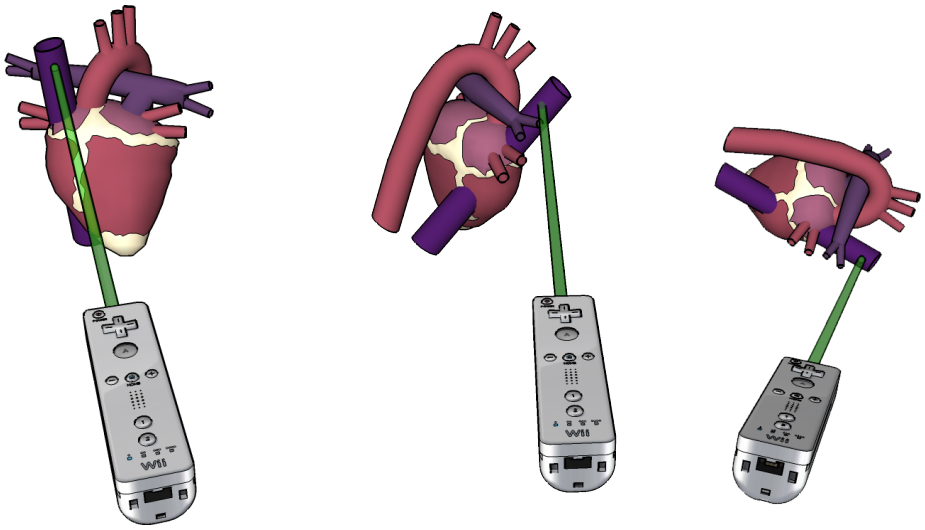


Figure 3.3: Manipulating a 3D model using Mesh-Grab.

With only button B (forefinger) on the Wii-mote, the acquired mesh is only translated (no rotation applied), then finally with only button A (thumb), interaction is limited to rotation only. Users can also use the d-pad on the Wii-mote to retract or expand the ray while on any of these modes (instead of translating the wand).

In the innertia version of the techniques, if there was motion upon release of the buttons, the object continued to move as if it were in a weightless environment with the following function:

$$[\boldsymbol{M}]_i = [\boldsymbol{M}]_{i-1} \cdot \alpha$$

where $\boldsymbol{M}$ is the transformation matrix at the frame when the button was released (frame i) and α[1] is the attenuation constant. The flicking algorithm applied translation and rotation separately, as such it was possible to make complicated combinations of moves, as if manipulating a weightless object in space. If the object was re-grabbed during flicking, motion would stop. During early trials we found that beginner users could not handle flicking very well, as such flicking was disabled for the experiment tasks. We think that better tuning of the flicking algorithm to take into account a backlog of more than the last frame of motion could potentially result in a much better interaction.

Arcball-3D

The design rationale behind Arcball-3D was that for simple meshes, it would be easy to predict how the mesh would behave when hit with the ray and "pulled" or "pushed", but for a complicated mesh with holes and extrusions the users might struggle to hit the mesh or manipulate it in the desired way. As such we hypothesized that a sphere bounding the mesh would result in an interaction that is more predictable and consistent across different mesh types.

Arcball-3D works in exactly the same way as Mesh-Grab. The ray cast from the wand intercepts a sphere, one that tightly encloses the manipulated mesh. The interaction modalities available following acquisition of the sphere are identical to Mesh-Grab.

Arcball-3D is fundamentally different from the original Arcball technique by Shoemake [63]. In the original Arcball, the screen location of the 2D cursor is projected onto a sphere and the resulting motion is an *interpretation* of the *2D* mouse motion. In Arcball-3D the ray-sphere intersection is calculated from real 3D inputs and the resulting motion is not a result of mathematical interpretation. It's the result of the actual motion, in *3D*, of the intersection point. Unlike the original Arcball, in Arcball-3D the ray can be cast from an arbitrary point in 3D space and thus it is possible to achieve manipulation from skewed

[1] 0.95 in our implementation

positions, as opposed to the original Arcball which only considers manipulation from the front.

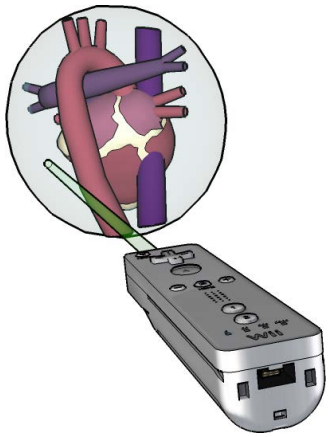


Figure 3.4: Arcball-3D, the ray intersects a bounding sphere instead of the object.

3.4 Evaluation

We conducted a formal experiment to evaluate the performance of our proposed techniques as compared with Scaled HOMER[73], one of the state-of-the-art in direct manipulation techniques. We wanted to test the following hypotheses:

- Mesh-Grab and Arcball-3D will significantly underperform compared to Scaled HOMER (hypothesis 1)
- Users will prefer the ray techniques overall because of fatigue using the direct interaction technique (hypothesis 2)

Based on the literature on 3D interactions[17], users usually prefer to keep DOF manipulation seperate when it comes to high DOF devices. As such we hypothesize that:

- Users will prefer the separate modes as opposed to the combined mode. i.e. users will prefer to separate translation from rotation (hypothesis 3).

Apparatus: Participants sat 3m away from a short-focus monoscopic 3D projector screen and held the wand in their dominant hand. The Wii-mote-based

wand had a wooden extension where a 3D tracker (Polhemus Fasttrak) was attached to avoid electromagnetic interference from the Wii-mote. The wand was tracked in 6-DOF. The wooden extension was attached in such a way that it does not in any way interfere with the Wii-mote's natural grip position (Figure 3.5). A foot pedal was provided to advance to the next trial (Figure 3.6). For Arcball-3D participants held the Wii-mote in their dominant hand, whereas for Scaled HOMER, users held the polhemus receiver wrapped in a spherical fingerball-like container and held the Wii-mote in their off-hand for button presses. We had users face the screen, as opposed to Figure 1.2, since we felt that interacting from a skewed position would be disadvantageous to Scaled HOMER.

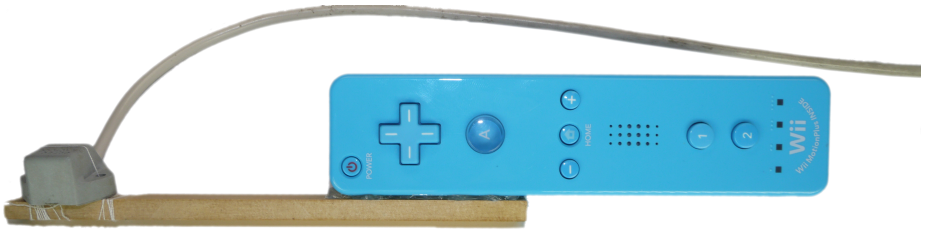


Figure 3.5: Wii-mote with the polhemus Fasttrak receiver attached. The extension does not affect the natural grip position of the Wii-mote.

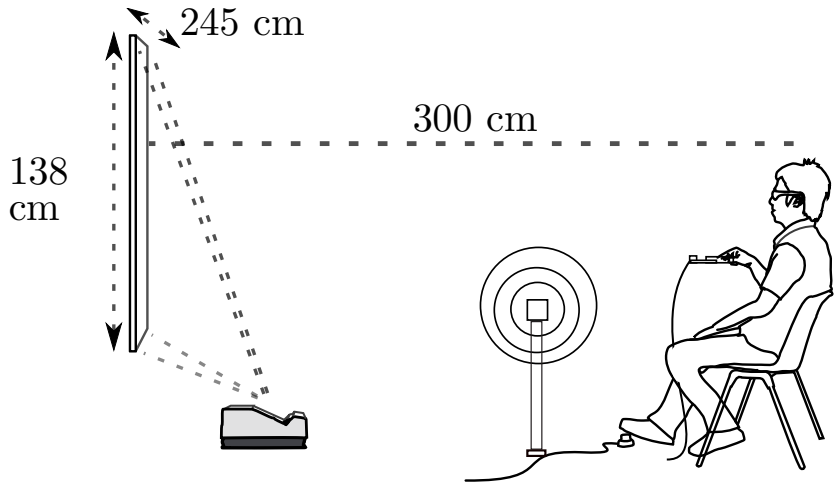


Figure 3.6: The experimental setup.

Task: The hypotheses were tested based on the collective results of two different tasks: A 6-DOF docking task, where a pyramid-shaped cursor was docked inside a wireframe copy, and a rotation experiment where the pyramid cursor and wireframe target were matched position-wise and all participants had to do was match the rotation. Participants received a brief explanation of the techniques and were allowed to practice briefly prior to commencement. They repeated

the tests with all three techniques in a counter-balanced order: Mesh-Grab, Arcball-3D and Scaled HOMER.

Subjects: 13 Male participants, undergraduate and graduate students in computer science took part in the experiment (21-30 years old). The experiment lasted approximately 30 mins.

Docking Experiment: In the 6-DOF docking experiment the target appears in 12 pre-defined locations that lie on a sphere centered at the starting position of the cursor, the center of the screen (Figure 3.7). The radius of the sphere was approximately 52cm on screen. Locations 1-4 are on the X and Y axis whereas locations 5-12 are each in one of the 8 octants of the system defined by the cursor starting position.

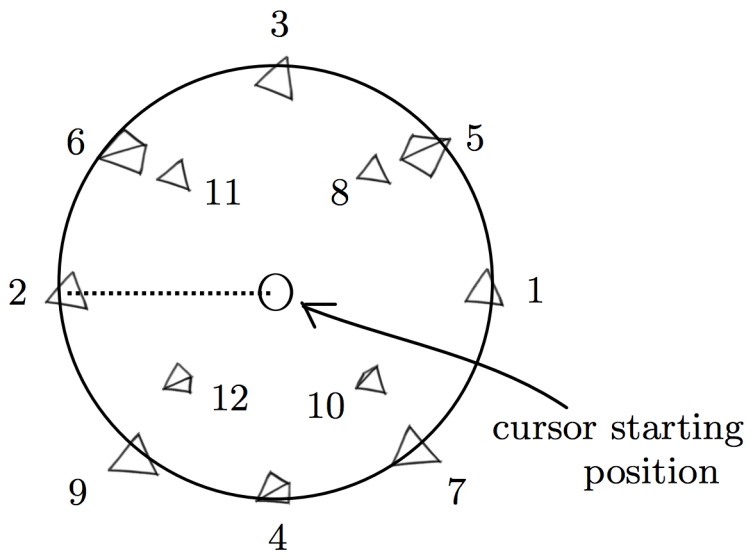


Figure 3.7: Targets were located on the surface of a sphere equidistant from the cursor starting point.

Targets were rotated about the $(-1, -1, -1)$ axis by a 10° interval depending on their position number starting from 20°, i.e., position 1 had a 20° rotation, 2 had 30°... 12 had 130°. Rather than random rotations, these rotations were chosen to allow for easier reproducibilty of the experiment. Participants selected the cursor using ray-casting (in all 3 techniques), then docked the cursor in the wireframe target and when they felt they had a good match they could press the foot-pedal to proceed. When the cursor pyramid's colored vertices approached the corresponding vertices of the target past a certain threshold (1.0f), the vertex lit green to signify a match (Figure 3.8). Participants could then press the pedal, in which case the cursor was reset in the starting position and the

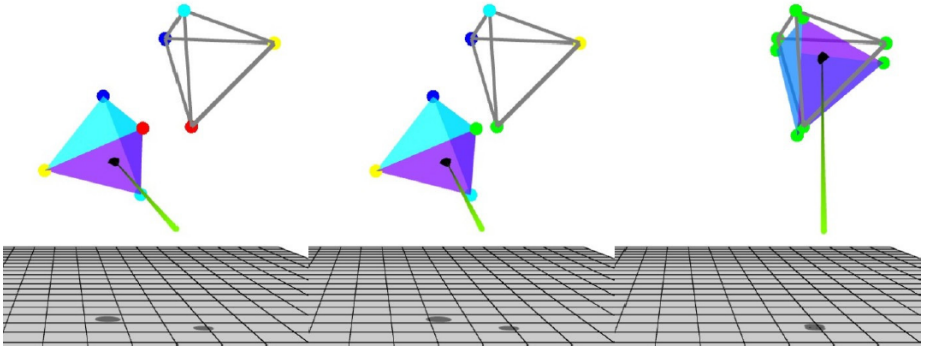


Figure 3.8: Docking the pyramid in the wireframe target.

target randomly appeared in one of the remaining positions until all 12 positions had been cycled twice (for a total of 12 positions $\times$ 2 times $\times$ 3 techniques = 72 trials).

During the experiments, we measured the following performance metrics:

- **Position Offset**, as a measure of the 3D Euclidean distance between the (cursor and target's) pyramid centers (only for the docking task).
- **Rotation Offset**, as a measure of the sum of the distances between each vertex of the cursor pyramid to the corresponding vertex of the target (only for the rotation task).
- **Movement Time** (MT) from the time the cursor appeared, until the time docking was complete (participant pressed the pedal).

3.5 Results

Docking Experiment Results: A within-subjects ANOVA showed a strong effect between technique and movement time ($F_{(2,24)} = 4.04$ $p < .031$). Movement times means are summarized in Figure 3.9. A Holm pairwise comparison revealed the significant differences between the techniques (Mesh-Grab vs Arcball-3D $p < 0.1$, Scaled-Homer vs Arcball-3D $p < 0.017$, Scaled-Homer vs Mesh-Grab $p < 0.1$).

As predicted (hypothesis 1) Scaled HOMER was the fastest among the 3 techniques. However, contrary to our expectations, Scaled HOMER was only slightly faster than the other two techniques (8.47s), closely followed by Arcball-3D

(9.69s) and then by Mesh-Grab (11.9s). We expected the performance gap to be wider and this result was a surprise as it indicated that our proposed techniques perform almost equally well when put up against the state-of-the-art.

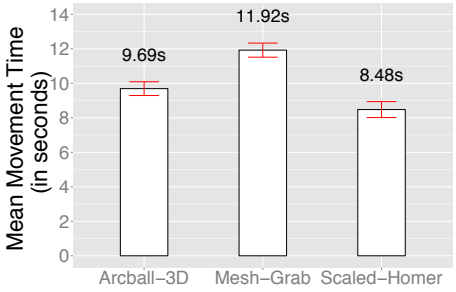


Figure 3.9: Mean movement time (MT) in seconds (Docking Task).

In terms of Position Offset, again to our surprise, Arcball-3D was the most accurate among the 3 techniques, marginally followed by Mesh-Grab and then Scaled HOMER ($F_{(2,24)} = 12.81$ $p < 0.001$). The Holm Post-hoc test revealed significant differences between Arcball-3D and Scaled HOMER ($p < 001$) as well as Mesh-Grab and Scaled HOMER ($p < .001$). Position Accuracy results are summarized in Figure 3.10.

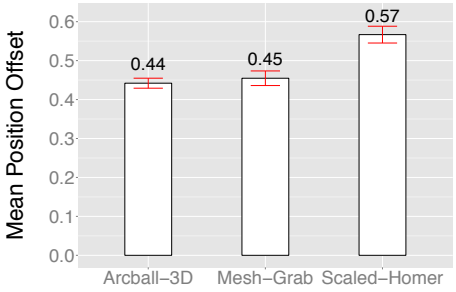


Figure 3.10: Mean position offset (Docking Task).

A reason for the poor performance of Scaled HOMER in accuracy could be the fact that participants need to hold their arm suspended in mid air for a number of seconds while they attempt to match the target accurately. Perhaps fatigue motivated them to skip accurate matching and end the trial sooner. Fatigue could also account for Scaled HOMER's performance gain in Movement Time. It is possible that because participants wanted to avoid fatigue, they attempted to expedite the trials.

It should be noted that this experiment, being very short in duration, favors Scaled HOMER, that probably requires a longer experiment before its weaknesses start becoming apparent (users annoyed by the cable of the tethered

tracker, fatigue onset etc.). This assumption further strengthens the case for Mesh-Grab and Arcball-3D.

There was also an effect of technique on Rotation Offset between techniques ($F_{(2,24)} = 3.35$ $p < 0.01$). A post-hoc analysis revealed that, using Arcball-3D, participants matched the target's rotation 12% more accurately than Mesh-Grab, ($p < 0.01$) but the differences with Scaled HOMER were not found to be significant ($p = 0.14$). No statistically significant difference was found between Mesh-Grab and Scaled HOMER ($p = 0.19$).

Rotation Experiment: In the Rotation experiment the two pyramids were centered at the same point and again users were asked to rotate to match the wireframe's rotation. Like the docking experiment, users were asked to match 12 preset rotations, twice, in random order. Again the vertices lit up when in close proximity(0.5f) to signify a match and users could press the foot pedal to advance to the next trial. Users repeated the task with all 3 techniques.

Rotation Experiment Results: On the rotation task, to our surprise, Arcball-3D had the lowest completion time mean (8.77s) followed by mesh-grab (10.34s) and then by Scaled HOMER (10.49). A repeated-measures ANOVA found no statistical difference ($F_{(2,24)} = 1.663$ $p < 0.22$), however, a pairwise Holm test revealed the significant differences: Arcball-3D vs Scaled HOMER $p < 0.001$, Arcball-3D vs Mesh-Grab $p < 0.001$, Mesh-Grab vs Scaled HOMER $p = 0.7$, i.e., no significant difference.

The results are summarized in Figure 3.11. Technique had no effect on accuracy in the rotation task ($F_{(2,24)} = 0.13$ $p = 0.87$). It should be noted that had outliers been removed, the statistics were more in favor of our proposed techniques but we chose to include *all* trials for the sake of completeness.

The rotation experiment results are slightly controversial as manipulating a fingerball for rotation has long been believed to be the most intuitive means for rotation [79]. Yet our results show that a ray-based technique can perform equally well or even better than direct manipulation. This result calls for further analysis and scrutiny. A potential explanation for Scaled HOMER's reduced performance in the rotation task is the fact that users had to hold the fingerball in one hand. Manipulating, clutching and avoiding the tethered line with just one hand could potentially be one of the reasons for the rotation performance deficit.

When presenting a 3D model to an audience (depending on the situation) one could argue that rotation is more important than translation, i.e., the model

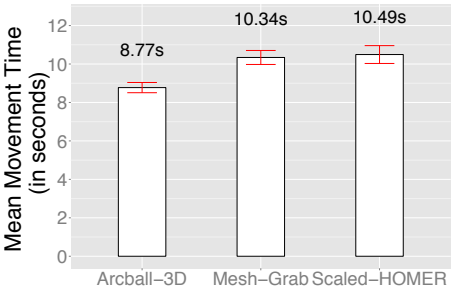


Figure 3.11: Rotation Task Completion Time means.

is centered and occupies as much of the screen as possible then the presenter rotates to show the features (perhaps that would be the case in the scenario where a professor is demonstrating anatomy). That is the reason the rotation only task was chosen as one of the two evaluation tasks. For an application scenario such as the aformentioned one, the results of the rotation experiment overall bear more gravity than the docking task.

Finally, analysis of the docking time per position for each technique suggests that Arcball-3D showed a slightly more "linear" response to the increasing rotation angles per docking position (Figure 3.12). Scaled HOMER's response was less

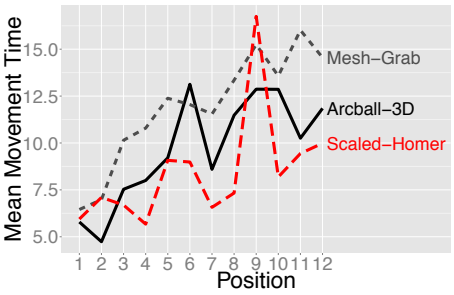


Figure 3.12: Mean Completion Time per target position (refer to Figure 3.7) for positions.

linear. This could perhaps be due to the fact that, past a certain angle of rotation, when using the fingerball-like-tracker for Scaled HOMER participants were often forced to clutch and handle the fingerball between their fingers. This perhaps explains the sudden jump for Scaled HOMER around position 9 (100° rotation). At that point users possibly having met the supination limit of their wrist were forced to clutch to keep rotating. Regression analysis on the results is required, however, before definite conclusions are reached.

Interaction Modes: The ray-based techniques had three interaction modes:

Table 3.1: Summary of evaluation results. Techniques ranked depending on their relative performance.

