Expressive Spatial Interfaces for Scientific Visualization and Artistic 3D Modeling

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

Bret Lowell Jackson

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

Doctor of Philosophy of Science

Advisor: Daniel F. Keefe

July, 2014

Acknowledgements

One of the strongest memories from my first year as a graduate student is John Carlis' description of the PhD process as a "joyful struggle". He could not have been more correct. There have been many ups and downs, successes and long nights in the lab. Ultimately, it has truly been a joyful struggle that would not be possible without the help and support of many people.

First, I would like to thank my funding sources that contributed to this research, including the National Science Foundation, the National Academies, and the University of Minnesota.

I am strongly indebted to the other members of the Interactive Visualization Lab who have served as coauthors and friends. In particular, David Schroeder has been invaluable for his camaraderie, willingness to help debug obscure errors, and expertise in photography and image editing. I have appreciated Dane Coffey's friendship, constructive feedback, and frequent companionship at many conferences. I am also grateful to Volcano Kim; Fedor Korsakov; members of my committee including Victoria Interrante, Barry Kudrowitz, and Joseph Konstan; and others in the University of Minnesota graphics group for provided additional feedback over the years.

To Dan Keefe, I could not have imagined a better advisor. Growing as a graduate student with your constant support and encouragement has been an unforgettably

wonderful experience. You have taught me much about writing, research, teaching, and mentoring. I look forward to many more years of your friendship and advice.

Finally, I would like to thank my family and most importantly my wife, Becky, for her love and encouragement. You put up with late nights, proof-read my papers, and gave energy to many helpful discussions. I could not have done it without you.

Abstract

This dissertation explores spatial human-computer interaction techniques to improve the control and expressiveness of 3D interactions. It investigates the requirements necessary for users to work more effectively with next-generation spatial interfaces, specifically in the context of scientific visualization and artistic 3D modeling where users currently struggle to express complex spatial concepts.

Examples of expressive spatial interfaces are presented and evaluated. In particular, we present new techniques for combining multi-touch with free-hand gestures for navigating visualizations and performing 3D surface modeling operations. Techniques for selecting and filtering volumetric data using lightweight props as well as active force-feedback are also introduced. Additionally, we present a spatial modeling interface for artistic 3D modeling using contextual interpretation of the user's input.

Several conclusions are drawn from these examples. Rich, parallel input and output streams enabled by recent advances in tracking hardware are particularly important for expressive interfaces. Additionally, there is a need for tighter integration of two and three-dimensional data and input. Contextual interpretation of user input enables users to specify more complex 3D concepts. Finally, many spatial tasks require immediate feedback to be expressive.

The primary contribution of this dissertation is a new class of interaction techniques called *Expressive Spatial Interfaces* that advance beyond the limited pointing and rotating interactions common in current-generation spatial interfaces. The techniques presented here can have a powerful impact on shaping the future of expressive spatial human-computer interaction with 3D graphics.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

> *The eye, which is called the window of the soul, is the principal means by which the central sense can most completely and abundantly appreciate the infinite works of nature; and the ear is the second, which acquires dignity by hearing of the things the eye has seen. If you, historians, or poets, or mathematicians had not seen things with your eyes you could not report of them in writing. And if you, O poet, tell a story with your pen, the painter with his brush can tell it more easily, with simpler completeness and less tedious to be understood.*
>
> – Leonardo da Vinci, *Notebooks: The Practice of Painting*

In this quote, Leonardo da Vinci characterizes what we strive to do in scientific visualization and art: convey ideas more easily and in a way that is less tedious to be understood. Da Vinci alludes to our eye's great potential for understanding information. In visualization, we commonly exert this potential by developing computer graphic rendering algorithms that communicate information or enable discovery of new insights. However, although the eyes and ears are important for conveying information, the hands are an equally important aspect that makes the painter so successful. His hands and their interface with the paint brush allow him an amazing amount of control and expressiveness in the strokes that he is able to produce on the canvas.

An artist is able to roll the brush between his or her fingers to paint with the thinner profile or precisely grasp it so that he/she paints with just the tips instead of the side of the bristles. Each of these interactions enable a different type of paint stroke and a different expressive outcome.

Through this physical interface an artist is able to work naturally to express complex concepts. Contrast this expressive control with the traditional interface of artistic computational tools. So often, an artist interacts with these tools through a two-dimensional mouse and keyboard. Instead of immediately and fluidly creating different brush strokes by changing how the brush is held, users must select from predetermined brush profiles in the software.

The lack of expression artists have with computational tools is an obstacle because these tools have advantages over traditional physical media. Artists can implement many creative ideas quickly without consuming expensive materials. They can effortlessly undo, make changes, or return to a previous state of a model, which supports greater creative potential [1]. Computational artistic tools even allow artists to explore ideas that would be impossible to physically produce, that are not constrained by the laws of reality or the bounds of practicality.

In addition to art, scientific visualization is another discipline that benefits from a physical interface. Physical models have been used for centuries as an interface for exploring complex concepts. For example, in the early 19th century, several researchers proposed different structures of the atom and molecules using physical models such as John Dalton's billiard ball model [2] or J. J. Thomson's Plum Pudding Model [3]. Similarly, Watson and Crick used physical ball-and-stick models to explain and test their discovery of the double helix structure of DNA.

In recent years, physical models and prototypes have been replaced by digital models that enable researchers to use computational tools to support their analysis. These

computational tools allow scientists to dynamically explore and query their data. However, with this transition we have lost the act of physically touching, manipulating, and exploring these models, which was often a driving force for generating new insights.

While frequently thought of as polar opposites, this dissertation explores scientific visualization and artistic 3D modeling as application domains because they both require a user to perform complex spatial tasks that are difficult with traditional input devices. Beyond their stereotypical differences, scientific visualization and art actually have many similarities that make them complementary for studying interfaces for computational tools.

Both disciplines are dedicated to questioning and searching deeply for meaning through continuous feedback between thinking and doing. At their simplest, both focus on communication. For artistic modeling, this involves expressing emotions and ideas through deliberate choices in material, techniques, and the 3D forms that are created. For visualization, this involves exploring and querying existing data. Despite their similarities, this fundamental difference in how they communicate, creating vs. querying, makes them particularly complementary for developing a broader understanding of expressive computational tool interfaces. Both disciplines require rich input, but the different ways that the two groups of users need to be expressive impacts the interface design.

1.1 The Potential of Spatial Interfaces for Computational Tools

One way to address the limited expressiveness of current computational tools is by new research in spatial user interfaces. These types of computer interfaces interpret human input that is directly physical or spatial, enabling the interactions to be more complex,

Figure 1.1: The author uses an expressive spatial interface to model virtual sculptures in a virtual reality CAVE environment.

fluid, modeless, and even collaborative.

Consider, for instance, the 3D modeling interface depicted in Figure 1.1 and described in Chapter 6. In this spatial interface, an artist is able to hold his hand out in front of him and by moving it through space, draw complex curves and surfaces. He can duck under the model to see it from a different viewpoint or walk closer to see increased detail. These expressive spatial actions are fundamentally different than using a mouse and keyboard to create art. Artists are able to interact in similar ways to how they would with physical media, and are able to avoid the tedious process of placing and adjusting many vertices or control points.

In addition, spatial interfaces have been shown to have an impact on supporting creativity. In an experiment comparing a tangible user interface for modeling using

3D blocks with a more traditional graphical user interface, Kim and Maher found that tangible interaction changed designers' spatial cognition [4]. With the tangible user interface, manipulation of the blocks was interrupted by short and frequent 3D modeling actions, suggesting that the tangible user interface had the potential for producing sudden insight and creative leaps in design thinking. Related work reveals similar results: physical modeling can overcome gaps in designers' mental models, leading to more ideas [5] and physical props can increase the richness of children's imagination during play and storytelling [6].

Likewise, spatial interfaces have the potential to make computational tools for scientific visualization more expressive. From a young age we develop fine motor control skills that enable us to manipulate objects and predict how they will react when we handle them. Spatial interfaces enable users to draw on these learned experiences from the real world. For scientific visualization this might allow a scientist to spend less time focusing on the interface. Instead, he or she can focus on the data exploration and testing hypotheses.

In addition, spatial interaction in 3D space makes many 3D visualization tasks easier, particularly when coupled with head-tracked stereoscopic virtual reality (VR) which improves understanding of complex 3D data [7]. Consider the many fundamental tasks in exploratory visualization: navigating to different viewpoints, selecting objects or locations in 3D, and placing display widgets such as seed-points and cut planes. Each of these spatial concepts requires multiple operations using 2D mouse input to accomplish them precisely in 3D space.

1.2 Limitations of Current-Generation Spatial Interfaces

Despite the advantages of spatial interfaces discussed in the previous section, current-generation interfaces have not realized their full potential for applications in visualization and art. We characterize current-generation spatial interfaces by the limited potential of the interaction and limited bandwidth between the user's spatial gestures and the digital data. For example, many spatial interfaces (e.g. [8, 9, 10]) use a tracked wand and a ray-casting approach to select objects at a distance and reposition them. The complexity of the 3D data that scientists can select using this style of spatial interaction has barely advanced beyond the "put-that-there" pointing interface [11] that was presented over three decades ago.

While newer scientific visualization interfaces enable more complex interactions, the expressiveness is still limited. For example, Particle Flurries [12] enables scientists to explore flow visualizations by placing their hands in the flow and releasing virtual tracer particles; however, this interaction is still essentially similar to the pointing technique described previously. The expressiveness of the spatial interaction is limited to indicating a single specific point in space.

Similarly, the ability for an artist to be expressive in current-generation spatial 3D modeling interfaces is limited. For instance, many spatial 3D modeling interfaces use instrumented tangible objects, such as blocks [13], but this limits the artist to only modeling shapes that can be easily built with the objects.

1.3 Conceptual Framework and Thesis Statement

In this dissertation, we present a new class of spatial interfaces called *Expressive Spatial Interfaces*. In contrast to the current-generation spatial interfaces discussed previously, recent developments in low-cost depth-sensing camera technology as a way to track the

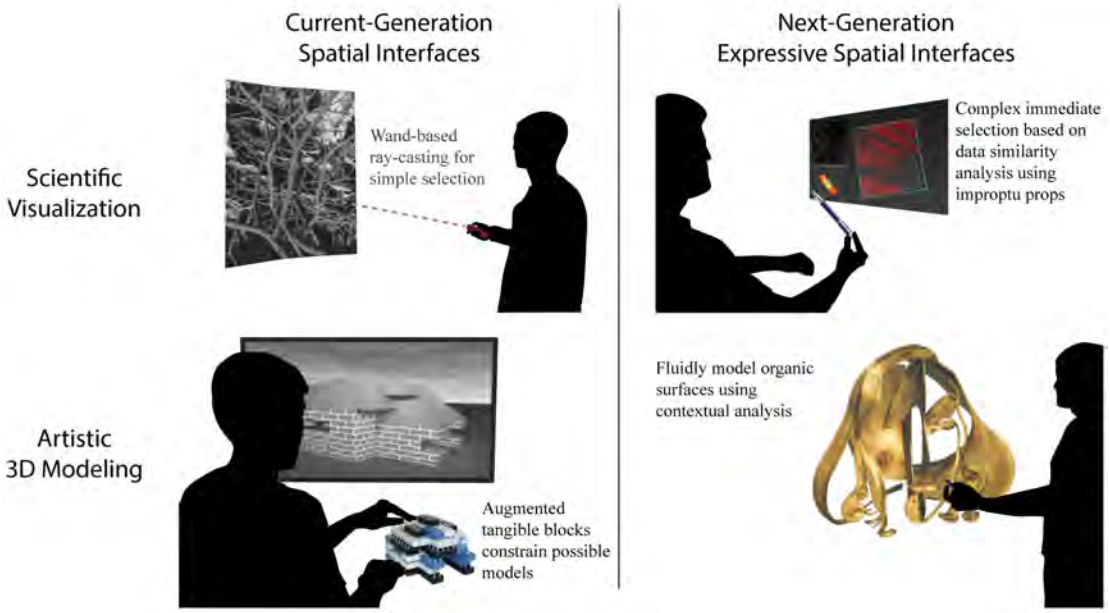

Figure 1.2: Current-generation spatial interfaces (left) limit user's ability to express complex spatial concepts in scientific visualization and artistic 3D modeling. Expressive spatial interfaces (right) are needed to enable more effective interaction with computational tools.

entire surface of the hands (rather than single points of contact) and increased availability of commodity stereoscopic displays have the potential to make spatial interfaces more expressive.

Using these technologies, spatial interfaces are starting to move towards this goal. For instance, newer tangible 3D modeling interfaces (e.g. [14]) make use of depth-cameras to avoid instrumenting individual blocks. We see the potential to go even further in the future, potentially using impromptu props already in the user's environment, an approach that is explored in Chapter 4.

A conceptual framework for spatial interfaces in this style is illustrated in Figure 1.2. We characterize these expressive spatial user interfaces by their ability to move beyond

the pointing and rotating interactions that are common today, and instead enable expressive interactions needed for more complex 3D concepts, like selecting multiple similar values from volumetric data.

Based on this framework, the central thesis of this dissertation is:

Expressive spatial interfaces enable scientists and artists to work more effectively, specifically in the context of scientific visualization and artistic 3D modeling where users currently struggle to express complex 3D concepts.

In the following chapters, we present several novel spatial interfaces applied to scientific visualization and artistic 3D modeling. Through the development of these interfaces we contribute conclusions to guide future researchers in creating the next generation of computational tools that enable scientists and artist to more effectively work with complex 3D concepts.

1.4 Overview of the Dissertation

The dissertation is organized as follows: Chapter 2 presents an overview of related work in spatial graphical interfaces. Chapters 3-5 present examples of expressive spatial interfaces that explore possible design decisions for increasing expressiveness and control of expressive spatial interfaces for visualization and art. In particular, Chapter 3 describes techniques for integrating 3D spatial gestures with 2D mulit-touch using anchored multi-touch gestures for simple 3D modeling operations and viewpoint navigation of 3D data. Chapter 4 presents a technique for interacting with 3D bioimaging data using a passive prop-based interface. Chapter 5 presents an interface for filtering and querying data displayed in a scientific visualization using a force-feedback device for improved control. Chapter 6 presents a spatial 3D modeling interface that explores

how context can be used to make the interaction more controlled. Finally, Chapter 7 presents discussions and conclusions summarizing lessons learned from development of the different expressive spatial interfaces.

2

Review of Spatial Interaction Techniques

This chapter presents a review of 3D spatial interfaces. Since Ivan Sutherland's pioneering work created the first tracked, head-mounted, stereo display in 1968 [15] and James Clark introduced the tracked wand for direct 3D manipulation in 1976 [16], many researchers have explored the potential of combining virtual reality with spatial interaction.

One of the primary issues that limits the expressiveness of spatial interfaces is a lack of control caused by fatigue and muscular jitter that users have when interacting in 3D space. Research has found that including haptic feedback, such as pressing against a multi-touch surface to set slicing planes and 3D splines [17] or using force-feedback devices for 3D drawing [18], can increase control of spatial user interfaces. Many of the interfaces presented in this dissertation explore the use of haptic support to improve control and expressiveness. The dissertation also explores how contextual interpretation of user input can improve control and might serve as a replacement for haptic support in

applications that require greater freedom of expression. As such, we structure our discussion by classifying the work into four categories relating to different haptic techniques and contextual interpretation:

1. Freehand interfaces where the user gestures in the air without support.
2. Tangible interfaces where the user receives passive haptic support from an object or surface.
3. Active haptic interfaces using force-feedback devices.
4. Contextual interpretation of user input for improving control.

Related work from each of these categories is described in the following sections.

2.1 Freehand Spatial Interfaces

Many techniques for freehand spatial interaction have been developed, allowing users to sweep their hands through space while a computer tracks their movements and interprets the input. With just a few movements, we can convey spatial relationships to the computer. This form of full-body scale interaction is expressive and descriptive for applications in art and design. In fact, a recent study found that shape modeling in virtual environments was more creatively stimulating and attractive than under 2D conditions [19]

Unsurprisingly, spatial interaction has been explored for use with 3D modeling. Building on Clark's head-mounted display 3D modeling system [16], Butterworth et al. presented 3DM [20]. The 3DM system allowed for the creation of full 3D surfaces by extruding a polyline through space as the user dragged and twisted the stylus. Our approach to modeling 3D surfaces by sweeping lines, presented in Chapter 6, bares

resemblances to this technique; however, while our approach is controlled by sweeping along additional guides, the 3DM extrusion was subject to issues of control that limited its expressiveness. Users felt empowered by the ability to experiment and make quick modifications, but reported that the lack of constraints made it difficult to create precise geometric shapes.

More recently, Schkolne et al. introduced Surface Drawing [21]. Surface Drawing allows a user to sweep his or her hand above a horizontal display to create surfaces. The profile of the surface can be changed by adjusting the hand posture. Sharing motivations with the work discussed in the next section, the interface also uses tangible props to make interaction more intuitive. For example, a pair of kitchen tongs could be used to "grasp" a virtual model and move it around.

CavePainting by Keefe et al. [22] is related to Surface Drawing and serves as inspiration for the modeling interface presented in Chapter 6. Rather than using a horizontal display, CavePainting uses a virtual reality CAVE, enabling an artist to fully engage his/her body in the painting process. By using a tracked paintbrush, an artist can use the system to draw 3D ribbons and tubes, layering them to create 3D art that is reminiscent of an artist creating a 2D painting by layering brush strokes on canvas. This form of working is expressive for drawing 3D lines, but does not lend itself as well to creating precise surfaces for 3D models.

The FreeDrawer interface by Wesche and Seidel [23], attempts to solve this issue by drawing a series of tubes to serve as a curve network that defines the boundaries of surfaces. Although this approach lets an artist create arbitrary surface shapes, the initial curve drawing still suffers from the same lack of control as CavePainting because both use unsupported freehand input. Our modeling approach is based on this technique of drawing surface boundaries, but we refine the way that these are created so that artist can model precise curves.

While spatial interfaces using these types of freehand gestures are common in art applications, their use is less prevalent for scientific visualizations that require precise control [24]. However, there are some exceptions. A notable early example is the work of Bryson and Levit [25] that enabled scientists to explore flow visualizations by using a tracked glove. Moving their hands through space, users could release virtual "smoke" to trace flow features. Particle Flurries [26] later extended this idea.

Predominately, freehand interactions in scientific visualization focus on techniques for navigation, selection, and manipulation. 3D operations are areas where reliable 3D input can have a major positive impact if it can be controlled enough to be expressive. For instance, several researchers (e.g. [27, 28, 29, 30]) have explored the potential of manipulating virtual objects using freehand gestures in the air above a multi-touch table. While these techniques extend our ability to interact naturally and expressively with virtual objects, they are not very precise. The Anchored Gestures technique presented in Chapter 3 has nearly the same potential for rich expressive interaction as these freehand gestures. However, motivated by the work of Kattinakere et al. [31] that found that resting the hands on the surface improved control in steering tasks above the surface, our gestures are anchored with touch contacts to the surface. Additionally, these anchored touch points allow fluid movement between traditional 2D multi-touch gestures and more expressive 3D gestures. This is similar to Marquardt et al.'s [32] idea of the continuous interaction space; however, the transition is even more fluid with our gestures because the hands never need to leave the surface.

This fluidity and immediacy of the input is a defining characteristic of next-generation spatial interfaces. Current generation interfaces, particularly in the games industry, tend to make use of categorical gestures. For example, SwordPlay [33] is a game in VR where users can cast spells by drawing a specific sequence of paths through the air. The user has to complete the entire action before there is any result.

One way to improve immediacy is through physics simulation. HoloDesk [34] is a particularly compelling approach that couples physics simulation with a depth camera. By tracking the entire surface of the user's hands, the simulation enables more natural manipulation of virtual objects because multiple points of contact can affect the results. Next-generation spatial interfaces will likely use this richer, immediate style of input.

In addition to manipulation, spatial interfaces have also been used for selection in scientific visualization. For instance, consider the task of selecting bundles of neural-fiber tracts in a brain visualization, or groups of streamlines in a flow visualization. One approach for selecting these complex 3D data is by sketching 3D lassos around the objects to be selected [35]. Although the level of control needed for this task can be increased through haptic assistance [36], another approach increases control for freehand input via a level of indirection. In Drag Drawing [18, 37] a user drags a virtual brush with a virtual tow-rope behind the tracked wand. The tow-rope acts as a filter, minimizing jittery motion for the brush. Using this technique, a scientist can precisely draw a 3D lasso for selection.

The work presented in this dissertation is inspired by this sort of compelling example of how spatial input expressiveness can be improved through smart interface decisions. In addition, we are particularly excited by the potential of extending these spatial interfaces with data-driven context, such as the selection interface in the 2D CINCH [38] system that lets users draw the shape of a neural fiber to select similarly shaped fibers. We use this work as a foundation for the Force Brushes interface presented in Chapter 5. Using contextual analysis of the user's spatial input will let us move beyond current-generation spatial interfaces to make them applicable for more complex spatial tasks.

2.2 Tangible Spatial Interfaces Using Passive Haptic Feedback

There is a long history of research into the use of tangible props as user interfaces. It is now well established in the 3D user interfaces and human-computer interaction communities that even passive haptic props typically provide great advantages over freehand interfaces in terms of spatial understanding and control over 3D operations.

Early work in this area focused primarily on the use of pens and palettes [39, 40]. For example, the 3-draw system [39] enabled an artist to draw 3D curves using a tracked pen. Others (e.g. [41]) have looked at combining 2D and 3D pen based input in VR environments.

More recently, the ModelCraft framework proposed by Song et al. [42, 43] combined folded paper props and a pen interface to capture digital annotations on the prop surface. While this paper-based system worked well for annotating architectural models with mostly flat surfaces, it is less well suited to the complex organic geometry intrinsic to many scientific datasets.

Using this work as a foundation, new research explores the potential of prop-based interfaces that are made possible (and increasingly practical) by emerging rapid prototyping technologies. Accurate physical 3D data printouts from rapid prototyping machines have been used as interaction props for 3D data visualization (e.g. for molecular biology visualization [44], cartographic GIS visualization [45], and infovis [46]).

Others [47, 48] have looked at combining spatial pen-based input with rapid prototypes. For instance, Kruszyński and van Liere [48] reported on requirements for tracking, calibration, latency, and printing for an application that uses a tracked pen to interface with tangible props printed from 3D coral data. This system was motivated in part by the need to analyze the precise 3D shape of specimens collected from coral

reefs which are too delicate to handle extensively. The 3D shape of the corals are particularly complex, thus, visualizing them as a physical rather than virtual object aids in understanding. The particular interface developed in this work was an interactive measurement system. A coral prop was held in one hand while a stylus held in the other was used to indicate precise locations on the surface of the coral, from which distance and thickness calculations were made.

While these rapid prototypes can provide a very realistic display of the data for interaction, they are not transferable to different datasets. New models must be printed for each set of data, which can slow down the scientific workflow. Motivated in part by this limitation, many tangible interaction props are more abstract (e.g. bricks as a graspable proxy for various objects [49, 50]). A study [51] by Colin Ware showed that the prop does not need to have the exact shape as the virtual object to facilitate interactions.

This finding is apparent in Hinckley's seminal work in this area, applied to scientific visualization. Hinckley used simple props tracked in space (a doll's head, a clear square of plastic) to help doctors fluidly explore and slice through brain imaging data [52].

The cubic mouse [53] was also used for navigating visualizations. Shaped like a cube with three cylindrical rods of a thickness similar to a pen, it was held in a user's hand. By twisting or translating one of the rods relative to the center of the device, the user could navigate and manipulate the virtual world.

In addition to visualization, tangible props have also been used for artistic interfaces. Balakrishnan et al. [54] introduced a flexible bend and twist sensitive strip called ShapeTape. By manipulating the strip, a user could create precise curves and surfaces through extrusion, lofting, and revolves. Although precise shapes could be created by using spring steel rods and jigs to constrain the movement, users found that the interface suffered from the "iron horse" effect. Named for the first automobiles that were

controlled and even shaped like a horse, the iron horse effect occurs when new designs mimic the properties of an analogous physical object too closely. ShapeTape suffered from being unable to create some input curves because the inherent properties of the material limited the bend curvature. It also was difficult to move from one workstation to another. While the tangible input gave the users increased control, it limited some of the advantages of working with a computational tool, such as the ability to ignore physical properties like gravity.

Similar physical interfaces have been used for modeling. Sheng et al. [55] presented a clay-like modeling system that uses a foam sponge as a physical proxy for the model. By tapping, twisting, sliding, and pressing on the surface of the sponge, the artist could create models. Unfortunately because of the limited tracking resolution and the way that the sponge responds to input, most of the models created using the system have a blobby topology that is characteristic of many of the modeling systems in this style.

Tangible props for visualization are not just limited to changing the physical input device, but also changing the physical display. For instance, Konieczny et al. [56] used a flexible tracked projection screen that the user could move and flex to see a virtual slice of 3D volumes.

Touch-sensitive display interfaces also share some similarity to our work in that they enable fluid, natural styles of interaction with data [57]. For example, CubTile [58] uses a cube with five out of the six sides covered with a multi-touch surface to leverage the 3D spatial layout in overcoming the 2D limitations of traditional touch screens. Other work has looked at changing the touch surface into a sphere. In particular, Grossman et al. [59] define several interaction techniques where the angle the finger makes with the surface, for instance pointing at the sphere vs. parallel to the display, can be used for different modes. This is similar to our approach for Anchored Gestures in Chapter 3.

2.3 Techniques Using Active Haptic Feedback

Haptic force-feedback has previously been used as a technique for interacting with medical, art, and scientific visualizations. For example, the nanomanipulator system [60] allowed scientists to feel microscopic surface details examined through a Scanning Tunneling Microscope. Similarly, Avila et al. [61] used haptic devices to feel the shape of a neuron model produced with a confocal scan.

Predominately haptic feedback for visualization complements the visual representation of the data, while not suffering from problems of occlusion and clutter. The Force Brushes interface, presented in Chapter 5, builds upon previous work (e.g. [62, 63, 64, 65, 66]) that used haptic constraints to force the input pen to follow integrated streamlines through the flow vector field. Our use of haptics shares similar motivation to that of lundin et al. [64], which allowed users to trace the path of a vortex in CFD data. However, rather than simply feeling the flow direction, our interface leverages the haptic forces to provide more controlled selection of regions of interest, as well as to provide input to control the visualization.

Using haptic feedback for controlled input shares similar motivation to the springs and constraints for 3D drawing by Snibbe et al. [67] and Drawing on Air [18], which used non-realistic forces to allow for more controlled input.

2.4 Interpreting User Input Based on Context

Several interfaces explore how user input can be interpreted based on the context of the interaction. For example, the Elasticurves interface [68] neatens drawn 2D strokes based on the drawing inertia and stroke dynamics. Similarly, the ILoveSketch system by Bae et al. [69] allows a user to more clearly define sketched lines by automatically averaging several strokes drawn in close proximity. It also supports automatic connection of two

curves if the second one is drawn tangentially to an existing curve. These examples primarily use previous user input as context for the analysis. In contrast, our approach presented in Chapter 6 uses additional data from 2D imagery, as well as previous input, to interpret the artist's intent with spatial input.

Drawing with the Flow by Schroeder et al. [70] is one example in this style that also uses additional 2D data as context. Inspired by the contextual "settling" of drawn strokes in ILoveSketch, this 2D interface allows artists to draw flow visualizations, while gradually settling the drawn strokes to maintain accuracy with the underlying flow data. Our approach uses a similar settling of the artist's input but uses contours from 2D imagery rather than a vector field to help guide and refine the resulting curves.

The modeling interface in Chapter 6 is closely related to the work by Tsang et al. [71] that allowed users to create 3D wire models by drawing 2D curves on orthogonal planes. Concept sketches could be pasted on the drawing plane that then guided the users input curves by attracting them towards curves in the sketch. Our approach builds on this technique, differing in several ways.

First, Tsang et al.'s interface used 2D input drawn on a Wacom tablet. In contrast, our spatial interface requires a different technique for indicating 2D reference curves in the inspirational imagery. Our technique also enables the artist to to create full 3D curves inspired by the imagery, rather than just 2D curves.

2.5 Summary of Related Work

In summary, unconstrained freehand interfaces allow for the most freedom of expression, but because the user's hands are unsupported they can lack precision and cause fatigue. At the other end of the spectrum, fully active haptic force-feedback devices can provide extremely precise control for visualization and artistic tasks. However, they are

expensive, unportable, and of limited resolution in the sense that they most often track and provide force-feedback for a single point such as a pen tip. Tangible interfaces can provide improved control while maintaining a high level of expressiveness. However, tangible interfaces are not without their own limitations, for instance being too application specific. Recent work using context to interpret user input shows promise for improving control as well. In the following chapters, we explore several hybrid interfaces that combine freehand input with haptic feedback and contextual interpretation, and we discuss how their control and expressiveness can be improved.

3

Anchored Above the Surface Interaction for 3D Modeling and Navigation

[1]From a rock climber gripping a hold to a musician plucking strings, our hands are extremely versatile, allowing us to interact with the world in amazing and expressive ways. The previous chapter described several approaches for integrating this type of spatial interaction into computational tools. Touch table interfaces are similar to this previous work in that they enable fluid, natural styles of interaction with data [57], and research has shown that the passive haptic feedback of placing one's fingers and/or hand against a surface facilitates accurate gestural interaction with data [17]. Multi-touch interfaces are now beginning to take advantage of this ability and, in doing so, revolutionize the way we interact with computers, enabling much richer and expressive

[1]This chapter is based on work published in the Proceedings of the 2012 Graphics Interface Conference [72]

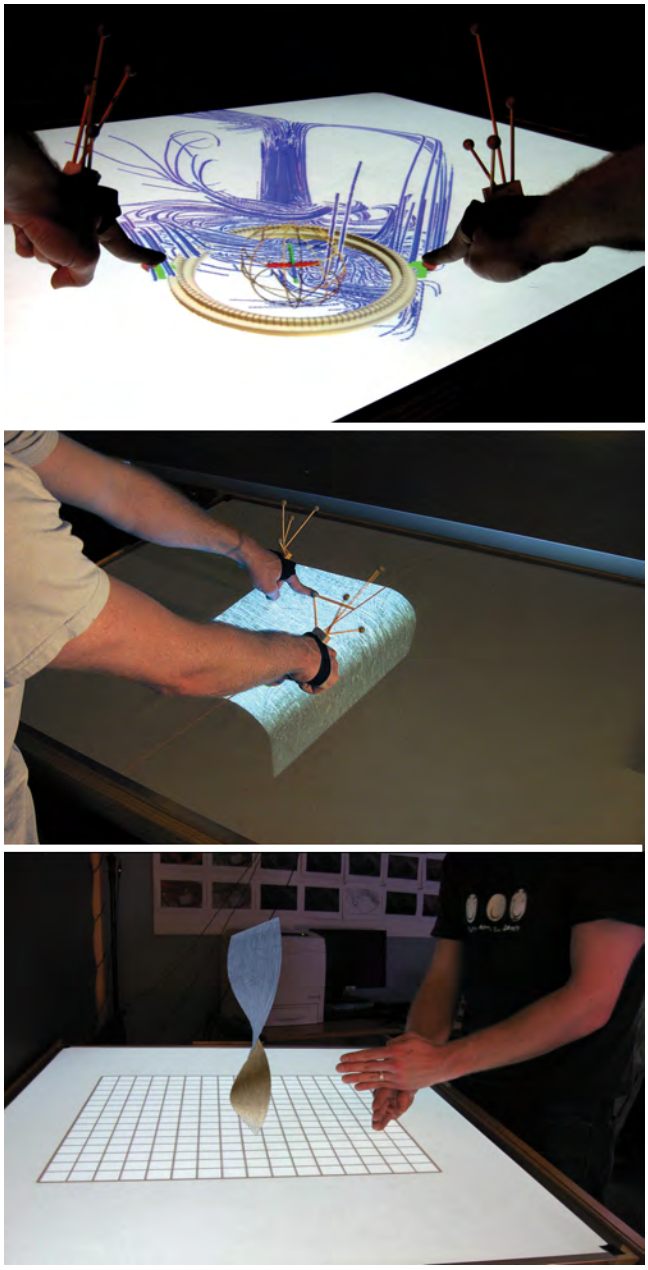

Figure 3.1: Anchored multi-touch gestures for 3D head-tracked stereoscopic applications. Top: Pivoting the fingers above the surface rolls a 3D visualization. Middle: A user bends a mesh while simultaneously specifying the curvature of the bend. Bottom: A user twists a 3D mesh by placing his hands on top of each other and rotating. (A digital rendering is superimposed to illustrate the stereoscopic effect of the display.)

input than with traditional mice and keyboards. In the future, we expect the higher-bandwidth input, somesthetic information, and collaboration support that multi-touch provides to continue to drive adoption and to open up new applications for this style of expressive human-computer interaction. To facilitate this expansion, in this chapter we explore how multi-touch input might be usefully complemented by 3D spatial input from the hands above the touch surface.

Our work is motivated by 3D applications, such as scientific visualizations and 3D modeling. Traditional multi-touch interactions are less successful in these applications, in part because the mapping from 2D surface input to 3D action is less clear than in the purely 2D case. This problem is compounded in head-tracked stereoscopic environments. In stereoscopic environments 3D objects viewed on a multi-touch table can be made to appear as though they float above the table or under the table. Even the traditional translate-rotate-scale gestures used so often in 2D multi-touch environments break down in this situation. Scale, for example, might be interpreted either as scaling the 3D scene or as translating the 3D scene up and out of the table toward the viewer's eyes. Extending multi-touch interfaces to include above-the-surface 3D inputs may provide the needed increased richness and expressiveness in order to address situations such as these.

Although some work has studied free-hand interactions in the air above a multi-touch surface, as discussed in the previous chapter, we explore the interactions space *just* above the surface, where the shape or movement of the hand is captured simultaneously with touch inputs on the surface. We call this type of interaction *anchored*, as the gesture is performed while maintaining an anchoring touch to the surface. This allows us to leverage the benefits of using the surface as a passive haptic support, potentially improving stability and control as well as lowering fatigue when compared to freehand 3D input in the air [31]. In the applications we demonstrate, users move fluidly between

traditional 2D gestures on the surface, and anchored 3D gestures above the surface.

To formalize our investigation and aid future researchers, we introduce a taxonomy of possible anchored gestures. Within the taxonomy, anchored gestures are classified based upon both physical properties (i.e., the way that gestures can be anchored to the surface) and movement properties (e.g., whether the fingers are moving in coordination).

We present two applications of anchored multi-touch: (1) an interface for navigating within 3D datasets, and (2) a surface bending interface for freeform 3D modeling. Both of these applications, shown in Figure 3.1, use anchored gestures to interact with 3D content on a head-tracked stereoscopic multi-touch display. In the following section, we discuss the related work for multi-touch interfaces and how they might be combined with spatial gestures. We then present the anchored gesture applications, and finally, we conclude the chapter with a discussion of lessons learned and future research directions.

3.1 Related Work

The interface presented in this chapter combines spatial gestures, similar in style to the freehand spatial interfaces presented in Section 2.1, with 2D multi-touch interaction. Here, we introduce related work that explores how multi-touch interaction can be made more expressive.