Technique	Quantitative Performance	User Preference	Cost Effectiveness	Freedom to Move	Fatigue
Arcball-3D	A-	A	A	A	A
Scaled HOMER	A	B	C	C	B
Mesh-Grab	B	C	A	A	A

Translation only (B), rotation only (A), and translation+rotation (A+B). We calculated what percentage of the task time was used in each mode. The results from the Mesh-Grab technique show that on average, during a trial, participants spent 43% of the task time without pressing any buttons (i.e., aiming the ray), 27% rotating and 25% translating, while only 4% of the task time was spent using the combined mode. This result indicates that users found it difficult to use the mode that combines rotation and translation and preferred to manipulate 3-DOF at a time. In Arcball-3D, results are almost identical but participants used the combined mode even less (1% of the entire time). We believe that adding to this discrepancy is perhaps the fact that users needed to press two buttons for the rotation+translation mode, yet that is only a speculation. These results confirm the third hypothesis and are in line with the literature on multi-DOF manipulations. Even though this analysis addresses the 2nd research question that was presented in the introduction, a better analysis that shows how much each mode contributed to each type of motion (rotation, translation etc) in the docking task could shed additional light to the usability of each mode.

Qualitative Evaluation: Following completion of the experiment with each technique, participants were asked to rate the techniques on the following aspects: Ease of use, Accuracy, Fatigue and Fun to use using a 7-point likert scale questionnaire (Example question: The technique was easy to use: [Strongly Disagree 1... 3 Neutral ... 7 - Strongly Agree]). A Kruskal-Wallis test and post-hoc analysis revealed that participants rated Arcball-3D as the easiest to use, followed by Scaled HOMER then Mesh-Grab. This confirms the second hypothesis.

The slightly poorer performance of Mesh-Grab could be due to the tetrahedron shape of the item that was being manipulated and that perhaps a cube or another convex shaped mesh could have different results.

Finally most participants commented on the arms fatigue and that it was easier to cast a ray from the hip than holding the tracker in front of the torso. There

were also comments on the awckward posture that is sometimes the case with the fingerball-like tracker on Scaled HOMER. Results of the questionnaires are summarized on table 3.2.

Table 3.2: Summary of the questionnaire responses.

Technique	Mesh	Arc3D	HOMER	K-Wallis
Easy to use	3.77	5.15	4.38	($p < .05$)
Accurate	3.08	4.54	5.23	($p < .01$)
Tiring	4.15	3.46	5.31	($p < .05$)
Fun to use	3.92	5.31	5.31	($p < .05$)

3.6 Discussion

All three techniques compared share similar weaknesses when it comes to performing axis-aligned motions or returning to the exact same point or via the exact same path. However such weaknesses exist in most devices with integral DOF control, like the mouse, the Space Navigator [1] or the Phantom [46] but that sort of motion is not necessarily required by our intended application scenario. Finally, a real-world application in education or visualization would ideally allow for annotations that might not be easy to do from a distance with a ray technique without input scaling and smoothing to account for distance from the screen amplifying arm jitter. We summarize the relative strengths and weaknesses of each technique on Table 3.1 by our subjective ranking.

Arcball-3D performed almost as well as Scaled HOMER, it was highly preferred by the users, it can be implemented at lower cost, it allows for free movement of the presenter or within a group of collaborators and induces little fatigue. Mesh-Grab received similar scores but since users showed the least preference and due to it's lower quantitative performance it ranks third overall. Scaled HOMER did well performance-wise but users did not prefer it and considering the tethering, high costs and fatigue, it ranks second.

Our implementation of flicking proved to be slightly troublesome for the experiment participants in the pilot trials but expert users users, that were exposed to the techniques for a long time, showed interest in flicking. The results in this work could potentially be transferred to interaction in Virtual Reality while wearing an HMD or to Augmented Reality (AR).

Novice users did not prefer the combined mode (translation+rotation) but expert users commented favorably and we believe a better tuned combined mode

algorithm could prove competent yet further studies need to be conducted.

3.7 Conclusion and Future Work

We presented an evaluation of two techniques for ray-based 3D object manipulation, Mesh-Grab and Arcball-3D. Our evaluation demonstrates that the new techniques perform very well against the state-of-the-art in direct manipulation, Scaled HOMER. Arcball-3D was the overall winner in our evaluation and this indicates that 3D object manipulation on a large display with ray-based techniques can perform well at short-to-medium distances. In addition to the user's preference, as ray-casting based techniques can be implemented with a wider variety of hardware or with off-the-shelf consumer products, we conclude that they are a better alternative to direct interaction for remote 3D manipulations.

Future Work: Flicking with physics was disabled for the experiment and tuning the physics algorithm better, remains as future work since we are interested in establishing how well users can throw objects around and predict their motion paths like they do with physical objects in the real world. We are also interested in exploring different controller form factors that can offer more input modalities (text input or a stylus for annotations etc).

Chapter 4

Spatial Characteristics of Raycasting-based vs Direct Interaction in a 6-DOF Docking Task

4.1 Introduction

A handheld tracked wand is the most popular method for selecting and manipulating data in 3D [75, 42]. As virtual reality, augmented reality or large/tile displays technology advances and 3D manipulation becomes more commonplace there is an increasing motivation to design effective 3D manipulation techniques[1]. understand the characteristics of 6-DOF interaction in depth in order . Despite this motivation other than some early work by Zhai [78, 79, 77] there are a limited number of previous studies on the subject. Analysing timestamped spatial data gathered from such a task is a challenge partly due to the large volume and the high dimensionality of the data.

This study contributes an analysis of a dataset from a 6-DOF docking experiment and attempts to provide quantifiable evidence that can aid in understanding the fatigue induced by common 3D manipulation techniques. We also contribute a methodology for averaging time-stamped 3D motion data in order

[1] A discussion regarding terminology can be found in the appendix

to study speed profiles from different docking trials. Finally we discuss our findings and contrast them with some earlier findings in the literature.

4.2 Related Work

Zhai [78] attempted to quantify coordination in 3D manipulation. They propose *efficiency* as a measure of quantifying coordination. The *efficiency* measure revealed performance differences between elastic rate and free position control. The study however, does not consider whether or not an efficient or coordinated motor performance is actually the least fatiguing and no attempt has been made since, to establish an association between their proposed measure, *efficiency*, and fatigue or effort. Acott et al. [2] also studied trajectory based tasks and proposed a "steering law" that models steering time.

Liu et al. [44] analysed dwelling in the ballistic and corrective phase of a 3D manipulation between real world and virtual reality. In addition to time spent in both phases being longer in VR, the authors also found that movements in VR comprised of more discrete sub-movements than in the real world. In follow-up work, the authors went on to propose a number of techniques that attempt to do real-time classification of motion for improving interaction [43]. Our work attempts to validate the existence of these stages in our docking task data and to establish whether or not these phases exist only during direct manipulation or also with raycasting-based manipulation.

In an attempt to tackle the issues of fatigue with direct techniques Katzakis et al. proposed Mesh-Grab and Arcball3D [39]. The techniques were compared against Scaled HOMER [73] and were found to perform on par. Although the authors designed the techniques for reduced fatigue they only present qualitative data from the questionnaires regarding fatigue.

The data used in this study come from the 6-DOF docking experiment of our earlier work. [39]. In the following section we will briefly present the techniques used in the study followed by the experiment, our analysis methodology and our results.

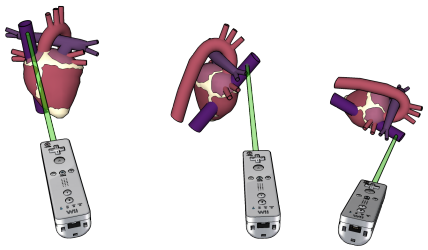


Figure 4.1: Manipulating a 3D model using Mesh-Grab.

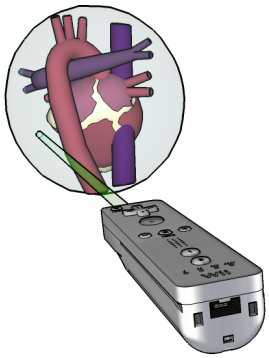


Figure 4.2: Arcball-3D: The ray intersects a bounding sphere instead of the object.

4.3 Ray Techniques: Mesh-Grab - Arcball-3D

Mesh-grab involves the user holding a wand and casting a ray from it into the 3D scene. The ray intersects a mesh. With the simultaneous press of buttons A+B (thumb and forefinger) the user can "skewer" the mesh at the intersection point. While the buttons are pressed the mesh behaves as if the ray was a solid link that hinges on the mesh with freedom to move on one hemisphere, like a ball-point joint (Figure 4.1). A constraint solver that ensures that, while the object is acquired, the acquired point always remains under the ray, at the same, locked, distance from the wand.

With only button B (forefinger) on the Wii-mote, the acquired mesh is only translated (no rotation applied, standard HOMER technique), then finally with only button A (thumb), interaction is limited to rotation only. Users can also use the d-pad on the Wii-mote to retract or expand the ray while on any of these modes (instead of translating the wand).

The logic behind Arcball-3D was that for simple meshes, it would be easy to predict how the mesh would behave when hit with the ray and "pulled" or "pushed",

but for a complicated mesh with holes and extrusions the users might struggle to hit the mesh or manipulate it in the desired way. As such we hypothesised that a sphere bounding the mesh would result in an interaction that is more predictable and consistent across different mesh types. Arcball-3D works in exactly the same way as Mesh-Grab. The ray cast from the wand intercepts a sphere, one that tightly encloses the manipulated mesh (figure 4.2). The interaction modes available following acquisition of the sphere are identical to Mesh-Grab.

4.4 Direct Technique: Scaled HOMER

Wilkes et al. proposed Scaled HOMER [73]. Scaled HOMER is an extension to the classic hand-centered manipulation HOMER technique by Bowman and Hodges [8] in which the authors combine PRISM-like[15] velocity-scaling with the already input scaled *HOMER* technique for improved performance. The authors reported an improvement over the standard HOMER technique, however, their evaluation task is, essentially, only a 2D task, not a 3D task. Scaled HOMER is one of the three techniques analysed in this work.

The reason these techniques were chosen is because both Scaled HOMER and Mesh-Grab/Arcball-3D can be executed with a wand.

4.5 Experiment

We briefly describe the experiment from Chapter 3, more details can be found there.

13 Male participants, undergraduate and graduate students in computer science took part in the experiment (21-30 years old). Participants sat across a projector screen and held a tracked wand. A foot pedal was provided to advance to the next trial (Figure 4.3)

Task: Subjects performed a 6-DOF docking task, where a pyramid-shaped cursor was docked inside a wireframe copy (Figure 4.4).

Participants received a brief explanation of the techniques and were allowed to practice briefly prior to commencement. They repeated the task with all three techniques in a counter-balanced order: Mesh-Grab, Arcball-3D and Scaled HOMER. It should be noted that essentially Mesh-Grab and Arcball-3D tech-

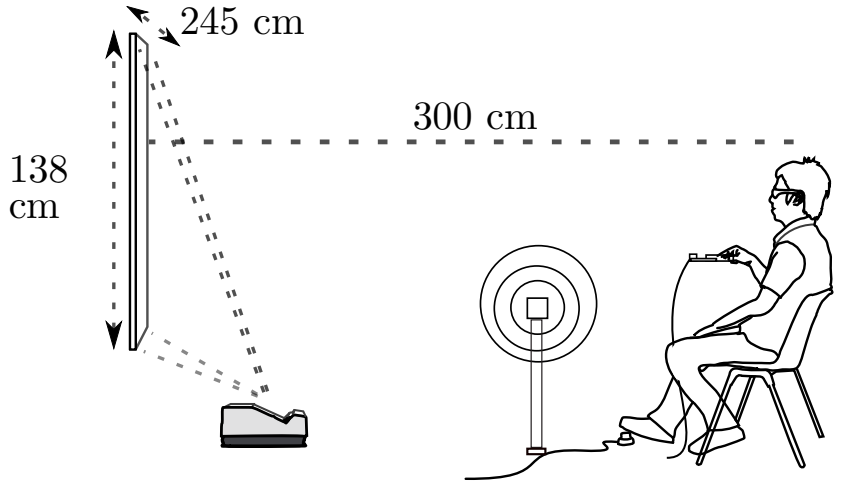


Figure 4.3: The experimental setup.

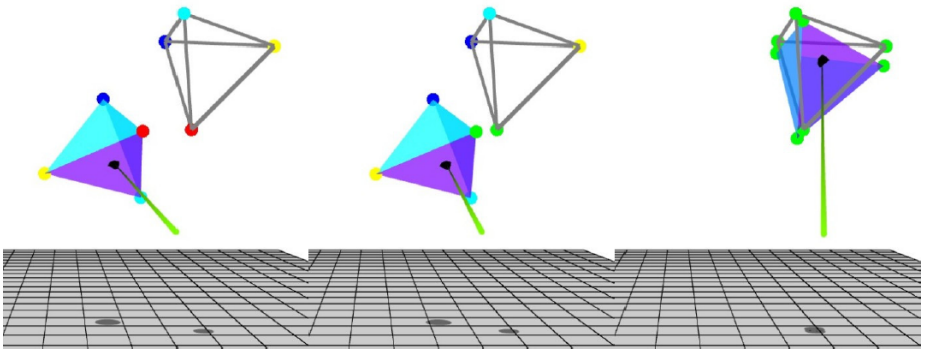


Figure 4.4: Docking the cursor in the wireframe target.

niques tested in chapter 3 were a subset of three techniques: The Mesh-Grab/Arcball-3D translation mode, a Mesh-Grab/Arcball-3D rotation mode, and a standard HOMER mode (not scaled) should the user wish to use it.

Task: The docking target appears in 12 pre-defined locations that lie on a sphere centered at the starting position of the cursor, the center of the screen (Figure 4.5). The radius of the sphere was approximately 52cm on screen. Locations 1-4 are on the X and Y axis whereas locations 5-12 are each in one of the 8 octants of the system defined by the cursor starting position.

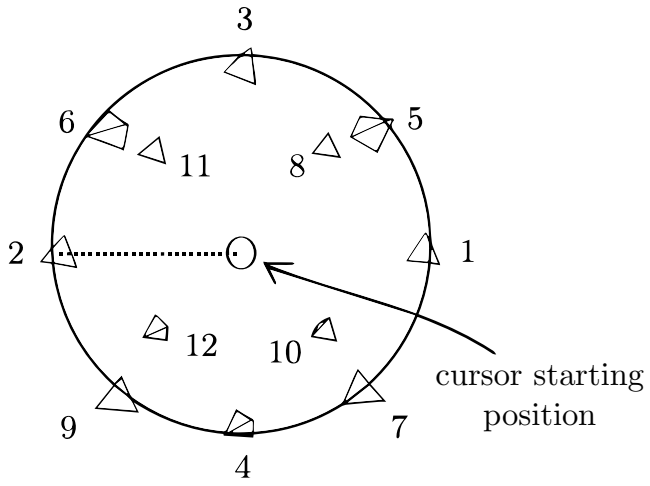


Figure 4.5: Targets were located on the surface of a sphere equidistant from the cursor starting point (used with permission).

Targets were rotated about the $(-1,-1,-1)$ axis by a 10° interval depending on their position number starting from 20°, i.e., position 1 had a 20° rotation, 2 had 30°... 12 had 130°. Participants selected the cursor using ray-casting (in all 3 techniques), then docked the cursor in the wireframe target. When they felt they had a good match they could press the foot-pedal to proceed. When the cursor pyramid's colored vertices approached the corresponding vertices of the target past a certain threshold, the vertex lit green to signify a match (Figure 4.4). Participants could then press the pedal, in which case the cursor was reset in the starting position and the target randomly appeared in one of the remaining positions until all 12 positions had been cycled twice (for a total of 12 positions $\times$ 2 times $\times$ 3 techniques = 72 trials).

4.6 Analysis Methodology, Results and Discussion

The authors of the original paper reported similar quantitative performance for all three techniques in docking times and accuracy. But the qualitative results were skewed in favor of Arcball 3D. Users were asked to rate techniques on a scale of 1..7. In terms of fatigue, which is the focus point of this work, users rated Scaled HOMER as the most fatiguing (average score 5.31) followed by Mesh-Grab (4.15) with ArcBall-3D the least fatiguing (3.92, $p < .05$).

During the task, the position of the tracker on the wand was recorded in a time series that spanned the beginning of the trial, until the pedal was pressed. A time-stamped series of coordinates in the 3D space of Euclidean geometry can be though of like any other time-series, yet standard statistical analysis tools [23] are not straightforward to apply because of the 3D nature of the samples. All analyses were conducted using R [57] and the raw data as well as the R scripts are available as appendix.

Learning Effect

A plot of task completion times for each trial is shown in figure 4.6 along with the learning curve from Zhai's experiment [77]. We used *Local Polynomial Regression Fitting* [37] on the learning effect data. The resulting fitted curve revealed some unusual trends in the learning effect. Usually, for each advancing trial users performance improves[78], but in this case users show a high variation in performance across the 24 trials. Despite the learning effects of Katzakis et al. being different from Zhai's they are similar to more recent results by Liu [44]. Both Liu and Zhai's experiments were longer than that of Katzakis et al. yet in the early stages where the learning effect should be greater there is a big difference. The discrepancy in learning effect curves could be due to the fact that Katzakis et al. allowed participants to practice beforehand (they don't report for exactly how long).

Although there was not much variation on the Y axis motion, participants arms covered a large area in the X-Z axis (looking down from the top - (Figure 4.7). In 3D manipulation, the X and Y axes are limited by the screen bounds but the Z axis (depth) is where there is most space for movement. That is the reason why we are examining the techniques from the top.

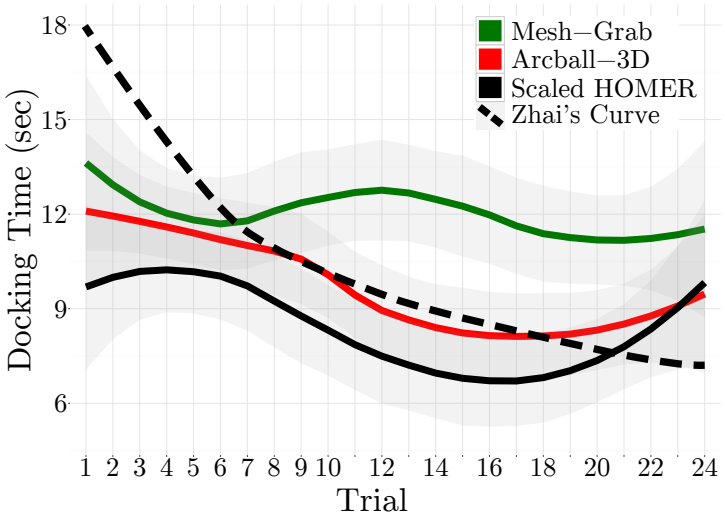


Figure 4.6: Learning effect for each technique. Trials using these techniques exhibited highly irregular learning curves.

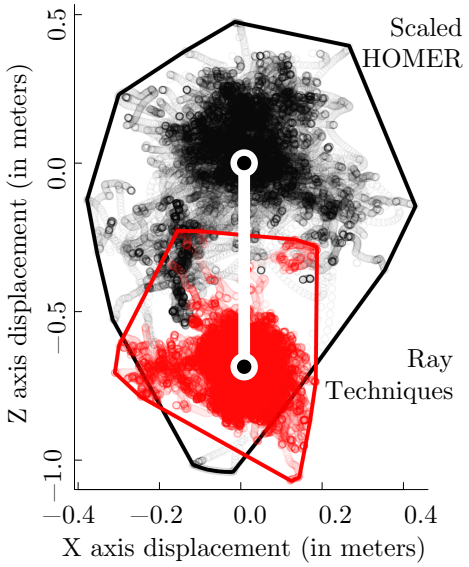


Figure 4.7: Areas covered as seen from the top. The ray techniques has been displaced vertically to avoid overlap.

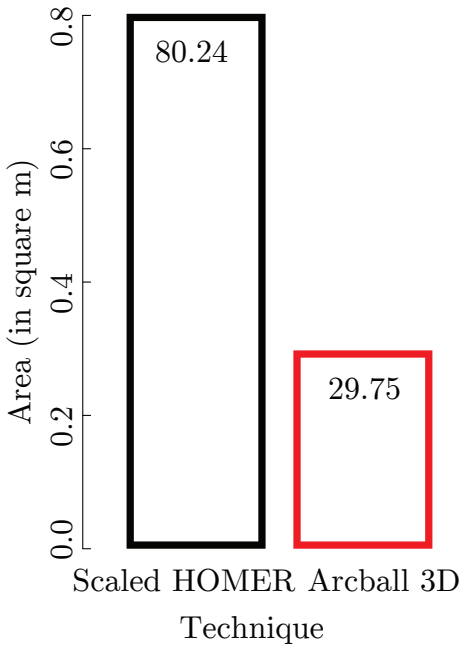


Figure 4.8: Total area covered by Arcball 3D vs Scaled HOMER. Y axis in m^2.

Because when each trial started the arm of the participant could have been at a different place, before looking at the area covered by the techniques, we first translated all trials on the (Y-Z) axis so that their first recorded point is located on (0,0), a form of normalization. We then calculated the area covered by the user's arm using the convex hull algorithm [12]. The convex hull of a planar set of vertices is the minimum-area convex polygon containing the planar set. This algorithm determines which points of a planar set are vertices of the convex hull. We used the method by Venables et al. [68] to calculate the area of the convex hull. In Arcball-3D the wand covered an area of $29.75cm^2$. In scaled HOMER, the area covered was $80.24cm^2$, more than *twice* the area of Arcball-3D (Figure 4.8). These results loosely correlate with the qualitative results reported by Katzakis et al. about fatigue.

Wrist speed across docking task

It has become commonplace for a 1-DOF or 2-DOF aimed motion to distinguish two distinct phases: A ballistic phase with a large, rapid motion, and a corrective phase where the user is making final adjustments before matching the target [76, 44]. Liu [44] proposed some measures for applying the same idea to 3D motions, i.e. dividing the 3D motion in ballistic and corrective phases. However, we were

interested in averaging all the trial profiles to see if the same phases can be detected for the overall resulting profile. Such a method could be applied to various interaction techniques and consistently provide a means for establishing whether or not these phases exist.

In Scaled HOMER the object is being manipulated directly so we expected some ballistic/corrective phases whereas in the ray-based techniques, large translations of the object can be performed by simply rotating the wrist, thus having little effect on the displacement of the tracker. It should be noted that Mesh-Grab and Arcball-3D both have a HOMER component so it's not unlikely that they exhibit some similar characteristics.

What follows is a methodology we followed for calculating the average speed of the wand across all trials in a set of docking trials of a technique:

1. We find the trial with the largest number of samples (i.e. trial with the longest duration) across all users and all positions. All other trials are to be re-sampled to match that number of samples. This decision was taken because it makes more sense to re-sample the shortest trials to match the longest one rather than re-sampling the longer trials to the shortest one, thus losing samples. The time-stamps of this longest trial will be used later on to interpolate the shortest ones.

2. For every docking trial l we calculate the speed of the tracker between two samples. s_x is the speed of the tracker. P_{t_x}, P_{t_y} and P_{t_z} are the x,y,z coordinates of the tracker at sample t while P_{t-1_x},P_{t-1_y},P_{t-1_z} are the x,y,z, coordinates at the previous sample. We start calculating speeds from the second sample of each trial and we assume a speed of zero on the first one (simple pythagorean theorem to calculate euclidean distance):

$$s_t = \frac{\sqrt{(P_{t_x} - P_{t-1_x})^2 + (P_{t_y} - P_{t-1_y})^2 + (P_{t_z} - P_{t-1_z})^2}}{time_t - time_{t-1}} \tag{4.1}$$

   This gives us a series of speeds/time-stamp pairs for that trial (yet note that at this point every trial has a different number of samples at different time intervals).

3. We normalize every single trial so that their duration is 1 (by dividing all time stamps by the last one). This gives us pairs for speeds/time-stamp where the times are from 0..1 and the speeds are the same speeds calculated before the normalisation.

4. All normalised trials are then linearly interpolated to find the speeds at the

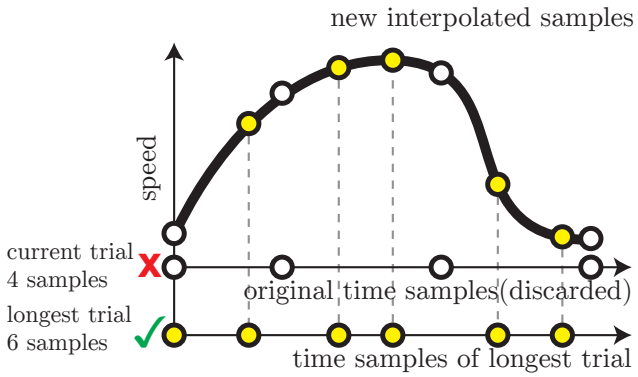


Figure 4.9: Interpolating a shorter trial to match the sample length and time-stamp of the longest trial. The old speed samples are dropped and new ones are interpolated based on the time-stamps of the longest trial.

sample times of the *longest trial* found in step 1 (Figure 4.9). Since there is a high sampling rate (120hz) linear interpolation does not affect the quality of the results. Since now all time-series have been re-sampled and intepolated, they all have the name number of speed samples at exactly the same points in time.

5. All normalised trials are then averaged to find the average speed for a certain time-stamp across ALL trials for this technique.

For the ray techniques, the exact same method was applied, for finding the rotation speed of the wands since Katzakis et al. also recorded the rotation vector of the wand at each sample. Between two different timestamps we calculated the angle of the tracker

This method was applied to the coordinate data from the experiment provided to us. The resulting speed profiles are plotted in figure 4.10.

The first thing to note is that speed of the wrist during scaled homer exceeded 6 cm/sec whereas speed of the tracker for the ray techniques remained under 0.5 cm/sec. This result also loosely correlates with the reduced fatigue reported by the participants.

Scaled HOMER: The scaled homer speed profile (Figure 4.11) does *not* exhibit a ballistic-correction phase like those reported by Liu. et al [44]. Based on observations of the docking trials the speed profiles can be understood as follows: In the beginning there is a small speed spike because participants make a rapid motion to select the target. Following that there is a phase where participants rotate the wand to match the desired rotation before finally translating the

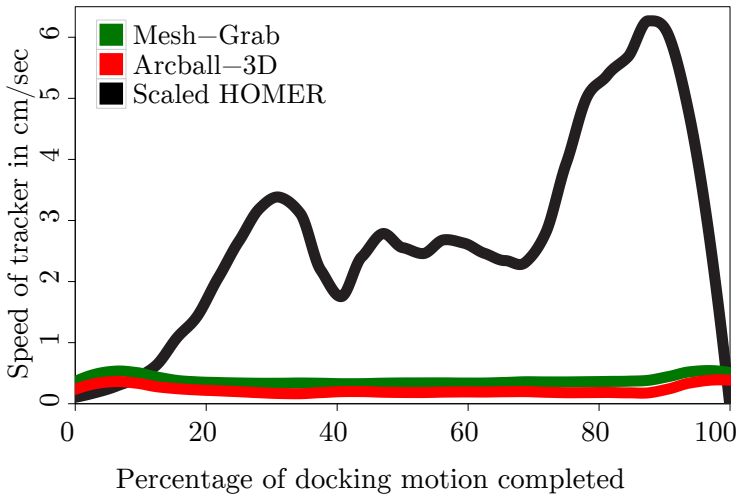


Figure 4.10: Speed profiles of all three techniques.

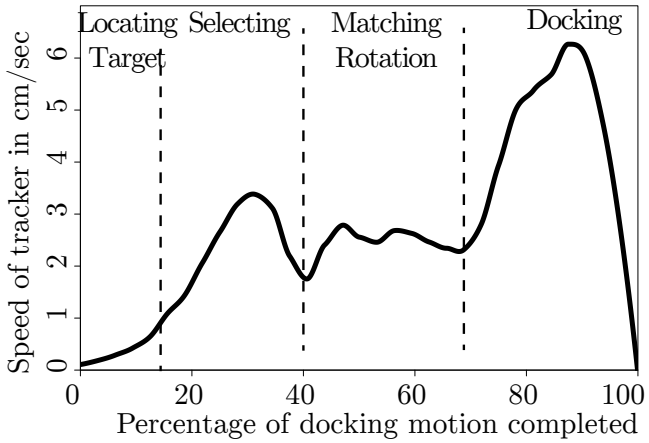


Figure 4.11: Speed profile for Scaled HOMER.

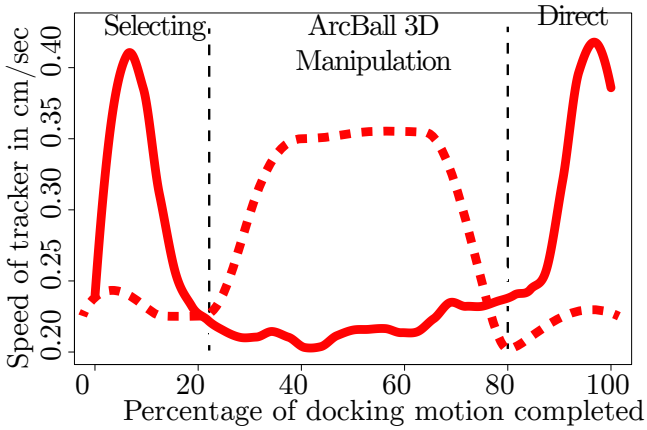


Figure 4.12: Speed profile for Arcball-3D.

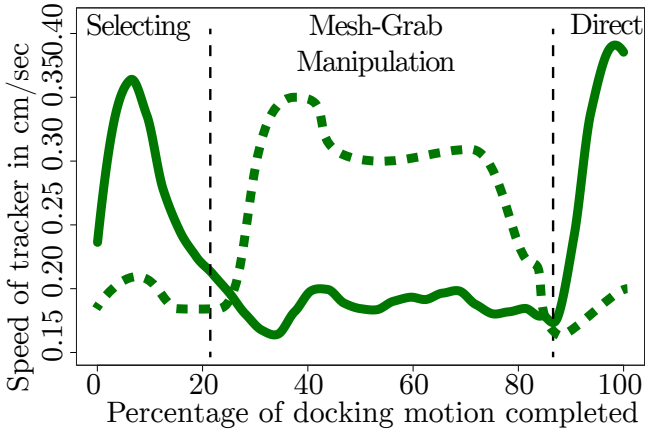


Figure 4.13: Speed profile for Mesh-Grab.

cursor into the target, where the last speed spike is seen. Based on Liu's finding one would expect to see a ballistic and corrective portion in that last docking phase with Scaled HOMER but there is none. A possible explanation for this could be Scaled HOMER's motion characteristics which adapt the speed of the object manipulated.

Arcball-3D and Mesh-Grab: The ray techniques, in addition to having much lower speeds also exhibited some distinct phases which also correlate with observations from the experiment. There is an initial speed spike to select the target, after which translation is handled by rotating the wrist, which is why speed decreases so much. Finally, we observed users switching to direct manipulation (HOMER) for the final part of the docking which is the final spike seen in the speed graphs (Figures 4.12 and 4.13). The rotation speed profiles (dotted lines) clearly show a ballistic phase where rotation of the ray was being used to translate the cursor.

So for the ray techniques, a ballistic phase was apparent from the rotation speed of the wand, but for the direct technique (scaled HOMER) the phases were not distinguishable.

4.6.1 Distance wrist moved

The distance the wrist travels throughout a single trial could be a good indication of the fatigue a technique imposes on the user. We analysed the average distance travelled by the wrist for every technique and found significant differences between techniques. Figure 4.14 shows the average distance the wrist moved across all trials for every technique. A within subjects ANOVA analysis showed a significant effect for *technique* on the measured variable *distance* ($F_{2,24} = 389.1$ $p < 0.001$). Post-Hoc tests (Holm's Sequentially Rejective Bonferroni Procedure) found significant differences. Arcball 3D 2.5cm < 20.6cm Scaled HOMER (t=20.2 p<0.001) and Mesh-Grab 2.6cm ¡ 20.6cm Scaled HOMER (t=19.49 p<0.001). No significant difference was found between Mesh-Grab and Arcball 3D.

In our previous work (Chapter 3) most participants commented on the arms fatigue and that it was easier to cast a ray from the hip than holding the tracker in front of the torso. Table 4.1 is a reproduction from their questionnaire data.

It is unlikely to be able to accurately correlate likert-scale questionnaire data with quantitative measurements yet our findings from the analyzed motion data

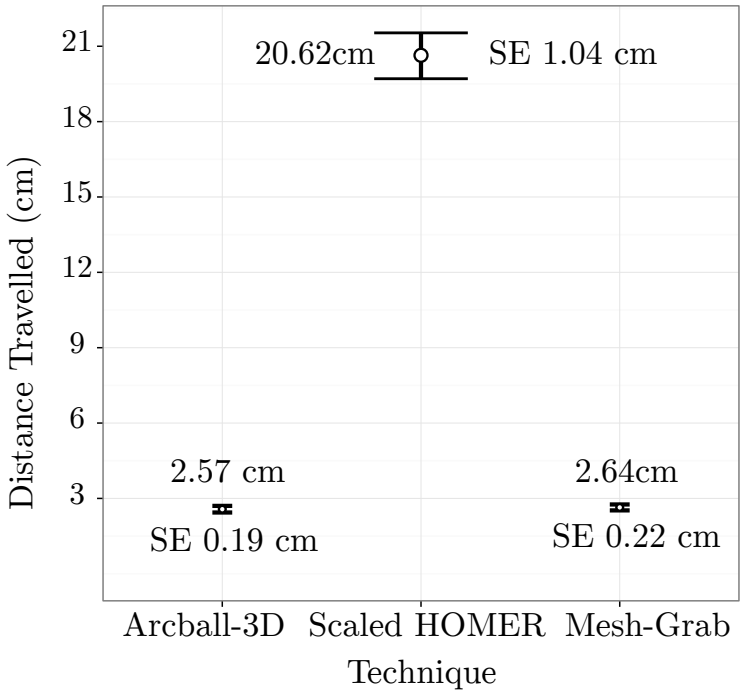


Figure 4.14: Average distance per trial the arm travelled for each technique with std.error.

Table 4.1: Summary of the questionnaire responses.

Technique	Mesh	Arc3D	HOMER	K-Wallis
Easy to use	3.77	5.15	4.38	($p < .05$)
Accurate	3.08	4.54	5.23	($p < .01$)
Tiring	4.15	3.46	5.31	($p < .05$)
Fun to use	3.92	5.31	5.31	($p < .05$)

loosely correlates with the responses from the questionnaires. Whether or not our analysis methodology is the most appropriate for trajectory data, remains to be validated, nevertheless this study makes some headway into understanding and quantifying.

Conclusion and Future Work

Summarising the contributions of this chapter: We proposed a methodology to average speeds of trajectory based tasks with different lengths. We applied this methodology to the results from a 6-DOF docking task and extracted some data that can be used to understand the differences between direct and ray-based techniques, especially regarding fatigue. The data from this study can be used to compare these techniques with other interaction techniques and thus gain some insight into their relative performance.

Although visual inspection of the speed profiles clearly shows the areas where speed spikes, a more rigorous analysis of the speed plots is necessary. Specifically, a set of mathematical tools that can be applied to a diverse set of interaction techniques must be identified and remains as future work.

Chapter 5

Unistroke: Facilitating Long Annotations on 3D Objects

5.1 Introduction

The ray techniques presented in chapter 3 established a way to manipulate a mesh in an integral way, with a smaller spatial footprint

Drawing on a 3D surface is important in computer graphics, virtual reality, computer aided design and entertainment or gaming. A 3D painting system should allow the user to paint on the target object in an efficient and intuitive way. The strokes that are performed to paint a 3D mesh are similar to 3D model sculpting, annotating etc. Drawing on a 3D surface is not as straightforward as on a 2D surface due to a number of challenges. Since the surface normal of the mesh can diverge from the view normal there are problems with consistent spacing of brush strokes. It is therefore desirable to paint with the viewpoint as much as possible close to the surface normal the mesh. For that purpose, it is often necessary to rotate the view.

In the case of long strokes, which is also the target of this study, there is often a disconnect, or a distortion when the user attempts to continue a stroke after rotating the view (figure 5.1). Our proposed technique, *Unistroke* attempts to allow long strokes on a mesh by automatically switching to rotation mode when

the stroke reaches the edges of the mesh. More details on this in the following section.

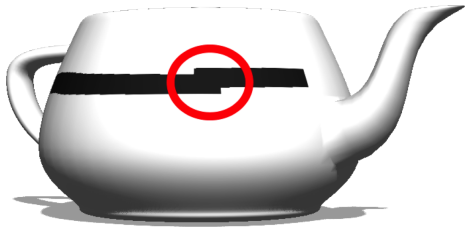


Figure 5.1: Attempting to resume the stroke after rotating the view. The continuation of the stroke ended up offset from the original path.

One of the problems in interactive mesh painting is how to map the "painted" segment on the 3D mesh's UV coordinates. To address this issue Igarashi et al. [33] presented a method to directly paint onto textures. They introduced methods to paint both sides of a mesh as well as to limit painting to certain areas when an abrupt change in geometry is detected.

Fu et al. [18] presented layerpaint a painting tool that is not limited to the front-most visible surface on the screen. Layerpaint allows the users to efficiently and interactively draw long strokes across different depth layers. Fu's evaluation, however, was done using an indirect stylus tablet (drawing surface is different from the display area) so the results of their evaluation are difficult to extend to tablet computers and touch.

Ortega et al. [51] presented ACCD a system for automatic camera control while drawing on 3D meshes. ACCD overcomes the need to rotate the view by keeping the camera close to the mesh surface normal as the stroke moves along the mesh. This approach is very useful for curved surfaces but because the ACCD algorithm needs to always raycast from the mouse point on the mesh it does not work consider sharp angle changes (like box edges) (figure 5.2).

In addition to the problem of falling off edges when ACCD is used against a mesh that is uneven camera motion is erratic and this causes the model to rotate in unpredictable ways. Depending on the mesh this behaviour could be unpleasant to the user.

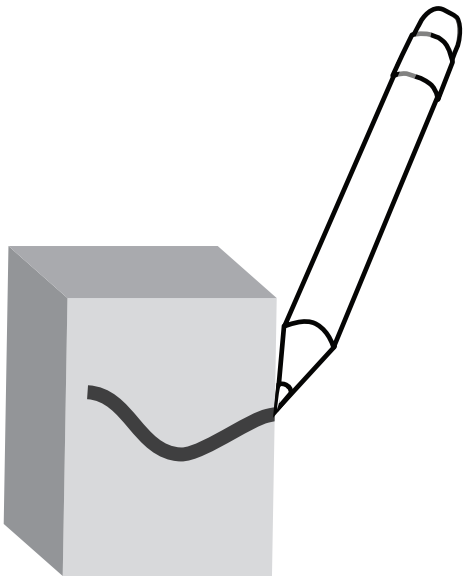


Figure 5.2: One of the cases not handled by the ACCD algorithm. When the stroke reaches an edge, it would fall of the edge and stop painting since there is no more mesh left for calculating the next camera position.

5.2 Unistroke

Unistroke attempts to allow long strokes on a mesh by automatically switching to an "arcball" rotation mode when the stroke reaches the edge of the mesh. The user then can rotate the mesh without lifting his finger/stylus. As soon as the desired viewpoint is reached the user hovers for a few seconds and can then continue the stroke. At first this solution might seem straightforward, however, there are a number of challenges involved. In order to keep the stroke smooth, during virtual arcball rotation the point at which the stroke exited the mesh must remain under the finger when the finger returns to rotate in the opposite direction. We thus switch to modes during the following three conditions:

- When the stroke falls off the mesh, the last point painted on the mesh becomes the rotation point (figure 5.3a)
- When there is an abrupt change in surface normal between the point painted and the last point panted (figure 5.3b)
- When there is an abrupt change in depth (figure 5.3c)

The algorithm that determines when to switch modes is straightforward (figure 5.4). For the case when the stroke falls out of the mesh the system casts a ray

(a) switching to rotation mode when the stroke reaches the edge of the mesh.

(b) A case where the normal direction does not change much, but the depth does

(c) switching to rotation mode when the stroke reaches a point where the normal changes abruptly

Figure 5.3: Situations when unistroke automatically switches mode to rotation.

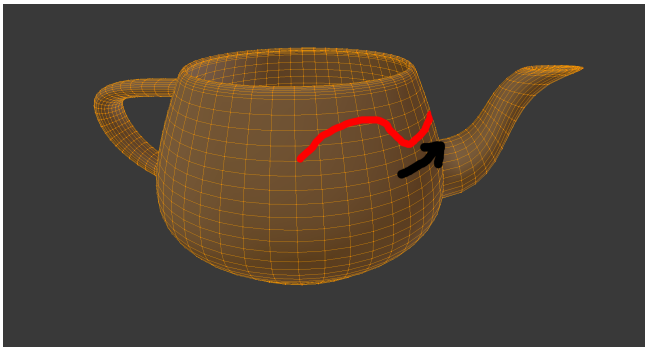
from the stylus to the mesh every time the stylus moves. If the raycast does not hit the mesh, then the last hit point is kept and that point is used as the basis for the arcball rotation algorithm.

For the case of a normal change (figure 5.4b) if the normal between the previous triangle and the current triangle passes a certain threshold, the system considers that a switching point.