3.1.1 Next Generation Multi-Touch

One goal of our work is to expand the expressiveness of multi-touch input to enable interactions that are closer to how we use physical objects in the real world. Multi-touch interactions are starting to move beyond using touches as 2D points similar to a mouse cursor, and instead take into account the touch contact shape to improve the expressiveness of the input. For example, in *RoomPlanner* [73] multiple objects can be moved at

once by touching the surface with the side of a hand, much the same way people would quickly reposition multiple objects sitting on a real table. Similarly, in *ShapeTouch*, Cao et al. [74] explored using the contact shape to model virtual contact forces. Although sensing touch contact shape expands the number of possible interactions, the input is still only sensed in the plane of contact. Our work looks to move beyond this contact plane to take into account the entire posture of the hand(s) above the surface. By changing the posture of the hand *just above* the surface, we are able to be ever more expressive.

Some work has explored the type of 2D interactions this would enable. For instance, Wang et al. [75] discuss how the finger pointing direction (yaw) could be used for pie menu selection or precise selection tasks. Additionally, this type of input could be used to estimate the hand occlusion region to enable dynamic placement of content, or to infer the position of multiple users around a touch surface. Although these techniques are starting to use the posture of the hands, the sensing is done only by inferring the finger direction from touch contact shape. Our approach makes use of low cost spatial tracking systems to enable a wider variety of interactions, for instance the ability to twist a finger around a touch point, which cannot be captured from touch contact shape alone. Additionally, we look beyond the types of flat 2D interactions like menu choice, to how the posture of the hands could be used in 3D interactions where traditional multi-touch has been less successful.

Recent advances in tracking technology have made these types of above-the-surface interactions possible. For example, the Z-touch [76] hardware by Takeoka et al. uses infrared laser planes to create a depth map of objects near the surface of the table and can sense the posture of the hands. We believe that this type of hardware opens up new and exciting opportunities for different types of interaction such as the anchored gestures we explore in this paper. Although their main contribution was the hardware, Takeoka

et al. also describe several interesting interaction techniques for drawing, controlling map zoom level, and Bézier curve control by varying the finger angle relative to the surface of the table. Like Takeoka et al., our depth camera-based tracking approach does not require users to wear additional hardware. An added advantage of our system is that recent advances in the entertainment market have made depth-sensing cameras, such as the Microsoft Kinect, available at low-cost.

3.2 A Taxonomy of Anchored Gestures

To better understand the design space that encompasses anchored gestures, we designed an anchored gesture taxonomy. Wobbrock et al. [77] provide a taxonomy of traditional multi-touch gestures, characterizing gestures based on their form, nature, binding, and flow. Freeman et al. [78], focusing on teaching users how to perform gestures, extend the form dimension by adding registration pose, continuation pose, and movement. Our taxonomy is complementary to this approach, extending the form category to encompass the design space of anchored above the surface gestures.

We extend the form of the gesture along two dimensions: the *physical properties*, and *movement properties* of the gesture. See Table 3.1. The physical properties take into account information about the number of fingers, similar to the registration pose dimension of Freeman et al. [78]. We also consider gestures that use more than one hand and classify the type of surface the gesture is anchored against.

The movement dimension denotes properties of the action, such as whether the movement is pivoting around the anchor point, or simply twisting relative to it. It also encompasses the idea of correlated movement between multiple fingers or multiple hands performing the gesture.

In the sections that follow, we describe two applications of anchored multi-touch

Table 3.1: A taxonomy of anchored multi-touch gestures.

Physical Properties	Touch Contact	Multiple Fingers Single Finger Area Contact
	Number of Hands	Bi-Manual Single Hand
	Anchoring Surface	Display Surface Tangible Prop Body Part
Movement Properties	Action Type	Pivot Twist
	Correlation	Correlated Anti-Correlated None
	Anchor Position	Stationary Mobile

interfaces that explore and demonstrate some of the possible gestures identified in Table 3.1. Before describing each application in detail, we begin with a brief introduction to the multi-touch hardware utilized in our investigation.

3.3 Hand Tracking

This section introduces two methods for sensing 3D motion just above the table surface. The first method utilizes commercially available optical motion capture technologies; the second uses a depth camera and includes a discussion of our extension of an existing finger tracking algorithm to work with anchored gestures.

Both approaches are implemented on the stereoscopic display table pictured in Figure 3.1. The display is head-tracked, so as the user moves his head around the table, the perspective projection of the scene is updated accordingly. A 7-camera OptiTrack system (NaturalPoint) is installed in the room to support head-tracking via reflective

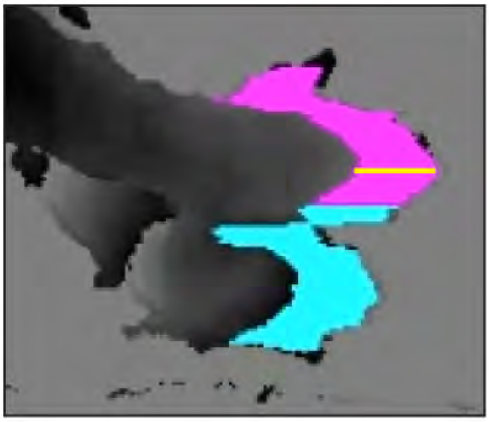

Figure 3.2: Depth image used to track hand motions. The hand regions detected are colored in magenta and cyan. The segment identified as the the tip of the hand by the first step in the algorithm is colored yellow.

markers.

The first method for tracking hand gestures also makes use of the Optitrack system. In this method, a constellation of reflective markers is affixed to the back of the hand using a thin elastic band (see top and middle images in Figure 3.1); the system tracks these markers, reporting 6 degrees of freedom tracking data for each hand.

There are two clear limitations to this technology. First, it requires wearing extra hardware on the hands. Second, it treats each hand as a rigid body, i.e., it tracks the whole hand, not individual fingers. Both of these limitations should be able to be solved using current low-cost commercially available depth cameras, such as the Microsoft Kinect. In the future, we expect that anchored gestures will be reliably captured via this type of depth-sensing technology or perhaps a multi-layer strategy built into the frame of the multi-touch table, as in the work of Takeoka et al. [76].

Our work takes a first step in this direction; the surface twisting interface uses a 640 $\times$ 480 pixel depth image collected at 30 Hz to track anchored gestures of the hands.

This particular gesture uses two hands, one placed on top of each other, as in Figure 3.1 bottom. The tracking algorithm follows a strategy inspired by a recently introduced finger tracking algorithm [79], with extensions here to identify the larger hand features and segment the two hands. First, the horizontal gradient is extracted from the depth image. Then, each row is searched for a pattern consisting of a high-magnitude gradient value followed by a smooth region at least 5 cm long. Since our camera is placed off to the side of the multi-touch table, we are able to identify the rightmost (in camera space) section matching this pattern as the tip of the hand.

With the hand tip identified, a smoothing operation is applied to the depth values, and the region of the image considered to contain the hand is grown by adding neighboring rows of pixels that exhibit similar depth patterns. Since the hands touch each other, after this step, the selected region will include both the top and bottom hands. The two can be separated by greedily growing from the top and bottom of the hands region, using row-wise difference to determine the similarity between each pair of rows. Finally, once the two hands are identified, the average horizontal gradient is calculated for each hand. The arctangent of these gradients is the rotation angle for each hand.

3.4 Application 1: Anchored 3D Modeling

The first application we explore is 3D modeling. There is a long history of research in free-form modeling tools using touch, pen, and freehand input (e.g., [80]), yet creating controllable, natural interfaces for specifying complex 3D modeling operations remains a major research challenge. Our work focuses on bending and twisting 3D surfaces which have inspired related prior work in both 2D and 3D interfaces [54, 81].

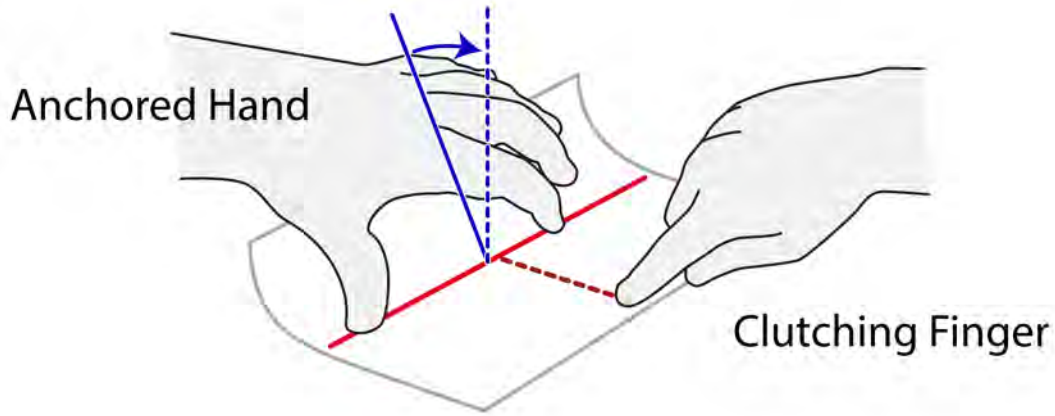

Figure 3.3: Surface bending by pivoting the hand around the anchored thumb and index finger. A clutch finger on the second hand locks the bend axis in place and specifies the amount of curvature.

3.4.1 Bend Gesture

The first interaction we explore deforms a mesh or surface by bending it (see Figure 3.1 middle). The gesture is performed by anchoring the thumb and index finger of one hand to the surface (Figure 3.3 left hand). These two multi-touch anchor points define the center-line axis of the bend, shown as a solid red line. A finger on the other hand (Figure 3.3 right hand) is then used as a clutch to lock the bend axis in place. By pivoting the anchored hand around the anchoring fingers, the user bends the mesh either up or down. As the gesture is performed, the user may slide the clutching finger towards or away from the bend axis to vary the curvature of the bend.

The direction and amount of bend to apply to the 3D model is calculated from tracking the movement of the hand above the surface, in this case, using the multi-camera optical tracking system, and interpreting the 3D motion relative to the multi-touch contacts on the surface. On the touch surface, the two points of contact on the anchored hand (thumb and index finger) indicate the axis about which the bend will be

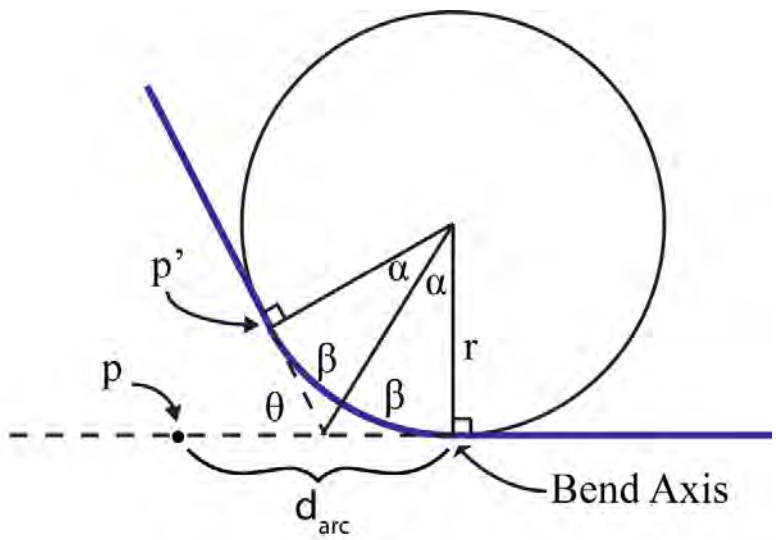

Figure 3.4: The surface is bent as if wrapping it around a cylinder.

applied. By transforming these 2D inputs to the same 3D coordinate system used by the 3D tracking device, we can define a 3D vector that specifies the direction from the thumb touch point to the index finger touch point. We will call this vector $\hat{b} = (b_x, b_y, b_z)$. After the clutching finger from the second hand is engaged, the 3D model will be bent around this axis in proportion to the amount of hand movement about the axis. As the hand is pivoted, the change in orientation relative to its initial pose is recorded and the raw 3×3 rotation matrix is decomposed into an axis of rotation, $\hat{a} = (a_x, a_y, a_z)$, and angle of rotation about that axis, ϕ. The bend angle, θ is calculated as the rotation of the tracker, ϕ, scaled by the projection of the tracker's rotation axis onto the bend axis.

$$\theta = (\hat{a} \cdot \hat{b})\phi \tag{3.1}$$

The next step is to deform each vertex on the 3D model to fit the bend specified. Intuitively, we think of this as rolling the mesh around the form of a cylinder, as pictured in Figure 3.4. The bend stops at the point where the tangent to the cylinder and the

original surface are θ degrees apart. This tangent point is indicated by p' in Figure 3.4, which is also the point to which the point p in the original 3D model will be deformed after the bend. Therefore, there are two cases for how the vertices should be deformed. Vertices that lie between point p and the bend axis on the original surface are deformed to fit the cylinder curve, whereas vertices further away from the bend axis are deformed to extend along the tangent line. We first describe how the distance from p to the bend axis is calculated for use in determining which case a vertex falls into, followed by how vertices in each case are deformed.

The point p is calculated so as to make the distance from the bend axis to p equal the distance along d_{arc}. Given a cylinder of radius, r, and bend angle, θ, d_{arc} is calculated as follows.

$$d_{arc} = 2\alpha r \tag{3.2}$$

The angle α varies with the bend angle θ which changes the tangent to the cylinder. To determine α, the angle β is first calculated assuming that the original surface is flat.

$$\beta = \frac{\pi - \theta}{2} \tag{3.3}$$

Once β is known, α can be calculated by making use of the triangle property that interior angles add to π.

$$\alpha = \frac{\pi}{2} - \beta \tag{3.4}$$

To move each vertex, v_i, the distance, d_i, is calculated to the closest point on the bend axis. In the case where $d_i \leq d_{arc}$, the deformed vertex will lie on the surface of the cylinder. To perform this mapping, v_i is first translated to the closest point on the bend

axis and then rotated around the center of the cylinder by an angle, ρ where:

$$\rho = 2\alpha(\frac{d_i}{d_{arc}}) \tag{3.5}$$

In the second case where $d_i > d_{arc}$, the deformed vertex will lie on the tangent extending beyond the cylinder. The mapping is accomplished by first translating v_i by the vector from p to p' to make sure that the flat section meets the cylinder without stretching the mesh. v_i is then rotated around the closest point on the bend axis by θ.

Using these equations, the mesh can be bent into or out of the table surface by positioning the cylinder center either above or below the bend axis. The curvature of the bend relates to the radius of the cylinder, with a larger radius creating a gentle bend or a smaller radius creating a sharper bend. The distance of the clutching finger to the closest point on the bend axis sets the cylinder radius.

The pivoting action of the fingers maps naturally to bending, much the same way a user might bend a piece of paper, while using the display surface as an anchoring surface provides the stability necessary to specify precise changes in the bend angle. The anchored technique integrates seamlessly with more traditional multi-touch inputs, for example, the touch by the clutching hand. This touch point can be moved on the surface during the bending operation to specify curvature, using a style of bi-manual input that is characteristic of many successful multi-touch interfaces.

3.4.2 Twist Gesture

The second 3D modeling interaction we explore deforms a mesh or surface by twisting it (see Figure 3.1 bottom). The gesture is performed using two hands. First one hand is placed (anchored) onto the table on its side (with little finger down and thumb up in the air). Then, the other hand is place on top of the first, again in a sideways orientation.

To twist the surface, the hands are spun around the natural axis that forms between them when held in this position.

In this interface, the above-the-surface gestures are captured using the depth camera and algorithm described in Section 3.3, which outputs the angle of rotation around the up axis for both hands. The motion of the bottom hand specifies the angle of twist for the bottom of the 3D mesh, and the top hand specifies the angle of twist for the top of the 3D mesh. These angles are updated each frame of the interaction based on the relative motion of each hand in the time since the previous frame. The raw rotation data for each frame is converted to an axis of rotation and an angle, which is transformed, as in Equation 3.1, to calculate the angle of rotation around the z-axis (pointing out of the screen) only. This rotation is then applied to each vertex. For each vertex, the angle to rotate around the z-axis is specified via a smooth linear interpolation between the twist angles specified for the top and bottom of the mesh.

$$\theta_i = \alpha\theta_{top} + (1 - \alpha)\theta_{bottom} \tag{3.6}$$

where α is the normalized distance from v_i to the top most vertex.

We were excited to explore this particular interface for the twisting modeling operation because it demonstrates how different anchored multi-touch interfaces can be from more traditional multi-touch. The use of an area contact in itself is relative new in the multi-touch community (e.g., [82]). As an input to a modeling tool, compared with a completely freehand (3D input in the air) alternative, it is immediately clear when experiencing this interface that passive haptic feedback provided by having the hands anchored to each other and the surface makes even small angular inputs controllable. In contrast, simultaneously specifying two angles of rotation (including holding one constant while making small adjustments to the other) would be nearly impossible to do

in a freehand 3D interface, given the muscular jitter and tracker errors commonly associated with these interfaces. Thus, although the application to 3D modeling interfaces presented here is limited to a demonstration of two specific modeling tools, we see a bright future for operations such as these to be extended into a full modeling application that could spur creativity by changing the way that 3D modelers create geometry, using more physical and sculptural digital interaction.

3.5 Application 2: Anchored 3D Navigation

The second application of anchored multi-touch that we explore is navigation within 3D scenes viewed on stereoscopic multi-touch tables. To date, little work has been done in the area of coupling multi-touch input with stereoscopic displays (notable exceptions are [83, 84]); designing improved interfaces for navigating through 3D datasets remains an important problem in this area of research.

In the following sections, we describe a complete interface for 3D navigation using anchored multi-touch. Traditional 2D gestures (pan, rotate, scale) are integrated into the interface. The anchored gestures address the specific problem of rotating and translating a 3D scene in and out of the table surface. Note that these interactions tend to be performed via indirect gestures, such as mapping an up and down motion on the surface to a translation in and out of the screen [85], when only 2D multi-touch input is available.

3.5.1 2D Navigation Gestures

Several 2D multi-touch gestures are now commonly used to position, scale, and rotate objects in the plane of a touch surface. We take these as a starting point for the navigation interface described next. Movement of a single touch point translates objects

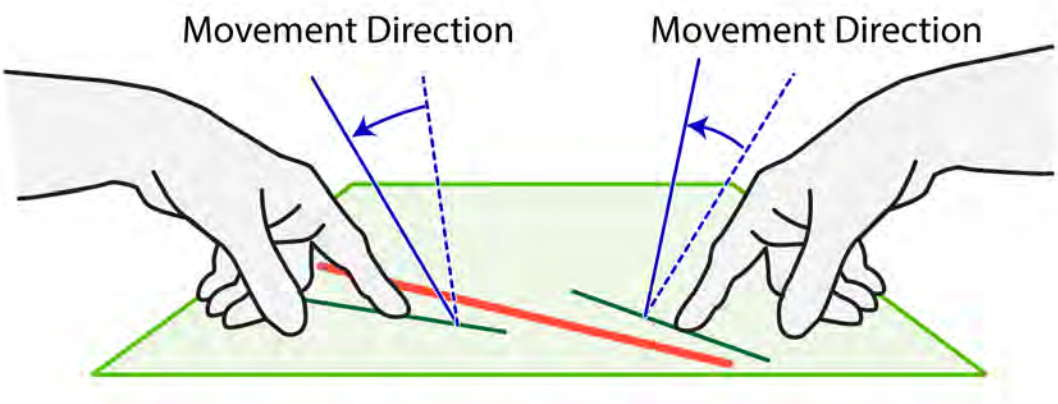

Figure 3.5: In anchored pivot rotation, the visualization space is rotated by pivoting the hands in a similar direction. The axis of rotation (in red) is derived from the axes of rotation for each hand (in green).

in the xy-plane of the table. Two touch points are used to indicate a rotation about a vertical axis out of the table and scale. Using the familiar metaphor, the points directly under the user's fingers appear to stick to the objects being manipulated during the interaction.

Unlike two-dimensional displays where a scale looks the same as a z-translation from the camera's viewpoint, head-tracked stereoscopic environments are different. A scale action increases the size of the rendered graphics, whereas a z-translation makes them appear as if they are rising up out of the touch surface. For this reason, below, we define a separate anchored gesture for z-translation.

3.5.2 Extensions for Anchored Pivot Rotation

To specify rotation around the x and y-axes on the table surface, we present a bi-manual interaction. The user touches the surface with a finger tip from each hand. See Figure 3.5. By keeping the touch point still on the surface (anchored) and pivoting his hands together he can specify a rotation in any direction. The fingers can be thought

of as mini joysticks.

To allow direct control, the rotation is limited to have the center of rotation lie on the touch surface, halfway between the fingers, and the rotation axis is constrained to lie within the plane of the table. Allowing explicit control of the center of rotation is important when viewing datasets that do not have an inherent center. (This feature is not included in most other multi-touch rotation techniques, but was specifically requested following the user study presented in [85].)

Control over the rotation is also achieved via a filtering mechanism. This gesture uses the motion of both hands, with each hand pivoting in the same direction and by similar amounts. The two motions are averaged to determine the final rotation. For each rendering frame, and for each hand, a vector, $\hat{v}_t$ is created from the touch position on the surface to the tracker position. $\hat{v}_t$ is compared to the previous frame's vector, $\hat{v}_{t-1}$ to find the rotation axis, $\hat{a} = (a_x, a_y, a_z)$ and angle, ϕ.

$$\hat{a} = \hat{v}_t \times \hat{v}_{t-1}, \text{ then projected onto the x-y plane} \tag{3.7}$$

$$\phi = \arccos(\|\hat{v}_t\| \cdot \|\hat{v}_{t-1}\|) \tag{3.8}$$

For applications that require extremely precise rotations, we explored the possibility of using a single hand to specify rotations. In this case, the gesture is anchored with the thumb and index finger of one hand, in a similar posture to the bend gesture described previously. The line on the surface connecting the two touch points is used as the rotation axis. This enables the user to precisely specify the axis of rotation and provides additional anchoring support to control rotations. The disadvantage relative to the bi-manual interaction is that the bi-manual gesture can be more ergonomic for rotations toward or away from the body. Also, the hands can be placed arbitrarily far apart in the bi-manual case, with is useful for limiting occlusions when stereoscopic

displays are used.

One challenge in designing above-the-surface interfaces is distinguishing 2D from 3D gestures while still maintaining modeless operation. Our solution uses a series of checks to determine whether the hands are pivoting around their anchors and whether the motions of the two hands are moving in the same direction by roughly the same amount (correlated) or moving in opposite directions (anti-correlated). To identify pivoting motion, the total amount of movement over a sliding time window is calculated. In practice, we found that touches moving less than 0.06 inches within a window size of 0.06 seconds worked well for distinguishing the traditional 2D multi-touch inputs describe above from the anchored gestures described in this section. For a motion to be correlated, the hands must be moving in the same direction and by the same amount, causing the rotation axes and angles from each hand to be similar. A simple threshold is sufficient to test this. We found that 70 degrees for rotation axes and 20 degrees for rotation angles worked well for a majority of our users. Although users quickly get better at moving their hands in the same direction with some practice, at first they have a tendency to pivot their hands slightly towards each other (the reason for the higher rotation axis threshold), particularly when rotating away from their body. We also found it useful to limit the rotation axis from changing during an anchored rotation operation unless the difference between the new axis and the current one is more than 20 degrees.

3.5.3 Extensions for Anchored Z-Translation

Translating objects into and out of the plane of the touch table is a difficult problem for multi-touch. Since this interaction occurs in a 3D stereoscopic environment, z-translation is distinct from simple zooming. We introduce an anchored gesture metaphor for z-translation that is inspired by the successful balloon selection technique [86], in

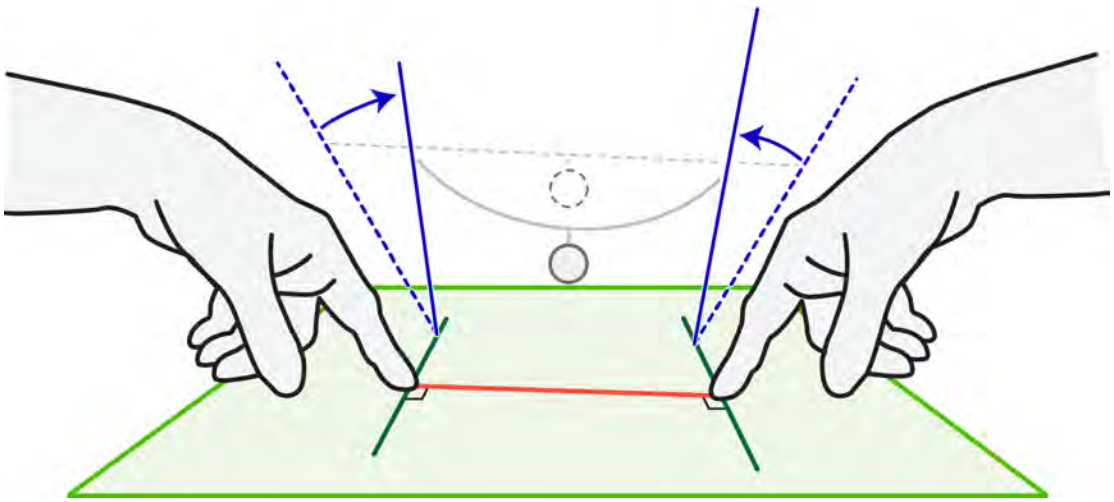

Figure 3.6: Illustration of the weighted string metaphor for z-translation. Pivoting the hands towards each other lowers the weight

which touches are imagined to control a string on a virtual balloon. In our case, with one finger from each hand touching the surface, picture a string running from one hand to the other (Figure 3.6.) A metaphorical weight in the center of the string is attached to the visualization space. Pivoting the hands towards each other creates slack in the string, causing the weight and the visualization space to drop lower (translating in the negative z direction). Pivoting the hands away from each other tightens the string bringing the weight and visualization space back up.

Although the motion in both this gesture and the rotation gesture described previously is to pivot the hands around a single touch point, they are distinguished by the correlation or anti-correlation of the direction of movement. For z-translation, we check that the hands are rotating in opposite directions by making sure that the rotation axes for each hand are pointing in opposite directions. In practice, we require that the difference between rotation axes for each hand be greater than 120 degrees. We also require that the direction of hand motion be along the line from one touch point to the other. When this is the case, the rotation axes for each hand will be close to perpendicular

with this line. This is tested by requiring the angle difference between rotation axes for each hand and the line between touch points be within 70 and 120 degrees.

In summary, the 3D navigation interface described here provides complete navigation capabilities for stereoscopic touch displays. The rotation interface includes two alternative techniques, one bi-manual, one using a single hand, but with two anchoring fingers. In comparing the bi-manual rotation interface with the z-translation interface, note that the key distinguishing aspect of the input is whether the motion of the two hands is correlated or anti-correlated.

3.6 User Study Evaluation

To provide a quantitative evaluation of the gestures developed we designed a comparative study to test the 3D navigation gestures and compare them to a state-of-the-art traditional multi-touch scientific visualization interface. The interface we picked as a point of comparison is the FI3D interface developed by Yu et al. [85]. FI3D allows the user to rotate, translate, and scale the visualization space as a unit to navigate the data by touching different elements of a frame widget that bounds the visualization then dragging onto the visualization space. Our implementation is inspired by this approach, but has a few differences. Unlike the original interface which was presented on a vertical screen, ours is displayed on the horizontal table described earlier. Based on preliminary feedback, we also added additional icons to the frame elements to help users remember their functions.

Task. Participants in the study were asked to perform a docking task. Four colored, solid pyramids were shown positioned in the view (Figure 3.6). The same scene was shown repositioned and reoriented, rendered as wire-frame outlines. Participants were asked to reposition the colored pyramids to match their corresponding wire-frames as

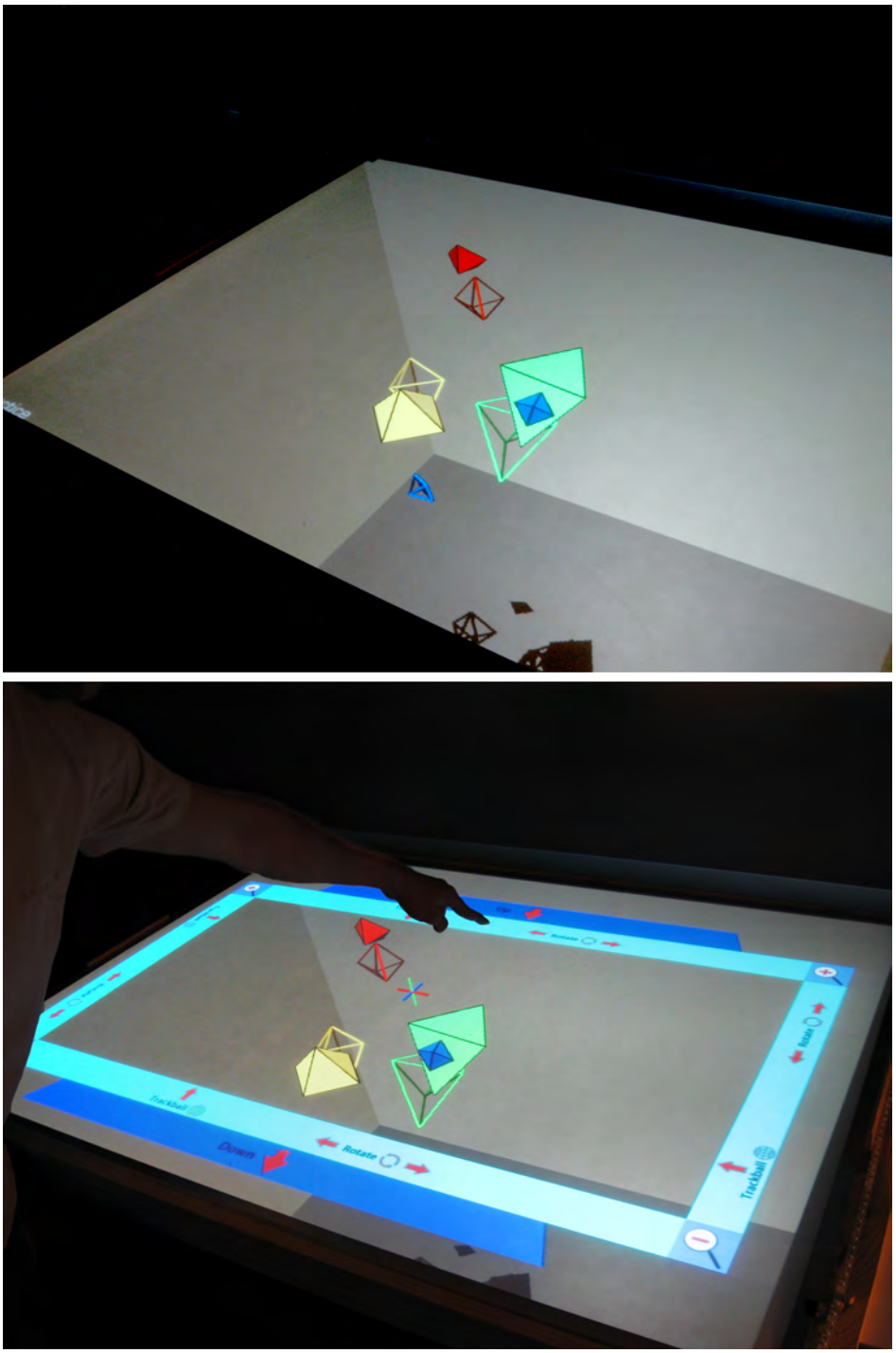

Figure 3.7: Docking task. Top: anchored interface. Bottom: FI3D inspired interface.

Table 3.2: Satisfaction for FI3D and Anchored Navigation Gestures. Mean score reported from 1 = Strongly Disagree to 7 = Strongly Agree.

	FI3D	Anchored Manipulation	Wilcoxon Signed Rank
The interaction technique allowed me to complete the task quickly	4.7	5.1	$Z = -0.787, p = 0.431$
The interaction technique was easy to use	4.3	4.6	$Z = -0.512, p = 0.609$
The interaction was intuitive	4.1	5.2	$Z = -2.209, p < 0.05$
I was able to be efficient	4.1	4.0	$Z = -0.171, p = 0.864$
I was able to be precise	5.0	4.5	$Z = -1.278, p = 0.201$

quickly and accurately as possible. The task was considered complete when the orientation of the solid pyramids was within 10 degrees of the target rotation and 0.25 inches of the target position for at least 0.8 seconds. When the pyramids were within this threshold they turned green. A time limit of two minutes was given for each trial before it was considered a failed trial.

Participants. Ten students (5 female) from our university completed the study. Nine reported prior use of 3D modeling or visualization software. Experience varied from none (1 participant) to more than 20 prior uses (3 participants). Eight participants also reported at least occasional video or computer game use.

Training. The two navigation modes were introduced individually and participants were encouraged to practice each before moving on to the next. This was followed by four practice trials with each interface. A five minute time limit was set for each trial in order to remind participants to complete the task as quickly and accurately as possible.

Design. We used a within-subjects design with one independent variable: *Interaction Technique* and two levels: Anchored Gestures and FI3D. Each participant performed 24 trials that were repeated for each technique, with the first four discarded as practice for the final analysis. Trials were presented in random order for each participant. The presentation order of technique was counterbalanced among participants. After completing the trials for each technique, participants were asked to rank the degree to which they agreed or disagreed with the statements listed in Table 3.2 using a seven point Likert scale.

Results. Two participants were unable to complete more than two-thirds of the trials within the time limit, for both techniques. As it was clear they had trouble understanding how to accomplish the task, their results were removed from further analysis. To understand whether anchored gestures would be faster to perform than 2D multi-touch input, we looked at trial completion time, which measures the amount of

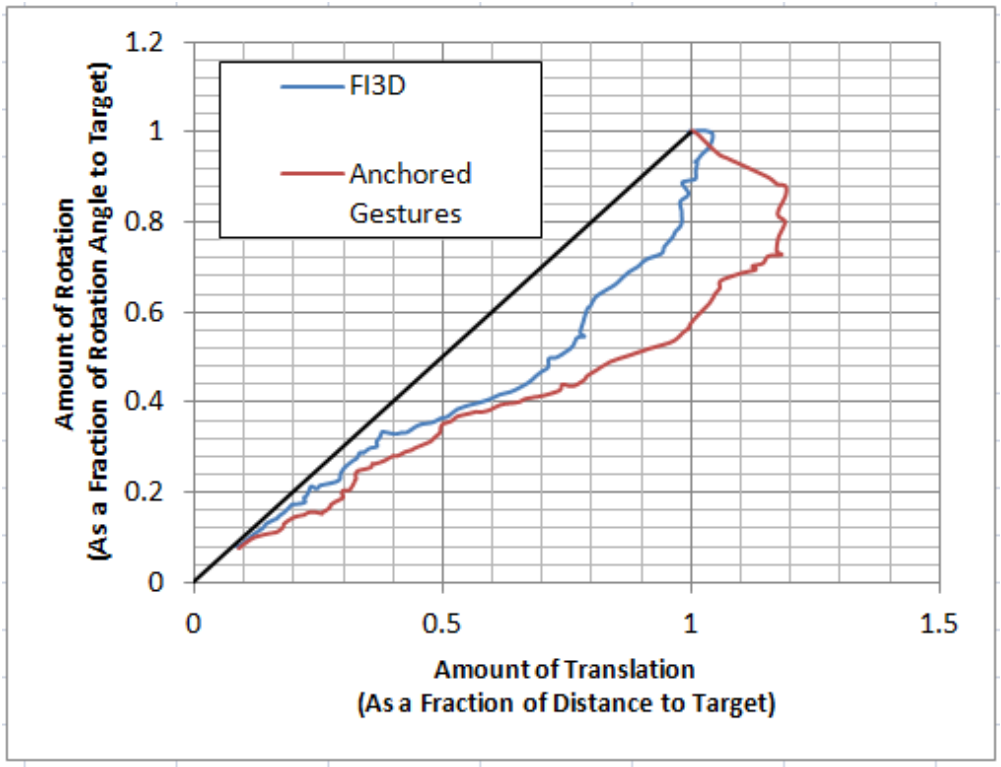

Figure 3.8: Average translation-rotation coordination ratios.

time it took to reposition the space from the time it first appears. The mean completion time for anchored gestural navigation was slightly faster, 59.45 seconds ($SD = 13.40$), compared to FI3D, 64.88 seconds ($SD = 8.51$). However, a repeated-measures ANOVA did not reveal a statistically significant effect for navigation technique on completion time ($F_{1,9} = 1.537, p < 0.246$).

In addition to speed, we were interested in understanding how the participants used anchored gestures and whether their strategies for adjusting the position and orientation differed across the two interaction techniques. To analyze these, we used the translation-rotation coordination ratio defined by Zhai et al. [87]. Using the approach defined in [88] we first calculated the translation-rotation coordination ratio by re-sampling each trial into 100 evenly spaced time intervals using linear interpolation. For each interval, this

ratio provides a two-dimensional coordinate. The average value for each interval is plotted in Figure 3.6. The curve starts at $(1, 1)$, where the objects are poorly oriented and positioned and proceeds to $(0, 0)$, where they are closely docked with the target. A reference line is shown to indicate the characteristic curve if both translation and rotation could be done with complete coordination.

Looking at Figure 3.6, it is clear that participants started the task by coarsely adjusting the rotation first then the translation. This is seen by the steeper decrease in the amount of rotation compared to the amount of translation on the right side of the curve. Note that both curves are under the reference line. One interesting effect that can be seen for the anchored gestures is that at the beginning of the trials the amount of translation gets steadily worse as the amount of rotation gets better. Essentially, as users start to correct the orientation, the position of the scene moves further from the target. We believe the difference in this trend compared to that of FI3D can be explained by our choice of making the center of rotation dependent on finger positioning on the table. We wanted users to be able to explicitly pick the center of rotation, which was a feature requested in the original FI3D study [85]. However, we noticed that participants seemed to anchor their fingers directly in front of them, rather than consciously positioning their touches where the center would be in a better position for the rotation they needed to accomplish. It may be that more training and/or a more self-revealing interface is needed to improve users' performance in this area, or it may be that this added flexibility in the interface proves to be too complex for users (at least novices) to understand. One participant commented, "I often found myself wanting to rotate around a certain axis or object but I had trouble controlling the center of rotation. I'd like to select one object and drag another object around it".

The results of the survey questions are summarized in Table 3.2. Using a Wilcoxon Signed Rank test, the only significant result was that participants found the anchored

interface more intuitive. One participant responded that anchored gestures "allowed me to make multiple movements at the same time, which sped up the entire process. [They] were also easier because I didn't have to reach across the table". When asked to rank which technique participants felt had the most control, more people preferred anchored gestures.

These quantitative results provide an important data point for comparing a fully functional interface based upon anchored multi-touch to related multi-touch systems; however, some of the most insightful feedback from the study came from our own observations of users and qualitative findings. We expand upon these in the following section, which includes feedback from discussion of "real world" uses of both the 3D navigation and modeling interfaces.

3.7 Qualitative Feedback

We analyzed the potential uses and impact of the anchored interfaces developed with a collaborating mechanical engineer and CAD expert. He was able to use the bend and twist gestures with very little training. For the bend gesture, his first thought was that it would be very useful during the design phase to virtually mimic sheet metal bending, used in creating tubes, boxes, and brackets. In particular, he thought it would be helpful to quickly plan out the order of multiple bending operations, as this is important when determining how an object will actually be fabricated. When fabricating more complex bended shapes, he told us designers occasionally want to constrain the length of the bend axis to only bend part of a side. In future work, this might be specified via an interface that constrains the length of the bend axis based on the placement of the hands and the distance between the anchoring thumb and finger.

One area of improvement he discussed is to display the bend surface with a thickness

rather than as a flat sheet. He thought this would be especially useful because the head-tracked stereoscopic display would make understanding clearances between objects or the tolerance ratio for curvature based on material type much easier to understand compared with traditional 2D displays used with CAD software. This approach was used in the modeling application presented in Chapter 6.

For the twist gesture, he found the relationship of the hands to the twisting surface clear and was easily able to understand how the surface reacted to his motions. He mentioned that he would like to be able to adjust the twist at more than just the top and bottom of the surface, so that the profile could be varied along the height. He also requested a way to adjust the gain of the rotation relative to his hand rotation. This would make it easier to specify large twists without lifting the hands above the surface and resetting.

Our own observations of the use of the anchored gestures are that users appear quite comfortable with the idea of interacting physically with their hands. During the training for the comparative study described earlier, we noticed that users were less hesitant to begin interacting with the anchored gestures than they were with the FI3D frame elements. Perhaps this because the anchored gestures are linked so closely to spatial movements of the body with which users are already familiar.

In observing use of the twist gesture, where the entire side of the hand is used as an anchor, we noticed that the physical height of the touch table becomes more important than it might be with more traditional multi-touch surfaces. Our touch surface is three feet high. Tall users needed to bend over slightly when standing in front of it to comfortably make the twist gesture. Thus, interface designers should be aware that certain anchored gestures may suggest changes to the ergonomics of surface displays relative to more traditional uses of multi-touch.

3.8 Conclusion

We have presented two applications exploring possible uses of anchored gestures. During informal use, anchored gestures were greeted with enthusiasm by users, who thought they have the potential to make multi-touch interaction even more intuitive. However, to be successful, designers must be cautious to make sure the gestures are ergonomic and self-revealing. We believe that, as multi-touch technology becomes more ubiquitous, the advantages of the passive haptic feedback and stability that anchored gestures provide will become even more useful, and we are excited by the potential to expand the impact of multi-touch, especially for 3D applications.

4

Lightweight Prop-Based Microscopy Visualization

[1]The previous chapter explored how multi-touch can be combined with spatial gestures to improve control and expressiveness for applications in scientific visualization and art. While the flat multi-touch surface can provide increased haptic support, it still limits the types of spatial interaction that can be performed.

In this chapter, we present a visualization interface that uses a lightweight prop as a passive haptic device to improve control. Building on our experience using a depth-camera to track the Anchored Twist Gesture in the previous chapter (See Section 3.3), a depth camera is also used here to track the user's hands and orientation of the prop. Although input technologies like depth-sensing cameras are not new inventions, these technologies have sparked a great deal of excitement recently due to the now widespread availability of low-cost quality sensors in this style, a development fueled in large part by recent applications to the games and entertainment industries.

[1]This chapter is based on work published in IEEE Transactions on Visualization and Computer Graphics [89]

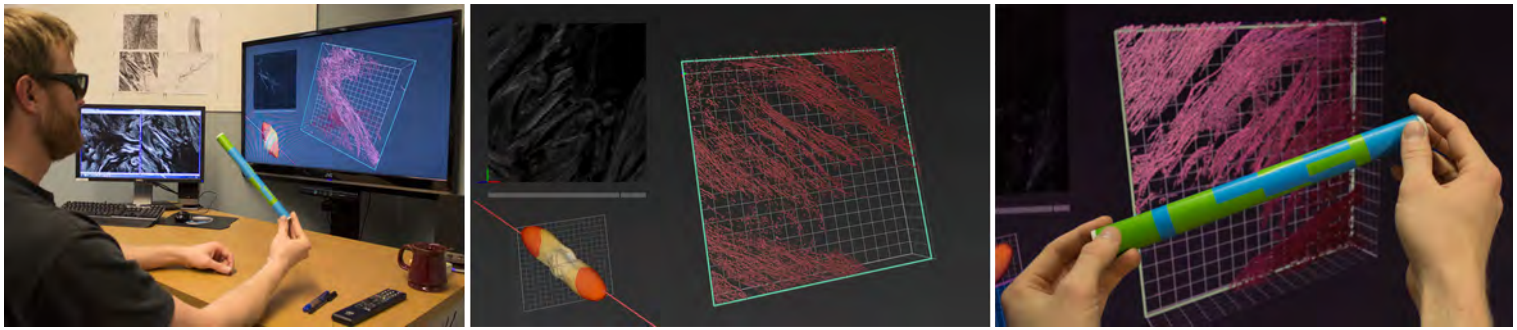

Figure 4.1: Left: Exploring fiber orientations in tissue using a paper prop and a commodity VR display. Middle: Linked views show: (1) a stereoscopic rendering of fibers; (2) a 3D fiber orientation histogram; and (3) 2D image slices. Note how only fibers oriented in the direction specified by the prop are rendered. Right: Patterns printed on the prop enable tracking of rolling and other gestures to provide a tangible 3D interface for the visualization.

Scientific visualization applications have great potential to benefit from these advances, including the emergence of low-cost 3D TV displays. For example, it is now quite reasonable to expect that a biologist, doctor, or other scientific researcher can work with a desktop-scale head-tracked stereoscopic display (fish-tank virtual reality) directly in his/her office. Just a few years ago, such a hardware system would have been prohibitively expensive and/or complex to maintain for personal data visualization. Given this context, our research addresses the important challenge of developing effective 3D user interfaces for desktop-scale stereoscopic data visualization. In particular, we aim to rethink the way that we interact with depth-sensing cameras, such as the Microsoft Kinect, to make this style of user interface more appropriate for accurate, real-time exploration of volume data.

The specific work described in this chapter is based on visualizing thin fiber structures from bioimaging data (Figure 4.1). The system was developed in collaboration with two biophotonics experts that use second-harmonic generation (SHG) microscopy to study the effects of tendonitis on collagen fiber organization in tissue [90]. We know from our collaborators that analyzing these data currently requires collecting image data and subsequently choosing the appropriate algorithm that optimizes sensitivity to detecting tendonitis and computational cost. This process is repeated, sometimes dozens of times, at the expense of depletion of resources (e.g., tens of hours of labor, hardware use), until the best site of interest of the specimen is chosen for study.

In contrast, in our vision of the future, scientists acquire a volumetric SHG image via real-time streaming to a low-cost 3D visualization system. They then explore the data from multiple vantage points, interactively select regions of interest, query underlying multidimensional data values, fit spatial data to models, and adjust visualization parameters. Based on the insight generated through this process, they re-adjust the SHG microscope (e.g., focus, scan angle, zoom) from within the interactive system to

look for specific queues in the collagen fiber organization that are highly correlated with the onset of tendonitis, queues that may not have been easily revealed or remain undiscovered using today's imaging.

Working toward this vision, this chapter presents the design and implementation of a low-cost stereoscopic data visualization system for analyzing collagen fibers captured via SHG microscopy. In addition to the discussion of the complete system and user feedback, our main technical contribution is a tangible user interface based on depth-camera technology. Following the motivation described above, we intend for this interface to be usable at a scientist's desktop. Therefore, the tangible props used in the interface are simply constructed from paper printouts. We argue, based on related research results in the 3D user interfaces community (e.g. [52]), that using passive haptic props dramatically increases the control and understanding that users have when working with 3D interfaces and data. Our work contributes a demonstration of how this can be achieved using today's hardware, while maintaining a "lightweight" interface in the sense that scientists are not required to buy a kit of plastic blocks [91] or generate a 3D rapid prototype object [48]. To realize the full-featured visualization system, we also contribute an algorithm for extracting thin, dense fiber features (centerlines and diameters) from volumetric data and a real-time stereoscopic rendering strategy for this type of fiber data.

4.1 Related Work

In addition to the spatial interfaces discussed in Chapter 2, our work builds upon visualization and feature detection in bioimaging data.

4.1.1 Visualization of Thin Fiber Structures

Scientific applications often require visualization of collections of 3D curves through a volume; both fluid flows (e.g., streamlines) and DT-MRI data (e.g., neural fiber tracts) require visualization of fiber-like data. Many graphics algorithms have been developed to depict these structures. We believe some of the most effective are the algorithms that strive to enhance the user's perception of depth and fiber crossings. A popular method for achieving this is rendering a "halo" around each fiber so that when two fibers cross each other their colors do not blend together. The frontmost fiber's halo (usually drawn in the same color as the background of the scene) occludes any fibers that are farther away from the viewer, making the frontmost fiber stand out as on top of the others. This has been implemented in volume renderings [92, 93] and, more recently, in GPU shaders [94]. Our rendering strategy builds upon the recent work of Everts et al. to also include perceptually motivated coloring, lighting, and stereoscopic rendering.

4.1.2 Feature Detection in Bioiomaging Data

Visualization of fiber orientation in biological tissue is important for disease assessment [95]. Our visualization system is designed to work with orientation data generated from bottom-up techniques which identify the orientation of individual fiber centerlines. A variety of techniques in this style exist; however, the scattered-snakelet approach [96] is appealing because it quantifies the orientation of short sections of each fiber. These short sections, called snakelets, can easily be compared with a user-specified vector to highlight fiber sections with a similar orientation. We introduce an adaptation of the scattered-snakelet approach that improves fiber segmentation and snakelet merging. This allows the technique to be applied to dense fiber tissues. Unlike other published methods for centerline extraction from dense tissues, the modified scattered-snakelet

approach is better able to handle fiber crossings or fibers with uneven surfaces because it does not rely on thinning [97] or recursively growing the centerline [98].

4.2 A Lightweight Tangible 3D User Interface

As shown in Figure 4.1, the system hardware consists of a 3D TV display, a depth-sensing camera placed in front of the display, and a simple paper prop held by the user. The following sections describe the key features and algorithms of the system. The workflow begins with a feature extraction step, which identifies features in the volumetric image stack and exports a set of fibers for visualization. The visualization system then imports these data to be displayed on the 3D TV.

From a scientific standpoint, the most critical aspect of these data to understand is the geometric alignment of fibers. For example, fibers in injured tendons are oriented in various directions, unlike normal tendons which exhibit highly aligned fibers [90]. So, understanding the alignment of fibers could lead to better diagnosis of tendonitis.

The prop design is motivated by our observations of how people naturally gesture when discussing these fiber data in front of a traditional 2D display. Even as we discussed the data within our own research group, we found ourselves naturally using the cylindrical shape of a pen as a handheld prop for indicating a 3D orientations. Although we discovered that current depth-camera technology was unable to robustly track the small diameter of a pen, the slightly larger diameter of a rolled piece of paper met the requirements that the prop be: (1) literally lightweight; (2) readily available in an office; and (3) contain a long axis for indicating a direction. Additionally, working with a rolled piece of paper makes it possible for a pattern to be easily printed on the surface to facilitate tracking and additional interactions.

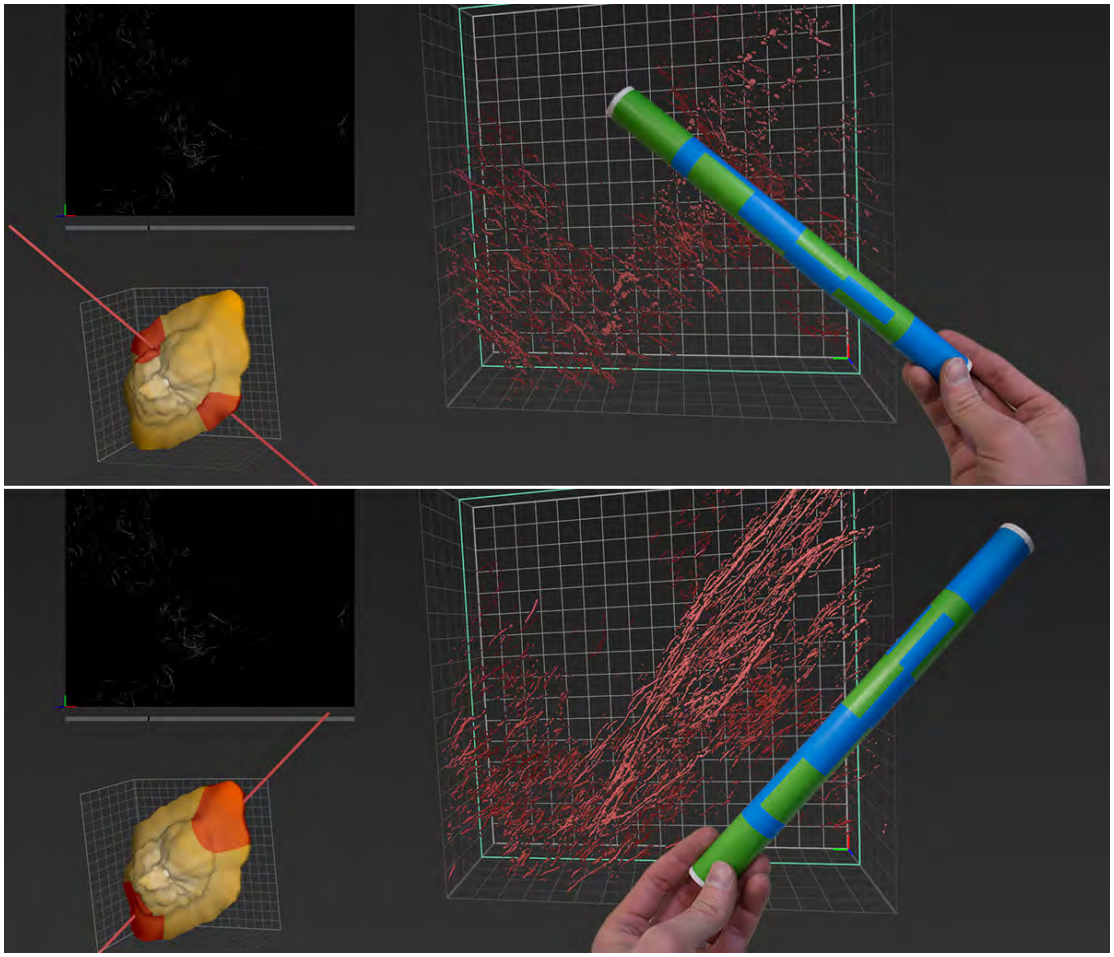

Figure 4.2: Holding the prop in one hand will show fibers oriented within a similarity threshold of the props orientation.

The prop interface supports three critical interaction tasks for exploratory visualization: (1) indicating a 3D vector to query the data, (2) setting scalar values, and (3) reorienting the 3D volume.

4.2.1 Gestures for Indicating a 3D Vector

Indicating a 3D vector is a common function when analyzing fiber data. A 3D prop-based approach provides an immediacy that is lacking in traditional 2D mouse-based

methods. Additionally, the prop serves as a physical display of the vector orientation, which is advantageous when collaborating and conveying spatial information to others.

To indicate a 3D vector, the prop is held with one hand in a pose that users naturally adopt when discussing the data. The orientation of the prop in 3D space is used to control the visibility of fibers, as shown in Figure 4.2. Fiber segments that align (within some similarity threshold) with the same orientation of the prop are displayed, while all other fiber segments are made completely transparent. The *orientation similarity threshold* used in this operation is simply a scalar value between 0 and 1. To determine whether fiber segments match the orientation of the prop, the orientation vector for the long-axis of the prop is compared to the orientation of each fiber segment; if the absolute value of the dot product of these two vectors is greater than the value for the orientation similarity threshold, then the corresponding fiber segment is displayed. The result is that the user is able to fluidly and quickly explore a fiber dataset to achieve an understanding of the predominate fiber orientations and their spatial distribution within the volume. The accompanying video demonstrates the fluidity of this interaction.

4.2.2 Gestures for Setting Scalar Values

The orientation similarity threshold described above is one example of a scalar value that needs to be set interactively. Since setting scalar values is a common operation in any visualization system, we judged it to be important to also support this action with the prop-based interface so that the user can set these values immediately while working with the interface, i.e., without needing to first put down the prop.

As illustrated in Figure 4.3, scalar values are set by using the prop like a 2D slider. The prop is held at its end by one hand, while the fingers of the other hand slide along the prop's length in a pinching gesture to indicate a value. Multiple scalar values can be set by holding the prop in different orientations as the user performs the gesture. For

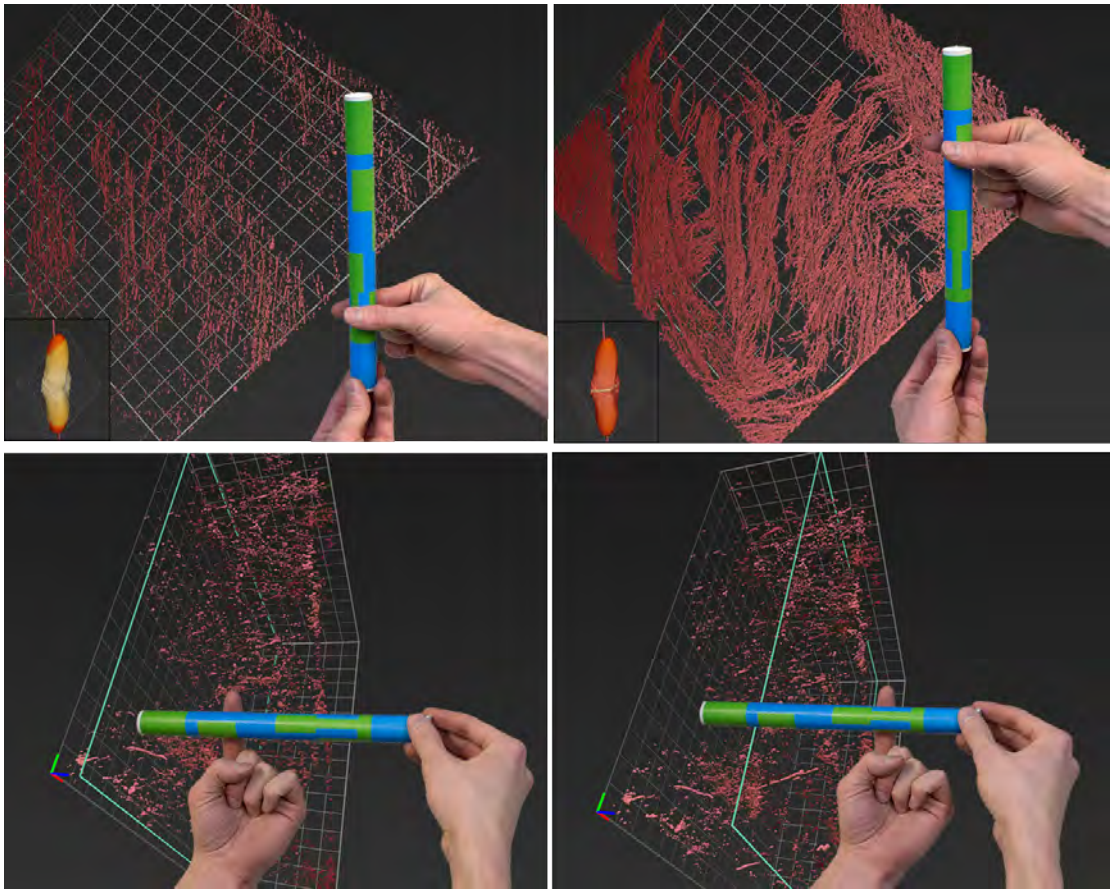

Figure 4.3: Top: Holding the prop with one hand and sliding the other up or down the prop will set the similarity threshold. Bottom: Sliding horizontally adjusts the position of the 2D image slice that is displayed.

instance, if the prop is held vertically the orientation similarity threshold is adjusted, while if it is held horizontally the index of the 2D image slice displayed in the upper left corner of the screen is adjusted.

The user's motion can be interpreted based either on the absolute position of where his/her finger crosses the prop or on the relative change in the sliding motion. For our applications, the orientation similarity threshold is set using the absolute position. The threshold is calculated using a non-linear scale to increase precision at the high end of

the scale where only the most similar fiber sections are rendered.

To adjust the index of the 2D image slice displayed in the upper left of the visualization, the user performs a similar gesture with the prop held horizontally. The same method of detection is used. However, it is useful to have more precision in this manipulation than with the threshold adjustment. So, a relative scale is used, and the user must clutch and slide his or her finger more than once along the length of the prop to navigate through an entire stack of about 150 images.

4.2.3 Gestures for Reorienting a 3D Volume

One of the most critical tasks to support in this visualization is reorienting the 3D volume to investigate the data from different viewpoints. Conceptually, this is similar to the task of indicating a 3D vector in that both tasks involve changing an orientation; however, we think of reorienting the volume as a more forceful action since it moves all of the data displayed on the screen. So, as shown in Figure 4.4 we use a more forceful grip on the prop (two hands rather than one). To the user, the impression is that his or her hands are grabbing onto the virtual scene and the scene responds to his or her rotational movement as expected. The rotation of the prop from one moment to the next is mapped to a corresponding rotation around the center of the scene. This follows a style typical of virtual reality navigation techniques (e.g. [99]).

The extension in our case is that rather than "grabbing the air", the user has a physical prop to hold. We found that this leads to a natural style of rotation for two of the three axes of rotation that is perhaps even more intuitive than "grabbing the air" due to the physical feedback provided by the prop. However, a control is still needed to rotate along the axis parallel to the long-axis of the prop. To accomplish this, we introduce another gesture – the user simply rolls the prop in his/her hands as shown in Figure 4.4 (bottom row). Section 4.3.6 describes how the specific pattern printed on

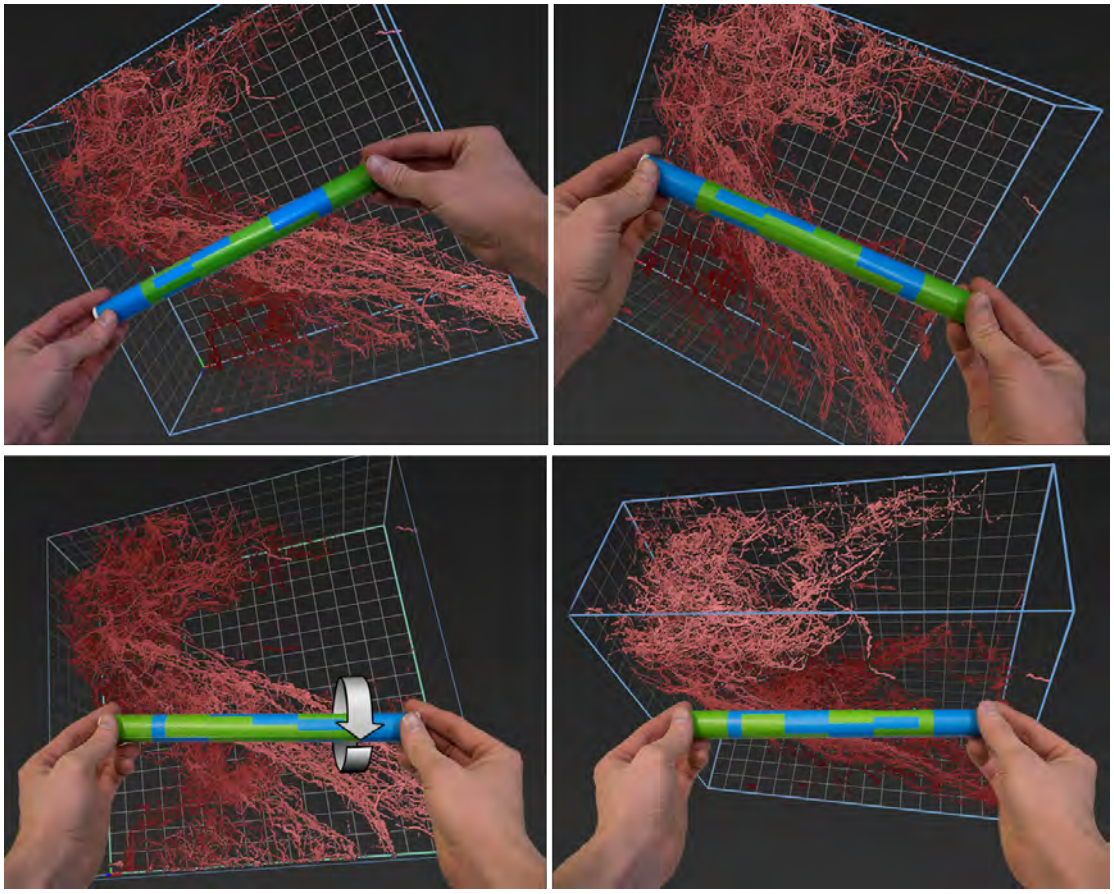

Figure 4.4: Holding the prop with both hands will 'grab' the volume, and rotating or rolling the prop will rotate or roll the volume around its center point.

the prop facilitates this interaction.

4.2.4 Seamless Transitions between Gestures

A final important characteristic of the interface is that it facilitates smooth transitioning between all of the gestures. Rather than explicitly switching the interaction mode with a button press or other discrete input, the design of this gesture set makes it possible to set the interaction mode implicitly based on how the user holds the prop. This type of spring-loaded mode selection has previously been found to reduce mode-based user

errors [100]. If the prop is held with one hand, then the gesture for indicating a 3D vector is recognized – this is the most common gesture. If the prop is held with two hands, then the position of the hand grips must be checked. If the hands are located at opposing ends of the prop, then the gesture for reorienting a 3D volume is recognized. If one hand is at an endpoint of the prop and the other is in the middle of the prop, then the gesture for setting a scalar value is recognized, and the orientation of the prop is used to determine exactly which scalar value is being set.

4.3 Implementation

As shown in Figure 4.5, the visualization system is implemented with three modules that execute in parallel. The data processing module uses the color and depth images produced by the depth camera to create a 3D point cloud of the volume in front of the display. This point cloud is passed to the point cloud processor module, which: spatially filters the data, segments out the hands and prop, determines the prop orientation, identifies whether the prop is held with one or two hands, and tracks the roll gesture. The point cloud processor module is able to process the point data in 10 ms on a quad core 3.4 GHz Intel i7-2600 CPU with 16 GB of RAM. The user interface module responds to updates from the cloud processor and performs the rendering.

4.3.1 Generating an Accurate 3D Point Cloud

The Microsoft Kinect produces color and depth images at 30 frames a second. These images are combined to project each color pixel to its corresponding 3D position, generating a 3D point cloud of the scene. Because the Kinect's color and IR camera are spaced apart, there is not a one-to-one mapping between the color and depth image pixels. At close distances, we found a significant misalignment between the color and

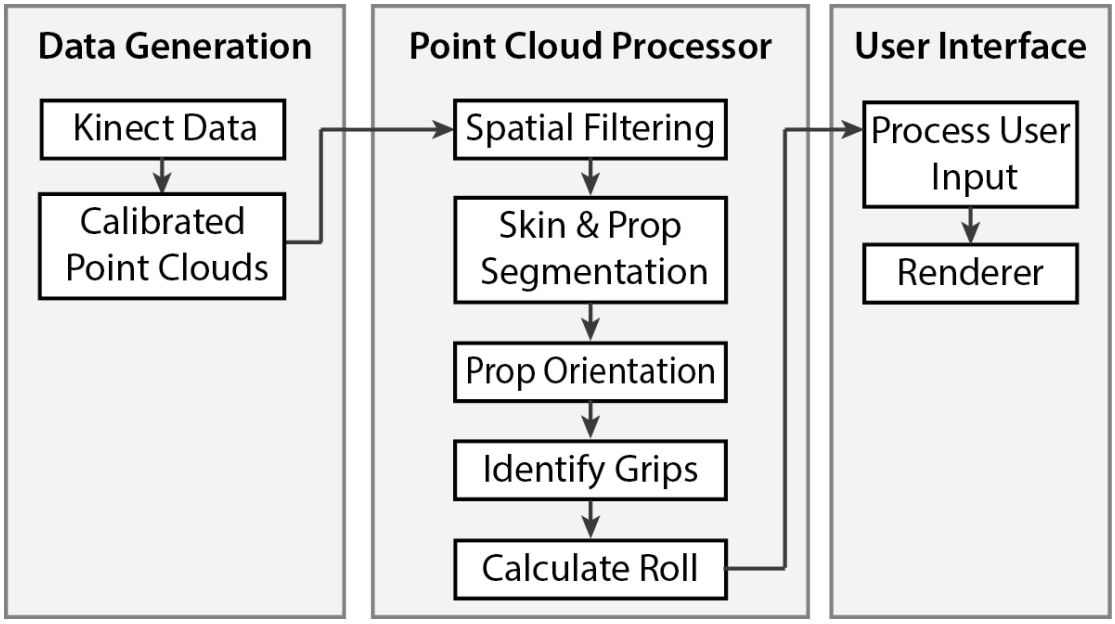

Figure 4.5: Overview of the visualization system's three modules. These modules work in parallel to track the user's interaction, process the resulting point cloud, and update the rendering based on the input.

depth in the factory calibration.

Our solution is to register the two images using a stereo calibration algorithm [101]. A checkerboard pattern is captured with both the IR and color cameras. Corresponding points in the images are used to determine the extrinsic and intrinsic parameters of each camera. Following a standard multi-camera calibration approach [102], the IR camera's intrinsic parameters (principal point and focal lengths) are used to reproject each 2D depth pixel into the IR camera's 3D coordinate space. Then, using the extrinsic parameters of the system (the rotation and translation between the cameras), these 3D points are transformed into the color camera's 3D coordinate system. Finally, the color camera's intrinsic parameters are used to project the 3D color points to 2D pixel locations in the color image. The result is a 3D point cloud where the correspondence between depth and color values is more accurate.

4.3.2 Filtering to the Volume of Interest

To improve performance of the point cloud processing, the 3D points are spatially filtered to remove points not in the vicinity of the prop. We assume that the closest point to the Kinect belongs to the user's hand or the prop. Using this assumption, points that lie further in depth than a band slightly wider than the length of the prop can be removed from the cloud. This removes the user's head, torso, and the background. Assuming the highest point in the remaining cloud lies on the hand or prop, we also filter vertically based on the length of the prop. Extraneous points on the wrist are filtered by removing points that lie under a quadratic surface.

4.3.3 Segmenting the Prop and Hands

The pattern printed on the paper prop is designed to enable a robust segmentation between the prop and the user's hands. The blue and green colors were chosen for their separability from skin colors which frequently exhibit red hues. We found that the results are more accurate than using a black and white pattern, where highlights or shadows on the hand caused frequent mis-classifications as prop points.

Since lighting, skin color, and the particular printer used to create the prop can all impact the color data reported by the camera, we have each new user perform a 30 second calibration routine where the prop is held in several pre-defined poses and the system records the color data observed for each of the three features the algorithm must detect: (1) Skin on the user's hands or fingers; (2) The blue portions of the prop; (3) The green portions of the prop. The color observations are stored in 2D histograms with 32×32 bins; different hues are arranged along one axis of each histogram and different saturation values are arranged along the other axis, similar to the approach in [103]. After normalizing the histograms, each one can be viewed as a probability distribution.

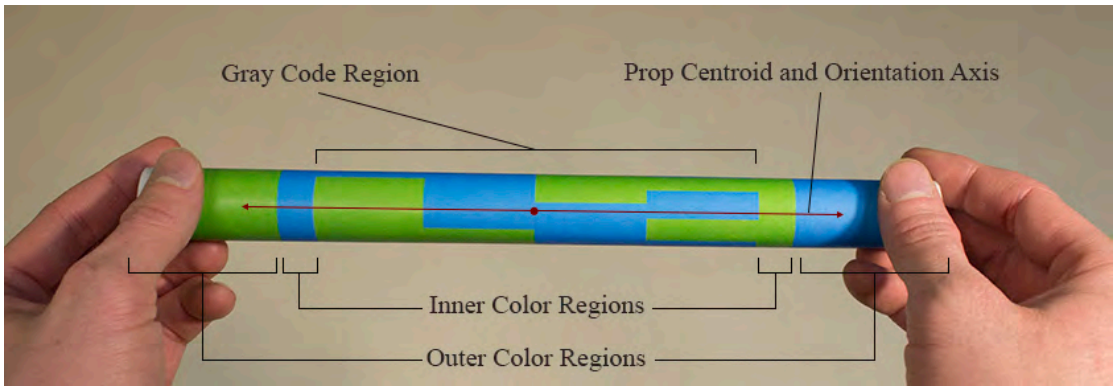

Figure 4.6: A specific pattern is printed on the paper prop to support multiple visualization tasks.