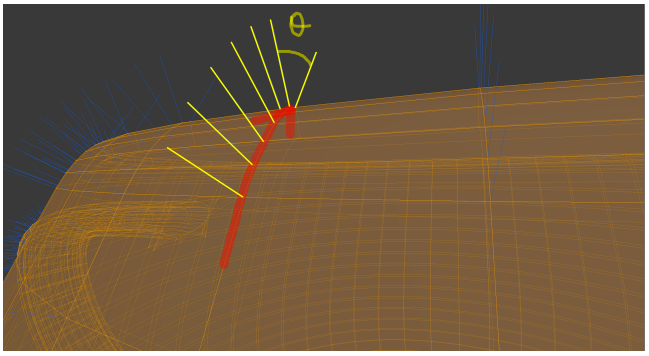
Unistroke allowed users to rotate the model by making strokes with their non-dominant hand similar to INSPECT (chapter 7). A two axis valuator for the x-y axes and two fingers pivoted about their center to rotate about the Z axis.

5.3 Evaluation

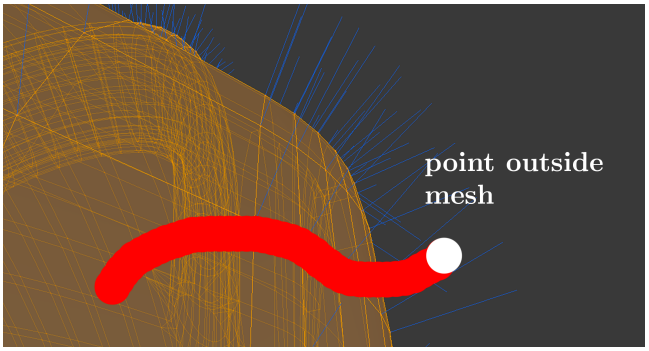
Our hypothesis was that because ACCD moves the camera around, it would be easier for users to draw a simple drawing on the side of a mesh using unistroke. That would reflect in their accuracy, as ACCD moves the camera around automatically. We thus asked users to trace two simple drawings on the side of a mesh. One of the drawings (figure 5.5a) had a house drawn across two sides of the teapot (to force users to rotate the view) while the other, a bunny (figure 5.5b)



(a) Making a stroke on the mesh



(b) A case where the normal direction changes significantly



(c) A case where the stroke falls outside the mesh.

Figure 5.4: Determining mode changes.

(a) Image to be traced during first task

(b) Image to be traced during second task

Figure 5.5: Situations when unistroke automatically switches mode to rotation.

5.3.1 Participants

16 participants took part in the study (within-subjects). 10 participants were male, 6 participants were female. Their ages ranged from 23 to 32 years (mean age 28 years). All participants were right handed. Participants reported little to no stylus/tablet experience.

5.3.2 Apparatus

Hardware Setup

The experiment was conducted using a computer running Mac OS X with a 2.9 GHz processor and 8 GB of RAM. The graphics card was an Intel HD Graphics 4000. A Wacom CINTIQ 13HD Touch was used as the display. Participants sat at a desk with the tablet tilted at a 35 degree angle. Participants held the stylus in their dominant hand, while touch-gesturing on the cintiq touchscreen with their non-dominant hand.

Software Setup

The experiment used custom software running on the PC, which was written in C++ and OpenGL 3.3. The software presented the experimental tasks described below. In both tasks, the scene was displayed with the target object floating in the middle of the screen against a dark background (figure 5.6). The software logged the trial length, the stylus cursor position and the touch points as well as other relevant metrics.

5.3.3 Procedure

Upon arrival to the lab, participants were given a short briefing about the experiment. This included a verbal explanation of the experiment purpose, the tasks, and a description of the techniques being compared in each task (described in detail below). Following the briefing, participants were asked to perform both the tracing task.

A white teapot appeared at the center of the screen at an upright orientation (figure 5.5b). Users were free to rotate before they started the trace but as soon as any action was performed, the timer began timing the user and the task was considered started.

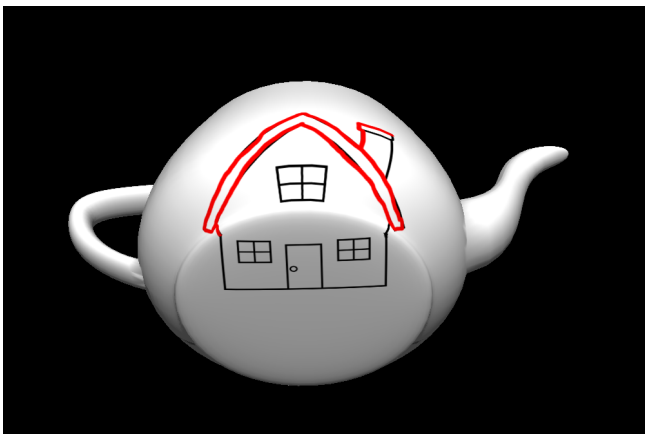


Figure 5.6: Screenshot of the experiment. A user is tracing the drawing

The dependent variables for this task were stroke distance (D), measured in mm, and completion time (CT) measured in seconds. D served as a measure of accuracy. Completion time was measured as the time from when the trial began to the time the participant pressed the foot pedal to indicate task completion.

5.3.4 Hypotheses

(h1) Unistroke will be more accurate than ACCD. We believe that camera movements will have an effect on the accuracy of participants **(h2)** Since ACCD allows for camera manipulation without lifting the pen, users will complete the task with less strokes using ACCD.

5.4 Results

The results were analyzed with an ANOVA statistical test. As predicted by our hypothesis, tracing without automatic camera control was significantly more accurate ($F_{1,15} = 278.8$, $p < 0.05$) than ACCD (figure 5.7). As predicted by the second hypothesis, users completed the task with less strokes ($F_{1,15} = 117.2$, $p < 0.05$) and with significantly quicker times ($F_{1,15} = 155.7$, $p < 0.05$) using ACCD (figure 5.8 - 5.9).

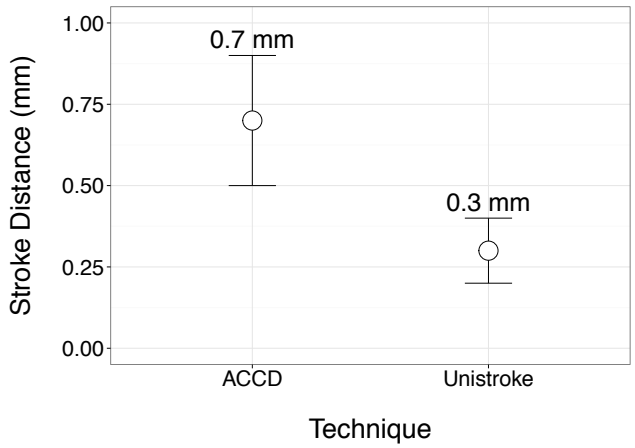


Figure 5.7: Users were significantly more accurate using unistroke (results are statistically significant).

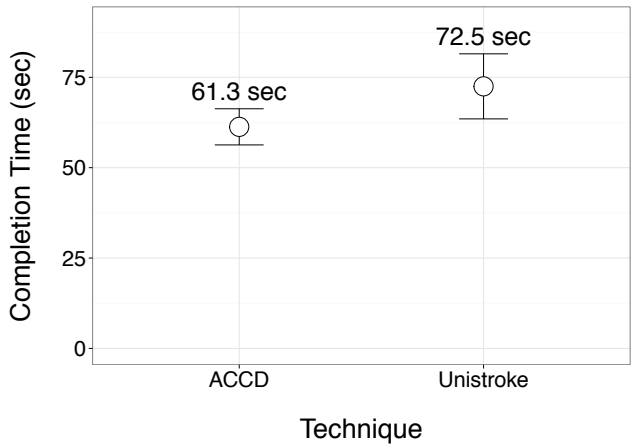


Figure 5.8: Users were significantly quicker using ACCD (results are statistically significant).

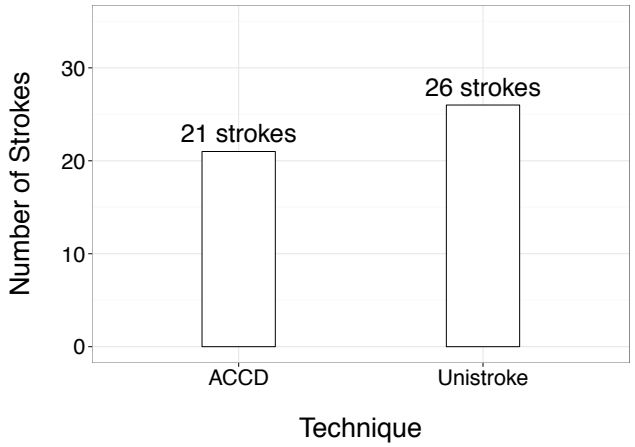


Figure 5.9: Users used less strokes with ACCD (results are statistically significant).

Figure 5.10: Drawing near the edges of a mesh could lead to false positives and switch the system to rotation mode.

5.5 Discussion

Unistroke has its limitations. Since our algorithm looks forward this makes it challenging to paint near the edges of a mesh (the system might accidentally switch to rotation mode). This is illustrated in figure 5.10.

This might also lead to problems when an uneven mesh is being panted, such as a mesh scanned using a 3D scanner (kinect fusion etc.). Such a mesh would have many holes, crevices and bumps which would result in many accidental mode changes.

Another limitation of unistroke is that rotation always happens about the origin

of the model (classic arcball). This rotation might not be a problem for drawing around cylindrical meshes such as a bottle or a vase, but for more complicated meshes the user might want to rotate differently. This could be solved by pressing a finger to define a new rotation pivot point with the off-hand. That sort of function though would require multi-touch which is not universally available in all touch systems. Which leads to another limitation that should be mentioned. Unistroke assumes the input device can differentiate between touch and stylus input. Most of these devices have some rudimentary palm rejection, however it is sometimes erroneous and as such the palm contact point could be mistaken for a touch or considered the pivot point for rotation.

These issues should be considered when designing systems that employ unistroke for rotation of the mesh.

Finally, although unistroke was envisioned as a method to annotate a mesh during a presentation, since it's a technique that requires the presenter to look at the tablet to annotate that would create a disconnect with the audience which relies on the gaze of the presenter for guidance.

The ideal painting technique is one that combines ideas from chameleon [33], ACCD [51] and unistroke and these techniques could all be used complementary depending on the situation.

5.6 Conclusion

We have presented unistroke. A novel method for making long strokes on 3D models from a single touch input.With additive manufacturing gaining popularity there is an increasing motivation to facilitate painting on 3D objects from simple inputs so that users can paint using touch screens and styli on tablets.

Finding ways to make mode changes more intuitive without relying on arbitrary thresholds remains a target for future work.

Chapter 6

Plane-Casting: 3D Cursor Translation using a Smartphone

6.1 Introduction

As computer graphics technology advanced, and display sizes grew, it became possible to view rich computer graphics from a distance, on a large display. With this the need to interact also ensued. Examples of situations where there is a need to interact in 3D from a distance include the following:

- A team of artists in a game studio is reviewing the latest 3D assets in their weekly meeting on a large display. Participants interact, review and discuss changes relating to the 3D models.
- A medical school professor is demonstrating the anatomy of the human heart by projecting 3D graphics on a large screen (Figure 1.2). Such a controller, that does not rely on external instrumentation for tracking, allows the professor to leave the podium and approach the students while still being able to interact with the model, thus making the class more engaging. Students can also use their smartphones to actively participate.
- A group of children interact with a virtual exhibit in a museum setting. They assemble a 3D puzzle of a fragmented archaeological find much like

an archaeologist would do and learn about the process.

In all the aforementioned situations, users need to position a 3D object easily, without being tethered by cables or confined to a small working volume because of the tracker. Users also need to interact in a fatigue free position that will not make their arms tired after a short period of time. Currently available methods for remote 3D control tend to lack in expressiveness, intuitiveness and are cumbersome and fatiguing to use.

As a solution to some of these issues, in this work we propose a novel technique for 3D positioning using a smartphone. Smartphones feature an array of orientation sensors which make it possible to calculate the device's orientation in 3D. By further employing 2 degrees of freedom (DOF) from the touch screen we demonstrate that with our proposed technique, Plane-Casting, it is possible to translate 3D cursor in space. The core idea of Plane-Casting is that the rotation of the smartphone controls a virtual plane that constrains the movement of the 3D cursor. Further to the potential for efficient 3D control, the wide availability of smartphones is an additional motivating factor for this work.

In the remainder of this paper we present two variations of Plane-Casting, *Pivot* Plane-Casting and *Free* Plane-Casting. We discuss their strengths and limitations and present results from a pilot study.

6.2 Related Work

Aside from the lack of head/viewpoint tracking, interacting with a 3D object being displayed on a large screen is not much different from interacting with a 3D object in immersive virtual reality (VR). There is a large body of work on manipulating objects from a distance in immersive Virtual Reality (VR) [53, 55, 66, 73]. Much of this work focuses on selecting and positioning remote objects. In applications where a 3D model is being demonstrated to an audience, however, the user would most likely need to interact in a limited depth, a form of fish tank VR or "*Shallow depth interaction*"[24].

As an alternative to wands and gloves, touch-input technology enables users to interact with digital contents by removing an indirection layer, and there are quite a few solutions for 3D interaction by using multi-touch.

Cohe et al. [11] introduced tBox, a 3D manipulation widget for touch screens and found that users could easily assemble a 3D object with it. Martinet et

al.[45] introduced some techniques for direct 3D object manipulations based on bimanual inputs.

Despite the existence of direct-touch based solutions, when the display size exceeds a certain threshold, however, touch input ceases to be an option as the user needs to cover a large area with physical movements while in some cases the display is out of reach (as is the case with projectors and tiled displays). In addition to the input problems of touch, physically approaching the display to interact limits the user's activity to a small area of the display, and the interacting user obscures the display for the rest of the group in the case of collaborative work.

Attempting to address some of the limitations of direct-touch input, freehand/gesture approaches have been proposed.

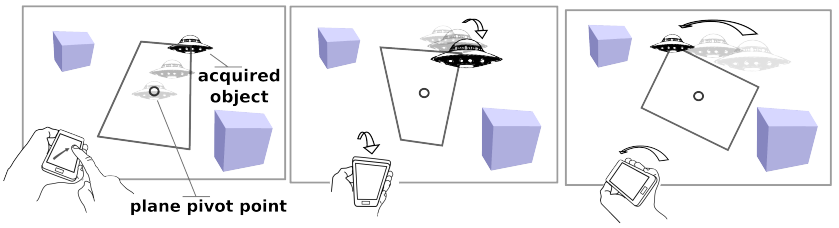
The Nintendo Wii-mote^{TM} is a popular choice for remote control but it depends on a two-state directional-pad for additional degrees of freedom. Other controllers like the 3Dconnexion SpaceNavigator^{TM} depend on desks and are tethered by cables, thus making them unsuitable for an active, engaging experience or for use in public or shared spaces.

Some other studies have looked at smartphone based 3D interaction:

Hachet et al.[22] propose a controller that attaches to the side of mobile phones to provide 3-DOF control, a solution which could be used for remote 3D control. They evaluate the design in a navigation scenario and report positive reactions from the users. Their approach is, however, based on proprietary hardware external to the device, and limited to rate control.

More recently, Jimenez et al.[36] use a hand-held device in a museum scenario for remote assembly of a puzzle-like task. Their work highlights some of the social aspects of using a hand-held interface in a collaborative task. Their evaluation suggests that a usable interface might better promote equal participation in a group task.

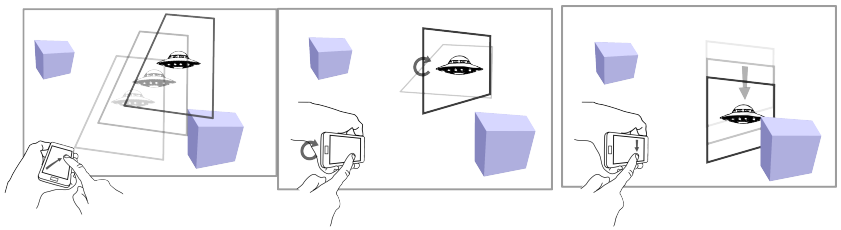
Finally, Song et al.[64] use a hand-held device in a large-display scenario. The device controls the position of a slicing plane in 5-DOF that explores volume rendering data and the authors present a novel technique to annotate them. Their approach unfortunately requires physical proximity to the screen which makes it unsuitable for remote or collaborative work and also depends on proprietary hardware attached to the hand-held device thus limiting it's applicability. Their paper offers a thorough review of the literature on hand-held/remote in-



(a) Pivot PC: The user gestures to translate the acquired object which moves on the plane.

(b) Plane follows rotation of device. Rotates about the pivot with the object bound to it.

(c) The object's rotation does not change, only its position does.



(d) Free PC: The acquired object moves on the plane but the plane stays attached to it.

(e) The pivot point of the plane is always fixed at the center of the object's bounding box.

(f) Free Plane-Casting can be thought of as a means to track the finger's relative motion to the display.

Figure 6.1: The two variations of Plane-Casting. Pivot Plane-Casting (6.1a,6.1b,6.1c) and Free Plane-Casting (6.1d,6.1e,6.1f)

.

teraction.

Although integral control of a plane used to constrain motion is unique to Plane-Casting, Bier's discussion on constraining motion in a scene composition scenario is one of the earliest references in the literature [5]. Bier further emphasizes the power of constraint-based systems in subsequent studies.

6.3 Plane-Casting

Pivot Plane-Casting

In *Pivot Plane-Casting* (Pivot PC) the shape of the touch screen is drawn as a rectangle at the center of the scene (Figure 6.1a). The user can rotate the device

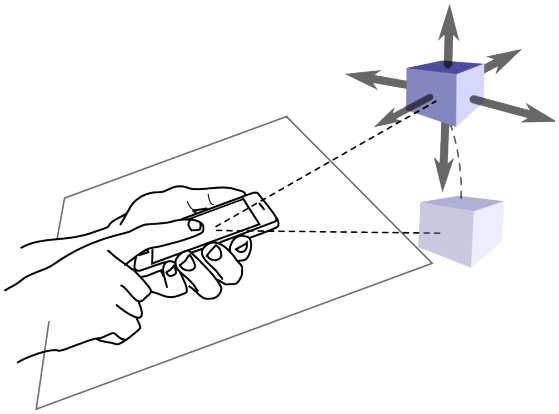


Figure 6.2: Pivot PC, moving vertically to the plane becomes easier as the object moves away from the pivot point of the plane.

to control the orientation of the plane (position-controlled). The plane's pivot point is at the center of the rectangle and always remains fixed at the center of the virtual space. The 3D cursor's movement is constrained on the plane defined by the rectangle but not limited to it's bounds. The user can translate the cursor on the plane by gesturing on the touch screen and rotate the plane to move the cursor to any point in 3D space (Figure 6.1b). In our implementation the tactile-sensor is a touch screen but any touch panel that can be tracked is suitable for Plane-Casting.

Free Plane-Casting

Free Plane-Casting (Free PC) is similar to Pivot PC but in this variation the plane's pivot point follows the cursor's motion in 3D space. Free PC shifts the center/pivot point around with every slide movement. The rectangle that defines the plane is thus always attached to the cursor that is being manipulated and they move as one (Figure 6.1d), with the orientation of the rectangle constantly re-defining the plane (Figure 6.1e).

In Pivot PC, placing the cursor away from the pivot point of the plane makes it easier to translate the object vertically, on the normal to the current plane thus making it easier to quickly change direction as would be the case in a game (Figure 6.2), yet by sacrificing accuracy. Due to Free PC's nature, only 2-DOF are instantly available at any time and moving in a direction on the normal to the current plane requires a supination/pronation movement (Figure 6.1e).

In our implementation of Free PC and Pivot PC selection of the object to be manipulated is done by a spherical cursor that intersects the desired object in a widely used "virtual hand" metaphor. Depending on the application, there are many strategies for object selection but that remains beyond the scope of this work.

6.4 Evaluation

Our pilot study evaluated the two techniques against each other in a 3D positioning task. A remote manipulation scenario like the ones mentioned in the introduction would require the ability to move an object in 3D, but which technique would be better for this? We wanted to test the following two hypotheses:

- H1 : Pivot PC will perform faster than Free PC since it only requires an initial alignment of the plane to the target.

- H2 : Free PC will be more accurate as it's essentially bringing the pivot point closer to the target and does not require a "steady hand" like Pivot PC.

12 right-handed male participants, students and faculty volunteered for the experiment (age mean 25). Participants had no prior experience in using the techniques.

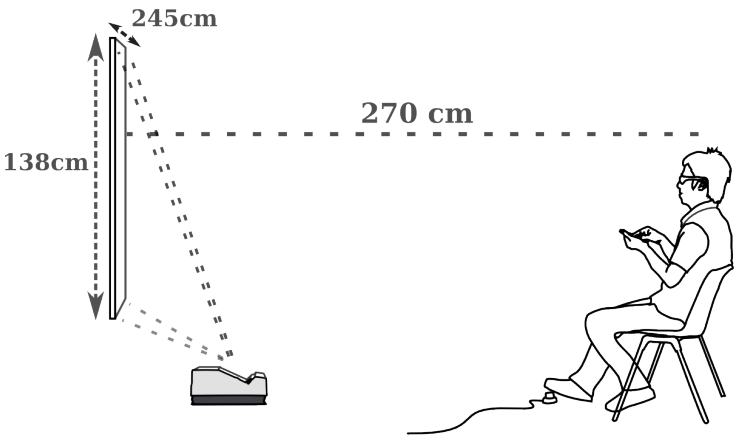


Figure 6.3: Illustration of the experimental setup.

6.4.1 Set-up

Subjects sat 270 cm from the projection screen of an ultra-short-focus projector (Sanyo PDG-DWL2500J). They were instructed to hold the smartphone (Samsung Galaxy SII) in their non-dominant hand while gesturing on the touch screen with their dominant hand (Figure 6.3). A foot switch was provided for advancing to the next trial. The projection screen had a width and height of 245 x138cm respectively with a 1280 x 800 display resolution in stereo (NVIDIA 3D Vision). Data transmission of the device orientation (400hz) was over an IEEE 802.11g WiFi link and was filtered with a 30 sample moving average filter for stabilization (see appendix A for more details).

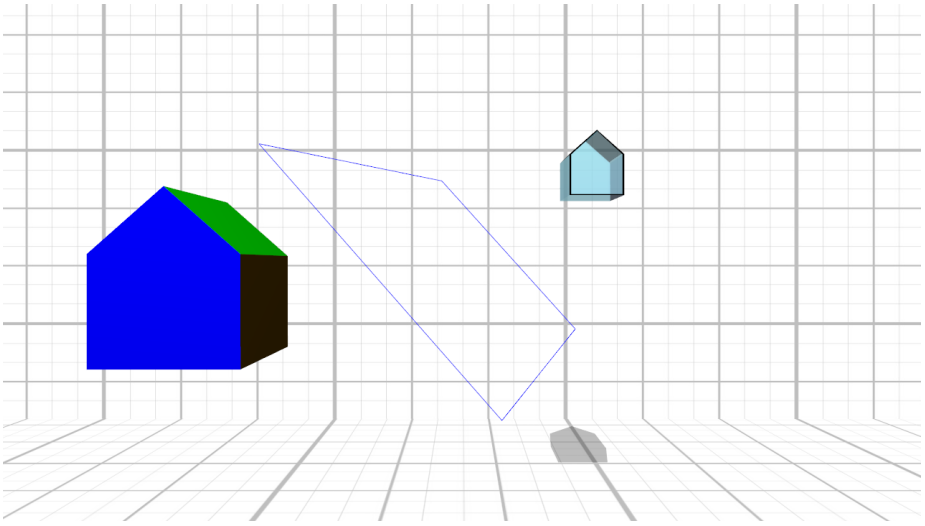


Figure 6.4: Screenshot of the evaluation task using Pivot PC. Participants had to dock the cursor (multi-colored-house) to the translucent target.

6.4.2 Task

In the evaluation task, the house-shaped cursor (and rectangle) appeared between the viewpoint and the far wall of the 3D space (Figure 6.4). When the experiment commenced, a translucent copy of the cursor appeared randomly at one of 12 pre-defined positions around the cursor (Figure 6.5) and subjects had to match the position of the cursor with that of the target under two conditions: 1) as quickly and 2) as accurately as possible. The targets appeared in positions distributed evenly on the surface of a sphere centered at the cursor's starting position with a radius of either 52 or 96 cm respectively (Figure 6.5). Each position was tested twice with each technique, one in the *Speed* and one in the

Accuracy condition (counterbalanced order).

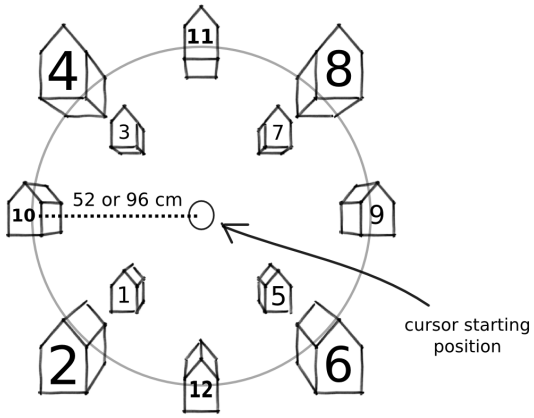


Figure 6.5: Layout of the targets. 12 positions evenly distributed on a sphere around the cursor starting position with 4 of them axis-aligned. Targets 1-8 are pivoted 45° about the Y and Z axes. The two missing front and back axis-aligned positions were not tested since they would occlude or be occluded by the cursor and thus slightly confuse participants.

When the cursor's bounding box intersected the target's bounding box, the target's bounding box would become visible signaling a match. Subjects could not end the trial if they did not have a match. When subjects felt they had achieved a good match, they pressed the foot switch and the trial ended with both the cursor and the target disappearing. Only the rectangle remained. In Free PC, the pivot point of the plane/rectangle would return to the center of the scene. The next trial would only begin when subjects returned the device to it's original orientation parallel to the ground at which point the cursor would re-appear at the center of the plane, and the target would appear at the next position to be tested.

All subjects received a brief explanation of the techniques and performed the task once with each technique as practice (order of techniques was also countebalanced). Subjects performed 12 positions $\times$ 2 radii $\times$ 2 techniques $\times$ 2 (speed vs. accuracy) = 96 trials (1152 total). The experiment lasted around 40 minutes per participant with no break between conditions.

6.5 Result and Discussion

We recorded the movement time (MT)[1], the accuracy as a measure of the euclidean distance (d) between the center of the cursor's bounding box and the target's bounding box at the time the subject pressed the foot switch. We also recorded the distance the participant's finger traveled on the touch screen (T)[2] during every trial.

6.5.1 Technique Effect

Results of a repeated-measures ANOVA showed that *Technique* had a strong effect on movement time (MT). The average MT for Free PC was 9.7 sec. vs 8.2 sec. for Pivot PC ($F_{(1,11)}$=17.1, p<.001). This confirms our first hypothesis (H1) of Pivot PC being the quicker technique of the two, though based on the fact that Free PC requires repeated supination/pronation we expected the performance difference to be greater.

Technique also had a significant effect on T, the amount participants gestured on the touch screen. An average amount of 190 pixels traveled was recorded when using Pivot PC whereas when using Free PC users gestured an average of 225 pixels ($F_{(1,11)}$=23.6, p<.001). These results suggest that Pivot PC is potentially suitable to implement on devices with smaller touch surfaces than Free PC.

Contrary to our second hypothesis (H2), neither technique was found to be more accurate (d=6.5 for FixedPC vs 6.8 for Free PC $F_{(1,11)}$=0.3, p=0.58).

6.5.2 Target Distance Effect

Whether the target was placed in the near radius 52 cm or the far radius 96 cm had no significant effect on the time cursors took to reach the target ($F_{(1,11)}$=3.2, p<0.09) with 8.9 sec. to reach the far ones vs. 8.7 sec. to reach the near ones.

There was also no effect on the amount users gestured on the touch screen between radii ($F_{(1,11)}$=1.5, p=0.2) with participants gesturing 203 px for the

[1]Movement time started counting either when participants rotated the device past a 2% threshold - thus signaling their attempt to align the plane to the target - or when they first swiped on the touch screen, whichever came first.

[2]The T measurement is the result of the total distance in pixels the finger traveled on the touch screen while pressed down during the trial.

near targets vs 210 px for the far targets.

Finally the distance of the target had no effect on accuracy as users were equally accurate on near and far targets ($F_{(1,11)}$=0.4, p=0.5).

Accuracy/Speed Tradeoff

As would be expected, there was a strong effect on movement time and accuracy by the speed/accuracy condition. As such when *MT* is concerned users had an average of 5.2 sec. in the *Speed* condition vs 12.7 sec. in the *Accuracy* condition ($F_{(1,11)}$=115.8 p<0.001). Similarly accuracy (d) was 5.6 in the *Accuracy* condition vs 8.7 in the *Speed* condition ($F_{(1,11)}$=27 p<0.001).

6.5.3 Target Position Effect

Target position had a strong effect on *MT*. Particularly positions 9 and 10 which are horizontally aligned to the starting position were the fastest in both techniques, as expected (Figure 6.6-6.7).

In Free PC (Figure 6.6) there was a relatively even distribution of movement times between the various positions. In Pivot PC (Figure 6.7), however, participants struggled with position 8 (mean time 15.5 sec., - $F_{(11,121)}$=3.8, p<0.01).

Observations of the subjects indicate that the reason for the problem in position 8 was the limited ulnar and radial flexion of the human arm which makes it slightly strenuous to reach that position. One of the strengths of Pivot PC is that there are many combinations of touch and rotation to reach the same position in 3D space. That is also part of it's weakness as it requires training to find which is the best combination of tilt/swiping to achieve the desired motion, something which, however, remains to be validated.

Finally, position of the target had no significant effect on accuracy ($F_{(11,121)}$=1.5, p=0.12) as participants were equally accurate in all positions.

6.5.4 Qualitative Results

Participants answered a post-experimental questionnaire asking them to rate Pivot PC and Free PC in terms of their intuitiveness and physical demands. 10/12 participants chose Free Plane-Casting over Pivot Plane-Casting both in

intuitiveness and in physical demands. This is despite the overall quantitative performance was better in Pivot PC for this specific task. Subjects also commented that in Free PC, since the cursor can only move when gesturing, there is less "pressure" to keep the device aligned with the target (as is required in Pivot PC - Figure 6.2) and that is cognitively (and physically) less demanding.

6.6 Conclusion and Future Work

We have introduced two variations of a novel technique for 3D cursor manipulation using a smartphone. Our pilot study verifies their usability and highlights some of the issues associated with each one. For the docking task Pivot PC seems to be the quantitative overall winner, but participants preferred Free PC. Further evaluation is required to ascertain their applicability to real-life scenarios. This work establishes a broad base upon which more specific Plane-Casting based 3D applications can be built.

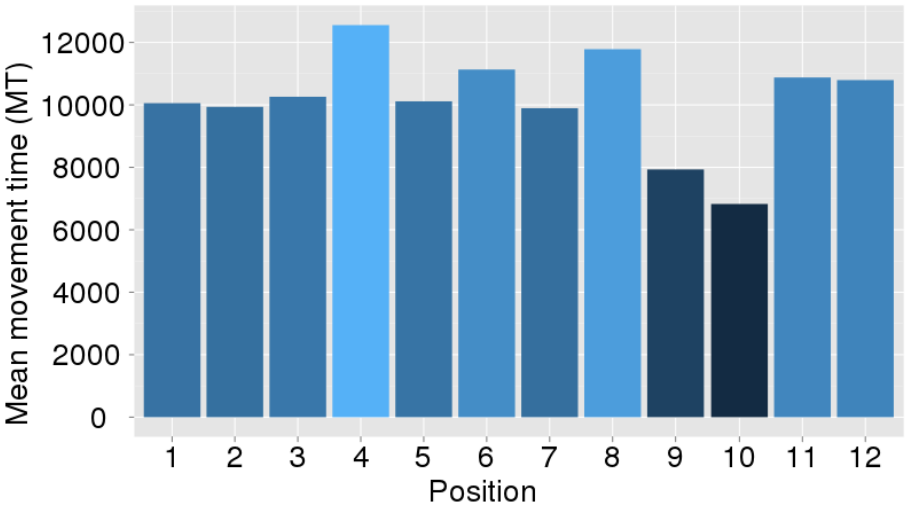


Figure 6.6: Free PC: Mean movement time for every target position (Figure 6.5).

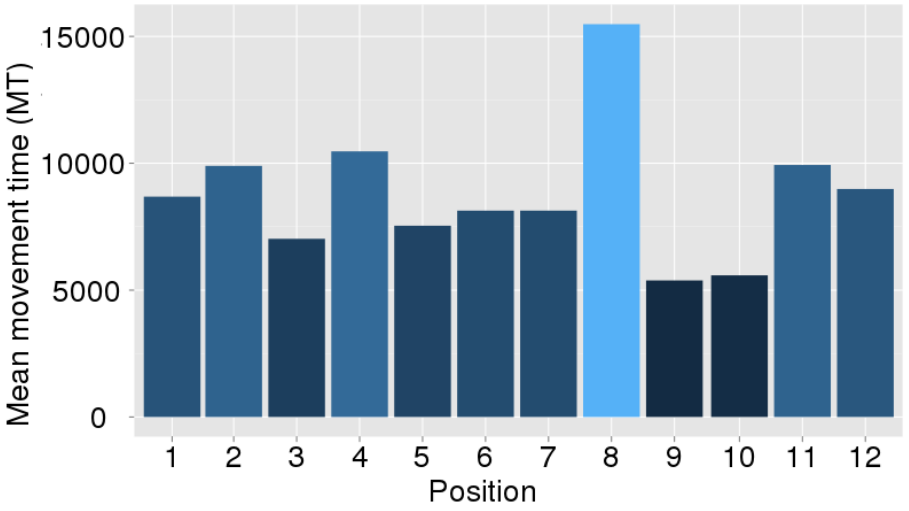


Figure 6.7: Pivot PC: Mean movement time for every target position. Position 8 (Figure 6.5) was significantly slower to reach (Y axis in seconds).

Chapter 7

INSPECT: Extending Plane-Casting for 6-DOF Manipulation

7.1 Introduction

Virtual object manipulation is required in a wide variety of application domains. From city planning [52] and CAD in immersive virtual reality [71], interior design [66] reality and virtual prototyping [28], to manipulating a multi-dimensional dataset for scientific data exploration [41], educational applications [67], medical training [62] and even sandtray therapy [26]. There is, in all these application domains, a demand for low-cost, intuitive and fatigue-free control of six degrees of freedom (DOF). Our work focuses on 6-DOF object manipulation at a distance with a large display for presentations and education. Examples of situations where there is a need to interact in 3D from a distance include the following:

Education: A professor is demonstrating human anatomy by displaying 3D graphics on a large projector screen. He uses his device to rotate the model and answer questions from the students. The nature of the device allows him to leave the podium and approach the students while still being able to interact with the model, thus making the class more engaging.

Engineering: An engineer is showing a 3D model of her latest design to team-

mates. The device is used to rotate and translate the model, define slicing planes to inspect the interior and discuss the design with other participants.

Entertainment: A group of children are playing a game in a museum while at the same time learning about physics by interacting with wooden blocks on a sandbox-like 3D environment on a large screen.

This work builds on Plane-Casting (Chapter 6) for 3D translation using a tracked touch-panel, motivated by its applicability on smartphones. Plane-Casting offers isotonic position control *without* using any external position trackers, save for the orientation sensors in the device. We extend Plane-Casting and introduce INSPECT, a set of novel interaction techniques for off-screen virtual object manipulation using a smartphone. INSPECT stands for **IN**direct **S**ix-DOF **P**lan**E** **C**ontrol **T**echnique, and it was designed for the purpose of inspecting a 3D object. We demonstrate that by using INSPECT it is possible, with a low-cost mode change, to perform 6-DOF virtual object selection and manipulation using a 3-DOF orientation-tracked touch panel and the 2-DOF per finger from the touch points.

The wide availability of smartphones and smartwatches motivates the need to explore the *indirect touch* design space and to identify an appropriate way to map the degrees of freedom afforded. We also evaluate INSPECT in a 3D movement task and a 3D rotation task, against a 'gold standard' direct technique with a magnetic tracker, to serve as baseline reference.

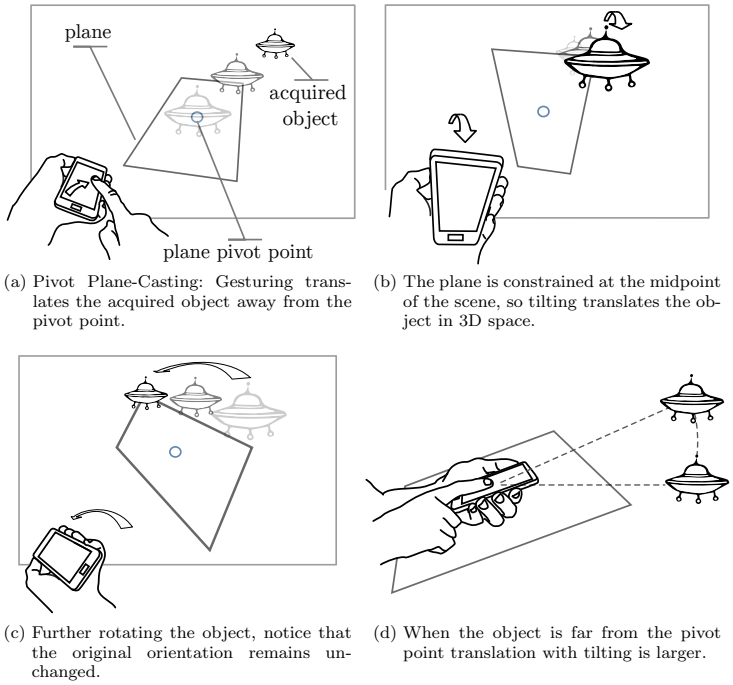
Please refer to chapter 2 for related work.

7.2 Proposed Techniques

7.2.1 Plane-Casting

The original Plane-Casting technique [38] supported 3D positioning of a preselected object. The core idea of *plane-casting* was that the manipulated object was free to move in 2D along a plane that was freely oriented in 3D space by the user. Two variants of Plane-Casting were proposed:

In the first variant, *Pivot* Plane-Casting, the plane rotated about a pivot point located in the center of the 3D space. The orientation of the smartphone controlled the orientation of the movement plane about the pivot point. Swiping on the display translated the object in the corresponding axis on the movement



(a) Pivot Plane-Casting: Gesturing translates the acquired object away from the pivot point.

(b) The plane is constrained at the midpoint of the scene, so tilting translates the object in 3D space.

(c) Further rotating the object, notice that the original orientation remains unchanged.

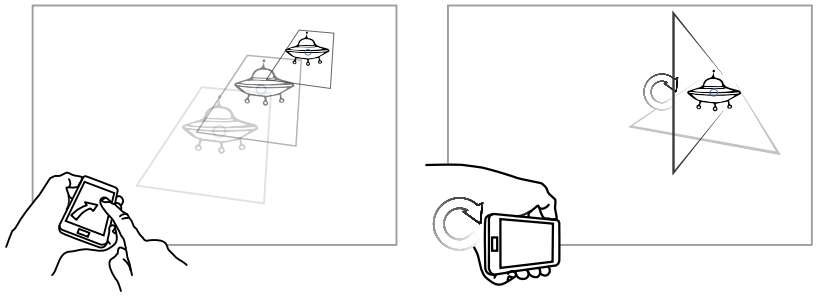
(d) When the object is far from the pivot point translation with tilting is larger.

Figure 7.1: Pivot Plane-Casting.

plane. Thus, by translating the object on the plane away from the pivot point and then rotating the plane, the object could be positioned at any point (Figure 7.1). A disadvantage of Pivot Plane-Casting is that it requires a "clutch" button to disable plane rotation. Without this capability, users would have to always hold the device at a fixed orientation to stabilize the object's position. They would thus be unable to relax their non-dominant hand (Figure 7.1d).

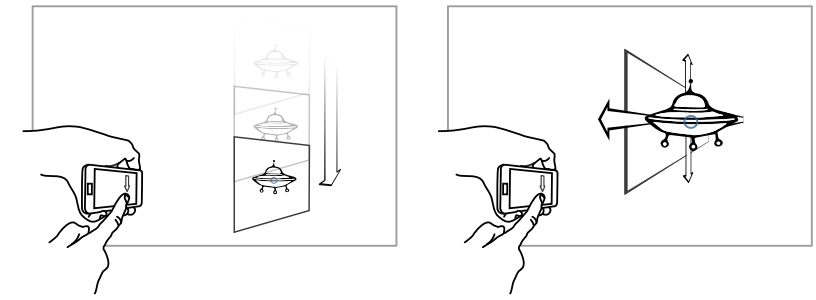
The second variation, *Free* Plane-Casting, is similar to Pivot Plane-Casting in that swiping on the touch surface translates the object on the movement plane. The primary difference between the techniques is that *Free* Plane-Casting also translates the plane and its pivot point along with the object (Figure 7.2). In a sense, the object and the plane are "interlocked", and move together in 3D space, always in the direction afforded by the plane's orientation.

The two Plane-Casting variants offered comparable quantitative performance. However, participants strongly preferred *Free* Plane-Casting. We hence use this variant as the basis for INSPECT, and simply refer to it as *Plane-Casting* for the remainder of this article.



(a) Free Plane-Casting: Gesturing translates the acquired object along with the plane.

(b) Pronating aligns the plane vertically.



(c) A vertical swipe moves the object vertically.

(d) On free Plane-Casting, only 2-DOF are instantly available at one time.