For a color of a particular hue and saturation, the probability that it belongs to the user's hands is determined by the value in the corresponding hue-saturation bin in the skin feature histogram. This nonparametric model is able to handle the non-linear boundaries between the skin and prop colors, is very quick to compute, and allows the system to be used under various static lighting conditions. Using this strategy, each point in the point cloud is classified as belonging the the user's hands, a blue portion of the prop, a green portion of the prop, or some other unknown object.

4.3.4 Detecting the Orientation of the Prop

Once the points have been classified, the prop orientation is calculated using the covariance matrix of the prop points normalized by their 3D centroid. Because the prop's length is much longer than its diameter, the maximum eigenvector of this matrix indicates the prop's 3D orientation. From frame to frame, we smooth the value of the orientation vector using the 1€ filter [104], a low-pass filter where the frequency cutoff is based on update speed to balance jitter and lag.

4.3.5 Identifying Grips

To distinguish between adjusting 3D orientation and other gestures, it is important to detect whether the prop is being gripped with one or two hands. First, the 3D locations of the prop endpoints are calculated. Then, the number of hand points within a 2 cm radius of the prop endpoints is computed. If this number exceeds a preset threshold (15 in our implementation), then a grip is identified at the corresponding end of the prop.

At least one of the user's hands will always occlude an endpoint of the prop, but since the prop's geometry and color pattern are known, the 3D location of the endpoints can still be determined. The prop has a solid color band, either green or blue, at each end, followed by a thinner band of the opposing color. (These inner and outer color bands are labeled in Figure 4.6.) The algorithm projects each point that has been classified as being part of the prop onto the prop's median axis to create a set of points that lie along a 3D line. Then, it searches for color patterns along this line of points, specifically the algorithm detects the edge between the inner and outer color bands. Once the 3D location of this edge is found, the endpoints of the prop are calculated by translating from these locations outward along the prop's orientation vector by the length of the outer color band.

4.3.6 Tracking Roll Gestures

A color pattern was designed for the central portion of the prop to detect the gesture used to reorient the volume by rolling the prop around its median axis. The pattern is a 4-bit Gray code, a cyclical reflected binary sequence where each successive number has a Hamming distance of one. This property makes it easy to detect errors from noisy sensor readings. Because the code is cyclical, i.e. only a single bit changes between the first and last number in the sequence, there is no discontinuity in tracking as the user

Table 4.1: The Gray codes used in the prop pattern with their corresponding binary and decimal numbers

Gray Code	Binary Number	Decimal Number
0000	0000	0
0001	0001	1
0011	0010	2
0010	0011	3
0110	0100	4
0111	0101	5
0101	0110	6
0100	0111	7
1100	1000	8
1101	1001	9
1111	1010	10
1110	1011	11
1010	1100	12
1011	1101	13
1001	1110	14
1000	1111	15

rolls past the end of the sequence.

To read the current Gray code, we first find the subset of the point cloud points that lie in the Gray code region of the prop. These are calculated in the same manner as described above for finding the edge between the inner and outer color bands, but this time, the inside edges of the inner color bands are detected and only points lying between these edges are used for analysis. These points are filtered to include only those that lie with a 2.5 mm radius of the median axis of the prop, so as to remove any edge points that might pick up color from the surrounding scene rather than the prop itself. The remaining points are divided into four bins according to their 3D positions along the length of the prop. The average color for each bin is calculated and the bin is assigned a 0 or 1 depending on whether the color is green or blue. The pattern is read from the green end to the blue end of the prop to create the Gray code. For example, the Gray code in Figure 4.6 would be read as 0111.

As the user rolls the prop, successive Gray codes are read and converted to their

corresponding binary number. Each increase or decrease in the binary number is mapped to a 5 degree rotation of the fiber volume around the axis indicated by the prop's orientation.

4.3.7 Detecting Finger Position Along the Prop

During the gestures used to set the orientation similarity threshold and for navigating through the image stack, the prop is used as a physical slider. The points belonging to the sliding finger will be classified as skin. So, to detect the crossing location, the skin points are checked to determine whether they lie within 8 mm of the median axis of the prop and within the length of the prop. The 3D centroid of these points is then calculated and projected onto the prop's axis, and the position of this point along the length of the axis is used to set the appropriate scalar value.

4.4 Real-Time Rendering

The stereoscopic visualization includes a volumetric display of the fiber data, a 3D fiber orientation histogram, and a number of other visual widgets to facilitate interaction.

4.4.1 Fiber Rendering

Each dataset consists of short individual fiber segments. A point-based rendering strategy, similar to that introduced by Everts et al. [94], is used to draw each segment; a GPU shader expands points into view-aligned quads, and then applies a depth-dependent halo.

We extend this algorithm to adjust the scale of points based on the fiber thickness and to enhance depth perception using a colormap from light to dark as points recede in depth. The shader is also used to determine the visibility of each point based on whether it lies within the current orientation similarity threshold. Rendering three

quads per segment (one per endpoint, and one at the midpoint), the algorithm renders at interactive framerates to a row-interlaced stereo image at a resolution of 1920x1080 on a NVIDIA GeForce GTX590.

Lambertian shading is applied to the quads so that a series of segments appears to be an extruded tube. The surface normals are computed in screen space as if the visible portion of the quad is an infinite cylinder pointing in the direction of the corresponding segment. This makes the cylindrical shape of the fiber data more readily apparent, while the colormap based on depth provides additional depth queues to the user. This double encoding of depth is particularly helpful in areas with few fiber crossings where the depth-dependent halo technique is less effective at conveying depth.

4.4.2 3D Fiber Orientation Histogram

A spherical histogram widget displayed in the lower-left of the visualization conveys the global distribution of fiber orientations within the volume (Figure 4.7). This widget encodes the orientation data via both shape and a white-to-orange colormap. The shape is elongated and more orange in the directions corresponding with the most prevalent fiber orientations. The spherical histogram and volume are linked to move in tandem when the volume is rotated. A grid drawn behind the histogram reinforces this connection. The prop's current orientation is visualized as a red line in the histogram widget. The similarity threshold is represented by shading all the surface points within that threshold in a red color.

4.4.3 Additional Visual Widgets

Changes in prop orientation and similarity threshold immediately update both the volume view and the histogram widget. When the user places his/her hands on either side of the prop to reorient the volume, the change in the prop's function is indicated by

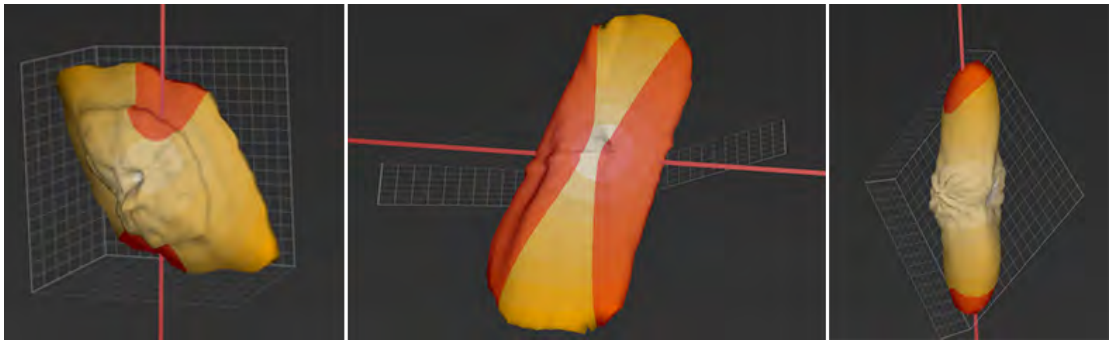

Figure 4.7: A spherical histogram depicts the distribution of fiber directions and the subset that is currently displayed based on the current orientation of the prop and the orientation similarity threshold.

drawing a blue outline around the bounding grid (Figure 4.4). A similar blue outline is drawn around the histogram when the user is adjusting the scalar threshold.

To complement the 3D fiber visualization, the 2D microscopy image slices are shown in the upper left corner of the display, illustrated in Figure 4.9. A single slice is displayed at a time. A 2D bar is displayed below the image to indicate the position of the current slice within the image stack, and the position of the slice is also rendered as a green outline in the fiber volume.

4.5 Identifying Fiber Structures

To generate the data used in the visualization, fiber centerlines are extracted from the volumetric image stack using an extended version of the Scattered Snakelets algorithm [96]. Snakelets are small active contours created by cutting traditional active contour snakes into shorter independent sections. These short sections are then computationally efficient to compare against the current prop orientation.

Conceptually, the algorithm works by iteratively moving snakelet endpoints towards the local maximum in the gradient of a voxel distance map. This approach is similar to

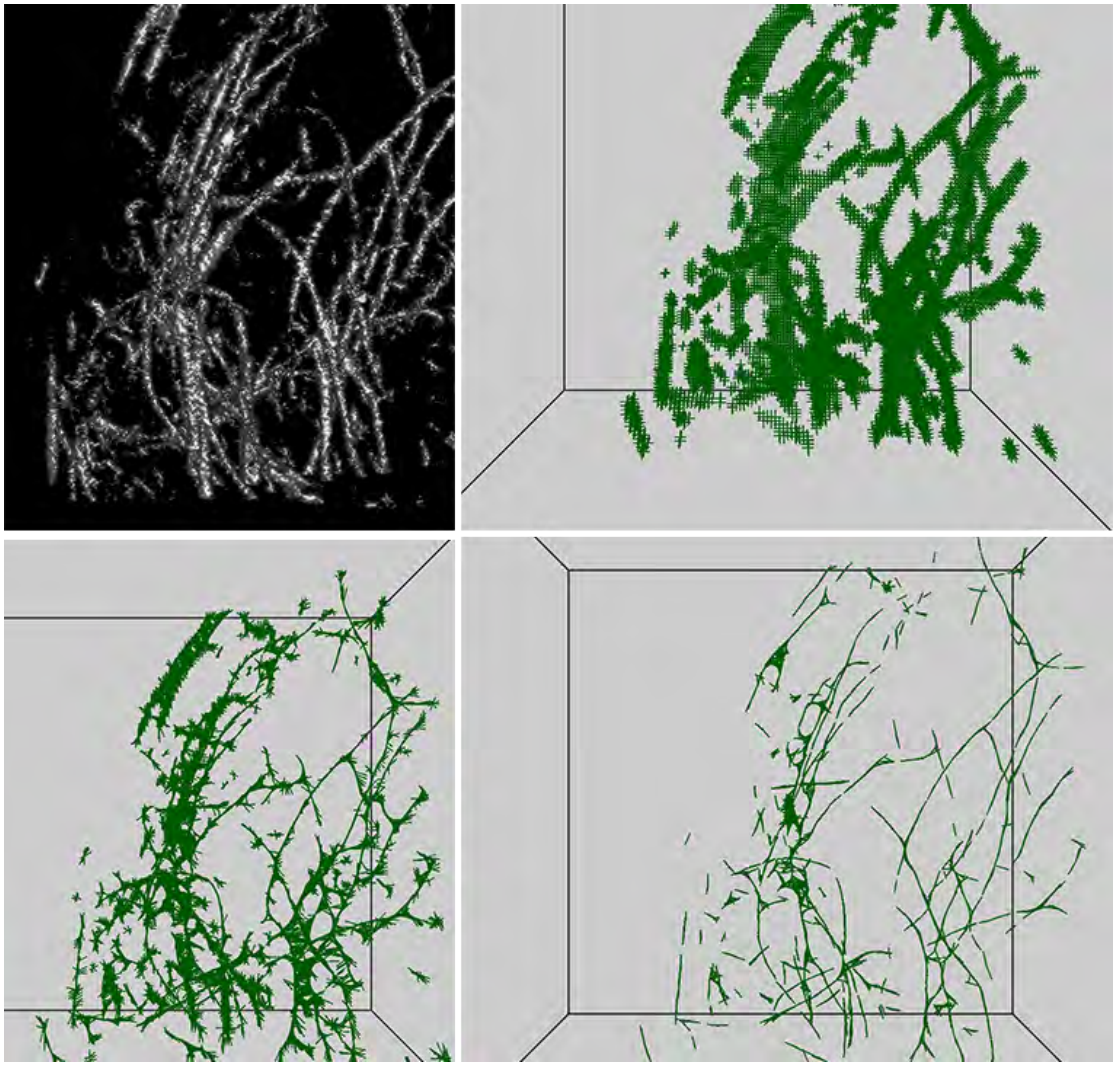

Figure 4.8: The Scattered-Snakelet approach for finding fiber centerlines. Top left: Volume rendering of collagen fiber. Top right: Snakelets are initialized on a grid. Bottom left: Snakelet endpoints are moved along the gradient of the distance map. Bottom right: Final centerlines.

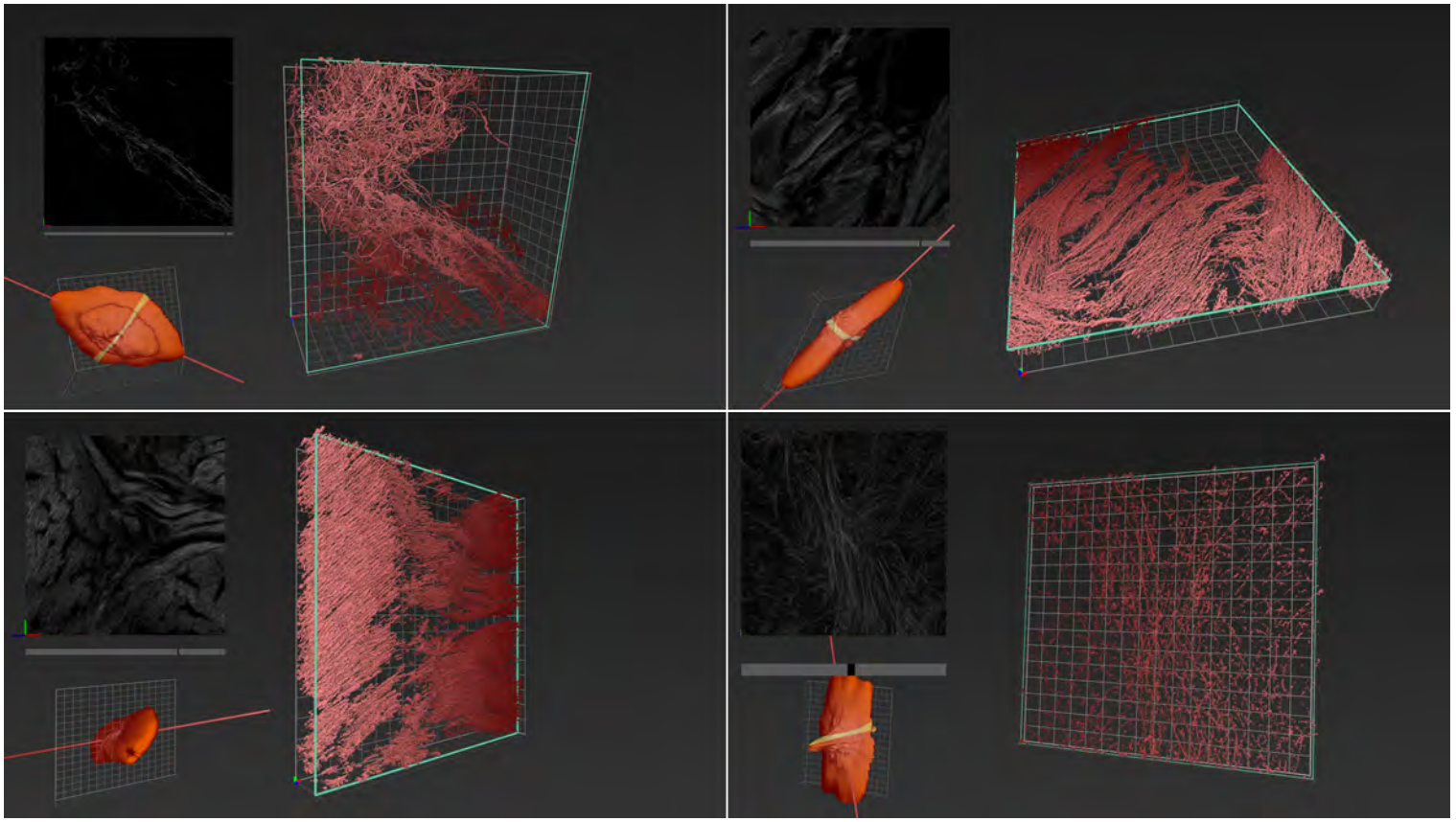

Figure 4.9: A variety of tissue datasets used with the system. Upper left: a collagen scaffold. Notice how the halos, lighting, and coloring help show the varying structure of the data. Upper right: pig sclera tissue. Lower left: a cross section of pig tendon. Lower right: rat cervix tissue.

the work of Prohaska et al. [105] which also uses a distance map gradient to find centerlines. Because the centerlines of fibers are furthest from the background, they contain the highest distance map values, and the snakelets gradually converge on the centerline. After convergence, adjacent and overlapping snakelets are merged by averaging their endpoints.

As opposed to the original algorithm that was designed to extract centerlines from large tubular structures like the small intestine, our extensions make it possible to apply the snakelet technique to dense biological tissues. Specifically, we use adaptive thresholding to create the Euclidean distance map, which is critical in our applications since variations in fiber density cause changes in the background intensity throughout the volume.To find adjacent snakelets for merging, a spherical radius is used, rather than an orthogonal plane as in the original algorithm. This helps reconnect gaps caused by segmentation errors. Finally, performance is improved by removing the active contour aspect of the original algorithm, keeping the snakelets as straight segments. Given the small length and large number of overlapping snakelets, users perceive the fibers as smooth curving paths.

The algorithm contains four steps:

1. Calculate a Euclidean distance map for the fiber voxels using adaptive thresholding:
    - Set the threshold to the maximum of the average value of the voxels within a six-voxel radius and a minimum value set empirically for each dataset.
2. Using a regular grid over the voxels (we use a grid size of 2 voxels), initialize three orthogonal snakelets at each grid point with a positive distance value (i.e., each grid point that contains some part of a fiber):
    - The initial length for these snakelets should be equal to the average fiber

Table 4.2: Runtime results for identifying fiber structures.

Dataset	Dimensions	Initialized Snakelets	Final Snakelets	Runtime (sec)
Collagen Scaffold	$2048 \times 2048 \times 165$	287,905	280,860	110.81
Pig Tendon	$1024 \times 1024 \times 138$	4,282,503	538,396	868.64
Pig Sclera	$1024 \times 1024 \times 161$	3,771,408	298,780	784.45
Pig Cornea	$1536 \times 4608 \times 16$	2,764,542	395,230	464.55
Rat Cervix	$512 \times 512 \times 27$	298,995	40,219	27.77

diameter within the dataset in order to avoid snakelets that lie within two different fibers (we use a length of 5 voxels).

3. Refine snakelet endpoints towards the fiber center:
   - For each snakelet, iteratively move each endpoint towards the local maximum in the gradient of the distance map.
   - Repeat until the endpoints move less than half of a voxel.
4. Remove snakelets that cross fibers and merge adjacent snakelets.

4.6 Results

Figure 4.9 shows screenshots from applying the visualization strategy to four different fiber datasets. Since these images capture just a point in time, the best indication of the results achieved with the system can be viewed in the accompanying video, which documents the way that users typically interact with the system.

Run times for the snakelet algorithm are reported in Table 4.2. The scattered-snakelet approach can be parallelized. Our implementation is parallelized using OpenMP on a machine with dual 2.5 GHz Intel Xeon E5-2640 with 6 cores each.

We have found that the performance of the fiber orientation algorithm depends strongly on producing a correct fiber segmentation and resulting distance map. For example, the image slices for the rat cervix dataset, shown in Figure 4.9 lower right,

contain a large amount of noise, leading to gaps along the 3D fiber renderings. Additionally, with dense fiber datasets, merging snakelets becomes much more important to reduce the number to an amount which can be easily rendered in real-time.

4.7 User Feedback and Evaluation

Our research team includes domain-science collaborators who study biophotonics and have helped us design and evaluate the system using their datasets. One is a graduate student in mechanical science and engineering. The other directs a biophotonics, optical physics, and engineering research lab at a top academic research institution. Both make regular use of computational analysis tools such as Matlab, but before starting this project neither had ever visualized their data in 3D. Both had used the Microsoft Kinect a few times previously to play games but never in a scientific context.

Our design process spanned more than one year and included more than 10 iterative design and critique sessions, with updated hardware and software tools delivered to our collaborators' site several times. They currently use the system one to two times a week to explore their data, and they estimate that they have spent over 80 hours to date using the system.

In addition to this ongoing process, we have also conducted two hour-long structured sessions to provide a more summative evaluation of the system. Since our team is separated by distance, these sessions were held over videoconferencing. Identical hardware and software systems were developed and installed at each site. Thus, one of the first bits of evaluative feedback is that the interface and algorithms presented here are robust to working in different labs, in different lighting conditions, and with different users controlling the interface.

Before the summative evaluation sessions, the domain scientists had only used prototypes of the system. At the beginning of each session, the participants were given a reminder tutorial of the visualization and interaction techniques. They were then able to use the system themselves to explore the data shown in Figure 4.9. While interacting with the data, they were encouraged to talk-out-loud and ask questions. Occasionally reminders about how to perform specific gestures were given. Afterwards, a semi-structured interview was conducted with notes recorded by the evaluators.

We structure the user feedback and evaluation along three axes: (1) fit within normal workspace activities, (2) utility of specific visualization features, and (3) impact within the application domain.

4.7.1 Axis 1: Fit within Normal Workspace Activities

High-level feedback on the system supports our goal of developing an interactive, desktop-based 3D visualization platform using consumer-level hardware. Our collaborators described new research in their field and how it might intersect with this style of data visualization. The ability to bring 3D visualization into the same physical space as the bioimaging hardware could create a paradigm shift in how bioimaging tools are used. One scientist commented, "From my perspective this is a big thing because you have brought it [the 3D visualization] to someone's desk ... it is incredible that you can buy this".

Currently, the scientists analyze their data by viewing static 2D slices or volume renderings. They use Matlab to calculate local fiber orientations, but for visualization, the information is decoupled into two separate 2D histograms of θ and ϕ spherical coordinates. In contrast to this approach, the collaborators readily confirmed that the ability to query the data and interactively display fibers at particular 3D orientations using the prop-based interface was immediately useful. They reported that they view

the system as being most helpful during initial stages of data analysis, which involves exploring their data to gain insights and identify specific hypotheses that could then be quantified after visualization using more traditional quantitative tools.

4.7.2 Axis 2: Utility of Specific Visualization Features

Users appreciated the ability to specify a 3D vector via a physical prop. One scientist commented "Taking the wand and pointing it in 3D space is as intuitive as it can get".

Some of the more complex interactions such as scalar adjustment were not as self-revealing. One scientist said, "This is so far from what we are used to today. This is a whole new world of interaction..." His comment is representative of the excitement we observed throughout the tool development, but we also understand that the interface and visualization require scientists to adopt a new way of thinking about their data analysis. This will take time to integrate into their current workflows. The same scientist continued, "How much ... do I have to think about what I'm doing to get a certain effect vs. can I draw upon my knowledge of how I use computers [today]".

The need to learn and remember gestures is one limitation of many gesture-based user interfaces, and it would be interesting to see if techniques that facilitate learning 2D gesture sets, such as the Gesture Bar [106], might also be applicable to prop-based 3D interfaces. Additional discussion on this theme revolved around how the paper prop is extremely simple, but does four different things. Users stated that they might actually prefer to move some of this functionality out of the prop-based interface and into more traditional interfaces, such as menu systems with which they are already familiar. We are interested in studying further whether this preference might change after more use of the system or whether a hybrid interface combining props with more traditional graphical interfaces might be more effective for some users. If so, the questions of how to facilitate seamless transitions between the different styles of interfaces and how to

most appropriately assign tasks to each interface would be interesting to study.

The 3D histogram for visualizing the fiber orientation distribution was evaluated as both useful and an interesting contrast to current methods. One user mentioned that he needed to learn how to interpret it relative to the 2D histograms used in current workflows. Again, there was an opinion that the interactive 3D visualization might be the best place to discover a 3D trend, but then more traditional Matlab-based 2D tools might be the right place to perform a quantitative analysis. Moving in the direction of including additional quantitative outputs in the visualization, one specific feature that was requested is to quantify the relative amount of fibers oriented in the direction indicated by the user rather than just showing the visual distribution.

The ability to reorient the 3D volume and the 3D histogram was evaluated as critical and effective. The domain scientists mentioned that this would let them explore where the interesting data features are and then start to develop the story that explains them. This led to a series of insights about useful extensions of the current interface. Extending the current approach of selecting subsets of the fibers based upon orientation, it would be useful to also be able to narrow the scope of the visualization to a smaller portion of the volume or zoom in to look at a feature in more detail.

4.7.3 Axis 3: Impact within the Application Domain

The domain scientists were enthusiastic about how this system could change their workflow, and as a result, the science they can do today. It is clear that trends can be observed in this interactive 3D visualization that are not easily discovered using current tools. The ability to see the fibers in a true 3D spatial context is regarded as a major change in this discipline. It is not yet clear exactly what impact this can have, but there is great excitement about the potential, and if visualization systems in this style can continue to advance toward the coupled real-time imaging and visualization that we describe as

motivation in the introduction to this chapter, this would be transformative in the field.

4.8 Extensions For Additional Applications

In terms of the broader applicability of the approach, we see the potential for the system to be generally applied to exploring vector-field datasets, such as fluid flow, weather data, transportation patterns, or marker trajectories from motion capture. In these applications, the prop orientation could be used to filter the display of vectors or segments of trajectories.

In addition, feedback from the domain scientists indicates strong potential for the system to be broadly applied to bioimaging datasets in addition to those shown in Figure 4.9. Nerve fibers were suggested as a specific useful next application area since there is even more variation in their spatial arrangements. Although many bioimaging datasets — including the ones referenced in this chapter — contain straight fibers, curving fibers are also common (e.g., neural fibers, blood vessels). One way to extend the interface to work with these datasets could be to utilize flexible props, perhaps in the style of recent flexible displays [56].

4.9 Limitations

One limitation of the current implementation is that the roll gesture requires users to roll the prop slowly enough that motion blur from the limited resolution of the low-cost depth camera does not cause mis-readings of the Gray code pattern. In general, we found that the tracking required to implement this interface today is right at the boundary of what can be successfully accomplished using today's commodity depth-sensing camera hardware. Although other low-cost interaction hardware, such as the

wand-based Playstation Move, might remove this limitation, tracking based on depth-sensing allows for a wider range of interaction techniques. For instance, it enables the software to detect whether the prop is held with one or two hands. It is not unreasonable to expect that computer webcams will be replaced with depth-cameras in the next 5–10 years, and as the camera resolution increases, we think that the paper prop could be replaced with a pen or pencil that are readily available on a scientist's desk. Looking towards this future, we are motivated to explore "lightweight" props rather than something like the "3D wand" used for the Playstation Move, which contains a fair amount of active electronics inside.

Our current implementation with just a single depth camera does lose tracking of the prop when it is held at an orientation that points directly into the display. In practice, this is not a problem for our particular application and datasets. Since the volumes are all smaller along one dimension, we orient this dimension into the screen, and if there is a need to specify an orientation that runs directly into the screen, the user simply reorients the volume. This limitation should also be able to be addressed with additional hardware (e.g., 2 cameras).

Finally, like most in-air gesture-based interfaces, there is some potential for user fatigue when using the system; however, the system is designed to track props at close range to the camera, enabling the user to rest his or her arms on the desk, which minimizes fatigue.

4.10 Conclusions

This work demonstrates the extent to which commodity visualization technologies, such as a depth sensing camera and low-cost 3D display, can now be combined to produce an interactive spatial interface with passive haptic feedback within a scientist's office

workspace. In the future, we are keen to extend this interface to support additional operations, such as controlling the SHG microscope. This could enable real-time streaming of data from the microscope to the visualization system, which would have a great positive impact on the imaging pipeline. More broadly, we are also excited to continue to advance the concept of "paper-based interactive visualization". We believe that there are many ways that regular materials (e.g., pens, paper) that we use naturally in our everyday work and discussions with collaborators might be employed to support more effective methods of interacting with, annotating, and exploring computer-based data visualizations. We believe this mixing of the real and physical world can often facilitate collaboration and make our visualizations more effective, more engaging, and more meaningful.

5

Force-Feedback Interaction for Controllable Filtering of Scientific Visualizations

[1]This chapter reports on applications of haptic feedback for controlled selection and filtering of volumetric data. In contrast with the previous two chapters, which explored passive haptic support, this interface investigates the use of active force-feedback using a SensAble Phantom device.

Controlled selection can play a critical role in the success of the visualization, enabling users to query, explore, and call up detailed data on demand. When working with volumetric datasets, traditional desktop, mouse, and keyboard interfaces can sometimes be used for these tasks; however, research has shown that traditional 2D interfaces can be difficult to use for real-world scientific tasks, such as selecting bundles of neural fiber tracks in dense 3D DT-MRI visualizations [108]. Our research explores the potential of

[1]This chapter is based on work published in the Proceedings of EuroVis 2012 [107]

novel haptic 3D user interface techniques for improving common interactive visualization tasks, such as volumetric filtering and selection, by making them more immediate, fluid, and controllable.

Our approach uses haptic feedback to improve the accuracy of both an initial 3D selection and progressive refinements to that selection. In addition, we seek to improve control through a connection to the underlying data context. Most of the existing 3D user interfaces that can be applied to visualization tasks take the approach of using direct input from the hand(s) to place boxes (e.g., [109]), a lasso (e.g., [110]), or some other widget in the 3D space of the visualization, then rely only upon the user's visual perception and motor control to precisely position these widgets. Our approach combines 3D user input with constraints defined interactively by multiple underlying data variables in order to enable selections of volumes that match the underlying spatial characteristics of the data.

We call the specific scientific visualization user interface technique introduced in this chapter *Force Brushes* (Figure 5.1). Brushing – a user interface technique that involves moving a user-controlled cursor over data values to highlight or select them – has been applied previously in both 2D (e.g., [111]) and 3D (e.g., [112]) visualizations and is an intuitive strategy for selecting a subset of data. Our approach extends this concept in several ways. We contribute a 3D brushing interface for data visualization that combines the following techniques: (1) using force feedback defined by features in the underlying dataset to guide the user in making an initial feature selection, (2) using force feedback to help control a 3D pulling gesture that smartly expands the selection to include regions with similar data characteristics, and (3) using a series of these brushing operations performed in sequence with different data variables to progressively refine the selection and explore the dataset.

Although the resulting interface technique could eventually be applied to many

different datasets, we have focused within the work presented in this chapter on the common case of 3D flows represented by multi-variate vector fields, such as the hurricane dataset shown in Figure 5.1. To provide an organizing visual and geometric structure for these data that is larger than a voxel, force brushes operate on integral curves (e.g., streamlines) through the data, with multiple data variables sampled along these curves. In other datasets, additional basic geometric data features may also be useful (e.g., vortex core lines, fibers in muscular and other tissue visualizations, neural fiber tracts in brain visualizations).

We begin the remainder of the chapter with a discussion of related work in volumetric selection interfaces. Then, we describe how force brushes can be used to first precisely select individual features, and then, using multiple brushes each tied to a different underlying data variable, intelligently extend the selection to include volumes of similar data. Finally, we close with a discussion of opportunities for future work.

5.1 Related Work

In this section we compare our work with related approaches for streamline selection in flow visualizations.

5.1.1 Streamline Selection

Several works, such as [12, 109, 38, 36, 17], examine the task of selecting streamlines in flow visualizations or dense DTI fiber tracts. This remains an active area of research. Predominately, these rely on spatially selecting volumes of interest with lassos, boxes or other widgets. Of particular note, is the haptic-assisted 3D lasso drawing system by Zhou et al. [36]. This system uses a force-feedback device to provide precise curve

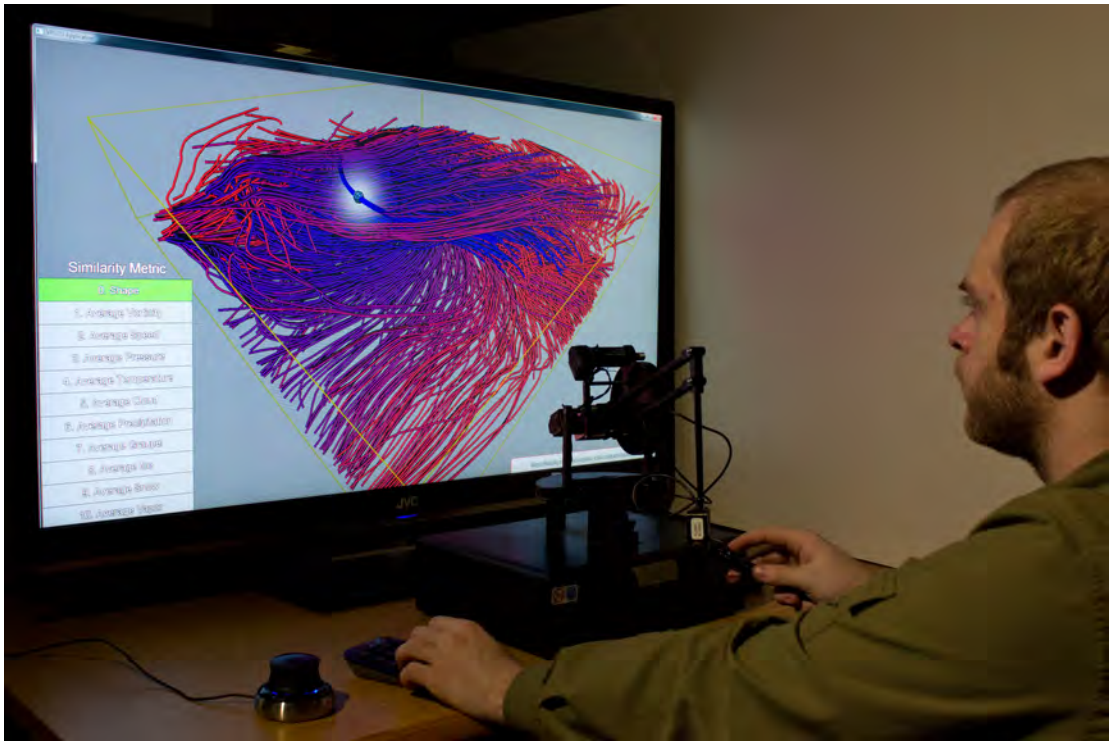

Figure 5.1: Force Brushes are used to explore a simulated multi-variate hurricane dataset. The user works in front of a 3D visualization with a SensAble Phantom Premium 1.5 device, holding the stylus in his dominant hand. Key presses made on a small keypad by the non-dominant hand are used to select different brushes, with each brush tied to a specific variable in the underlying dataset.

drawing for 3D lassos; however, in contrast to our approach it does not use the underlying data to help constrain the selection, relying only on the scientist drawing free-form curves.