Figure 7.2: Free Plane-Casting. This technique was preferred by the users and is used as the basis for INSPECT.

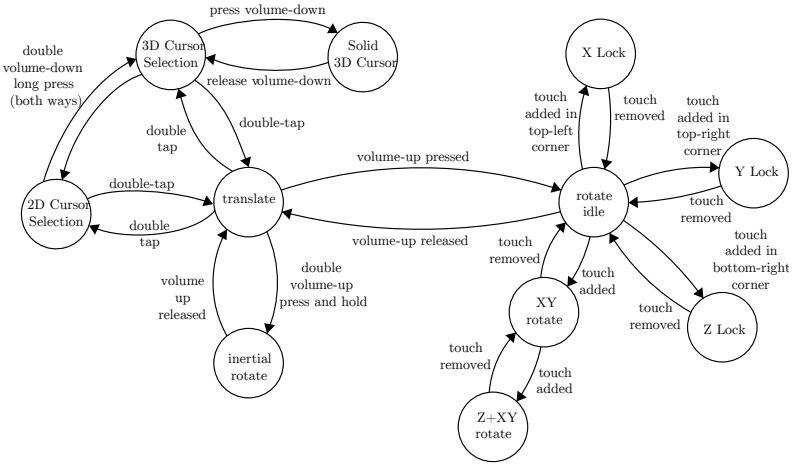


Figure 7.3: Finite state machine transitions between the modes available in INSPECT.

7.2.2 INSPECT - Design Decisions

Like Plane-Casting before it, INSPECT was intended for use with smartphones. Hence we carefully considered the capabilities of these devices in implementing INSPECT. Smartphones typically provide 3-DOF rotations from the combination of accelerometer, magnetometer and gyroscope, 2-DOF per finger from the touch-screen (usually with 2-3 fingers), and volume-up and volume-down buttons. Although some devices provide additional inputs, we used this minimal subset as these are the only universally available input streams on smartphones.

INSPECT was designed based on Jacob's findings on performance gains when "*the structure of the perceptual space of a graphical interaction task mirrors that of the control space of the input device*" [35]. This is indeed the case with INSPECT, which in addition to having a good perceptual match (smartphone orientation matches that of the movement plane), benefits from the use small muscle-groups [79], since most of the work is done by the fingers. Finally, the technique works with the dominant hand fingers interacting very close to the off-hand palm. This establishes a frame of reference for the dominant hand to manipulate against [20]. This design leverages the benefits of bimanual interaction that have been repeatedly discussed in the literature [4].

We also wanted to allow users to use the technique while looking directly at the large display, without having to look at the device. During presentations, the presenter's gaze guides the audience and if the presenter were to look at his device screen to manipulate it would create a disconnect with the audience. This also allows the presenter to interact in a natural standing pose, with their arms resting by their torso while the device is supported by both hands. Holding the device near the torso was a key design point for fatigue-free interaction. In contrast, many wand or gesture-based techniques require the user to hold the device with the arms extended, which induces fatigue. Finally, a small device held with the non-dominant hand gives users the freedom to point to the display with their dominant hand between manipulations. This is essential during presentations, for example.

Extensions to translation mode

To improve object translation, we added a "flick" gesture to Plane-Casting. This allows the user to launch the object inertially in the direction of the flick. In position-tracked wands, controlled by the arm, flicking motions are not so easy to perform because flicking requires a rapid acceleration of the wand. Such an

accelerated motion is less than trivial to perform, and gets even more difficult when repetition is required. A finger gliding on a touch-surface, on the other hand, lends itself well to flicking. Flicking provides an alternative to the gain functions often used in 3D user interfaces to scale input [73]. Moreover, inertial flicking is often used in smartphone UIs for scrolling and other tasks. Consequently, we expect that smartphone users will be able to adopt flicking quickly due to its familiarity. Much like smartphone UIs, touching the touchscreen after flicking an object stops its movement. While inertial flicking has been explored previously for direct touch techniques [74, 11], and for 2D graphics[3] to the best of our knowledge, it has not been used with off-screen touch for 3D manipulations. To translate using flicking, the same gesture is used, as in Plane-Casting. When the finger is lifted following a gesture, if it has crossed a certain speed threshold, the object is launched inertially with flicking.

Translating objects with Plane-Casting required repeated supination/pronation motions for fine positioning orthogonal to the movement plane. We expected that this may frustrate users. Consequently, we added pinch gestures to move the object along the current movement plane's normal vector. Pinching the fingers *away* translates the object parallel to, and in the direction of the plane normal. Conversely, pinching the fingers *together* (or "un-pinching") translates the object in the opposite direction. When holding the device upright, this mapping is similar to that used by Sticky Tools [25] or most touch interfaces where pinching away brings the object closer to the surface (zoom). This yields vertical motion relative to the movement plane, and could be useful in visualisation applications where the plane acts as a slicing plane. We refer to this mode as *pinch translate*.

Extensions for rotation

In addition to the translation extensions discussed above, we added a new mode to enable rotation. Several tabletop systems use rotation techniques where the fingers directly touch the manipulated object and/or the display surface [74, 24, 25, 67]. However, we propose off-screen rotation using indirect touch which has not been explored previously in 3D graphics. The smartphone's volume-up button switches the system to rotation mode while being held pressed.

Including an explicit mode change for rotation might initially appear cumbersome. However, we argue that low-cost mode changes do not introduce a high cognitive demand. For example, the smartphone's volume buttons are available at natural grip positions. Pressing these buttons has a very low cognitive cost,

similar to the "shoulder" or trigger buttons on modern game controllers. In addition, thumb pressure is counteracted by the forefinger when pressing these buttons, so the mode change has minimal (if any) effect on the movement plane orientation.

There is another interesting side effect of holding a button to switch between the translation and rotation modes. It is possible for experienced users to use the inertial flick feature, switch to rotation mode, and rotate while the object is still flying. While a form of simultaneous rotation and translation is also possible with Sticky Tools [25] and Wilson's work [74], we argue that this is easier with an explicit mode change and that integrated translation and rotation modes might lead to accidental input or unpredictable/irreversible motions.

The rotation modes are straightforward:

Horizontal finger motion (on the touch-screen X axis[1]) rotates the object about the world Y axis (Figure 7.4). Vertical motion (device Y axis) rotates about the world X axis. This mode provides integral rotations on the X and Y axes that are performed with a single finger and will be referred to as *XY rotate*. *XY rotate* should not be confused with ARCBALL [63] despite the similarities. ARCBALL uses a function to project the 2D touch points onto a virtual sphere whereas *XY rotate* simply converts translation of the touch point to rotation. *XY rotate* thus exhibits a distinctly different behavior to ARCBALL. XY Rotate is a form of two-axes valuator [10] implementation for indirect touch.

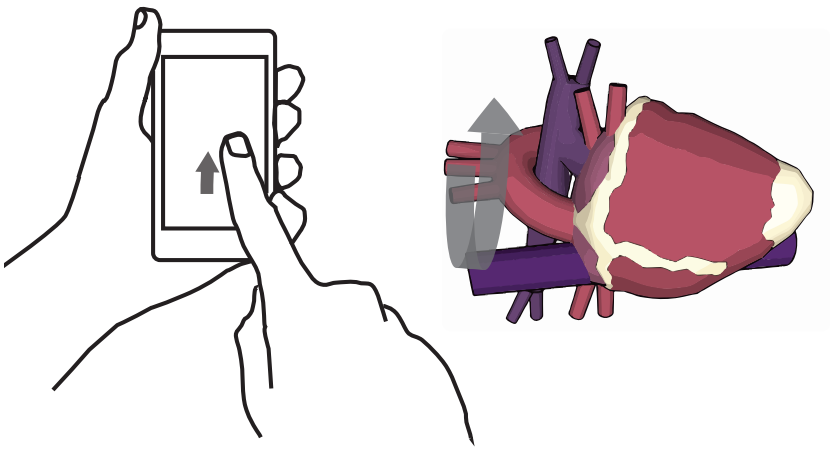


Figure 7.4: Vertical motion rotates about the X axis of the 3D world.

[1] A note on axes: When the device is held upright (Figure 7.5), the axes on the device are identical to the axes on the display.

To rotate the object about the Z axis we use two fingers which are pivoted about their midpoint. If the two fingers are moved in parallel, their motion is interpreted as a single touch point which induces the same rotation as *XY rotate*. This feature allows minor corrective adjustments to the X and Y axes while rotating about the Z axis without requiring lifting a finger from the screen or further mode changes. This mode will be referred to as *Z+XY rotate*. The *Z+XY rotate* mode is only possible because INSPECT is based on indirect touch. In direct touch systems the parallel motion of the two fingers is usually mapped to translation [25]. As such, a three axis integral rotation mode is, to the best of our knowledge, unique to our system. Users can also make fluid transitions between single finger and two finger rotations as desired. *Z+XY rotate* feels similar to rotating a physical trackball yet is different from Arcball+ by Rousset et al. [61]. Arcball+ uses the midpoint to rotate like the classical ARCBALL [63] algorithm. We avoided this approach because ARCBALL is known to affect the Z axis as well.

In any of the rotation modes, the orientation of the device is ignored. Rotations are always performed as if the device was held vertical facing the display.

During pilot testing, participants indicated that rotating unfamiliar objects without an obvious "up" orientation did not present any problems. However, rotating objects that had a clear "up" direction required more controlled rotations. For example, when rotating the human heart (a comparatively unfamiliar object), participants were happy with their rotation, even if the heart was slightly tilted off the Y axis. In contrast, when rotating an office chair (a familiar object with a clear "up" direction) participants would try harder to ensure the orientation was absolutely correct. For this reason, we decided to add single-axis constrained rotations.

Single-axis constrained rotations are activated by touching the display corners. Analysis of our pilot study touch data revealed that users rarely reach the touchscreen corners while moving objects with Plane-Casting (Figure 7.11a). Consequently, we decided to use the screen corners for explicit rotation mode changes. The natural shape of the hand allows for a stationary finger in the screen corner, while another finger moves freely to control one DOF (Figure 7.5). Thus, we introduced the following rotation mode changes depending on the touch point of the first finger to touch the display. Fingers are obviously not detected, but we make recommendations on which finger to use for better ergonomics:

(X) The forefinger on the top-right corner constrains rotation about the dis-

play's Y axis. The thumb is used to control rotation. (Figure 7.5). **(Y)** A thumb on the bottom-left corner constrains rotation to the display's X axis with the forefinger is used to control rotation. **(Z)** A thumb on the bottom-right corner constrains rotation to the the display's Z axis. The forefinger is used to control rotation.

For example, touching the top-right corner of the touchscreen activates Y axis constrained rotation mode (Figure 7.5). The thumb's vertical motion on the touchscreen is ignored and only the horizontal component rotates the object about the constrained Y axis.

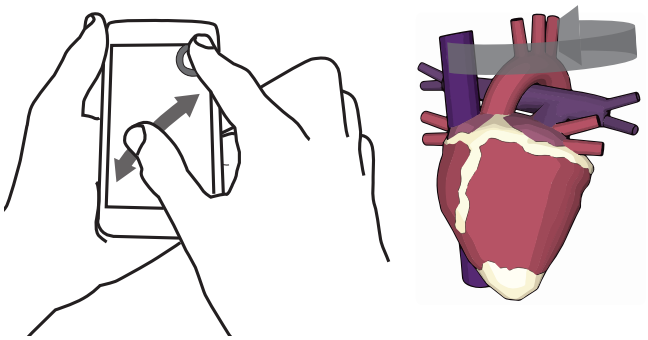


Figure 7.5: When in rotation mode, placing the first finger on the corner constrains rotation to a single axis (Y-axis in this case).

Using axis constraint mode requires the first touch to be near the corner. The finger is subsequently free to roam the touch-screen so long as it remains touched. This was intended so that the initiating finger doesn't impede the motion of the second finger, which actually performs the rotation. This novel way of accessing additional modes by placing a touch on the corners is a feature unique to INSPECT, and could be further extended for accessing additional modes with double-taps, swipes, etc.

We decided against using more than two fingers for switching modes or added functionality. Unlike tablets, where users have access to a large surface, small smartphone touch screens cannot easily accommodate many fingers. Similarly, using more than two fingers precludes implementing our technique on *very* small touch screens, such as those found on smart watches. Similar to the translation mode, we added inertial flicking to all three axes in rotation mode. Similar to the translation mode, if a finger exceeded a certain speed threshold, the system kept the object spinning about its center with a fixed decay rate. This entire set of indirect touch rotation techniques will be referred to as *touch-rotate* in

the evaluation section.

Finally, we also supported direct rotation using the smartphone orientation sensors. Direct rotation was activated by double-tapping and holding the volume-up button. Rotation was relative to the device orientation at the time of the button press. This mode allowed users to clutch to avoid strenuous wrist positions. This technique will be referred to as *phone-inertial* in the evaluation. INSPECT's inertial rotation mode is similar to that of the Flying Mouse [69]. However, the designers of the Flying Mouse made an unexpected design choice. They require users to keep their thumb on a UI widget on the touch screen to access various modes (including the inertial rotation mode). In addition to having to aim (and keep the finger stationary) to access the mode, when the device is rotated the user can no longer see the UI widget and is thus difficult to know if he/she is pressing it correctly. Finally, such a design choice makes it difficult to do the rotation bimanually.

It should be noted that some state-of-the-art *direct* touch techniques (like tBox [11] and Sticky Tools [25]) which are designed for tabletops, can be adapted for use on vertical or handheld touch displays. Such techniques are not mutually exclusive and could complement INSPECT, depending on how close the user is to the display.

7.2.3 Selection

We also considered two object selection modes for use with INSPECT. The first used a relative 2D cursor. In this mode, one finger moves the cursor relative to its current position, similar to the trackpad commonly found on notebook computers [3]. Objects under the 2D cursor are highlighted and touching the screen with a second finger or double-tapping selected the object. We also prototyped a *virtual hand*-like selection mode. In this mode, a spherical cursor (virtual hand) is controlled by Plane-Casting and intersects the desired object. Either of these two selection modes could be applicable depending on the context of usage. We found the former to be quicker, yet the latter offers the potential to select occluded objects or "nudge" objects to reveal the desired one (e.g. in a 3D sandbox game/educational application with collision detection). A double-tap and long press on the volume-down button switched between the 2D and 3D cursors. When in 3D cursor mode, holding the volume down button made the 3D cursor solid for bumping against other objects (Figure 7.3). Neither of those selection modes was formally evaluated because there is extensive literature on 3D selection.

7.3 Evaluation

There is no widely accepted standard for off-screen 6-DOF control and the Flying Mouse[69] for the iPhone only works with companion apps. We thus elected to compare against a 'gold standard' technique using a 6-DOF Polhemus sensor. Direct manipulation using a tracked wand is widely used and accepted as being an intuitive way of interacting with a 3D object. For example, similar techniques are extensively used in 3D interaction for visualization and virtual reality [73]. Although we did not expect INSPECT to outperform the wand technique, we thought it necessary to provide this comparison as reference.

Although numerous precision-enhancing techniques have been proposed for direct manipulation techniques, we instead opted for a basic "virtual hand" type technique using the Polhemus-tracked wand. There are three main reasons for this. First, most precision-enhancing techniques require tracking the user's body[73], which is contrary to our low-instrumentation objective. Second, our intended application scenario assumes a shallow 3D space, which does not require users to position objects so far away (remote positioning was a primary reason why enhanced precision functions were first developed [55, 73]). Finally, using a "raw" direct manipulation technique such as that used in our study makes it easier to replicate the experiment. We used a Polhemus tracker as it should offer better accuracy than other low-cost devices (e.g., the Sony PS Move and PSEye camera).

7.3.1 Participants

Twenty participants took part in the study (within-subjects). Fourteen participants were male. Their ages ranged from 21 to 36 years (mean age 28 years). All had normal or corrected-to-normal vision. All but one were right handed. Participants self-reported regular gaming habits but little to no 3D user interface experience. Participants were compensated with $20 upon completion of the experiment.

7.3.2 Apparatus

Hardware Setup

The experiment was conducted using a PC running Ubuntu Linux with a 2.4 GHz processor and 16 GB of RAM. The graphics card was an NVIDIA 680GT with 2 GB of RAM. A Sanyo PDG-DWL2500J ultra-short focus projector was used as the display. The projector offers a 1920x1080 pixel resolution at a 16:9 aspect ratio. The display size was 320 cm diagonally, and the display was monoscopic. Participants stood 300 cm from the projected screen. Figure 7.6 depicts the equipment setup used in the experiment.

In the INSPECT condition, participants held a smartphone in their non-dominant hand, while touch-gesturing on the smartphone's touchscreen with their dominant hand. The smartphone used for the experiment was a Samsung *Galaxy SII* running Android OS 4.0.3. This device features a 10.5 cm diagonal screen at a 1280 x 720 pixel resolution, a quad-core 1.4 GHz processor, and 1 GB of RAM. The smartphone was connected to the PC via a WiFi network.

In the wand condition participants held a Sony Move controller in their dominant hand. Although the *Move* is normally tracked by the Sony *PSEye* camera, we instead used a Polhemus *Fastrack* receiver for superior tracking accuracy. The control-display ratio was set to a fixed gain (no acceleration) for both translation and rotation in this condition. Participants held the top button (thumb) on the wand to activate object translation and the rear trigger button (forefinger) to activate object rotation. Both translation and rotation were relative to the position/orientation of the wand upon pressing the button.

A 3Dconnexion *Space Navigator* was also used as a foot switch. The sole purpose of the device was to advance trials.

Software Setup

The experiment used custom software running on the PC, which was written in C++ and OpenGL 3.3. Custom software on the smartphone was written in Java and communicated with the PC software via Google protocol buffers for the sensor data and TUIO messages for the touch events. The software presented the experimental tasks described below. In both tasks, the scene was displayed with the target object floating over a "floor" - a flat plane with a cross-hatch pattern textured on it. Shadow rendering was included to help facilitate

depth perception. Both the cursor and the target cast shadows onto the floor (figure 7.7). The software also logged the trial length, the cursor position and orientation, touch events on the smartphone, and other relevant metrics.

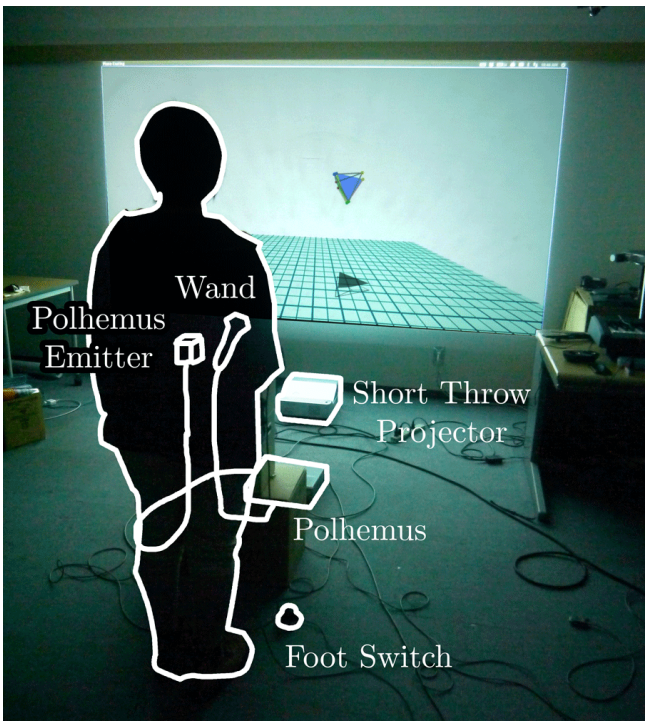


Figure 7.6: Photo of a user taking the experiment.

7.3.3 Procedure

Upon arrival to the lab, participants were given a short briefing about the experiment. This included a verbal explanation of the experiment purpose, the tasks, and a description of the techniques being compared in each task (described in detail below). The experimenter then demonstrated the available control modes and participants were allowed to practice both techniques until they felt confident to begin the task. Following the briefing, participants were asked to perform both the movement task and the rotation task.

Once participants felt they had a good match to either the target position or orientation (depending on the task), they would press the foot switch to advance to the next trial. Rather than enforcing a fixed "success" threshold, participants were free to judge when the match was accurate enough. This allows data from the experiment to be additionally analysed for 3D pointing (Fitts-

law) type studies. Participants were asked to maintain a consistent balance between speed and accuracy throughout the task. They always completed the movement task followed by the rotation task. The motivation to split the 6-DOF docking task to a movement and rotation task was that depending on the form factor of the smartphone, the volume up and down buttons can be difficult to press. We thought that this would introduce a new variable to the completion time and would make it difficult for other researchers to replicate our study. Additionally, because INSPECT is a *set* of techniques, a 6-DOF docking task would not allow us to individually assess the strengths and weaknesses of each sub-technique of INSPECT. Splitting the task in a movement task and a rotation task will potentially reveal weak points of INSPECT and show areas that need improvement.