Most similar to our approach of creating a selection based on a specific streamline in the data is the shape marking operation in the CINCH system [38]. This operation allows the user to draw a representative 2D line on a tablet and returns a selection of similar lines. Although our technique of selecting a representative line from the data is more constrained than drawing a free-hand line, we avoid any ambiguity associated with projecting a drawn 2D line into a 3D volume. In addition, we enable progressive

refinement of the simulation based on multiple variables and show how force-feedback can be used in this context.

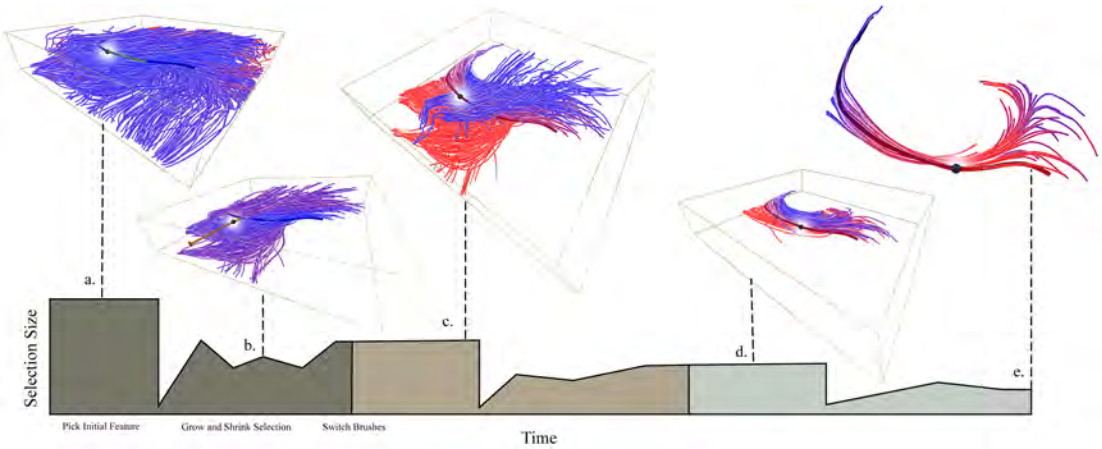

Figure 5.2: A sample workflow using Force Brushes. From left to right the selection is refined using a series of three brushes. For each brush, the user first selects an initial feature of interest, either an entire line or a subset of a line as in (a); then pulls out away from the line to grow or shrink the selection as in (b); then releases the stylus button to push the current selection to the stack as in (c). Additional brushes can be applied to further refine the selection based on other data variables as in (d) and (e).

5.2 Visualizing Hurricane Isabel: A Use Case For Multi-Variate Flow Feature Selection

An ideal use case for multi-variate flow selection is visualizing and analyzing weather data. These data frequently contain multiple variables such as wind speed, pressure, amount of moisture, and temperature. Our application, shown in Figure 5.1, allows scientists to selectively filter the visualization to identify critical regions of interest.

To do this, the typical workflow involves first picking a brush, such as *similar shape* or *average wind speed*. This updates the visualization to display just these data. Then, the haptic stylus is moved into the flow volume to explore the data and find a feature of

interest. A magic lens clips the geometry between the camera and the stylus enabling the user to see deep into the flow volume. The user picks a feature of interest by pressing and holding the stylus button. To indicate a subregion of a line, the user drags along the line, as in Figure 5.2a. Next, to perform a selection based on the identified feature, the user pulls the brush away from the line, as in Figure 5.2b. This has the effect of selecting only the single initial feature of interest identified by the user (e.g., one line), but as the user pulls the stylus further away from the initial line, the selection grows to include data from similar lines, where similarity is defined by the underlying data and the brush type (e.g., pressure, velocity, temperature). This process is fluid; in order to pick just the right subset of data, the user will typically repeatedly pull the brush far away and then move it back closer to the line while watching the selection update dynamically, as shown in the accompanying video. To complete the operation, the user releases the stylus button, which pushes the active selection onto a stack (Figure 5.2c). Then, using the same interface, the user can continue to refine the selection with additional brushes. For example, in Figure 5.2, points (c), (d), and (e) each show the end result of applying a brush to the data. When applied in succession, these brushes can be used to precisely refine a selection based on several data variables. With this typical high-level workflow in mind, the following section describes in detail the force feedback, data-driven constraints, and other technical concepts needed to make Force Brushes work.

5.3 Force Brushes

We describe in detail the two sequential tasks performed for each brush: selecting a region of interest along a streamline to filter the visualization, and growing the selection. To make switching between brushes quick, each brush is mapped to a separate number

on a wireless number pad (Figure 5.1). While one hand controls the haptic pen, the other is free to swap instantly between different brushes with a simple tap.

5.3.1 Selecting an Initial Feature

After snapping to a streamline to select it, each type of force brush can be used to select a region of interest. By pressing the button on the haptic pen, the scientist brushes along the line, releasing the button to end the selection length (shown in Figure 5.3). Haptic constraints keep the pen anchored to the selected streamline. In our implementation this is accomplished using an OpenHaptics surface constraint with a snap distance of 20.0. After the selection is completed, a metric for the selected feature is calculated based on the brush type. For instance, if the average pressure brush was used, the average pressure along the length of the selection is calculated, while the peak temperature brush calculates the maximum temperature in the selected range. The group of lines that are shown is then filtered to only contain streamlines that contain the same calculated value as the selected region.

Selection based on shape works slightly differently than the other brushes, which operate on one dimensional values. There are several previously published methods to calculate similarity between lines (e.g. [113, 114, 115]). We use a similarity metric that was originally designed to compare DTI fiber tracts [116], an application area that is also applicable to our technique. This measure is based on a weighted average of distances from each sample point along the selected streamline to the closest point on another, in order to determine how closely two streamlines follow a similar path. The distances near the end points of the line are weighted more heavily. To speed up computations involving comparison between entire streamlines, we pre-compute a similarity matrix that includes the similarity between each streamline and every other.

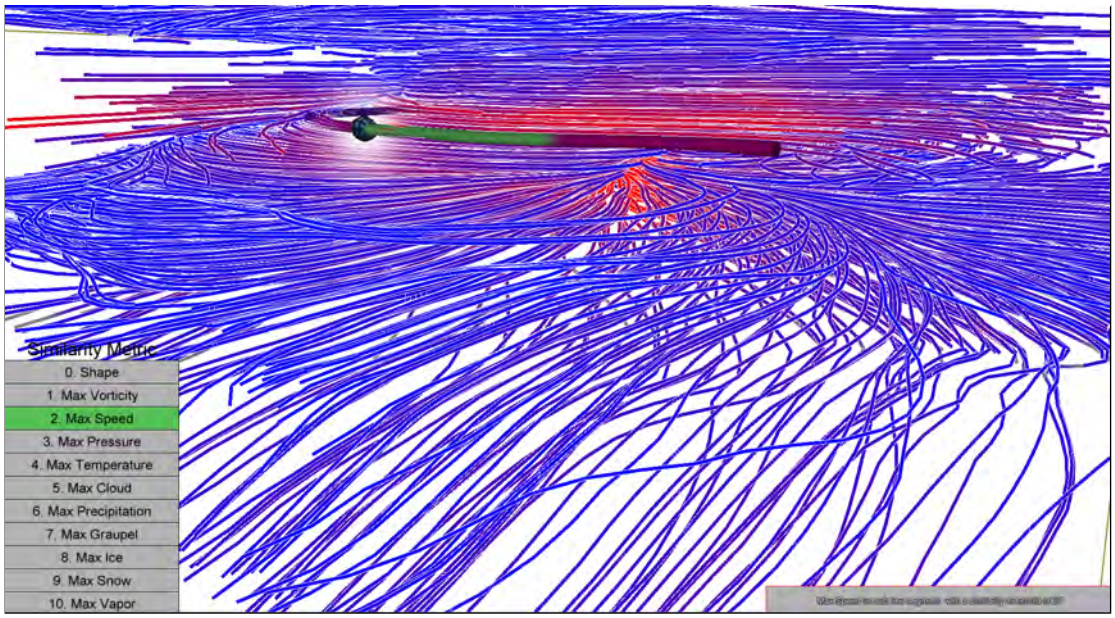

Figure 5.3: Dragging the haptic brush along a streamline selects a subset of the line, shown in green.

5.3.2 Growing the Selection

The selection growing gesture is distinguished from initial feature selection by pulling perpendicular to the selected streamline rather than along it (Shown in Figure 5.4). Once the point of the haptic pen leaves the line, it is constrained to a plane perpendicular to the streamline. The distance between the pen tip and the point where the streamline intersects the plane is normalized to $(0, 1]$ using a maximum distance of 40 percent of the haptic workspace. With our Phantom Premium 1.5 haptic device this presents a total movement of about 16 cm. The normalized distance is used to add streamlines back to the selection. For example, if the currently enabled brush is average pressure, the average pressure for each streamline is calculated. The range of streamline average pressures is mapped to the normalized distance, so that at a value of one all the lines would be added to the selection, while a value closer to zero would only add lines that

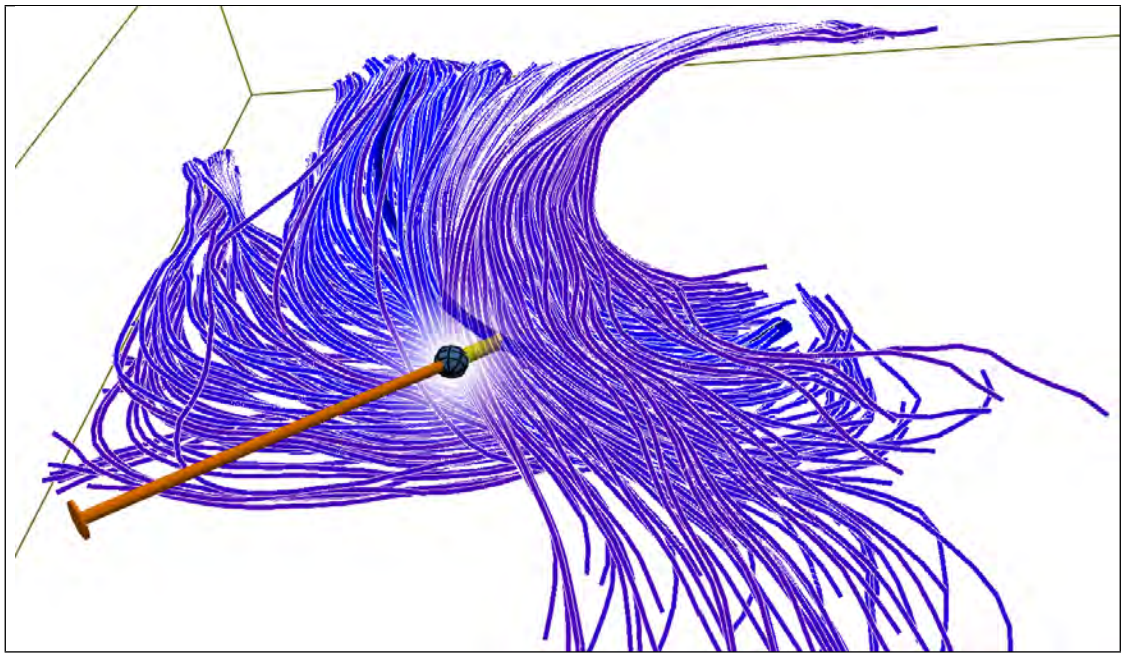

Figure 5.4: Pulling perpendicularly away from the initially selected feature grows the selection volume.

have average pressures very similar to the selected region.

5.3.3 Providing Visual Feedback

To help guide the user, we color map the streamlines based on their similarity. There are two choices for this mapping: absolute coloring relative to all of the data, or relative coloring based on similar features to the currently selected line. For some variables, such as shape, only relative color mapping makes sense. Our implementation uses relative coloring for shape and absolute coloring for the other variables.

To help with streamline selection we render a magic lens around the point of the haptic pen. This lens uses a gaussian opacity filter to allow the user to continue to see the selected line if it would otherwise be occluded by lines in front of it. See Figure 5.3.

During the growing operation, we render a ruler (shown in orange in Figure 5.4) extending perpendicularly to the selected line to provide spatial clues for how far the haptic pen tip (shown as a sphere) has been pulled away from the selected line.

5.4 Conclusions

Force Brushes are motivated by a need for more precise control over selection and other interactive data querying operations required by 3D visualizations. Although we have yet to formally evaluate the precision and control provided by Force Brushes as compared to other related approaches, our experience with the interface suggests that the combination of data-driven haptic constraints, haptic-assisted growing of selections, and progressive applications of multiple data brushes enables fast and accurate selections in dense 3D visualizations.

Compared to the passive haptic support in the interfaces discussed in the previous chapters, the active force-feedback that Force Brushes provide enables a higher level of control. In particular, the use of data context based on similarity with multiple variables greatly expands the complexity of 3D selections that are possible for scientific visualizations.

While the Force Brush technique is successful for scientific applications, it is still limited in applicability to less data-rich environments. In the next chapter, we explore how to apply its successful characteristics (locking onto curves and using these to provide context) to artistic 3D modeling while also enabling more freedom of expression.

6

Artistic 3D Modeling with Context

Spatial interaction techniques, such as those discussed in the previous three chapters, provide scientists and artists with improved control, expressiveness, and immediacy when interacting with complex 3D data. The drawback with this type of interaction is that the additional degrees of freedom enable additional ambiguity in how the input should be interpreted.

For example, Bollensdorff et al. [117] performed an experiment to identify how users prefer to interact with a virtual 3D environment on a multi-touch device. They showed participants an animation of a moving 3D object and asked how they would reproduce the motion using multi-touch gestures. One of the primary findings was that participants used the same interaction (i.e. same number of fingers and same gestures) to achieve clearly different transformations of the 3D object.

This is a common issue with gestural interaction. One approach to solve it is to interpret the user input based on the context of the interaction. In practice, it is extremely difficult to develop artificial intelligence algorithms to implement contextual

analysis in general, but domain-based knowledge can be used to constrain the problem for specific cases. For instance, one of the most successful aspects of the Force Brush interface described in Chapter 5 is how the underlying data context helped guide the interaction. Control was improved for initial data selection by haptic constraints that moved the users hand and input device to exact data values positioned in space. Additionally, the complexity of the types of queries users could perform was increased by the combination of a haptic gesture that expanded the selection based on the underlying context of data similarities.

In this chapter, we present a context-driven 3D sketch-based modeling application, called Modeling with Context. This application explores how photographs and 2D sketches that serve as inspiration or are deliberately created as a first step in the modeling process can be used as context to help interpret artist's spatial input when 3D modeling. This artistic application area was chosen specifically because this creative domain requires very different considerations for how the input should be interpreted compared to the scientific visualization applications presented in the previous chapters. In addition to improving the control of spatial interaction, we hypothesize that interpreting 3D user input based on the context will also increase the expressiveness and complexity of the creative results.

Consider Figure 6.1, which shows an innovative sheet metal sculpture by Pablo Gargallo. This sculpture was formed by cutting complex 2D shapes out of sheet metal (Figure 6.1 right) and bending and shaping them into 3D forms (Figure 6.1 left). The sculpture's smooth curves demonstrate a high level of expressiveness and control. Our work draws inspiration from these types of sculptures because their complex organic curves would be difficult to model using conventional software. By using a spatial interface and integrating data-driven context, we hope to provide the same amount of expressiveness and control that Gargallo had when working with real physical media.

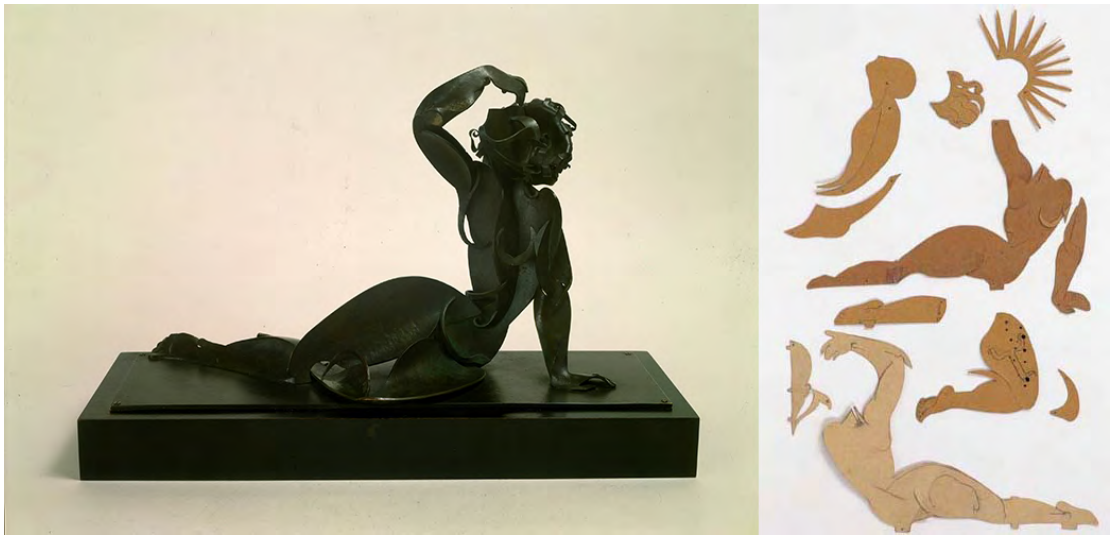

Figure 6.1: Pequeña Bacante Reclinada, 1929. Left: An innovative sheet-metal sculpture by Pablo Gargallo [118]. The artist bent and twisted flat pieces of metal into complex three-dimensional forms that would be difficult to create using traditional digital modeling tools. Right: paper templates the artist used in designing the sculpture [119]

In the next section, we contrast our Modeling with Context techniques with other examples that integrate inspirational materials into the modeling environment and interpret user input based on context. Then, we describe our application and implementation details. We present artistic modeling results, and finally some lessons learned, discussion, and conclusions.

6.1 Related Work

This work builds on several related applications that integrate concept sketches and photographs directly into the modeling environment and interpret user input based on the interaction context. In this section, we describe specific examples and how they differ from the work presented in this chapter.

6.1.1 Integrating Concept Sketches and Photographs

Integrating concept sketches and photographs into artistic computational tools to serve as inspiration has previously been explored (e.g. [120, 121, 122]). For instance, the Modeling-in-Context application by Lau et al [120] enabled an artist to sketch a 3D model directly on a photograph to gain a sense of perspective and relative dimensions. Like most of this related work, such as ShadowDraw [121], the use of inspirational materials in Modeling-in-Context predominately focuses on giving the artist a greater visual sense of proportion and layout.

Our sketch-based modeling tool also supports artists in determining proportion and layout by enabling them to place 2D imagery directly in the modeling environment; however, while Modeling-in-Context and ShadowDraw use drawn 2D input, our approach uses full 3D input. Because 3D input is less constrained than 2D, our application also improves the control an artist has while creating curves through space. The inspirational imagery serves as context to interpret and subtly correct jitter in the artist-drawn curves. This technique is described in Section 6.2.2.

6.2 Modeling with Context

As shown in Figure 6.2, the system hardware consists of a four wall, head-tracked stereoscopic CAVE environment. The artist interacts with the application using two tracked styluses, a primary drawing one held in the dominate hand and a secondary one held in the non-dominate hand. Because the physical interface and passive haptic feedback it provides are so important for improving control, as demonstrated with the microscopy visualization in Chapter 4, the styluses were custom 3D printed to mimic the shape of a large pen. An artist can grip them in a comfortable and familiar way, while using the fine motor control of his/her finger tips. Each stylus contains two push

Figure 6.2: The author uses the Modeling with Context application to create virtual sculptures in a virtual reality CAVE environment.

buttons. Additional implementation details about the CAVE hardware, input devices, and development of the underlying VR toolkit can be found in Appendix A.

A primary issue with many artistic virtual reality environments is that as soon as the artist steps into a CAVE or puts on a head-mounted display, he or she looses the context of his or her previous sketches and photographs that serve as artistic inspiration. For Modeling with Context, the modeling workflow, shown in Figure 6.3, begins with the artist placing 2D concept sketches and photographs as slides directly within the VR environment. Contours in these sketches are used as context for automatically refining the shape of 3D curves, called rails, that the artist creates. These rails serve as a wireframe describing the contours, shape, and topology of a 3D model. Finally, surfaces are created by selecting a rail and sweeping it along additional connecting rails. Here, the connecting rails are used as context for defining the topology of the surface.

We center our approach to spatial 3D modeling on "rails" because it would be difficult to draw a controlled surface shape directly; however, previous work (e.g. [18]) has shown that users are able to draw controlled curves through space. Other 3D modeling systems (e.g. [39]) have used feature curves to define 3D shape as well. Predominately these systems extract curves from existing geometry for editing operations rather than defining new curves for surface creation [123, 124]. The few systems that do use curves for creating new geometry use them to define an implicit surface; however the use of an optimization to define the resulting surface rather than explicit user input can lead to blobbly shaped models as in the FiberMesh system [125].

In contrast, our approach uses rails as a wire-frame scaffold, similar in style to the way that artists draw construction lines in 2D and 3D [126]. The rails serve as guides for sweep operations to define surfaces.

An additional motivation for using rails in this way comes from theories of human perception, specifically how the human visual system recovers 3D form from 2D retinal

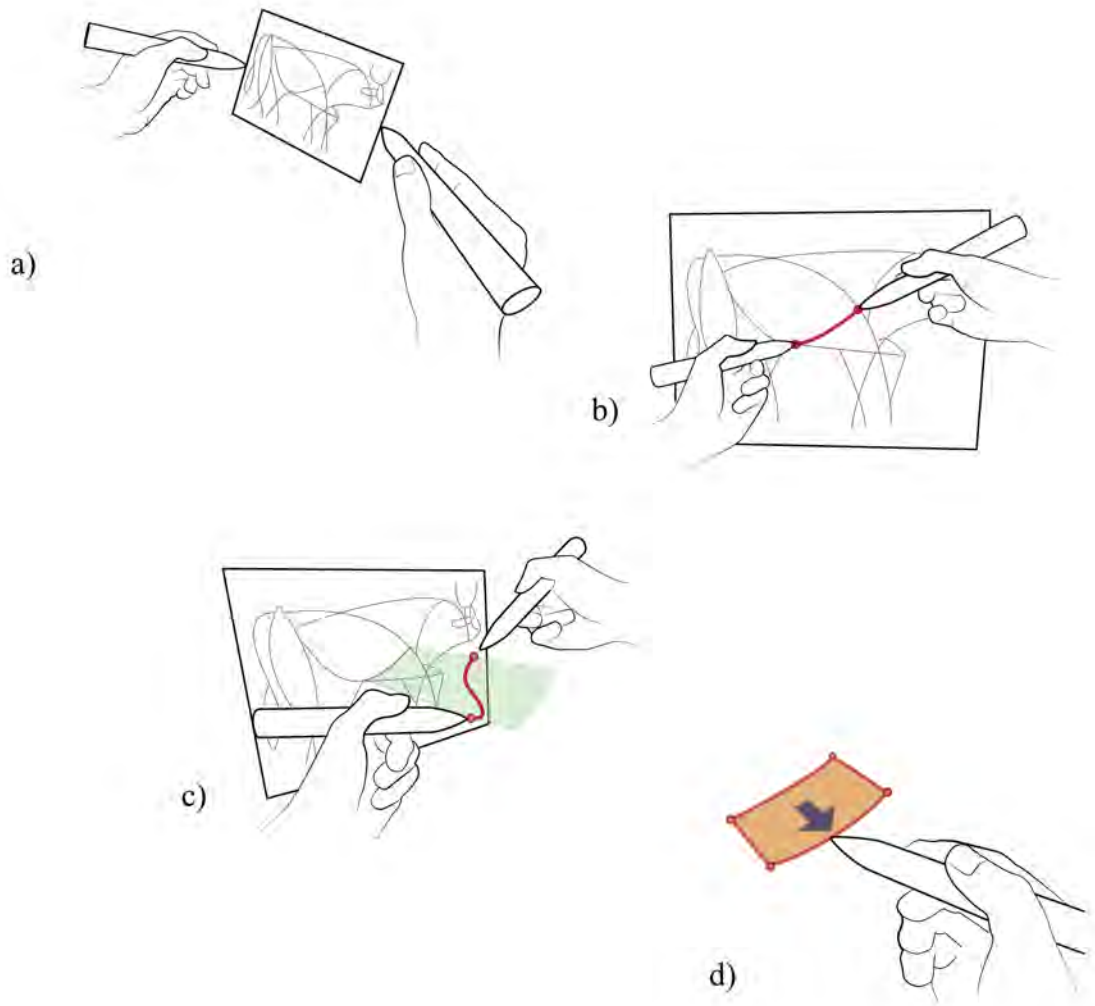

Figure 6.3: Modeling workflow. (a) An artist first places a 2D image in space. (b-c) He/she selects a contour in the image which creates a surface projecting from the slide (shown in green). The artist can then adjust the shape of the curve in 3D space while maintaining the projection to the original contour. (d) Finally, the artist selects the new curve and sweeps it through space to create a surface.

images. Gestalt theory holds that the cortical areas in the brain extract primitive features from the retina images which are aggregated through grouping into more complex representations [127]. Contours have been proposed as the primitives that define shape and object boundaries because they are easily detected even in the presence of distracting stimuli [128] and the curvature they represent has a high-degree of viewpoint-invariance [129, 130]. Work in cognitive science has suggested that form processing on the basis of contours may involve propagation of contour information away from the edges into the interiors of surfaces which explains our ability to generate and understand 3D curved surfaces given only the 2D contours in line drawings or silhouettes [131].

In the following sections, we describe the key features and algorithms of the rail-based surface creation technique.

6.2.1 Integrating 2D Concept Sketches and Photographs into the VR Environment

To integrate 2D photographs or concept sketches into the VR environment, the images are loaded as 2D textures. Each image is displayed as a slide in a grid virtually floating along one wall of the CAVE, see Figure 6.4. The artist selects a slide by reaching for it with both styluses, similar to how one might lift a picture off of a wall. When both styluses are close enough to slide, the image animates to look like it is grasped between the artist's hands. A button click on the primary drawing stylus confirms the selection, and allows the slide to be moved around and positioned in space.

As the artist is positioning the slide, moving the styluses further apart or closer together will scale the image to make it larger or smaller. Clicking the button on the primary drawing stylus locks it into its current position. Figure 6.5 shows an example of how this feature might be used by an artist. Several outline sketches are positioned around the work in progress. Additional photographs of the sculpture being modeled

Figure 6.4: 2D concept sketches and photographs can be placed as slides in the VR environment to serve as context for modeling actions. The slides are displayed in a floating grid near one of the CAVE walls. An artist selects a slide by reaching out and "grabbing" it using both styluses.

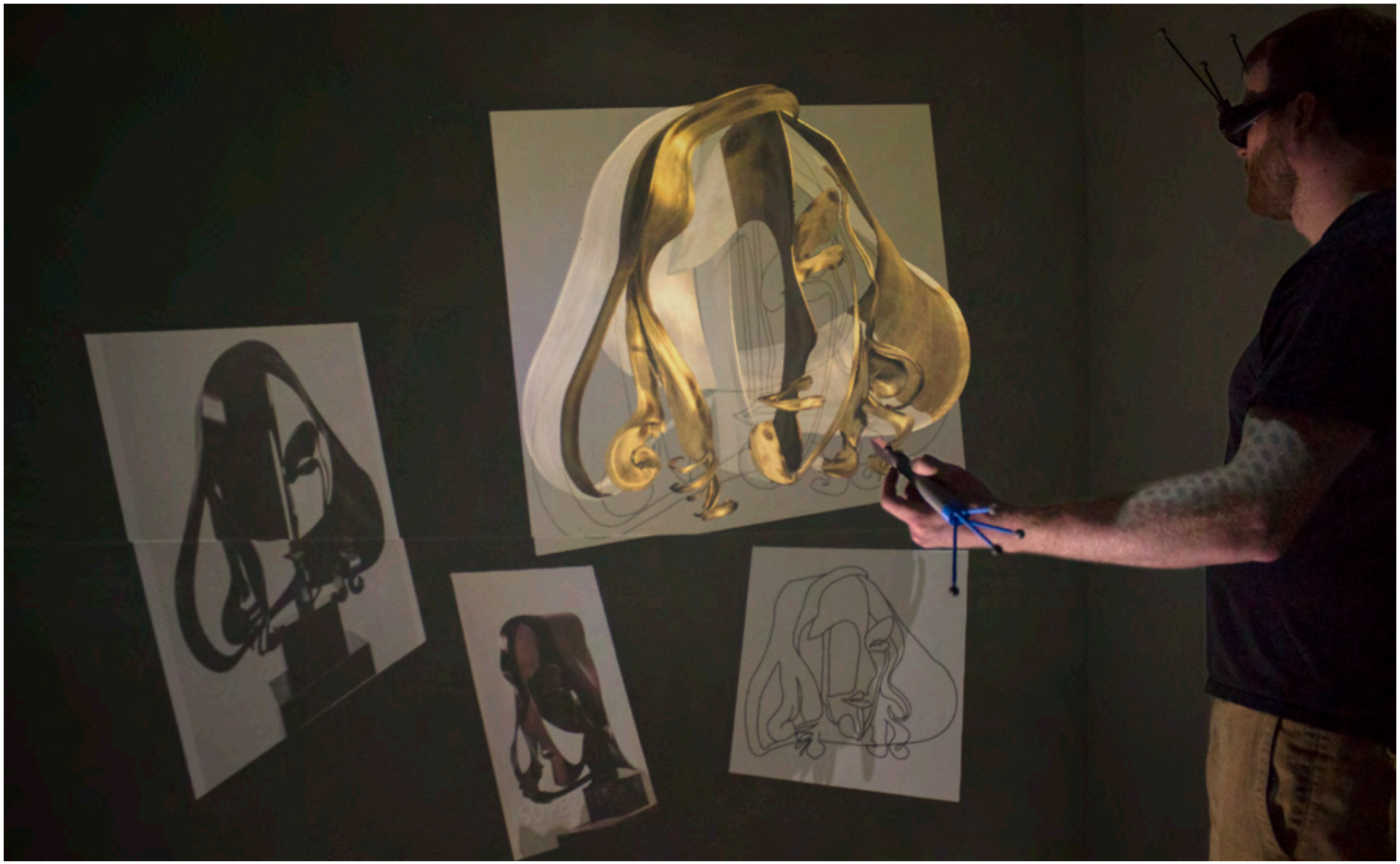

Figure 6.5: An example of how 2D imagery might by used in the VR environment. A primary sketch is placed in the scene and used to create rails and model surfaces. Additional sketches and photographs of the sculpture being modeled are placed in the space around the model to provide immediate reference.

are also shown in space.

6.2.2 Creating 3D Rails

To start the modeling process (shown in Figure 6.3), the artist must first create one or more 3D curves, called guide rails. The rails serve as a wire-frame outline of the model to help with proportion, and are used for creating the mesh surfaces as described in the next section. There are two different ways of creating rails: (1) directly drawing curves through space, and (2) pulling curves off 2D imagery placed in the modeling environment.

Directly Drawing Rails

The first way of creating rails is by directly drawing them through space in the style of Cave Painting [22]. The artist holds the tracked drawing stylus in his or her dominate hand, and while holding down a button, moves the tip through space indicating the centerline path of the rail.

Each rail is represented as a 3D Catmull-Rom Spline [132]. As the user moves the stylus, the position is tracked over time and used as spline control points. A rail is rendered as a tube by interpolating points along its spline and connecting them with small cylinders. The beginning and endpoints of the rail are rendered with slightly larger spheres. This spline-based approach smooths out small jitters in the stylus position as it is moved through space.

In order to draw connected rails, the virtual representation of the drawing stylus will snap to an existing rail endpoint if the tip is within a close radius. Our implementation uses a snap radius of 2.54 cm. Starting to draw a rail from a snapped endpoint will connect the rails at the starting point. Similarly, an artist can connect the end of a rail he or she is currently drawing by releasing the drawing button within the snap radius

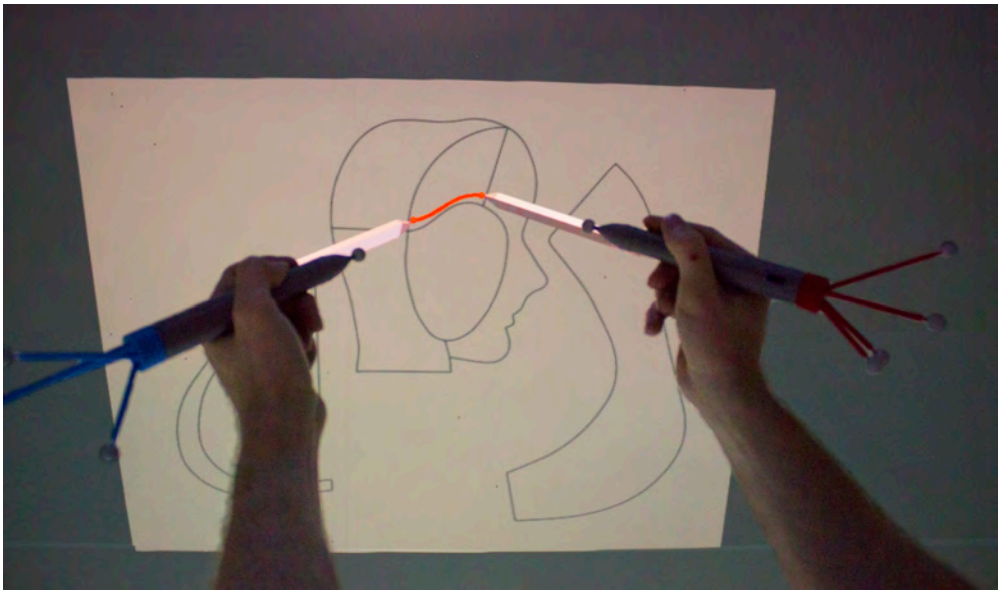

Figure 6.6: Contours are selected in 2D imagery using a spline projected onto the surface of a slide. The spline curvature is adjusted by rotating the styluses. The spline will morph slightly using the underlying image context to improve selection control.

of an endpoint. While the drawing button is held down, the virtual cursor does not snap to avoid inadvertently changing the drawn curve, but endpoints within the snap radius are highlighted in yellow to indicate that a possible connection will be created if the button is released.

Creating Rails from 2D Imagery

The second way of creating 3D rails uses the context of 2D contours from inspirational slides that the artist has placed in the scene. When both styluses are close to a slide placed in the modeling environment, a cubic Bézier curve is drawn between them projected on the surface of the 2D imagery, as shown in Figures 6.3b and 6.6.

Each stylus is used to set two of the four Bézier curve control points (P_0–P_3). To set the curve's outer control points (P_0 and P_3), the positions of both styluses are

projected onto the plane of the slide. The two inner control points (P_1 and P_2) are created by translating each stylus' position along the vectors formed by their long axes, and projecting these locations onto the surface of the slide. The translation distance is set to half the distance between the styluses. Thus, when pointed directly at each other the resulting curve will not contain any loops. By pivoting the styluses, thus changing the position of the inner Bézier control points, the artist is able to adjust the curvature of the spline.

By moving the spline endpoints and adjusting the curvature, an artist can select contours in the inspirational imagery. The spline is set by clicking the primary drawing button. The spline will morph slightly using the context of the underlying imagery to more closely follow the path of the contour.

This is accomplished in a similar way to the Snakelet center-finding algorithm presented in Chapter 4. The algorithm has four steps: (1) Filtering and Thresholding, (2) Distance Map Calculation, (3) Gradient Calculation, and (4) Curve Point Iteration.