Specific procedural details for each task are outlined below.

Movement Task

The movement task required matching the cursor position to the target position. INSPECT and the Wand technique were compared using this task.

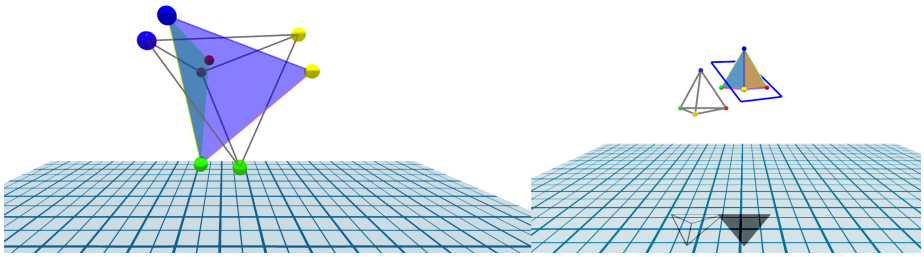


Figure 7.7: Screenshot of the rotation (left) and movement (right) task respectively.

Upon starting a movement task trial, a semi-transparent tetrahedral cursor was already acquired for movement. The cursor appeared in the center of the screen (Figure 7.7). Participants were instructed to move the cursor to match the position of a wireframe target using the current technique (either INSPECT, or Wand). The target was the same shape and size as the cursor. The four corners of the cursor and target were each a different color sphere. The coloured corners were primarily important in the rotation task (i.e., to determine orientation of the cursor and target), but were shown in the movement task for consistency. During the movement task, the cursor and target always maintained an upright

orientation so as to rule out effects of rotation during this task. The software presented targets at 12 pre-defined positions, one at a time, randomly shuffled for every participant. Each position corresponded to one of 12 vertices of a regular icosahedron centered at the origin. Each position was tested with two different distances from the origin (same direction, twice the euclidean distance).

Rotation Task

The rotation task required that the participant match the cursor orientation to the target orientation (Figure 7.7). This was performed independent of the movement task, and was used to compare the two smartphone based rotation techniques (rotation using touch and using the inertial sensors) to direct rotation using the wand.

As in the movement task, the cursor and the target both appeared centered at the origin in the rotation task. Participants had to match the cursor to 12 pre-defined rotations shuffled for each participant, twice. Target rotations were generated in the same pseudo-random order for all participants. The rotation task included two techniques using the smartphone (touch rotation and direct rotation) and direct rotation using the wand.

7.3.4 Design

Due to the differences in the tasks, we present design details for each task separately. Participants always completed the movement task prior to the rotation task. However, all other condition orderings within each task were counterbalanced according to a Latin square.

Movement Task

The movement task used a single within-subjects independent variable, movement technique. The two movement techniques compared were wand and INSPECT. For each movement technique, participants performed multiple movement tasks at two different distances in each of 12 directions. Consequently, each participant completed a total 2 movement techniques $\times$ 2 movement distances $\times$ 12 directions = 48 movement trials. Over all 20 participants, this corresponds to a total of 960 movement trials.

The dependent variables for this task were movement time (MT), measured in seconds, and the euclidean distance between the target and cursor centres upon trial completion, measured in cm. This latter dependent variable served as a measure of accuracy. Movement time was measured as the time from when the trial began to the time the participant pressed the foot pedal.

Rotation Task

The single within-subjects independent variable for the rotation task was rotation technique. Three rotation techniques were compared: touch-rotate, phone-inertial, and wand direct. Each rotation technique was evaluated twice with each of 12 randomly generated target rotation angles. This produced a total of 3 rotation techniques $\times$ 2 repetitions $\times$ 12 target orientations = 72 rotation trials for each participant. Over all 20 participants, this yielded 1440 rotation trials in total. The dependent variables for the rotation task were rotation time (RT), measured in seconds. This was measured as the time from when the target appeared until the time participants pressed the foot pedal upon completing the rotation trial.

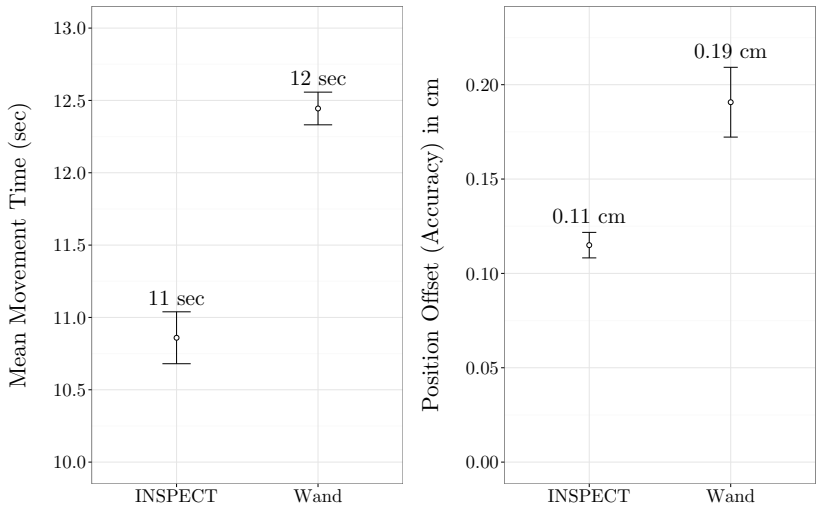
7.3.5 Hypotheses

We hypothesized that the wand technique will be faster than the other techniques **(h1)** as it leverages natural movements that participants are accustomed with from their daily life. Also that the wand technique and phone-inertial mode will be less accurate than the touch rotation techniques **(h2)** because holding the wand in a distal position will have an adverse effect. We also believe that the phone inertial mode will suffer from the form factor of the smartphone, which is not ideal for rotations like a fingerball [30] is for example. Finally, that due to the nature of holding the smartphone close to the torso and the wand in a distal position we expected that overall participants will prefer INSPECT and will complain about fatigue using the wand technique **(h3)**.

7.4 Results

7.4.1 Movement Task Results

During the movement task there were only two techniques. Consequently, the data were analyzed using a one-way ANOVA. Technique had a significant effect on movement time (MT). ($\mathbf{F} = 126.9, p < 0.001$). Participants completed the task 12% quicker using INSPECT (Figure 7.8a). This is contrary to our first hypothesis, but encouraging overall as it shows that interaction with INSPECT is quicker than direct manipulation with the wand.



(a) Movement Time (lower is better).

(b) Distance between cursor and target (lower is better).

Figure 7.8: Results from the movement task (with Standard Error).

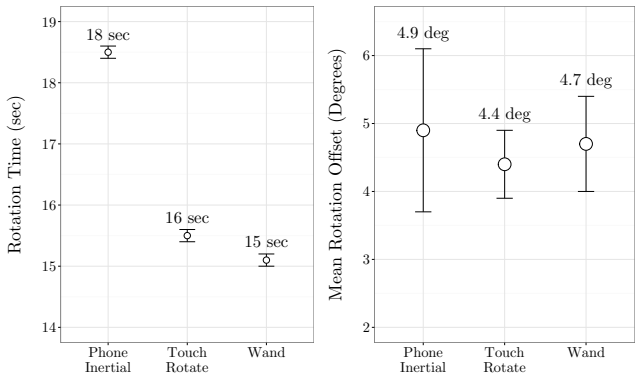
A one way ANOVA revealed a significant effect on accuracy for movement technique ($\mathbf{F} = 188.764, p < 0.001$) (Figure 7.8b). Participants were more accurate in matching the target position using INSPECT than with the wand - the error distance with the wand was about 40% higher than that of INSPECT. The average distance mismatch for the wand was 0.19 cm with 0.11 cm for INSPECT. This confirms our second hypothesis: (h2) INSPECT is more accurate than the wand.

Log file analysis revealed that participants rarely used the flick gestures. Approximately 10% of the recorded frames used flicking. We believe there are two reasons for this. First, participants had no prior experience using the techniques. As such, they did not feel confident launching the object around with

inertia-based flicks. Second, the distance to the targets were not long enough to warrant flicking. We believe a different task requiring longer translation distances (e.g., an outdoor AR task) would increase the value of flicking.

7.4.2 Rotation Task Results

The data from the rotation task was subjected to a repeated measures ANOVA test. Technique had a significant effect on rotation time (RT) ($F_{2,23} = 162.15, p < 0.05$). A post-hoc analysis indicated that the phone inertial rotation was slower than touch rotation and the wand. No statistical significance was found between the wand and touch rotate. Mean scores for rotation time are shown in Figure 7.9a.



(a) RT means for rotation task (with Standard Error). Lower is better.

(b) Rotational Offset means for rotation task (with Standard Error). Lower is better.

Figure 7.9: Results from the rotation task (with Standard Error).

Upon ending each rotation trial, rotation accuracy was calculated as the angular difference (extracted from the quaternion) between the cursor orientation and the target orientation (Figure 7.9b). The repeated measures ANOVA showed no statistical significance ($F_{2,22} = 1.13, p = 0.78$).

These results partially validate h2 for the rotation task yet we expected the time gap to be larger than two seconds. The form factor of the smartphone is indeed not ideal for rotations. We also hypothesized that touch rotate will perform better than the direct rotation techniques (wand and the phone inertial). This was not the case, however, and in an attempt to discover the reasons we analysed the time users spent on each rotation mode of touch rotate.

7.4.3 Mode dwelling during rotation task

During the rotation task, some of the pre-defined target orientations were simple 90° rotations about a single axis. In those trials, participants simply had to use the corresponding axis constraint rotation mode and control one DOF to accurately match the rotation. Had the participants used the axis lock modes they would have achieved almost perfect rotations However, despite the availability of the constraint modes, participants did not use them very much (Figure 7.10)

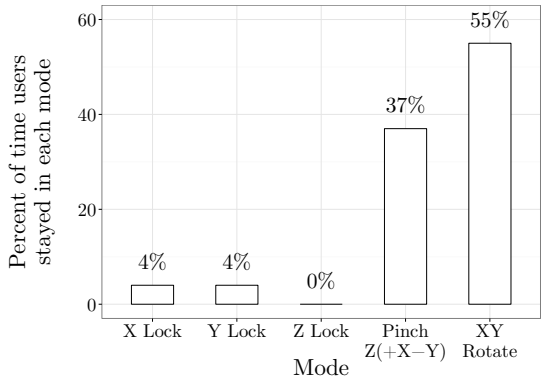


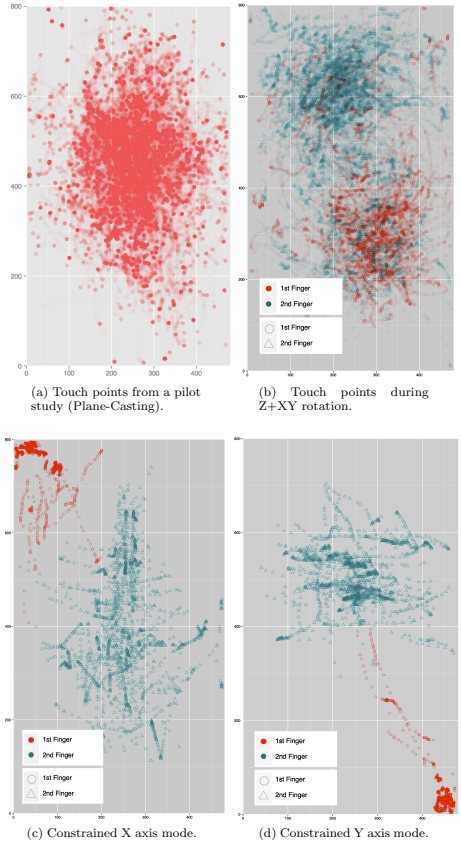
Figure 7.10: Mode dweling during the rotation task: It is interesting to note that participants did not use the Z axis constraint mode at all. They completely ignored that mode in favor of Z+XY rotate.

We believe one reason for the relative under-use of the constrained rotation modes was the somewhat arbitrary choice of corner-to-axis mapping. There was no easy mnemonic or meaningful mapping for the participants to remember to activate the constraint rotation modes. Consequently, participants instead spent most of their time in the single finger rotation mode (XY Rotate), followed by the pinch rotation mode for Z axis control. The XY Rotate mode (single finger) controls two axes. That is two out of three axes to be manipulated. This should theoretically give it close to a 66% use. Interestingly, this was not the case. We speculate that this was because while the pinch rotate mode is mainly used for Z axis rotation, it can also control rotation about the X and Y axes by simultaneously moving both fingers parallel to each other.

We postulate that these mode dwelling results partially explain why, during the rotation task, touch rotate and wand performed similarly both in terms of completion time and accuracy. Because participants did not use the additional modes that would have potentially offered an advantage of accuracy and speed.

7.4.4 Touch Areas

Visualization of participants' touch-points confirms findings from our pilot studies. Depending on the control mode, the participants' touches do not cover the entire touchscreen and some areas are unused (Figure 7.11). For example, the bottom of the screen during the Z+XY rotation mode (Figure 7.11b) was unused. These under-utilized parts of the touchscreen offer screen real-estate for adding further system control options, or UI-widgets.



(a) Touch points from a pilot study (Plane-Casting).

(b) Touch points during Z+XY rotation.

(c) Constrained X axis mode.

(d) Constrained Y axis mode.

Figure 7.11: Touch points during the various rotation modes. Red points represent the first finger to touch the screen whereas blue represents the second finger.

7.5 Subjective Results

Following completion of the experiment, participants were asked to state their preference between INSPECT and the wand technique. They were also asked

to choose which technique felt more accurate and which technique had greater impact on the limbs (in terms of fatigue), 3 questions in total.

The qualitative results were notably skewed in favor of INSPECT. 17/20 participants preferred INSPECT overall. 18/20 thought that it felt more accurate and was less fatiguing. Participants commented that INSPECT was both fun to use and easy to understand. They further commented that the buttons on the side of the Samsung Galaxy SII were slightly hard to press which made mode changes slightly difficult. We agree with this assessment, and believe that INSPECT would benefit from having a few easy to press buttons on the forefinger side of the device. Finally, a number of participants commented favourably on the Z+XY rotation mode. The liked that the mode also allowed for XY rotation adjustments, and commented that they found it easy to use.

7.6 Discussion

Indirect touch, as opposed to direct touch interaction, suffers from the problem of selection. With Sticky Tools or tBox, the beginning of the touch gesture can simultaneously indicate object selection. With INSPECT, the manipulated object must be explicitly selected first with a selection step. This might complicate the use of INSPECT in applications requiring frequent selection among multiple different objects. Also, the magnetometer in smartphones is slightly susceptible to electromagnetic interference. If the user moves away from the display and sits while resting his arms on a metal structure (e.g., a desk) they may need to re-calibrate the orientation to avoid drift. The rotation mode would be unaffected in this case.

The movement task results come as a bit of a surprise. The wand technique leverages experience from daily use of the arms. We thus expected novice participants to perform better with it than with INSPECT. However, performance with INSPECT was actually better than the wand. This result validates our design decisions and indicates that a technique designed based on the aforementioned design principles shows tangible benefits. Namely:

- Bimanual in nature.
- Perceptual space of interaction task mirroring control space of input device.
- Small muscle groups - Interacting with palm/fingers rather than arm/-

forearm.

- Dominant hand interacting close to the Off-hand.

The rotation mode dwelling results as well as the low use of the flicking mode indicate that our novice participants were not familiarised enough with INSPECT to access the additional modes and confined themselves to just using the basic ones. Even if performance in the evaluation tasks could have benefited from the use of these modes, the task did not demand their use and as such participants did not make the extra effort to utilise them. We think that with time and experience users will feel comfortable accessing, and benefit from these additional modes. Nevertheless, making these modes as well as additional ones more easily accessible could possibly have significant benefits for INSPECT. This is a point for improvement that must be carefully considered. Although the wide availability of smartphones motivates designs that can be used with smartphones, a different form factor, one that has access to a greater number of easily accessible buttons without affecting the size of the touch area could be beneficial.

In our experiments, users stood directly across from the display. In a collaborative situation that might be the case, but in a presentation scenario, the presenter would most likely be standing in a skewed position, to the side of the display. Unlike HOMER and other ray based techniques, INSPECT does not require the smartphone's position to be tracked relative to the display. Because of this we believe INSPECT will be more robust to the user moving relative to the display. That, however, remains to be demonstrated experimentally and is a potential goal for future research.

The recent proliferation of touch devices such as smartwatches and smartphones/tablets of different sizes begs the following questions: What effect does the screen size have on the performance of INSPECT? The screen real-estate of a smart-watch would potentially make it difficult to control modes that require more than one finger, like *pinch translate* and *Z(+X-Y) rotate*. How could one overcome screen size limitations?

7.7 Conclusion and Future Work

We have presented INSPECT, a set of novel indirect touch techniques for 3D manipulation using a low cost input device such as a smartphone. The proposed technique to a certain extent meets the design goals set at the beginning of this

paper, such as simplicity, accuracy and low-instrumentation/cost. INSPECT was overwhelmingly preferred over the wand technique by our experiment participants. The evaluation revealed that INSPECT performs 12% faster than a baseline wand technique for a 3D translation task while achieving 40% better accuracy and performs almost on-par at a 3D rotation task. The diverse rotation modes proved challenging for our novice participants and finding ways to enable simple access to a variety of modes remains a target for future work.

Chapter 8

Conclusion

8.1 Discussion

An interface that is to be used for 3D manipulation during presentations should satisfy a number of design criteria. In this section we attempt to demonstrate how each part of this thesis addresses the design goals set at the beginning. This thesis set out to mainly address the following goals:

- Interacting with small movements (to avoid interfering with other users - bystanders)
- Keeping limbs close to the body to avoid fatigue.
- Simplicity in input and tracking.
- Low cost.

In the following sections we clarify how each proposed technique individually satisfies these goals:

8.1.1 Mesh-Grab and Arcball 3D

Mesh-Grab and Arcball 3D were designed so that the wand can be held close to the body and so users can simultaneously rotate and translate the object by casting a ray and flicking the wrist. The introduction of inertial physics in

the manipulation allows for big distances to be covered with only small flicking motions to launch the object. The rationale behind this design was that users would get accustomed to the physics and the interaction style and would eventually be able to master the technique, achieving their desired rotation and translation with minimal wrist motions, similar to how somebody can skilfully throw a crumbled piece of paper into a waste basket from a distance without standing up and walking to it.

Implementation of Mesh-Grab and Arcball 3D requires that the wand is tracked in a much smaller range compared to existing techniques (figure 8.1).

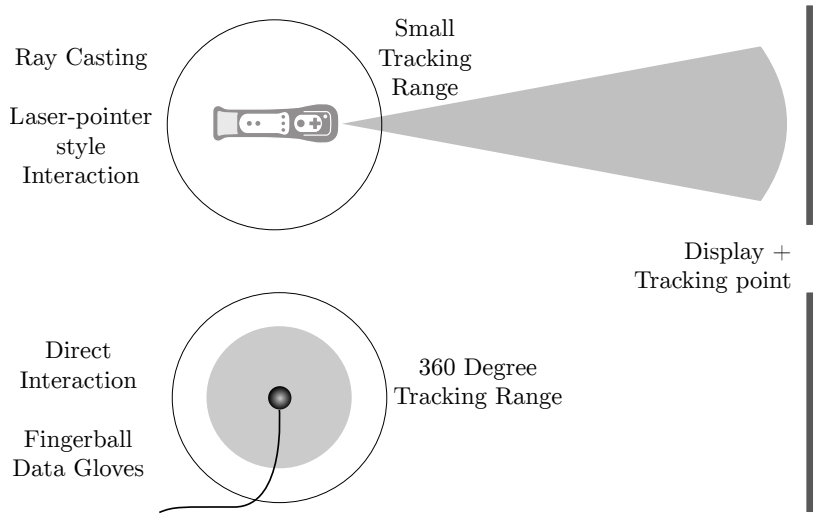


Figure 8.1: INSPECT requires tracking a shorter range as opposed to finger-ball or other wand-based direct techniques.

As a result these techniques can be implemented with single point tracking such as a camera or a set of infrared LEDs on the display with the camera on the wand (like the wii-mote) as opposed to multiple tracking points or tethered magnetic trackers (Polhemus^{TM} [54]) used by other solutions.

Our evaluation demonstrated that Mesh-Grab and Arcball 3D perform similarly against a state-of-the-art baseline interaction technique while at the same time being more accurate. Given this similar quantitative performance combined with the preference of the users and the gains in hardware setup simplicity it's safe to conclude that Mesh-Grab and Arcball 3D to a large extent meet the design goals set at the beginning of the thesis.