When the slide texture is loaded into the program, it is filtered and thresholded to identify which sections of the image correspond with drawn outlines (foreground) and which sections correspond with the blank canvas (background). First, the image is converted to grayscale using OpenCV [133]. The grayscale image is thresholded to convert it to a black and white image, and a morphological opening operation is applied to remove small noise pixels from the foreground.

In the second step of the algorithm, a Euclidean distance map is calculated for each pixel in the black and white image. Foreground pixels are labeled with the distance to the closest background pixel, while background pixels are labeled with the negative distance to the closest foreground pixel.

The gradient of the distance map is then calculated using Sobel derivatives. The derivatives are calculated in the horizontal and vertical directions and stored separately.

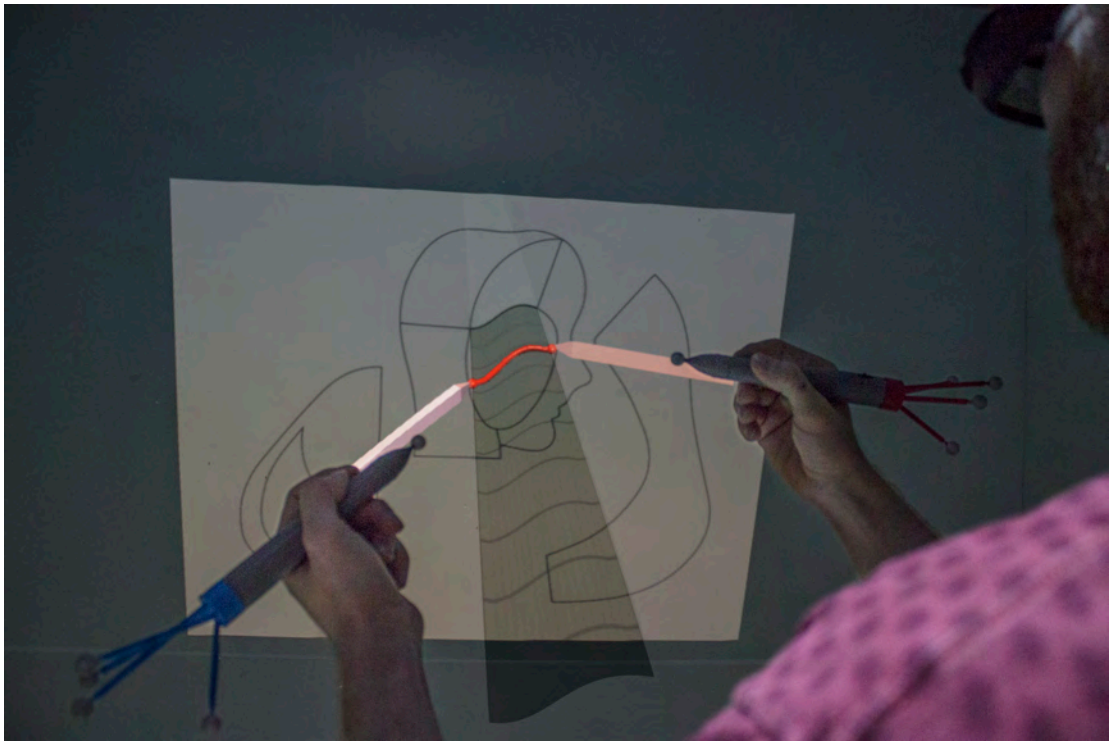

Figure 6.7: A 3D rail is extracted from a 2D contour in an inspirational sketch. The artist controls a 3D spline that is constrained to lie on the projection of the 2D contour.

Finally, to snap the artist's Bézier curve to the contour in the underlying image, a list of 3D points lying along the curve in the plane of the slide is calculated by interpolation. Each 3D point is converted to its u and v texture coordinate in the 2D slide image. Finally, the points, now in 2D, are iteratively refined by gradient ascent, using the gradient derivatives of the distance map. When a point converges, the outcome is that it lies on the centerline of the drawn line. These final (u, v) positions are converted back to their 3D locations, to get the final snapped curve.

Once the curve morphs to the underlying imagery, a guiding surface is creating, extending perpendicularly outward from the slide in both directions along the slide normal. The topology of the surface follows the path indicated by the morphed spline (See Figure 6.7).

A second Bézier curve is used to set the 3D shape of the rail along the resulting guide surface. The closest points on the edges of the guide surface to the styluses serve as end control points for the curve. Again, the interior control points are set by translating along the long axis of each stylus by half the distance between the styluses. Finally, to make the rail, points are interpolated along the Bézier curve and projected to the closest point on the guide surface. These projected points serve as the control points for the final Catmull-Rom spline making the rail. When the artist clicks the drawing button again, the rail will be created in space.

Similarly to the direct drawing approach, rails can be created in this manner that connect to existing rails. When selecting a 2D contour in a slide, the endpoints of all existing rails are projected onto the slide surface. These projected points serve as snap points during the initial contour spline selection.

6.2.3 Combining, Dividing, and Deleting Rails

In designing a model, it is possible that the artist wants to combine, divide, or delete rails to change the surface topology created through the different types of surface sweeps shown in Table 6.1. This is accomplished using the drawing stylus. To combine two rails that share an endpoint, the artist snaps the drawing cursor to the connecting endpoint and clicks the secondary button. The endpoint is deleted and the two rails are combined into a single continuous rail.

Similarly, to divide a rail in two, the artist moves the drawing cursor to the point where he or she would like to divide the rail and clicks the primary drawing button. A new endpoint is added. The original rail is divided and replaced by two rails that join at the new endpoint.

Rails, slides, and surfaces can also be deleted using the secondary button on the drawing stylus. If the stylus is close to an existing rail or surface, the object will

highlight in yellow. Clicking the secondary button will delete it. Deletion operations also use the context of the rail connections, for example deleting the endpoint of a rail with no other connections will delete the rail.

6.2.4 Sweeping Surfaces along Guide Rails

Surface meshes are created by sweeping a profile rail along guide rails, in an approach that is similar to the Birail operation in Maya [134] or the Sweep2 operation in Rhino [135]. However, unlike these conventional modeling tools that require the artist to carefully specify each curve, our approach automatically determines the rails that are involved in the modeling operation. To create a surface, the artist selects a rail and pulls it in a direction. The resulting surface is based on two contextual factors: (1) The direction the artist initially pulls, and (2) The number and direction of the rails connecting to the selected rail.

A taxonomy of rail-based freeform 3D modeling is presented in Table 6.1. In Modeling with Context, rails are represented in an undirected graph data structure, where the nodes represent rail endpoints and the edges represent connections between rails. This allows us to categorize modeling operations based on the topology of the graph. The taxonomy uses this information, defining surfaces created using one, two, or three connecting guide rails.

The direction the artist sweeps the initial profile rail is used to determine which connecting rails serve as guides for the surface creation. The sweep direction is compared against the "direction" of each rail that connects to the initial profile rail. Calculating the a rail's direction is trivial if it is straight, as the average direction of the individual line segments that form the rail can be used. However, this approach does not work as well when the rail is curved; consider the case where the user draws a semi-circular rail. In practice, we found that averaging the segment directions for the first third of each

Table 6.1: Possible surface sweeps depend on the number of connecting rails.

Sweep Type	Rail Connections	Resulting Surface	Description
One Rail			Single rail sweep. The surface is created by duplicating the selected rail and connecting the sides using straight edges.
Two Rails			The surface is created by duplicating the selected rails and joining them at the opposite intersection point.
Three Rails			The selected rail is rotated and scaled to intersect the opposite endpoints of the guide rails.
			Special case. If the guide rails are connected by another rail, the surface is created by interpolating between the selected rail and the end rail.
			Special case if the guide rails are connect to each other.

rail worked well as a compromise between avoiding noise from averaging a low number of line segments and avoiding errors from highly curved rails.

Once the directions for each rail are calculated, they are compared with the direction the artist initially pulled. If the angle between the two vectors is within a threshold (60 degrees in our implementation), we assume that the artist meant to sweep along the rail. Note, that if there are multiple connecting rails at a single endpoint of the selected rail, only the one with an orientation most similar to the initially pulled direction is considered for determining the sweep case.

Once the connections and sweep type are determined, a surface mesh is calculated. Throughout the rest of this section we will refer to the specific rails involved in the sweep operation using the following terminology: The initially selected profile rail is called the beginning rail; the rails that the profile is swept along, creating the sides of the surface are called the guide rails (Note, that in the one and two rail sweep cases, these might be automatically calculated rather than drawn by the artist, as described below); the profile at the end of the sweep is called the end rail.

The algorithm proceeds using the following steps:

1. Create missing guide rails for the one and two rail sweep cases.
2. Identify or create the end rail.
3. Calculate interpolated profiles between beginning and end rails.
4. Align the interpolated profiles with points along the guide rails.
5. Create vertices at each interpolated point, offset along the positive and negative normal directions to add surface thickness.
6. Triangulate between the vertices to create a mesh.

First, for the one and two rail sweep cases, the missing guide rails must be calculated. For the one rail case, the guide rails are formed by linearly interpolating between the endpoints of the beginning and end rails. For the two rail sweep, the existing guide rail is copied and translated so that its starting point intersects the opposite end of the beginning rail, forming the opposite side of the mesh.

Next, the end rail is determined based on the sweep type. For a one rail sweep, the end profile is a copy of the beginning rail, translated to stay attached to the artist's stylus. For two rail sweeps, the end rail is a copy of the beginning rail scaled, translated, and rotated to intersect with the opposite endpoints of the guide rails.

For three rail sweeps there are three possible options for the end rail. If there is an edge in the rail graph connecting the opposite endpoints of the two guide rails, the connecting edge is set as the end rail. If there is not a connection between the opposite endpoints, the end rail is calculated in the same way as the two rail case. The beginning rail is scaled, translated, and rotated so that its endpoints intersect the opposite endpoints of the guide rails. Finally, the last possible option occurs when the opposite endpoints of the guide rails meet at the same point (forming a triangular shape). Here, the beginning rail is scaled, translated, and rotated to line up with the second to last point on both guide rails. The extra triangular region between the second to last points and the final end intersection point is handled separately as discussed below.

Once all the rails that form the sides of the surface are determined, the interior surface profiles must be calculated by interpolating between the beginning and end rails. First, a transformation matrix is calculated that scales, rotates, and translates the end rail so that its endpoints intersect with the endpoints of the beginning rail. The points that make up the end rail are then multiplied by this transformation matrix to convert them into the same coordinate space relative to the beginning rail. This avoids issues where the length of the beginning and end rails are dramatically different. Points

can then be interpolated between the sets of beginning and transformed end rail points.

In the next step, an equivalent number of intermediate points along each guide rail are calculated by interpolating along the rail splines. To align the interpolated interior profiles calculated in the previous step with the guide rails, the interpolated profiles are scaled, translated, and rotated so that their endpoints intersect the intermediate points on both sides of the mesh.

To create vertices for the final mesh, surface normals are first calculated from adjacent interior profiles. The points along each profile are duplicated and offset along the positive and negative surface normals to create the vertices. This adds thickness to the resulting mesh. The offset distance can be set by the artist as a configuration value loaded at runtime. In our application, the offset distance is set to 2.3mm, which replicates a piece of thick sheet metal that might be used for creating physical sculptures in the Gargallo style.

Finally, a triangulated mesh is created by joining adjacent vertices along the top and bottom surfaces. The two sides are connected along the thin edge by additional triangle strips linking the vertices that were offset along the normal directions.

There is one special case when the two guide rails end at the same point. As mentioned previously, the end rail in this case connects the second-to-last intermediate points on the guide rails. To create the final triangular mesh, an additional triangle fan is created linking each point in the end rail with the final intersection point of the guide rails.

One of the advantages this spatial interface has over conventional tools, is that the artist can easily specify how far to sweep the surface along the guides, potentially stopping earlier than the full length. Although the entire surface is calculated for the full length, only a portion is displayed to the artist, which increases as they sweep further.

To display a portion of the surface, the distance the artist has pulled the stylus

from his/her initial click point is mapped along one of the guide rails. Only the surface vertices that connect to the sides before this point are displayed as a mesh to the artist.

6.2.5 Reorienting and Scaling the Model

Reorienting the artwork is accomplished using the secondary stylus in the non-dominate hand. Clicking and holding the button on the stylus grabs the virtual space. This behaves similarly to picking up a real physical object, where changes in the orientation or position of the stylus also move the virtual space in a corresponding way.

To scale, while grabbing the space with the secondary stylus, the artist also clicks and holds the primary button on the primary stylus. This is like grabbing the virtual space with two hands. Moving the hands further apart increases the size of the virtual artwork similarly to how you might stretch an object. Moving the hands closer together scales down the virtual artwork.

6.2.6 Modeless Transitions between Modeling Operations

One of the primary advantages of using a spatial interface is that the user does not need to explicitly set the current interaction mode. Instead, the current modeling operation is set based on context. In our implementation, this is based on the proximity of the cursors to existing objects in the artwork. If both of the styluses are held near a slide placed in space, the interface automatically transitions to selecting a 2D contour in the slide imagery. Otherwise, the interface remains in a direct drawing mode. This type of modeless interface has previously been found to reduce mode related errors [100].

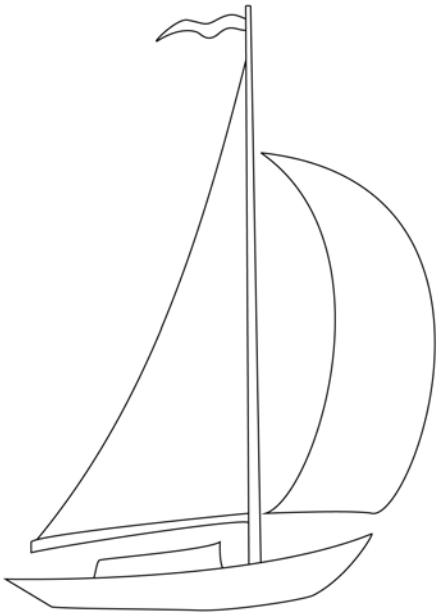

Figure 6.8: Sailboat sketch used during the user evaluation task.

6.3 Exploratory User Study

To evaluate the impact of modeling spatially, we present results from an exploratory user study evaluation.

6.3.1 Methods

The user study was designed as follows:

Participants. Three university students and a professional architectural designer participated in the study. Half of the participants are female. All reported limited prior use of virtual reality systems. Experience with 3D modeling software varied from 5–20 prior uses (2 participants) to more than 20 prior uses (2 participants). Two participants also reported at least occasional video or computer game use.

Task. Participants were asked to model a sailboat using the 2D sketch shown in Figure 6.8. A sailboat was chosen specifically because sails commonly have an organic shape that would be difficult to model with conventional software. Participants placed the sketch in the VR environment, after which they were told to take as long as they needed to model a sailboat using any of the modeling techniques previously described.

Training. To start participants thinking creatively about the 3D form of a sailboat, each participant was asked to browse Google image results for the search term "sailboat" for two minutes. Following this introduction, each participant was given a walk-through of the program's features while he/she performed each action. Individual actions were repeated until the participants were able to successfully complete them. Finally, the participants were asked to start modeling the base of the sailboat. Participants were encouraged to ask questions and given reminders about how to use specific features. Once they demonstrated proficiency with the tool by creating several surfaces for the base, the program was restarted and the participants began the modeling task on their own. Training times varied between 10 and 25 minutes for each participant. Overall, each participant spent about one hour in the virtual reality environment.

6.3.2 Results

Models created during the sessions are illustrated in Figure 6.9. The software automatically logged system events and created a summary report for each participant. The logged summary data is shown in Table 6.2. The author's results for the task are also included as a point of comparison for an expert user of the system. The author's model is shown in Figures 6.14 and 6.15.

6.3.3 Analysis and Discussion

This section analyzes and discusses the results of the exploratory user study evaluation.

Figure 6.9: Sailboat models created during user evaluation. First column: participant 1. Second column: participant 2. Third column: participant 3. Fourth column: participant 4

Table 6.2: Summary of logged data from a user evaluation modeling a sailboat. Values indicate the number of occurrences.

Participant	Time (min)	Freehand Rails	Contour Rails	Split Rails	Joined Rails	One Rail Sweeps	Two Rail Sweeps	Three Rail Sweeps	Rail Deletions	Surface Deletions	Moves	Scales
P. 1	32	19	24	7	5	7	1	5	40	4	197	9
P. 2	54	69	24	52	35	31	37	27	175	53	483	51
P. 3	28	16	63	23	14	5	5	8	58	7	92	23
P. 4	21	88	1	51	54	18	6	22	115	27	101	16
Author	32	8	57	7	15	2	1	23	34	9	186	24

Combining Freehand and Contour-based Rail Drawing

To understand the use of the tool, one important aspect is the ease with which an artist can combine freehand drawing in space with the contour selection from a slide. As shown in Table 6.2, all participants used some combination of both techniques.

In particular, participant 4 used almost all free-hand drawing (88 free-hand rails, 1 contour rail). She described herself as impatient. Even though the contour-based rails created smoother curves, she felt that the two step process (indicating a 2D contour, placing it in space) was not as immediate as directly drawing. This finding is also supported by the fact that she was able to create her model in the quickest amount of time (21 minutes) compared to the other participants that used more contour-based rails. There is clearly a trade-off between the immediacy of freehand drawing and the control provided by the contour approach.

Participant 2 also reflected on this trade-off, but expressed the need for both rail creation approaches. Like Participant 4, he also used more freehand rails (69 free-hand rails, 24 contour rails), although this can be explained in part by the additional features of the anchor and signature that he added to the model (Figure 6.9 second column), which do not appear in the original sailboat sketch. He reported that he liked the "straight line control [of the contour approach] but also the ability to express yourself [with freehand]".

Interestingly, both participants that used the most freehand rails also had the most rail deletions (175 for participant 2 and 115 for participant 4). We speculate that this is caused in part because the difficulty of freehand drawing caused more errors, i.e. rails that did not have the shape that the user wanted, which were deleted; however, we also saw that the immediacy of freehand drawing enabled more exploration of form. One user who did not participate in the study expressed the importance of easily deleting

rails, saying "I like that I can make stuff and then make it go away". We attribute this to previous research (e.g. [1]) that has found the ability to undo to be particularly important for encouraging creative exploration.

In terms of workflow, most participants built up a wireframe of rails to define the form before filling in surfaces. This can be seen by the greater number of two and three rail sweeps (with the exception of Participant 1), and also the greater number of rail deletions compared to surface deletions. From observing the modeling process, surfaces were most often deleted when they occluded rails or parts of the slide where the participant wanted to work. After the participants finished working in the occluded region they would then recreate the deleted surface. This indicates that some form of 'x-ray' lens feature might be useful in future implementations to avoid issues of occlusion.

Surfaces that were deleted because of errors were most often caused when there were multiple connecting rails running in almost the same direction. In this case, if the user did not initially pull exactly in the direction of the specific guide he/she wanted the contextual analysis choose the wrong guide. It was easy for participants to delete the incorrect surface and re-sweep by pulling in a more precise direction, but future work might look at ways of swapping the guide rail dynamically as the user pulls in different directions, rather than basing the guide rails on just the initial sweep direction.

During the modeling workflow, participants also liked the ability to hide the rails and slides. This allows the artist to get an idea of how the final model will look without the "scaffolding" occluding the view. Clicking and holding the secondary button on the stylus in the non-dominate hand will hide the rails and slides. Releasing the button makes them reappear. Like the rest of the interface, this feature is context sensitive. For example, if the button is clicked while the artist is creating a rail by pulling a contour along a guiding surface (See Figure 6.7), the other rails will hide, but the guide surface will remain visible until the rail is set in space.

Splitting and Recombining Rails

The ability to combine and split guide rails is particularly important for modeling in this style. Participant 2 described a technique he used for more controlled freehand input by splitting rails. He found that by moving his entire arm quickly through space he was able to draw smoother rails. The downside to drawing quickly is that then it becomes difficult to start and end the rail at specific points. His approach was to draw quickly without focusing on the endpoints. After the rail was created he could easily split it at exactly the endpoints he wanted and delete the extra end sections. Using this technique, he had the highest number of split rails (52 occurrences) and deleted rails (175 occurrences) of any participant. We have also seen other users use this technique effectively, particularly when they want to end a rail near the surface of a slide.

Combining 2D and 3D Workflows

The combination of 2D and 3D workflows is particularly important for next-generation expressive spatial interfaces. One participant in the user study expressed this thought, saying "I like that what you draw on a napkin in a coffee shop doesn't go to waste". He refers to the ability to place 2D imagery directly in the virtual reality environment. This enables artists to still utilize the 2D design process of sketching and ideation that they are used to, while leveraging the benefits of modeling spatially.

One participant mentioned that she liked the ability to place the slides in space rather than just seeing them projected on a wall. This let her use both sides of the slide, sometimes pulling contours towards herself other times pushing them away on the opposite side of the slide. This task was facilitated by rendering each slide slightly transparent so that the artist can see the contours behind it.

Even for artists that do not use the contour-based approach for rail creation, the

slide was still useful for indicating correct proportions. Participant 4 used almost all freehand drawing; however, she created her model centered around the slide. Each surface roughly aligned with contours in the slide.

Several of the participants also demonstrated a surprising side-effect of modeling around a 2D slide placed in space. The flatness of the slide as one of the only frames of reference in the modeling environment may have impacted the rails that the participants created. For instance, several participants pulled contours straight out from the slide and keep them relatively parallel to the slide surface. This can be seen in the sails in the models by participants 1 and 2 in Figure 6.9.

Participant 3 took this approach to the extreme by creating a completely planar sailboat. His model is interesting because although it is flat, each surface has a very precise shape defined by the contours. This demonstrates the control that artists have using the contour-based approach to rail creation. It would be almost impossible to model this flat form using freehand gestures in space.

The effect of the planar frame of reference provided by the slide seems to be minimized with more experience using the tool. Contrast the flat edges of the participants' models with the more complex curves seen in the author's sailboat model (Figure 6.14). The impact of a reference frame for novice users is also supported by studies of young children learning to draw in 2D. It has been shown that young children have trouble drawing horizontal, vertical, and oblique lines inside a circular reference frame, but are significantly more accurate when drawing inside a square [136, 137]. Even using the rectangular borders of the piece of paper can serve as a reference frame to increase line drawing ability of straight lines in young children.

Scaling, Reorienting, and Physicality

Each of the participants increased and decreased the scale of the virtual model many times. Participant 3 specifically commented on this feature's usefulness. When modeling the flag at the top of the mast, he found that at a 1:1 scale between the virtual space and the physical space the endpoints for these rails would be within the snap radius of each other making it difficult to draw connecting rails from a specific endpoint. However, because the snap radius is measured in physical space, he was able to increase the scale, essentially increasing the distance between endpoints. This gives an artist an increased amount of precision when snapping to a specific guide rail, or choosing a rail to sweep into a surface.

Participants also found that ability to reorient the model to be useful. For instance, to model the mast of the sailboat, several users found it awkward to select a long vertical contour, reaching one arm far above their heads while keeping the other at waist level. By rotating the slide and model on its side, they were able to keep their arms horizontally out in front of their bodies and comfortably create the mast rails.

Reorientation of the model was also combined with changing the physical viewpoint within the CAVE. Each participant walked around the model or peeked over surfaces to see behind them. Some participants even squatted down to look up at the sails. We observed participants reacting to the presence of the virtual model surfaces almost as if they were really physical by walking around them rather than through them.

This type of spatial engagement with the digital artwork is very different than a traditional modeling experience. In future work, we would like to explore the impact this might have on artists' creativity when they are able to engage their entire body during the 3D modeling process. During a recent demonstration of the tool to a local architect she remarked on the spatial nature of the interface and its potential impact on

her work. “What I really like is that I can use my whole body. There’s more freedom with fewer constraints. I feel like I can be a little more creative.”

Creativity Impact

We were surprised by the different modeling results participants were able to create based on the same initial sketch. In particular, the participants were able to create their own additional creative features to help define the model. For example, participant 2 added a signature and anchor to his boat. Participant 3 added a sailor climbing in the rigging. Participant 4 added a keel and rudder that did not exist in the original sketch. Although we used the same sketch for consistency in comparing between the different results, several participants expressed the desire to use their own sketch of a sailboat to start modeling.

6.4 Additional Results

In addition to the user study modeling results, we demonstrate the control and expressiveness that this interface allows through recreations of existing 3D sculptures and original models created by the first author using the Modeling with Context application.

6.4.1 Models Created by the Author

Pablo Gargallo’s sheet-metal sculptures (See Figures 6.1 and 6.10) are a primary motivation for the work presented in this chapter. The complex shapes and curves that make up each sculpture intertwine and arc gracefully through space. Yet, at the same time, these organic curves are what make his sculptures so hard to model using conventional 3D modeling tools.

The ability to recreate Gargallo’s sculptures with the same amount of control and

Figure 6.10: Left: Greta Garbo Con Sombrero. 1930 [138]. A sculpture by Pablo Gargallo. Right: Paper templates Gargallo used for creating the sheet-metal pieces that form the sculpture [139].

expressiveness that Gargallo had when working with physical materials has been a driving motivator and a benchmark for our success.

The following sections present modeling results, both replications of existing Gargallo sculptures as well as original pieces of 3D digital art, created by the author. These examples illustrate the level of control and expression afforded by the spatial interaction techniques with an expert user of the system.

Recreations of Gargallo's Sculptures

Figure 6.11 shows a digital recreation of Pablo Gargallo's sculpture called *Greta Garbo Con Sombrero*, which was created using the tool. Compare the model with the image of the original sculpture in Figure 6.10. One of the first bits of evaluative feedback comes through visual comparison. The system allows artists to faithfully recreate these types of complex forms in 3D. With almost no prior 3D modeling experience, this sculpture

Figure 6.11: Recreation of Pablo Gargallo's sculpture, *Greta Garbo Con Sombrero*.

Figure 6.12: *Leo*. Lion mask sculpture created by the author.

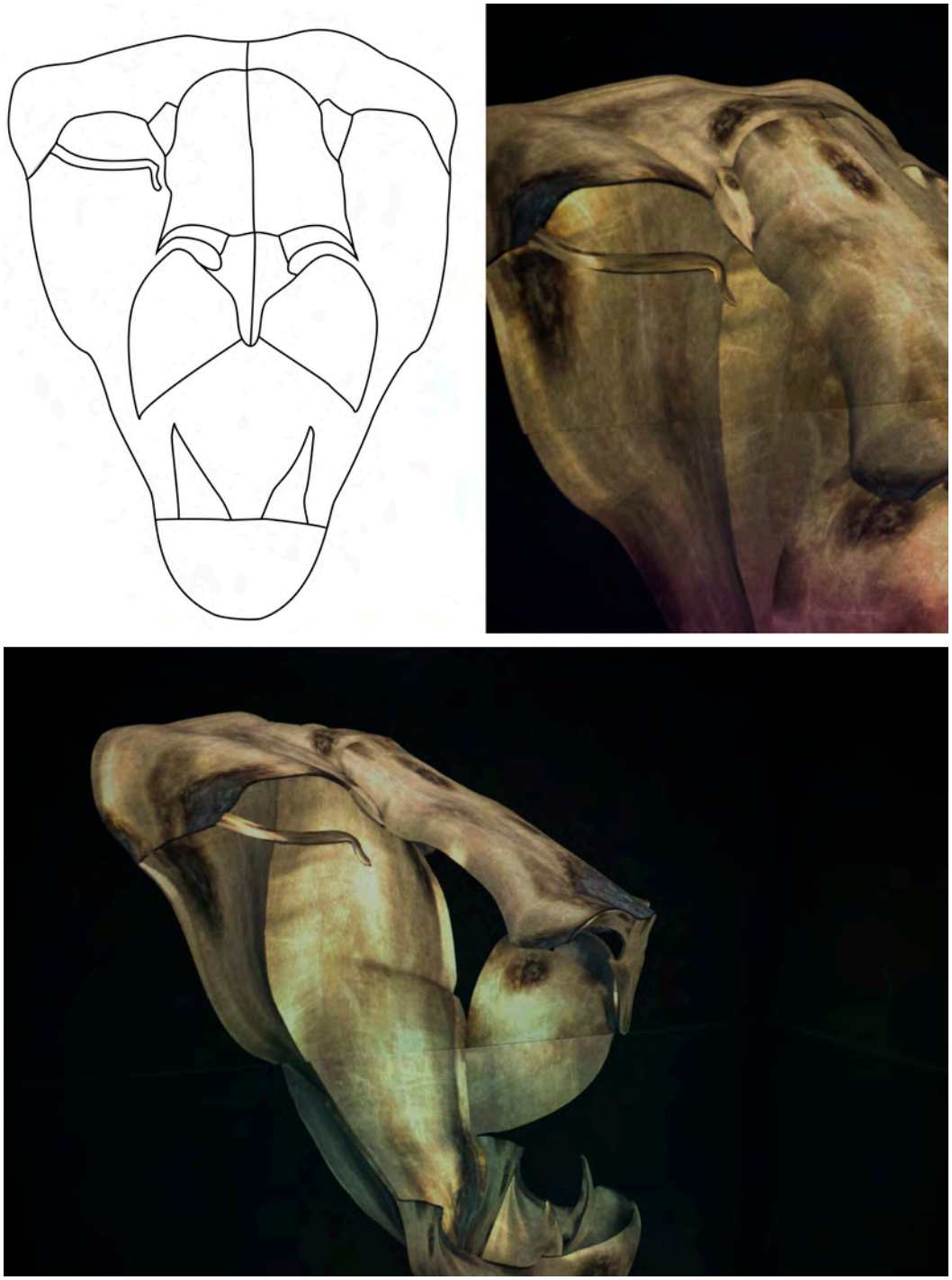

Figure 6.13: Details of the lion mask sculpture.

Figure 6.14: Sailboat modeled by the author.

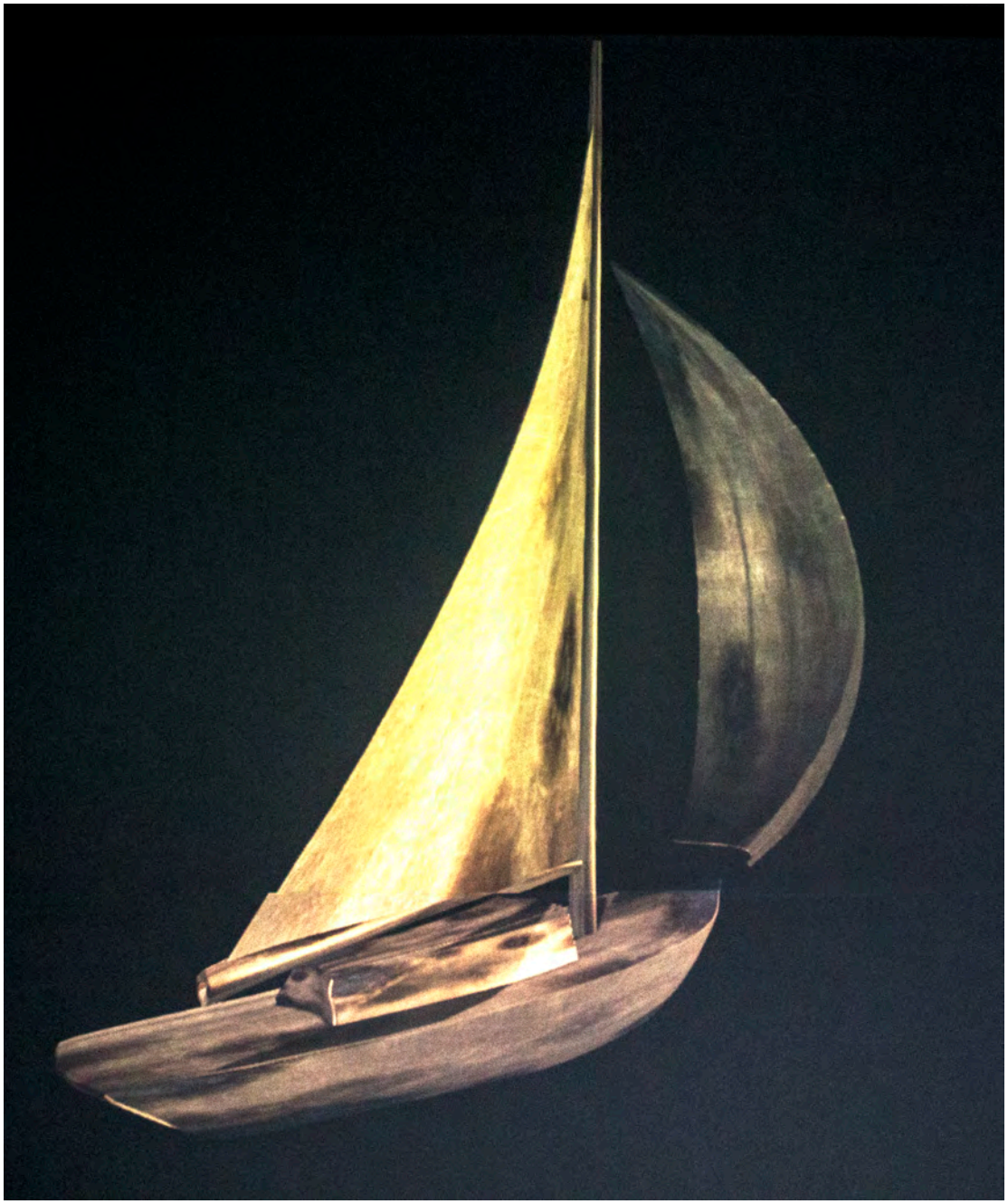

Figure 6.15: Sailboat modeled by the author.

was recreated in about five hours.

The digital sculpture was created using a variety of the spatial modeling techniques described previously. First, the profile of the face was created by placing a slide in the center of the VR cave. The slide texture showed outlines of the original paper templates shown in the right half of Figure 6.10. Contours were selected for the face and head profile, using the techniques described in Section 6.2.2, and were used to create guide rails for the face cross section. Then, a second slide showing traced outlines from a photograph of the original sculpture was placed perpendicularly to the initial profile slide. Contours from this slide were used to create guide rails to form the majority of the surfaces.

Original Models

In addition to recreating existing sculptures, the interface enables artists to create new digital works of art from inspirational imagery. Figures 6.12 and 6.13 show an original sculpture of a lion mask created by the author from a 2D sketch of a lion's face in about an hour and a half.

Figures 6.14 and 6.15 show a sailboat created from the sketch used during the user evaluation (Figure 6.8). Although best viewed using a head-tracked stereoscopic display, even printed on paper the images of these models show the great amount of expressiveness an artist has to create surface details, such as the shape of the lion nose or ruffles in the sail, using this spatial interface.

Figure 6.17 shows another example of an original sculpture. In this case, rather than using an original sketch, the model was inspired by the stylized lithographs of a bull by Pablo Picasso. Using the image shown in Figure 6.16 as inspiration, the model is conceptualized as a 3D structure of overlapping plates curving outward to define the form of the chest. This model was created in about two hours, which includes ideation

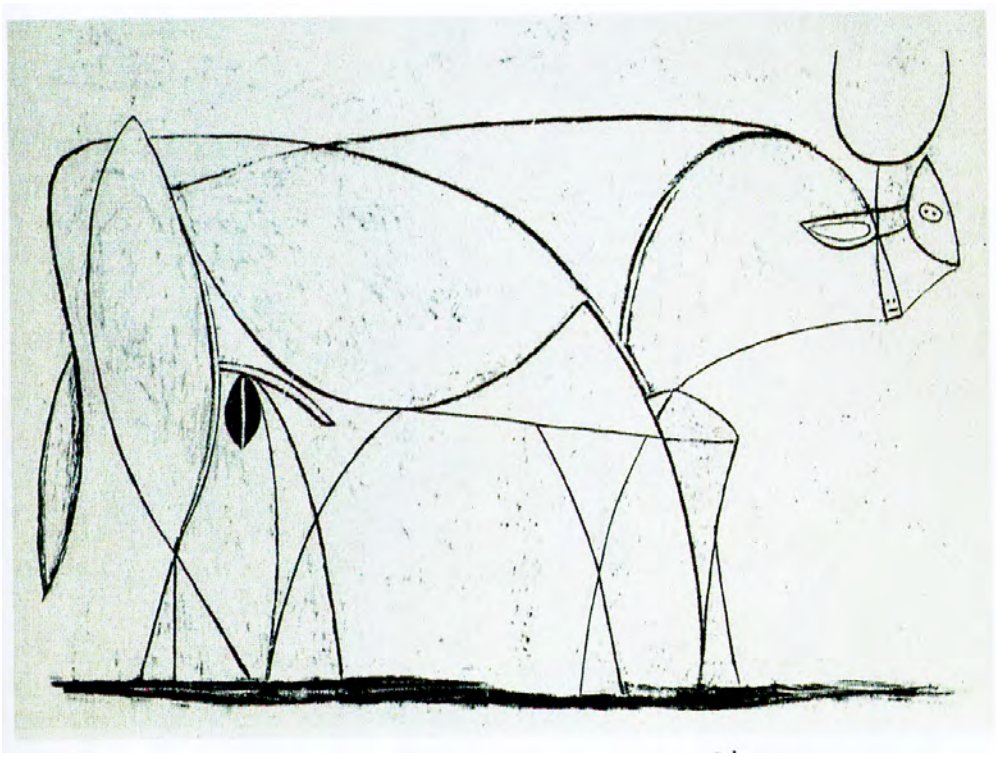

Figure 6.16: Bull (Plate IX). Lithographic plate of a stylized bull created by Pablo Picasso in 1946 [140]. This image served as inspiration for the bull model shown in Figure 6.17.

Figure 6.17: Sculpture of a bull created in the Gargallo style by the author.

and time experimenting with different 3D forms.

6.5 Discussion

This section discusses strategies for effective use of the tool.

6.5.1 Use of Reference Materials

The use of reference images for contour-based rail creation is particularly effective for creating smooth 3D curves with a higher level of control than freehand drawing. Although jitter in the hands makes it difficult to precisely set a spline curve that exactly matches a 2D contour in an images, morphing the spline slightly using the context from the underlying imagery allows the artist to smoothly and quickly place a 3D curve in space.

The control this provides, both in terms of creating smooth curves as well as correct proportion, can be seen in the sculpture of Greta Garbo which used a combination of freehand drawing and contour selection. Most of the model was created using guide rails from the reference imagery, allowing its form to closely match the real physical sculpture.

6.5.2 Freehand Drawing

Although freehand drawing is less controlled, the immediacy it provides makes it equally important for an expressive modeling interface. For example, the curly hair in the lower left of the Greta Garbo model was created directly through free-hand drawing of the guide rails. The tight curves of the hair lent themselves well to this looser style of input.

During the creation of the models shown in Section 6.4, free-hand drawing was frequently used as a way to quickly explore possible shapes by sketching rails off to

the side of the workspace. These sketches were fast and messy, with rails sometimes drawn on top of each other as the curves were refined. Once the shape of a curve and how it related to the existing rails was defined, the contour selection method, with the additional control it provides, was used to create the final smooth curve. The rail concept sketches could then easily be deleted.

6.5.3 Creating Long Rails

One limitation of bi-manual spatial interfaces is that users are limited by the length that they can spread their arms apart. This can make it difficult to select long contours in a placed slide. This issue can be avoided using two different methods, which participants in the exploratory user study employed naturally without any prompting.

The first method for easily creating long rails is simply to reduce the size of the slide by scaling the virtual space. This makes the contour endpoints closer together so that the artist does not need to reach as far apart. For example the edges of the bull's surfaces in Figure 6.17 were created by reducing the scale to be able to reach the full length of the bull's back.

The second method creates long rails by creating multiple smaller rails along a contour and then recombining them. This method works best for selecting long complex contours in a slide. The contour selection interface uses a spline with only three inflection points which limits the possible curvature that can be selected in one operation. For example, even though the spline will morph to more closely fit a complex contour, it would be difficult to select a spiral shape in one operation. This feature was used during the creation of the *Leo* mask sculpture shown in Figure 6.12. The back of the mask was created by selecting multiple connecting contour sections to result in a complex series of curves. These rails were then combining into a single rail to sweep the back surface.

6.6 Conclusion

Modeling with Context demonstrates the potential of combining spatial interfaces with contextual interpretation. It provides ways of integrating 2D inspirational imagery directly into the VR environment to help define 3D rails used for surface creation. Using these mechanisms, artists are able to create complex surfaces with a great amount of control and expressiveness, as shown in our modeling results. We attribute the ease of creation and control the system provides, even with novice users, to its ability to leverage the contextual information when determining how to respond to user input.

Our modeling results show how the interface can be used in multiple ways. We have demonstrated how it can be used to recreate existing sculptures such as those by Pablo Gargallo. Additionally, it can be used to create new sculptures inspired by existing artwork such as Picasso's bull, or entirely new models leveraging the artists ideation sketches, as in the lion mask. Modeling with Context takes an important first step towards more intelligent spatial user interfaces by giving artists the control and expressiveness they need to create models in this style.

7

Conclusions and Future Work

In this final chapter, we summarize the primary contributions of this dissertation and present a discussion of future work.

7.1 Summary of Contributions

The primary contributions for the interfaces presented in this dissertation and the conclusions we can draw from them are discussed below.

7.1.1 Spatial Multi-Touch Gestures

In Chapter 3, we address the issue of intuitive multi-touch interaction with stereoscopic 3D content. Interacting with traditional 2D multi-touch is limiting for expressing some 3D operations, while interacting in the air above the surface suffers from a lack of control and increased fatigue.

To address this issue, we develop the concept of anchored multi-touch gestures. This concept extends multi-touch interfaces by using gestures based on both multi-touch surface input and 3D movement of the hand(s) above the surface. We call this

type of interaction anchored, as the gesture is performed while maintaining an anchoring touch to the surface. This allows us to leverage the benefits of using the surface as a passive haptic support, improving stability and control as well as lowering fatigue when compared to freehand 3D input in the air. Users move fluidly between traditional 2D gestures on the surface, and anchored 3D gestures above the surface.

We introduce a taxonomy to characterize the design space for anchored multi-touch gestures, and present two example applications: (1) an interface for navigating 3D datasets, and (2) a surface bending interface for freeform 3D modeling. Both of these applications use anchored gestures to interact with 3D content on a head-tracked stereoscopic multi-touch display.

The interface was evaluated with a formal user study where participants ranked the anchored gestures as more intuitive then the state of the art interfaces used for comparison. In addition to this formal evaluation, we provide a discussion of qualitative insights and lessons learned during the creation of the tool and through use with a collaborating mechanical engineer.

We conclude that spatial interfaces, like Anchored Gestures, provide increased expressiveness over traditional 2D multi-touch gestures when working with 3D stereoscopic content. Additionally, the passive haptic support that the touch surfaces provides has the potential to improve control and lower fatigue compared with free-hand interactions in the air above the surface.

Combined in this way, spatial gestures and multi-touch are particularly well suited for complicated tasks such as artistic surface bending where the physicality and multiple concurrent inputs can be used to define the resulting surface. This style of interface is also beneficial for scientific visualizations that combine 2D and 3D data because the combination of 2D and 3D input enables users to pick the most appropriate degree of freedom for a specific task.

By adding a small amount of additional sensing capability to current multi-touch hardware, these types of anchored multi-touch gestures can easily be captured to increase the ability of artists and scientists to work intuitively in a controlled and expressive way with 3D content in a multi-touch environment.

7.1.2 Prop-Based Visualization for Future Scientific Workspaces

Recently, 3D printed rapid-prototypes have been shown to give scientists a better spatial sense of the data and facilitate navigation in visualizations [47, 48]; however, they are still costly to produce for use as tangible props. Inspired by watching collaborators gesture with a pen or pencil while discussing their data, we envision a future where scientists are able to pick up objects already in their workspace and use them as props to query and interact with their 3D data.

Although we were unable to track objects as small as a pencil with current low-cost tracking technology; we explore how lightweight (i.e. low-cost and easy to produce) props made out of paper, also found in a scientist's office, can be used to make spatial interaction more controlled. We anticipate that as tracking technology advances in resolution, the spatial gestures we have contributed will transfer directly to interacting with other cylindrical props such as a pen.

The user interface was designed to support exploratory visualization of thin fiber structures, commonly found in microscopy visualizations of biological tissues. From a scientific standpoint, the most critical aspect of these data to understand is the geometric alignment of fibers. We contribute a prop-based interface to query fibers based on their orientation. In addition to using the paper prop to support querying fiber directions, the gesture set includes actions to reorient the 3D volume to investigate the data from different viewpoints and set scalar values.

To support these interactions using a low-cost depth-sensing camera, new tracking

algorithms were developed to compensate for the low-resolution and noisy tracking data. Additionally, to realize the fully-featured visualization system, we developed an algorithm for extracting thin, dense fiber features (centerlines and diameters) from volumetric data.

Based on our evaluation with domain scientists using the system to explore their microscopy data, we conclude that spatial interfaces that use interaction props already at hand in a scientists workspace can dramatically change scientists' workflows by facilitating controlled, real-time interaction with their data.

Use of emerging visualization technologies such as depth sensing cameras and low-cost 3D displays has the potential to bring these advantages directly to a scientists desktop, which is a change from the current norm of prohibitively expensive and/or complex to maintain visualization labs. This approach is particularly imperative to facilitate wider adoption of these expressive spatial interfaces by the scientific community.

7.1.3 Controllable Filtering Using Context and Force-Feedback

Force Brushes was motivated by a need for more precise control over selection and other interactive data querying operations required by 3D visualizations. We introduce the concept of haptic brushes for querying a weather dataset comprised of streamlines integrated through a vector field of hurricane Isabel wind speed.

Control is improved for selection operations through haptic constraints that keep the pen anchored to the selected streamline. Similar haptic constraints lock the possible motion of the pen to a perpendicular plane for a pulling operation that grows the selection to include additional streamlines with similar data characteristics.

We conclude that by combining haptic force-feedback constraints with data-driven context, users are able to overcome many of the limitations of selecting complex 3D data like streamlines curving through space. Using a sequence of brushing operations

with different data variables, users can progressively refine the selection and explore the dataset with a great amount of precision. Much of this precision comes from the higher degrees of freedom that scientists have when sweeping a pen through 3D space to select their 3D data.

7.1.4 Context-Based Artistic 3D Modeling

Modeling with Context builds on the data-driven context approach used for Force Brushes, but applies it to artistic 3D modeling. We present a way of integrating 2D inspirational materials into the virtual reality modeling environment. Control for creating 3D curves, called rails, is increased by using the context of nearby contours in the reference imagery. Sweep operations are introduced to intelligently create 3D mesh surfaces based on both the user's input and the context of connecting rails.

The modeling results of the exploratory user study show that even novice users with a limited amount of training (10-25 minutes) are able to create 3D models using the system. Our other modeling results show that users had the control and expressiveness to recreate intricate physical sculptures, such as those by Pablo Gargallo, as digital models. Additionally, we show how the system can be applied for modeling new creative works based on a user's sketches or exiting 2D works of art.

We conclude that spatial interfaces using data-driven context can significant increase an artist's ability to work with computational tools in a controlled way while maintaining the expressiveness he or she has with physical media. Even without haptic support, it is clearly evident in the artwork produced by the tool that artists with a limited amount of prior 3D modeling experience can capture elaborate 3D forms with the system.

7.2 Future Work

During the development of the spatial interfaces discussed in the previous chapters, several opportunities for future work became apparent. We discuss several of the most important ones here.

7.2.1 Approaches for Tracking Spatial Input

Spatial input has traditionally been limited by the ability to track a user's body motions with high accuracy and low latency. Conventional approaches commonly use electromagnetic, ultrasonic, or optical tracking systems to measure the position and orientation of a small number of 3D locations, commonly defined by reflective markers or sensor devices. Over time, these systems have become fairly robust, hence their use in several of the interfaces described in this dissertation.

However, because these systems are only able to track a limited number of points, their use for increasingly complex and expressive spatial input is limited. Consider, for example, the way that you interact with real physical objects like this dissertation in your hands. You support its weight with multiple points of contact, and turn the pages with multiple fingers. This high-dimensional interaction is what gives you the control and expressiveness we have when handling physical objects.

One approach to supporting this more complex style of interaction is through the use of depth-sensing camera technology to track entire surfaces rather than a few points. This technique was explored in Chapter 4, but new advances in both hardware and tracking algorithms must still be developed. In particular, we recommend three areas of future spatial tracking research identified through the development of the lightweight prop interface in Chapter 4: (1) Algorithms for adaptive tracking of arbitrary objects,

(2) Approaches for low-cost desktop-scale tracking, and (3) Real-time tracking techniques.

One of the primary motivations for the interface described in Chapter 4 is to enable scientists to pickup objects in their environment like a pen or pencil and use them as impromptu interaction props. To realize this vision, new algorithms for adaptive tracking of arbitrary objects must be created. Specifically, this may include creating new unsupervised learning algorithms to automatically recognize an object held in a user's hands and start tracking it.

Similarly, approaches for low-cost small-scale desktop tracking must be developed. A variety of approaches are typically used to recognize objects in point clouds, but many rely on matching known 2D features (e.g. SIFT [141]) or 3D features (e.g. Viewpoint Feature Histograms [142], Signature of Histograms of Orientations Descriptors [143]) to points in the cloud. These feature-based descriptors typically rely on consistent normals, curvature, or spacing between points. These approaches failed when applied to track an object like a pencil with low-cost commodity hardware because of a large amount of noise in the tracking data relative to the small size of the prop.

Finally, many spatial tracking approaches based on point cloud processing do not run in real-time. For example, sample consensus based techniques, like RANSAC model fitting, are frequently used with noisy data. However, we found that fitting a cylindrical model to points in the cloud using an implementation from the Point Cloud Library [144] caused a noticeable latency for the user that is unacceptable for expressive input.

If these three tracking research areas are addressed and robust libraries and toolkits are created, we expect that they will contribute greatly to new expressive spatial interaction techniques for visualization, art, and other fields.

7.2.2 High-Dimensional Active Haptics

Similar to the development of new tracking algorithms, active force-feedback devices must advance as well. Much of this dissertation explores the use of passive haptics because we have found that it provides control advantages over free-hand spatial interaction and is less constrained than existing active haptic force-feedback devices.

Existing active haptic devices are able to render precise forces, but the feedback is usually limited to a single point. This limitation constrains the possible application of these devices for expressive spatial interfaces. For example, consider the SensAble Phantom device used in the filtering interface in Chapter 5. With this active haptic device, the user holds a pen-shaped stylus. The force-feedback is applied to the stylus tip. This stylus interface works well for controlled brushing over lines (as in our interface) or drawing precise curves through space (e.g. Drawing-on-Air [18]), but it is less practical for many other spatial tasks.

Consider the modeling interface in Chapter 6. In the future, one way that an artist might want to edit the form of an existing surface is by reaching out and shaping it with multiple fingers, similar to modeling with real physical clay. This type of expressive spatial interaction will only be possible if new active haptic devices are developed that can provide precise force feedback over multiple points of contact, potentially the entire surface of a user's hand or body.

Previous work has recently explored this area. For example, using grids of vibrotactile actuators [145, 146] or oscillating electrical fields [147] to create tactile sensations for the user. In particular, development of high-dimensional force-feedback devices that do not require the user to wear any physical instrumentation will be important for enhancing users' freedom of expression. AIREAL [148] is a device in this style, which uses vortices of air to create forces the user can feel. While these examples are promising,

advances must still be made to expand the descriptiveness and precision of the feedback.

7.2.3 Learn-Ability and Self-Revealing Spatial Interfaces

One of the primary issues with expressive spatial interfaces, particularly for new users, is that they can be difficult to learn. This might seem like a paradox when spatial interfaces are also commonly called "natural" user interfaces. People make the assumption that because they are natural there is always a clear expectation for how the user should proceed. Unfortunately, beyond simple manipulation tasks this is frequently not the case.

For example, our collaborators using the microscopy visualization interface presented in Chapter 4 were conflicted at times because the interface was so different than how they traditionally use computers. Occasionally, they needed to be reminded of specific gestures. Over time, as tracking technology matures and spatial interfaces become more common for scientific and artistic applications, this will likely change. Users will start to learn common interaction metaphors in the same way that most people have learned 2D multi-touch metaphors like pinch to zoom. However, in the meantime, new approaches must be developed to teach spatial gestures or make them more self-revealing. GestureBar [106] is a notable example in this style applied to learning 2D gestures. Similarly, ShadowGuides [78] uses virtual shadows to indicate to users how to complete multi-touch gestures.

In the future, we plan to explore how similar techniques might translate to learning 3D gestures. For example, virtual shadows might work well for learning free-hand interfaces in large virtual environments like the CAVE used in Chapter 6, but a 3D version of GestureBar might be more appropriate for fish-tank style VR where gestures tend to have smaller movements and menu-based systems are more common. Another possible approach, particularly for paper-based props like the one used in Chapter 4, is

to print instructions directly on the prop. These instructions would be always available, showing users how to place their fingers and the specific gestures to perform.

7.2.4 The Effect of Expressive Spatial Interfaces on Creativity

The focus of this dissertation is on improving the control and expressiveness of spatial interfaces; however, during the user evaluation of the 3D modeling interface, users brought up how the expressive spatial interface made them feel more creative.

Previous work in the psychology and cognitive science domains (e.g. [149, 150]) has found that fluid gestures and actions can enhance creativity, but to our knowledge little work has been done examining how spatial interfaces coupled with virtual reality affect creativity.

This is an important future research direction for expressive spatial interfaces. A greater understanding of how they affect creative thinking would enhance our ability to design effective creativity support tools. The outcomes of this would be beneficial to many domains besides art. For example, enhanced creativity might lead to more insights found when visualizing scientific data.

7.3 Conclusions

The work presented in this dissertation explores how spatial input can be made more expressive and controlled through a variety of approaches in two application domains.

For artistic domains like 3D modeling, we conclude that the greater physicality and directness of free-hand input is imperative for expressive spatial interfaces. However, this free-hand input must be controllable to achieve expressive results. Control can be provided through haptic constraints, as in the passive haptic surface used for Anchored Gesture modeling in Chapter 3, but we have found that contextual interpretation of the

input offers a similar level of control with fewer constraints for the artist's movement. Additionally, the integration of 2D and 3D content is a key feature of effective spatial interfaces in this domain. Artists frequently work with 2D media, such as paper and pencil sketches, during the ideation phase of design, and an expressive spatial interface must be able to integrate these materials into the virtual reality environment.

For scientific visualization applications, expressiveness of the input is important for quickly specifying complex spatial operations. Even more importantly, this spatial input must have precise control. In this scientific domain, we found that a combination of haptic constraints and data-driven context provides maximum control. While active force-feedback devices provide this high level of precision, their expense and limited concurrency (usually only tracking a single pen) makes them less applicable to more complex visualization tasks. For these tasks, we recommend approaches like Anchored Gestures that take full advantage of both the enhanced expressiveness of spatial gestures and the increased degrees of freedom from multiple hands and multiple concurrent touch points. However, interface designers must remain vigilant to the increased learning curve caused by the higher degrees of freedom.

Overall, the potential impact of this work is great. The techniques presented here provide more effective ways for spatially interacting in a controlled and expressive way. They have the potential to dramatically change how people approach and use computational tools when working with three-dimensional data or content. Although the focus of this work is on supporting 3D artistic creation and visualization, we believe the results are generalizable to other domains. We anticipate that the work proposed here can have a powerful impact on shaping the future of expressive human-computer interaction for the next-generation of computational tools.

References

[1] B Shneiderman, G Fischer, M Czerwinski, M Resnick, and B Myers. Creativity support tools: Report from a U.S. national science foundation sponsored workshop. *International Journal Of Human-Computer Interaction*, 20(2):61–77, 2006.

[2] Elizabeth C. Patterson. *John Dalton and the Atomic Theory: The Biography of a Natural Philosopher*. Doubleday, New York, 1970.

[3] J.J. Thomson. On the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure. *Philosophical Magazine*, 7(39):237–265, March 1904.

[4] Mi Jeong Kim and Mary Lou Maher. The impact of tangible user interfaces on spatial cognition during collaborative design. *Design Studies*, 29(3):222–253, 2008.

[5] V.K. Viswanathan and J.S. Linsey. Enhancing student innovation: Physical models in the idea generation process. In *IEEE Frontiers in Education Conference*, pages 1–6, Oct 2009.

[6] Sharon Lynn Chu, Francis Quek, Luke Gusukuma, and Joshua Tanenbaum. The effects of physicality on the child's imagination. In *Proceedings of the 9th ACM*

Conference on Creativity and Cognition, pages 93–102, New York, NY, USA, 2013. ACM.

[7] Colin Ware and Glenn Franck. Evaluating stereo and motion cues for visualizing information nets in three dimensions. *ACM Transactions on Graphics*, 15(2):121–139, 1996.

[8] 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 Symposium on Interactive 3D Graphics*, pages 35–ff., New York, NY, USA, 1997. ACM.

[9] 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, New York, NY, USA, 2002. ACM.

[10] Ji-Young Oh and Wolfgang Stuerzlinger. Laser pointers as collaborative pointing devices. In *Proceedings of Graphics Interface*, pages 141–149, 2002.

[11] Richard A. Bolt. Put-that-there: Voice and gesture at the graphics interface. *SIGGRAPH Computer Graphics*, 14(3):262–270, July 1980.

[12] Jason S Sobel, Andrew S Forsberg, David H Laidlaw, Robert C Zeleznik, Daniel F Keefe, Igor Pivkin, George E Karniadakis, Peter Richardson, and Sharon Swartz. Particle flurries: synoptic 3D pulsatile flow visualization. *IEEE Computer Graphics and Applications*, 24(2):76–85, 2004.

[13] David Anderson, James L. Frankel, Joe Marks, Aseem Agarwala, Paul Beardsley, Jessica Hodgins, Darren Leigh, Kathy Ryall, Eddie Sullivan, and Jonathan S.

Yedidia. Tangible interaction + graphical interpretation: A new approach to 3D modeling. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 393–402, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[14] Andrew Miller, Yn White, Emiko Charbonneau, Zach Kanzler, and Joseph J. Laviola. Interactive 3D model acquisition and tracking of building block structures. *IEEE Transactions on Visualization and Computer Graphics*, (4):651–659, April 2012.

[15] Ivan E. Sutherland. A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, AFIPS '68 (Fall, part I), pages 757–764, New York, NY, USA, 1968. ACM.

[16] James H. Clark. Designing surfaces in 3-d. *Communications of the ACM*, 19(8):454–460, August 1976.

[17] Dane Coffey, Nicholas Malbraaten, Trung Le, Iman Borazjani, Fotis Sotiropoulos, Arthur G. Erdman, and Daniel F. Keefe. Interactive slice WIM: Navigating and interrogating volume datasets using a multi-surface, multi-touch VR interface. *IEEE Transactions on Visualization and Computer Graphics*, 18(10):1614–1626, 2012.

[18] Daniel F. Keefe, Robert C. Zeleznik, and David H. Laidlaw. Drawing on air: Input techniques for controlled 3D line illustration. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):1067–1081, Sept 2007.

[19] Helen Perkunder, Johann Habakuk Israel, and Marc Alexa. Shape modeling with sketched feature lines in immersive 3d environments. In *Proceedings of the Seventh*

Sketch-Based Interfaces and Modeling Symposium, SBIM '10, pages 127–134, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[20] Jeff Butterworth, Andrew Davidson, Stephen Hench, and Marc. T. Olano. 3DM: A three dimensional modeler using a head-mounted display. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, I3D '92, pages 135–138, New York, NY, USA, 1992. ACM.

[21] Steven Schkolne, Michael Pruett, and Peter Schröder. Surface drawing: Creating organic 3D shapes with the hand and tangible tools. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, pages 261–268, New York, NY, USA, 2001. ACM.

[22] Daniel F. Keefe, Daniel Acevedo Feliz, Tomer Moscovich, David H. Laidlaw, and Joseph J. LaViola Jr. CavePainting: A fully immersive 3D artistic medium and interactive experience. In *Proceedings of I3D 2001*, pages 85–93, 2001.

[23] Gerold Wesche and Hans-Peter Seidel. Freedrawer: A free-form sketching system on the responsive workbench. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '01, pages 167–174, New York, NY, USA, 2001. ACM.

[24] Daniel F. Keefe. Free-form VR interactions in scientific visualization. IEEE Visualization 2008 Workshop on Scientific Workflow with Immersive Interfaces for Visualization, October 2008.

[25] Steve Bryson and Creon Levit. The virtual windtunnel: An environment for the exploration of three-dimensional unsteady flows. In *Proceedings of the 2nd Conference on Visualization '91*, VIS '91, pages 17–24, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.

[26] Jason S. Sobel, Andrew S. Forsberg, David H. Laidlaw, Robert C. Zeleznik, Daniel F. Keefe, Igor Pivkin, George E. Karniadakis, Peter Richardson, and Sharon Swartz. Particle flurries: Synoptic 3D pulsatile flow visualization. *IEEE Computer Graphics Applications*, 24(2):76–85, March 2004.

[27] H. Benko, E.W. Ishak, and S. Feiner. Cross-dimensional gestural interaction techniques for hybrid immersive environments. In *Proceedings of IEEE Virtual Reality, 2005*, pages 209–216, 2005.

[28] Otmar Hilliges, Shahram Izadi, Andrew D. Wilson, Steve Hodges, Armando Garcia-Mendoza, and Andreas Butz. Interactions in the air: Adding further depth to interactive tabletops. In *Proceedings of the 22nd annual ACM Symposium on User Interface Software and Technology*, pages 139–148, 2009.

[29] Sriram Subramanian, Dzimitry Aliakseyeu, and Andrés Lucero. Multi-layer interaction for digital tables. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 269–272, 2006.

[30] Bruno R. De Araùjo, Géry Casiez, and Joaquim A. Jorge. Mockup builder: Direct 3D modeling on and above the surface in a continuous interaction space. In *Proceedings of Graphics Interface 2012*, GI '12, pages 173–180, Toronto, Ont., Canada, Canada, 2012. Canadian Information Processing Society.

[31] Raghavendra S. Kattinakere, Tovi Grossman, and Sriram Subramanian. Modeling steering within above-the-surface interaction layers. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 317–326, 2007.

[32] N. Marquardt, R. Jota, S. Greenberg, and J. Jorge. The continuous interaction space: Interaction techniques unifying touch and gesture on and above a digital surface. In *Proceedings of the 13th IFIP TCI3 Conference on Human Computer*

Interaction - INTERACT 2011, page 16 pages, Lisbon, Portugal, September 5-9 2011. Earlier version with different author order as Report 2011-993-05 (January, 2011).

[33] M. Katzourin, D. Ignatoff, L. Quirk, J.J. LaViola, and O.C. Jenkins. Swordplay: Innovating game development through VR. *IEEE Computer Graphics and Applications*, 26(6):15–19, Nov 2006.

[34] 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*, CHI '12, pages 2421–2430, New York, NY, USA, 2012. ACM.

[35] Daniel F. Keefe, Daniel Acevedo, Jadrian Miles, Fritz Drury, Sharon M. Swartz, and David H. Laidlaw. Scientific sketching for collaborative VR visualization design. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):835–847, July 2008.

[36] Stephen Correia Wenjin Zhou and David H. Laidlaw. Haptics-Assisted 3D lasso drawing for tracts-of-interest selection in DTI visualization. In *IEEE Visualization Poster Compendium*, 2008.

[37] Daniel F. Keefe, Robert C. Zeleznik, and David H. Laidlaw. Tech-note: Dynamic dragging for input of 3D trajectories. In *Proceedings of IEEE Symposium on 3D User Interfaces 2008*, pages 51–54, 2008.

[38] David Akers. CINCH: a cooperatively designed marking interface for 3D pathway selection. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 33–42, Montreux, Switzerland, 2006.

[39] E. Sachs, A. Roberts, and D. Stoops. 3-draw: A tool for designing 3D shapes. *IEEE Computer Graphics and Applications*, 11(6):18–26, 1991.

[40] Richard Stoakley, Matthew J. Conway, and Randy Pausch. Virtual reality on a WIM: interactive worlds in miniature. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 265–272, Denver, Colorado, United States, 1995. ACM Press/Addison-Wesley Publishing Co.

[41] M. Hachet and P. Guitton. The interaction table: A new input device designed for interaction in immersive large display environments. In *Proceedings of the Workshop on Virtual Environments 2002*, EGVE '02, pages 189–196, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[42] Hyunyoung Song, François Guimbretière, Chang Hu, and Hod Lipson. Modelcraft: Capturing freehand annotations and edits on physical 3D models. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, pages 13–22, 2006.

[43] Hyunyoung Song, François Guimbretière, and Hod Lipson. The modelcraft framework: Capturing freehand annotations and edits to facilitate the 3D model design process using a digital pen. *ACM Trans. Comput.-Hum. Interact.*, 16(3):14:1–14:33, September 2009.

[44] Alexandre Gillet, Michel Sanner, Daniel Stoffler, David Goodsell, and Arthur Olson. Augmented reality with tangible auto-fabricated models for molecular biology applications. In *Proceedings of the conference on Visualization '04*, VIS '04, pages 235–242, Washington, DC, USA, 2004. IEEE Computer Society.

[45] Wolf dieter Rase. Visualization of three-dimensional gis objects using rapid prototyping technology. In *MIPRO, 2010 Proceedings of the 33rd International Convention*, 2010.

[46] Yvonne Jansen, Pierre Dragicevic, and Jean-Daniel Fekete. Evaluating the Efficiency of Physical Visualizations. In *Proceedings of the 2013 Annual Conference on Human Factors in Computing Systems (CHI 2013)*, pages 2593–2602, Paris, France, 2013. ACM, ACM.

[47] Bret Jackson and Daniel F. Keefe. Sketching over props: Understanding and interpreting 3D sketch input relative to rapid prototype props. IUI 2011 Sketch Recognition Workshop, 2011.

[48] Krzysztof Kruszyski and Robert van Liere. Tangible props for scientific visualization: concept, requirements, application. *IEEE Virtual Reality*, 13:235–244, 2009.

[49] George W. Fitzmaurice, Hiroshi Ishii, and William A. S. Buxton. Bricks: laying the foundations for graspable user interfaces. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 442–449, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[50] Hiroshi Ishii and Brygg Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 234–241, New York, NY, USA, 1997. ACM.

[51] Colin Ware and Jeff Rose. Rotating virtual objects with real handles. *ACM Trans. Comput.-Hum. Interact.*, 6(2):162–180, 1999.

[52] J.C. Goble, K. Hinckley, R. Pausch, J.W. Snell, and N.F. Kassell. Two-handed spatial interface tools for neurosurgical planning. *Computer*, 28(7):20 –26, 1995.

[53] Bernd Fröhlich and John Plate. The cubic mouse: a new device for three-dimensional input. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 526–531, New York, NY, USA, 2000. ACM.

[54] Ravin Balakrishnan, George Fitzmaurice, Gordon Kurtenbach, and Karan Singh. Exploring interactive curve and surface manipulation using a bend and twist sensitive input strip. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, I3D '99, pages 111–118, New York, NY, USA, 1999. ACM.

[55] Jia Sheng, Ravin Balakrishnan, and Karan Singh. An interface for virtual 3D sculpting via physical proxy. In *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*, GRAPHITE '06, pages 213–220, New York, NY, USA, 2006. ACM.

[56] J. Konieczny, C. Shimizu, G. Meyer, and D. Colucci. A handheld flexible display system. In *Proceedings of IEEE Visualization*, pages 591–597, 2005.

[57] Tijmen Klein, Florimond Guéniat, Luc Pastur, Frédéric Vernier, and Tobias Isenberg. A design study of direct-touch interaction for exploratory 3D scientific visualization. *Computer Graphics Forum*, 31(3pt3):1225–1234, 2012.

[58] Jean-Baptiste de la Rivière, Cédric Kervégant, Emmanuel Orvain, and Nicolas Dittlo. CubTile: a multi-touch cubic interface. In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology*, pages 69–72, 2008.

[59] Tovi Grossman, Daniel Wigdor, and Ravin Balakrishnan. Multi-finger gestural interaction with 3D volumetric displays. *ACM Transactions on Graphics*, 24:931–931, 2005.

[60] Russell M. Taylor, Warren Robinett, Vernon L. Chi, Frederick P. Brooks, Jr., William V. Wright, R. Stanley Williams, and Erik J. Snyder. The nanomanipulator: a virtual-reality interface for a scanning tunneling microscope. In *Proceedings of the 20th annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 127–134, 1993.

[61] Ricardo S. Avila and Lisa M. Sobierajski. A haptic interaction method for volume visualization. In *Proceedings of the 7th conference on Visualization '96*, VIS '96, pages 197–ff., 1996.

[62] T. van Reimersdahl, F. Bley, T. Kuhlen, and C. H. Bischof. Haptic rendering techniques for the interactive exploration of CFD datasets in virtual environments. In *Proceedings of the Workshop on Virtual Environments 2003*, EGVE '03, pages 241–246, 2003.

[63] M. Ikits, J.D. Brederson, C.D. Hansen, and C.R. Johnson. A constraint-based technique for haptic volume exploration. In *IEEE Visualization*, pages 263 –269, 2003.

[64] Karljohan E. Lundin, Mattias Sillen, Matthew D. Cooper, and Anders Ynnerman. Haptic visualization of computational fluid dynamics data using reactive forces. In *Proceedings of the Conference on Visualization and Data Analysis, part of IS&T/SPIE Symposium on Electronic Imaging*, volume 5669, 2005.

[65] Bob Ménélas, Mehdi Ammi, and Patrick Bourdot. A flexible method for haptic rendering of isosurface from volumetric data. In *Proceedings of the 6th Iinternational Conference on Haptics: Perception, Devices and Scenarios*, EuroHaptics '08, pages 687–693, 2008.

[66] Michel Abdul-Massih, Bedřich Beneš, Tong Zhang, Christopher Platzer, William Leavenworth, Huilong Zhuo, Edwin R. García, and Zhiwen Liang. Augmenting heteronanostructure visualization with haptic feedback. In *Proceedings of the 7th International Conference on Advances in Visual Computing - Volume Part II*, pages 627–636, 2011.

[67] S. Anderson S. Snibbe and B. Verplanki. Springs and constraints for 3D drawing. Mass. Inst. of Technology Artificial Intelligence Laboratory. Technical Report 1643, 1998.

[68] Yannick Thiel, Karan Singh, and Ravin Balakrishnan. Elasticurves: Exploiting stroke dynamics and inertia for the real-time neatening of sketched 2D curves. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, pages 383–392, New York, NY, USA, 2011. ACM.

[69] Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. ILoveSketch: As-natural-as-possible sketching system for creating 3D curve models. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*, pages 151–160, New York, NY, USA, 2008. ACM.