8.1.2 Plane-Casting and INSPECT

Although ray-based techniques such as Mesh-Grab and Arcball 3D allow for simpler tracking compared to the state of the art techniques, there was much room for improvement. Plane-Casting makes further improvements in the setup simplicity since it is applicable on any smartphone. The design of Plane-Casting and INSPECT means the smartphone controller does not need proximity or line-of-sight to the display and is, therefore, even more suitable for situations with multiple users such as a large display in a busy classroom or an interactive museum exhibit. Finally, the nature of the hardware, with smartphones being widely available and users being very familiar with them adds further gains in cost and simplicity.

Although the work on Plane-Casting and INSPECT did not quantify position of the arms like the wand techniques, they are also designed with reduced limb strain in mind, since it is possible to hold the smartphone against the torso and operate it with the small muscle groups of the forearm (figure 8.2). Finally flicking in INSPECT is employed to help cover large distances, just like the ray techniques only this time flicking is performed with the fingers rather than the wrist, which further reduces strain to the limbs. Flicking is often used in direct 2D touch interfaces to scroll up and down so we postulate that employing flicking in a touch interaction setting will leverage the user's experience from daily smartphone use.

Figure 8.2: INSPECT and Plane-Casting are meant to be used bi-manually with the arms resting on the ribcage with only the forearms and the fingers moving.

8.1.3 Unistroke

Unistroke addresses the problem of making long strokes for annotation on a 3D mesh with a single input. In the event of a large display with a laser-pointer like device as input or an interactive whiteboard there is often a single point input point for interaction. Unistroke, therefore makes headway by facilitating easy painting of long strokes from a single 2D input point, thus facilitating interaction with a simpler hardware setup. Users in our pilot study found unistroke easy to learn and use but that perhaps a more competent version of unistroke would require multiple input points and that this would complicate the input hardware setup.

8.2 Design of Interaction Techniques

A school of thought in virtual reality argues for the design of techniques as close as possible to real-world metaphors so that users can interact seamlessly and can leverage their real world skills. i.e. Direct interaction using the hands, haptic feedback and immersion.

Results from our experiment on Mesh-Grab and Arcball 3D though demonstrate that because users often need to interact using other means rather than directly with their hands that is not always desirable. e.g. When using ray-based techniques for 6-DOF manipulation, putting a "bubble" around the 3D object resulted in better performance from the users. The same cannot be said to hold true for manipulating objects in the real world. Putting a plastic bubble around all objects would not make handling them easier.

The same is true for Plane-Casting and INSPECT. There is no real-world metaphor equivalent to those interaction techniques yet users were able to master them and even out-perform the direct techniques that match the real world much better. The inertial flicking in Mesh-Grab and INSPECT emulate weightless conditions in space. Although users don't have experience from the real world (except a few astronauts perhaps) they quickly adopted the techniques and preferred them over the ones that matched the real world more closely.

Designers of interactive systems might tend to design techniques with real-world metaphors in mind but we postulate that perhaps that is not always the best approach. Kulik et al. [40] have also previously discussed this same point.

The techniques presented in this thesis were designed with large displays and presentations in mind, but we argue that they could be equally applicable to immersive Virtual Reality with an HMD or a CAVE system, augmented reality, electronic entertainment etc.

8.3 Summary - Future Work

This thesis contributes three novel interaction techniques that address fatigue, simplicity and cost in 6-DOF virtual object manipulation for presentations educations and collaborations in front of large displays. The techniques presented to a large extent satisfy the design goals set at the beginning of this thesis. They perform on par or on occasion better than their baseline counterparts and were overall preferred by participants in our user studies.

The set of techniques studied as part of this thesis indicate that interacting using the fingers with the limbs close to the body and with small movements is a competent alternative to similar state-of-the-art devices and techniques.

A problem that remains in all the solutions presented in this thesis is system control this was particularly evident in INSPECT where the form factor of the smartphone only provides two hardware volume buttons. Desktop computing has access to the keyboard and therefore immediate access to 101 buttons (multipled by 3 using modifiers for 303+ keys). These buttons in combination with the WIMP interface paradigm provide an easy hierarchical, discoverable means for accessing modes and for sending commands to the system. How could a small factor handheld device have similar accessibility to modes is a question worth answering.

We envision a future where a single point RGBD[1] camera [49], an electromyographic (EMG) band or some other tracking technology is used for tracking the hands and fingers. Just like in INSPECT users gestured against the smartphone screen and that input was transferred into the 3D world, we believe that the main hand will be touching/gesturing against the off-hand (figure 8.3 - This sort of pressure to the fingers/arm could be picked up by an EMG band). As soon such a tracking system is robust the question then becomes: “How can we better leverage the rich expressiveness of the human hand to control computer graphics, issue commands to systems or control robots?”.

[1]RGBD camera: a camera that can sense depth by casting infrared light into the scene in addition to normal RGB color.

Figure 8.3: Vision of the future: Users gesturing against their off hand while an RGBD camera or an EMG armband is responsible for tracking the fingers.

Bibliography

[1] 3DConnexion. Space navigator. http://www.3dconnexion.com/products/spacenavigator.html, Last Accessed: July 2015.

[2] Johnny Accot and Shumin Zhai. Performance Evaluation of Input Devices in Trajectory-based Tasks: An Application of the Steering Law. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, pages 466–472, New York, NY, USA, 1999. ACM.

[3] Apple. Magic Trackpad. http://www.apple.com/magictrackpad/, Last Accessed: July 2015.

[4] Ravin Balakrishnan and Ken Hinckley. Symmetric bimanual interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '00, pages 33–40, New York, NY, USA, 2000. ACM.

[5] Eric Allan Bier. Skitters and jacks: interactive 3d positioning tools. In *Proceedings of the 1986 workshop on Interactive 3D graphics*, pages 183–196. ACM, 1987.

[6] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Amsterdam, The Netherlands, 2002.

[7] Doug A Bowman and Larry F Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Proceedings of the 1997 workshop on Interactive 3D Graphics*, pages 35–38. ACM, 1997.

[8] Doug A Bowman, Donald B Johnson, and Larry F Hodges. Testbed evaluation of virtual environment interaction techniques. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 26–33. ACM, 1999.

[9] Margaret Bullowa. *Before speech: The beginning of interpersonal communication*. Cambridge University Press, Cambridge, UK, 1979.

[10] Michael Chen, S. Joy Mountford, and Abigail Sellen. A study in interactive 3-d rotation using 2-d control devices. *SIGGRAPH Computer Graphics*, 22(4):121–129, June 1988.

[11] Aurélie Cohé, Fabrice Dècle, and Martin Hachet. tbox: a 3d transformation widget designed for touch-screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3005–3008. ACM, 2011.

[12] William F Eddy. A new convex hull algorithm for planar sets. *ACM Transactions on Mathematical Software (TOMS)*, 3(4):398–403, 1977.

[13] Douglas Engelbart. X-Y Position indicator for a display system, November 17 1970. US Patent 3,541,541.

[14] Sean Follmer, Daniel Leithinger, Alex Olwal, Akimitsu Hogge, and Hiroshi Ishii. inform: dynamic physical affordances and constraints through shape and object actuation. In *Proceedings of the Annual ACM Symposium on User Interface Software and Technology* ,UIST, volume 13, pages 417–426, 2013.

[15] Scott Frees, G Drew Kessler, and Edwin Kay. Prism interaction for enhancing control in immersive virtual environments. *ACM Transactions on Computer Human Interaction*, TOCHI, 14(1):2, 2007.

[16] B Frohlich, John Plate, Jürgen Wind, Gerold Wesche, and M Gobel. Cubic-mouse-based interaction in virtual environments. *Computer Graphics and Applications, IEEE*, 20(4):12–15, 2000.

[17] Bernd Fröhlich, Jan Hochstrate, Verena Skuk, and Anke Huckauf. The globefish and the globemouse: two new six degree of freedom input devices for graphics applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 191–199, New York, NY, USA, 2006. ACM.

[18] Chi-Wing Fu, Jiazhi Xia, and Ying He. Layerpaint: A multi-layer interactive 3d painting interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 811–820, New York, NY, USA, 2010. ACM.

[19] Google. Protocol Buffers. https://developers.google.com/protocol-buffers/ Last Accessed: July 6, 2015.

[20] Yves Guiard. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *Journal of motor behavior*, 19:486–517, 1987.

[21] Martin Hachet, Fabrice Dècle, Sebastian Knödel, and Pascal Guitton. Navidget for easy 3d camera positioning from 2d inputs. In *3D User Interfaces (3DUI), 2008 IEEE Symposium on*, 2008.

[22] Martin Hachet, Joachim Pouderoux, and Pascal Guitton. 3D Elastic Control for Mobile Devices. *IEEE Computer Graphics and Applications*, 28(4):58–62, 2008.

[23] James Douglas Hamilton. *Time series analysis*, volume 2. Princeton university press Princeton, 1994.

[24] M. Hancock, S. Carpendale, and A. Cockburn. Shallow-depth 3d interaction: Design and evaluation of one-, two-and three-touch techniques. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1147–1156. ACM, 2007.

[25] Mark Hancock, Thomas Ten Cate, and Sheelagh Carpendale. Sticky tools: full 6dof force-based interaction for multi-touch tables. In *Proceedings of the Conference on Interactive Tabletops and Surfaces*, ITS, pages 133–140. ACM, 2009.

[26] Mark Hancock, Thomas ten Cate, Sheelagh Carpendale, and Tobias Isenberg. Supporting sandtray therapy on an interactive tabletop. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2133–2142, New York, NY, USA, 2010. ACM.

[27] Robert Held, Ankit Gupta, Brian Curless, and Maneesh Agrawala. 3d puppetry: a kinect-based interface for 3d animation. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 423–434. ACM, 2012.

[28] Otmar Hilliges, David Kim, Shahram Izadi, Malte Weiss, and Andrew Wilson. Holodesk: direct 3d interactions with a situated see-through display. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2421–2430. ACM, 2012.

[29] Juan David Hincapié-Ramos, Xiang Guo, Paymahn Moghadasian, and Pourang Irani. Consumed endurance: A metric to quantify arm fatigue of mid-air interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 1063–1072, New York, NY, USA, 2014. ACM.

[30] Ken Hinckley, Joe Tullio, Randy Pausch, Dennis Proffitt, and Neal Kassell. Usability analysis of 3d rotation techniques. In *Proceedings of the 10th*

Annual ACM Symposium on User Interface Software and Technology, UIST '97, pages 1–10, New York, NY, USA, 1997. ACM.

[31] Ken Hinckley and Daniel Wigdor. Input technologies and techniques. *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, pages 151–168, Lawrence Erlbaum, 2002.

[32] G. Hirzinger. Robot-teaching via force-torque-sensors. In *Proceedings of the sixth European Meeting on Cybernetics and Systems Research*, Elsevier Science, 1982.

[33] Takeo Igarashi and Dennis Cosgrove. Adaptive unwrapping for interactive texture painting. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 209–216. ACM, 2001.

[34] Robert J. K. Jacob, Linda E. Sibert, Daniel C. McFarlane, and M. Preston Mullen. Integrality and separability of input devices. *ACM Transactions on Computer Human Interaction*, TOCHI, 1(1):3–26, March 1994.

[35] Robert J. K. Jacob, Linda E. Sibert, Daniel C. McFarlane, and M. Preston Mullen, Jr. Integrality and separability of input devices. *ACM Transactions on Computer Human Interaction*, TOCHI, 1(1):3–26, 1994.

[36] Priscilla Jimenez and Leilah Lyons. An exploratory study of input modalities for mobile devices used with museum exhibits. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 895–904. ACM, 2011.

[37] Trevor J Hastie John M Chambers. *Statistical models in S - Local Regression Models*. Chapman and Hall, New York, 1992.

[38] Nicholas Katzakis, Kiyoshi Kiyokawa, and Haruo Takemura. Planecasting: 3d cursor control with a smartphone. In *Proceedings of 3DCHI: Touching and Designing 3D User Interfaces* workshop in *SIGCHI Conference on Human Factors in Computing Systems*, 3DCHI, pages 13–21. ACM, 2012.

[39] Nicholas Katzakis, Kazuteru Seki, Kiyoshi Kiyokawa, and Haruo Takemura. Mesh-grab and arcball-3d: Ray-based 6-dof object manipulation. In *Proceedings of the 11th Asia Pacific Conference on Computer Human Interaction*, APCHI '13, pages 129–136, New York, NY, USA, 2013. ACM.

[40] A. Kulik. Building on realism and magic for designing 3d interaction techniques. *Computer Graphics and Applications, IEEE*, 29(6):22 –33, nov.-dec. 2009.

[41] Bireswar Laha and Doug A Bowman. Volume cracker: a bimanual 3d interaction technique for analysis of raw volumetric data. In *Proceedings of the 1st symposium on Spatial user interaction*, pages 61–68. ACM, 2013.

[42] J.J. LaViola. Bringing vr and spatial 3d interaction to the masses through video games. *Computer Graphics and Applications, IEEE*, 28(5):10–15, Sept 2008.

[43] Lei Liu and Robert van Liere. Designing 3d selection techniques using ballistic and corrective movements. *International Journal of Human-computer Studies*, 69(3):170–181, 2009.

[44] Lei Liu, Robert van Liere, Catharina Nieuwenhuizen, and J-B Martens. Comparing aimed movements in the real world and in virtual reality. In *Virtual Reality Conference, 2009. VR 2009*, pages 219–222. IEEE, 2009.

[45] Anthony Martinet, Gery Casiez, and Laurent Grisoni. The design and evaluation of 3d positioning techniques for multi-touch displays. In *3D User Interfaces (3DUI), 2010 IEEE Symposium on*, pages 115–118. IEEE, 2010.

[46] Thomas H Massie and J Kenneth Salisbury. The phantom haptic interface: A device for probing virtual objects. In *Proceedings of the ASME winter annual meeting, symposium on haptic interfaces for virtual environment and teleoperator systems*, volume 55, pages 295–300. IOS Press, 1994.

[47] Brad A Myers, Rishi Bhatnagar, Jeffrey Nichols, Choon Hong Peck, Dave Kong, Robert Miller, and A Chris Long. Interacting at a distance: measuring the performance of laser pointers and other devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 33–40. ACM, 2002.

[48] T. Ohnishi, N. Katzakis, K. Kiyokawa, and H. Takemura. Virtual interaction surface: Decoupling of interaction and view dimensions for flexible indirect 3d interaction. In *3D User Interfaces (3DUI), 2012 IEEE Symposium on*, pages 113–116, March 2012.

[49] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BMVC*, volume 1, page 3, 2011.

[50] Optitrack. Optitrack Inc. Optical Tracking through retro-reflective markers.

[51] Michaël Ortega and Thomas Vincent. Direct drawing on 3d shapes with automated camera control. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2047–2050. ACM, 2014.

[52] Wayne Piekarski and Bruce H Thomas. Tinmith-metro: New outdoor techniques for creating city models with an augmented reality wearable computer. In *Wearable Computers, 2001. Proceedings. Fifth International Symposium on*, pages 31–38. IEEE, 2001.

[53] J.S. Pierce, B.C. Stearns, and R. Pausch. Voodoo dolls: seamless interaction at multiple scales in virtual environments. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 141–145. ACM, 1999.

[54] POLHEMUS. Polhemus fastrack^{TM}, a Magnetic 3d Tracker. http://polhemus.com/motion-tracking/all-trackers/fastrak, Last Accessed: July 2015.

[55] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. The go-go interaction technique: non-linear mapping for direct manipulation in vr. In *Proceedings of the Annual ACM Symposium on User Interface Software and Technology* ,UIST, pages 79–80. ACM, 1996.

[56] Ivan Poupyrev, T Ichikawa, S Weghorst, and M Billinghurst. Egocentric object manipulation in virtual environments: empirical evaluation of interaction techniques. In *Computer Graphics Forum*, 17:41–52. Wiley Online Library, 1998.

[57] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014.

[58] OTOY Real-time Path Tracing. http://brigade.otoy.com/, Last Accessed: July 2015.

[59] SE Reed, O Kreylos, S Hsi, LH Kellogg, G Schladow, MB Yikilmaz, H Segale, J Silverman, S Yalowitz, and E Sato. Shaping watersheds exhibit: An interactive, augmented reality sandbox for advancing earth science education. In *AGU Fall Meeting Abstracts*, 1:1, 2014.

[60] Jason L. Reisman, Philip L. Davidson, and Jefferson Y. Han. A screen-space formulation for 2d and 3d direct manipulation. In *Proceedings of the Annual ACM Symposium on User Interface Software and Technology* ,UIST, pages 69–78, New York, NY, USA, 2009. ACM.

[61] Élisabeth Rousset, François Bérard, and Michaël Ortega. Two-finger 3d rotations for novice users: surjective and integral interactions. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, pages 217–224. ACM, 2014.

[62] Richard M Satava. Virtual reality surgical simulator. *Surgical endoscopy*, 7(3):203–205, Springer. 1993.

[63] K. Shoemake. Arcball: a user interface for specifying three-dimensional orientation using a mouse. In *Graphics Interface*, 92:151–156, 1992.

[64] Peng Song, Wooi B. Goh, Chi W. Fu, Qiang Meng, and Pheng A. Heng. WYSIWYF: exploring and annotating volume data with a tangible handheld device. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1333–1342. ACM, 2011.

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

[66] R. Stoakley, M.J. Conway, and R. Pausch. Virtual reality on a wim: interactive worlds in miniature. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI 95 pages 265–272. ACM Press/Addison-Wesley Publishing Co., 1995.

[67] Edward Tse, Min Xin, Viktor Antonyuk, Henry Lai, and Carl Hudson. Making 3d content accessible for teachers. In *ITS*, pages 125–134, New York, NY, USA, 2013. ACM.

[68] William N Venables and Brian D Ripley. *Modern applied statistics with S*. Springer Science & Business Media, 2002.

[69] ViSSee. Flying mouse. http://itunes.apple.com/us/app/flying-mouse/id590741906, Last Accessed: January 2013.

[70] Simon Voelker, Chat Wacharamanotham, and Jan Borchers. An evaluation of state switching methods for indirect touch systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 745–754, New York, NY, USA, 2013. ACM.

[71] J Whyte, N Bouchlaghem, A Thorpe, and R McCaffer. From cad to virtual reality: modelling approaches, data exchange and interactive 3d building design tools. *Automation in Construction*, 10(1):43–55, Elsevier. 2000.

[72] Daniel Wigdor, Hrvoje Benko, John Pella, Jarrod Lombardo, and Sarah Williams. Rock & rails: extending multi-touch interactions with shape gestures to enable precise spatial manipulations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI 11, pages 1581–1590. ACM, 2011.

[73] C. Wilkes and D.A. Bowman. Advantages of velocity-based scaling for distant 3d manipulation. In *VRST*, pages 23–29. ACM, 2008.

[74] Andrew D Wilson, Shahram Izadi, Otmar Hilliges, Armando Garcia-Mendoza, and David Kirk. Bringing physics to the surface. In *Proceedings of the Annual ACM Symposium on User Interface Software and Technology* ,UIST, pages 67–76. ACM, 2008.

[75] C.A. Wingrave, B. Williamson, Paul D. Varcholik, Jeremy Rose, A. Miller, E. Charbonneau, Jared Bott, and J.J. LaViola. The wiimote and beyond: Spatially convenient devices for 3d user interfaces. *Computer Graphics and Applications, IEEE*, 30(2):71–85, March 2010.

[76] Robert Sessions Woodworth. Accuracy of voluntary movement. *The Psychological Review: Monograph Supplements*, 3(3):i, 1899.

[77] Shumin Zhai. User performance in relation to 3d input device design. *ACM Siggraph Computer Graphics*, 32(4):50–54, 1998.

[78] Shumin Zhai and Paul Milgram. Quantifying coordination in multiple dof movement and its application to evaluating 6 dof input devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 320–327. ACM Press/Addison-Wesley Publishing Co., 1998.

[79] Shumin Zhai, Paul Milgram, and William Buxton. The influence of muscle groups on performance of multiple degree-of-freedom input. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 308–315. ACM, 1996.

Appendix A

Relaying sensor data

A.1 Data relay

There are two forms of data being sent from the smartphone to the system that renders the graphics. The smartphone's sensor data and touch events. The relay happens as follows:

There are two different threads on the smartphone that send data to the main system. The first thread serialises the sensor data and puts them on the network in the form of UDP packets. Serialization is done using google protocol buffers[19]. In the main system side, a thread keeps scanning the input buffer for incoming protocol buffer messages,

The protobuf data structure defined in the ".proto" file for the compiler is as follows:

```
option optimize_for = LITE_RUNTIME;

package keimote;

enum MsgType {
ROTATION = 0;
BUTTON = 4;
}

message PhoneEvent {
```

```
required MsgType type = 1;
optional float x = 2;
optional float y = 3;
optional float z = 4;
optional float w = 5;
optional int32 buttontype = 7;
optional bool state = 8;
}
```

List of Figures

List of Tables