[70] David Schroeder, Dane Coffey, and Daniel F. Keefe. Drawing with the flow: A sketch-based interface for illustrative visualization of 2D vector fields. In *Proceedings of ACM SIGGRAPH/Eurographics Sketch-Based Interfaces and Modeling 2010*, pages 49–56, 2010.

[71] Steve Tsang, Ravin Balakrishnan, Karan Singh, and Abhishek Ranjan. A suggestive interface for image guided 3D sketching. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 591–598, New York, NY, USA, 2004. ACM.

[72] Bret Jackson, David Schroeder, and Daniel F. Keefe. Nailing down multi-touch: Anchored above the surface interaction for 3D modeling and navigation. In *Proceedings of Graphics Interface*, pages 181–184. Canadian Information Processing Society, 2012.

[73] Mike Wu and Ravin Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *Proceedings of the 16th annual ACM Symposium on User Interface Software and Technology*, pages 193–202, 2003.

[74] Xiang Cao, Andrew D. Wilson, Ravin Balakrishnan, Ken Hinckley, and Scott E. Hudson. ShapeTouch: Leveraging contact shape on interactive surfaces. In *3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems. TABLETOP 2008*, pages 129–36, 2008.

[75] Feng Wang and Xiangshi Ren. Empirical evaluation for finger input properties in multi-touch interaction. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 1063–1072, 2009.

[76] Yoshiki Takeoka, Takashi Miyaki, and Jun Rekimoto. Z-touch: an infrastructure for 3D gesture interaction in the proximity of tabletop surfaces. In *ACM International Conference on Interactive Tabletops and Surfaces*, pages 91–94, 2010.

[77] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. User-defined gestures for surface computing. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 1083–1092, 2009.

[78] Dustin Freeman, Hrvoje Benko, Meredith Ringel Morris, and Daniel Wigdor. ShadowGuides: visualizations for in-situ learning of multi-touch and whole-hand

gestures. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 165–172, 2009.

[79] Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. OmniTouch: wearable multitouch interaction everywhere. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 441–450, 2011.

[80] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. *ACM Trans. Graph.*, 24:1134–1141, 2005.

[81] Brookshire D. Conner, Scott S. Snibbe, Kenneth P. Herndon, Daniel C. Robbins, Robert C. Zeleznik, and Andries van Dam. Three-dimensional widgets. In *Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 183–188, 1992.

[82] 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 2011 annual conference on Human factors in computing systems*, pages 1581–1590, 2011.

[83] Frank Steinicke, Klaus H. Hinrichs, Johannes Schöning, and Antonio Krüger. Multi-touching 3D data: Towards direct interaction in stereoscopic display environments coupled with mobile devices. In *Advanced Visual Interfaces (AVI) Workshop on Designing Multi-Touch Interaction Techniques for Coupled Public and Private Displays*, pages 46–49, 2008.

[84] Dimitar Valkov, Frank Steinicke, Gerd Bruder, and Klaus H. Hinrichs. 2D touching of 3D stereoscopic objects. In *ACM Proceedings of CHI 2011 Conference on Human Factors in Computing Systems*, 2011.

[85] Lingyun Yu, P. Svetachov, P. Isenberg, M.H. Everts, and T. Isenberg. FI3D: Direct-touch interaction for the exploration of 3D scientific visualization spaces.

IEEE Transations on Visualization and Computer Graphics, 16(6):1613–1622, 2010.

[86] H. Benko and S. Feiner. Balloon selection: A multi-finger technique for accurate low-fatigue 3D selection. In *IEEE Symposium on 3D User Interfaces (3DUI) 2007*, 2007.

[87] 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, 1998.

[88] A. Martinet, G. Casiez, and L. Grisoni. Integrality and separability of multitouch interaction techniques in 3D manipulation tasks. *IEEE Transactions on Visualization and Computer Graphics*, PP(99):1, 2011.

[89] Bret Jackson, Tung Yuen Lau, David Schroeder, Kimani C. Toussaint Jr., and Daniel F. Keefe. A lightweight tangible 3D interface for interactive visualization of thin fiber structures. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2802–2809, December 2013.

[90] Mayandi Sivaguru, Sushmitha Durgam, Raghu Ambekar, David Luedtke, Glenn Fried, Allison Stewart, and Kimani C. Toussaint. Quantitative analysis of collagen fiber organization in injured tendons using fourier transform-second harmonic generation imaging. *Optics Express*, 18(24):24983–24993, 2010.

[91] A. Miller, B. White, E. Charbonneau, Z. Kanzler, and J.J. LaViola. Interactive 3D model acquisition and tracking of building block structures. *IEEE Transactions on Visualization and Computer Graphics*, 18(4):651–659, 2012.

[92] V. Interrante and C. Grosch. Strategies for effectively visualizing 3D flow with volume LIC. In *Proceedings of the 8th conference on Visualization*, pages 421–424. IEEE Computer Society Press, 1997.

[93] Andreas Wenger, Daniel F. Keefe, Song Zhang, and David H. Laidlaw. Interactive volume rendering of thin thread structures within multivalued scientific datasets. *IEEE Transactions of Visualization and Computer Graphics*, 10(6):664–672, 2004.

[94] Maarten H. Everts, Henk Bekker, Jos B.T.M. Roerdink, and Tobias Isenberg. Depth-dependent halos: Illustrative rendering of dense line data. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1299–1306, 2009.

[95] Paul Campagnola, Molly A. Brewer, Visar Ajeti, Patricia Keely, Kevin Eliceiri, Manish Patankar, and Karissa Tilbury. SHG imaging of cancer. In *Biomedical Optics*, page BSu4B.1. Optical Society of America, 2012.

[96] Christoph Spuhler, Matthias Harders, and Gabor Székely. Fast and robust extraction of centerlines in 3D tubular structures using a scattered-snakelet approach. In *Proceedings of SPIE Medical Imaging*, volume 6144, pages 1295–1302, 2006.

[97] Slawomir Krucinski, Izabella Krucinska, Srinivasan Veeravanallur, and Krzysztof Slot. Computer-assisted analysis of the extracellular matrix of connective tissue. In *Proceedings of SPIE Medical Imaging*, volume 3034, pages 950–963, 1997.

[98] Jun Wu, Bartlomiej Rajwa, David L. Filmer, Christoph M. Hoffmann, Bo Yuan, Ching-Shoei Chiang, Jennie Sturgis, and J. Paul Robinson. Analysis of orientations of collagen fibers by novel fiber-tracking software. *Microscopy and Microanalysis*, 9(6):574–580, 2003.

[99] Mark R. Mine, Frederick P. Brooks, Jr., and Carlo H. Sequin. Moving objects in space: exploiting proprioception in virtual-environment interaction. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 19–26. ACM, 1997.

[100] Abigail J. Sellen, Gordon P. Kurtenbach, and William A. S. Buxton. The prevention of mode errors through sensory feedback. *Human-Computer Interaction*, 7(2):141–164, June 1992.

[101] Daniel Herrera, Juho Kannala, and Janne Heikkilä. Joint depth and color camera calibration with distortion correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(10):2058–2064, 2012.

[102] Nicolas Burrus. Kinect calibration. `http://burrus.name/index.php/Research/KinectCalibration`, June 2013.

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

[104] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 1€ filter: a simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2527–2530. ACM, 2012.

[105] S. Prohaska and H.-C. Hege. Fast visualization of plane-like structures in voxel data. In *Proceedings of IEEEVisualization*, pages 29–36, 2002.

[106] Andrew Bragdon, Robert Zeleznik, Brian Williamson, Timothy Miller, and Joseph J. LaViola, Jr. Gesturebar: improving the approachability of gesture-based interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2269–2278. ACM, 2009.

[107] Bret Jackson, Dane Coffey, and Daniel F. Keefe. Force Brushes: Progressive Data-Driven Haptic Selection and Filtering for Multi-Variate Flow Visualizations. In *Proceedings of EuroVis 2012*, pages 7–11, Vienna, Austria, 2012. Eurographics Association.

[108] Wenjin Zhou, Peter G. Sibley, Song Zhang, David Tate, and David H. Laidlaw. Perceptual coloring and 2D sketching for segmentation of neural pathways. Siggraph 2006, Poster Compendium, 2006.

[109] A. Sherbondy, D. Akers, R. Mackenzie, R. Dougherty, and B. Wandell. Exploring connectivity of the brain's white matter with dynamic queries. *Visualization and Computer Graphics, IEEE Transactions on*, 11(4):419–430, 2005.

[110] Daniel F. Keefe, Robert C. Zeleznik, and David H. Laidlaw. Tech-note: Dynamic dragging for input of 3D trajectories. In *Proceedings of IEEE Symposium on 3D User Interfaces 2008*, pages 51–54, 2008.

[111] Helmut Doleisch, Martin Gasser, and Helwig Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *Proceedings of the symposium on Data visualisation 2003*, VISSYM '03, pages 239–248, 2003.

[112] Bernd Hentschel, Marc Wolter, Peter Renze, Wolfgang Schrder, Christian H. Bischof, and Kuhlen Torsten. Hybrid parallelization for multi-view visualization of time-dependent simulation data. In *Proceedings of EGPGV'2009*, pages 79–86, 2009.

[113] I. Corouge, S. Gouttard, and G. Gerig. Towards a shape model of white matter fiber bundles using diffusion tensor MRI. In *Biomedical Imaging: Nano to Macro, 2004. IEEE International Symposium on*, pages 344 – 347, 2004.

[114] Anders Brun, Hae-Jeong Park, Hans Knutsson, and Carl-Fredrik Westin. Coloring of DT-MRI fiber traces using laplacian eigenmaps. In *Computer Aided Systems Theory - EUROCAST 2003*, volume 2809, pages 518–529, 2003.

[115] Zhaohua Ding, John C. Gore, and Adam W. Anderson. Case study: reconstruction, visualization and quantification of neuronal fiber pathways. In *Proceedings of the conference on Visualization '01*, VIS '01, pages 453–456, 2001.

[116] Cagatay Demiralp and David H. Laidlaw. Similarity coloring of DTI fiber tracts. In *Proceedings of DMFC Workshop at MICCAI*, 2009.

[117] Björn Bollensdorff, Uwe Hahne, and Marc Alexa. The effect of perspective projection in multi-touch 3D interaction. In *Proceedings of Graphics Interface 2012*, GI '12, pages 165–172, Toronto, Ont., Canada, Canada, 2012. Canadian Information Processing Society.

[118] Archives Pablo Gargallo. Pablo gargallo, petite bacchante couchée, cuivre, 1929, 2014. Public Domain via Wikimedia Commons.

[119] Rafael Ordonez Fernández. *Catálogo Museo Pablo Gargallo*. Zaragoza: Ayuntamiento de Zaragoza, 2004.

[120] Manfred Lau, Greg Saul, Jun Mitani, and Takeo Igarashi. Modeling-in-context: user design of complementary objects with a single photo. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, SBIM '10, pages 17–24, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[121] Yong Jae Lee, C. Lawrence Zitnick, and Michael F. Cohen. Shadowdraw: Real-time user guidance for freehand drawing. *ACM Transactions on Graphics*, 30(4):27:1–27:10, 2011.

[122] Patrick Paczkowski, Min H. Kim, Yann Morvan, Julie Dorsey, Holly Rushmeier, and Carol O'Sullivan. Insitu: sketching architectural designs in context. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, SA '11, pages 182:1–182:10, New York, NY, USA, 2011. ACM.

[123] Karan Singh and Eugene Fiume. Wires: A geometric deformation technique. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 405–414, New York, NY, USA, 1998. ACM.

[124] Ran Gal, Olga Sorkine, Niloy J. Mitra, and Daniel Cohen-Or. iWIRES: An analyze-and-edit approach to shape manipulation. *ACM Transactions on Graphics*, 28(3):33:1–33:10, July 2009.

[125] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. FiberMesh: Designing freeform surfaces with 3D curves. *ACM Transactions on Graphics*, 26(3), July 2007.

[126] Ryan Schmidt, Azam Khan, Karan Singh, and Gord Kurtenbach. Analytic drawing of 3D scaffolds. *ACM Transactions on Graphics*, 28(5):149:1–149:10, December 2009.

[127] G. Humphrey. The psychology of the gestalt. *Journal of Educational Psychology*, 15(7):401–412, Oct. 1924.

[128] rni Kristjánsson and Peter Ulric Tse. Curvature discontinuities are cues for rapid shape analysis. *Perception and Psychophysics*, 63(3):390–403, 2001.

[129] James T. Todd. The visual perception of 3D shape. *Trends in Cognitive Sciences*, 8(3):115–121, March 2004.

[130] Marc K. Albert and Peter U. Tse. The role of surface attraction in perceiving volumetric shape.

[131] Peter U. Tse. A contour propagation approach to surface filling-in and volume formation. *Psychology Review*, 109(1):91–115, Jan 2002.

[132] E. Catmull and R. Rom. A class of local interpolating splines. *Computer Aided Geometric Design*, pages 317–326, 1974.

[133] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[134] Autodesk. Maya. `http://www.autodesk.com/products/autodesk-maya/`, 2014.

[135] Robert McNeel and Associates. Rhinoceros. `http://www.rhino3d.com/`, 2014.

[136] Phyllis W. Berman, Joseph G. Cunningham, and John Harkulich. Construction of the horizontal, vertical, and oblique by young children: Failure to find the oblique effect. *Child Development*, 45(2):474–478, 1974.

[137] Phyllis W. Berman. Young children's use of the frame of reference in construction of the horizontal, vertical, and oblique. *Child Development*, 47(1):259–263, 1976.

[138] Pierre Courthion. *Gargalio Sculptures et Dessins*. Albert Skira, Paris, France, 1937.

[139] Pablo Gargallo. Greta garbo con sombrero, 10 cartones recortados. `http://www.zaragoza.es/ciudad/museos/es/gargallo/obras/foto_Gargallo?id=107`, June 2014.

[140] Pablo Picasso. Bull plate IX. `http://www.wikiart.org/en/pablo-picasso/bull-plate-ix-1946`, June 2014.

[141] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999.

[142] R.B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, pages 2155–2162, Taipei, Taiwan, 2010.

[143] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Proceedings of the 11th European Conference on Computer Vision*, ECCV '10, pages 356–369, Heraklion, Greece, 2010. Springer-Verlag.

[144] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[145] Ali Israr, Seung-Chan Kim, Jan Stec, and Ivan Poupyrev. Surround haptics: Tactile feedback for immersive gaming experiences. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, pages 1087–1090, New York, NY, USA, 2012. ACM.

[146] Ali Israr and Ivan Poupyrev. Tactile brush: Drawing on skin with a tactile grid display. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2019–2028, New York, NY, USA, 2011. ACM.

[147] Olivier Bau, Ivan Poupyrev, Mathieu Le Goc, Laureline Galliot, and Matthew Glisson. Revel: Tactile feedback technology for augmented reality. In *ACM*

SIGGRAPH 2012 Emerging Technologies, SIGGRAPH '12, pages 17:1–17:1, New York, NY, USA, 2012. ACM.

[148] Rajinder Sodhi, Ivan Poupyrev, Matthew Glisson, and Ali Israr. Aireal: Interactive tactile experiences in free air. *ACM Trans. Graph.*, 32(4):134:1–134:10, July 2013.

[149] Michael L Slepian and Nalini Ambady. Fluid movement and creativity. *Journal of Experimental Psychology: General*, 141:625–629, Nov 2012.

[150] Susan Goldin-Meadow and Sian L. Beilock. Actions influence on thought: The case of gesture. *Perspectives on Psychological Science*, 5(6):664–674, 2010.

[151] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart. The cave:audio visual experience automatic virtual environment. *Communications of the ACM*, 35(6):64–72, 1992.

[152] K. Reda, A. Febretti, A. Knoll, J. Aurisano, J. Leigh, A. Johnson, M.E. Papka, and M. Hereld. Visualizing large, heterogeneous data in hybrid-reality environments. *IEEE Computer Graphics and Applications*, 33(4):38–48, July 2013.

[153] Thomas A. DeFanti, Gregory Dawe, Daniel J. Sandin, Jurgen P. Schulze, Peter Otto, Javier Girado, Falko Kuester, Larry Smarr, and Ramesh Rao. The StarCAVE, a third-generation CAVE and virtual reality optiportal. *Future Generation Computer Systems*, 25(2):169–178, 2009.

[154] A. Kenyon, J. van Rosendale, S. Fulcomer, and D. Laidlaw. The design of a retinal resolution fully immersive VR display. In *IEEE Virtual Reality*, pages 89–90, March 2014.

[155] Russell M. Taylor, II, Thomas C. Hudson, Adam Seeger, Hans Weber, Jeffrey Juliano, and Aron T. Helser. VRPN: A device-independent, network-transparent VR peripheral system. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '01, pages 55–61, New York, NY, USA, 2001. ACM.

[156] Trimble. Sketchup pro 2014. `http://www.sketchup.com/`, 2014.

[157] Makerbot Industries. Makerbot replicator 2x. `http://www.makerbot.com/`, 2014.

Appendix A

Supporting Virtual Reality Hardware and Software Development

The artistic modeling interface described in Chapter 6 uses a virtual reality CAVE environment. This appendix describes the design of a low-cost CAVE, supporting input devices, and a software toolkit developed to support these types of virtual reality applications.

A.1 Development of a Low-Cost CAVE Virtual Reality Environment

The modeling hardware environment is a four wall VR CAVE (cave automatic virtual environment) [151]. Originally developed at the University of Illinois, CAVE's provide an immersive experience to the user. In the last decade, several research institutions

Figure A.1: A low-cost virtual reality CAVE built in the Interactive Visualization Lab.

have implemented changes to the original cave. These changes include using tiled LCD displays instead of projectors [152] for higher resolution and contrast, or additional angled display surfaces to minimize ghosting [153] of the stereo rendering. Most recently, Brown University has created a new CAVE environment with the goal of reaching retinal resolution and a large field of view while minimizing seams in the projection surface that distract from the user's immersion [154].

While these updates are impressive, they are extremely expensive, costing over one million dollars to build [153]. As such, they are frequently built in large spaces and used by multiple invested groups. In contrast, our approach, shown in Figure A.1, was to design and build a low-cost alternative directly in our lab, made possible by the recent availability of low-cost 3D projectors.

Figure A.2: Two projectors mounted in the ceiling are used to project on the floor of the cave.

A.1.1 Projectors

Our CAVE implementation uses three rear-projection side-walls and a front-projection floor. Each side wall has two Dell s300 short-throw projectors for a total resolution of 1024x1024 per wall running at 120 hz. The floor uses two Dell 1610hd projectors mounted horizontally with mirrors to bounce the light to the floor, see Figure A.2.

Because the projected images contain a slight barrel distortion from the lens, the images must be warped and blended. This is accomplished using the NVIDIA Warping and Blending API. Custom software was written to iteratively adjust the projection shape to compensate for the distortion.

A.1.2 Frame and Screen Design

Shown in Figure A.1, the frame is made from 80/20 extruded aluminum T-slot profiles that bolt together using modular connections. The frame contains mounting points for

each projector and also supports the screen. The screen material consists of standard rear-projection screen that was hand grommeted. Bungee ropes, threaded through the grommets, stretch it tight around the frame.

A.1.3 Input Devices: Rapid Prototyping 3D styluses

As discussed throughout this dissertation, the physical interface is extremely important for increased control of spatial interaction. Frequently, interaction in virtual reality environments uses game controllers or wand devices; however, the ergonomic layout of these types of input devices is not as appropriate for the modeling interface presented in Chapter 4.

We were unable to find a commercial input device that fit our requirements. Thus, we created two identical rapid prototypes that fit our needs. Shown in Figure A.3, the styluses resemble large markers. This shape makes it easy to hold the stylus in the artist's hand, and he or she is able to easily manipulate it with his or her fingers.

Electronics and Software

The electronic components of each stylus consist of a Wixel micro-controller, two buttons, a lithium polymer battery, and a power switch, shown in Figure A.4. The Wixel was chosen as a microcontroller because it contains an integrated RF radio to communicate button presses to the host computer. These interactions are communicated to the VR software through the VRPN [155] library. The Wixel is also fairly small which reduced the necessary width of the stylus design.

In order to maintain the light weight necessary to prevent fatigue when drawing, a rechargeable 110mAh lithium polymer battery was used. To maximize the low capacity of this battery, the microcontroller software was written to use a low power mode of the Wixel. In the default state, the stylus software turns off the voltage regulator and

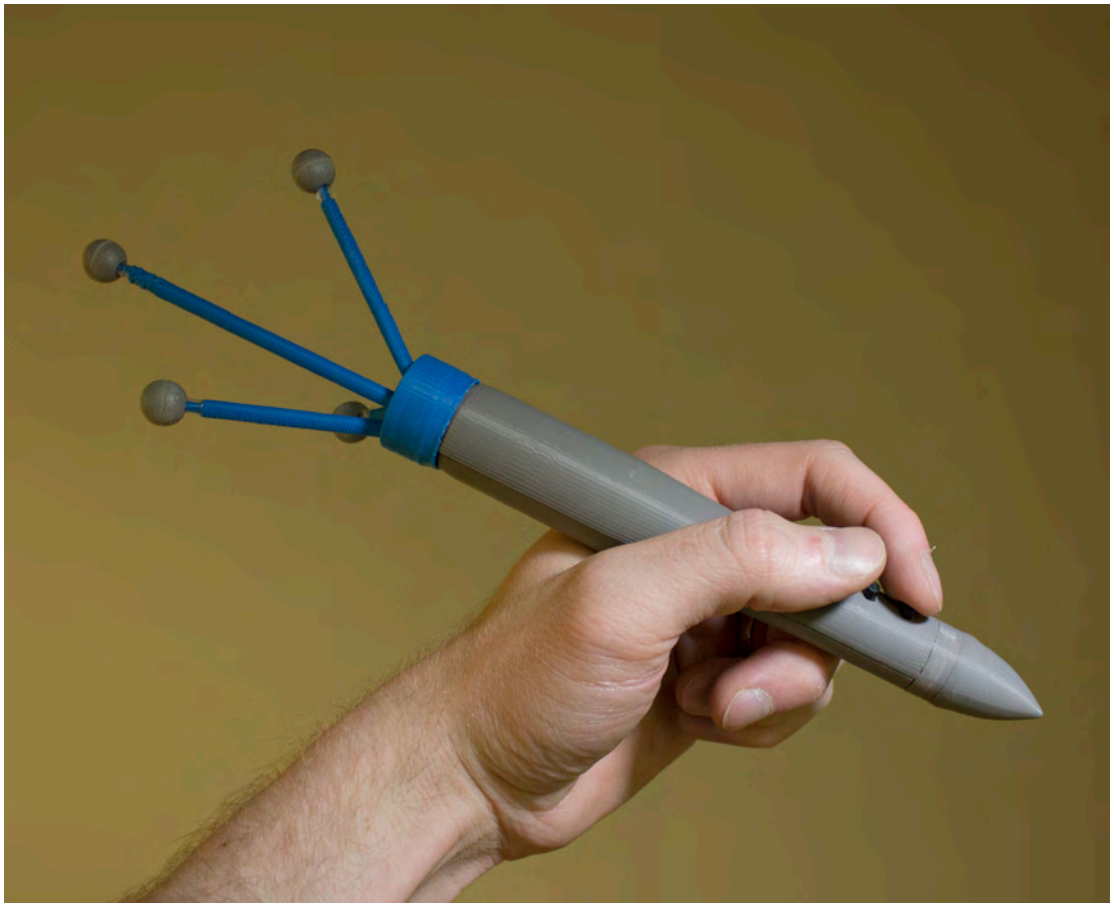

Figure A.3: 3D printed, rapid prototype, interaction stylus. Shaped like a pen, this light-weight stylus enables a more natural 3D drawing technique. Reflective markers are integrated into the rear of the stylus for 3D tracking.

crystal oscillator in the micro-controller. When the user presses one of the buttons, an interrupt command is sent to wake up the processor and send a button down event over the RF radio to the host computer. When the button is released, a button up event is sent and the Wixel returns to its low power mode. Using this software, we have found the battery will last several days with constant usage, and over a month with intermittent use.

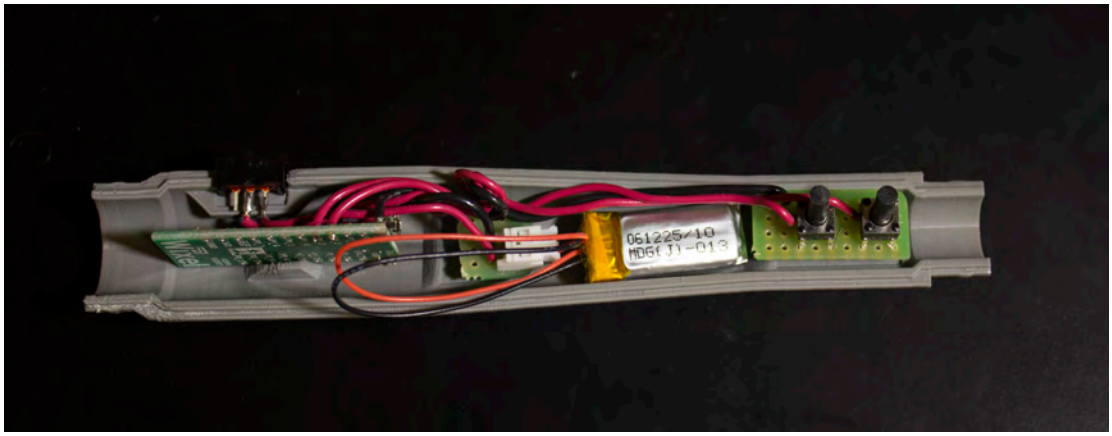

Figure A.4: Electronic components of 3D interaction stylus.

Case Design

The stylus case was modeled using Trimble Sketchup [156] and 3D printed on a Makerbot [157]. Shown in Figure A.5, the case consists of four parts: two halves of the barrel, a tracking end cap, and a tip. The ends were designed to connect together using an annular snap joint similar to a pen cap. This joint consists of a small bead running around the outside of the barrel ends and a corresponding groove in the interior of the end pieces, see detail in Figure A.6. The snap joint holds the ends securely, but makes it easy to take apart the stylus to change the battery. Additionally, the replaceable endcaps enable interchangeable tracking antler configurations for the two different styluses.

A.2 Development of a Virtual Reality Toolkit

To make developing virtual reality applications easier, many virtual reality toolkits have been developed, such as VRJuggler, CalVR, and FreeVR. Predominately, these software libraries handle display and input device configuration. They abstract the graphics pipeline to support head-tracked stereoscopic rendering.

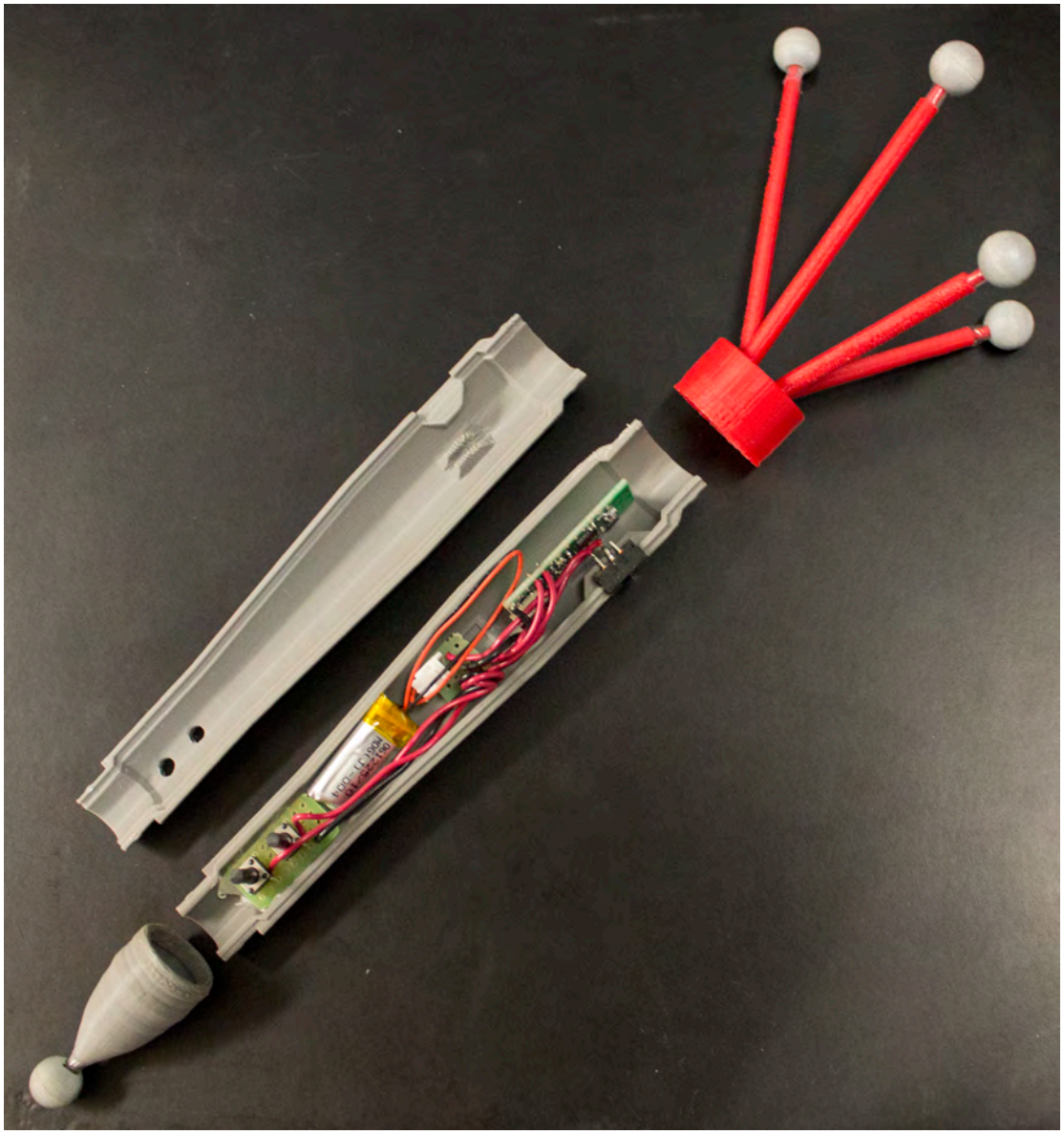

Figure A.5: The stylus case is made from four 3D printed parts. Two center barrels hold the electronic components. The endcaps snap on and include the stylus tip and a tracking marker configuration for 3D tracking.

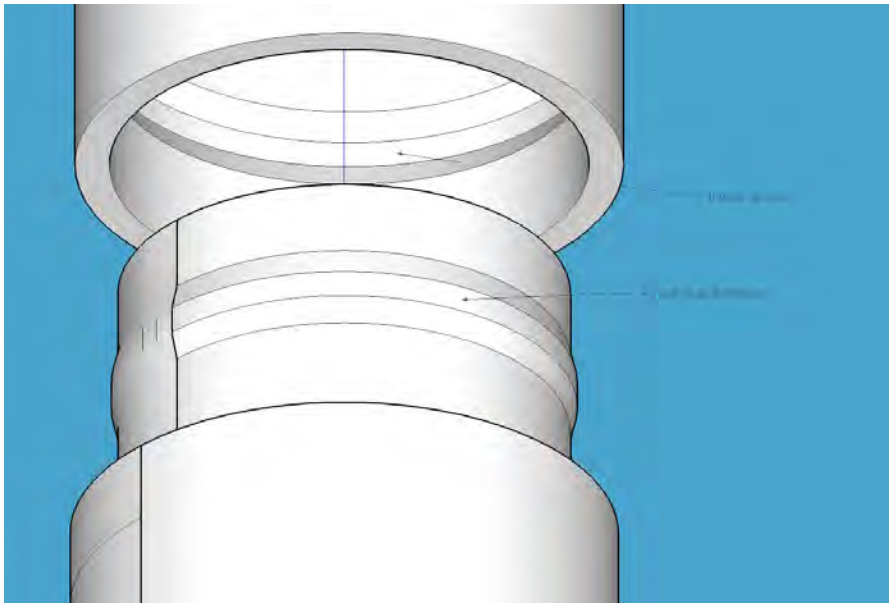

Figure A.6: Detail of annular joint connecting stylus endcaps.

With the completion of our low-cost CAVE environment, we needed a toolkit for developing VR applications. We found that no existing library met all of our needs. Therefore, we decided to develop our own open source library called MinVR. This required a fair amount of 'reinventing the wheel'. However, from our experience we learned several lessons to help guide future VR researchers. The first lesson is that developing a VR toolkit was not too difficult. Additionally, the advantages we gained by customizing it exactly to our hardware setup, as well as the familiarity gained through development, made the time invested worthwhile.

A.2.1 MinVR: Inspiration and Design Philosophy

Development of MinVR primarily grew out of the dissatisfaction we had with our existing VR toolkit, VRG3D. MinVR addresses three VR toolkit properties that we found to have a big impact in how efficiently and effectively we are able to develop VR software

in our lab. The big three are: (1) minimum-as-possible library, (2) support for the latest hardware configurations, and (3) flexible user input. In the following sections, we discuss these ideas in more detail, and explain how MinVR attempts to address them.

Minimum-as-possible library

VR toolkits should be as minimum-as-possible. Their primary purpose is to act as middleware to interface with input devices and setup the rendering environment for different screen configurations. Yet, most VR toolkits also directly integrate graphics toolkits as well. For example, VRG3D is dependent on the G3D graphics library, MiddleVR is tied to Unity, and CalVR makes use of OpenSceneGraph.

In the environment of academic research labs, the particular requirements for an individual project might necessitate using a different graphics toolkit. For example, one project might use OpenSceneGraph to manage large virtual environments, while another project might use Unity to create a VR environment with the iPhone. In our lab, we have found that a VR toolkits dependency on a particular graphics toolkit leads to hacked solutions where multiple toolkits are cobbled together in an ad hoc way to meet project needs. This slows development, leads to more bugs, and makes the code harder to understand.

Through the development of MinVR, we have attempted to fix this problem by having a core library that handles traditional VR toolkit tasks and separate AppKits that allow a developer to plugin the graphics toolkit of his/her choice.

Support for the Latest Hardware and Configurations

In the last five years, the number of display outputs on graphics cards has increased, the number of processor cores in a CPU has more than doubled, and it is now quite reasonable to put four graphics cards in a single machine. We have found that these

hardware advances enable us to run our CAVE from a single machine rather than the more common clustered setup.

This single machine configuration has significant advantages making it easier to write and debug VR software. When debugging, a developer can set breakpoints and step through code in the entire application rather than just on a single cluster node. However, many VR toolkits are not developed with this situation in mind. In order to efficiently use new hardware, modern VR toolkits must support multi-threaded rendering to make use of multi-cored architecture and GPU affinity to support multi-GPU systems.

Flexible User Input

Although most VR toolkits support a variety of input devices through VRPN, trackd, or other input libraries, they frequently make assumptions about the type of input which limits flexibility. For example, many VR toolkits assume the primary interaction is through a traditional wand device with a single orientation and position. This is frequently not the case in academic environments, particularly when researching 3D user interfaces. For instance, with the recent commoditization of low-cost depth cameras, it is now quite reasonable to assume that the input might be a point cloud. VR toolkits must be flexible to the ever evolving types of input and limit the assumptions they make.

In MinVR, we have tried to support this flexibility by defining events based on a name string and associated data. In addition to being flexible in the type of input this scheme supports, it also makes it very easy to prototype, simulate, or debug interactions. Because input is based on strings it is easy for developers to alias input devices to different names, or simulate 3D input with a mouse or keyboard.